Пермский филиал федерального государственного автономного образовательного учреждения высшего

 “Национальный исследовательский университет

“Высшая школа экономики”

*Факультет экономики, менеджмента и бизнес-информатики*

Дмитриев Арсений Алексеевич

Лабораторная работа №7. «Разработка Windows-приложения»

*Отчет по практической работе*

студента образовательной программы «Программная инженерия»

по направлению подготовки 09.03.04 Программная инженерия

руководитель

к. тех. н., доцент каф.

инф. техн. в биз.

Викентьева О.Л.

Пермь, 2020 г.

Постановка задачи

Общая

1. Разработать Windows – приложение, реализующее работу с одномерными, двумерными и рваными массивами.
2. Реализовать сохранение массивов в файле и загрузку ранее сохраненного массива из файла.

Вариант №7

|  |  |  |
| --- | --- | --- |
| Удалить все четные элементы | Добавить К строк в конец матрицы | Удалить первую строку, в которой встречаются нули |

Анализ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Функция | Входные данные | Классы входных данных | Выходные данные |
| Основные функции | | | | |
| 1 | Формирование одномерного массива | Размер массива | Целое число >=1 | Одномерный массив |
| Целое число <=0 | Сообщение об ошибке |
| 1.1 | Формирование одномерного массива с помощью ДСЧ | Вещественное число |
| Нечисло |
| 1.2 | Формирование одномерного массива с клавиатуры | Элементы массива | Целое число | Одномерный массив |
| Вещественное число | Сообщение об ошибке |
| Нечисло |
| 2 | Удаление четных элементов из одномерного массива | * Массив * Размер массива | Непустой массив, есть и четные, и нечетные | Измененный массив |
|
|
| Непустой массив, только нечетные элементы | Массив без изменений |
|
|
| Непустой массив, только четные элементы | Пустой массив |
|
|
| Пустой массив | Сообщение об ошибке |
| 3 | Формирование двумерного массива | Количество строк в массиве | Целое число >=1 | Двумерный массив |
| Целое число <=0 | Сообщение об ошибке |
| Формирование двумерного массива с помощью ДСЧ | Вещественное число |
| Нечисло |
| 3.1 | Количество столбцов в массиве | Целое число >=1 | Двумерный массив |
| Целое число <=0 | Сообщение об ошибке |
| Вещественное число |
| Нечисло |
| 3.2 | Формирование двумерного массива с клавиатуры | Элементы массива | Целое число | Двумерный массив |
| Вещественное число | Сообщение об ошибке |
| Нечисло |
| 4 | Добавление К строк в конец матрицы | К - число добавляемых строк | Целое число >=1 | Измененный массив |
| Целое число <=0 | Сообщение об ошибке |
| Вещественное число |
| Нечисло |
| 5 | Формирование рваного массива | Количество строк в массиве | Целое число >=1 | Рваный массив |
| Целое число <=0 | Сообщение об ошибке |
| Формирование рваного массива с помощью ДСЧ | Вещественное число |
| Нечисло |
| 5.1 | Количество элементов в каждой строке | Целое число >=1 | Рваный массив |
| Целое число <=0 | Сообщение об ошибке |
| Вещественное число |
| Нечисло |
| 5.2 | Формирование рваного массива с клавиатуры | Элементы массива | Целое число | Рваный массив |
| Вещественное число | Сообщение об ошибке |
| Нечисло |
| 6 | Удаление первой строки, содержащей хотя бы один ноль | Массив | Непустой массив, содержащий нули | Измененный массив |
|
| Непустой массив, в котором нет элементов равных 0 | Массив без изменений |
|
| Пустой массив | Сообщение об ошибке |
|
|
|
|
|
|

Проектирование форм

1. MainForm

Основная форма программы. На ней организованы меню и вывод для пользователя

|  |  |  |
| --- | --- | --- |
| Элемент управления | Функция | Ограничения |
| readButton | Вызов диалогового окна для ввода массива текущего типа | - |
| createButton | Вызов диалогового окна для ввода размеров для рандомной генерации массива текущего типа | - |
| activityButton | Для одномерного и рваного: вызов действия над массивом согласно задаче Для матрицы - вызов диалогового окна для ввода доп. информации для совершения действия | - |
| arrayOutTextBox | Выполнение печати массива для пользователя | ReadOnly |
| arraysComboBox | Выбор пользователем текущего (рабочего) типа массива | Выбор из 3-х типов |
| readLabel | Подпись для кнопки чтения массива | - |
| createLabel | Подпись для кнопки создания случайного массива | - |
| activityLabel | Подпись для кнопки действия над массивом | - |
| choiseLabel | Подпись для списка выбора типа массива | - |
| outLabel | Подпись для вывода массива | - |
| filesMenuStrip | Вызов выпадающего меню для работы с файлами | - |
| saveFileOption | Вызов окна для сохранения массива в файл | - |
| loadFileOption | Вызов окна для загрузки массива из файла | - |
| helpFileOption | Вызов окна со справкой | - |

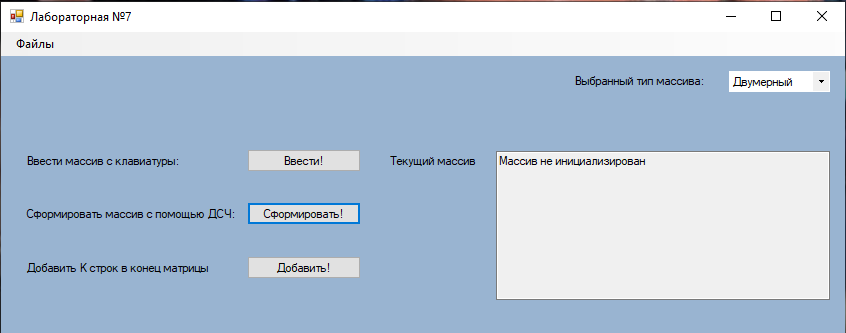


Рисунок 1 – MainForm

1. DlgArray Form

Форма для работы с одномерным массивом.

|  |  |  |
| --- | --- | --- |
| Элемент управления | Функция | Ограничения |
| lengthTextBox | Поле для ввода размера массива | Разрешен ввод только цифр, ограничение на размер не более 20 |
| elementsTextBox | Поле для ввода элементов массива (видно только при запуске для ручного ввода массива) | Разрешен ввод цифр, пробелов, знаков "минус". Ограничение на элементы - не больше 100 по модулю; на количество - введенный размер в поле lengthTextBox |
| lengthLabel | Подпись для поля ввода размера |  |
| elementsLabel | Подпись для поля ввода элементов (виден только при запуске для ручного ввода массива) |  |
| okButton | Запускает валидацию полей ввода. В случае успеха - возвращает введенные данные в основную форму | Прохождение валидации |
| cancelButton | Отмена ввода. Возврат в форму DialogResult.Cancel | - |

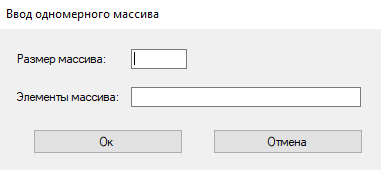


Рисунок 2.1 - DlgArray – Для ввода массива

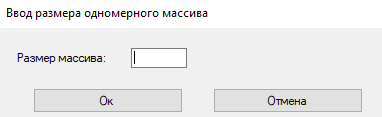


Рисунок 2.2 – DlgArray – Для формирования случайного массива заданного размера

1. DlgMatrix Form

|  |  |  |
| --- | --- | --- |
| Элемент управления | Функция | Ограничения |
| rowsTextBox | Поле для ввода количества строк в матрице | Разрешен ввод только цифр, ограничение на строки: не более 20 |
| columnsTextBox | Поле для ввода количества столбцов | Разрешен ввод только цифр, ограничение на строки: не более 20 |
| elementsTextBox | Поле для ввода элементов массива (видно только при запуске для ручного ввода массива и ) | Разрешен ввод цифр, пробелов и знаков "минус". Ограничение на элементы - не больше 100 по модулю; на размеры - введенные размеры |
| rowsLabel | Подпись для поля ввода размера | - |
| columnsLabel | Подпись для поля ввода столбцов | - |
| elementsLabel | Подпись для поля ввода элементов (виден только при запуске для ручного ввода массива) | - |
| okButton | Запускает валидацию полей ввода. В случае успеха - возвращает введенные данные в основную форму | Прохождение валидации |
| cancelButton | Отмена ввода. Возврат в форму DialogResult.Cancel | - |

Форма для работы с матрицей.

