МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования

**«Московский технический университет связи и информатики»**

Кафедра «Информатика»

**Отчёт по лабораторным работам за 2 семестр**

**по дисциплине**

**«Алгоритмизация и программирование»**

Выполнил: студент гр. БЭИ2202 Кулешов А. С.

Проверил: доц. Шакин В. Н.

Вариант: 17

Москва, 2023 г.

**Оглавление**

[**Лабораторная работа №8 2**](#_Toc137626573)

[**Лабораторная работа №9 8**](#_Toc137626574)

[**Лабораторная работа №10 17**](#_Toc137626575)

[**Лабораторная работа №11 25**](#_Toc137626576)

[**Лабораторная работа №12 33**](#_Toc137626577)

[**Лабораторная работа №13 39**](#_Toc137626578)

МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования

**«Московский технический университет связи и информатики»**

Кафедра «Информатика»

Лабораторная работа №8

**«Программная реализация типовых алгоритмов с регулярными циклическими структурами»**

**по дисциплине**

**«Алгоритмизация и программирование»**

Выполнил: студент гр. БЭИ2202 Кулешов А. С.

Проверил: доц. Шакин В. Н.

Москва, 2022 г

1) Задание на разработку приложения

Создать приложение Windows Forms на C++ с именем Проект-ЛР1, которое должно вычислять периметр и площадь треугольника по заданным значениям координат трех его вершин.

Проект-ЛР1 должен иметь простой интерфейс пользователя:

• две командные кнопки для запуска и завершения работы приложения

• шесть текстовых полей для ввода и отображения входных данных;

• два текстовых поля, отображающих выходные данные;

• поясняющие текстовые надписи.

2) Формализация и уточнение задания

Задано:

x1, y1; x2, y2; x3, y3 –значения координат вершин треугольника

Требуется определить:

Р = A+B+C – периметр треугольника;

S = sqrt(Pp(Pp - A)\*(Pp -B)\*(Pp-C))– площадь треугольника;

где A, B, C – длины сторон треугольника (расстояния между заданными

точками вершин треугольника), определяемые по формулам:

A = sqrt((x1− x2)^2 + (y1− y2)^2) ;

B = sqrt((x3− x2)^2 + (y3− y2)^2) ;

C = sqrt((x3 – x1)^2 + (y3-y1)^2);

Рр = Р/2 – полупериметр.

Должны выполняться следующие ограничения на значения входных данных и соотношения между ними:

A>0 и B>0 и C>0 и A+B>C и A+C>B и B+C>A

Создам проект в Windows Forms. Назову его «Проект-ЛР1».

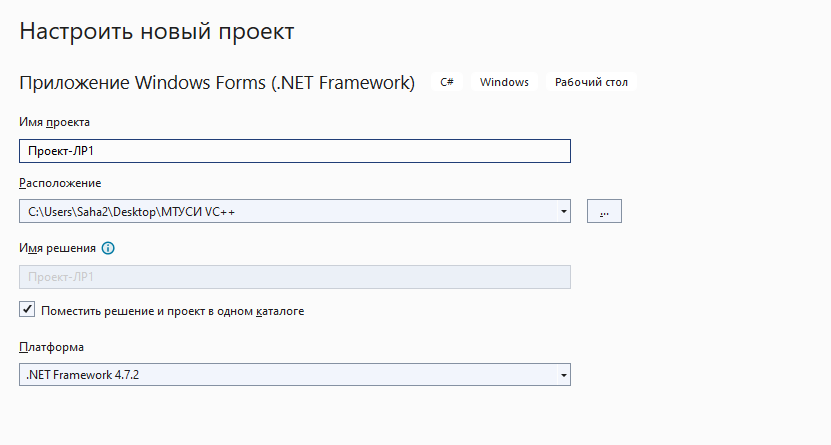


Рисунок 1 – окно создания проекта

Используя панель инструментов создам «Черновой вариант приложения»

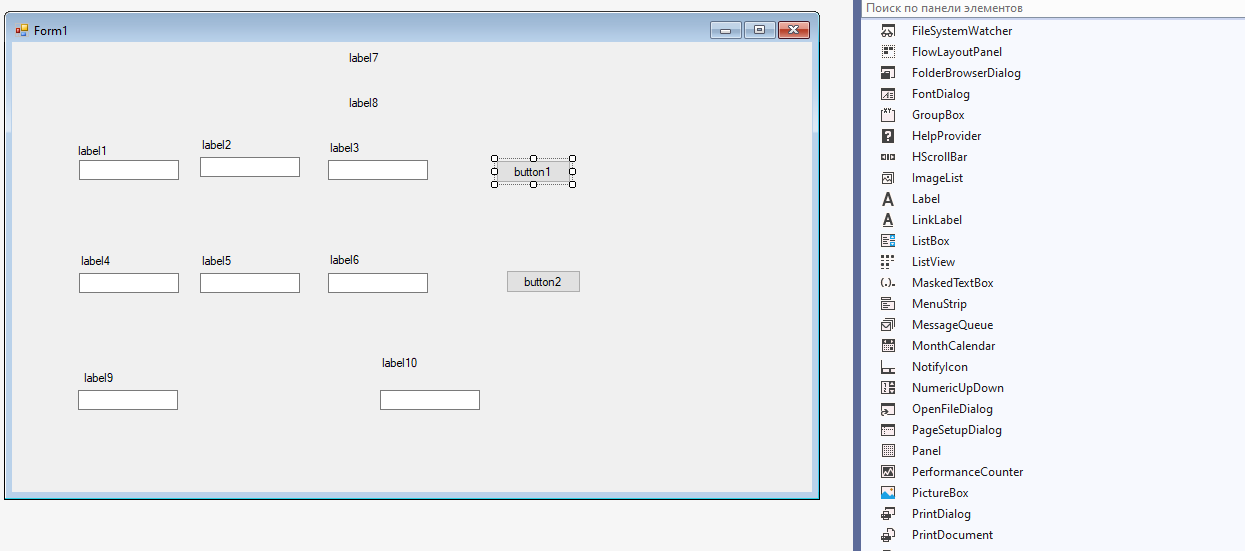


Рисунок 2 – «Черновой вариант»

