**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. 2373 |  | Маркова З.А. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2022

**Цель работы.**

Изучение одномерных статических массивов. Изучение алгоритмов сортировки и поиска.

**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.

Быстрая сортировка (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 выбирается из массива 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 массив на три непрерывных отрезка, следующие друг за другом: меньше опорного, раны 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.

**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.

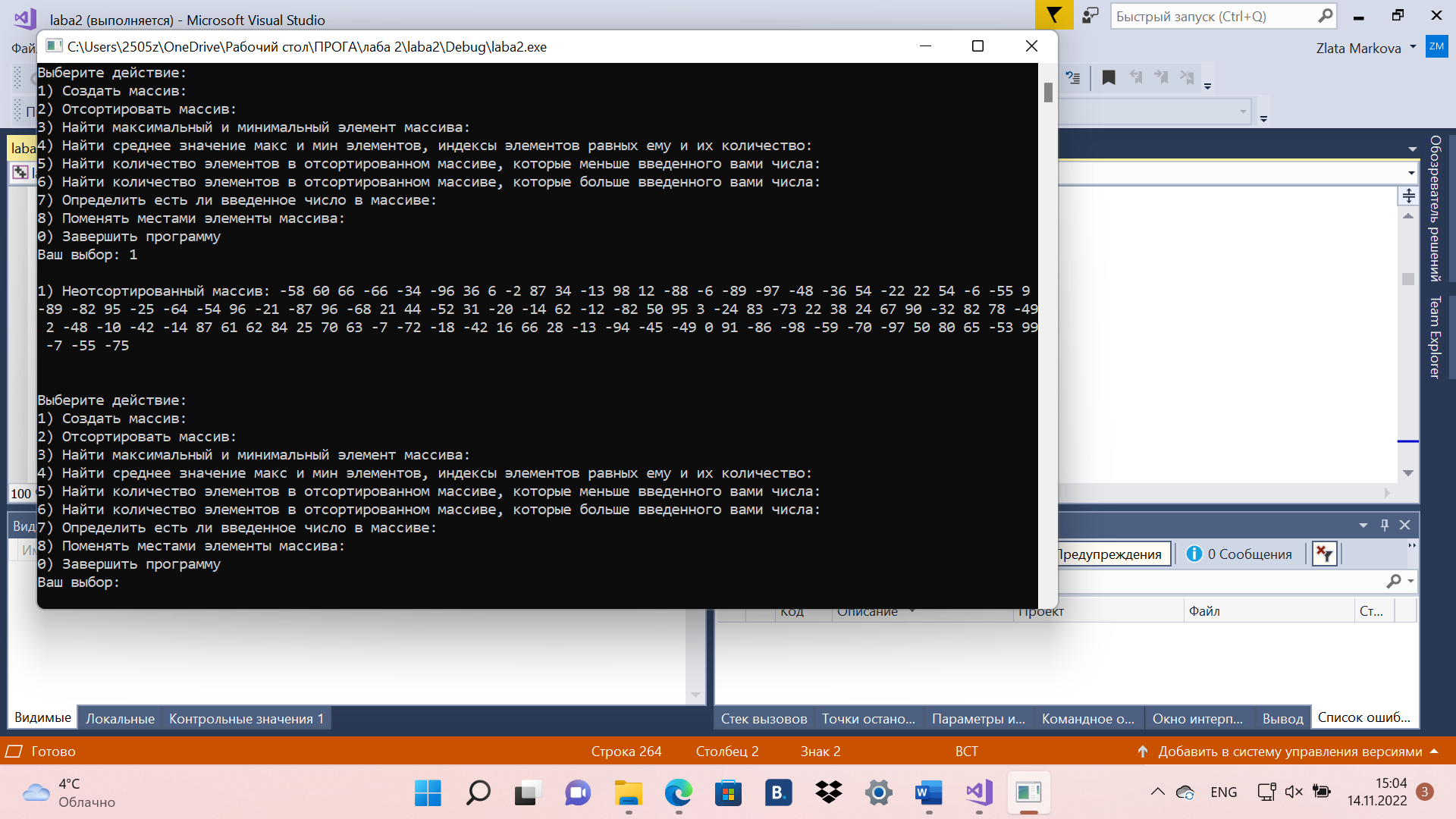
Должна присутствовать возможность запуска каждого пункта многократно.