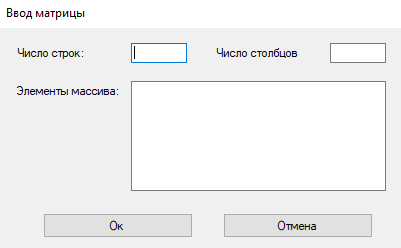
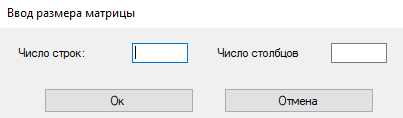


Рисунок 3.1 – DlgMatrix – Для ввода матрицы

Рисунок 3.2 – DlgMatrix – Для формирования случайной матрицы заданного размера

1. DlgJagged Form

Форма для работы с рваным массивом

|  |  |  |
| --- | --- | --- |
| Элемент управления | Функция | Ограничения |
| lengthTextBox | Поле для ввода количества строк в массиве | Разрешен ввод только цифр, ограничение на размер не более 20 |
| elementsTextBox | Поле для ввода элементов массива и размера строк | Разрешен ввод цифр, пробелов, знаков "минус". Ограничение на элементы - не больше 100 по модулю; на число строк - введенный размер в поле lengthTextBox |
| lengthLabel | Подпись для поля ввода размера | - |
| elementsLabel | Подпись для поля ввода элементов (виден только при запуске для ручного ввода массива) | - |
| okButton | Запускает валидацию полей ввода. В случае успеха - возвращает введенные данные в основную форму | Прохождение валидации |
| cancelButton | Отмена ввода. Возврат в форму DialogResult.Cancel | - |

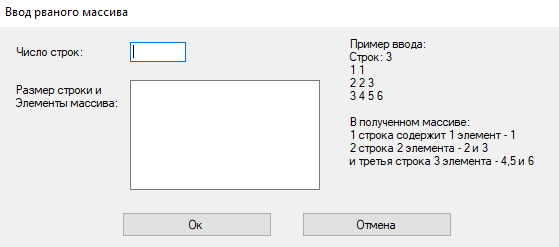


Рисунок 4.1 - DlgJagged – Ввод рваного массива

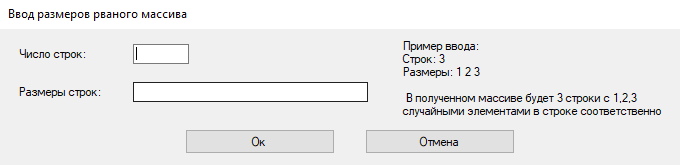


Рисунок 4.2 – DlgJagged – Формирование случайного рваного массива с заданными размерами

Код программы

1. Class Arrays – логика

using System;

namespace Lab7

{

class Arrays

{

public const int MAX\_ELEM = 100;

public const int MIN\_ELEM = -100;

public const int MAX\_SIZE = 20;

public const int MIN\_SIZE = 1;

public const string EMPTY = "Пустой массив";

public const string INIT = "Массив не инициализирован";

private int type;

private string stringFormArray;

private int[] arr;

private int[,] mtx;

private int[][] jag;

public Arrays(int typeOfArray)

{

this.type = typeOfArray;

switch (type) {

case 0:

arr = null;

break;

case 1:

mtx = null;

break;

case 2:

jag = null;

break;

}

stringFormArray = INIT;

}

override public string ToString()

{

string txt = "";

switch (type)

{

case 0:

{

if (arr.Length == 0) txt = EMPTY;

for (int i = 0; i < arr.Length - 1; i++) txt += arr[i].ToString() + " ";

txt += arr[arr.Length - 1].ToString();

break;

}

case 1:

{

int rows = mtx.GetLength(0), col = mtx.GetLength(1);

if (rows == 0) txt = EMPTY;

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

{

for (int j = 0; j < col - 1; j++) txt += mtx[i, j].ToString() + " ";

txt += mtx[i, col - 1];

if(i != rows - 1) txt += "\r\n";

}

break;

}

case 2:

{

int rows = jag.GetLength(0);

if (rows == 0) txt = EMPTY;

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

{

for (int j = 0; j < jag[i].Length - 1; j++) txt += jag[i][j].ToString() + " ";

txt += jag[i][jag[i].Length - 1].ToString();

if (i != rows - 1) txt += "\r\n";

}

break;

}

}

return txt;

}

#region IntLogic

private static int[] CreateRandomArray(int sizeOfArray)

{

int[] arr = new int[sizeOfArray];

Random rand = new Random();

for (int i = 0; i < sizeOfArray; i++) arr[i] = rand.Next(MIN\_ELEM, MAX\_ELEM);

return arr;

}

private static int[] ArrayDeleteEven(int[] arr)

{

int sizeArr = arr.Length;

int sizeNew = 0;

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

{

if (arr[i] % 2 != 0) sizeNew++;

}

int[] newArr = new int[sizeNew];

int newInd = 0;

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

{

if (arr[i] % 2 != 0)

{

newArr[newInd] = arr[i];

newInd++;

}

}

return newArr;

}

private static int[,] CreateRandomMatrix(int row, int column)

{

int[,] matr = new int[row, column];

Random rand = new Random();

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

{

for (int j = 0; j < column; j++) matr[i, j] = rand.Next(MIN\_ELEM, MAX\_ELEM);

}

return matr;

}

private static int[,] MatrixAddRows(int[,] matr, int[,] addMatr)

{

int[,] newMatr;

if (matr == null) newMatr = addMatr;

else

{

int row = matr.GetLength(0);

int col = addMatr.GetLength(1);

int k = addMatr.GetLength(0);

newMatr = new int[row + k, col];

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

{

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

{

newMatr[i, j] = matr[i, j];

}

}

for (int i = row; i < row + k; i++)

{

int t = i - row;

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

{

newMatr[i, j] = addMatr[t, j];

}

}

}

return newMatr;

}

private static int[][] CreateRandomJagged(int row, int[] sizes)

{

int[][] jag = new int[row][];

Random rand = new Random();

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

{

int size = sizes[i];

jag[i] = new int[size];

for (int j = 0; j < size; j++) jag[i][j] = rand.Next(MIN\_ELEM, MAX\_ELEM);

}

int fl = rand.Next(1, 4);

if (fl != 3)

{

int i = rand.Next(0, row);

int j = rand.Next(0, jag[i].GetLength(0));

jag[i][j] = 0;

}

return jag;

}

private static bool JaggedFirstRowWithZero(ref int[][] jag, ref int rowDel)

{

bool find = false;

int row = jag.GetLength(0);

for (int i = 0; i < row && !find; i++)

{

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

{

if (jag[i][j] == 0)

{

find = true;

rowDel = i;

}

}

}

return find;

}

private static int[][] JaggedDeleteZeroRow(int[][] jag)

{

int rowDel = -1;

int row = jag.GetLength(0);

bool find = JaggedFirstRowWithZero(ref jag, ref rowDel);

if (!find) return jag;

else

{

int[][] jag1 = new int[row - 1][];

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

{

jag1[i] = new int[jag[i].Length];

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

{

jag1[i][j] = jag[i][j];

}

}

for (int i = rowDel + 1; i < row; i++)

{

jag1[i - 1] = new int[jag[i].Length];

for (int j = 0; j < jag[i].Length; j++) jag1[i - 1][j] = jag[i][j];

}

return jag1;