Теперь подпишу label'ы в соответствии с заданием

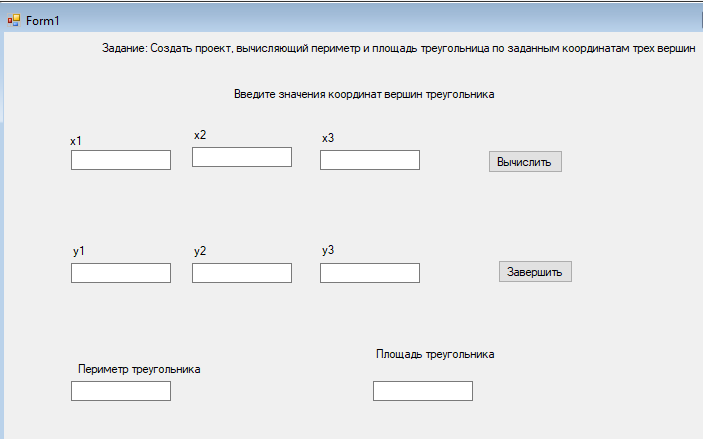


Рисунок 3 – «Черновой вариант» с подписями

Выровняю label’ы и button’ы, чтобы форма смотрелась красивее

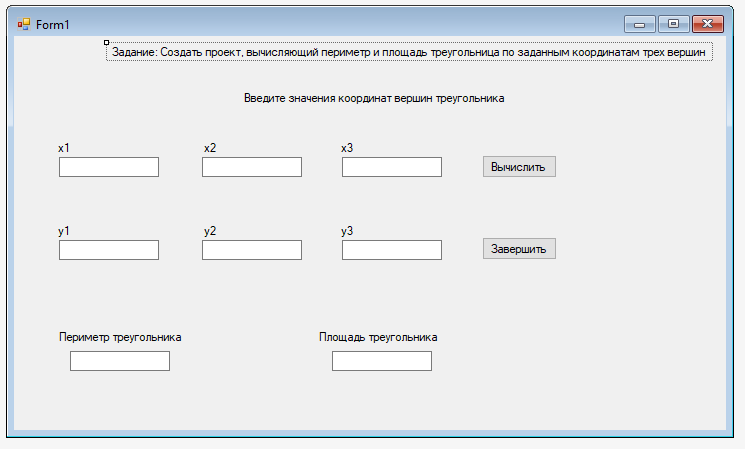


Рисунок 4 – Выровненная форма

Теперь задам соответствующий функционал кнопкам. Для начала получу информацию из textBox’ов, конвертирую её в double, и вычислю A, B, C и Pp.

**private** **void** **ButtonCalc\_Click**(**object** sender, EventArgs e)

{

**double** x1 = Double.Parse(textBoxX1.Text);

**double** x2 = Double.Parse(textBoxX2.Text);

**double** x3 = Double.Parse(textBoxX3.Text);

**double** y1 = Double.Parse(textBoxY1.Text);

**double** y2 = Double.Parse(textBoxY2.Text);

**double** y3 = Double.Parse(textBoxY3.Text);

**double** A = Pythagoras(x1, x2, y1, y2);

**double** B = Pythagoras(x2, x3, y2, y3);

**double** C = Pythagoras(x3, x1, y3, y1); ;

**double** P = A + B + C;

**if** (A <= **0** || B <= **0** || C <= **0** || A + B < +C || A + C <= B || B + C <= A) **return**;

**double** S = Heron(A, B, C);

textBoxP.Text = P.ToString();

textBoxS.Text = S.ToString();

}

Остаётся лишь добавить вычисление периметра и площади, а затем передать эти данные в соответствующие textBox’ы. Также реализую функцию выхода из приложения.

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

{

Application.Exit();

}

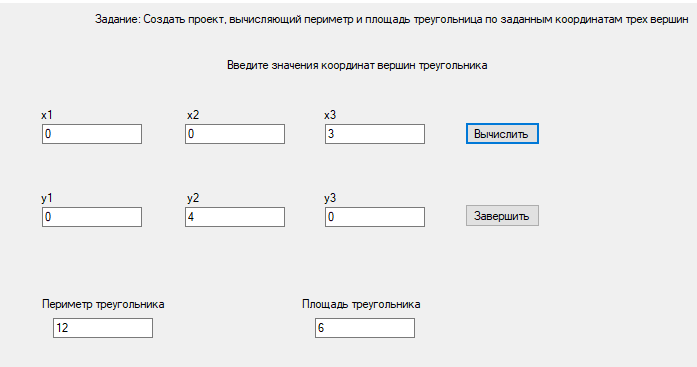


Рисунок 5 – тестирование работы прототипа

Как можно заметить на знаменитом «Египетском треугольнике» приложение работает корректно. Также кнопка выхода работает корректно. Однако, в соответствии с требованиями к работе необходимо сделать несколько правок в код и в графическое оформление формы.

Также, поменяю фон формы на более приятный «мягкий синий». Поставлю соответствующие название, а также поменяю шрифт на «Arial, 12 пт, полужирный» для некоторого текста. Увеличу другие текста, а также сделаю textBox’ы для ответов недоступными для пользолвателя

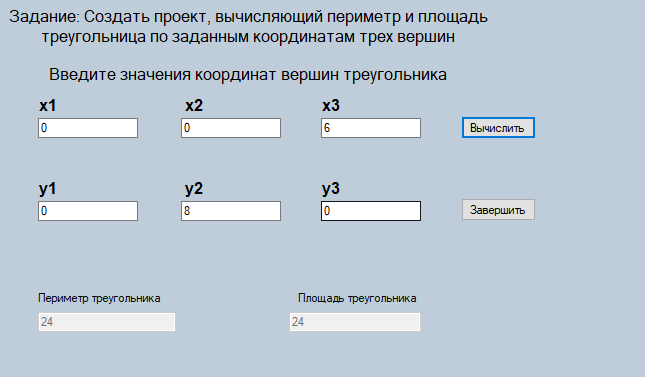


Рисунок 6 – Новые изменения

Теперь остаётся отредактировать код, добавив вычисления теоремы Пифагора и Герона, как отдельные функции.

**double** **sqr**(**double** arg)

{

**return** arg \* arg;

}

// Теорема Пифагора. C = sqrt(a^2 + b^2)

**double** **Pythagoras**(**double** x1, **double** x2, **double** y1, **double** y2)

{

**return** Math.Sqrt(sqr(x1 - x2) + sqr(y1 - y2));

}

// Теорема Герона. Вычисляет площадь треугольника по заданным сторонам

**double** **Heron**(**double** A, **double** B, **double** C)

{

**double** Pp = (A + B + C) / **2**;

**return** Math.Sqrt(Pp \* (Pp - A) \* (Pp - B) \* (Pp - C));

}

Я уже проверял функционал программы для некоторых входных данных, но проверю ещё раз для некоторых других