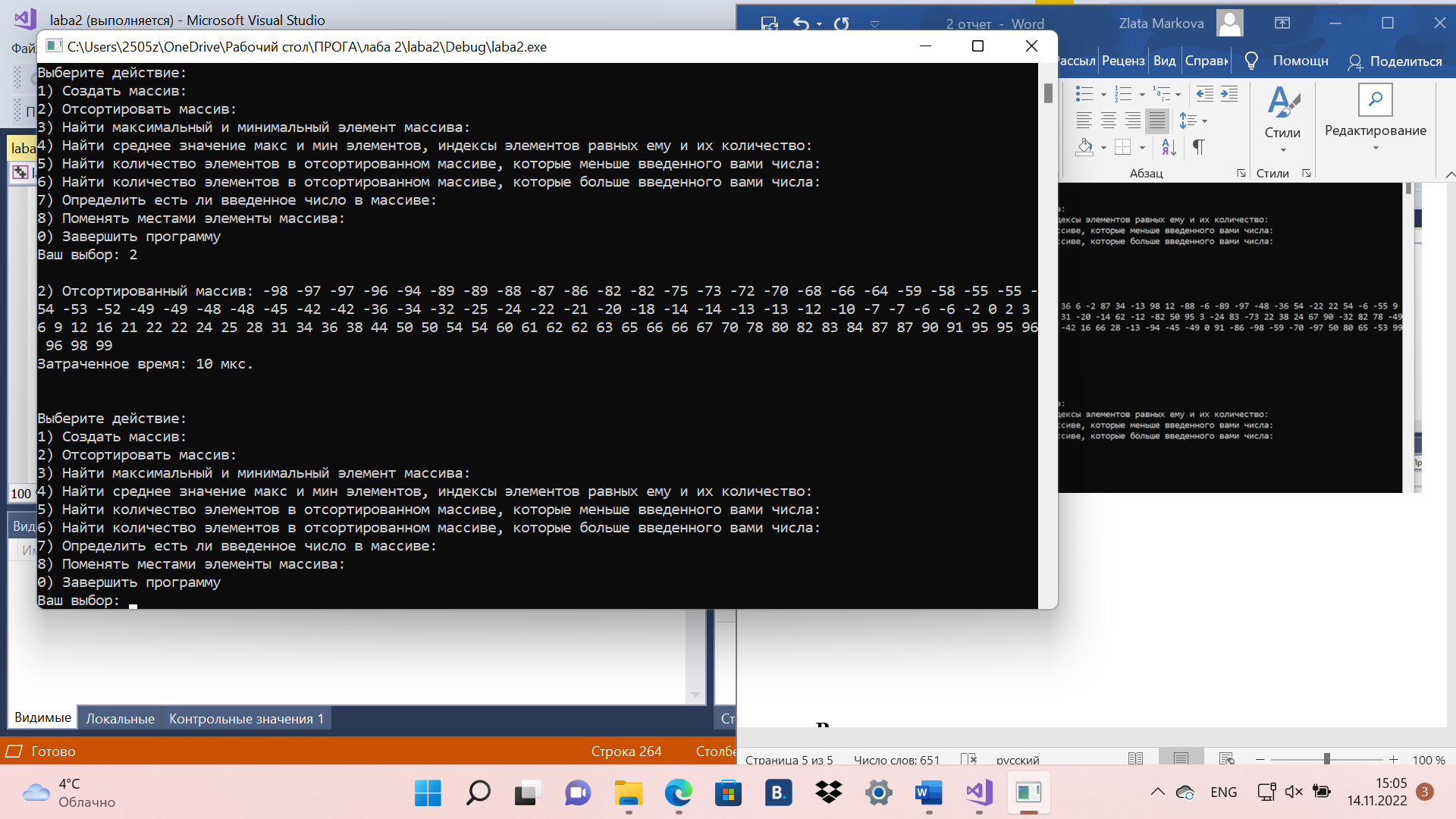
**Выполнение работы.