}

}

#endregion

#region Define

public void Define(string txt, string txtSize, bool random)

{

if (random)

{

int row = Convert.ToInt32(txtSize);

jag = new int[row][];

string[] nums = txt.Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);

int[] size = new int[row];

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

{

size[i] = Convert.ToInt32(nums[i]);

}

jag = CreateRandomJagged(row, size);

}

else

{

int row = Convert.ToInt32(txtSize);

jag = new int[row][];

string[] str = txt.Split(new char[] { '\n' }, StringSplitOptions.RemoveEmptyEntries);

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

{

string[] nums = str[i].Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);

int size = Convert.ToInt32(nums[0]);

jag[i] = new int[size];

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

{

jag[i][j] = Convert.ToInt32(nums[j + 1]);

}

}

}

stringFormArray = this.ToString();

}

public void Define(string txt, string txtSize)

{

int size = Convert.ToInt32(txtSize);

arr = new int[size];

string[] nums = txt.Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);

for (int i = 0; i < size; i++) arr[i] = Convert.ToInt32(nums[i]);

stringFormArray = this.ToString();

}

public void Define(string txt, string txtSize1, string txtSize2)

{

int row = Convert.ToInt32(txtSize1);

int col = Convert.ToInt32(txtSize2);

mtx = new int[row, col];

string[] str = txt.Split(new char[] { '\n' }, StringSplitOptions.RemoveEmptyEntries);

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

{

string[] nums = str[i].Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);

for (int j = 0; j < col; j++) mtx[i, j] = Convert.ToInt32(nums[j]);

}

stringFormArray = this.ToString();

}

public void DefineRandom(string txtSize)

{

int size = Convert.ToInt32(txtSize);

arr = CreateRandomArray(size);

stringFormArray = this.ToString();

}

public void DefineRandom(string txtSize1, string txtSize2)

{

int row = Convert.ToInt32(txtSize1);

int col = Convert.ToInt32(txtSize2);

mtx = CreateRandomMatrix(row, col);

stringFormArray = this.ToString();

}

#endregion

#region Action

public void PerformAction()

{

if (type == 0)

{

arr = ArrayDeleteEven(arr);

if (arr.Length == 0)

{

stringFormArray = EMPTY;

arr = null;

}

else stringFormArray = this.ToString();

}

else

{

jag = JaggedDeleteZeroRow(jag);

if (jag.GetLength(0) == 0)

{

stringFormArray = EMPTY;

jag = null;

}

else stringFormArray = this.ToString();

}

}

public void PerformAction(string txt, string txtSize, string txtSize2 = "")

{

if (txt == "")

{

PerformActionMtxRandom(Convert.ToInt32(txtSize), Convert.ToInt32(txtSize2));

return;

}

string[] str = txt.Split(new char[] { '\n' }, StringSplitOptions.RemoveEmptyEntries);

int row = Convert.ToInt32(txtSize);

int col;

if (mtx == null) col = Convert.ToInt32(txtSize2);

else col = mtx.GetLength(1);

int[,] addMatr = new int[row, col];

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

{

string[] nums = str[i].Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);

for (int j = 0; j < col; j++) addMatr[i, j] = Convert.ToInt32(nums[j]);

}

mtx = MatrixAddRows(mtx, addMatr);

stringFormArray = this.ToString();

}

private void PerformActionMtxRandom(int row, int col)

{

int[,] addMtx = new int[row, col];

Random rand = new Random();

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

{

for(int j = 0; j<col; j++) addMtx[i, j] = rand.Next(MIN\_ELEM, MAX\_ELEM);

}

mtx = MatrixAddRows(mtx, addMtx);

stringFormArray = this.ToString();

}

#endregion

#region BackToForm

public string GetStringFormArray()

{

return this.stringFormArray;

}

public int GetMtxColumns()

{

if (mtx == null) return 0;

return mtx.GetLength(1);

}

public string GetLength()

{

switch (type)

{

case 0: return arr.Length.ToString();

case 1: return $"{mtx.GetLength(0)}\n{mtx.GetLength(1)}";

case 2: return jag.GetLength(0).ToString();

default: return "";

}

}

#endregion

}

}

1. Class InputHandler – проверка введенных данных

using System;

using System.Text.RegularExpressions;

namespace Lab7

{

class InputHandler

{

private static int sizeLowerBound = Arrays.MIN\_SIZE;

private static int sizeUpperBound = Arrays.MAX\_SIZE;

private static int elementLowerBound = Arrays.MIN\_ELEM;

private static int elementUpperBound = Arrays.MAX\_ELEM;

public static bool CheckSize(string txt, ref string error)

{

string pat = @"\d+";

if (Regex.IsMatch(txt, pat))

{

int num;

try

{

num = Convert.ToInt32(txt);

}

catch (OverflowException)

{

error = "В качестве размера массива введено число, неподдерживаемое по величине";

return false;

}

if (num >= sizeLowerBound && num <= sizeUpperBound) return true;

else

{

error = $"Размер массива не попадает в рабочий диапазон от {sizeLowerBound} до {sizeUpperBound}";

}

}

else

{

if (txt == String.Empty) error = "Размер не был введен";

else error = "Размер должен быть выражен целым числом";

}

return false;

}

public static bool CheckArray(string txt, string txtSize, out string error)

{

string[] nums = txt.Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);

error = "";

if (CheckSize(txtSize, ref error))

{

int size = Convert.ToInt32(txtSize);

if (nums.Length == size)

{

int okNums = 0;

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

{

int num;

try

{

num = Convert.ToInt32(nums[i]);

}

catch (FormatException)

{

error = $"Элемент {i + 1} - не целое число";

return false;

}

catch (OverflowException)

{

error = $"В качестве элемента массива {i + 1} введено число, неподдерживаемое по величине";

return false;

}

if (num >= elementLowerBound && num <= elementUpperBound) okNums++;

else

{

error = $"Элемент {i + 1} не попадает в рабочий диапазон от {elementLowerBound} до {elementUpperBound}!";

break;

}

}

if (okNums == size) return true;

}

else

{

if(size < nums.Length) error = "Введено меньше элементов, чем указано!";

else error = "Введено больше элементов, чем указано!";

}

}

return false;

}

public static bool CheckMatrix(string txt, string txtSize1, string txtSize2, out string error)

{

error = "";

bool size1Ok = CheckSize(txtSize1, ref error);

bool size2Ok = CheckSize(txtSize2, ref error);

if (size1Ok && size2Ok)

{

int row = Convert.ToInt32(txtSize1);

int size = Convert.ToInt32(txtSize2);

string[] rows = txt.Split(new char[] { '\n' }, StringSplitOptions.RemoveEmptyEntries);

if (rows.Length == row)

{

int okRows = 0;

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

{

string[] nums = rows[i].Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);

if (nums.Length == size)

{

int okNums = 0;

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

{

int num;

try

{

num = Convert.ToInt32(nums[j]);

}

catch (FormatException)

{

error = $"Элемент {j + 1} в строке {i + 1} - не целое число";

return false;

}

catch (OverflowException)

{

error = $"В качестве элемента {i + 1} в строке {i + 1} введено число, неподдерживаемое по величине";

return false;

}

if (num >= elementLowerBound && num <= elementUpperBound) okNums++;

else

{

error = $"Элемент {i + 1} в строке {i + 1} не попадает в рабочий диапазон от {elementLowerBound} до {elementUpperBound}!";

break;

}

}

if (okNums == size) okRows++;

else return false;

}

else

{

error = $"Количество введенных элементов в строке {i + 1} не соответствует указанному количеству столбцов";

return false;

}

}

if (okRows == row) return true;

}

else

{

if(row < rows.Length) error = "Введено меньше строк, чем указано";

else error = "Введено больше строк, чем указано";

}

}

else if (!size1Ok && !size2Ok)

{

string err = "";

size2Ok = CheckSize(txtSize1, ref err);

error = err + "\n\r" + error;

}

return false;

