**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра информационных систем**

отчет

**по практической работе №2**

**по дисциплине «Программирование»**

Тема: **«Одномерные статические массивы»**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 0324 |  | Костюхина Э.В. |
| Преподаватель |  | Глущенко А.Г |

Санкт-Петербург

2020

**Цель работы.**

Знакомство с одномерными статическими массивами, способами сортировки и бинарным поиском.

**Основные теоретические положения.**

Массив представляет собой индексированную последовательность однотипных элементов с заранее определенным количеством элементов. Наглядно одномерный массив можно представить, как набор пронумерованных ячеек, в каждой из которых содержится определенное значение.

Сортировка – процесс размещения элементов заданного множества объектов в определенном порядке. Когда элементы отсортированы, их проще найти, производить с ними различные операции. Сортировка напрямую влияет на скорость алгоритма, в котором нужно обратиться к определенному элементу массива. Большинство алгоритмов по обработке массивов реализуются с помощью циклов. Ввод и вывод массивов не являются исключением.

Алгоритм бинарного поиска – классический алгоритм поиска в отсортированном массиве, который использует дробление массива на половины. Если элемент, который необходимо найти, присутствует в списке, то бинарный поиск возвращает ту позицию, в которой он был найден.

**Постановка задачи.**

Необходимо написать программу, которая:

1) Создает целочисленный массив размерности N = 100. Элементы массивы должны принимать случайное значение в диапазоне от -99 до 99.

2) Отсортировать заданный в пункте 1 элементы массива […] сортировкой (от меньшего к большему). Определить время, затраченное на сортировку, используя библиотеку chrono.

3) Найти максимальный и минимальный элемент массива. Подсчитайте время поиска этих элементов в отсортированном массиве и неотсортированном, используя библиотеку chrono.

4) Выводит среднее значение (если необходимо, число нужно округлить) максимального и минимального значения. Выводит индексы всех элементов, которые равны этому значению, и их количество.

5) Выводит количество элементов в отсортированном массиве, которые меньше числа a, которое инициализируется пользователем.

6) Выводит количество элементов в отсортированном массиве, которые больше числа b, которое инициализируется пользователем.

7) Выводит информацию о том, есть ли введенное пользователем число в отсортированном массиве. Реализуйте алгоритм бинарного поиска. Сравните скорость его работы с обычным перебором. (\*)

8) Меняет местами элементы массива, индексы которых вводит пользователь. Выведите скорость обмена, используя библиотеку chrono.

Должна присутствовать возможность запуска каждого пункта многократно.

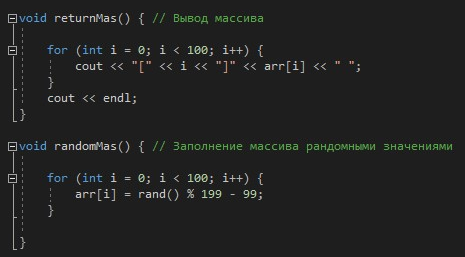
**Выполнение работы.**

Для выполнения поставленной задачи была написана программа на языке C++, которая помогла выполнить поставленную задачу. Итоговый код программы представлен ниже.

Для упрощения программы я воспользовалась функциями, написав отдельную функцию для каждого пункта задания.

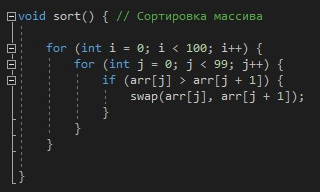
**1)** Создает целочисленный массив размерности N = 100. Элементы массивы должны принимать случайное значение в диапазоне от -99 до 99.

Для выполнения первого пункта задания я создала функцию returтMas(), которая выводит массив и функцию randomMas(), которая заполняет массив случайными значениями (arr[i] = rand() % 199 - 99;).

