# 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. 3372 | Шорсткая А.А. |
| Преподаватель | Глущенко А. Г. |

Санкт-Петербург 2023

# Цель работы.

Разработать программу, позволяющая выполнять различные операции с массивами, включая сортировку и поиск.

# Основные теоретические положения.

Массив – структура данных, хранящая набор значений, идентифицируемых по индексу или набору индексов, принимающих целые значения из некоторого заданного непрерывного диапазона. Одномерный массив можно рассматривать как реализацию абстрактного типа данных. Подробная информация о массиве представлена на рисунке 1.



Рисунок 1 – Одномерный массив

Сортировка массива – это расположение элементов массива в некотором заданном порядке (по возрастанию или убыванию). Например, на рисунке 2 представлена сортировка пузырьком.

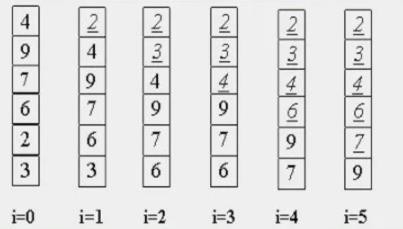


Рисунок 2 – Сортировка пузырьком В данной работе представлены следующие сортировки:

* сортировка полным перебором (от меньше к большему) – это сортировка выполняющая полный перебор всех элементов массива сравнивая и переставляя их при удовлетворенности условий;
* «быстрая сортировка» – один из самых популярных алгоритмов, в котором используется принцип «разделяй и властвуй»;
* сортировка «слиянием» – это сортировка, где мы делится массив пополам, каждый из них сортируется слиянием и потом соединяются оба массива.

Поиск элемента в массиве дает возможность удостоверится о наличии данного элемента в части массива.

В неотсортированном массиве поиск элемента необходимо просматривать весь массив (линейный поиск).

В отсортированном массиве поиск элемента можно осуществлять, не просматривая весь массив. Например, бинарный поиск представлен на рисунке 3.

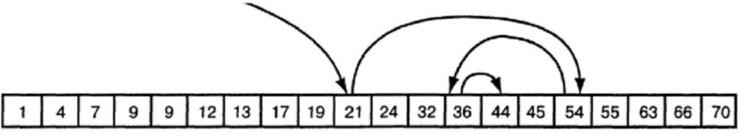


Рисунок 3 – Бинарный поиск

# Постановка задачи.

Разработать алгоритм и написать программу, которая позволяет:

1. Создать 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. Подсчитать время поиска этих элементов в отсортированном массиве и неотсортированном, используя библиотеку chrono.
4. Выводить среднее значение (если необходимо, число нужно округлить) максимального и минимального значения в отсортированном и неотсортированном. Выводить индексы всех элементов, которые равны этому значению, и их количество. Подсчитать время поиска.
5. Выводить количество элементов в отсортированном массиве, которые меньше числа a, которое инициализируется пользователем.
6. Выводить 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. Менять местами элементы массива, индексы которых вводит пользователь. Выводить скорость обмена, используя библиотеку chrono.

# Выполнение работы.

Код программы представлен в приложении А.

# Выводы.

Данная практическая работа позволяет сформировать навыки и умения работы с массивами, усвоить принцип сортировки и поиска, сравнить время работы различных функций и методов.

# Ход работы

1. При запуске программы необходимо выбрать номер действия (рисунок 1).

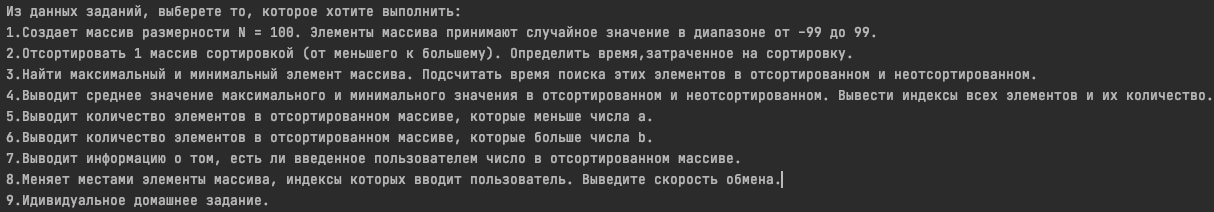


Рисунок 1 – Выбор номер поведения программы

1. При выборе пункта «1» выводится массив размерности N = 100. Элементы массива принимают случайное значение в диапазоне от -99 до 99 (рисунок 2).

