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 3**](#_Toc137637884)

[**Лабораторная работа №9**](#_Toc137637885) **13**

[**Лабораторная работа №10**](#_Toc137637886) **21**

[**Лабораторная работа №11**](#_Toc137637887) **30**

[**Лабораторная работа №12**](#_Toc137637888) **38**

**[Лабораторная работа №13](#_Toc137637889) 44**

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

**«Интегрированная среда разработки Visual Studio .NET и технология визуального событийного**

**программирования»**

**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++, которое

должно вычислять периметр и площадь треугольника по заданным значени-

ям 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 = – площадь треугольника;

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

точками вершин треугольника), определяемые по формулам:

A = ;

B = ;

C = ;

р = 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

**3) Разработка приложения**

3.1) Создание проекта

Запустить VS, и, выбрав шаблон .NET Framework приложения WinForms, назову проект «Проект-ЛР1». Рисунок 1.

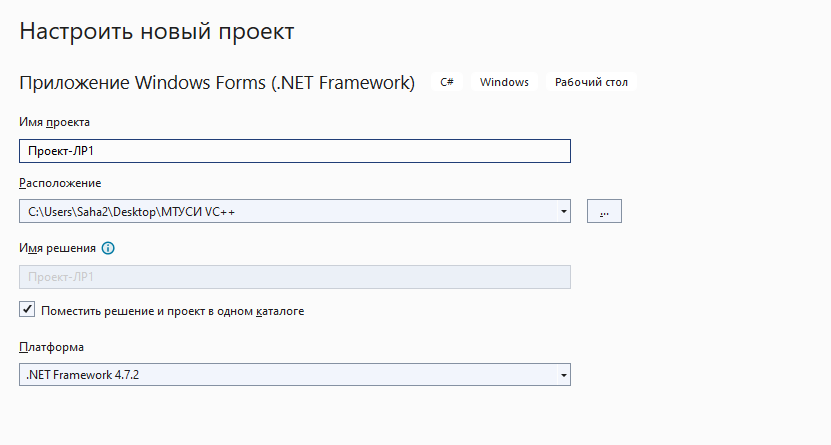


Рисунок 1 – окно создания проекта

3.2) Разработка графического интерфейса пользователя

Изменить масштаб формы, добавить в нее кнопки, текст и элементы ввода текста. Рисунок 2.

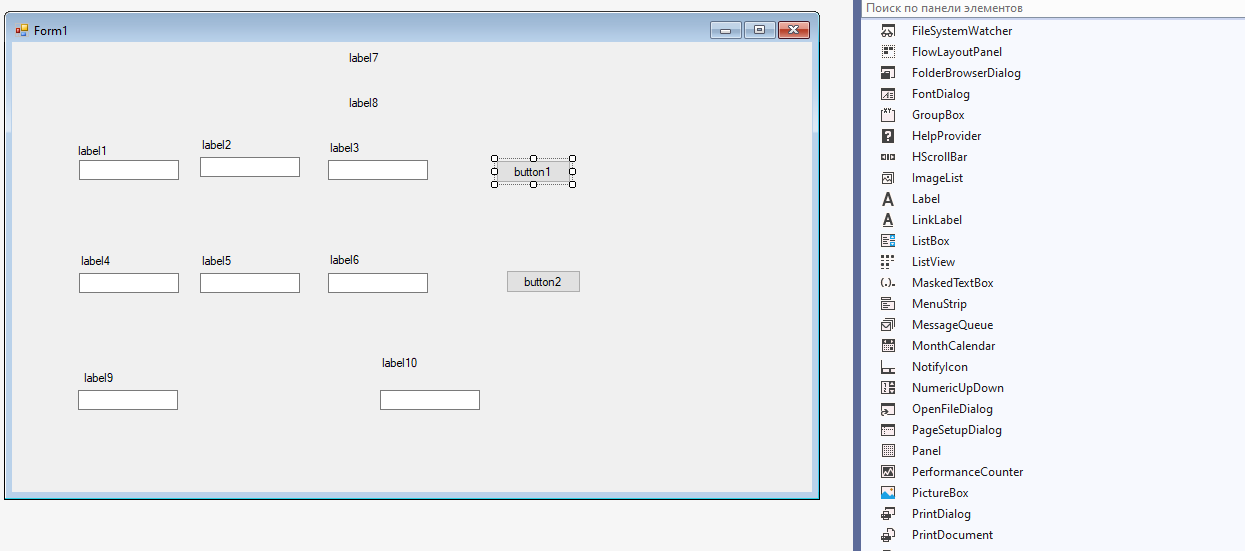


Рисунок 2 – Макет формы

3.3) Установка свойств объектов

Изменив свойства объекта button1, переключиться на вторую кнопку button2 при помощи Окна Свойств. Рисунок 3.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 3 – Все объекты формы

После подробного изучения окна свойств разработка пользовательского интерфейса окончена. Рисунок 4.

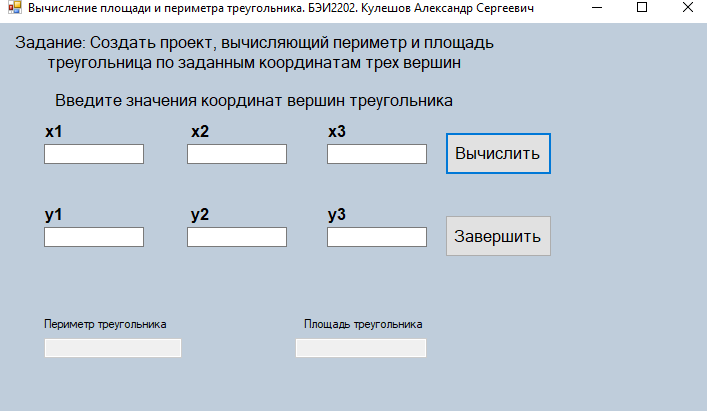


Рисунок 4 – Готовый пользовательский интерфейс

3.4) Таблица используемых объектов и их свойств

В таблицу используемых объектов и их свойств заносятся все объекты формы, значения их свойства ***Name*** и тех свойств, которые отличаются от установленных **VS** по умолчанию. В таблице 1 приведены некоторые объекты формы с их свойствами.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Объект** | **Класс объекта** | | **Свойство** | | **Значение свойства** | |
| Форма | Form | | Name | | MyForm | |
| Text | | Вычисление периметра и площади треугольника (Кулешов Александр Сергеевич. БЭИ2202) | |
| BackColor | | Control | |
| Текстовое поле | TextBox | | Name | | textBoxX1 | |
| Text | |  | |
| Font | | Microsoft Sans Serif; 7,8pt | |
| Метка | Label | Name | | Label | |
| Text | | X1 | |
| TextAlign | | MiddleCenter | |
| Font | | Arial Black; 12pt; style=Bold | |
| ForeColor | | Maroon | |
| BorderStyle | | FixedSingle | |
| Кнопка | Button | Name | | ButtonCalc | |
| Text | | Вычислить | |
| Font | | Microsoft Sans Serif; 7,8pt | |

