**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 гр. 0323 |  | Квачадзе С.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. (\*)

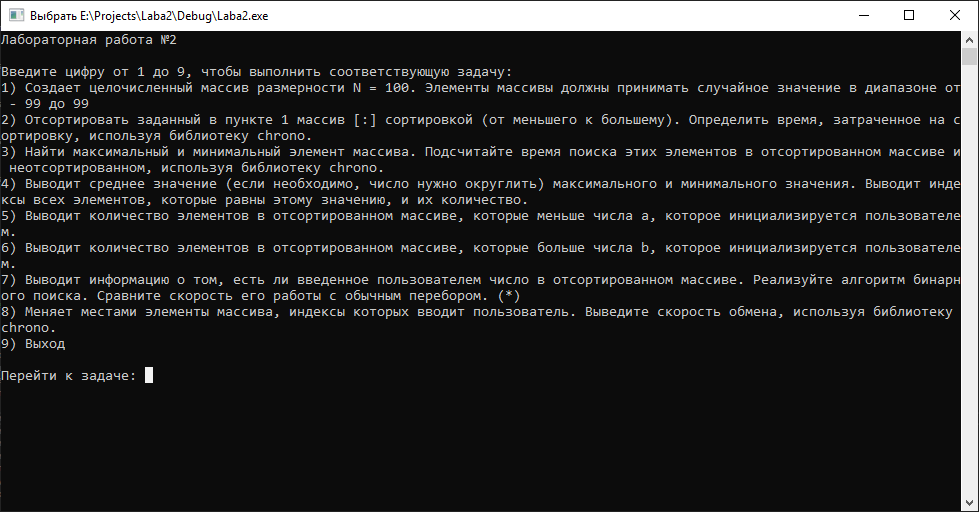
8) Меняет местами элементы массива, индексы которых вводит пользователь. Выведите скорость обмена, используя библиотеку chrono.

Должна присутствовать возможность запуска каждого пункта многократно.

**Экспериментальные результаты.**

Вначале в консоль выводится интерактивное меню, позволяющее выбрать одну из задач, которые программа может выполнить.

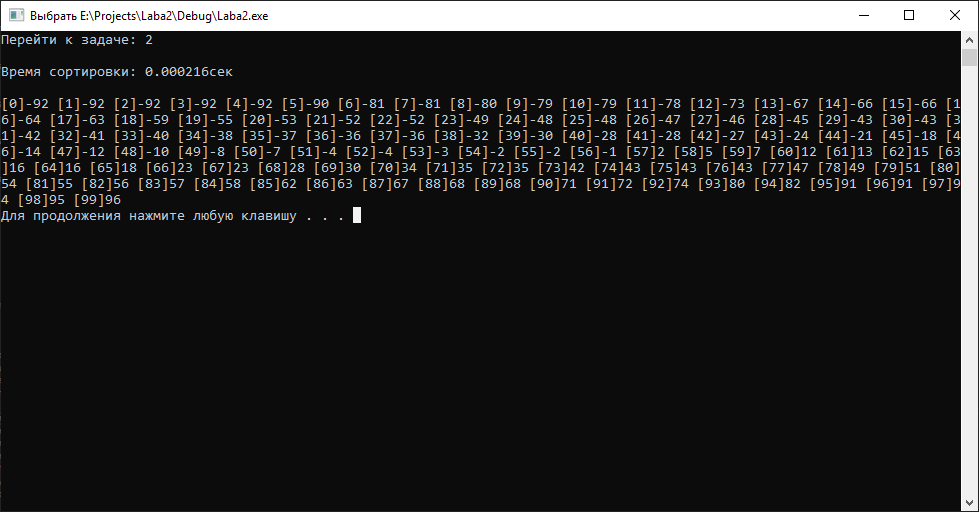
Чтобы продолжить, нужно выбрать номер задания.