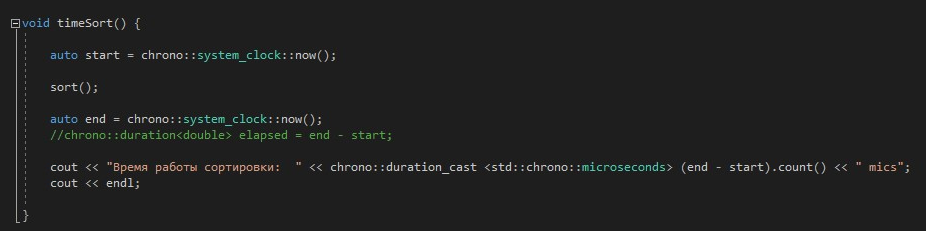


**2)** Отсортировать заданный в пункте 1 элементы массива […] сортировкой (от меньшего к большему). Определить время, затраченное на сортировку, используя библиотеку chrono.

Для выполнения второго пункта задания я создала функцию sort(), которая сортирует заданный ранее массив. Для этого я использовала самую стандартную сортировку “bubble sort”.

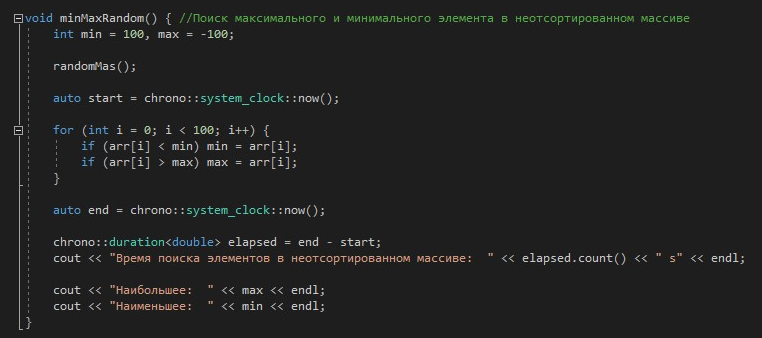


Так же функция timeSort() выводит время, которое было затрачено на сортировку в микросекундах. (используется библиотека chrono)

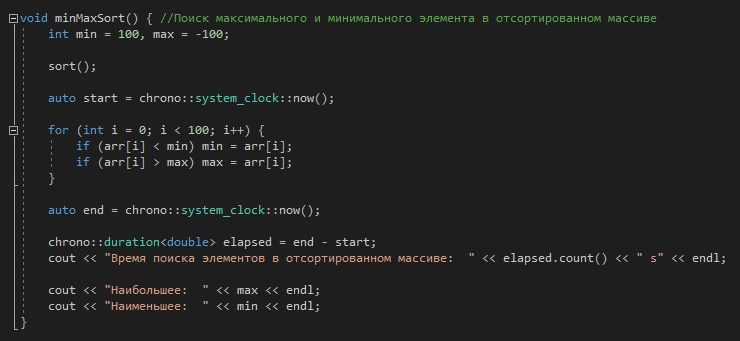


**3)**  Найти максимальный и минимальный элемент массива. Подсчитайте время поиска этих элементов в отсортированном массиве и неотсортированном, используя библиотеку chrono.

Для выполнения третьего пункта я создала функцию minMaxRandom(), которая находит максимальные и минимальный элемент в неотсортированном массиве, а также выводит время, затраченное на поиск.

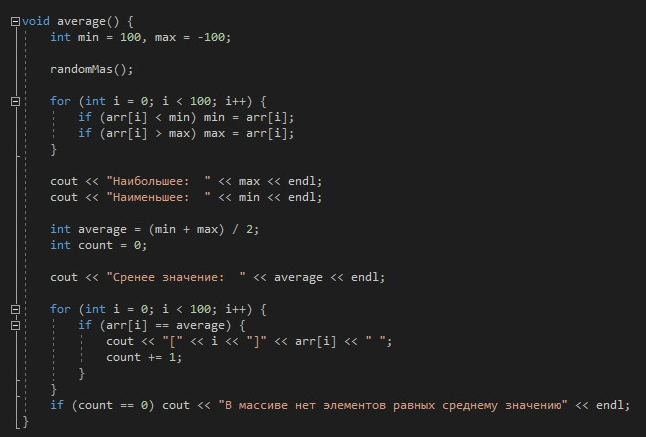


И функцию minMaxSort(), которая делает все то же самое в отсортированном массиве.



