**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

|  |  |  |
| --- | --- | --- |
| Студент(ка) гр. | Чесноков М. 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.

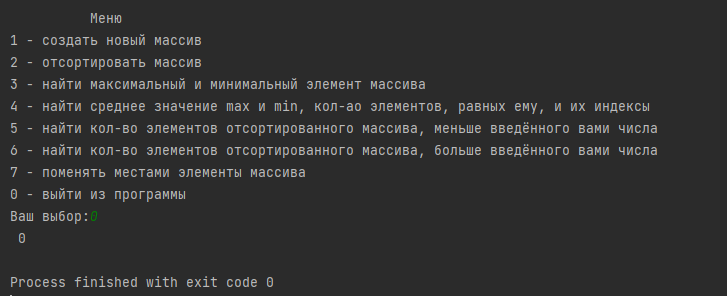
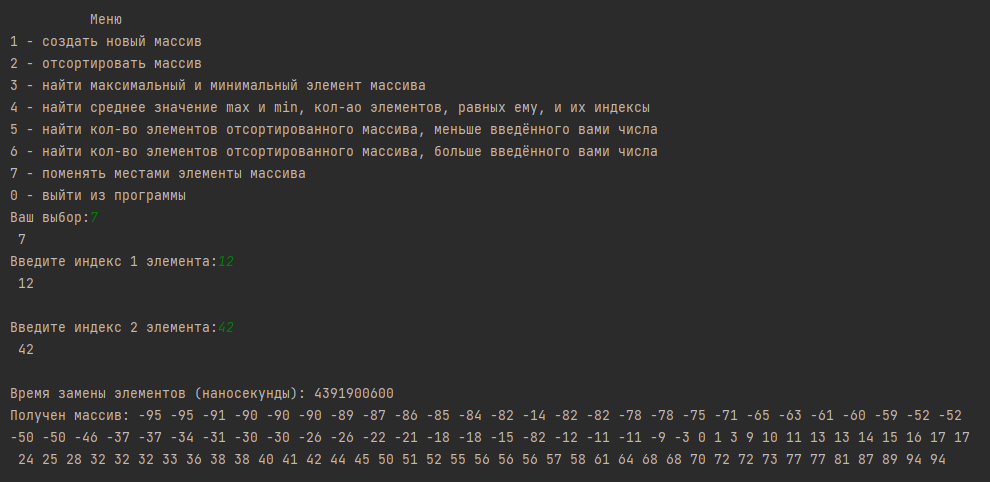
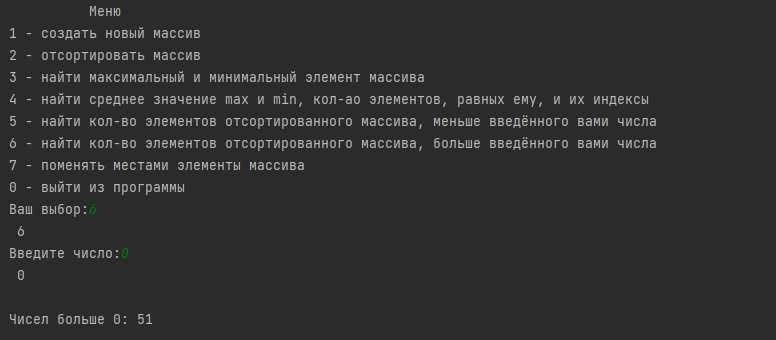