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

3.4) Схемы алгоритмов процедур и схема иерархии процедур

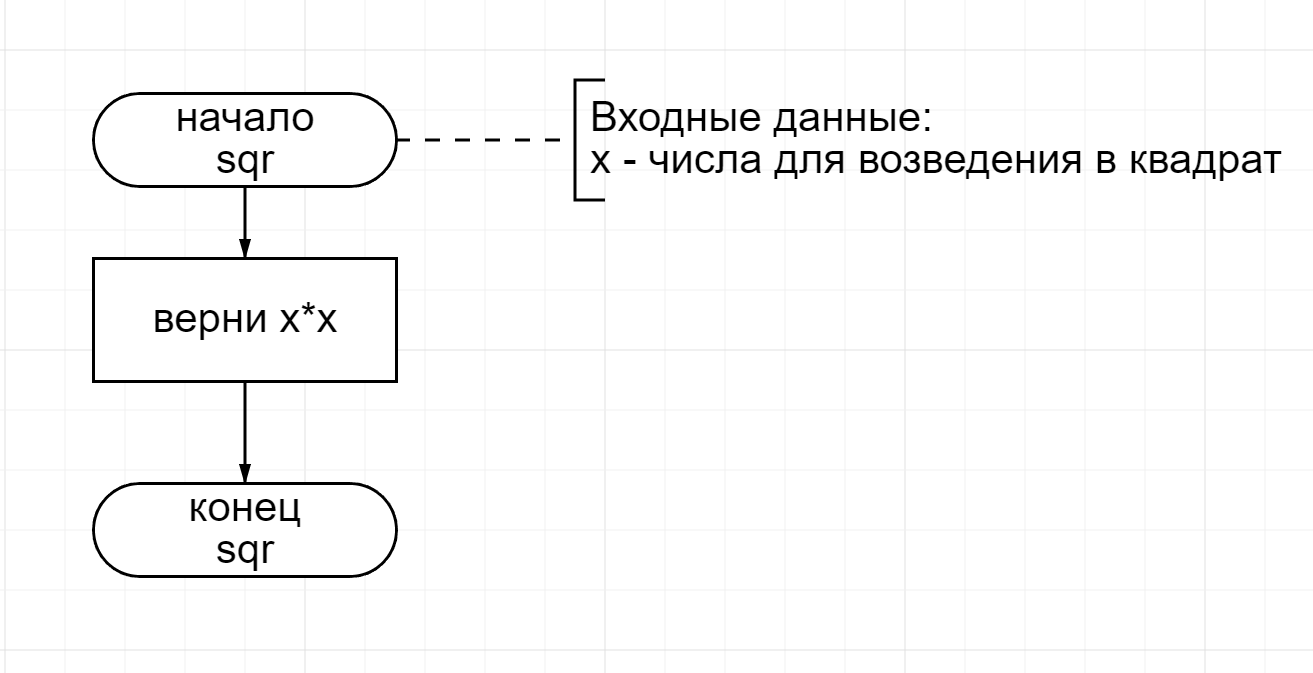


Рисунок 5 – Cхема алгоритма функции sqr

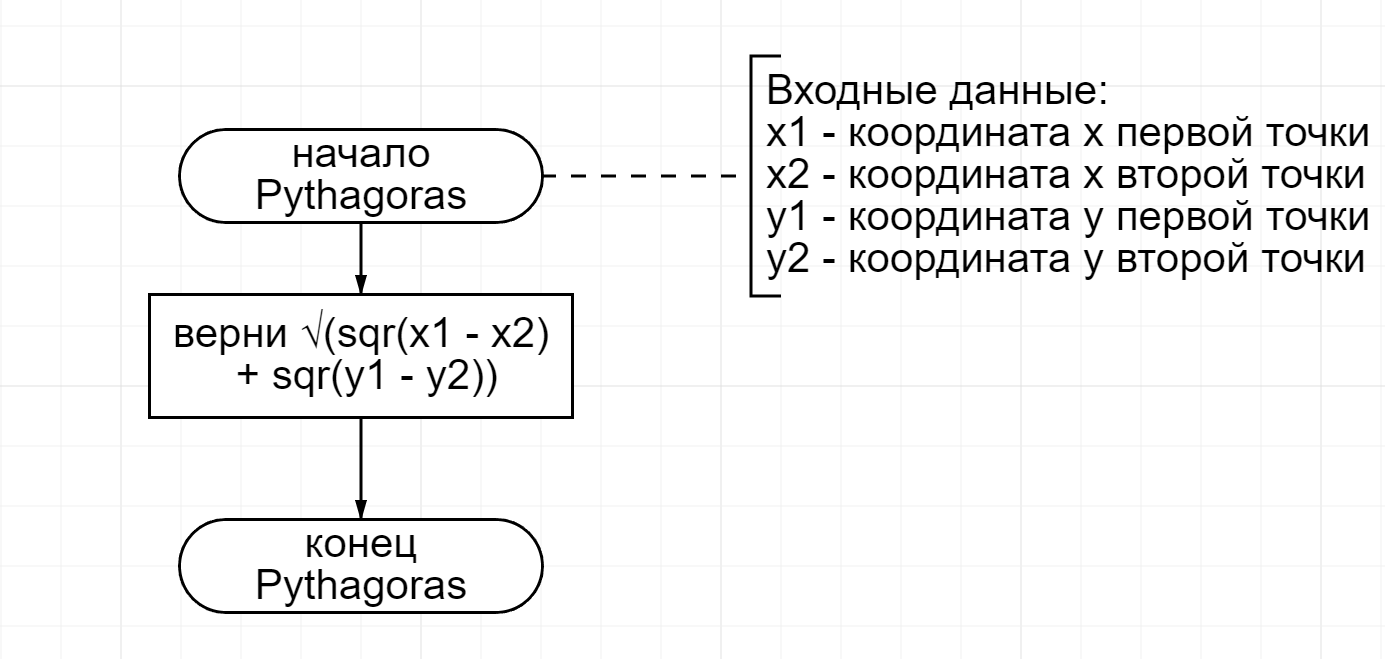


Рисунок 6 - Cхема алгоритма функции Pythogoras

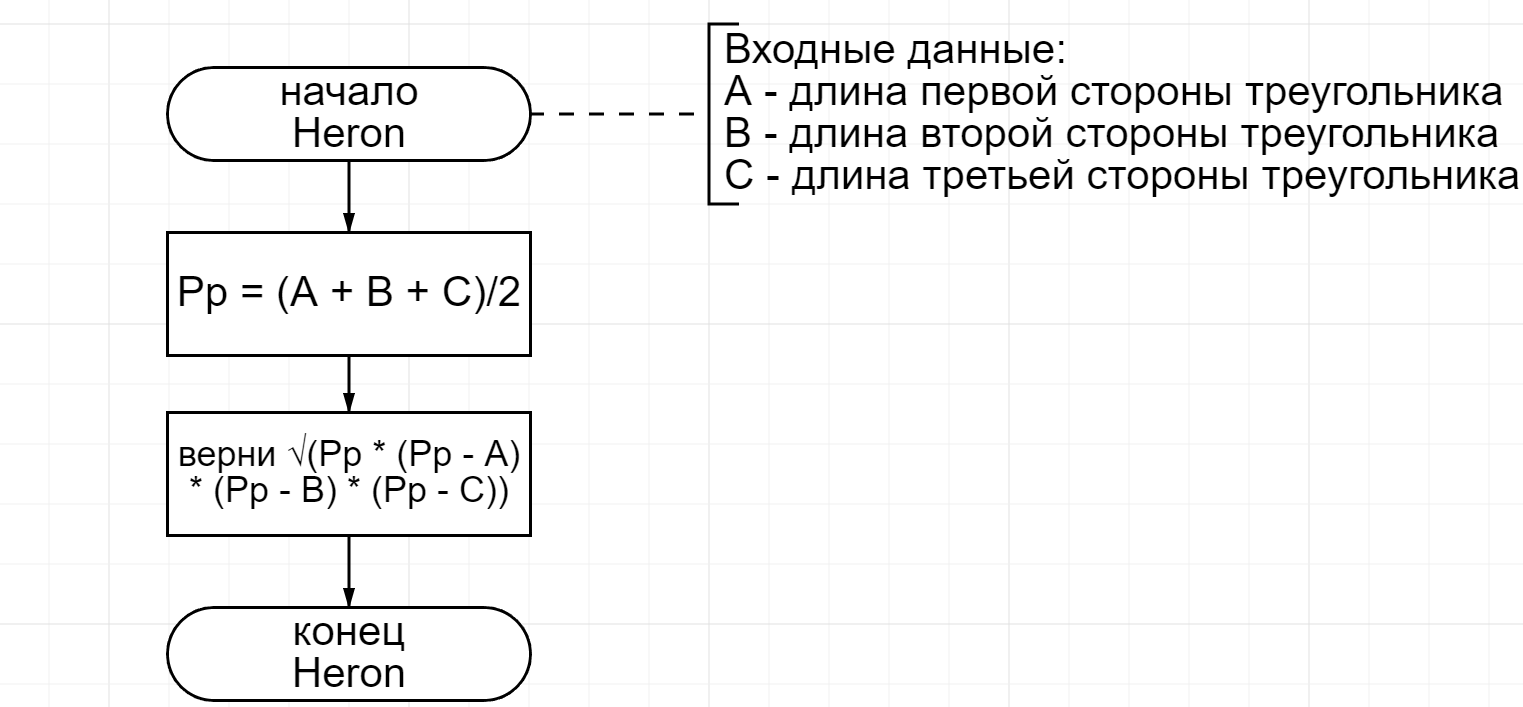


Рисунок 7 - Cхема алгоритма функции Heron

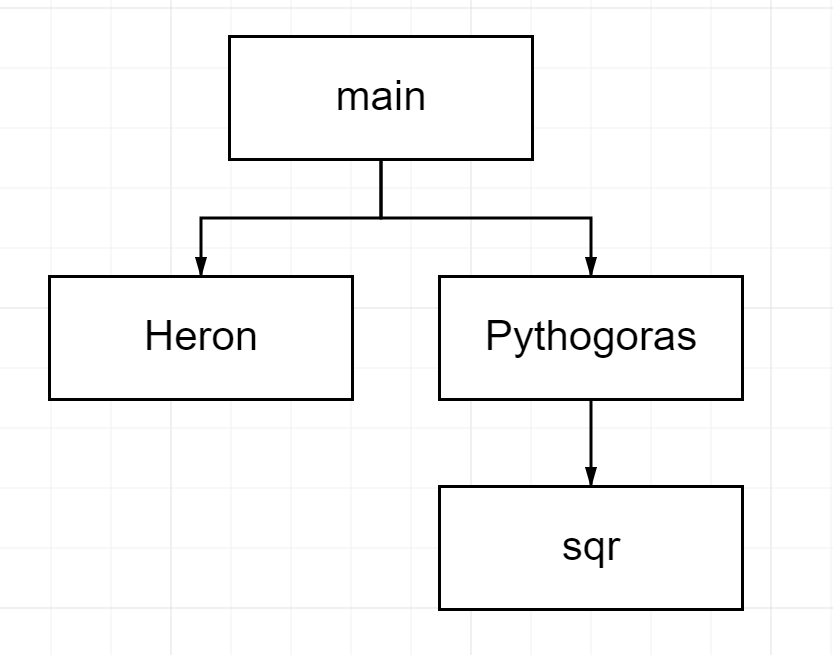


Рисунок 8 - Схема иерархии функций для решения задачи

3.6) Программный код проекта

Реализую получение данных из объектов label, а затем произведу вычисления

**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();

}

Теперь реализую геометрические функции

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

{

**return** arg \* arg;

}

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

{

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

}