}

public static bool CheckJaggedRandom(string txt, string txtSize, out string error)

{

error = "";

if (CheckSize(txtSize, ref error))

{

string[] nums = txt.Trim().Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);

int size = Convert.ToInt32(txtSize);

if (nums.Length == size)

{

bool ok = true;

for (int i = 0; i < size && ok; i++) ok = CheckSize(nums[i], ref error);

return ok;

}

else

{

error = "Введенное количество размеров строк не соответствует указанному";

if (nums.Length > size) error += " (больше)";

else error += " (меньше)";

}

}

return false;

}

public static bool CheckJagged(string txt, string txtSize, out string error)

{

error = "";

if (CheckSize(txtSize, ref error))

{

int row = Convert.ToInt32(txtSize);

string[] rows = txt.Trim().Split(new char[] { '\n' }, StringSplitOptions.RemoveEmptyEntries);

if (rows.Length == row)

{

int okRows = 0;

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

{

string[] nums = rows[i].Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);

if (CheckSize(nums[0], ref error))

{

int size = Convert.ToInt32(nums[0]);

if (nums.Length - 1 == size)

{

int okNums = 0;

for (int j = 1; j < size + 1; j++)

{

int num;

try

{

num = Convert.ToInt32(nums[j]);

}

catch (FormatException)

{

error = $"Элемент {j + 1} в строке {i + 1} - не целое число";

return false;

}

catch (OverflowException)

{

error = $"В качестве элемента {i + 1} в строке {i + 1} введено число, неподдерживаемое по величине";

return false;

}

if (num >= elementLowerBound && num <= elementUpperBound) okNums++;

else

{

error = $"Элемент {i + 1} в строке {i + 1} не попадает в рабочий диапазон от {elementLowerBound} до {elementUpperBound}!";

break;

}

}

if (okNums == size) okRows++;

else return false;

}

else

{

error = $"Указанный размер строки {i + 1} не соответствует количеству введенных элементов";

return false;

}

}

else return false;

}

if (okRows == row) return true;

}

else

{

error = "Указанное количество строк не соответствует введенному количеству!";

return false;

}

}

return false;

}

}

}

1. MainForm – основное окно

using System;

using System.IO;

using System.Windows.Forms;

using System.Text.RegularExpressions;

namespace Lab7

{

public partial class MainForm : Form

{

private static string[] activities = { "Удалить четные элементы:", "Добавить К строк в конец матрицы", "Удалить первую строку с нулями" };

private static Arrays arr;

private static Arrays mtx;

private static Arrays jag;

public MainForm()

{

InitializeComponent();

readLabel.Text = "Ввести массив с клавиатуры:";

createLabel.Text = "Сформировать массив с помощью ДСЧ:";

readButton.Text = "Ввести!";

createButton.Text = "Сформировать!";

choiseLabel.Text = "Выбранный тип массива: ";

outLabel.Text = "Текущий массив";

arr = new Arrays(0);

mtx = new Arrays(1);

jag = new Arrays(2);

arraysComboBox.SelectedIndex = 0;

}

private void arraysComboBox\_SelectedIndexChanged(object sender, EventArgs e)

{

activityLabel.Text = activities[arraysComboBox.SelectedIndex];

activityButton.Text = activities[arraysComboBox.SelectedIndex].Split(' ')[0] + '!';

switch (arraysComboBox.SelectedIndex)

{

case 0:

{

arrayOutTextBox.Text = arr.GetStringFormArray();

break;

}

case 1:

{

arrayOutTextBox.Text = mtx.GetStringFormArray();

break;

}

case 2:

{

arrayOutTextBox.Text = jag.GetStringFormArray();

break;

}

}

}

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

{

switch (arraysComboBox.SelectedIndex)

{

case 0:

{

DlgArray dlg = new DlgArray(0);

dlg.ShowDialog();

if (dlg.DialogResult == DialogResult.OK)

{

arr.Define(dlg.GetStringOfElements(), dlg.GetLengthAsString());

arrayOutTextBox.Text = arr.GetStringFormArray();

}

dlg.Dispose();

break;

}

case 1:

{

DlgMatrix dlg = new DlgMatrix(0, 0);

dlg.ShowDialog();

if (dlg.DialogResult == DialogResult.OK)

{

mtx.Define(dlg.GetStringOfElements(), dlg.GetRowsAsString(), dlg.GetColumnsAsString());

arrayOutTextBox.Text = mtx.GetStringFormArray();

}

dlg.Dispose();

break;

}

case 2:

{

DlgJagged dlg = new DlgJagged(0);

dlg.ShowDialog();

if (dlg.DialogResult == DialogResult.OK)

{

jag.Define(dlg.GetStringOfElements(), dlg.GetRowsAsString(), false);

arrayOutTextBox.Text = jag.GetStringFormArray();

}

dlg.Dispose();

break;

}

}

}

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

{

switch (arraysComboBox.SelectedIndex)

{

case 0:

{

DlgArray dlg = new DlgArray(1);

dlg.ShowDialog();

if (dlg.DialogResult == DialogResult.OK)

{

arr.DefineRandom(dlg.GetLengthAsString());

arrayOutTextBox.Text = arr.GetStringFormArray();

}

dlg.Dispose();

break;

}

case 1:

{

DlgMatrix dlg = new DlgMatrix(1, 0);

dlg.ShowDialog();

if (dlg.DialogResult == DialogResult.OK)

{

mtx.DefineRandom(dlg.GetRowsAsString(), dlg.GetColumnsAsString());

arrayOutTextBox.Text = mtx.GetStringFormArray();

}

dlg.Dispose();

break;

}

case 2:

{

DlgJagged dlg = new DlgJagged(1);

dlg.ShowDialog();

if (dlg.DialogResult == DialogResult.OK)

{

jag.Define(dlg.GetStringOfElements(), dlg.GetRowsAsString(), true);

arrayOutTextBox.Text = jag.GetStringFormArray();

}

dlg.Dispose();

break;

}

}

}

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

{

switch (arraysComboBox.SelectedIndex)

{

case 0:

{

string change = arr.GetStringFormArray();

if(arrayOutTextBox.Text == Arrays.EMPTY)

{

MessageBox.Show("Массив пуст!", "Предупреждение", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

else if(arrayOutTextBox.Text == Arrays.INIT)

{

MessageBox.Show("Массив не инициализирован!", "Предупреждение", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

else

{

arr.PerformAction();

arrayOutTextBox.Text = arr.GetStringFormArray();

if (arrayOutTextBox.Text == change) MessageBox.Show("В массиве нет четных. Ничего не было удалено.", "Предупреждение", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

break;

}

case 1:

{

string change = mtx.GetStringFormArray();

DlgMatrix dlg = new DlgMatrix(2, mtx.GetMtxColumns());

dlg.ShowDialog();

if(dlg.DialogResult == DialogResult.OK)

{

mtx.PerformAction(dlg.GetStringOfElements(), dlg.GetRowsAsString(), dlg.GetColumnsAsString());

arrayOutTextBox.Text = mtx.GetStringFormArray();

}

dlg.Dispose();

break;

}

case 2:

{

string change = jag.GetStringFormArray();

if (arrayOutTextBox.Text == Arrays.EMPTY)

{

MessageBox.Show("Массив пуст!", "Предупреждение", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

else if (arrayOutTextBox.Text == Arrays.INIT)

{

MessageBox.Show("Массив не инициализирован!", "Предупреждение", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

else

{

jag.PerformAction();

arrayOutTextBox.Text = jag.GetStringFormArray();

if (arrayOutTextBox.Text == change)

{

MessageBox.Show("В массиве нет строк с 0. Ничего не было удалено.", "Предупреждение", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

break;

}

}

}

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

{

SaveFileDialog saveFileDialog = new SaveFileDialog();

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

if (saveFileDialog.ShowDialog() == DialogResult.OK)

{

Stream fileStream = saveFileDialog.OpenFile();

using (StreamWriter writer = new StreamWriter(fileStream))