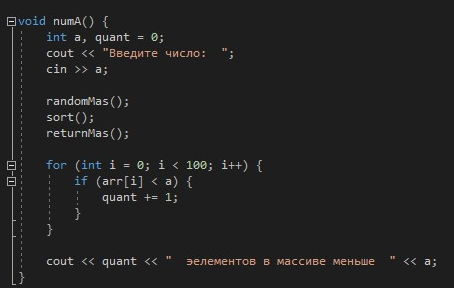
**4)** Выводит среднее значение (если необходимо, число нужно округлить) максимального и минимального значения. Выводит индексы всех элементов, которые равны этому значению, и их количество.

Для выполнения четвертого пункта была создана функция average(), которая выводит среднее значение максимального и минимального значения и индексы элементов, которые равны этому значению.



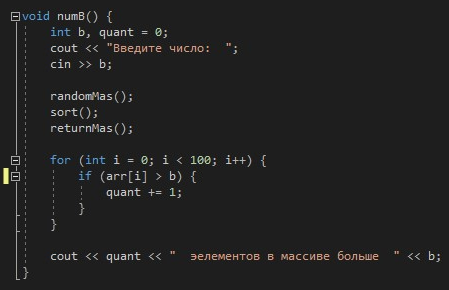
**5)** Выводит количество элементов в отсортированном массиве, которые меньше числа a, которое инициализируется пользователем.

Функция numA() выводит количество элементов, которые меньше числа, заданного пользователем.



**6)**  Выводит количество элементов в отсортированном массиве, которые больше числа b, которое инициализируется пользователем.

Функция numB() выводит количество элементов, которые больше числа, заданного пользователем.

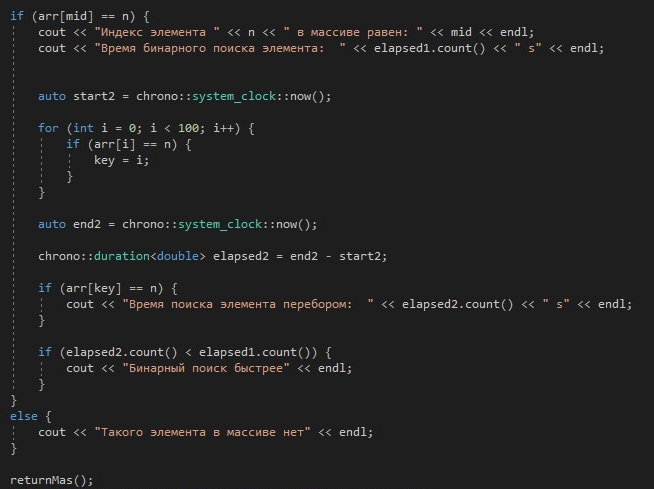


**7)** Выводит информацию о том, есть ли введенное пользователем число в отсортированном массиве. Реализуйте алгоритм бинарного поиска. Сравните скорость его работы с обычным перебором. (\*)

В первой части программы binSearch() реализуется алгоритм бинарного поиска элемента и подсчитывается время, затраченное ан поиск.

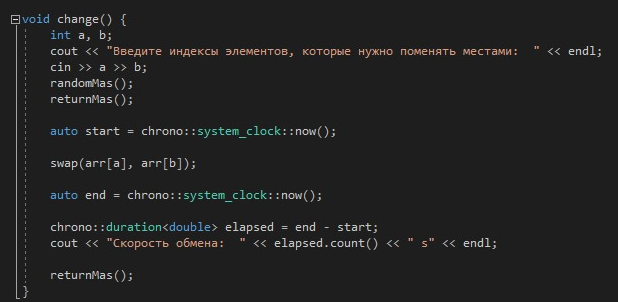


Во второй части программы реализуется поиск элемента при помощи обычного перебора, подсчитывается время и сравнивается со временем работы бинарного поиска. Если элемент был не найден в массиве, то программа говорит об этом.