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

{

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

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

}

**4) Результаты выполнения программы.**

Результат представлен на рисунке 9. Вычисления произведены верно. Результат работы программы на тестах совпадает с результатом, полученным при вычисление на калькуляторе

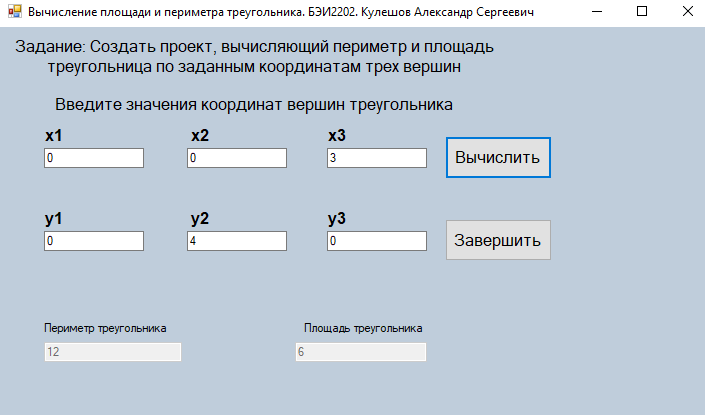


Рисунок 9 – Результат выполнения программы

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

**«Создание приложений Windows Forms для формирования и обработки автоматических 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) Индивидуальное задание на разработку проекта**

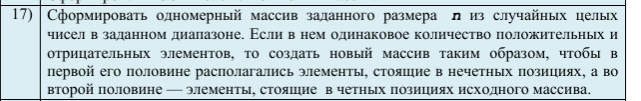


Рисунок 1 – индивидуальное задание

Разработать схему алгоритма и программный код функции task – решения задачи вычисления среднего значения и нового массива.