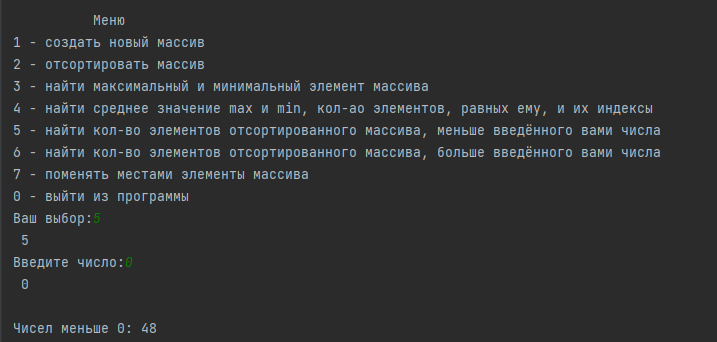
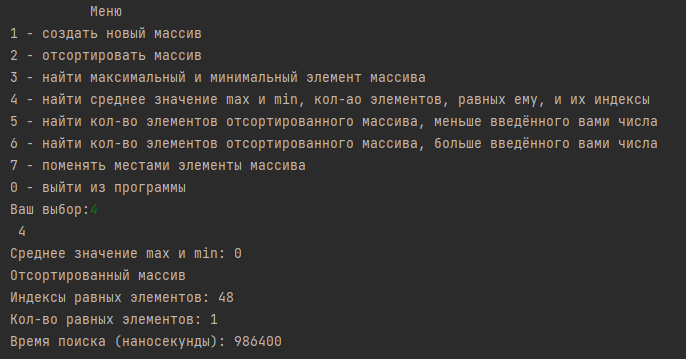
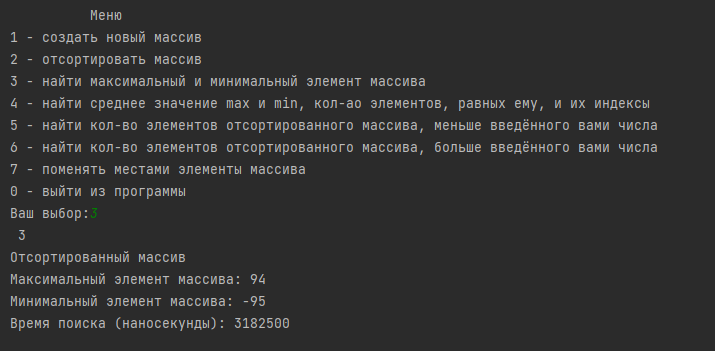
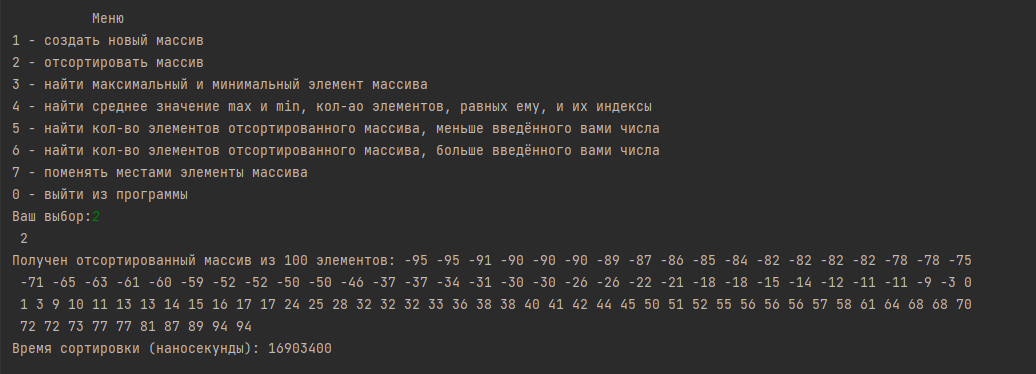
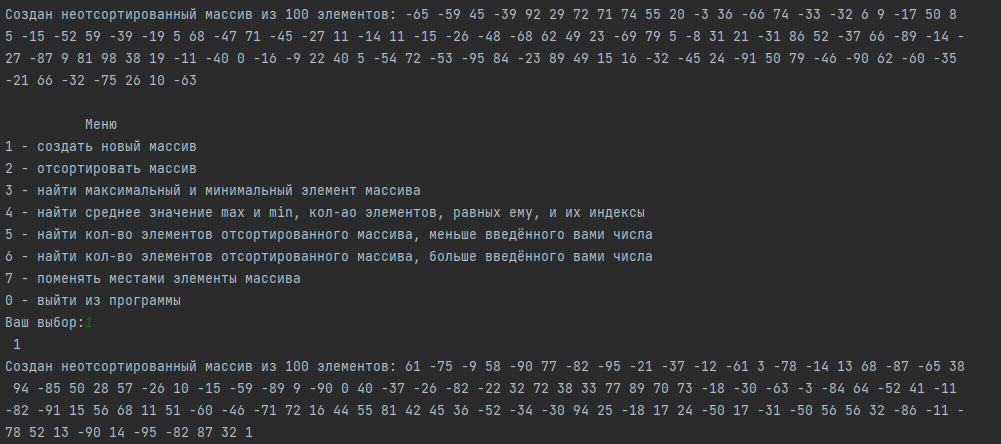
**Выполнение работы.**

Код программы представлен в приложении А.

