**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра информационных систем (ИС)**

отчет

**по лабораторной работе №2**

**по дисциплине «Программирование»**

Тема: Одномерные статические массивы

|  |  |  |
| --- | --- | --- |
| Студентка гр. 1323 |  | Кудряшова Ю. Д. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2021

**Цель работы.**

Ознакомиться со структурой одномерного массива, обработкой данных одномерных массивов; изучить различные виды сортировок; провести временную оценку различных действий с массивами.

**Основные теоретические положения.**

Массив представляет собой индексированную последовательность однотипных элементов с заранее определенным количеством элементов.

Все массивы можно разделить на две группы: одномерные и многомерные. Описание массива в программе отличается от объявления обычной переменной наличием размерности массива, которая задается в квадратных скобках после имени.

Элементы массива нумеруются с нуля. При описании массива используются те же модификаторы (класс памяти, const и инициализатор), что и для простых переменных. Размерность массива вместе с типом его элементов определяет объем памяти, необходимый для размещения массива, которое выполняется на этапе компиляции, поэтому размерность должна быть задана целой положительной константой или константным выражением.

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.

Quick sort – одна из самых быстрых сортировок. Эта сортировка, по сути,

является существенно улучшенной версией алгоритма пузырьковой сортировки.

Общая идея алгоритма состоит в том, что сначала из массива выбирается

элемент, который называется опорным. От выбора опорного элемента не зависит корректность алгоритма, но в отдельных случаях может сильно зависеть

его эффективность. Затем необходимо сравнить все остальные элементы с

опорным и переставить их в массиве так, чтобы разбить его на три непрерывных отрезка, следующие друг за другом: меньше опорного, равный опорному

и больше опорного. Для меньших и больших значений необходимо выполнить

рекурсивно ту же последовательность операций, если длина отрезка больше

единицы.

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.

Сам алгоритм имеет следующий вид:

1. Определение значения в середине массива (или иной структуры данных). Полученное значение сравнивается с ключом (значением, которое необходимо найти).

2. Если ключ меньше значения середины, то необходимо осуществлять

поиск в первой половине элементов, иначе – во второй.

3. Поиск сводится к тому, что вновь определяется значение серединного

элемента в выбранной половине и сравнивается с ключом.

4. Процесс продолжается до тех пор, пока не будет определен элемент,

равный значению ключа, или не станет пустым интервал для поиска

**Переменные и их назначение**

Таблица 1

|  |  |  |
| --- | --- | --- |
| **Переменная** | **Тип** | **Назначение** |
| arr[] | int | Целочисленный массив |
| n | const int | Константное значение размера массива |
| size | int | Размер массива для передачи в функцию |
| option | unsigned int | Для выбора действия |
| begin, end | system\_clock::time\_point | Для замера времени |
| time | duration<double> | Для замера и вывода времени |
| max\_value, min\_value | int | Границы генерации случайных чисел |
| first, last | int | Границы обрабатываемой части массива |
| f,l | int | Вспомогательные переменные для сортировки |
| middle | int | Середина текущего участка(bin, quick) |
| max, min | int | Максимальный, минимальный элемент массива |
| average | int | Среднее значение |
| count | int | Счётчик |
| numb\_a | int | Число, элементы меньшие которого -искомые |
| numb\_b | int | Число, элементы большие которого -искомые |
| quan\_s | int | Счётчик меньших |
| quan\_b | int | Счётчик больших |
| required\_num | int | Искомое значение(bin) |
| left, right | int | Границы участка поиска |
| flag | bool | Признак наличия/отсутствия элемента |
| index\_1, index\_2 | int | Индексы элементов, которые меняют местами |