**2) Формализация задачи**

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.

Вызов функций формирования массивов и вывода нового массива производится в событийной процедуре по нажатию кнопки Вычислить.

**3) Разработка графического интерфейса пользователя приложения**

Графический интерфейс проекта изображен на рисунке 2.

Форма содержит текстовые поля TextBox для ввода исходных данных, списки ListBox для вывода значений массивов, а также кнопки Button для вычисления и выхода из приложения.

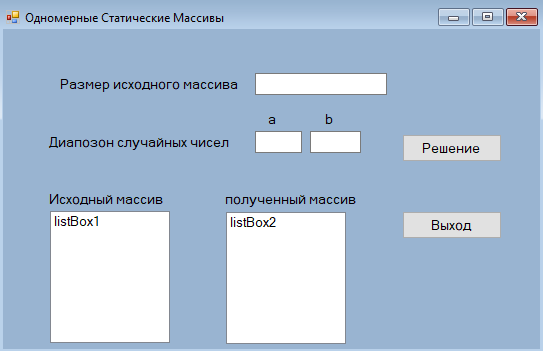


Рисунок 2 – пользовательский интерфейс

**4) Разработка схемы алгоритма функции task**

Схема алгоритма изображена на рисунке 3.

Входными параметрами функции task являются массив arr и число элементов в массиве n. Функция реализует алгоритм предложенный в индивидуальном задании

Сформированный массив new\_arr является возвращаемым значением функции

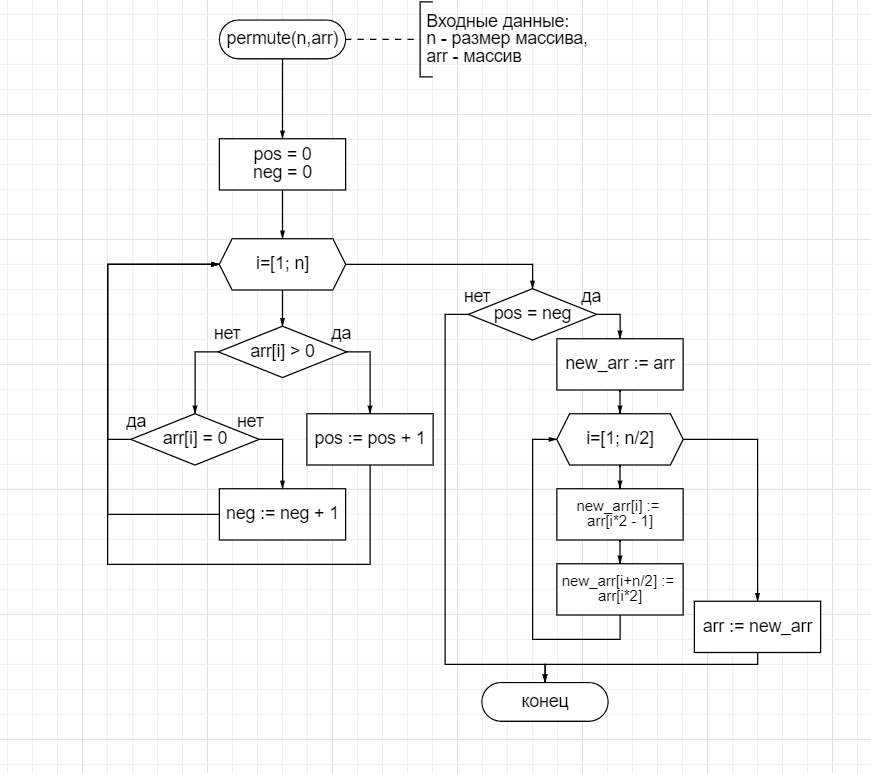


Рисунок 3 – схема алгоритма функции преобразования массива

**5) Разработка программного кода приложения**

Для начала реализую получение данных из полей label, для этого задам соответствующий код для методов Click у 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 остаётся реализовать саму функцию по преобразованию массива

**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;

}

**6) Результаты выполнения приложения**

На рисунках 4-7 приведены результаты выполнения приложения при различных исходных данных.