**

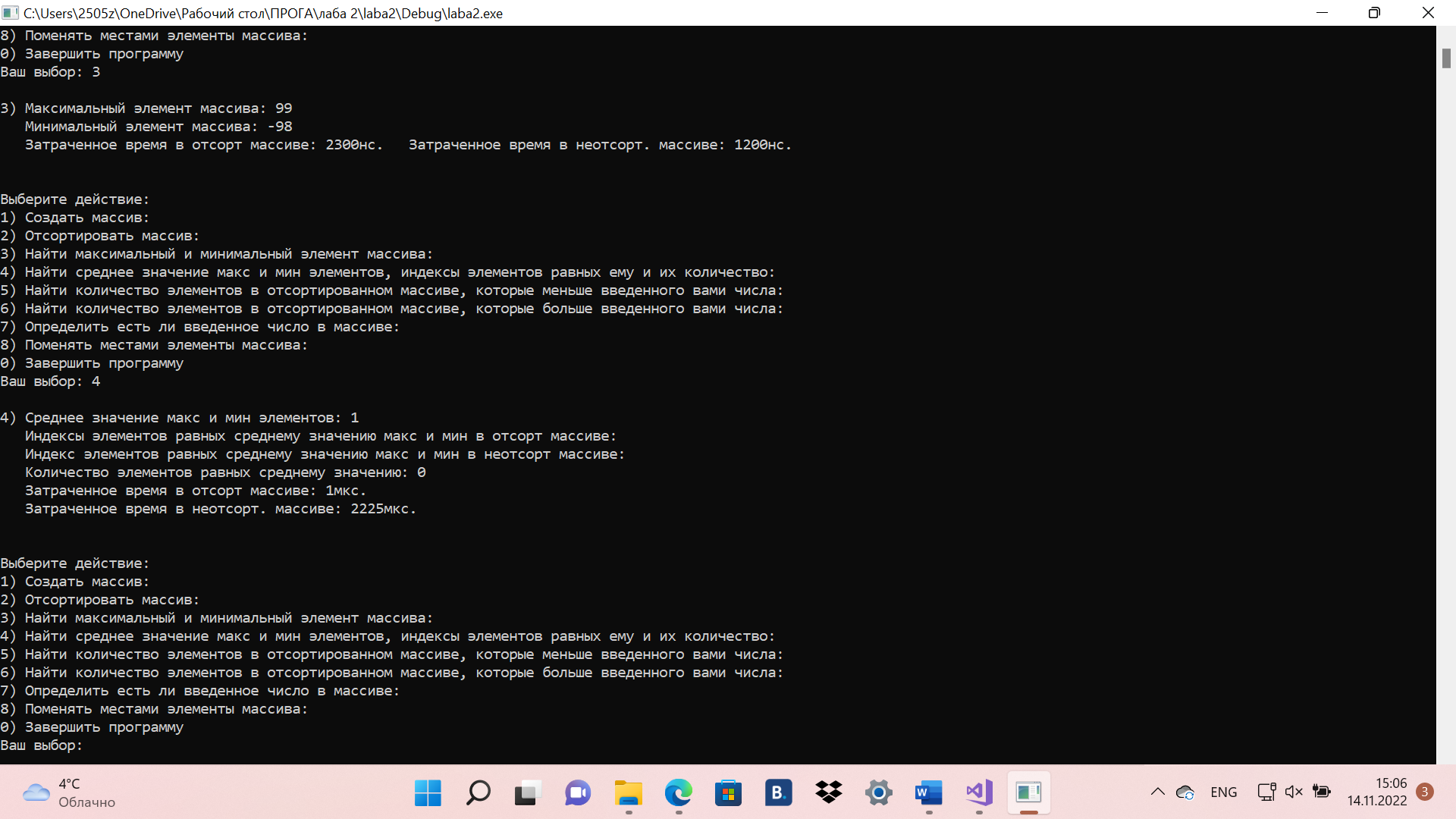
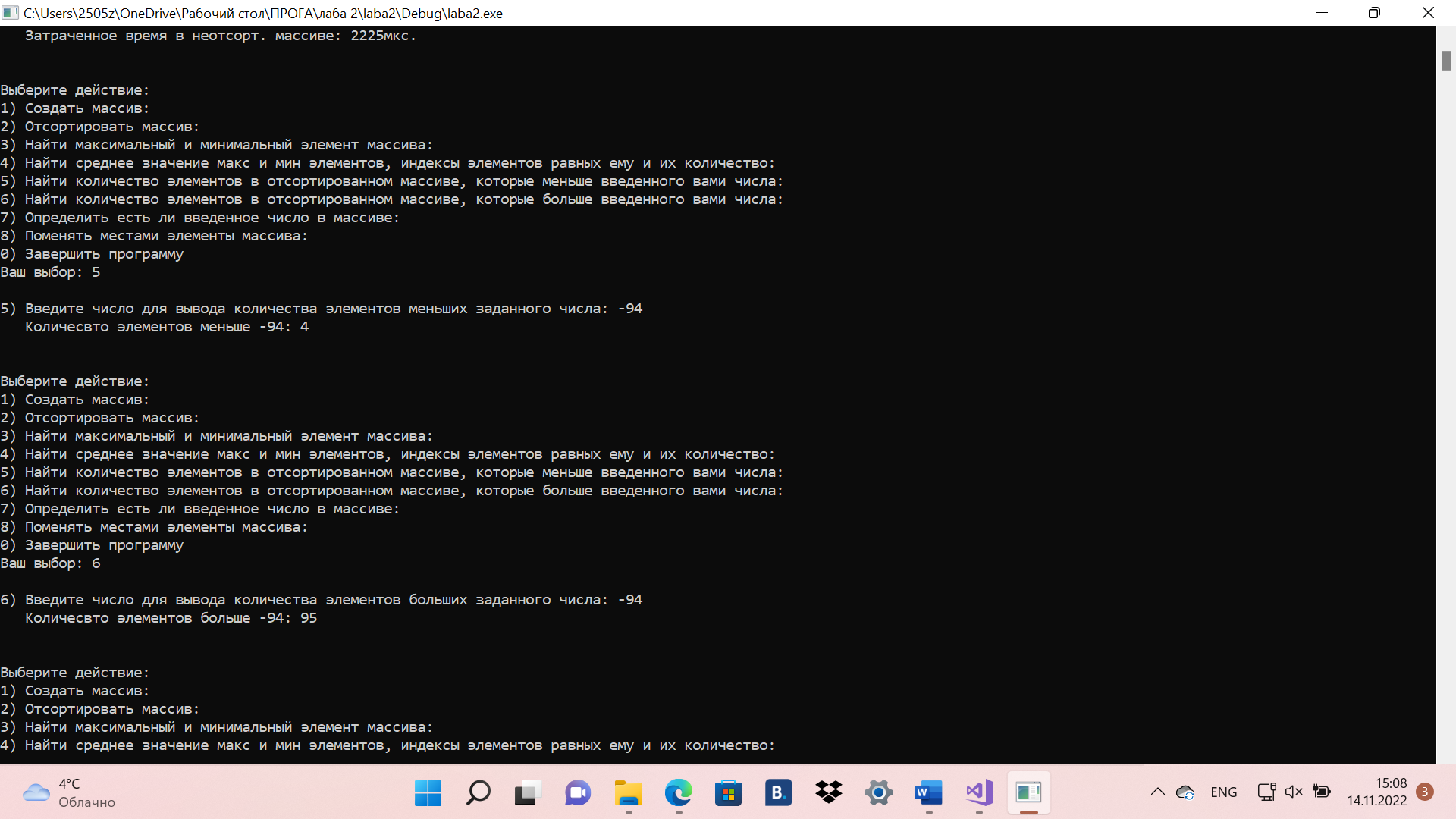
Код программы представлен в приложении А.