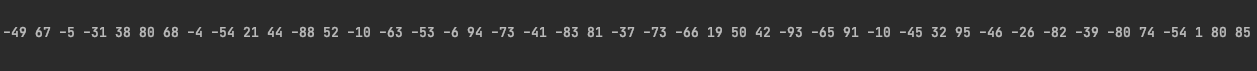


Рисунок 2 – Вывод массива

1. При выборе пункта «2» осуществляется сортировка «пузырьком» и отображение время выполнения сортировки (рисунок 3).

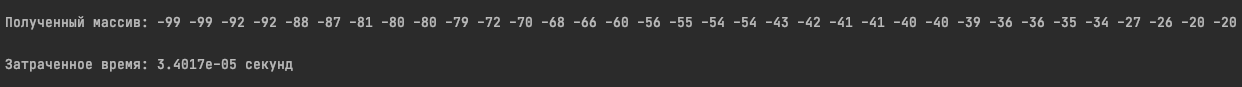


Рисунок 3 – Сортировка «пузырьком»

1. При выборе пункта «3» осуществляется поиск минимального и максимального элемента в неотсортированном и отсортированном массиве и расчет времени этого поиска (рисунок 4).

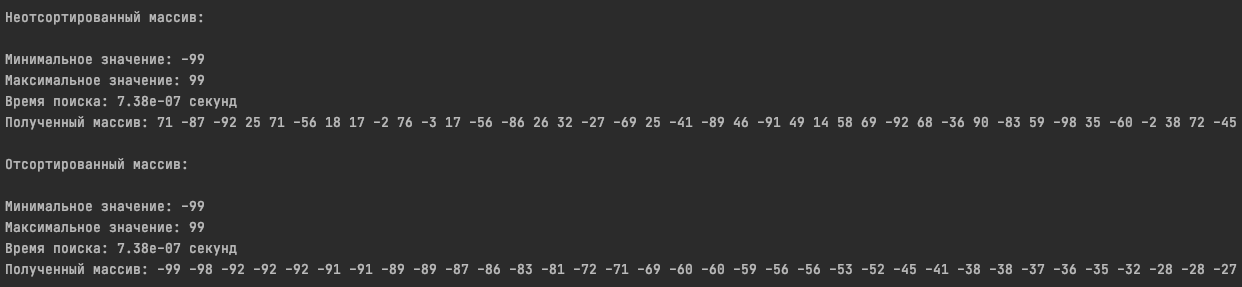


Рисунок 4 – Сортировка «слиянием»

1. При выборе пункта «4» отображается среднее минимального и максимального элемента в неотсортированном и отсортированном массиве, также выводятся индексы всех элементов и их количество и время поиска (рисунок 5)

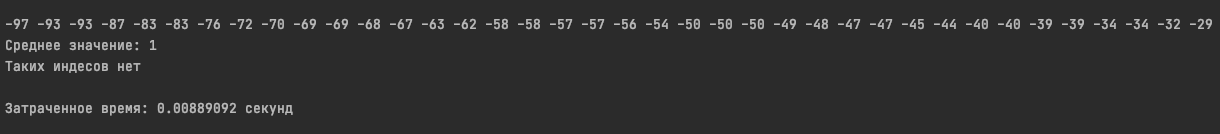


Рисунок 5 – Подсчет и поиск среднего в массиве в неотсортированном и отсортированном массиве

6. При выборе пункта «5» и «6» отображается количество элементов массива, меньше и больше введенного значения пользователя соответственно (рисунок 6,7).

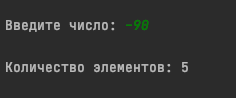


Рисунок 6 – Отображение количества элементов, меньше введённого значения

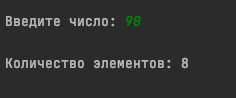


Рисунок 7 – Отображение количества элементов, больше введённого значения

1. При выборе пункта «7» выводится информацию о том, есть ли введенное пользователем число в отсортированном массиве, также сравнивает скорость работы бинарного поиска с обычным перебором (рисунок 8)

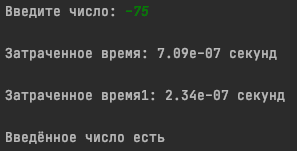


Рисунок 8 – Отображение информации о том, что есть введённое число в массиве и сравнение скорости работы бинарного поиска с обычным перебором

1. При выборе пункта «8» осуществляется смена элементов массива при помощи индексов, введенных пользователем и выводится скорость обмена (рисунок 9)