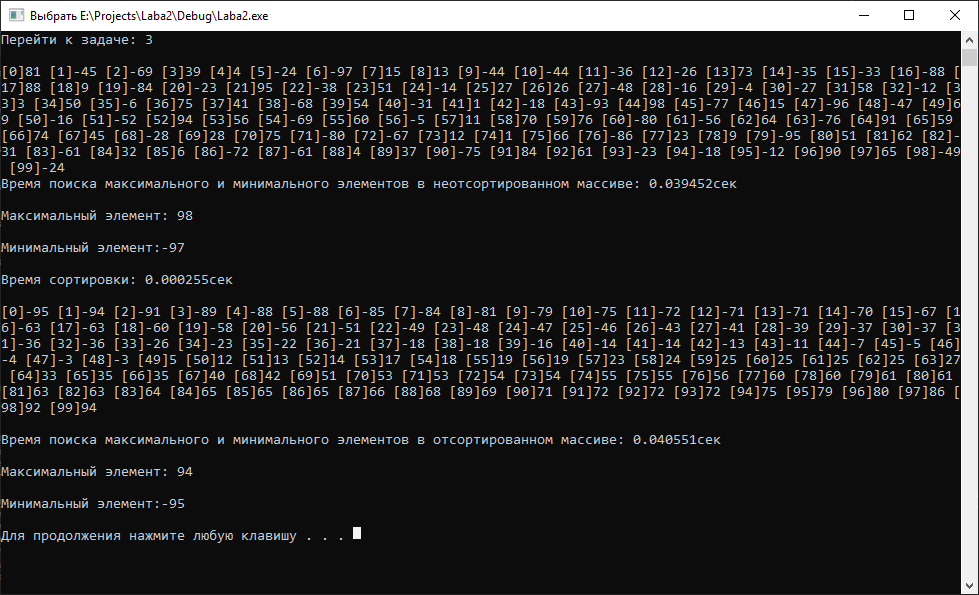
Пример работы первого задания:



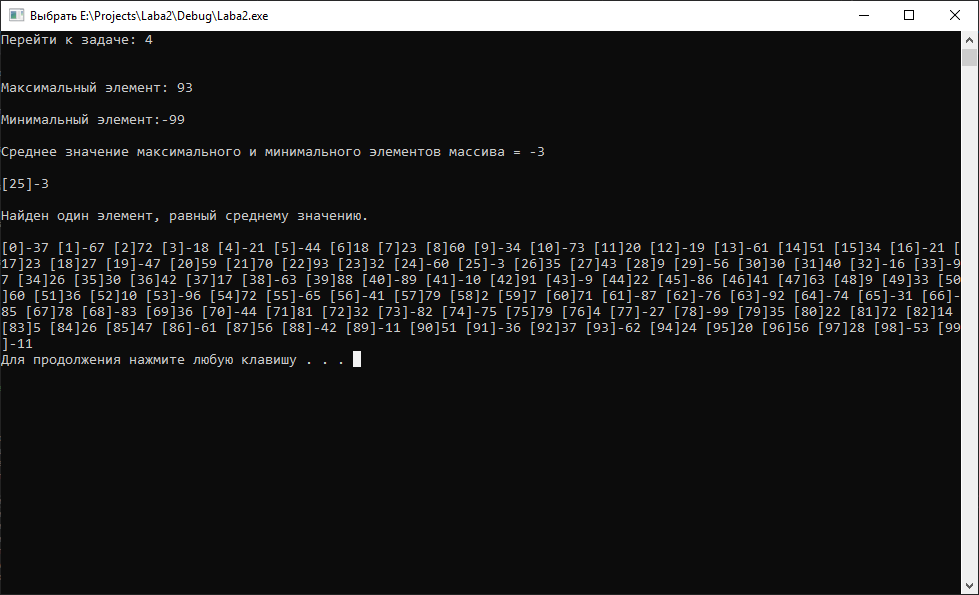
Пример работы второго задания:



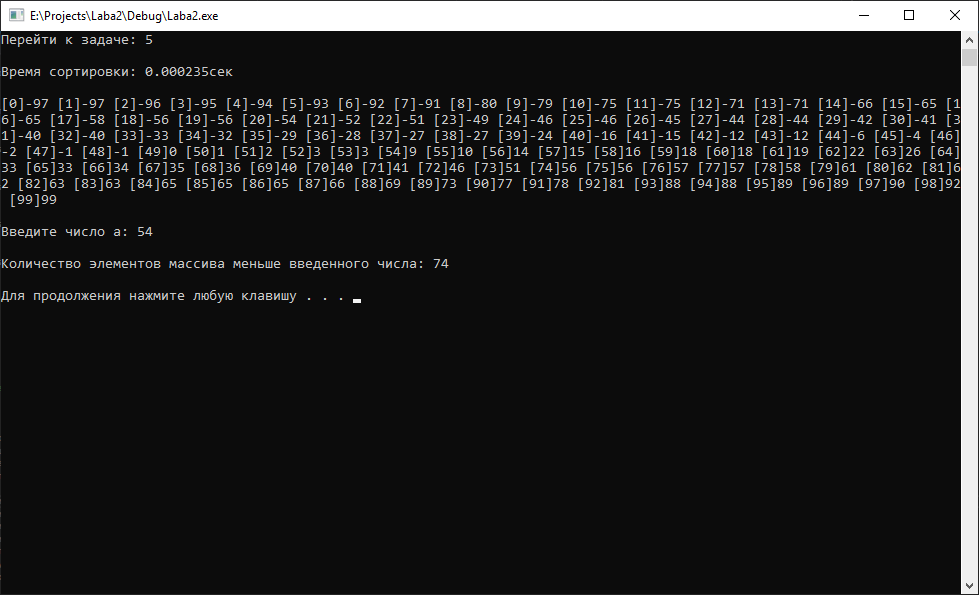
Пример работы третьего задания:



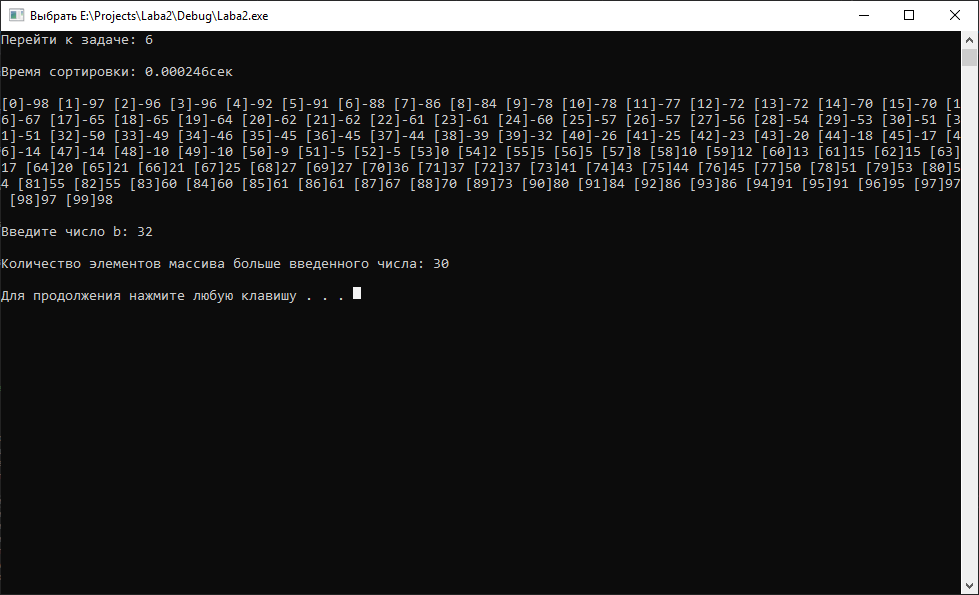
Пример работы четвертого задания:



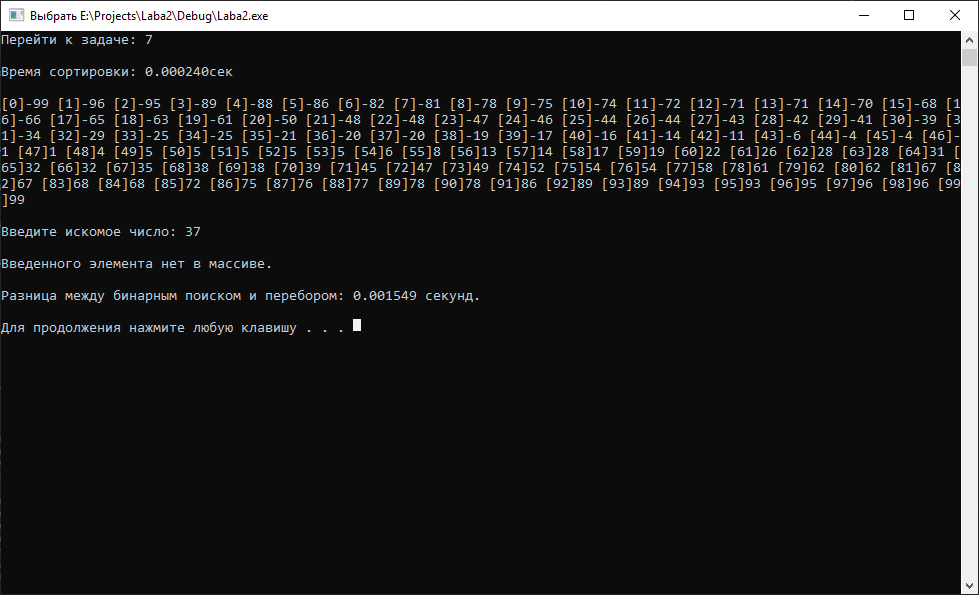
Пример работы пятого задания:



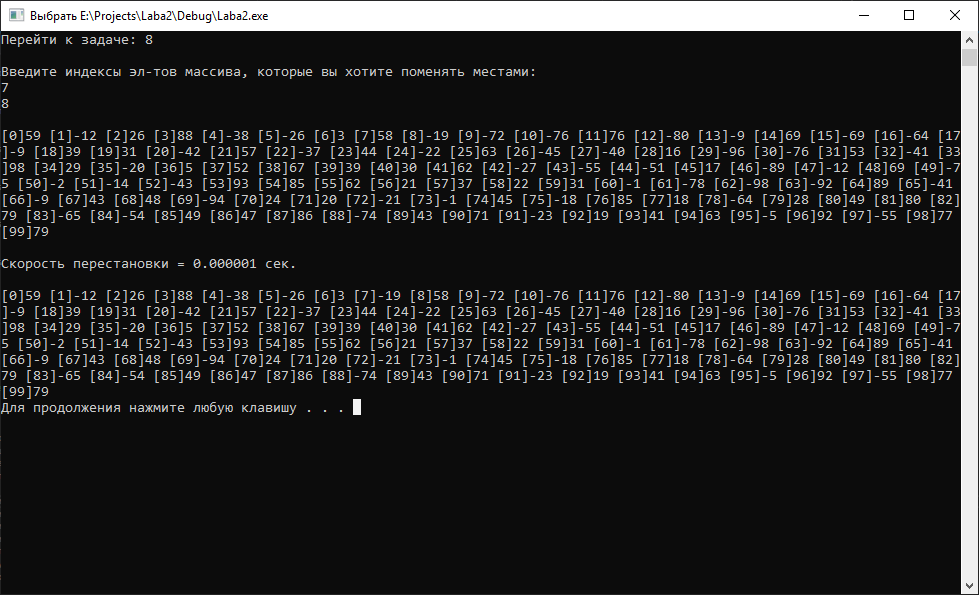
Пример работы шестого задания:



Пример работы седьмого задания:



Пример работы восьмого задания:



Полный код программы представлен в приложении А.

**Выводы.**

Практика показывает, что с одномерными статическими массивами работать намного легче, когда они отсортированы. Также полезно знать разные виды сортировок и уметь пользоваться бинарным поиском, который работает эффективнее обычного перебора элементов.

**ПРИЛОЖЕНИЕ А**

**ПОЛНЫЙ КОД ПРОГРАММЫ**

#include <iostream>

#include <chrono>

using namespace std;

using namespace chrono;

short int menu();

void massiv();

void Task1();

void Task2();

void Task3\_unsort();

void Task3\_sort();

void Task4();

void Task5();

void Task6();

void Task7();

void Task8();

int main() // меню

{

setlocale(LC\_ALL, "Russian");

short int choose\_menu, exit = 0;

while (1)

{

choose\_menu = menu();

switch (choose\_menu)

{

case 1:

Task1(); massiv(); system("pause"); system("cls");

break;

case 2:

Task1(); Task2(); massiv(); system("pause"); system("cls");

break;

case 3:

Task3\_unsort(); Task3\_sort(); system("pause"); system("cls");

break;

case 4:

Task4(); massiv(); system("pause"); system("cls");

break;

case 5:

Task5(); system("pause"); system("cls");

break;

case 6:

Task6(); system("pause"); system("cls");

break;

case 7:

Task7(); system("pause"); system("cls");

break;

case 8:

Task8(); system("pause"); system("cls");

break;

case 9:return 0;

break;

}

cout << endl;

}

}

int arr[100];

short int menu() // меню, выбор задания

{

setlocale(LC\_ALL, "Russian");

short int i;

cout << "Лабораторная работа №2 \n\n";

cout << "Введите цифру от 1 до 9, чтобы выполнить соответствующую задачу:\n";

cout << "1) Создает целочисленный массив размерности N = 100. Элементы массивы должны принимать случайное значение в диапазоне от - 99 до 99\n";

cout << "2) Отсортировать заданный в пункте 1 массив […] сортировкой (от меньшего к большему). Определить время, затраченное на сортировку, используя библиотеку chrono.\n";

cout << "3) Найти максимальный и минимальный элемент массива. Подсчитайте время поиска этих элементов в отсортированном массиве и неотсортированном, используя библиотеку chrono.\n";

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

cout << "5) Выводит количество элементов в отсортированном массиве, которые меньше числа a, которое инициализируется пользователем.\n";

cout << "6) Выводит количество элементов в отсортированном массиве, которые больше числа b, которое инициализируется пользователем.\n";

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

cout << "8) Меняет местами элементы массива, индексы которых вводит пользователь. Выведите скорость обмена, используя библиотеку chrono.\n";

cout << "9) Выход\n" << endl;

cout << "Перейти к задаче: ";

while (1)

{

cin >> i;

if ((i > 0) & (i <= 9))

break;

else

cout << "Некорректный ввод, повторите попытку.\n";

}

cout << "\n";

return i;

}

void Task1() // Заполнение массива случайными числами от -99 до 99

{

for (int i = 0; i < 100; i++)

{

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

}

}

void massiv() // Вывод массива

{

for (int i = 0; i < 100; i++)

{

cout << '[' << i << ']' << arr[i] << " ";

}

cout << endl;

}

void Task2() // Сортировка массива

{

system\_clock::time\_point start = system\_clock::now();

int iRight = 99;

int iLeft = 0;

int iSwap = iRight;

for (int g = 0; g < 100; g++)

{

for (int i = iLeft; i < iRight; i++)

{

if (arr[i] > arr[i + 1])

{

std::swap(arr[i], arr[i + 1]);

iSwap = i;

}

}

iRight = iSwap;

for (int i = iRight; i > iLeft; i--)

{

if (arr[i] < arr[i - 1])

{

std::swap(arr[i], arr[i - 1]);

iSwap = i;

}

}

iLeft = iSwap;

if (iLeft >= iRight) break;

}

system\_clock::time\_point end = system\_clock::now();

duration<double> sort = end - start;

cout << "Время сортировки: " << std::fixed << sort.count() << "сек" << endl << endl;

}

void Task3\_unsort() // Максимальный и минимальный элементы неотсортированного массива. Время работы.

{

auto startMaxMinUnsort = system\_clock::now();

Task1();

massiv();

int min = 100, max = -100;

for (int i = 0; i < 100; i++)

{

if (arr[i] < min) min = arr[i];

if (arr[i] > max) max = arr[i];

}

auto endMaxMinUnsort = system\_clock::now();

duration<double> sortMaxMinUnsort = endMaxMinUnsort - startMaxMinUnsort;

cout << std::fixed << "Время поиска максимального и минимального элементов в неотсортированном массиве: " << sortMaxMinUnsort.count() << "сек" << endl << endl;

cout << "Максимальный элемент: " << max << endl << endl;

cout << "Минимальный элемент:" << min << endl << endl;

}

void Task3\_sort() // Максимальный и минимальный элементы в отсортированном массиве.

