**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. | Звягина М.С. |  |
| 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.

Shaker sort – модификация пузырьковой сортировки. Принцип работы этой сортировки аналогичен bubble sort: попарное сравнение элементов и последующий обмен местами. Но имеется существенное отличие. Как только максимальный элемент становится на свое место, алгоритм не начинает новую итерацию с первого элемента, а запускает сортировку в обратную сторону. Алгоритм гарантирует, что после выполнения первой итерации, минимальный и максимальный элемент будут в начале и конце массива соответственно.

Затем процесс повторяется до тех пор, пока массив не будет отсортирован. За счет того, что сортировка работает в обе стороны, массив сортируется на порядок быстрее. Очевидным примером этого был бы случай, когда в начале массива стоит максимальный элемент, а в конце массива – минимальный. Shaker sort справится с этим за 1 итерацию, при условии, что другие элементы стоят на правильном месте.

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.**

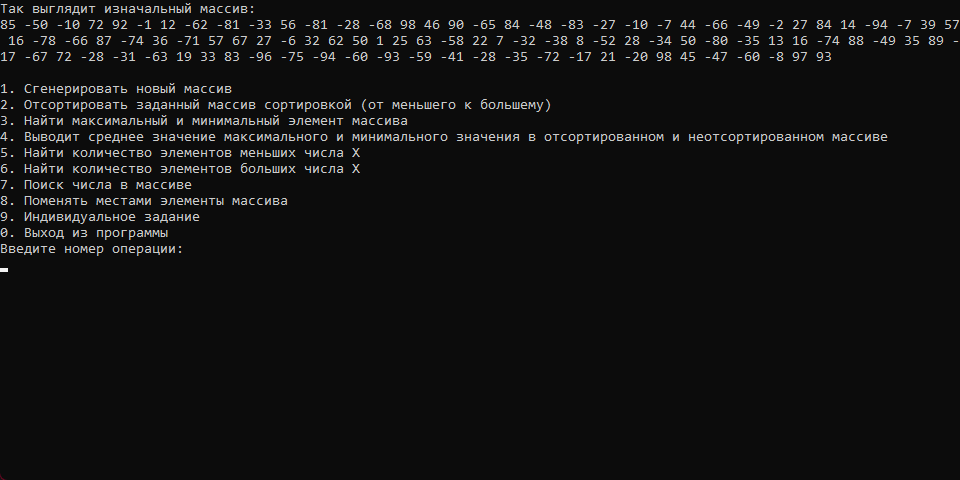
Код программы представлен в приложении А.

Блок описания кода и использованных алгоритмов

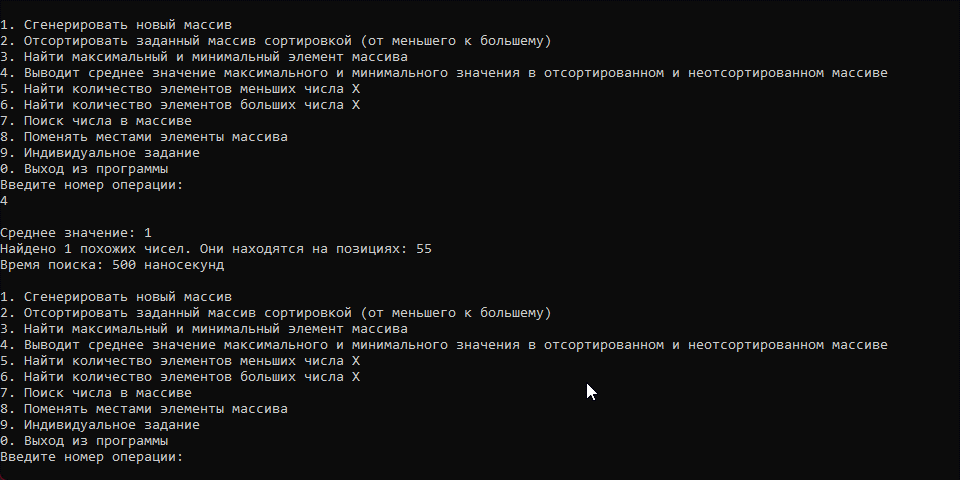
Для выполнения второго пункта реализована возможность выбора сортировки для сравнения времени работы каждого алгоритма. Для выбора опорного элемента написан алгоритм подсчёта среднего значения и поиска среди значений массива, ближайшего к среднему. Для измерения времени выполнения той или иной функции используются объекты типа steady\_clock из библиотеки chrono. Для выполнения третьего пункта реализован алгоритм поиска наибольшего и наименьшего значений перебором. Для выполнения четвёртого пункта я просто сортирую массив и работаю с первым и последним значениями, т. к. они являются минимальным и максимальным соответственно. 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**