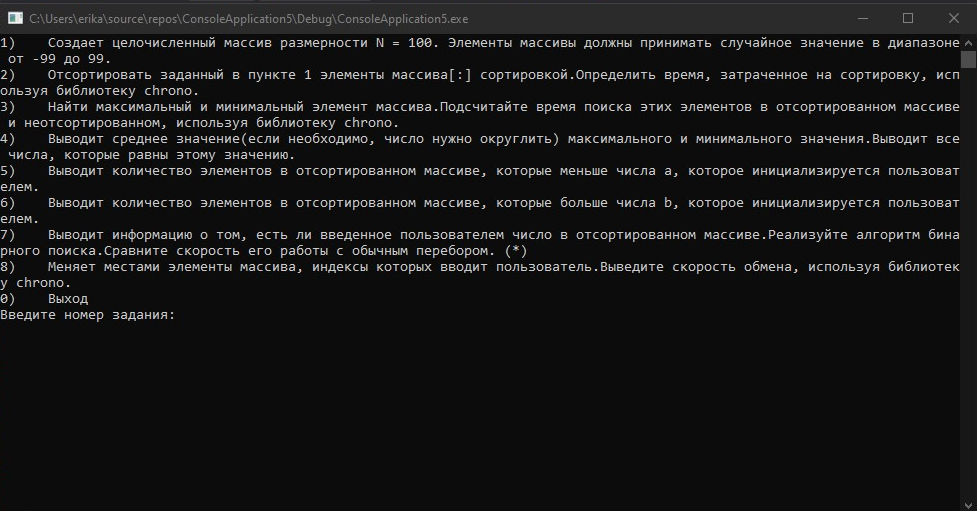
**8)** Меняет местами элементы массива, индексы которых вводит пользователь. Выведите скорость обмена, используя библиотеку chrono.

Функция change() меняет местами элементы с индексами, которые задает пользователь и подсчитывает время замены.



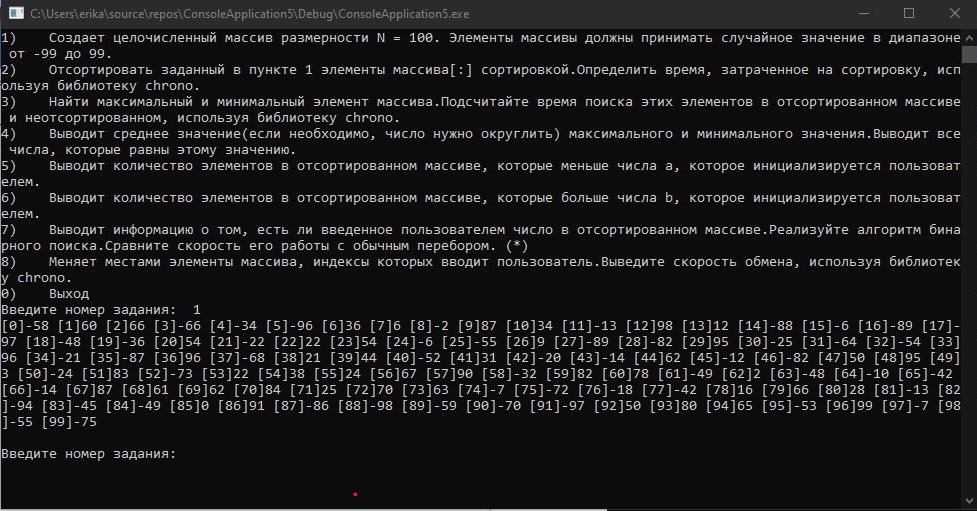
**Результаты работа программы.**

В начале в консоль выводится список заданий, которые программа может выполнить.

