**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 статистические массивы

|  |  |  |
| --- | --- | --- |
| Студент гр. 2373 | Шаймухаметов Р.Н. |  |
| Преподаватель | Глущенко А. Г. |  |

Санкт-Петербург

2022

**Цель работы.**

Изучить массивы, способы их инициализации, сортировки. Научиться работать с поиском данных внутри массива, работать с 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.

Элементы массива нумеруются с нуля. При описании массива используются те же модификаторы (класс памяти, const и инициализатор), что и для простых переменных.

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.

Простейшая из сортировок – сортировка обменом (пузырьковая сортировка). Вся суть метода заключается в попарном сравнении элементов и последующем обмене. Таким образом, если следующий элемент меньше текущего, то они меняются местами, максимальный элемент массива постепенно смещается в конец массива, а минимальный – в начало. Один полный проход по массиву может гарантировать, что в конце массива находится максимальный элемент.

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 массив […] 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. (\*)

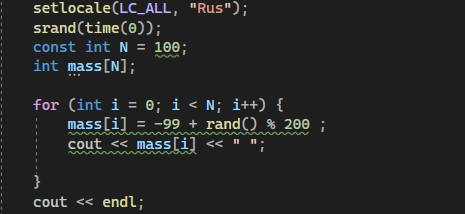
8)     Меняет местами элементы массива, индексы которых вводит пользователь. Выведите скорость обмена, используя библиотеку chrono.

**Выполнение работы.**

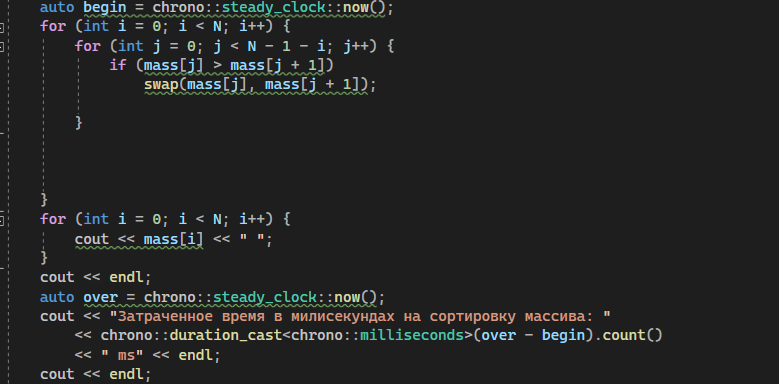
Полный код программы:

Алгоритм выполнения программы:

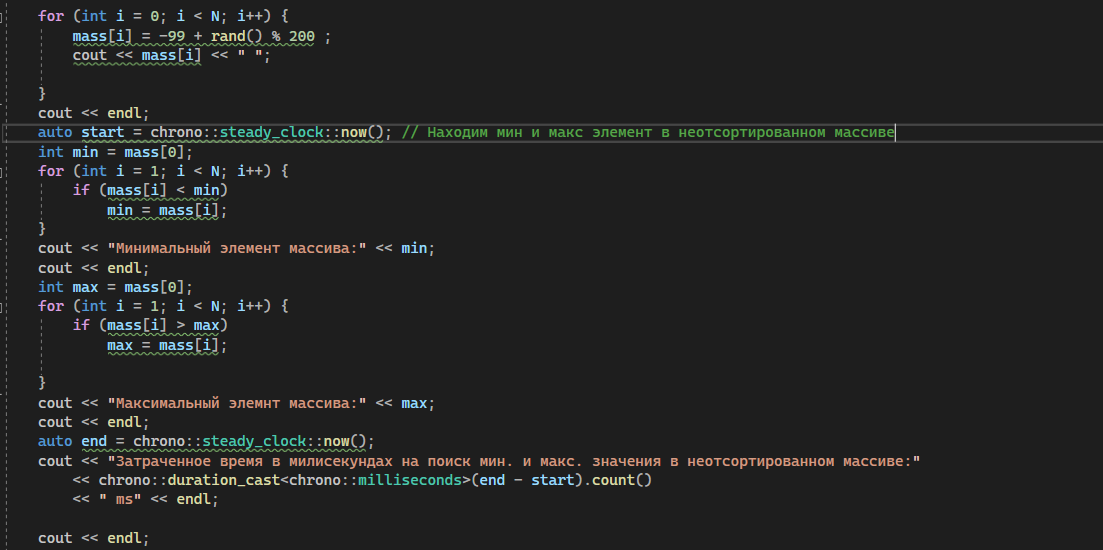
1. Создаем массив размерностью 100 со случайными значениями элементов из промежутка [-99; 99].

