**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра САПР**

отчет

**по лабораторной работе №4**

**по дисциплине «Программирование»**

Тема: Использование файлов для ввода-вывода данных. Реализация программы для обработки последовательности значений с использованием динамического массива

|  |  |  |
| --- | --- | --- |
| Студент гр. 2309 |  | Савин П.А. |
| Преподаватель |  | Калмычков В.А. |

Санкт-Петербург

2022

**Формулировка задания.**

Дан массив A. Переставить в нем элементы следующим образом: поменять местами первый отрицательный элемент со средним отрицательным элементом, второй – со следующим за средним и т. д.

**Анализ задания.**

Изначально не сказано, какого рода элементы находятся в массиве, но поскольку упомянуто про отрицательные элементы, становится очевидно, что речь идет про массив чисел (целых или вещественных). Возьмем для удобства целые числа – сути задания это не меняет, просто целые проще в восприятии.

Также не оговорено, каким образом расположены нужные элементы и как их переставлять в различных случаях. Определено, что отрицательные элементы располагаются в массиве в произвольном порядке, их кол-во может быть и четным, и нечетным.

В случае четного кол-ва все элементы обменяются, в случае нечетного кол-ва последний элемент останется на своем месте т.к. у него нет пары.

**Математическая** **постановка.**

Дано:

Выполнить:

Перестановка элементов по правилу из формулировки

Решение:

1. Пробежаться по массиву один раз и посчитать кол-во отрицательных элементов
2. Определить индекс отрицательного элемента, стоящего посередине (для четного кол-ва середина не выражена, поэтому серединой считается половина кол-ва элементов плюс один)
3. Начать поэлементный пробег по массиву с первого элемента до первого попавшегося отрицательного элемента по следующей схеме:
   1. Если очередной элемент не отрицательный, бежим до первого отрицательного элемента слева и обмениваем его с уже известным элементом справа, после чего сдвигаемся с левого края на один элемент вправо и также бежим, но уже с правого края до первого попавшегося отрицательного элемента
   2. Повторять, пока кол-во обменов не будет равным целой половине от кол-ва отрицательных элементов

**Контрольный пример.**

Для контрольного примера был взят массив . Следовательно, после всех перестановок он должен принять вид

**Особенности реализации задания на компьютере.**

Сама по себе задача – лишь половина работы. Вторая половина заключается в программной стороне – все данные считываются с входного файла и в выходной файл же записываются. Более того, нужно реализовать обработку массивов, создаваемых как в статической памяти, так и в динамической.

Массивы в статической памяти имеют строго определенный заранее размер, поэтому входной файл должен иметь число, определяющее размер подмассива (т.е. в какую область исходного массива записать данные и работать с ними). Пусть для удобства это число будет стоять 1 строкой, а на 2 строке будут записываться элементы исходного набора через пробел (потому что так поток ввода считывает данные). Но тут исходный набор еще не является массивом – число в 1 строке не гарантирует запись всего набора. Собственно, возникают различные случаи:

1. Число в 1 строке не положительное – массив не может быть такой длины, следовательно, число нужно приравнять к 1 и записать в массив ровно 1 элемент
2. Число в 1 строке меньше размера исходного набора – в массив запишется ровно столько элементов, сколько требуется
3. Число в 1 строке больше размера исходного набора – в массив запишется весь набор, а справа будут нули
4. Число в 1 строке больше константного размера, определяющего массив (константный размер определяется в программе и от пользователя не зависит, но предупреждение о нем будет, если возникнет превышение) – набор элементов выйдет за пределы области памяти массива, вызывая неопределенное поведение, поэтому число нужно приравнять к этому размеру и заполнить массив

Но на этом со статической памятью не все. Размер подмассива определен, а вот сам исходный набор данных не имеет определенности: он может и быть, а может и не быть. Следовательно, если никакого набора не было предоставлено, всю область подмассива нужно заполнить нулями, равно как и если размер набора данных меньше требуемого.

С динамической памятью проще – размер заранее не определен, но его можно вычислить, прочитав файл один раз, а во второй раз записать все элементы. Обращение к динамической памяти происходит не по имени массива, а по указателю, который хранит некий адрес в памяти (в нашем случае, начало области массива). Память выделяется конструктором new, освобождается деструктором delete. Но при этом возникают проблемы с чтением файла – если набор данных пустой т.е. файл пустой, поток ввода все равно считает ноль и только потом достигнет EOF (end of file). Так что нужно этот ноль как-то отлавливать и без возникающих проблем с выделением нулевой памяти записывать в массив.