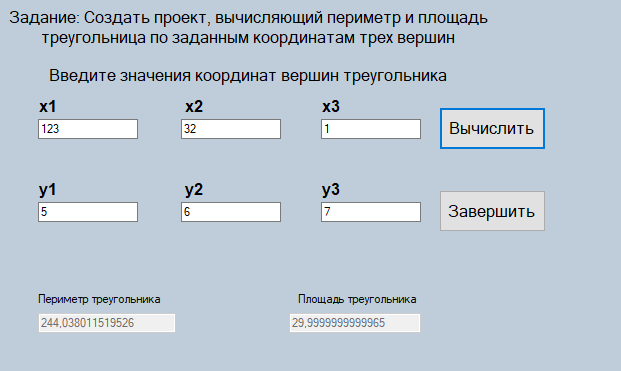


Рисунок 7 – Тест №1

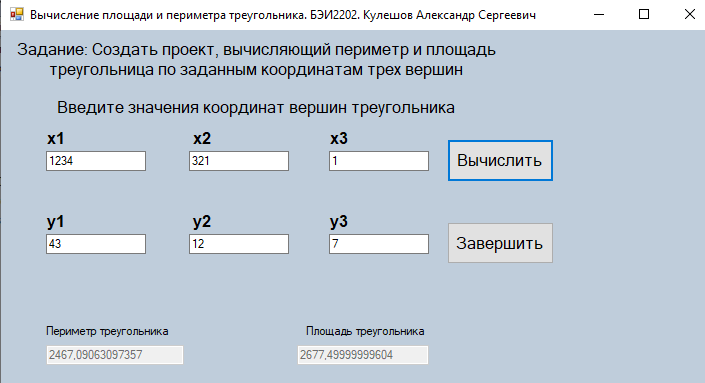


Рисунок 8 – Тест №2

Результат работы программы на тестах совпадает с результатом, полученным при вычисление на калькуляторе. Программа работает корректно

МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования

**«Московский технический университет связи и информатики»**

Кафедра «Информатика»

Лабораторная работа №9

**«Структурированные данные. Программирование**

**алгоритмов формирования и обработки одномерных массивов»**

**по дисциплине**

**«Алгоритмизация и программирование»**

Выполнил: студент гр. БЭИ2202 Кулешов А. С.

Проверил: доц. Шакин В. Н.

Вариант: 17

Москва, 2023 г

1. Общее Задание.

1) Изучите вопросы программирования алгоритмов формирования и обработки одномерных массивов.

2) Выберите вариант задания из таблицы 1

3) Разработайте проект графического интерфейса пользователя.

4) Решите поставленную задачу формирования и обработки автоматических

одномерных массивов. Для этого разработайте схемы алгоритмов и

программные коды следующих функций:

• функции ввода исходных данных, предусмотрев контроль значений

исходных данных;

• функции формирования одномерного массива из случайных чисел в

заданном диапазоне;

• функции создания нового одномерного массива согласно вашему

индивидуальному

заданию. При необходимости разбейте эту функцию на ряд вспомогательных функций;

• функции вывода одномерного массива в элемент управления

ListBox.

5) Разработайте программный код событийной процедуры, которая вызывает описанные выше функции для решения поставленной задачи.

6) Создайте проект приложения Windows Form, содержащий:

• графический интерфейс;

• файлы с раздельно откомпилированными программными кодами

функций, необходимых для решения задачи:

* функций ввода исходных данных, формирования исходного массива

и вывода результатов;

* функций, решающих поставленную задачу.
* заголовочный файл с прототипами разработанных функций;
* событийную процедуру, которая должна содержать только операторы

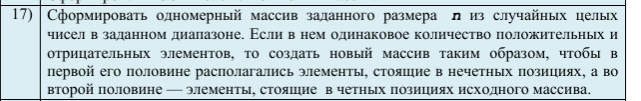
вызова пользовательских функций.

7) Подготовьте тесты для контрольного решения задачи, предусмотрев

различные значения исходного массива данных для решения. Для

проверки работы функции создания нового массива при различных

1. Индивидуальное задание



1. Формализация и уточнение задания

Из условия задания следует, что его выполнение распадается на

следующие этапы, реализуемые соответствующими функциями приложения:

• формирование исходного массива из случайных целых чисел – функция

input;

• формирование нового массива из простых чисел исходного массива –

функция task;

• вывод нового массива из простых чисел в заданный список (элемент

управления ListBox) формы – функция output.

Для проверки, является ли элемент исходного массива простым

числом, используется вспомогательная функция simple.

Вызов функций формирования массивов и вывода нового массива

производится в событийной процедуре по нажатию кнопки Решить задачу.

1. Схема алгоритма

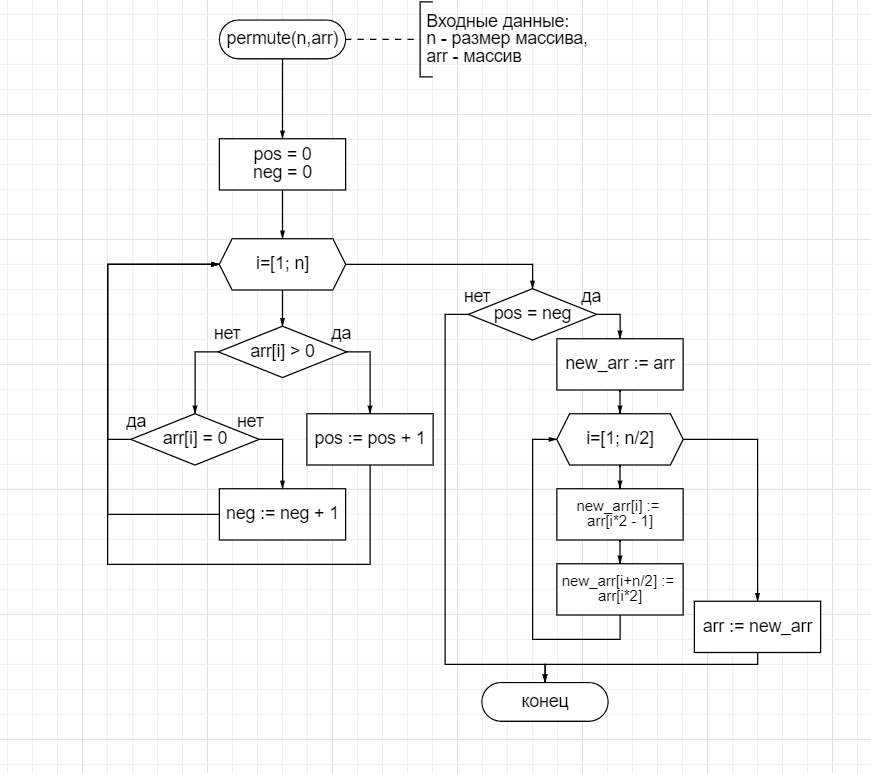


Рисунок 1 – схема алгоритма преобразования массива

1. Теперь создам форму

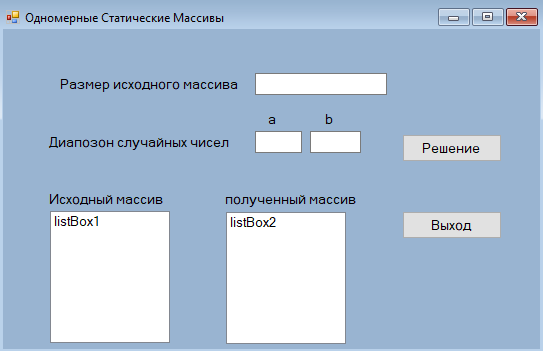


Рисунок 2 – полученная форма

Теперь реализую код для кнопок

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

{

listBox1.Items.Clear();

listBox2.Items.Clear();