{

int type = arraysComboBox.SelectedIndex;

switch (type)

{

case 0:

{

writer.WriteLine(type.ToString());

writer.WriteLine(arr.GetLength());

writer.Write(arr.GetStringFormArray());

break;

}

case 1:

{

writer.WriteLine(type.ToString());

writer.WriteLine(mtx.GetLength());

writer.Write(mtx.GetStringFormArray());

break;

}

case 2:

{

writer.WriteLine(type.ToString());

writer.WriteLine(jag.GetLength());

writer.Write(jag.GetStringFormArray());

break;

}

}

}

}

}

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

{

using (OpenFileDialog openFileDialog = new OpenFileDialog())

{

openFileDialog.InitialDirectory = "c:\\";

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

openFileDialog.FilterIndex = 1;

if (openFileDialog.ShowDialog() == DialogResult.OK)

{

string filePath = openFileDialog.FileName;

Stream fileStream = openFileDialog.OpenFile();

using (StreamReader reader = new StreamReader(fileStream))

{

string fileContent = reader.ReadToEnd();

string error = "";

try

{

string[] lines = fileContent.Split(new char[] { '\n' }, StringSplitOptions.RemoveEmptyEntries);

string tmp = "";

foreach (string line in lines)

{

if (line.Trim().Length != 0) tmp += line.Trim() + "\n";

}

lines = tmp.Split(new char[] { '\n' }, StringSplitOptions.RemoveEmptyEntries);

int type = 0;

if(Regex.IsMatch(lines[0], @"[012]"))

{

type = Convert.ToInt32(lines[0]);

}

arraysComboBox.SelectedIndex = type;

switch (type)

{

case 0:

{

if (InputHandler.CheckArray(lines[2], lines[1], out error))

{

arr.Define(lines[2], lines[1]);

arrayOutTextBox.Text = arr.GetStringFormArray();

}

break;

}

case 1:

{

if (InputHandler.CheckSize(lines[1], ref error)){

int row = Convert.ToInt32(lines[1]);

string txt = "";

for(int i = 3; i<3+row; i++)

{

txt += lines[i] + "\n";

}

if(InputHandler.CheckMatrix(txt, lines[1], lines[2], out error))

{

mtx.Define(txt, lines[1], lines[2]);

arrayOutTextBox.Text = mtx.GetStringFormArray();

}

}

break;

}

case 2:

{

if(InputHandler.CheckSize(lines[1], ref error)){

int row = Convert.ToInt32(lines[1]);

string txt = "";

for (int i = 2; i < 2 + row; i++)

{

txt += lines[i] + "\n";

}

if(InputHandler.CheckJagged(txt, lines[1], out error))

{

jag.Define(txt, lines[1], false);

arrayOutTextBox.Text = jag.GetStringFormArray();

}

}

break;

}

}

if (error != "") MessageBox.Show($"Произошла ошибка при чтении файла:\n{error}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

catch

{

MessageBox.Show("При чтении файла произошла неизвестная ошибка!", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

}

}

}

}

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

{

string message = "Перед загрузкой массива из файла убедитесь, что загрузочный файл имеет расширение txt.\nТакже проверьте что он соответсутвует форме:\n";

message += "<Номер типа (0 - одномерный, 1 - двумерный, 2 - рваный)>\n<Размеры массива (для одномерного - длина, матрицы - на отдельных строках число строк и столбцов,";

message += "рваного - число строк)>\nДалее сам массив, в такой же форме как в приложении\n\nПример одномерного:\n0\n4\n1 -3 95 13\n\nМатрицы:\n1\n2\n3\n1 54 -7\n4 16 64\n\n";

message += "Рваного:\n2\n4\n5 3 -8 25 -7 44\n1 0\n4 3 5 2 -1\n3 3 1 2\n\nВ случае ошибки проверьте, что все строки заканчиваются цифрой (то есть не пробелом) и что в файле отсутсвуют пустые строки";

MessageBox.Show(message, "Справка", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

}

1. DlgArray

using System;

using System.ComponentModel;

using System.Drawing;

using System.Text.RegularExpressions;

using System.Windows.Forms;

namespace Lab7

{

public partial class DlgArray : Form

{

private int actionId;

private ErrorProvider sizeError;

private ErrorProvider elemError;

public DlgArray(int actionId)

{

InitializeComponent();

FormBorderStyle = FormBorderStyle.FixedSingle;

this.actionId = actionId;

okButton.Text = "Ок";

cancelButton.Text = "Отмена";

// 0 - Ввод всего массива

// 1 - Ввод только размера

if (actionId == 0)

{

this.Size = new Size(400, 180);

this.Text = "Ввод одномерного массива";

elementsTextBox.Size = new Size(230, 20);

elementsTextBox.Multiline = false;

lengthLabel.Text = "Размер массива:";

elementsLabel.Text = "Элементы массива: ";

okButton.Location = new Point((this.Size.Width - 330) / 2, 100);

cancelButton.Location = new Point(okButton.Location.X + 180, 100);

}

else

{

this.Size = new Size(400, 130);

this.Text = "Ввод размера одномерного массива";

lengthLabel.Text = "Размер массива:";

okButton.Location = new Point((this.Size.Width - 330) / 2, 60);

cancelButton.Location = new Point(okButton.Location.X + 180, 60);

elementsTextBox.Hide();

elementsLabel.Hide();

}

sizeError = new ErrorProvider();

sizeError.BlinkStyle = ErrorBlinkStyle.NeverBlink;

elemError = new ErrorProvider();

elemError.BlinkStyle = ErrorBlinkStyle.NeverBlink;

}

private void lengthTextBox\_Validating(object sender, CancelEventArgs e)

{

string errorMsg = "";

if (!InputHandler.CheckSize(lengthTextBox.Text, ref errorMsg))

{

e.Cancel = true;

lengthTextBox.Select(0, lengthTextBox.Text.Length);

sizeError.SetError(lengthTextBox, errorMsg);

}

}

private void lengthTextBox\_Validated(object sender, EventArgs e)

{

sizeError.SetError(lengthTextBox, String.Empty);

}

private void elementsTextBox\_Validating(object sender, CancelEventArgs e)

{

string errorMsg;

if (!InputHandler.CheckArray(elementsTextBox.Text, lengthTextBox.Text, out errorMsg))

{

e.Cancel = true;

elementsTextBox.Select(0, elementsTextBox.Text.Length);

elemError.SetError(elementsTextBox, errorMsg);

}

}

private void elementsTextBox\_Validated(object sender, EventArgs e)

{

elemError.SetError(elementsTextBox, String.Empty);

}

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

{

if (!Regex.IsMatch(e.KeyChar.ToString(), @"\d") && e.KeyChar != 8)

{

e.Handled = true;

}

}

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

{

if (!Regex.IsMatch(e.KeyChar.ToString(), @"[\d\s\-]") && e.KeyChar != 8)

{

e.Handled = true;

}

}

public string GetLengthAsString()

{

return lengthTextBox.Text.Trim();

}

public string GetStringOfElements()

{

return elementsTextBox.Text.Trim();

}

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

{

if (this.ValidateChildren(ValidationConstraints.Visible)) this.Close();

else this.DialogResult = 0;

}

}

}

1. DlgMatrix

using System;

using System.ComponentModel;

using System.Drawing;

using System.Text.RegularExpressions;

using System.Windows.Forms;

namespace Lab7

{

public partial class DlgMatrix : Form

{

private int actionId;

private ErrorProvider elemError;

private ErrorProvider size1Error;

private ErrorProvider size2Error;

public DlgMatrix(int actionId, int columnSize)

{

InitializeComponent();

this.FormBorderStyle = FormBorderStyle.FixedSingle;

this.actionId = actionId;

okButton.Text = "Ок";

cancelButton.Text = "Отмена";

if (actionId == 0)