**Разработка интерфейса пользователя.**

Поскольку вся работа ведется с файлами, пользователю не предоставлен никакой интерфейс, кроме логирования в выходной файл.

**Описание используемых данных.**

|  |  |  |
| --- | --- | --- |
| Тип | Имя | Назначение |
| int | tmp, tmp\_tmp | заглушки, применяющиеся в разных частях кода |
| est\_count, real\_count, arr\_count | хранение значений для ожидаемого числа элементов, реального числа элементов и размера массива (для работы со статической памятью) |
| left\_index, right\_index, neg\_count, swaps | хранение значений для индекса с левого края, правого края, кол-ва отрицательных элементов и перестановок (для работы алгоритма) |
| A | исходный массив (статическая память) |
| p | указатель на массив (динамическая память) |
| size | константный размер массива (требуется для инициализации массива в статической памяти) |

**Организация ввода-вывода.**

|  |  |  |  |
| --- | --- | --- | --- |
| Библиотека | Имя | Команда | Назначение |
| iostream |  | setlocale | смена локализации программы на русский |
| ifstream | infile | >> | считывание с файла через поток |
| eof | функция, проверяющая, достиг ли поток конца файла (EOF) |
| clear | очистка потока, сброс установленных флагов |
| close | закрытие файла |
| open | открытие файла |
| ofstream | outfile | << | запись в файл через поток |
|  |  | close | закрытие файла |

**Представление алгоритма решения задачи.**

1. Подключение необходимых библиотек
2. Заход в main, начало программы
3. Установка русской локализации
4. Открытие входного и выходного файла
5. Входной файл открылся?
   1. Да:
      1. Объявление переменных
      2. Для статической памяти:
         1. Объявление массива
         2. Считывание ожидаемого размера массива, логирование в файл
         3. Считывание исходного набора данных, подсчет их кол-ва, логирование в файл
         4. Ожидаемый размер > 0?
            1. Да: Ожидаемый размер > константный размер?