**int** n = **int**.Parse(textBox1.Text);

**int** a = **int**.Parse(textBox2.Text);

**int** b = **int**.Parse(textBox3.Text);

**int**[] arr = **new** **int**[n];

Random rnd = **new** Random();

**for** (**int** i = **0**;i < n; ++i)

{

arr[i] = rnd.Next() % (b-a+**1**) + a;

listBox1.Items.Insert(i, arr[i]);

Console.WriteLine(arr[i]);

}

arr = permute(n, arr);

**for** (**int** i = **0**; i < n; ++i)

{

listBox2.Items.Insert(i, arr[i]);

}

}

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

{

Application.Exit();

}

А теперь реализую функцию permute

**int**[] **permute**(**int** n, **int**[] arr)

{

**int** pos = **0**;

**int** neg = **0**;

**for** (**int** i = **0**; i < n; ++i)

{

**if** (arr[i] > **0**)

{

pos++; **continue**;

}

**if** (arr[i] < **0**)

{

neg++;

}

}

**if** (neg != pos) **return** arr;

**int**[] new\_arr = **new** **int**[n];

**for**(**int** i = **0**; i < n/**2**; ++i)

{

new\_arr[i] = arr[i \* **2**];

new\_arr[n / **2** + i] = arr[i \* **2** + **1**];

}

**return** new\_arr;

}

Наконец, проверю работоспособность алгоритма на тестовых данных

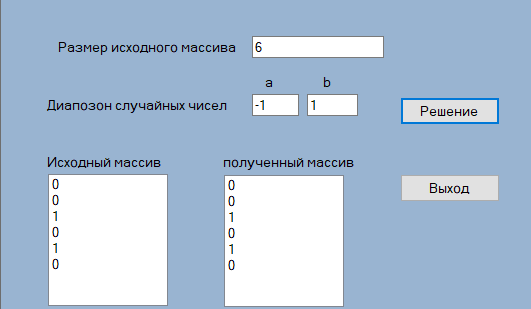


Рисунок 3 – Первый набор тестовых данных

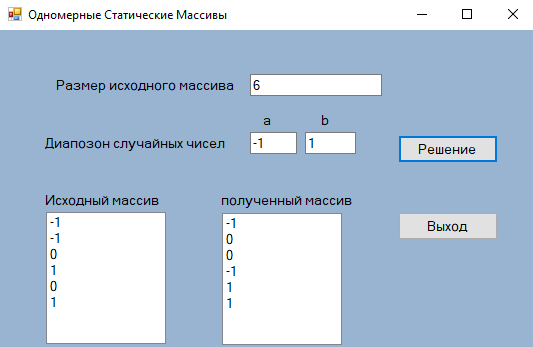


Рисунок 4 – Второй набор тестовых данных

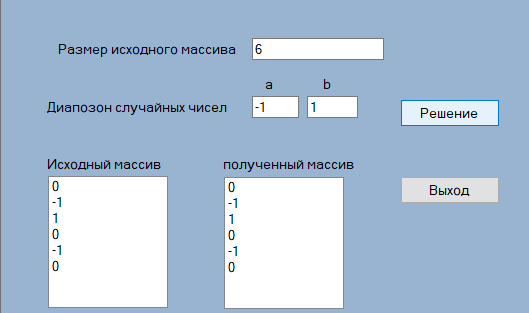


Рисунок 5 – Третий набор тестовых данных

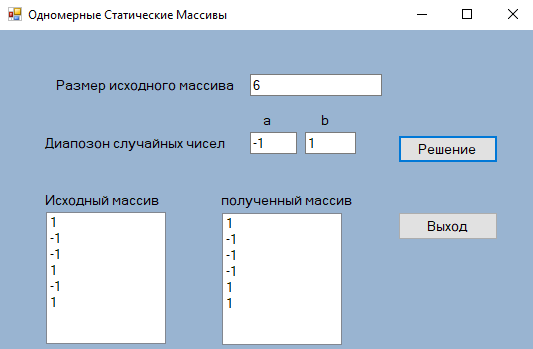


Рисунок 6 – Четвёртый набор тестовых данных

Как можно заметить в первом и третьем наборе тестовых данных количество отрицательных и положительных элементов не совпадает, поэтому массив не меняется.

Во втором и четвёртых примерах количество отрицательных и положительных элементов совпадает, поэтому массив изменяется.

Программа работает корректно

МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования

**«Московский технический университет связи и информатики»**

Кафедра «Информатика»

Лабораторная работа №10

**«Создание приложений Windows Forms**

**для формирования и обработки динамических**

**одномерных массивов»**

**по теме**

**«Указатели и динамические массивы»**

**по дисциплине**

**«Алгоритмизация и программирование»**

Выполнил: студент гр. БЭИ2202 Кулешов А. С.

Проверил: доц. Шакин В. Н.

Вариант: 17

Москва, 2023 г

1. Общее Задание.

1) Изучите вопросы использования указателей и программирования алгоритмов формирования и обработки одномерных динамических массивов.

2) Выберите вариант задания из таблицы 1

3) Разработайте проект графического интерфейса пользователя.

4) Решите поставленную задачу формирования и обработки динамических

массивов. Для этого разработайте схемы алгоритмов и программные

коды следующих функций:

• функции ввода и контроля дополнительных данных (при наличии

таких данных в индивидуальном задании);

• функции ввода элементов исходного массива из многострочного

текстового поля, создания и формирования динамического массива

из этих элементов;

• функции создания нового динамического одномерного массива

согласно вашему индивидуальному заданию.

• функции вывода динамического одномерного массива в элемент

управления ListBox.

5) Разработайте программный код событийной процедуры, которая вызывает

описанные выше функции для решения поставленной задачи.

6) Создайте проект приложения Windows Form, содержащий:

• графический интерфейс;

• файлы с раздельно откомпилированными программными кодами

функций, необходимых для решения задачи:

• функций ввода данных, создания и формирования массива и вывода

результатов;

• функций, решающих поставленную задачу.

• заголовочный файл с прототипами разработанных функций;

• событийную процедуру, которая должна содержать только операторы

вызова пользовательских функций.

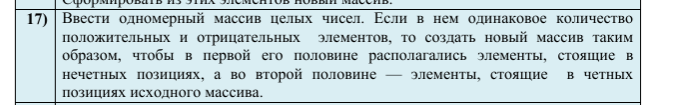
7) Подготовьте тесты для контрольного решения задачи, предусмотрев

различные значения исходного массива данных для решения.

8) Выполните проект и получите результаты.

9) Докажите правильность полученных результатов на заранее разработанных тестах.

1. Индивидуальное задание



1. Формализация и уточнение задания

Из условия задания следует, что его выполнение распадается на

следующие этапы, реализуемые соответствующими функциями приложения:

• формирование исходного массива чтением чисел из многострочного

текстового поля формы – функция input;

• определение количества простых чисел в исходном массиве, чтобы

узнать, надо ли формировать новый массив – функция analys;

• формирование нового массива из простых чисел исходного массива

– функция task;

• вывод нового массива из простых чисел в заданный список (элемент

управления ListBox) формы – функция output.