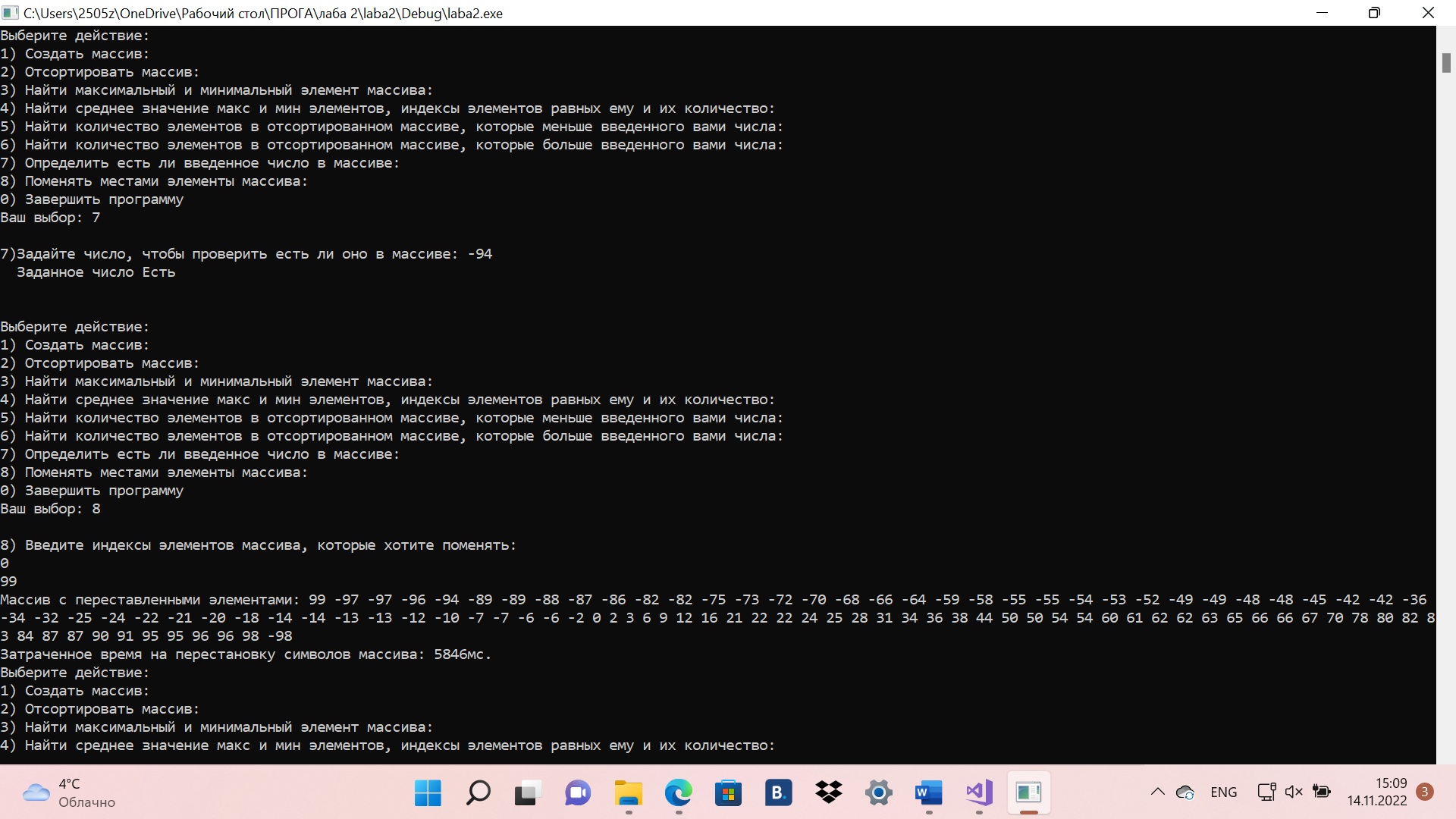
Блок описания кода и использованных алгоритмов

Для выполнения второго пункта реализована Insert sort (сортировка вставками). Для измерения времени выполнения той или иной функции используются объекты типа steady\_clock из библиотеки chrono. Для выполнения третьего пункта реализован алгоритм поиска наибольшего и наименьшего значений перебором. Для выполнения четвёртого пункта я беру максимальный и минимальный элементы, найденные в третьем пункте. С помощью перебора ищу индексы элементов равных среднему значению. В пятом и шестом пунктах перебором считаем количество элементов больше или меньше введённого значения. В седьмом пункте реализован алгоритм бинарного поиска. В восьмом пункте просто считываем два значения и меняем их местами.