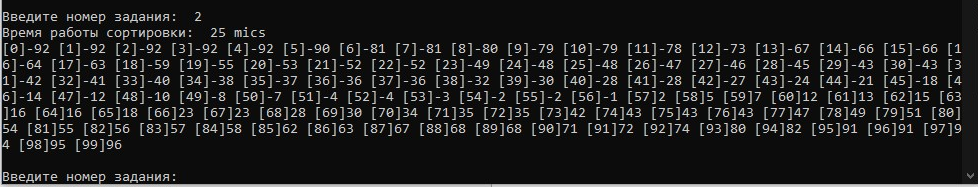


Далее нужно выбрать номер задания.

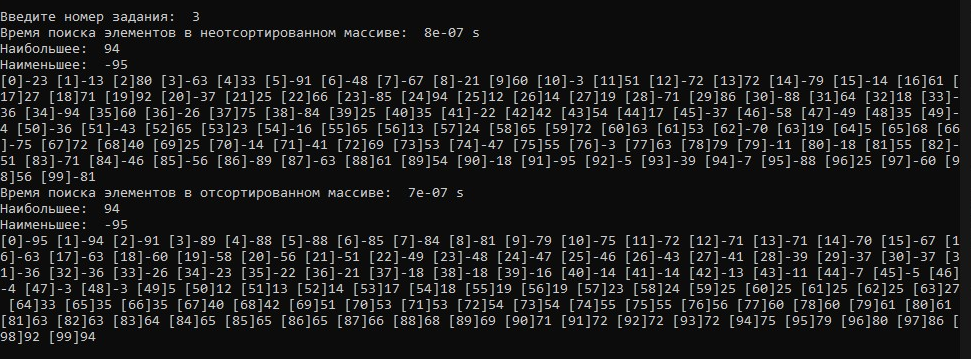
Пример работы первого задания:

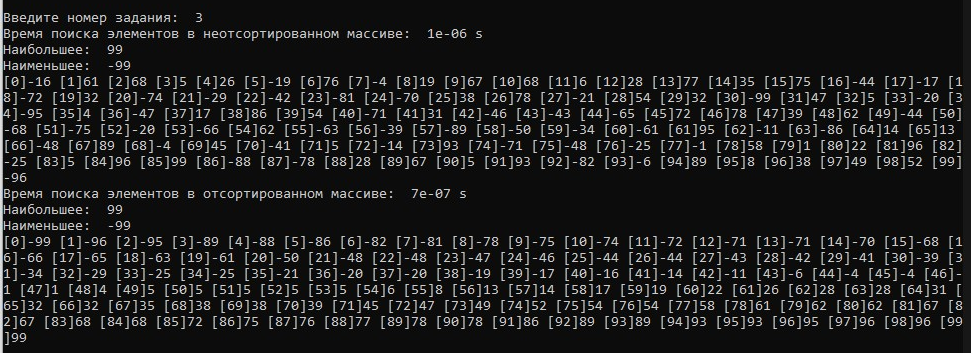


Пример работы второго задания:

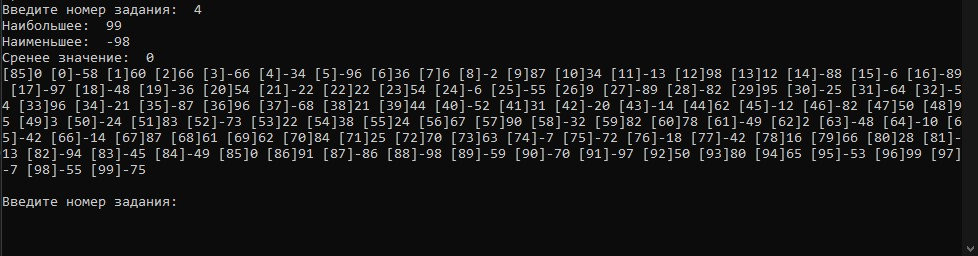


Пример работы третьего задания:

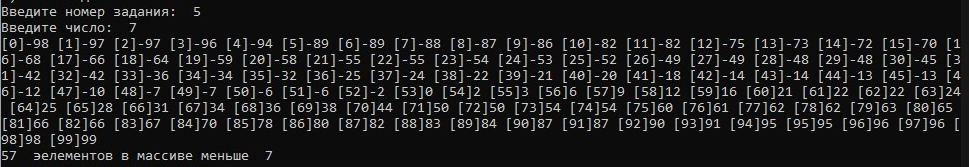




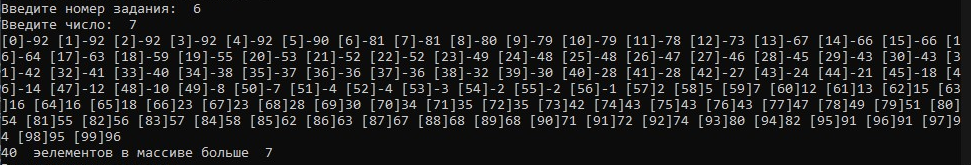
Пример работы четвертого задания:



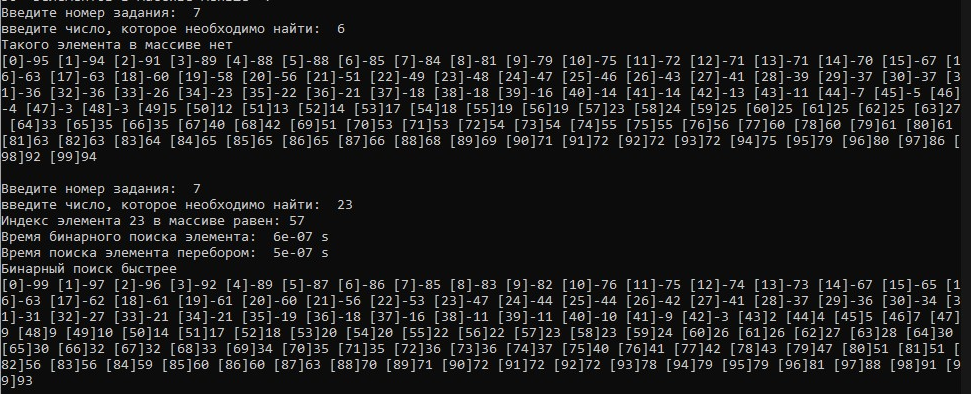
Пример работы пятого задания:



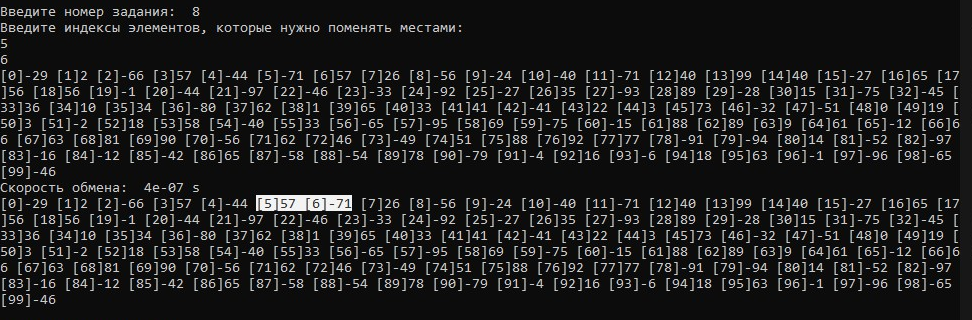
Пример работы шестого задания:



Пример работы седьмого задания:



Пример работы восьмого задания:



**Выводы.**

С одномерными статическими массивами работать легче, когда они отсортированы. Так же важно знать разные виды сортировок, которые могут пригодиться в разных ситуациях и заданиях и уметь пользоваться бинарным поиском.