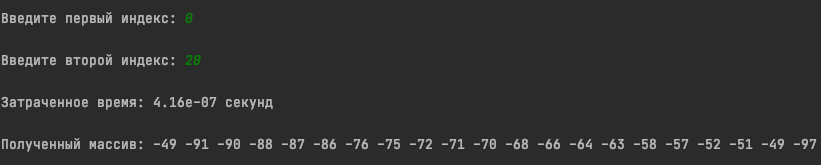


Рисунок 9 – Смена элементов

# ПРИЛОЖЕНИЕ А РАБОЧИЙ КОД

Листинг А.1 – Рабочий код программы

#include <iostream>

#include <chrono>

using namespace std;

int binarySearch(int arr[], int value, int start, int end) {

if (end >= start) {

int mid = start + (end - start) / 2;

if (arr[mid] == value) {

return mid;

}

if (arr[mid] > value) {

return binarySearch(arr, value, start, mid - 1);

}

return binarySearch(arr, value, mid + 1, end);

}

return -1;

}

void shell(int arr[], int size) {

int step = size / 2;

while (true) {

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

for (int j = i; j < size; j += step) {

for (int tempj = j; tempj >= step; tempj -= step) {

if (arr[tempj] < arr[tempj - step]) {

swap(arr[tempj], arr[tempj - step]);

}

else {

break;

}

}

}

}

if (step == 1) {

break;

}

step /= 2;

}

}

int main() {

setlocale(LC\_ALL, "ru");

srand(time(NULL));

constexpr int N = 100;

int arr[N] = {};

int choose = 0;

int maxnots, minnots, maxs, mins, a, b, k, k2, sum, Index1, Index2;

auto start = chrono::high\_resolution\_clock::now();

auto end = chrono::high\_resolution\_clock::now();

chrono::duration<float> durationTime = end - start;

while (choose != -1) {

cout << "\n\n\nВыберите задание, которое хотите выполнить: \n"

"1.Создает целочисленный массив размерности N = 100. Элементы массива принимают случайное значение в диапазоне от -99 до 99.\n"

"2.Отсортировать 1 массив сортировкой (от меньшего к большему). Определить время,затраченное на сортировку.\n"

"3.Найти максимальный и минимальный элемент массива. Подсчитать время поиска этих элементов в отсортированном и неотсортированном.\n"

"4.Выводит среднее значение максимального и минимального значения в отсортированном и неотсортированном. Вывести индексы всех элементов и их количество. Подсчитать время поиска.\n"

"5.Выводит количество элементов в отсортированном массиве, которые меньше числа a.\n"

"6.Выводит количество элементов в отсортированном массиве, которые больше числа b.\n"

"7.Выводит информацию о том, есть ли введенное пользователем число в отсортированном массиве.\n"

"8.Меняет местами элементы массива, индексы которых вводит пользователь. Выведите скорость обмена.\n"

"9.ИДЗ.\n\n\n";

cout << "Задание: ";

cin >> choose;

cout << "\n";

if (choose != 1 and choose != 2 and choose != 3 and choose != 4 and choose != 5 and choose != 6 and choose != 7

and choose != 8 and choose != 9)

cout << "Такого номера задания нет \n";

else

int arr[N];

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

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

}

switch (choose) {

case 1:

cout << '\n' << "Полученный массив: ";

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

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

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

}

cout << '\n' << '\n';

break;

case 2:

start = chrono::high\_resolution\_clock::now();

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

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

if (arr[j] > arr[j + 1]) {

int arr1 = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = arr1;

}

}

}

end = chrono::high\_resolution\_clock::now();

durationTime = end - start;

cout << "Полученный массив: ";

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

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

}

cout << '\n' << '\n' << "Затраченное время: " << durationTime.count() << " секунд" << '\n' << '\n';

break;

case 3:

maxnots = arr[0];

minnots = arr[0];

start = chrono::high\_resolution\_clock::now();

for (int i = 1; i < N; ++i) {

if (arr[i] > maxnots) {

maxnots = arr[i];

}

if (arr[i] < minnots) {

minnots = arr[i];

}

}

end = chrono::high\_resolution\_clock::now();

durationTime = end - start;

cout << '\n' << "Неотсортированный массив: " << '\n' << '\n' << "Минимальное значение: " \

<< minnots << '\n' << "Максимальное значение: " << maxnots << '\n' \

<< "Время поиска: " << durationTime.count() << " секунд" << '\n';

cout << "Полученный массив: ";