**Блок скриншотов работы программы**







**Выводы.**

Изучен навык работы с одномерными статическими массивами. Изучены базовые алгоритмы сортировок и поиска. Изучен навык бинарного поиска.

Бинарный поиск может быть медленнее на таких множествах значений как у нас (малого размера), но при увеличении размера массива бинарный поиск будет показывать куда большую скорость, чем обычный перебор.

Приложение А

рабочий код

#include <iostream>

#include <chrono>

using namespace std;

using namespace chrono;

int main()

{

const int n = 100;

int sort[n];

int nesort[n];

time\_point<steady\_clock> start;

time\_point<steady\_clock> end;

time\_point<steady\_clock> start1;

time\_point<steady\_clock> end1;

time\_point<steady\_clock> start2;

time\_point<steady\_clock> end2;

time\_point<steady\_clock> start3;

time\_point<steady\_clock> end3;

time\_point<steady\_clock> start4;

time\_point<steady\_clock> end4;

time\_point<steady\_clock> start5;

time\_point<steady\_clock> end5;

while (true)

{

setlocale(0, "");

cout << "Выберите действие: " << "\n";

cout << "1) Создать массив:" << "\n";

cout << "2) Отсортировать массив:\n";

cout << "3) Найти максимальный и минимальный элемент массива:\n";

cout << "4) Найти среднее значение макс и мин элементов, индексы элементов равных ему и их количество:\n";

cout << "5) Найти количество элементов в отсортированном массиве, которые меньше введенного вами числа:\n";

cout << "6) Найти количество элементов в отсортированном массиве, которые больше введенного вами числа:\n";

cout << "7) Определить есть ли введенное число в массиве:\n";

cout << "8) Поменять местами элементы массива:\n";

cout << "0) Завершить программу\n";

cout << "Ваш выбор: ";

int p;

cin >> p;

cout << "\n";

if (p == 0)

break;

switch (p)

{

case 1:

//1 ЗАДАНИЕ

cout << "1) Неотсортированный массив: ";

for (int i = 0; i < n; i++)

{

nesort[i] = rand() % 199 - 99;

cout << nesort[i] << " ";

}

for (int i = 0; i < n; i++)

{

sort[i] = nesort[i];

}

cout << "\n\n\n";

break;

case 2:

//2 ЗАДАНИЕ

cout << "2) Отсортированный массив: ";

start = chrono::steady\_clock::now();

int x, j;

for (int i = 1; i < n; i++)

{

x = sort[i];

j = i;

while ((j >= 0) && (x < sort[j - 1]))

{

sort[j] = sort[j - 1];

j--;

}

sort[j] = x;

}

end = chrono::steady\_clock::now();

for (int i = 0; i < n; i++)

{

cout << sort[i] << " ";

}

cout << "\n";

cout << "Затраченное время: " << duration\_cast<microseconds>(end - start).count() << " мкс.";

cout << "\n\n\n";

break;

case 3:

//3 ЗАДАНИЕ

//поиск max и min в отсортированном массиве

start1 = chrono::steady\_clock::now();

int mns;

mns = sort[0];

int mas;

mas = sort[0];

for (int i = 0; i < n; i++)

{

if (sort[i] > mas)

mas = sort[i];

if (sort[i] < mns)

mns = sort[i];

}

end1 = chrono::steady\_clock::now();

//поиск max и min в неотсортированном массиве

start2 = chrono::steady\_clock::now();

int mn;

mn = nesort[0];

int ma;

ma = nesort[0];

for (int i = 0; i < n; i++)