Вызов функций формирования динамических массивов и вывода

нового массива производится в событийной процедуре по нажатию кнопки

Решить задачу.

1. Схема алгоритма

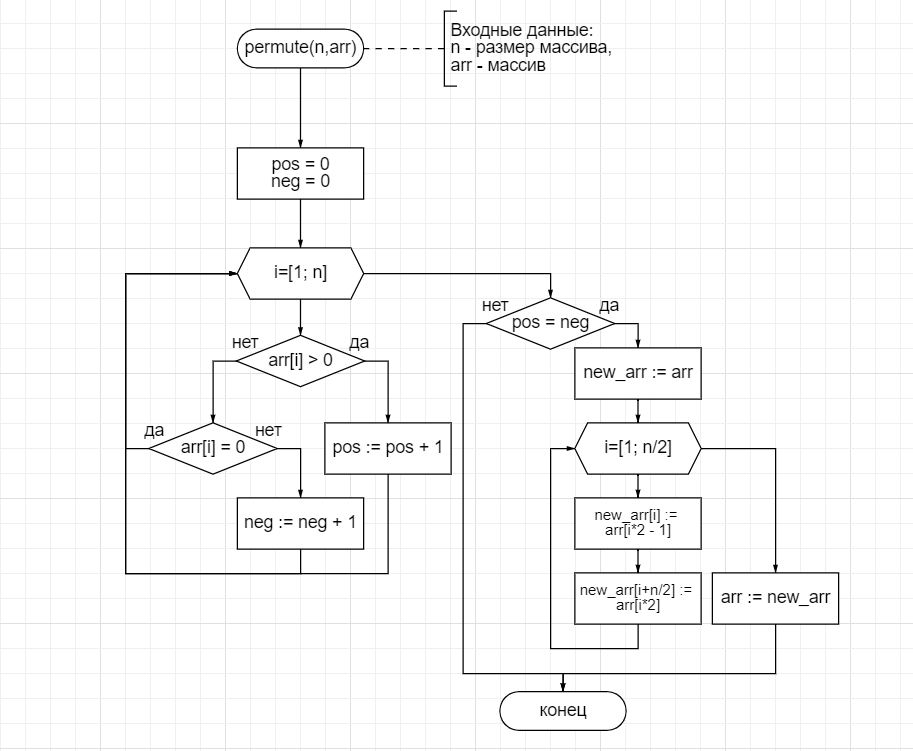


Рисунок 1 – схема алгоритма преобразования массива

1. Теперь создам форму

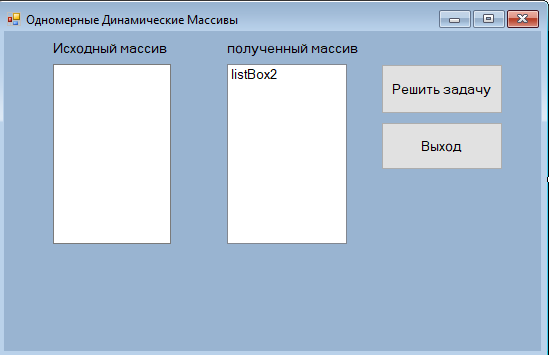


Рисунок 2 – полученная форма

Теперь реализую код для кнопок

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

{

listBox2.Items.Clear();

**string** data = textBox1.Text;

**int** size = data.Count(t => t == '\n') + **1**;

Console.WriteLine(data);

**int**[] arr = **new** **int**[size];

**int** h = **0**;

**string** carret = "";

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

{

**if** (data[i] == '\n'){

arr[h++] = Int32.Parse(carret);

carret = "";

}**else**{

carret += data[i];

}

}

arr[h++] = Int32.Parse(carret);

arr = permute(size, arr);

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

{

listBox2.Items.Insert(i, arr[i]);

}

}

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

{

Application.Exit();

}

Затем реализую функцию permute, которая преобразует массив согласно условию

**int**[] **permute**(**int** n, **int**[] arr)

{

**int** pos = **0**;

**int** neg = **0**;

**for** (**int** i = **0**; i < n; ++i)

{

**if** (arr[i] > **0**)

{

pos++; **continue**;

}

**if** (arr[i] < **0**)

{

neg++;

}

}

**if** (neg != pos) **return** arr;

**int**[] new\_arr = **new** **int**[n];

**for**(**int** i = **0**; i < n/**2**; ++i)

{

new\_arr[i] = arr[i \* **2**];

new\_arr[n / **2** + i] = arr[i \* **2** + **1**];

}

**return** new\_arr;

}

Наконец, проверю работоспособность алгоритма на тестовых данных

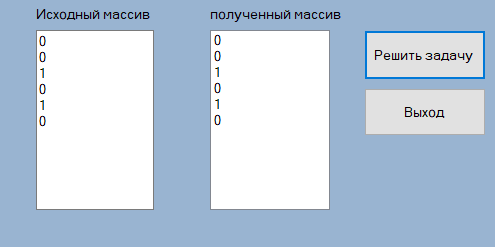


Рисунок 3 – Первый набор тестовых данных

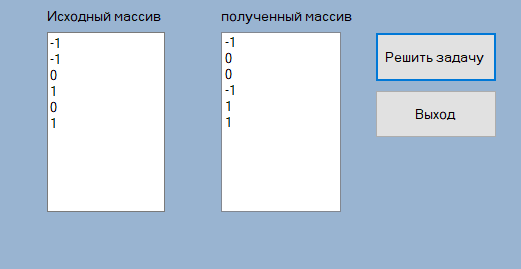


Рисунок 4 – Второй набор тестовых данных

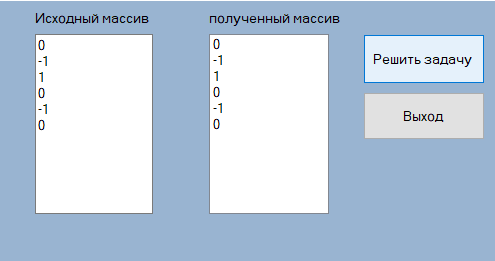


Рисунок 5 – Третий набор тестовых данных

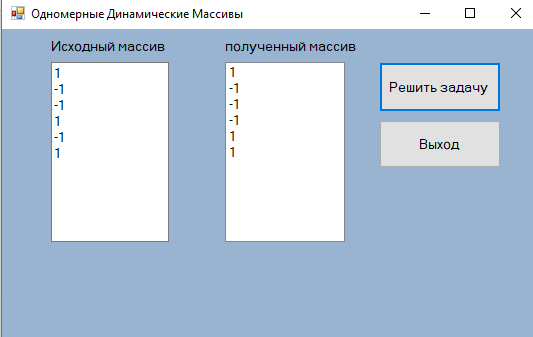


Рисунок 6 – Четвёртый набор тестовых данных

Как можно заметить в первом и третьем наборе тестовых данных количество отрицательных и положительных элементов не совпадает, поэтому массив не меняется.