**Ход работы**

1. Реализовано начальное меню в функции start, выполняющее вызов других функций
2. Функция createArray заполняет целочисленный массив размерности N = 100, созданный в основной функции, случайными значениями в диапазоне от –99 до 99.
3. Функция quickSort сортирует заданные в п. 1 элементы массива. Определяется время, затраченное на сортировку, используется библиотека chrono
4. Функция findMaxMin находит максимальный и минимальный элементы массива. Ищется время поиска этих элементов в отсортированном массиве и неотсортированном, используется библиотека chrono.
5. Функция magicWithAverage выводит среднее значение (округляет число) максимального и минимального значения и выводит все числа, которые равны этому значению
6. Функция findSmaller выводит количество элементов в отсортированном массиве, которые меньше числа a (инициализируется пользователем)
7. Функция findBigger выводит количество элементов в отсортированном массиве, которые меньше числа b (инициализируется пользователем)
8. Функция binSearch выводит информацию о том, есть ли введенное пользователем число в отсортированном массиве. Реализован алгоритм бинарного поиска. Произведено сравнение скорости его работы со скоростью обычного перебора
9. Функция replaceElements меняет местами элементы массива, индексы которых вводит пользователь. Выводится скорость обмена.

**Результат работы программы**

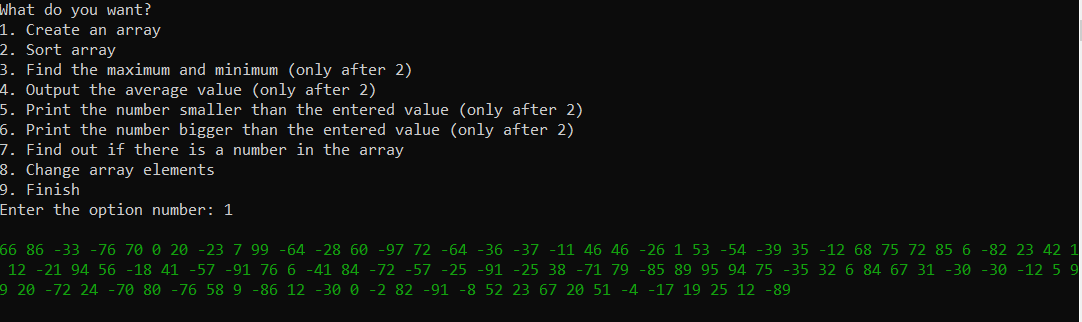


Рисунок 1

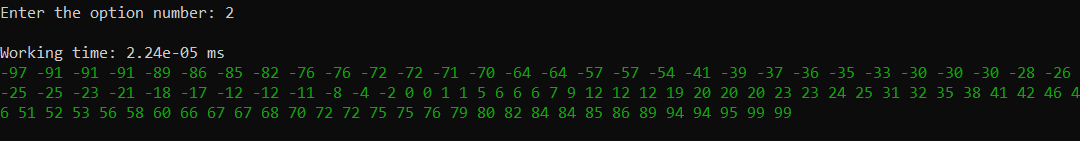


Рисунок 2

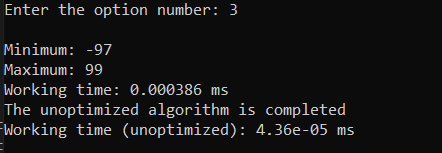


Рисунок 3

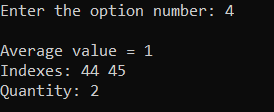


Рисунок 4

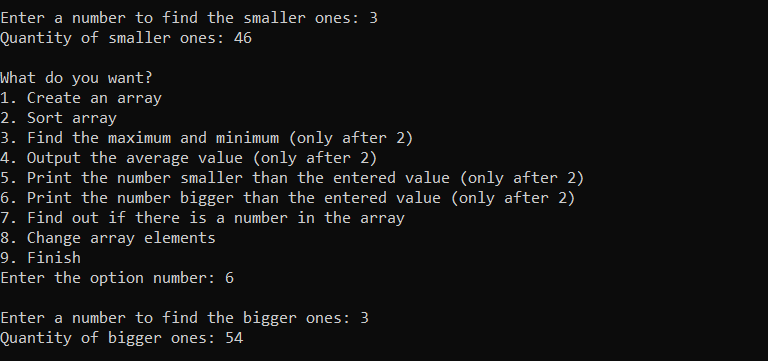


Рисунок 5

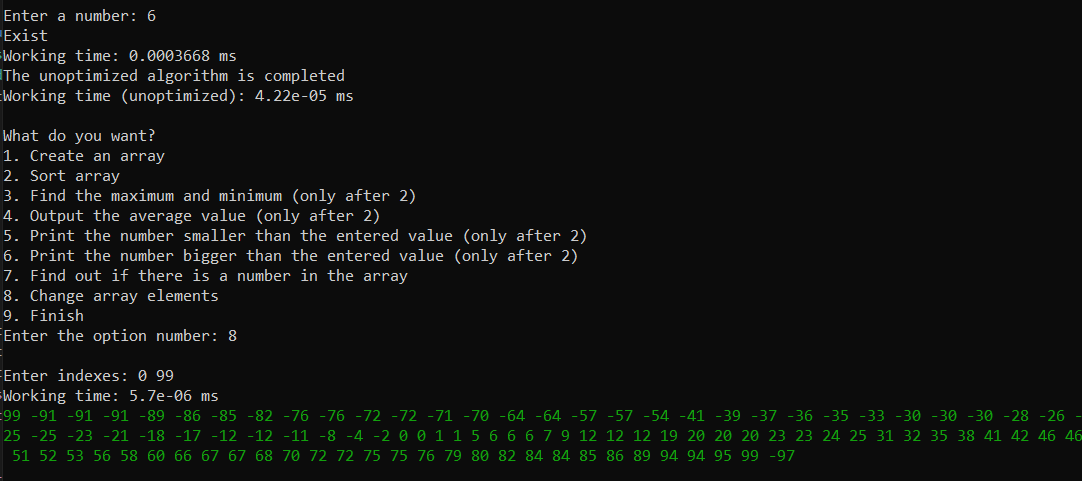


