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»**

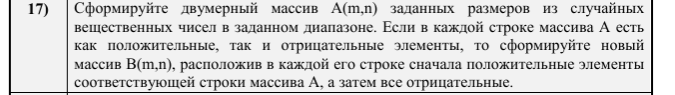
Выполнил: студент гр. БЭИ2202 Кулешов А. С.

Проверил: доц. Шакин В. Н.

Вариант: 17

Москва, 2023 г

1. Индивидуальное задание



1. Схема алгоритма

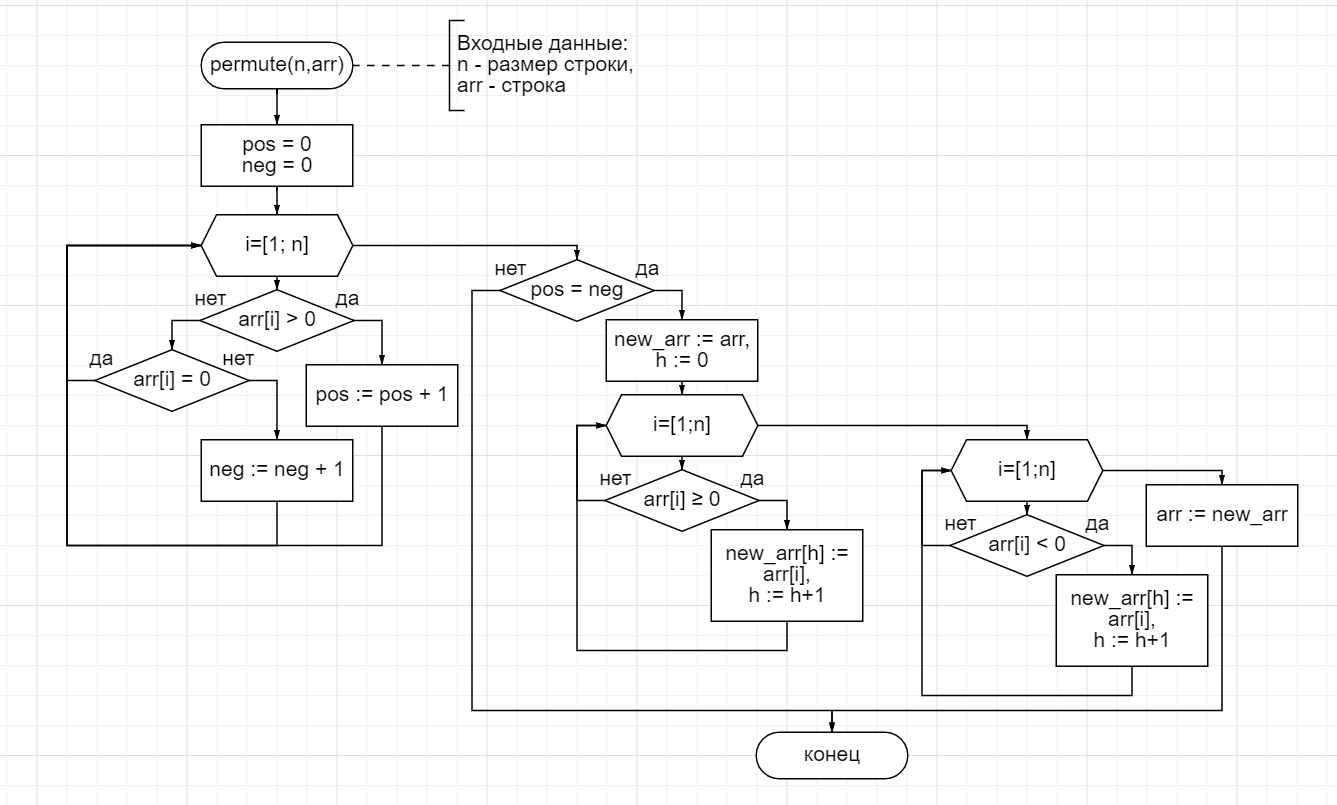


Рисунок 1 – схема алгоритма преобразования массива

1. Теперь создам форму

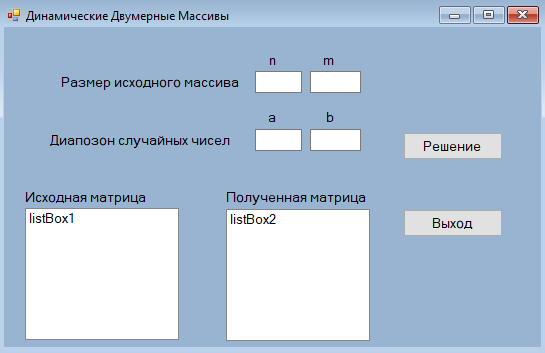


Рисунок 2 – полученная форма

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

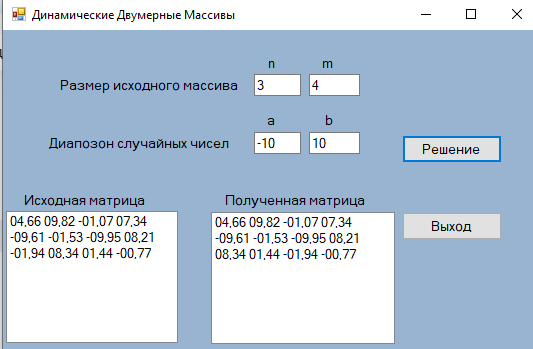


Рисунок 3 – Первый набор тестовых данных

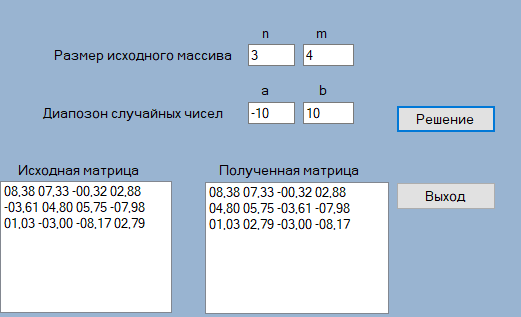


Рисунок 4 – Второй набор тестовых данных

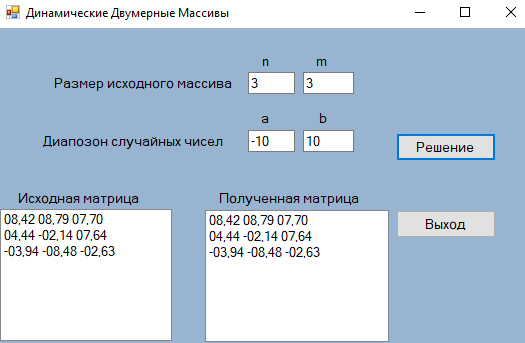


Рисунок 5 – Третий набор тестовых данных

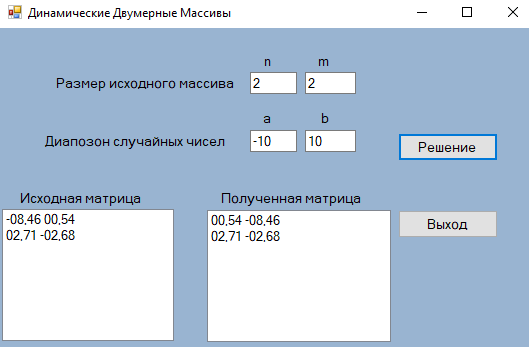


Рисунок 6 – Четвёртый набор тестовых данных