Во втором и четвёртых примерах количество отрицательных и положительных элементов совпадает, поэтому массив изменяется.

Программа работает корректно

МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования

**«Московский технический университет связи и информатики»**

Кафедра «Информатика»

Лабораторная работа №11

**«Создание приложений Windows Forms для**

**формирования и обработки двумерных массивов»**

**по теме**

**«Формирование и обработка двумерных массивов»**

**по дисциплине**

**«Алгоритмизация и программирование»**

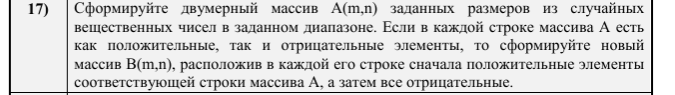
Выполнил: студент гр. БЭИ2202 Кулешов А. С.

Проверил: доц. Шакин В. Н.

Вариант: 17

Москва, 2023 г

1. Индивидуальное задание



1. Схема алгоритма

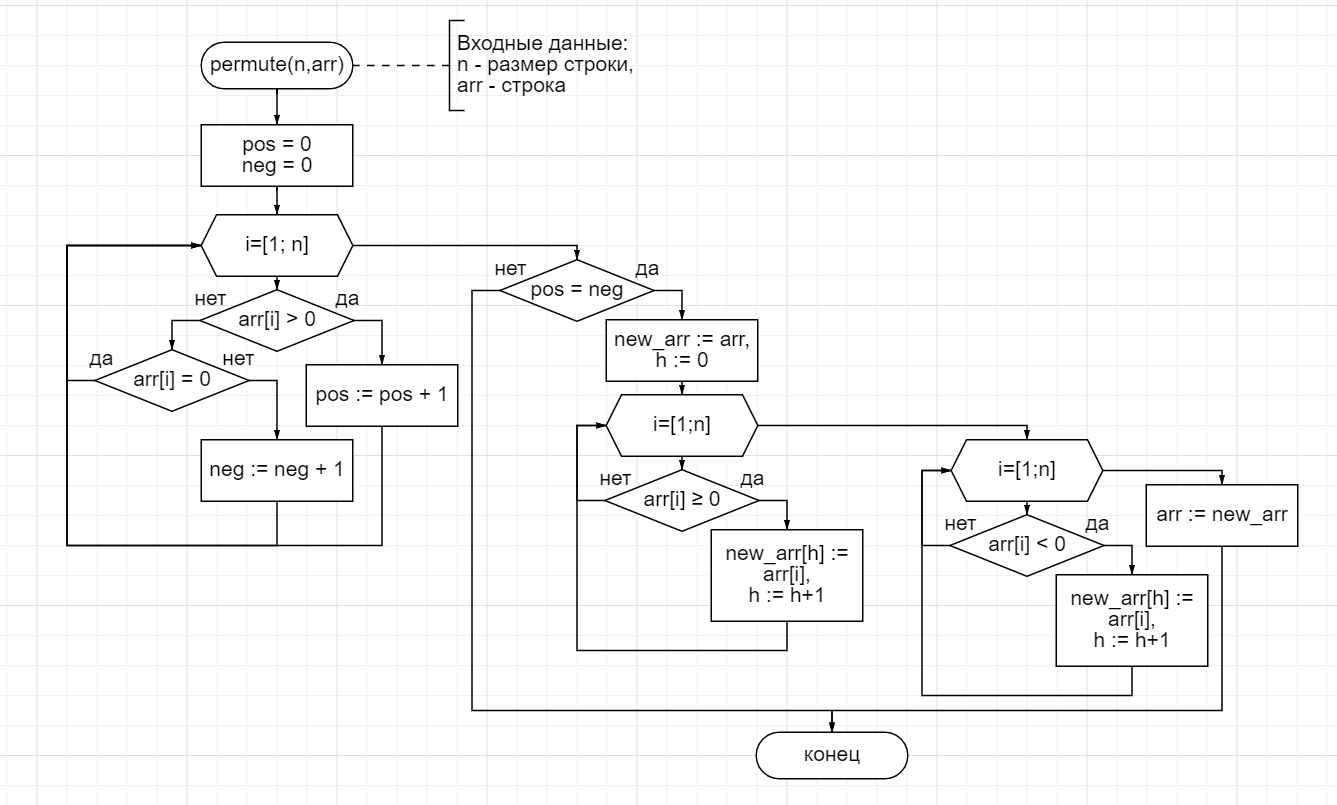


Рисунок 1 – схема алгоритма преобразования массива

1. Теперь создам форму

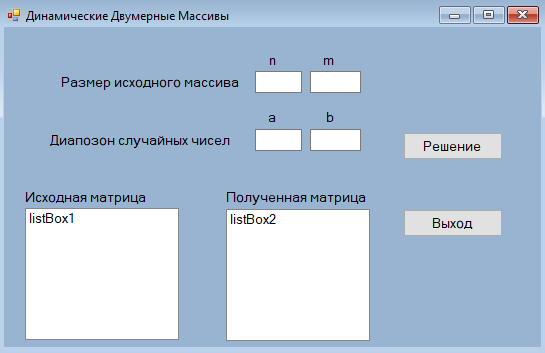


Рисунок 2 – полученная форма

Теперь реализую код для кнопок

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

{

listBox1.Items.Clear();

listBox2.Items.Clear();

**int** n = **int**.Parse(textBox1.Text);

**int** m = **int**.Parse(textBox4.Text);

**int** a = **int**.Parse(textBox2.Text);

**int** b = **int**.Parse(textBox3.Text);

**double**[,] arr = **new** **double**[n, m];

Random rnd = **new** Random();

**for** (**int** i = **0**;i < n; ++i)

{

**string** str = "";

**for** (**int** j = **0**; j < m; ++j)

{

arr[i,j] = rnd.NextDouble() \* (b - a) + a;

str += arr[i, j].ToString("00.00") + " ";

Console.WriteLine(arr[i,j]);

}

listBox1.Items.Insert(i, str);

}

arr = permute(m, arr,n);

**for** (**int** i = **0**; i < n; ++i)

{

**string** str = "";

**for** (**int** j = **0**; j < m; ++j)

{

str += arr[i, j].ToString("00.00") + " ";

Console.WriteLine(arr[i, j]);

}

listBox2.Items.Insert(i, str);

}

}

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

{

Application.Exit();

}

Теперь создам функцию, которая преобразует двумерный массив. Назову её permute

**double**[,] permute(**int** n, **double**[,] arr, **int** nn)

{

**double**[,] new\_arr = **new** **double**[nn, n];

**for** (**int** row = **0**; row < nn; ++row) {

**int** pos = **0**, neg = **0**, h =**0**;

**for** (**int** i = **0**; i < n; ++i)

{

**if** (arr[row, i] >= **0**){

pos++; **continue**;

}

**if** (arr[row, i] < **0**)

neg++;

}

**if** (neg != pos){ // Не менять массив

**for** (**int** i = **0**; i < n; ++i){

new\_arr[row, h++] = arr[row, i];