{

if (nesort[i] > ma)

ma = nesort[i];

if (nesort[i] < mn)

mn = nesort[i];

}

end2 = chrono::steady\_clock::now();

cout << "3) Максимальный элемент массива: " << mas << "\n";

cout << " Минимальный элемент массива: " << mns << "\n";

cout << " Затраченное время в отсорт массиве: " << duration\_cast<nanoseconds>(end1 - start1).count() << "нс.";

cout << " Затраченное время в неотсорт. массиве: " << duration\_cast<nanoseconds>(end2 - start2).count() << "нс.";

cout << "\n\n\n";

break;

case 4:

//4 ЗАДАНИЕ

int sum;

int mid;

int k;

k = 0;

sum = mas + mns;

if (sum % 2 != 0)

sum += 1;

mid = sum / 2;

cout << "4) Среднее значение макс и мин элементов: " << mid << "\n";

cout << " Индексы элементов равных среднему значению макс и мин в отсорт массиве: ";

start3 = chrono::steady\_clock::now();

for (int i = 0; i < n; i++)

{

if (sort[i] == mid)

{

cout << i << " ";

k++;

}

}

end3 = chrono::steady\_clock::now();

cout << "\n";

cout << " Индекс элементов равных среднему значению макс и мин в неотсорт массиве: ";

start4 = chrono::steady\_clock::now();

for (int i = 0; i < n; i++)

{

if (nesort[i] == mid)

{

cout << i << " ";

}

}

cout << "\n";

cout << " Количество элементов равных среднему значению: " << k;

end4 = chrono::steady\_clock::now();

cout << "\n";

cout << " Затраченное время в отсорт массиве: " << duration\_cast<microseconds>(end3 - start3).count() << "мкс." << "\n";

cout << " Затраченное время в неотсорт. массиве: " << duration\_cast<microseconds>(end4 - start4).count() << "мкс." << "\n";

cout << "\n\n";

break;

case 5:

//5 ЗАДАНИЕ

cout << "5) Введите число для вывода количества элементов меньших заданного числа: ";

int ka;

ka = 0;

int a;

cin >> a;

for (int i = 0; i < n; i++)

{

if (sort[i] < a)

ka += 1;

}

cout << " Количесвто элементов меньше " << a << ": " << ka;

cout << "\n\n\n";

break;

case 6:

//6 ЗАДАНИЕ

cout << "6) Введите число для вывода количества элементов больших заданного числа: ";

int b;

cin >> b;

int kb;

kb = 0;

for (int i = 0; i < n; i++)

{

if (sort[i] > b)

kb += 1;

}

cout << " Количесвто элементов больше " << b << ": " << kb;

cout << "\n\n\n";

break;

case 7:

//7 ЗАДАНИЕ

int num;

int st;

st = 0;

int en;

en = n;

cout << "7)Задайте число, чтобы проверить есть ли оно в массиве: ";

cin >> num;

int midl;

for (int i = 0; i < n; i++)

{

for (int i = st; i < en; i++)

{

midl = ((st + en) / 2) + ((st + en) % 2);

if (num < sort[midl])

en = midl;

if (num > sort[midl])

{

st = midl + 1;

}

if (num == sort[st] or num == sort[midl])

{

break;

}

}

if (num == sort[st] or num == sort[midl])

{

cout << " Заданное число Есть";

break;

}

else

{

cout << " Заданного числа Нет";

break;

}

}

cout << "\n\n\n";

break;

case 8:

//8 ЗАДАНИЕ

cout << "8) Введите индексы элементов массива, которые хотите поменять: " << "\n";

start5 = chrono::steady\_clock::now();

int m, l;

cin >> m >> l;

int swap;

swap = sort[l];

sort[l] = sort[m];

sort[m] = swap;

end5 = chrono::steady\_clock::now();

cout << "Массив с переставленными элементами: ";

for (int i = 0; i < n; i++)

{

cout << sort[i] << " ";

}

cout << "\n" << "Затраченное время на перестановку символов массива: " << duration\_cast<milliseconds>(end5 - start5).count() << "мс." << "\n";

break;

}

}

}