Да: логирование в файл, приведение к случаю 4 и проверка на случаи 2 и 3

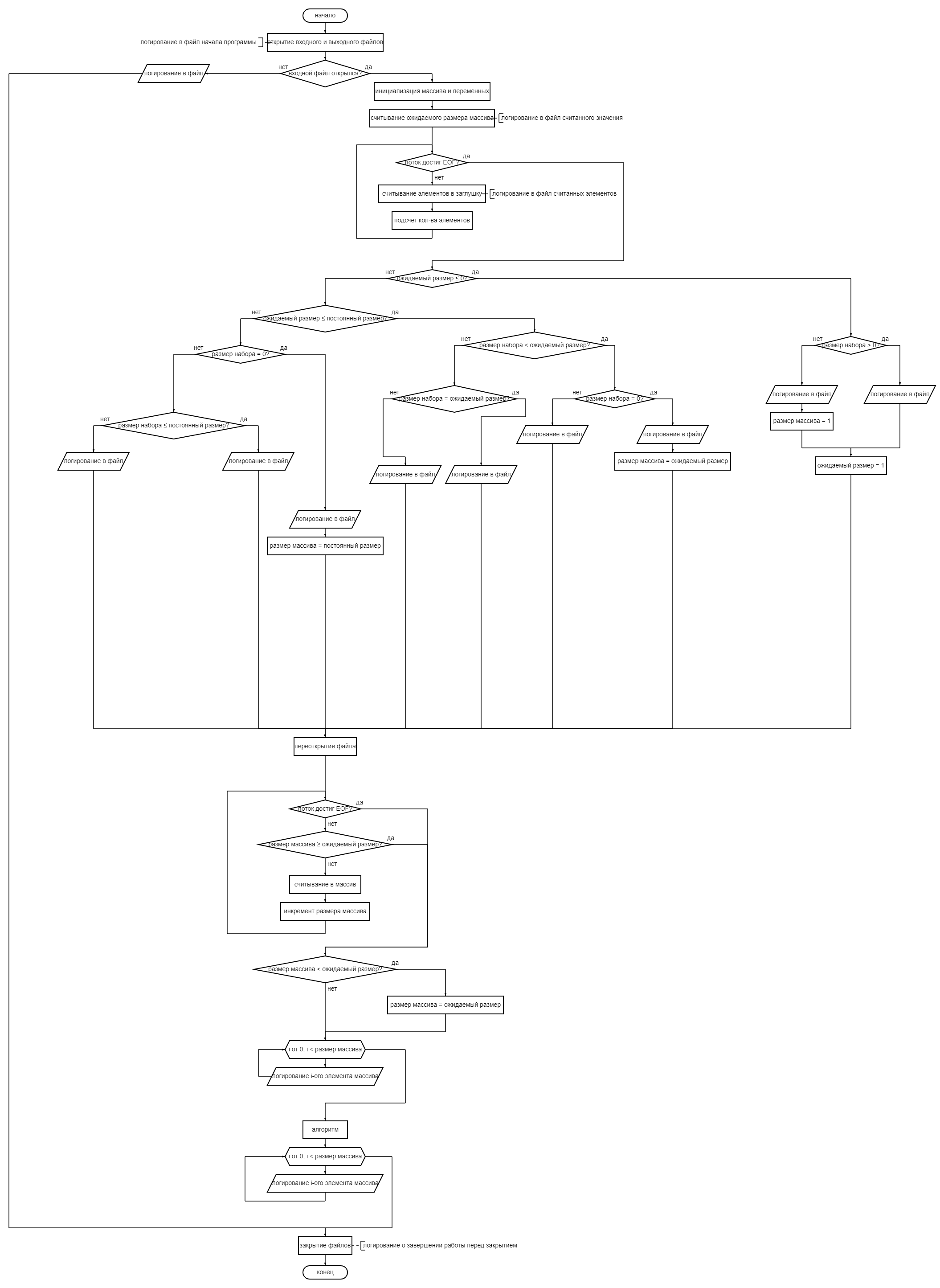
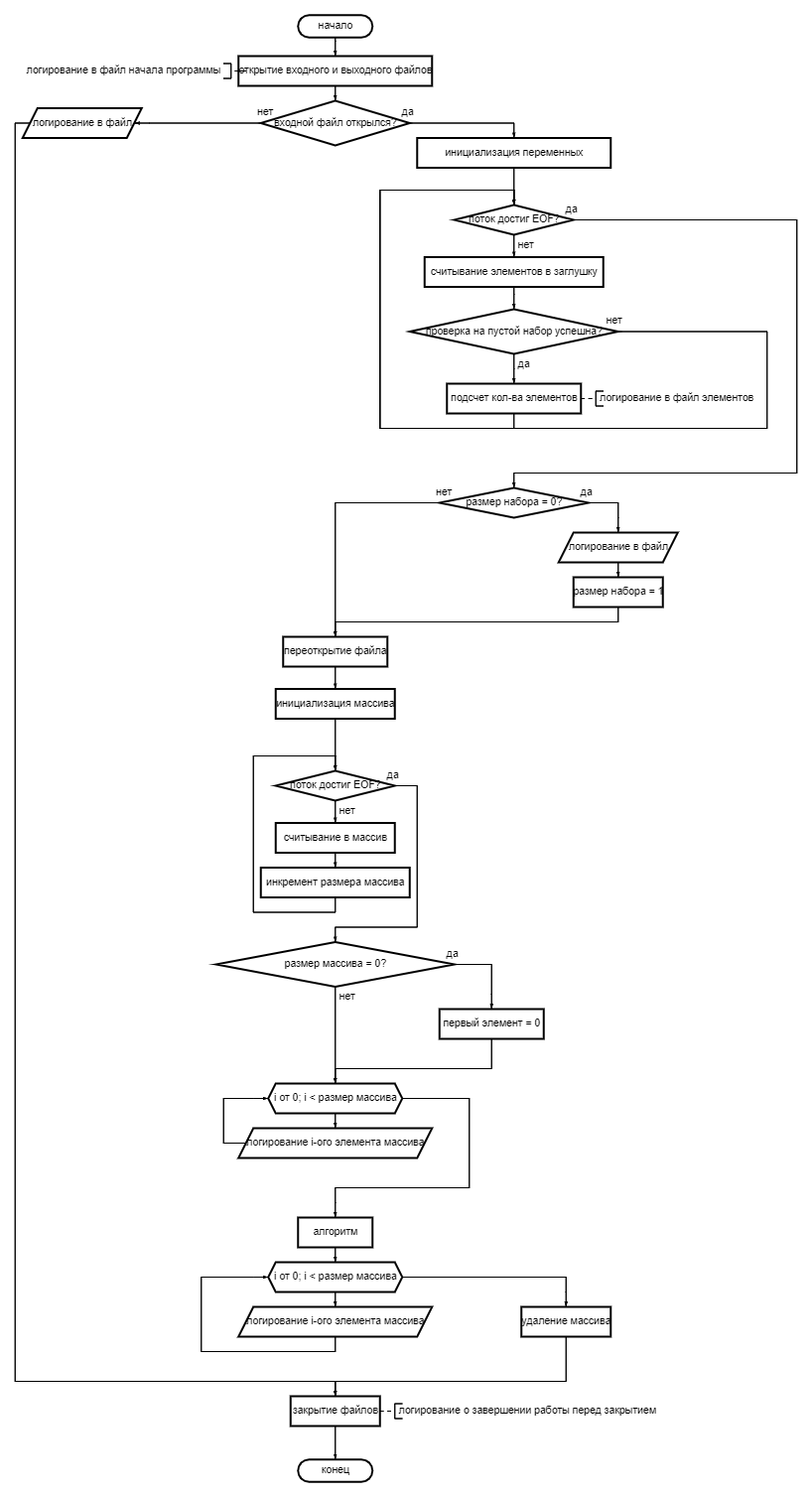
Нет: логирование в файл и проверка на случаи 2 и 3

