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

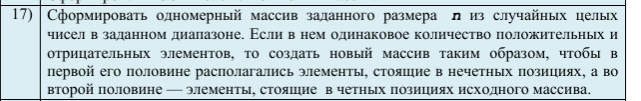
вызова пользовательских функций.

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

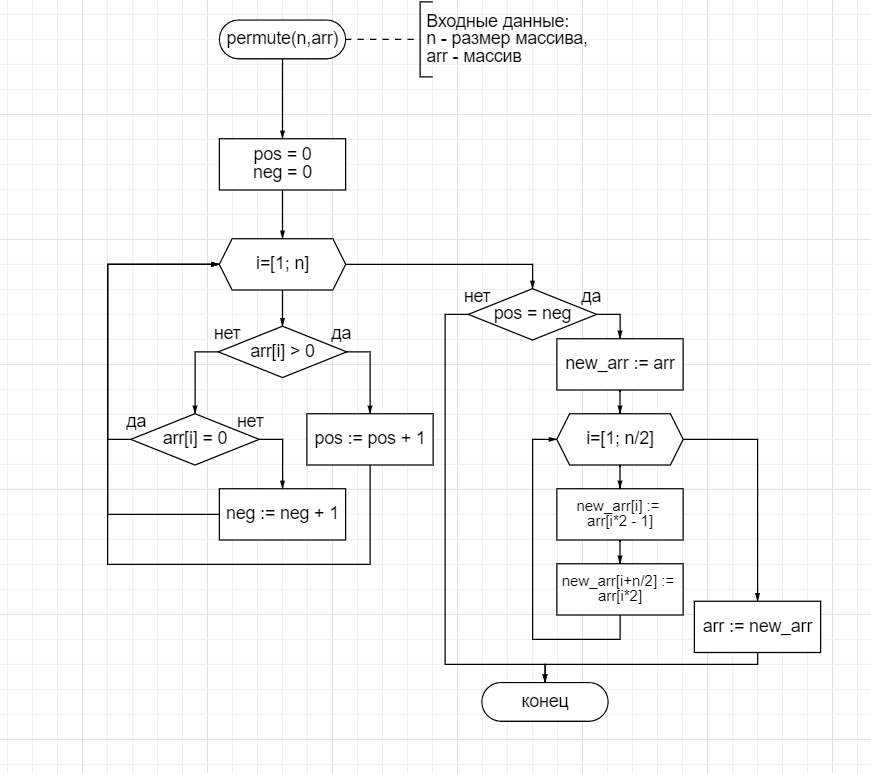


Рисунок 1 – схема алгоритма преобразования массива

1. Теперь создам форму

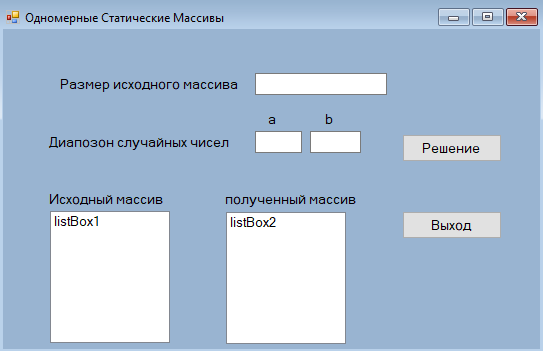


Рисунок 2 – полученная форма

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

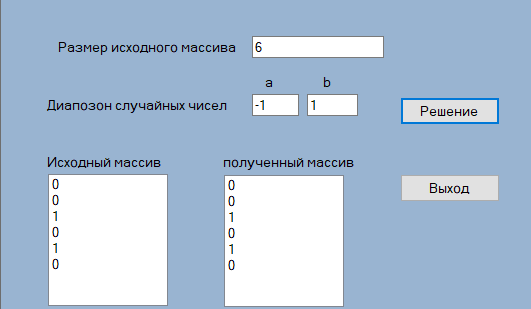


Рисунок 3 – Первый набор тестовых данных

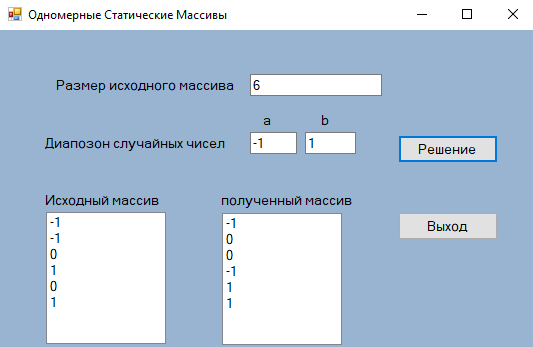


Рисунок 4 – Второй набор тестовых данных

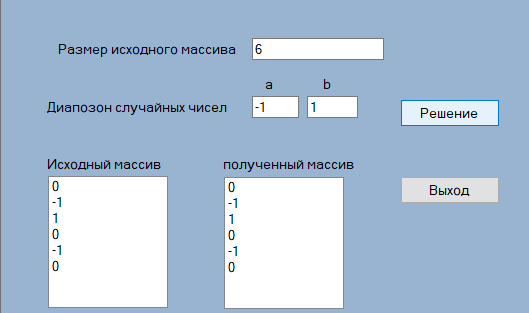


Рисунок 5 – Третий набор тестовых данных

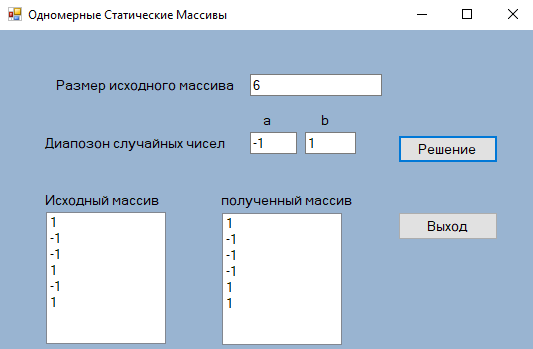


Рисунок 6 – Четвёртый набор тестовых данных

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