{

this.Size = new Size(420, 260);

this.Text = "Ввод матрицы";

elementsTextBox.Size = new Size(255, 110);

rowsLabel.Text = "Число строк:";

columnsLabel.Text = "Число столбцов";

elementsLabel.Text = "Элементы массива: ";

okButton.Location = new Point((this.Size.Width - 330) / 2, 185);

cancelButton.Location = new Point(okButton.Location.X + 180, 185);

}

else if(actionId == 1)

{

this.Size = new Size(420, 130);

this.Text = "Ввод размера матрицы";

rowsLabel.Text = "Число строк:";

columnsLabel.Text = "Число столбцов";

okButton.Location = new Point((this.Size.Width - 330) / 2, 60);

cancelButton.Location = new Point(okButton.Location.X + 180, 60);

elementsTextBox.Hide();

elementsLabel.Hide();

}

else

{

this.Size = new Size(420, 260);

this.Text = "Ввод матрицы";

elementsTextBox.Size = new Size(255, 110);

rowsLabel.Text = "Число строк:";

columnsLabel.Text = "Число столбцов";

elementsLabel.Text = "Элементы массива: ";

if (columnSize > 0)

{

columnsTextBox.Text = columnSize.ToString();

columnsTextBox.ReadOnly = true;

}

okButton.Location = new Point((this.Size.Width - 330) / 2, 185);

cancelButton.Location = new Point(okButton.Location.X + 180, 185);

}

size1Error = new ErrorProvider();

size1Error.BlinkStyle = ErrorBlinkStyle.NeverBlink;

size2Error = new ErrorProvider();

size2Error.BlinkStyle = ErrorBlinkStyle.NeverBlink;

elemError = new ErrorProvider();

elemError.BlinkStyle = ErrorBlinkStyle.NeverBlink;

}

private void elementsTextBox\_Validating(object sender, CancelEventArgs e)

{

string errorMsg;

string txt = elementsTextBox.Text;

if (!(InputHandler.CheckMatrix(elementsTextBox.Text, rowsTextBox.Text, columnsTextBox.Text, out errorMsg) ^ (actionId == 2 && txt.Trim() == "")))

{

e.Cancel = true;

elementsTextBox.Select(0, elementsTextBox.Text.Length);

elemError.SetError(elementsTextBox, errorMsg);

}

}

private void elementsTextBox\_Validated(object sender, EventArgs e)

{

elemError.SetError(elementsTextBox, String.Empty);

}

private void rowsTextBox\_Validating(object sender, CancelEventArgs e)

{

string errorMsg = "";

if (!InputHandler.CheckSize(rowsTextBox.Text, ref errorMsg))

{

e.Cancel = true;

rowsTextBox.Select(0, rowsTextBox.Text.Length);

size1Error.SetError(rowsTextBox, errorMsg);

}

}

private void rowsTextBox\_Validated(object sender, EventArgs e)

{

size1Error.SetError(rowsTextBox, String.Empty);

}

private void columnsTextBox\_Validating(object sender, CancelEventArgs e)

{

string errorMsg = "";

if (!InputHandler.CheckSize(columnsTextBox.Text, ref errorMsg))

{

e.Cancel = true;

columnsTextBox.Select(0, columnsTextBox.Text.Length);

size2Error.SetError(columnsTextBox, errorMsg);

}

}

private void columnsTextBox\_Validated(object sender, EventArgs e)

{

size2Error.SetError(columnsTextBox, String.Empty);

}

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

{

if (!Regex.IsMatch(e.KeyChar.ToString(), @"\d") && e.KeyChar != 8)

{

e.Handled = true;

}

}

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

{

if (!Regex.IsMatch(e.KeyChar.ToString(), @"[\d\s\-]") && e.KeyChar != 8)

{

e.Handled = true;

}

}

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

{

if (!Regex.IsMatch(e.KeyChar.ToString(), @"\d") && e.KeyChar != 8)

{

e.Handled = true;

}

}

public string GetStringOfElements()

{

return elementsTextBox.Text.Trim();

}

public string GetRowsAsString()

{

return rowsTextBox.Text.Trim();

}

public string GetColumnsAsString()

{

return columnsTextBox.Text.Trim();

}

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

{

if (this.ValidateChildren(ValidationConstraints.Visible)) this.Close();

else this.DialogResult = 0;

}

}

}

1. DlgJagged

using System;

using System.ComponentModel;

using System.Drawing;

using System.Text.RegularExpressions;

using System.Windows.Forms;

namespace Lab7

{

public partial class DlgJagged : Form

{

private int actionId;

private ErrorProvider sizeError;

private ErrorProvider elemError;

public DlgJagged(int actionId)

{

InitializeComponent();

this.FormBorderStyle = FormBorderStyle.FixedSingle;

this.actionId = actionId;

okButton.Text = "Ок";

cancelButton.Text = "Отмена";

if (actionId == 0)

{

this.Size = new Size(580, 260);

this.Text = "Ввод рваного массива";

elementsTextBox.Size = new Size(190, 110);

rowsLable.Text = "Число строк:";

elementsLabel.Text = "Размер строки и\n\rЭлементы массива: ";

helpLabel.Text = "Пример ввода:\nСтрок: 3\n1 1\n2 2 3\n3 4 5 6\n\nВ полученном массиве:\n1 строка содержит 1 элемент - 1\n2 строка 2 элемента - 2 и 3\nи третья строка 3 элемента - 4,5 и 6";

helpLabel.Location = new Point(350, 10);

okButton.Location = new Point((this.Size.Width - 330) / 2, 185);

cancelButton.Location = new Point(okButton.Location.X + 180, 185);

}

else

{

this.Size = new Size(700, 180);

this.Text = "Ввод размеров рваного массива";

elementsTextBox.Size = new Size(235, 20);

elementsTextBox.Multiline = false;

rowsLable.Text = "Число строк:";

elementsLabel.Text = "Размеры строк: ";

helpLabel.Text = "Пример ввода:\nСтрок: 3\nРазмеры: 1 2 3\n\n В полученном массиве будет 3 строки с 1,2,3\nслучайными элементами в строке соответственно";

helpLabel.Location = new Point(400, 10);

okButton.Location = new Point((this.Size.Width - 330) / 2, 100);

cancelButton.Location = new Point(okButton.Location.X + 180, 100);

}

sizeError = new ErrorProvider();

sizeError.BlinkStyle = ErrorBlinkStyle.NeverBlink;

elemError = new ErrorProvider();

elemError.BlinkStyle = ErrorBlinkStyle.NeverBlink;

}

private void rowsTextBox\_Validating(object sender, CancelEventArgs e)

{

string errorMsg = "";

if (!InputHandler.CheckSize(rowsTextBox.Text, ref errorMsg))

{

e.Cancel = true;

rowsTextBox.Select(0, rowsTextBox.Text.Length);

sizeError.SetError(rowsTextBox, errorMsg);

}

}

private void rowsTextBox\_Validated(object sender, EventArgs e)

{

sizeError.SetError(rowsTextBox, String.Empty);

}

private void elementsTextBox\_Validating(object sender, CancelEventArgs e)

{

string errorMsg;

bool condition = (actionId == 0) ? !InputHandler.CheckJagged(elementsTextBox.Text, rowsTextBox.Text, out errorMsg) :

!InputHandler.CheckJaggedRandom(elementsTextBox.Text, rowsTextBox.Text, out errorMsg);

if (condition)

{

e.Cancel = true;

elementsTextBox.Select(0, elementsTextBox.Text.Length);

elemError.SetError(elementsTextBox, errorMsg);

}

}

private void elementsTextBox\_Validated(object sender, EventArgs e)

{

elemError.SetError(elementsTextBox, String.Empty);

}

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

{

if (!Regex.IsMatch(e.KeyChar.ToString(), @"\d") && e.KeyChar != 8)

{

e.Handled = true;

}

}

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

{

if (!Regex.IsMatch(e.KeyChar.ToString(), @"[\d\s\-]") && e.KeyChar != 8)

{

e.Handled = true;