Рисунок 6

**Экспериментальные результаты.**

Таблица 2

|  |  |
| --- | --- |
| **Input** | **Output** |
| **66 86 -33 -76 70 0 20 -23 7 99 -64 -28 60 -97 72 -64 -36 -37 -11 46 46 -26 1 53 -54 -39 35 -12 68 75 72 85 6 -82 23 42 1 12 -21 94 56 -18 41 -57 -91 76 6 -41 84 -72 -57 -25 -91 -25 38 -71 79 -85 89 95 94 75 -35 32 6 84 67 31 -30 -30 -12 5 99 20 -72 24 -70 80 -76 58 9 -86 12 -30 0 -2 82 -91 -8 52 23 67 20 51 -4 -17 19 25 12 -89** | **-97 -91 -91 -91 -89 -86 -85 -82 -76 -76 -72 -72 -71 -70 -64 -64 -57 -57 -54 -41 -39 -37 -36 -35 -33 -30 -30 -30 -28 -26 -25 -25 -23 -21 -18 -17 -12 -12 -11 -8 -4 -2 0 0 1 1 5 6 6 6 7 9 12 12 12 19 20 20 20 23 23 24 25 31 32 35 38 41 42 46 46 51 52 53 56 58 60 66 67 67 68 70 72 72 75 75 76 79 80 82 84 84 85 86 89 94 94 95 99 99** |
| **9 76 68 -57 -60 -92 -49 -84 57 -45 74 -71 23 -36 -40 44 -15 18 37 48 83 -60 -26 58 5 14 26 -83 61 59 1 85 13 15 72 -81 -91 14 -38 99 15 21 -67 -90 5 71 -90 -11 55 24 -97 -18 19 21 -92 73 -58 -10 13 65 12 -17 75 67 -60 81 80 -11 66 -73 -92 99 -6 8 69 -13 -99 40 -49 41 1 19 -27 -76 55 87 -50 -40 -15 16 8 37 77 9 -30 85 -59 49 -64 58** | **-99 -97 -92 -92 -92 -91 -90 -90 -84 -83 -81 -76 -73 -71 -67 -64 -60 -60 -60 -59 -58 -57 -50 -49 -49 -45 -40 -40 -38 -36 -30 -27 -26 -18 -17 -15 -15 -13 -11 -11 -10 -6 1 1 5 5 8 8 9 9 12 13 13 14 14 15 15 16 18 19 19 21 21 23 24 26 37 37 40 41 44 48 49 55 55 57 58 58 59 61 65 66 67 68 69 71 72 73 74 75 76 77 80 81 83 85 85 87 99 99** |

**Выводы.**

Проведена работа со структурой одномерного массива, обработкой данных одномерных массивов; изучены различные виды сортировок; реализована сортировка quicksort; проведена временная оценка различных действий с массивами.