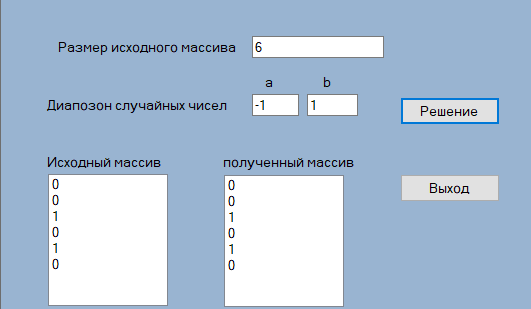


Рисунок 4 – Первый набор тестовых данных

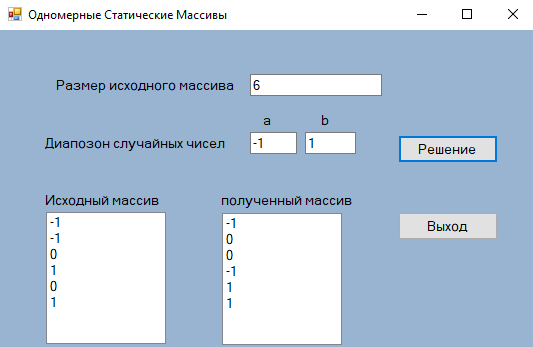


Рисунок 5 – Второй набор тестовых данных

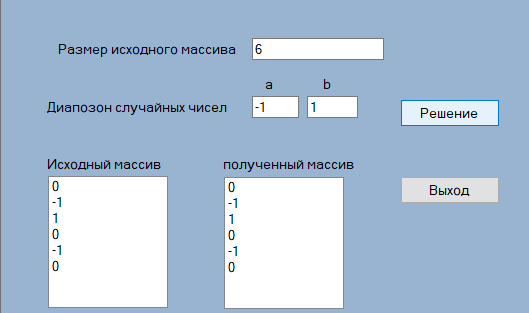


Рисунок 6 – Третий набор тестовых данных

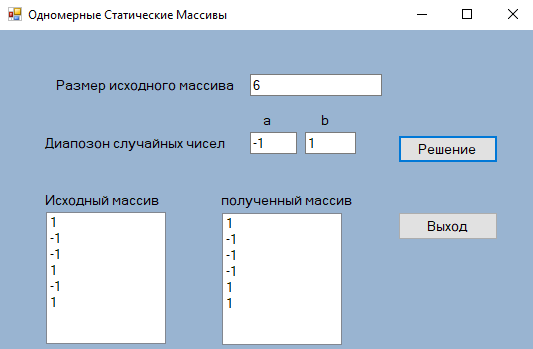


Рисунок 7 – Четвёртый набор тестовых данных

**7) Доказательство правильности результатов**

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.

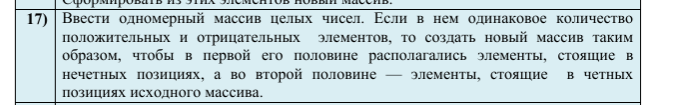
7) Подготовьте тесты для контрольного решения задачи, предусмотрев

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**

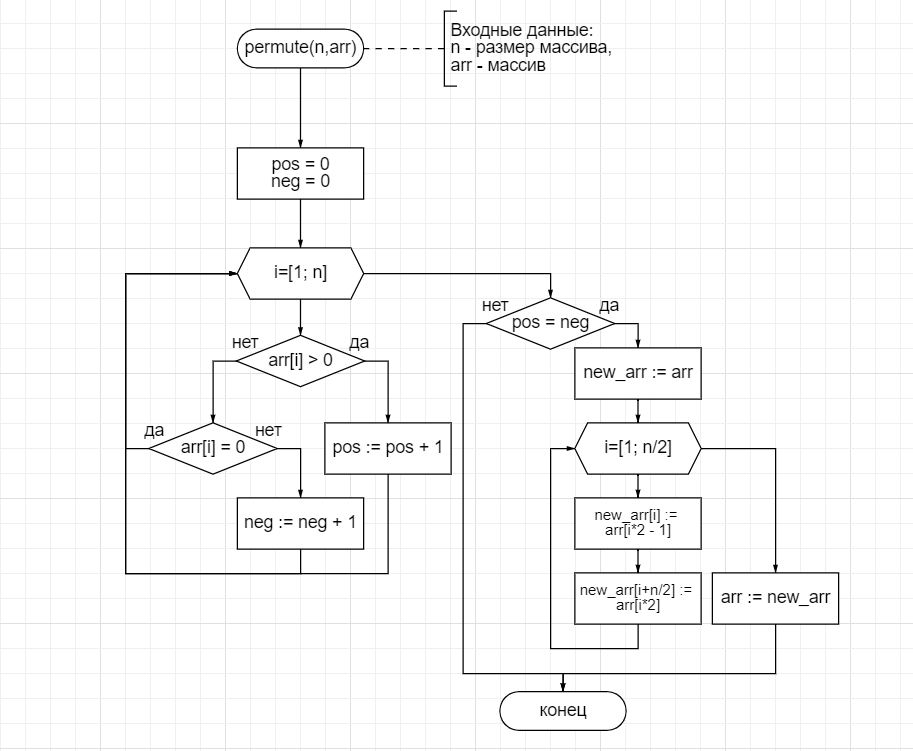


Рисунок 1 – схема алгоритма преобразования массива

1. **Разработка графического интерфейса приложения**

Графический интерфейс приложения представлен на рисунке 2

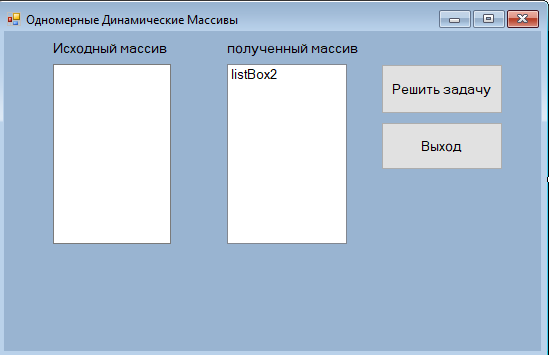


Рисунок 2 – Графический интерфейс проекта

Форма содержит текстовые поля TextBox для ввода исходных данных, список ListBox для вывода значений массивов, а также кнопки Button для вычисления и выхода из приложения.

**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;

}

1. Результаты выполнения приложения

На рисунках 3-6 представлены примеры выполнения

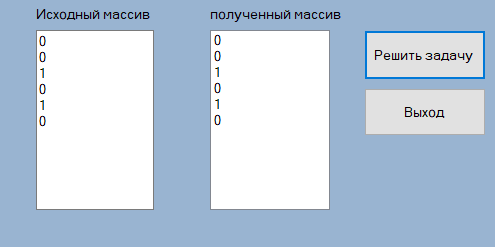


Рисунок 3 – Первый набор тестовых данных

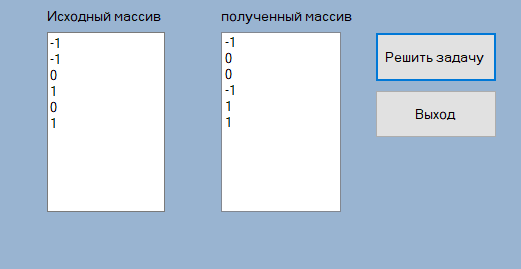


Рисунок 4 – Второй набор тестовых данных

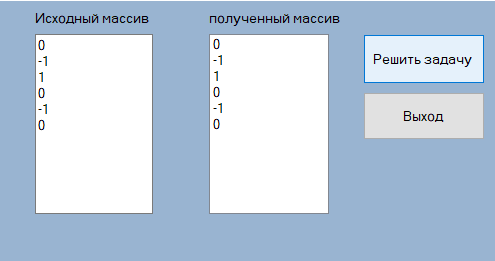


Рисунок 5 – Третий набор тестовых данных

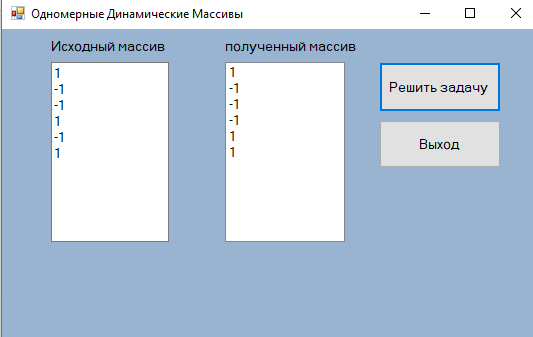


Рисунок 6 – Четвёртый набор тестовых данных

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 дисциплине**

**«Алгоритмизация и программирование»**

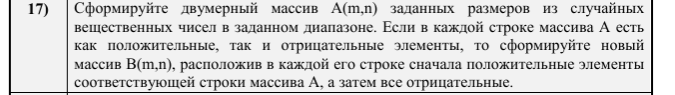
Выполнил: студент гр. БЭИ2202 Кулешов А. С.