Блок описания кода и использованных алгоритмов

Для выполнения первого пункта реализован алгоритм быстрой сортировки. Для выбора опорного элемента написан алгоритм подсчёта среднего значения и поиска среди значений массива, ближайшего к среднему. Для измерения времени выполнения той или иной функции используются объекты типа time\_point из библиотеки chrono. Для выполнения второго пункта реализован алгоритм поиска наибольшего и наименьшего значений перебором. Для выполнения третьего пункта не реализовывался алгоритм поиска max и min значений и алгоритм из второго не использовался, т.к. функция из второго пункта просто выводила найденные значения в консоль, ничего не возвращая. Поэтому я просто сортирую массив и работаю с первым и последним значением. В пятом и шестом пунктах просто перебором считаем количество элементов больше или меньше введённого значения. В седьмом пункте реализован алгоритм бинарного пункта без рекурсии. В восьмом пункте просто считываем два значения и меняем их местами как в любой сортировке.

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>  
#include <cmath>  
using namespace std;  
using namespace chrono;  
  
  
int main(){  
 time\_point<steady\_clock, duration<\_\_int64, ratio<1, 1000000000>>> s, e;  
 srand(time(0));  
 setlocale(LC\_ALL, "ru\_RU.UTF-8"); //Я пишу в CLion, тут по другому русский язык подключать надо, в отличии от VS  
 system("chcp 65001");  
 const int n = 100;  
 int arr[n], r, sort = 0, sr, nm, k, max, min;  
  
 //Создание массива (1)  
 cout << "Создан неотсортированный массив из 100 элементов: ";  
 for (int i = 0; i < n; i++){  
 arr[i] = rand() % 198 - 99;  
 cout << arr[i] << " ";  
 }  
 sort = 0; //Проверка на сортировку  
 cout << endl << endl;  
  
 while (true){  
  
 //Меню  
 cout << " Меню\n" <<  
 "1 - создать новый массив\n" <<  
 "2 - отсортировать массив\n" <<  
 "3 - найти максимальный и минимальный элемент массива\n" <<  
 "4 - найти среднее значение max и min, кол-ао элементов, равных ему, и их индексы\n" <<  
 "5 - найти кол-во элементов отсортированного массива, меньше введённого вами числа\n" <<  
 "6 - найти кол-во элементов отсортированного массива, больше введённого вами числа\n" <<  
 "7 - поменять местами элементы массива\n" <<  
 "0 - выйти из программы\n";  
  
 cout << "Ваш выбор: ";  
 cin >> r;  
 if (r == 0) break;  
 switch (r){  
  
 case 1:  
 //Создание нового массива (1)  
 cout << "Создан неотсортированный массив из 100 элементов: ";  
 for (int i = 0; i < n; i++){  
 arr[i] = rand() % 198 - 99;  
 cout << arr[i] << " ";  
 }  
 sort = 0;  
 cout << endl << endl;  
 break;  
  
 case 2:  
 //Сортировка массива shaker-sort (2)  
 s = steady\_clock::now();  
 nm = n - 1;  
 cout << "Получен отсортированный массив из 100 элементов: ";  
 for (int i = 0; i < n; i++){  
 for(int j = 0; j < n - 1 - i; j++){  
 if (arr[j] > arr[j+1]){  
 swap (arr[j], arr[j+1]);  
 }  
 if (arr[nm-j] < arr[nm-(j+1)]){  
 swap (arr[nm-j], arr[nm-(j+1)]);  
 }  
 }  
 }  
 for (int i = 0; i < n; i++){  
 cout << arr[i] << " ";  
 }  
 sort = 1;  
 e = steady\_clock::now();  
 cout << endl << "Время сортировки (наносекунды): "<< duration\_cast<nanoseconds>(e - s).count() << endl << endl;  
 break;  
  