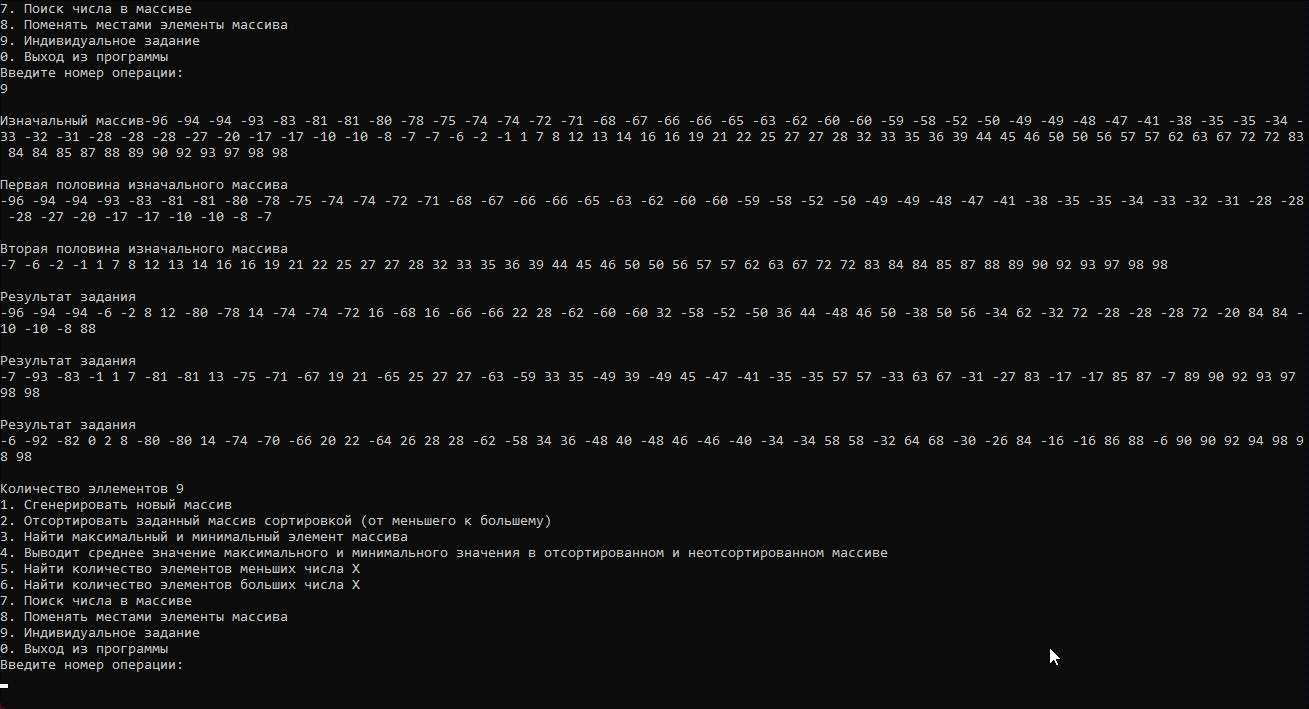
Скриншот 1



Скриншот 2



Скриншот 3



**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 <ctime>

#include <chrono>

using namespace std;

using namespace chrono;

void Shakersort(int A[]) {

for (int n = 100; n > 0; --n) {

for (int i = 0; i < n - 1; ++i) {

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

swap(A[i], A[i + 1]);

}

for (int i = n - 2; i > 0; --i) {

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

swap(A[i], A[i - 1]);

}

}

}

void ViewArray(int A[], int n=100) {

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

cout << A[i] << " ";

cout << "\n";

}

void TwoArray(int A[]) {

int B[50], C[50];

for (int i = 0; i < 50; i++)

B[i] = A[i];

for (int i = 50; i < 100; i++)

C[i-50] = A[i];

cout << "Изначальный массив";

ViewArray(A);

cout << "\n";

cout << "Первая половина изначального массива\n";

ViewArray(B, 50);

cout << "\n";

cout << "Вторая половина изначального массива\n";

ViewArray(C, 50);

cout << " \n";

for (int i = 0; i < 50; i++) {

if (B[i] % 2 != 0) {

for (int j = 0; j < 50; j++) {

if (C[j] % 2 == 0) {

swap(B[i], C[j]);

break;

}

}

}

}

cout << "Результат задания\n";

ViewArray(B, 50);

cout << "\n";

cout << "Результат задания\n";

ViewArray(C, 50);

cout << "\n";

for (int i = 0; i < 50; i++)

if (C[i] % 2)

C[i]++;

cout << "Результат задания\n";

ViewArray(C, 50);

cout << "\n";

int result = 0;

for (int i = 0; i < 50; i++)

if (B[i] > C[i])

result++;

cout << "Количество эллементов " << result << endl;

}

void menu() {

cout << "1. Сгенерировать новый массив\n";

cout << "2. Отсортировать заданный массив сортировкой (от меньшего к большему)\n";