Проверил: доц. Шакин В. Н.

Вариант: 17

Москва, 2023 г

**1) Индивидуальное задание на разработку проекта**



**2) Формализация задачи**

Из условия задания следует, что его выполнение распадается на следующие этапы, реализуемые соответствующими функциями приложения:

* формирование исходного массива из случайных вещественных чисел – прямая реализация в main;
* вывод исходного массива в заданный список формы – прямая реализация в main;
* формирование нового массива с обратными строками исходного массива – функция permute;

Вызов функции permute в производится в событийной процедуре по нажатию кнопки Вычислить.

**3) Разработка графического интерфейса пользователя приложения**

Графический интерфейс проекта изображен на рисунке 1.

Форма содержит текстовые поля TextBox для ввода исходных данных, списки ListBox для вывода значений массивов, а также кнопки Button для вычисления и выхода из приложения.

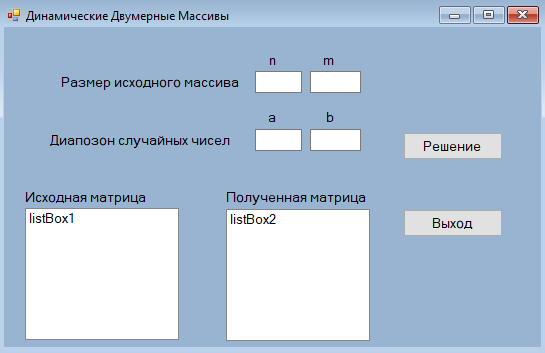


Рисунок 1 – Графический интерфейс приложения

**4) Разработка схем алгоритмов функций**

Схема алгоритма функции permute представлена на рисунке 2.

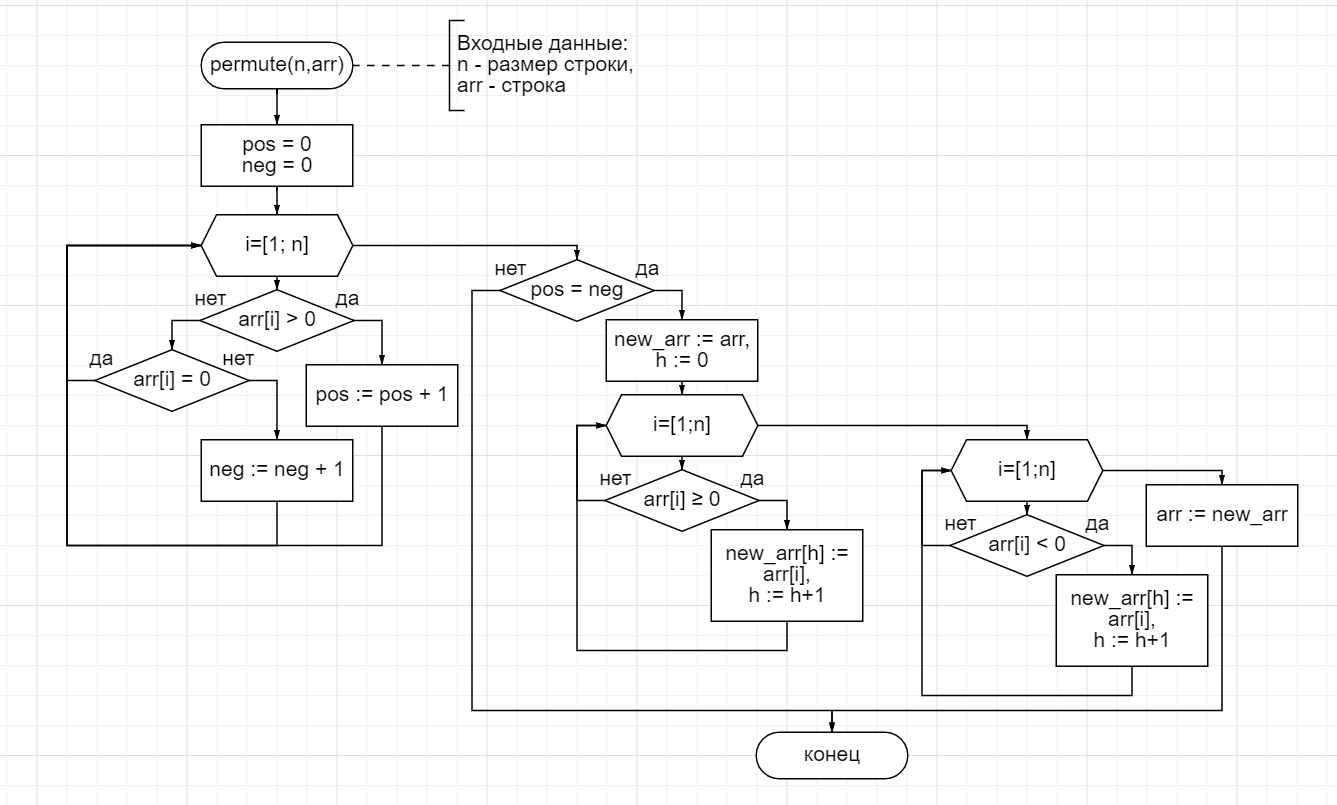


Рисунок 2 – схема алгоритма преобразования массива

**5) Разработка программного кода приложения.**

В данном проекте информацию с полей генератора случайных чисел я соберу во время события нажатия на кнопку, затем вызову функцию permute внутри события.

**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();

}

Наконец, реализую саму функцию permute, которая преобразует массив данных в новый, согласно условиям индивидуального задания

**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;

}

**6) Результаты выполнения приложения**

На рисунках 3-6 представлены результаты выполнения приложения при различных исходных данных.