 case 3:  
 //Нахождение max и min элемента массива (3)  
 max = -101;  
 min = 101;  
 if (sort == 1){  
 s = steady\_clock::now();  
 max = arr[n-1];  
 min = arr[0];  
 cout << "Отсортированный массив" << endl;  
 cout << "Максимальный элемент массива: " << max << endl << "Минимальный элемент массива: " << min;  
 e = steady\_clock::now();  
 cout << endl << "Время поиска (наносекунды): "<< duration\_cast<nanoseconds>(e - s).count() << endl << endl;  
 }  
 else{  
 s = steady\_clock::now();  
 cout << "Неотсортированный массив" << endl;  
 for (int i = 0; i < n; i++){  
 if (arr[i] > max){  
 max = arr[i];  
 }  
 if (arr[i] < min){  
 min = arr[i];  
 }  
 }  
 cout << "Максимальный элемент массива: " << max << endl << "Минимальный элемент массива: " << min;  
 e = steady\_clock::now();  
 cout << endl << "Время поиска (наносекунды): "<< duration\_cast<nanoseconds>(e - s).count() << endl << endl;  
 }  
 break;  
  
 case 4:  
 //Среднее значение max и min,кол-во равных элементов и их индексы (4)  
 sr = max+min;  
 k = 0;  
 if ((sr % 2) != 0) {  
 sr += 1;  
 }  
 sr /= 2;  
 cout << "Среднее значение max и min: " << sr << endl;  
 if (sort == 1) {  
 s = steady\_clock::now();  
 cout << "Отсортированный массив" << endl;  
 cout << "Индексы равных элементов: ";  
 for(int i = 0; i < n; i++){  
 if (arr[i] == sr){  
 cout << i << " ";  
 k++;  
 }  
 if (arr[i] > sr){  
 e = steady\_clock::now();  
 cout << endl << "Кол-во равных элементов: " << k;  
 cout << endl << "Время поиска (наносекунды): "<< duration\_cast<nanoseconds>(e - s).count() << endl << endl;  
 break;  
 }  
 }  
 }  
 else {  
 s = steady\_clock::now();  
 cout << "Неотсортированный массив" << endl;  
 cout << "Индексы равных элементов: ";  
 for (int i = 0; i < n; i++) {  
 if (arr[i] == sr) {  
 cout << i << " ";  
 k++;  
 }  
 }  
 e = steady\_clock::now();  
 cout << endl << "Кол-во равных элементов: " << k;  
 cout << endl << "Время поиска (наносекунды): "<< duration\_cast<nanoseconds>(e - s).count() << endl << endl;  
 }  
 break;  
  
 case 5:  
 //Кол-во элементов отсортированного массива, которые меньше введённого числа (5)  
 int a;  
 k = 0;  
 if (sort == 1){  
 cout << "Введите число: ";  
 cin >> a;  
 for (int i = 0; i < n; i++){  
 if (arr[i] < a){  
 k++;  
 }  
 else {  
 break;  
 }  
 }  
 cout << endl << "Чисел меньше " << a << ": " << k << endl;  
 }  
 else{  
 cout << "Сначала отсортируйте массив" << endl;  
 }  
 break;  
  
 case 6:  
 //Кол-во элементов отсортированного массива, которые больше введённого числа (6)  
 int b;  
 k = 0;  
 if (sort == 1){  
 cout << "Введите число: ";  
 cin >> b;  
 for (int i = n-1; i >= 0; i--){  
 if (arr[i] > b){  
 k++;  
 }  
 else {  
 break;  
 }  
 }  
 cout << endl << "Чисел больше " << b << ": " << k << endl << endl;  
 }  
 else{  
 cout << "Сначала отсортируйте массив" << endl << endl;  
 }  
 break;  
  
 case 7:  
 //Поменять местами элементы массива (8)  
 int i1, i2;  
 cout << "Введите индекс 1 элемента: ";  
 cin >> i1;  
 cout << endl << "Введите индекс 2 элемента: ";  
 cin >> i2;  
 s = steady\_clock::now();  
 swap (arr[i1], arr[i2]);  
 e = steady\_clock::now();  
 cout << endl << "Время замены элементов (наносекунды): "<< duration\_cast<nanoseconds>(e - s).count() << endl;  
 cout << "Получен массив: ";  
 for (int i = 0; i < n; i++){  
 cout << arr[i] << " ";  
 }  
 cout << endl << endl;  
 break;  
  
 default:  
 cout << "Пункт меню выбран неправильно\n";  
 }  
 }  
}