}

**continue**;

}

**for** (**int** i = **0**; i < n; ++i){ // Положительные

**if** (arr[row, i] >= **0**)

new\_arr[row, h++] = arr[row, i];

}

**for** (**int** i = **0**; i < n; ++i){ // Отрицательные

**if** (arr[row, i] < **0**)

new\_arr[row, h++] = arr[row, i];

}

}

**return** new\_arr;

}

Наконец, проверю работоспособность алгоритма на тестовых данных

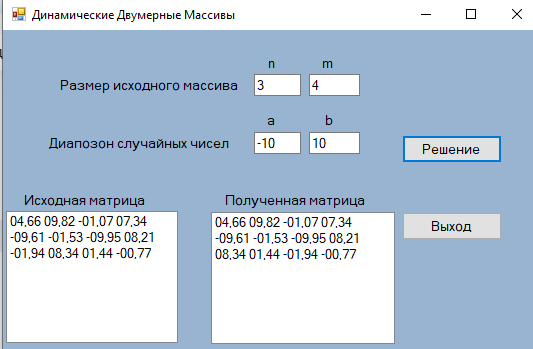


Рисунок 3 – Первый набор тестовых данных

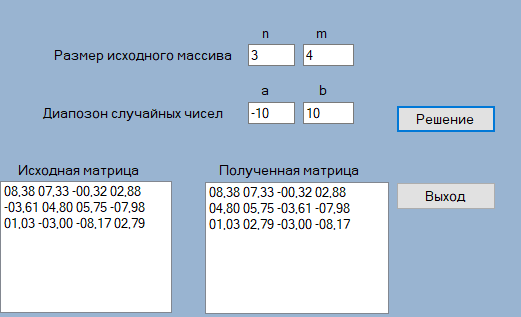


Рисунок 4 – Второй набор тестовых данных

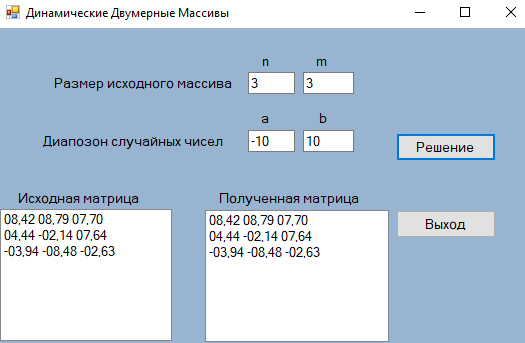


Рисунок 5 – Третий набор тестовых данных

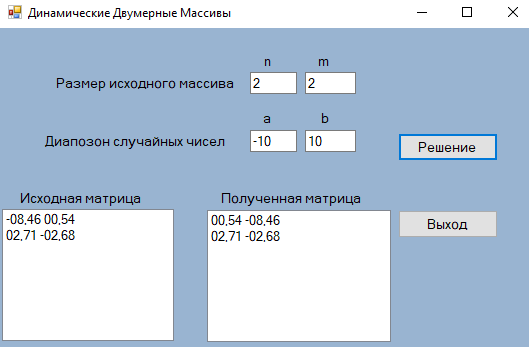


Рисунок 6 – Четвёртый набор тестовых данных

МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования

**«Московский технический университет связи и информатики»**

Кафедра «Информатика»

Лабораторная работа №12

**«Создание приложений Windows Forms**

**для обработки строк»**

**по теме**

**«Символьные и строковые данные в С++»**

**по дисциплине**

**«Алгоритмизация и программирование»**

Выполнил: студент гр. БЭИ2202 Кулешов А. С.

Проверил: доц. Шакин В. Н.

Вариант: 17

Москва, 2023 г

1. Индивидуальное задание



1. Схема алгоритма

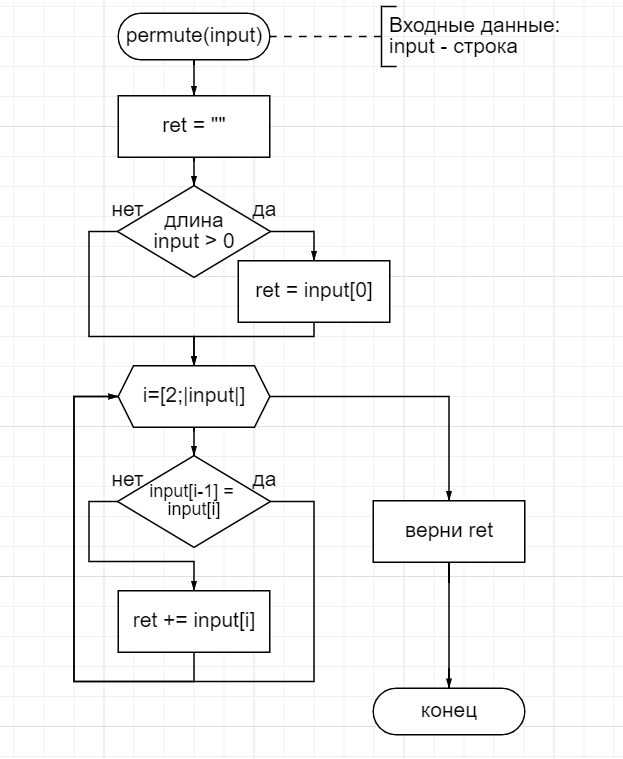


Рисунок 1 – схема алгоритма преобразования строки

1. Теперь создам форму

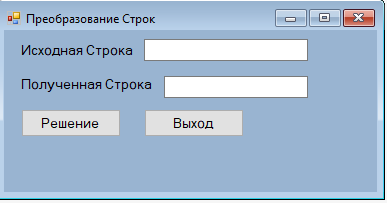


Рисунок 2 – полученная форма

Теперь реализую код для кнопок

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

{

textBox2.Text = permute(textBox1.Text);

}

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

{

Application.Exit();

}

Теперь реализую код для функции, которая будет преобразовывать строку. Назову её, как и в своих прежних работах - permute

String **permute**(String input)

{

String ret = "";

**if** (input.Length > **0**) ret = (input[**0**]).ToString();

**for**(**int** i = **1**;i < input.Length; ++i)

{

**if** (input[i - **1**] != input[i]) ret += input[i];

}

**return** ret;

}

Наконец, проверю работоспособность алгоритма на тестовых данных

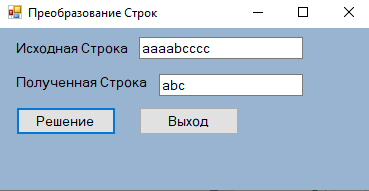


Рисунок 3 – Первый набор тестовых данных

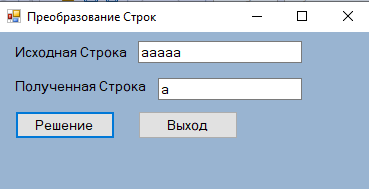


Рисунок 4 – Второй набор тестовых данных

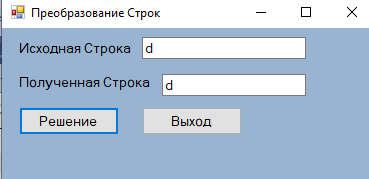


Рисунок 5 – Третий набор тестовых данных

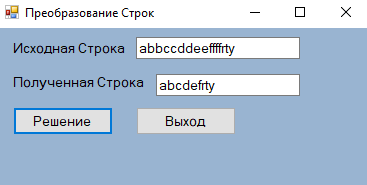


Рисунок 6 – Четвёртый набор тестовых данных

Программа работает корректно на наборах тестовых данных.

МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования

**«Московский технический университет связи и информатики»**

Кафедра «Информатика»

Лабораторная работа №13

**«Создание приложений Windows Forms**

**для работы с потоковыми файлами»**

**по теме**

**«Создание приложений, работающих с потоковыми**

**файлами»**

**по дисциплине**

**«Алгоритмизация и программирование»**

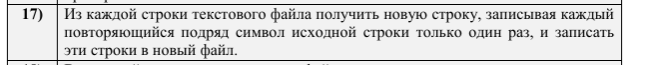
Выполнил: студент гр. БЭИ2202 Кулешов А. С.

Проверил: доц. Шакин В. Н.

Вариант: 17

Москва, 2023 г

1. Индивидуальное задание



1. Схема алгоритма

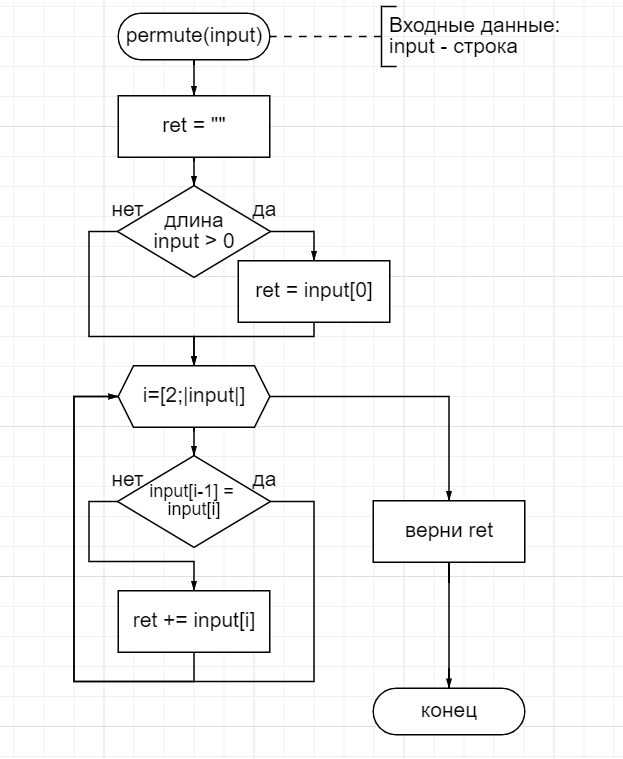


Рисунок 1 – схема алгоритма преобразования строки

1. Теперь создам форму

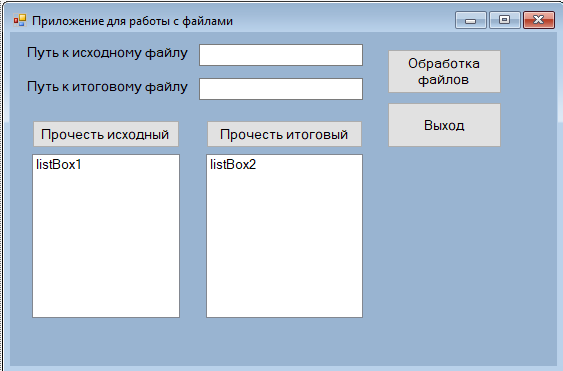


Рисунок 2 – полученная форма

Теперь реализую код для кнопок

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

{

**string**[] lines = System.IO.File.ReadAllLines(textBox1.Text);

**string** final\_content = "";

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

{

final\_content += permute(lines[i]) + '\n';

}

System.IO.File.WriteAllText(textBox2.Text, final\_content);

}

**private** **void** **button4\_Click**(**object** sender, EventArgs e)

{

listBox2.Items.Clear();

**string**[] lines = System.IO.File.ReadAllLines(textBox2.Text);

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

{

listBox2.Items.Insert(i, lines[i]);

}

}

**private** **void** **button3\_Click**(**object** sender, EventArgs e)

{

listBox1.Items.Clear();

**string**[] lines = System.IO.File.ReadAllLines(textBox1.Text);

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

{

listBox1.Items.Insert(i, lines[i]);

}

}

Теперь реализую функцию для преобразования строки

String **permute**(String input)

{

String ret = "";

**if** (input.Length > **0**) ret = (input[**0**]).ToString();

**for**(**int** i = **1**;i < input.Length; ++i)

{

**if** (input[i - **1**] != input[i]) ret += input[i];

}

**return** ret;

}

Наконец, проверю работоспособность программы.

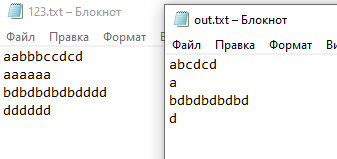


Рисунок 3 – функционал файлового ввода-вывода работает

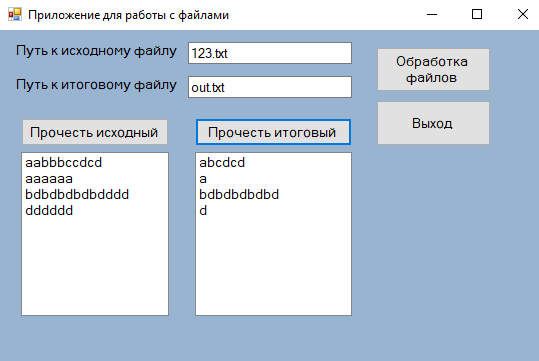


Рисунок 4 – Вывод в listBox’ы тоже работает

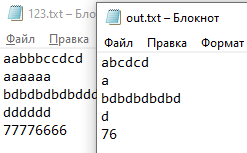


Рисунок 5 – Ещё раз отредактируем файл 123.txt

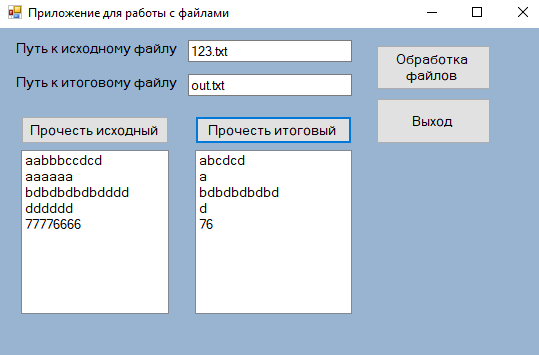


Рисунок 6 – Ещё раз проверим вывод в listBox’ы