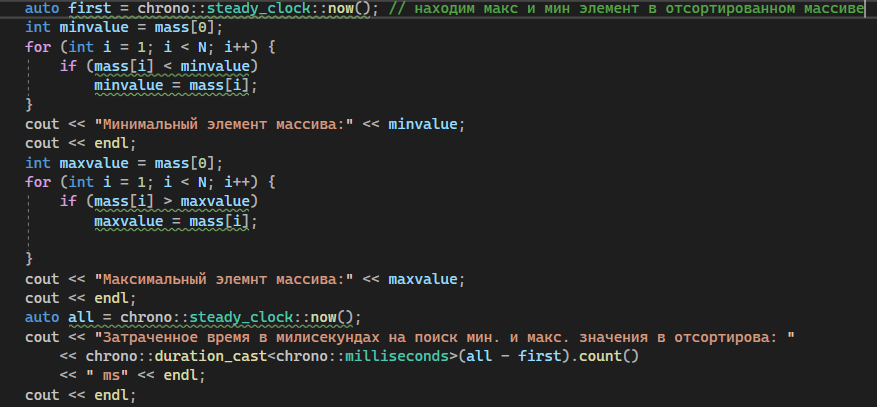


1. Сортируем массив при помощи пузырьковой сортировки и замеряем время, затраченное на сортировку. Выводим полученное время (в наносекундах) и отсортированный массив.

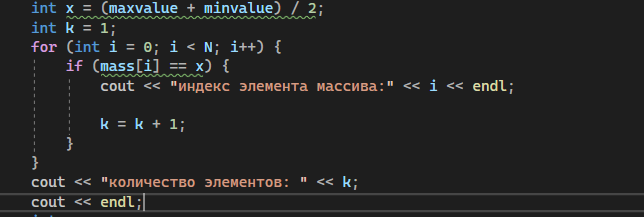


1. Находим минимальное и максимальное значение в отсортированном и неотсортированном массиве. Сравниваем затраченное время на каждый из массивов.

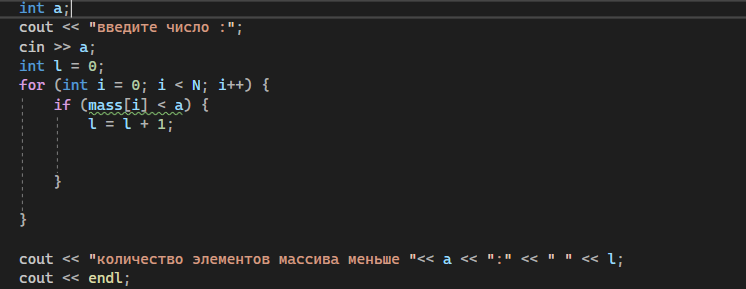




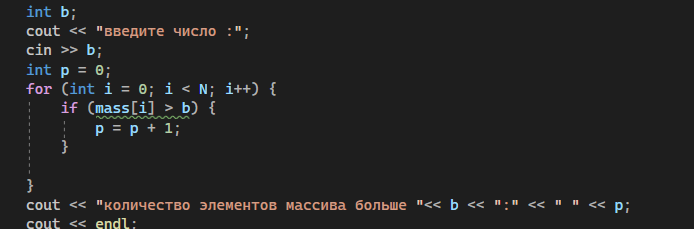
1. Находим среднее значение максимального и минимального элементов массива. Если среднее значение получилось вещественного типа, округляем его до целого в соответствии с математическими нормами. Находим индексы элементов, равных среднему значению, в отсортированном и неотсортированном массивах, и их количество. Если таких не оказалось, выводим соответствующий вердикт.



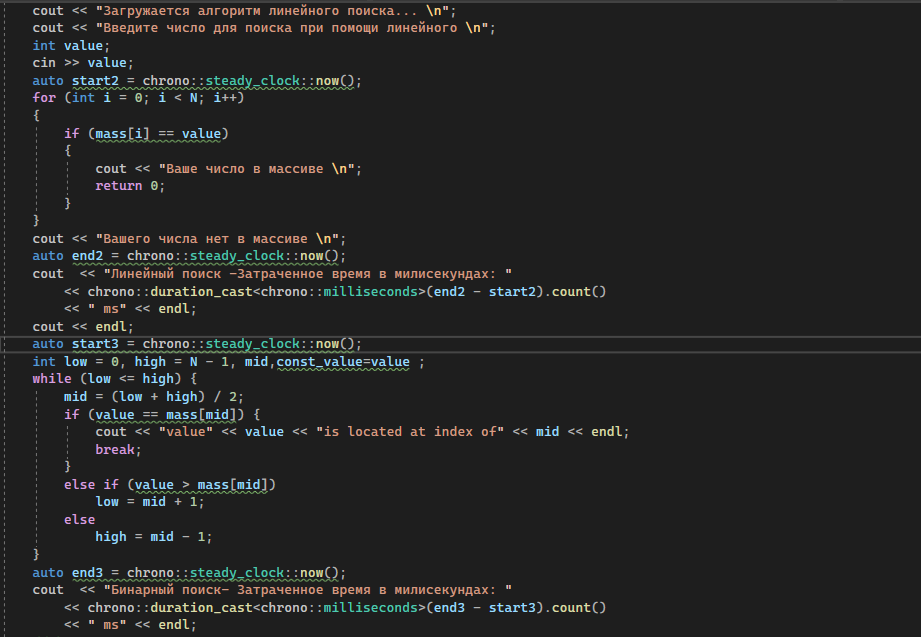
1. Выводим количество элементов, которые меньше значения введённого пользователем с клавиатуры.