* + - * 1. Нет: логирование в файл, приведение к случаю 1 и проверка на случаи 2 и 3
      1. Переоткрытие входного файла с очисткой потока и установленных флагов
      2. Считывание первой строки в заглушку
      3. Пока не достигнут EOF
         1. Размер массива >= ожидаемый размер?

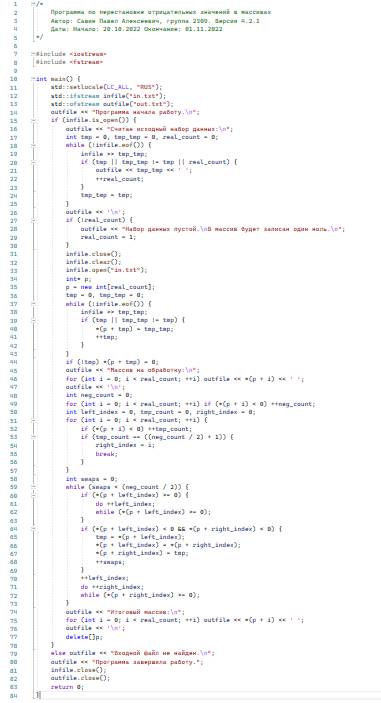
Да: прервать цикл

* + - * 1. Считывание очередного элемента исходного набора в соответствующее место в массиве, инкремент размера массива
      1. Размер массива < ожидаемый размер?
         1. Да: приравнять к ожидаемому
    1. Для динамической памяти:
       1. Считывание исходного набора данных, подсчет их кол-ва, логирование в файл
       2. Размер набора данных = 0?
          1. Да: логирование в файл, приведение к случаю 1 и проверка на случаи 2 и 3
       3. Переоткрытие входного файла с очисткой потока и установленных флагов
       4. Пока не достигнут EOF
          1. Считывание очередного элемента в заглушку
          2. Проверка на пустой набор
       5. Размер массива = 0?
          1. Да: записать в первый элемент массива 0
    2. Логирование в файл массива на обработку
    3. Выполнение алгоритма (описан в математической постановке)
    4. Логирование в файл итогового массива
    5. Для динамической памяти: освобождение выделенной под массив памяти
  1. Нет: логирование в файл, закрытие файлов, выход с main, завершение программы

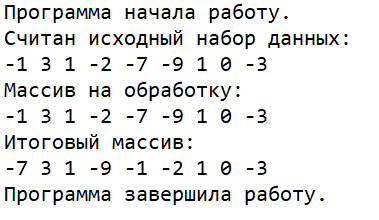
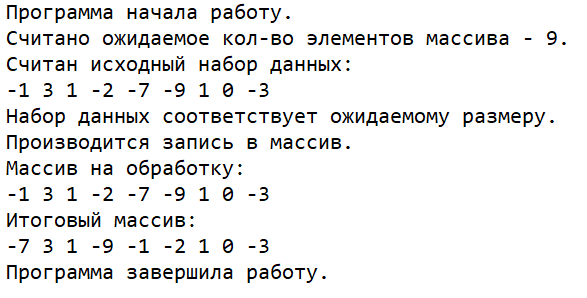
**Блок-схема.**

**** 

**Текст программы.**

****

**Результаты работы программы.**

****

**Вывод о проделанной работе.**

Изучены массивы как структура организации данных, способы их объявления и работа с ними.