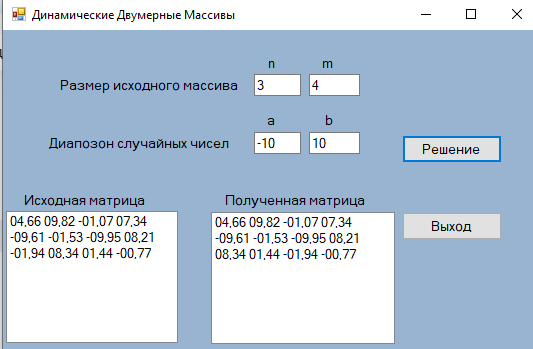


Рисунок 3 – Первый набор тестовых данных

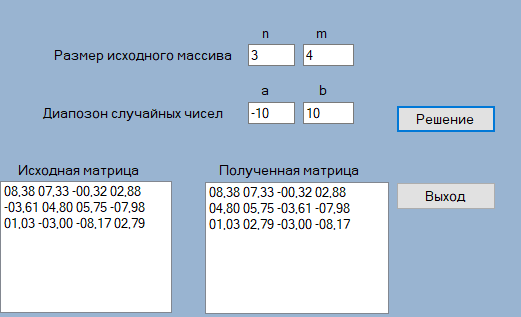


Рисунок 4 – Второй набор тестовых данных

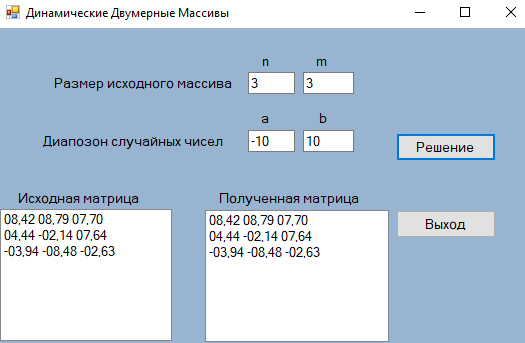


Рисунок 5 – Третий набор тестовых данных

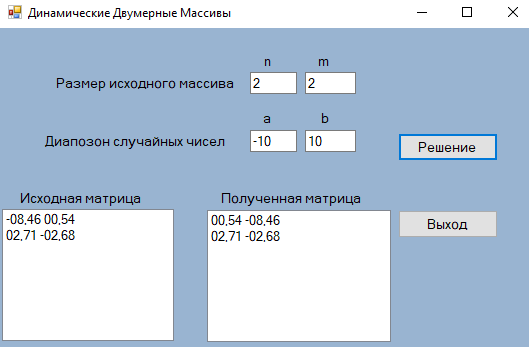


Рисунок 6 – Четвёртый набор тестовых данных

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 на разработку проекта**



**2) Формализация задачи**

Из условия задания следует, что его выполнение распадается на следующие этапы, реализуемые соответствующими функциями приложения:

* Ввод исходной строки;
* Удаление повторяющихся символов из исходной строки;
* Вывод изменённой строки.

**3) Разработка графического интерфейса пользователя приложения**

Графический интерфейс проекта изображен на рисунке 1

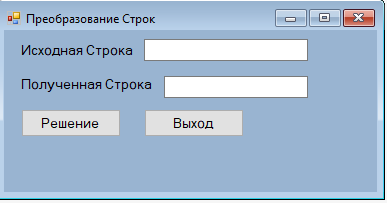


Рисунок 1 – Графический интерфейс приложения

**4) Разработка схем алгоритмов функций**

Схема алгоритма функции permute, решающая поставленную задачу, представлена на рисунке 2.

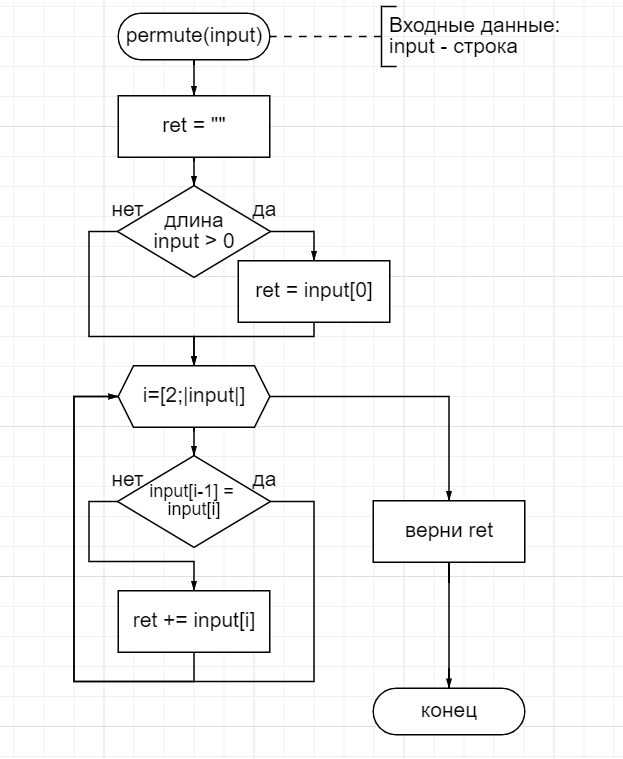


Рисунок 2 – схема алгоритма преобразования строки

**5) Разработка программного кода приложения**

В данном проекте событие нажатия на кнопку лишь передаёт данные из поля textBox1 в функцию permute, а затем результат функции в поле text объекта textBox2. Реализую передачу этих данных, также не забуду о выходе из приложения.

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

{

textBox2.Text = permute(textBox1.Text);

}

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

{

Application.Exit();

}

Наконец, остаётся лишь реализовать функцию для удаления повторяющихся символов.

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;

}

**6) Результаты выполнения приложения**

На рисунках 3-6 приведены результаты работы приложения для разных случаев исходных строк, как можно заметить дубликаты символов действительно удаляются. Программа работает корректно на тестовых наборах данных.

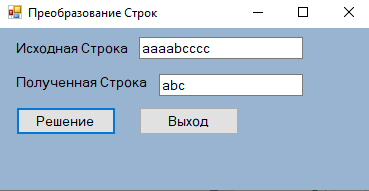


Рисунок 3 – Первый набор тестовых данных

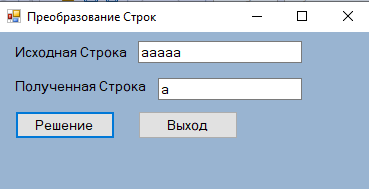


Рисунок 4 – Второй набор тестовых данных

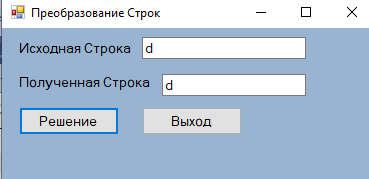


Рисунок 5 – Третий набор тестовых данных

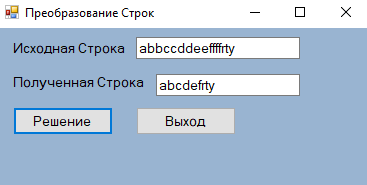


Рисунок 6 – Четвёртый набор тестовых данных

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 №13

**«Создание приложений Windows Forms**

**для работы с потоковыми файлами»**

**по теме**

**«Создание приложений, работающих с потоковыми**

**файлами»**

**по дисциплине**

**«Алгоритмизация и программирование»**

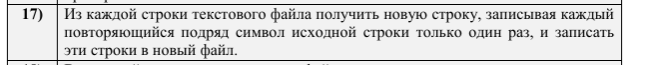
Выполнил: студент гр. БЭИ2202 Кулешов А. С.