}

}

public string GetRowsAsString()

{

return rowsTextBox.Text.Trim();

}

public string GetStringOfElements()

{

return elementsTextBox.Text.Trim();

}

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

{

if (this.ValidateChildren(ValidationConstraints.Visible)) this.Close();

else this.DialogResult = 0;

}

}

}

Тестирование функций

1. Ввод одномерного массива

|  |  |  |  |
| --- | --- | --- | --- |
| № | Ввод | Ожидаемый результат | Результат работы |
| Т1 | 0 | Сообщение об ошибке | Сообщение об ошибке |
| Т2 | 1  115 | Сообщение об ошибке | Сообщение об ошибке |
| Т3 | 5  –1 12 --- -4- --1 | Сообщение об ошибке | Сообщение об ошибке |
| Т4 | 4  2 12 -8 13 | Массив | Массив |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Критерии\тесты | Т1 | Т2 | Т3 | Т4 |
| Критерии входных данных: размер массива | | | | |
| Целое число >=1 и <=20 |  | + | + | + |
| Целое число =0 или >20 | + |  |  |  |
| Критерии входных данных: элементы массива | | | | |
| |Целое число| <=100 |  | + | + | + |
| |Целое число| >100 |  | + |  |  |
| Нечисло (неправильные строки с '-') |  |  | + |  |
| Критерии длины набора | | | | |
| Единичный набор |  | + |  |  |
| Длина >1 |  |  | + | + |
| Критерии выходных данных | | | | |
| Массив |  |  |  | + |
| Сообщение об ошибке | + | + | + |  |

Черный ящик:

1. Формирование одномерного массива через ДСЧ

|  |  |  |  |
| --- | --- | --- | --- |
| № | Ввод | Ожидаемый результат | Результат работы |
| Т1 | 32 | Сообщение об ошибке | Сообщение об ошибке |
| Т2 | 4 | Массив из 4 элементов | Массив из 4 элементов |

Черный ящик:

|  |  |  |
| --- | --- | --- |
| Критерии\тесты | Т1 | Т2 |
| Критерии входных данных: размер массива | | |
| Целое число >=1 и <=20 |  | + |
| Целое число =0 или >20 | + |  |
| Критерии выходных данных | | |
| Массив |  | + |
| Сообщение об ошибке | + |  |

1. Удаление четных элементов

|  |  |  |  |
| --- | --- | --- | --- |
| № | Ввод | Ожидаемый результат | Результат работы |
| Т1 | 2 4 3 1 8 7 | 3 1 7 | 3 1 7 |
| Т2 | 1 3 -5 17 | 1 3 -5 17 | 1 3 -5 17 |
| Т3 | 2 8 -44 0 -6 | Пустой массив | Пустой массив |
| Т4 | Пустой массив | Сообщение об ошибке | Сообщение об ошибке |

Черный ящик:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Критерии\тесты | Т1 | Т2 | Т3 | Т4 |
| Критерии входных данных: массив | | | |  |
| Не пустой | + | + | + |  |
| Пустой |  |  |  | + |
| Критерии длины набора: четные элементы | | | |  |
| Четных =0 |  | + |  |  |
| Четных >=1 | + |  |  |  |
| Четных = размер массива |  |  | + |  |
| Критерии выходных данных | | | |  |
| Массив | + | + |  |  |
| Массив стал пустым |  |  | + |  |
| Сообщение об ошибке |  |  |  | + |

1. Ввод матрицы

|  |  |  |  |
| --- | --- | --- | --- |
| № | Ввод | Ожидаемый результат | Результат работы |
| Т1 | 0 0 | Сообщение об ошибке | Сообщение об ошибке |
| Т2 | 21 5 (Любая матрица21х5) | Сообщение об ошибке | Сообщение об ошибке |
| Т3 | 1 3  2 -101 4 | Сообщение об ошибке | Сообщение об ошибке |
| Т4 | 2 3  1 -1- 5  4 4 4-- | Сообщение об ошибке | Сообщение об ошибке |
| Т5 | 1 5  1 2 -3 -4 5 | Заданная матрица | Заданная матрица |

Черный ящик:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Критерии\тесты | Т1 | Т2 | Т3 | Т4 | Т5 |
| Критерии входных данных: число строк и столбцов | | | | | |
| Целое число >=1 и <=20 |  | + | + | + | + |
| Целое число =0 или >20 | + | + |  |  |  |
| Критерии входных данных: элементы массива | | | | | |
| |Целое число| <=100 |  |  | + | + | + |
| |Целое число| >100 |  |  | + |  |  |
| Нечисло (неправильные строки с '-') |  |  |  | + |  |
| Критерии выходных данных | | | | | |
| Массив |  |  |  |  | + |
| Сообщение об ошибке | + | + | + | + |  |

1. Формирование матрицы через ДСЧ

|  |  |  |  |
| --- | --- | --- | --- |
| № | Ввод | Ожидаемый результат | Результат работы |
| Т1 | 32 0 | Сообщение об ошибке | Сообщение об ошибке |
| Т2 | 15 31 | Сообщение об ошибке | Сообщение об ошибке |
| Т3 | 5 8 | Матрица 5х8 | Матрица 5х8 |

Черный ящик:

|  |  |  |  |
| --- | --- | --- | --- |
| Критерии\тесты | Т1 | Т2 | Т3 |
| Критерии входных данных: число строк | | | |
| Целое число >=1 и <=20 |  | + | + |
| Целое число =0 или >20 | + |  |  |
| Критерии входных данных: число столбцов | | | |
| Целое число >=1 и <=20 |  |  | + |
| Целое число =0 или >20 | + | + |  |
| Критерии выходных данных | | | |
| Массив |  |  | + |
| Сообщение об ошибке | + | + |  |

1. Добавление строк к матрице

|  |  |  |  |
| --- | --- | --- | --- |
| № | Ввод | Ожидаемый результат | Результат работы |
| Т1 | 25 (4х10)  Любая корректная матрица 25х10 | Сообщение об ошибке | Сообщение об ошибке |
| Т2 | 2 (3х5)  1 4 -1500  350 5 -79 | Сообщение об ошибке | Сообщение об ошибке |
| Т3 | 3 (9х2)  1 5  --1 -1  3 2 | Сообщение об ошибке | Сообщение об ошибке |
| Т4 | 1 (6х7)  1 -55 38 17 -19 99 1 | Измененная матрица | Измененная матрица |
| Т5 | 2 3 (Пустой массив)  -1 5 9  3 -8 -7 | Новая матрица | Новая матрица |

(AхB) – размеры матрицы, к которой добавляют строки. Матрица заполняется с помощью ДСЧ

Черный ящик:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Критерии\тесты | Т1 | Т2 | Т3 | Т4 | T5 |
| Вход: Массив | | | | | |
| Не пустой | + | + | + | + |  |
| Пустой |  |  |  |  | + |
| Вход: Число добавляемых строк | | | | | |
| Целое число >=1 и <=20 |  | + | + | + | + |
| Целое число =0 или >20 | + |  |  |  |  |
| Критерии входных данных: элементы массива | | | | | |
| |Целое число| <=100 |  | + | + | + | - |
| |Целое число| >100 |  | + |  |  | - |
| Нечисло (неправильные строки с '-') |  |  | + |  | - |
| Критерии выходных данных | | | | | |
| Новая матрица |  |  |  | + | + |
| Сообщение об ошибке | + | + | + |  |  |

1. Ввод рваного массива

|  |  |  |  |
| --- | --- | --- | --- |
| № | Ввод | Ожидаемый результат | Результат работы |
| Т1 | 0 | Сообщение об ошибке | Сообщение об ошибке |
| Т2 | 3  1 1  0  3 1 5 -3 | Сообщение об ошибке | Сообщение об ошибке |
| Т3 | 2  1 5  3 1- | Сообщение об ошибке | Сообщение об ошибке |
| Т4 | 2  4 1 100 101 -1  3 5 -212 42 | Сообщение об ошибке | Сообщение об ошибке |
| Т5 | 4  1 1  2 -2 3  3 -4 5 -6  1 100 | Заданный массив | Заданный массив |