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

cout << "4. Выводит среднее значение максимального и минимального значения в отсортированном и неотсортированном массиве\n";

cout << "5. Найти количество элементов меньших числа X\n";

cout << "6. Найти количество элементов больших числа X\n";

cout << "7. Поиск числа в массиве\n";

cout << "8. Поменять местами элементы массива\n";

cout << "9. Индивидуальное задание\n";

cout << "0. Выход из программы\n";

cout << "Введите номер операции: \n";

}

void handleForCase1(int Ar[]) {

cout << "Новый массив:\n";

srand(time(0));

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

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

ViewArray(Ar);

cout << "\n";

}

void handleForCase2(int A[]) {

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

Shakersort(A);

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

duration<double> sec = end - start;

ViewArray(A);

cout << chrono::duration\_cast<chrono::microseconds>(end - start).count() << " микросекунд\n\n";

}

void handleForCase3(int A[]) {

Shakersort(A);

int min, max;

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

min = A[0];

max = A[99];

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

cout << "Максимальный элемент: " << max << "\nМинимальный элемент: " << min << "\n";

cout << "Отсортированный: ";

cout << chrono::duration\_cast<chrono::nanoseconds>(end - start).count() << " наносекунд\n";

min = A[0];

max = A[0];

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

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

if (A[i] > max)

max = A[i];

if (A[i] < min)

min = A[i];

}

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

cout << "Неотсортированный: ";

cout << chrono::duration\_cast<chrono::nanoseconds>(end - start).count() << " наносекунд\n\n";

}

void handleForCase4(int A[]) {

Shakersort(A);

int max = A[99], min = A[0];

double mid = round((max + min) / 2);

cout << "Среднее значение: " << mid << "\n";

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

int count = 0;

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

if (A[i] == mid)

count += 1;

}

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

cout << "Найдено " << count << " похожих чисел. Они находятся на позициях: ";

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

if (A[i] == mid)

cout << i + 1 << " ";

}

cout << "\nВремя поиска: ";

cout << chrono::duration\_cast<chrono::nanoseconds>(end - start).count() << " наносекунд\n\n";

}

void handleForCase5(int A[]) {

int x, count = 0;

cout << "Введите число X: ";

cin >> x;

Shakersort(A);

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

if (A[i] < x)

count += 1;

else break;

}

cout << "Кол-во элементов меньших " << x << " = " << count << "\n\n";

}

void handleForCase6(int A[]) {

int x, count = 0;

cout << "Введите число X: ";

cin >> x;

Shakersort(A);

for (int i = 99; i >= 0; i--) {

if (A[i] > x)

count += 1;

else break;

}

cout << "Кол-во элементов больших " << x << " = " << count << "\n\n";

}

void handleForCase7(int A[]) {

Shakersort(A);

int left = -99, right = 99;

int mid, key;

cout << "Введите искомое число: ";

cin >> key;

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

while (left < right) {

mid = (left + right) / 2;

if (A[mid] > key)

right = mid;

else

left = mid + 1;

}

right -= 1;

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

if (A[right] == key)

cout << "\nТакое число в массиве есть\n";

else

cout << "\nТакого числа в массиве нет\n";

cout << "Время поиска: ";

cout << chrono::duration\_cast<chrono::nanoseconds>(end - start).count() << " наносекунд\n";

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

bool k = false;

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

if (A[i] == key) {

k = true;

}

}

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

cout << "Время поиска перебором: ";

cout << chrono::duration\_cast<chrono::nanoseconds>(end - start).count() << " наносекунд\n\n";

}

void handleForCase8(int A[]) {

int x1, x2;

cout << "Введите 2 элемента, которые вы хотите поменять местами: ";

cin >> x1 >> x2;

cout << "\n";

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

swap(A[x1 - 1], A[x2 - 1]);

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

ViewArray(A);

cout << "Скорость обмена: ";

cout << chrono::duration\_cast<chrono::nanoseconds>(end - start).count() << " наносекунд\n\n";

}

int main() {

setlocale(0, "");

int input = 1;

int A[100];

srand(time(0));

cout << "Так выглядит изначальный массив:\n";

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

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

ViewArray(A);

cout << "\n";

while (input){

menu();

cin >> input;

cout << "\n";

switch (input) {

case 1: {

handleForCase1(A);

break;

}

case 2: {

handleForCase2(A);

break;

}

case 3: {

handleForCase3(A);

break;

}

case 4: {

handleForCase4(A);

break;

}

case 5: {

handleForCase5(A);

break;

}

case 6: {

handleForCase6(A);

break;

}

case 7: {

handleForCase7(A);

break;

}

case 8: {

handleForCase8(A);

break;

}

case 9: {

TwoArray(A);

break;

}

case 0: {

exit;

}

default: {

cout << "Что-то снова пошло не так, давай заново. \n\n";

}

}

}

}