Проверил: доц. Шакин В. Н.

Вариант: 17

Москва, 2023 г

**1) Индивидуальное задание на разработку проекта**



**2) Формализация задачи**

Из условия задания следует, что его выполнение распадается на следующие этапы, реализуемые соответствующими функциями приложения:

* открытие входного файла для чтения строк;
* открытие выходного файла для записи строк;
* чтение из входного файла очередной строки;
* вывод очередной строки в соответствующий объект формы;
* обработка очередной строки с использованием функции, разработанной в предыдущей лабораторной работе;
* вывод результата обработки в выходной файл;
* вывод результата обработки в соответствующий объект формы.

**3) Разработка графического интерфейса пользователя приложения**

Графический интерфейс проекта изображен на рисунке 1.

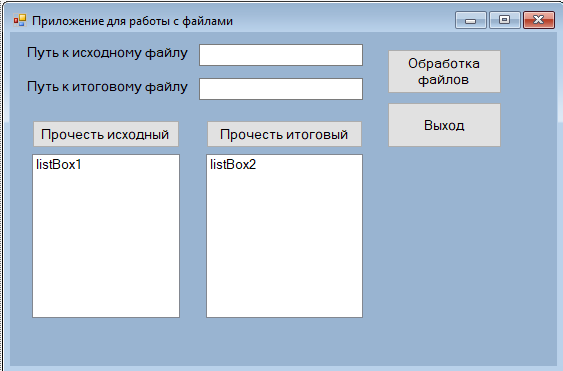


Рисунок 1 – Графический интерфейс приложения

**4) Разработка схем алгоритмов функций**

Схема алгоритма функции permute, решающая поставленную задачу, представлена на рисунке 2.

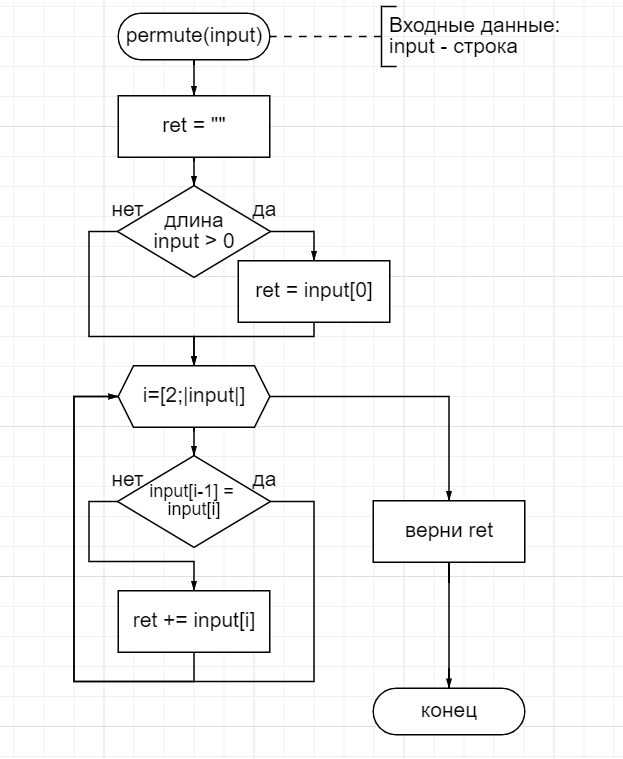


Рисунок 2 – схема алгоритма преобразования строки

**5) Разработка программного кода приложения**

В данном проекте в соответствии с заданием необходимо считать информацию из файла – для этого использую соответсвующие функции для построчного считывания из файла, реализую этот процесс в событие нажатия на кнопку, затем вызову функцию permute для каждой строки, а потом наполню данными TextBox и файл.

**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]);

}

}

Наконец, использую функцию из предыдущей лабораторной работы

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;

}

**6) Результаты выполнения приложения**

На рис. 3-6 приведены результаты работы приложения и файлового ввода-вывода.

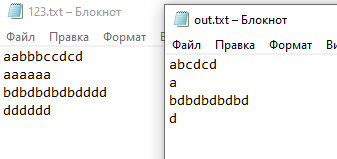


Рисунок 3 – функционал файлового ввода-вывода работает

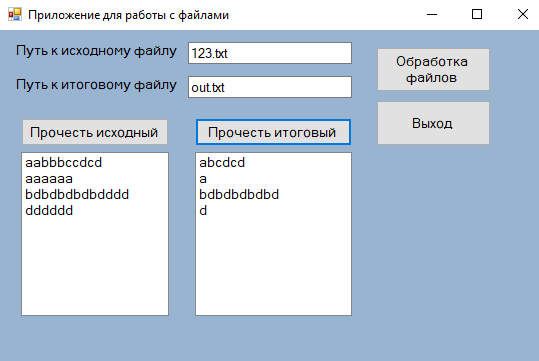


Рисунок 4 – Вывод в listBox’ы тоже работает

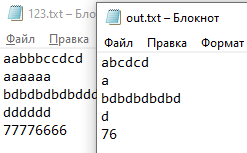


Рисунок 5 – Ещё раз отредактируем файл 123.txt

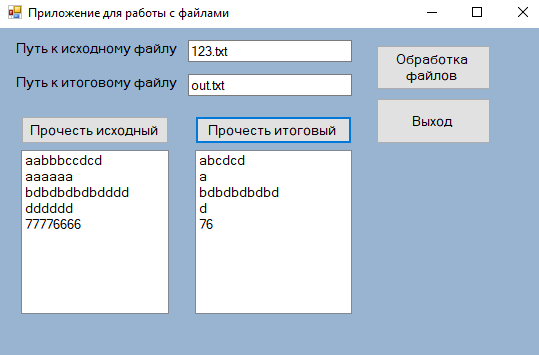


Рисунок 6 – Ещё раз проверим вывод в listBox’ы