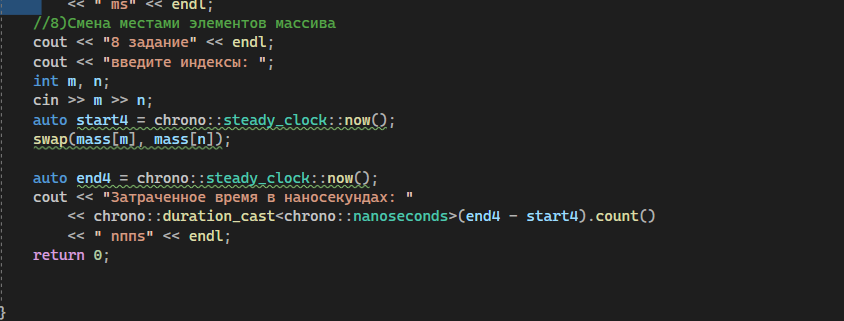
1. Выводим количество элементов, которые больше значения введённого пользователем с клавиатуры.



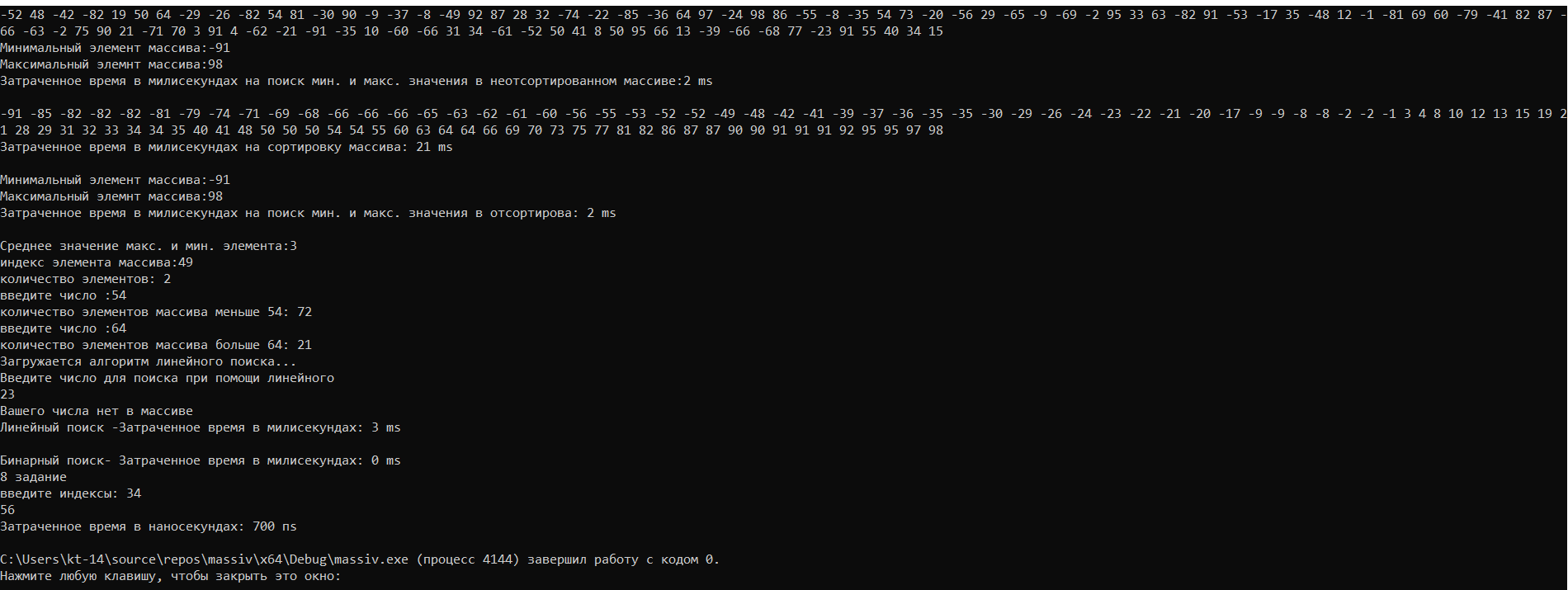
1. Ищем в массиве число, равное введённому с клавиатуры пользователем числу с помощью обычного перебора и бинарного поиска. Сравниваем затраченное время.



1. Меняем местами элементы отсортированного массива, индексы которых вводит пользователь.



Пример выполнения работы:



**Выводы.**

Я ознакомился с понятием массивов. Изучил теорию работы с ними (способы инициализации, сортировки, вывода, поиска отдельных элементов). Научился определять время, затраченное на выполнение определённого куска программы. Ознакомился с алгоритмом бинарного поиска и применил его в работе. Разные способы сортировки обладают своими преимуществами и недостатками. Главный критерий оценки примененного способа – его скорость работы. Алгоритм бинарного поиска очень удобен и практичен для массивов, в особенности больших, поскольку скорость его работы в разы меньше скорости работы обычного перебора.