{

auto startMaxMinSort = system\_clock::now();

Task1();

Task2();

massiv();

system\_clock::time\_point endMaxMinSort = system\_clock::now();

duration<double> sortMaxMin = endMaxMinSort - startMaxMinSort;

cout << endl;

cout << "Время поиска максимального и минимального элементов в отсортированном массиве: " << std::fixed << sortMaxMin.count() << "сек" << endl << endl;

cout << "Максимальный элемент: " << arr[99] << endl << endl;

cout << "Минимальный элемент:" << arr[0] << endl << endl;

}

void Task4() // Среднее значение минимального и максимального элементов. Кол-во элементов массива, равных среднему значению и их индекс

{

Task1();

int min = 100, max = -100, average;

for (int i = 0; i < 100; i++) // поиск минимального и максимального значений

{

if (arr[i] < min) min = arr[i];

if (arr[i] > max) max = arr[i];

}

average = (min + max) / 2; // поиск среднего значения

cout << endl;

cout << "Максимальный элемент: " << max << endl << endl;

cout << "Минимальный элемент:" << min << endl << endl;

cout << "Среднее значение максимального и минимального элементов массива = " << average << endl << endl;

int count = 0;

for (int i = 0; i < 100; i++) // индекс среднего значения

{

if (arr[i] == average)

{

cout << "[" << i << "]" << arr[i] << endl;

count += 1;

}

}

cout << endl;

if (count == 0)

cout << "В массиве нет элементов, равных среднему значению." << endl << endl;

else if (count == 1)

cout << "Найден один элемент, равный среднему значению." << endl << endl;

else if (count >= 2 && count <= 4) cout << "Найдено " << count << " элемента, равных среднему значению." << endl << endl;

else cout << "Найдено" << count << "элементов, равных среднему значению." << endl << endl;

}

void Task5() // Количество элементов массива, меньше введенного числа

{

Task1();

Task2();

massiv();

int a, z = 0;

cout << endl;

cout << "Введите число a: ";

cin >> a;

for (int i = 0; i < 100; i++)

{

if (arr[i] < a) z++;

}

cout << endl;

cout << "Количество элементов массива меньше введенного числа: " << z << endl << endl;

}

void Task6() // Количество элементов массива, больше введенного числа

{

Task1();

Task2();

massiv();

int b, x = 0;

cout << endl;

cout << "Введите число b: ";

cin >> b;

for (int i = 0; i < 100; i++)

{

if (arr[i] > b) x++;

}

cout << endl;

cout << "Количество элементов массива больше введенного числа: " << x << endl << endl;

}

void Task7() // бинарный поиск

{

Task1();

Task2();

massiv();

int bin, min = 0, max = 100, mid;

bool key = false;

cout << endl;

cout << "Введите искомое число: ";

cin >> bin;

cout << endl;

system\_clock::time\_point start1 = system\_clock::now();

while (min <= max) // бинарный поиск

{

mid = (min + max) / 2;

if (arr[mid] == bin) break;

if (arr[mid] < bin) min = mid + 1;

if (arr[mid] > bin) max = mid - 1;

}

if (arr[mid] == bin) {

cout << "Введенный элемент есть в массиве, индекс: " << mid << endl << endl;

}

else {

cout << "Введенного элемента нет в массиве." << endl << endl;

}

system\_clock::time\_point end1 = system\_clock::now();

duration<double> sort1 = end1 - start1;

system\_clock::time\_point start2 = system\_clock::now();

for (int i = 0; i < 100; i++) { // поиск перебором

if (arr[i] == bin) {

key = true;

}

}

system\_clock::time\_point end2 = system\_clock::now();

duration<double> sort2 = end2 - start2;

cout << fixed << "Разница между бинарным поиском и перебором: " << sort1.count() - sort2.count() << " секунд." << endl;

cout << endl;

}

void Task8() // Меняет местами элементы массива, индексы которых вводит пользователь

{

int one, two;

cout << "Введите индексы эл-тов массива, которые вы хотите поменять местами: " << endl;

cin >> one >> two;

cout << endl;

Task1();

massiv();

cout << endl;

system\_clock::time\_point start = system\_clock::now();

swap(arr[one], arr[two]);

system\_clock::time\_point end = system\_clock::now();

duration<double> sort = end - start;

cout << "Скорость перестановки = " << std::fixed << sort.count() << " сек." << endl << endl;

massiv();

}