Черный ящик:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Критерии\тесты | Т1 | Т2 | Т3 | Т4 | Т5 |
| Критерии входных данных: размер массива | | | | | |
| Целое число >=1 и <=20 |  | + | + | + | + |
| Целое число =0 или >20 | + |  |  |  |  |
| Критерии входных данных: размер строк | | | | | |
| Целое число >=1 и <=20 |  |  | + | + | + |
| Целое число =0 или >20 |  | + |  |  |  |
| Критерии входных данных: элементы массива | | | | | |
| |Целое число| <=100 |  |  | + | + | + |
| |Целое число| >100 |  |  |  | + |  |
| Нечисло (неправильные строки с '-') |  |  | + |  |  |
| Критерии выходных данных | | | | | |
| Массив |  |  |  |  | + |
| Сообщение об ошибке | + | + | + | + |  |

1. Формирование рваного массива через ДСЧ

|  |  |  |  |
| --- | --- | --- | --- |
| № | Ввод | Ожидаемый результат | Результат работы |
| Т1 | 23 | Сообщение об ошибке | Сообщение об ошибке |
| Т2 | 4  2 21 3 15 | Сообщение об ошибке | Рваный массив |
| Т3 | 3  1 5 3 | Рваный массив | Рваный массив |

Черный ящик:

|  |  |  |  |
| --- | --- | --- | --- |
| Критерии\тесты | Т1 | Т2 | Т3 |
| Критерии входных данных: размер массива | | | |
| Целое число >=1 и <=20 |  | + | + |
| Целое число =0 или >20 | + |  |  |
| Критерии входных данных: размер строк | | | |
| Целое число >=1 и <=20 |  |  | + |
| Целое число =0 или >20 |  | + |  |
| Критерии выходных данных | | | |
| Массив |  |  | + |
| Сообщение об ошибке | + | + |  |

1. Удаление первой строки с 0

|  |  |  |  |
| --- | --- | --- | --- |
| № | Ввод | Ожидаемый результат | Результат работы |
| Т1 | 3  6 1 4 -8 3 15 2  5 6 14 12 -9 -7  1 1 | Массив без изменений | Массив без изменений |
| Т2 | 3  4 2 1 0 6  3 1 7 -19  4 0 0 0 1 | 1 7 -19  0 0 0 1 | 1 7 -19  0 0 0 1 |
| Т3 | 4  2 -4 10  3 1 7 15  1 0  3 -8 12 0 | -4 10  1 7 15  -8 12 0 | -4 10  1 7 15  -8 12 0 |
| Т4 | 4  6 1 49 -15 8 10 -78  3 18 -7 1  4 14 -5 16 1  5 0 1 2 3 4 | 1 49 -15 8 10 -78  18 -7 1  14 -5 16 1 | 1 49 -15 8 10 -78  18 -7 1  14 -5 16 1 |
| Т5 | 1  6 4 8 5 1 0 -12 | Пустой массив | Пустой массив |
| Т6 | Пустой массив | Сообщение об ошибке | Сообщение об ошибке |

Черный ящик:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Критерии\тесты | Т1 | Т2 | Т3 | Т4 | Т5 | Т6 |
| Критерии входных данных: массив | | | |  |  |  |
| Не пустой | + | + | + | + | + |  |
| Пустой массив |  |  |  |  |  | + |
| Критерии длины набора: четные элементы | | | |  |  |  |
| Строк с нулями = 0 | + |  |  |  |  |  |
| Строк с нулями >= 1 |  | + | + | + |  |  |
| Строк всего = 1, Строк с нулями = 1 |  |  |  |  | + |  |
| Критерии поиска: первая строка с нулями | | | |  |  |  |
| В начале |  | + |  |  |  |  |
| В середине |  |  | + |  |  |  |
| В конце |  |  |  | + |  |  |
| Критерии выходных данных | | | |  |  |  |
| Массив без изменений | + |  |  |  |  |  |
| Измененный массив |  | + | + | + |  |  |
| Пустой массив |  |  |  |  | + |  |
| Сообщение об ошибке |  |  |  |  |  | + |

Тестирование программы

|  |  |  |  |
| --- | --- | --- | --- |
| Действие | Должно произойти | Произошло | Поле вывода |
| Сформировать одномерный массив из 5 эл. с помощью ДСЧ | Создается массив из 5 случайных элементов и выводится в текст бокс | Создался массив из 5 случайных элементов и вывелся в текст бокс | -1 22 11 -5 73 |
| Изменение текущего типа массива на рваный | Поменялись тексты:  1) в лейбле у кнопки действия  2) в текст боксе для вывода на "Массив не инициализирован" | Поменялись тексты в лейбле у кнопки действия и в текст боксе для вывода на "Массив не инициализирован" | Массив не инициализирован |
| Ввод рваного массива без строк с 0 | Ввод рваного массива без строк с 0 | Ввод рваного массива без строк с 0 | 2 3  4 -10 11  3 |
| Удаление первой строки с 0 | Сообщение об ошибке | Сообщение об ошибке | 2 3  4 -10 11  3 |
| Изменение текущего типа массива на одномерный | Поменялись тексты: 1) в лейбле у кнопки действия 2) в текст боксе для вывода на созданный ранее одномерный массив | Поменялись тексты: 1) в лейбле у кнопки действия 2) в текст боксе для вывода на созданный ранее одномерный массив | -1 22 11 -5 73 |
| Ввод одномерного массива с несколькими (но не всеми) четными элементами | Создается массив из введенных элементов и выводится в текст бокс | Создается массив из введенных элементов и выводится в текст бокс | 10 -99 17 8 45 |
| Удаление четных элементов из него | Удалились все четные элементы | Удалились все четные элементы | -99 17 45 |
| Удаление четных элементов из него | Сообщение что ничего не было удалено | Сообщение что ничего не было удалено | -99 17 45 |

***Сценарий 1***

|  |  |  |  |
| --- | --- | --- | --- |
| Действие | Должно произойти | Произошло | Поле вывода |
| Изменение текущего типа массива на матрицу | Поменялись тексты: 1) в лейбле у кнопки действия | Поменялись тексты в лейбле у кнопки действия | Массив не инициализирован |
| Сформировать матрицу 3х5 с помощью ДСЧ | Создается матрица 3х5 из случайных элементов и выводится в текст для вывода | Создается матрица 3х5 из случайных элементов и выводится в текст для вывода | 58 -20 -28 -6 45 -43 24 78 92 42 9 -75 2 -41 -21 |
| Добавить к матрице 3 строки случайных элементов | Матрица обновляется, меняется вывод на текст боксе | Матрица обновляется, меняется вывод на текст боксе | 58 -20 -28 -6 45 -43 24 78 92 42 9 -75 2 -41 -21 -57 66 64 -18 -83 43 -6 -55 -94 -80 -45 -48 29 -44 -67 |
| Сохранить матрицу в файл | В выбранном каталоге создается новый файл с текущей матрицой | В выбранном каталоге создается новый файл с текущей матрицой | -//-//- |
| Добавить еще 1 строку | Матрица обновляется, меняется вывод на текст боксе | Матрица обновляется, меняется вывод на текст боксе | 58 -20 -28 -6 45 -43 24 78 92 42 9 -75 2 -41 -21 -57 66 64 -18 -83 43 -6 -55 -94 -80 -45 -48 29 -44 -67 -34 -35 68 24 -77 |
| Загрузить ранее созданный файл с матрицей | Загружается матрица из файла, обновляется вывод пользователю | Загружается матрица из файла, обновляется вывод пользователю | 58 -20 -28 -6 45 -43 24 78 92 42 9 -75 2 -41 -21 -57 66 64 -18 -83 43 -6 -55 -94 -80 -45 -48 29 -44 -67 |

***Сценарий 2***