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

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

}

cout << '\n' << '\n';

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

for (int j = 0; j < N - 1; j++) {

if (arr[j] > arr[j + 1]) {

int arr2 = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = arr2;

}

}

}

start = chrono::high\_resolution\_clock::now();

maxs = arr[99];

mins = arr[0];

end = chrono::high\_resolution\_clock::now();

durationTime = end - start;

cout << "Отсортированный массив: " << '\n' << '\n' << "Минимальное значение: " \

<< mins << '\n' << "Максимальное значение: " << maxs << '\n' \

<< "Время поиска: " << durationTime.count() << " секунд" << '\n';

cout << "Полученный массив: ";

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

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

}

cout << '\n' << '\n';

break;

case 4:

sum = 0;

k = 0;

maxnots = arr[0];

minnots = arr[0];

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

if (arr[i] > maxnots) {

maxnots = arr[i];

}

if (arr[i] < minnots) {

minnots = arr[i];

}

}

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

for (int j = 0; j < N - 1; j++) {

if (arr[j] > arr[j + 1]) {

int arr3 = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = arr3;

}

}

}

sum = ((maxnots + minnots) / 2);

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

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

}

cout << '\n' << "Среднее значение: " << sum << '\n';

start = chrono::high\_resolution\_clock::now();

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

if (arr[i] == sum) {

k += 1;

cout << "Индекс элемента равный среднему значению: " << i << endl;

}

}

if (k == 0) {

cout << "Таких индесов нет" << endl;

}

else

cout << "Количество: " << k << endl;

end = chrono::high\_resolution\_clock::now();

durationTime = end - start;

cout << "Затраченное время: " << durationTime.count() << " секунд" << '\n' << '\n';

break;

case 5:

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

for (int j = 0; j < N - 1; j++) {

if (arr[j] > arr[j + 1]) {

int arr4 = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = arr4;

}

}

}

cout << '\n' << "Введите число: ";

cin >> a;

k = 0;

while (arr[k] < a) {

k += 1;

}

cout << '\n' << "Количество элементов: " << k << '\n' << '\n';

break;

case 6:

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

for (int j = 0; j < N - 1; j++) {

if (arr[j] > arr[j + 1]) {

int arr4 = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = arr4;

}

}

}

cout << '\n' << "Введите число: ";

cin >> b;

k2 = 99;

while (arr[k2] > b) {

k2 -= 1;

}

cout << "Количество элементов: " << 99 - k2 << '\n' << '\n';

break;

case 7: {

int choice = 0;

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

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

if (arr[j] > arr[j + 1]) {

int arr1 = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = arr1;

}

}

}

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

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

}

cout << '\n';

cin >> choice;

cout << '\n';

start = chrono::high\_resolution\_clock::now();

int is\_in\_array = binarySearch(arr, choice, 0, N);

end = chrono::high\_resolution\_clock::now();

cout << '\n' << "Затраченное время: " << durationTime.count() << " секунд" << '\n' << '\n';

const auto start1 = chrono::high\_resolution\_clock::now();

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

if (choice == arr[i] || i == N - 1)

{

const auto end1 = chrono::high\_resolution\_clock::now();

const std::chrono::duration<float> durationTime1 = end1 - start1;

cout << '\n' << "Затраченное время1: " << durationTime1.count() << " секунд" << '\n' << '\n';

break;

}

}

if (is\_in\_array != -1)

cout << "находится в массиве\n";

else

cout << "не находится в массиве\n";

break;

}

case 8:

cout << '\n' << "Введите первый индекс: ";

cin >> Index1;

cout << '\n' << "Введите второй индекс: ";

cin >> Index2;

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

for (int j = 0; j < N - 1; j++) {

if (arr[j] > arr[j + 1]) {

int arr4 = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = arr4;

}

}

}

start = chrono::high\_resolution\_clock::now();

swap(arr[Index1], arr[Index2]);

end = chrono::high\_resolution\_clock::now();

durationTime = end - start;

cout << '\n' << "Затраченное время: " << durationTime.count() << " секунд" << '\n';

cout << '\n' << "Полученный массив: ";

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

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

}

cout << '\n';

break;

case 9:

int tempArr[N];

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

tempArr[i] = arr[i];

}

cout << "Бабл сорт\n";

start = chrono::high\_resolution\_clock::now();

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

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

if (tempArr[j] > tempArr[j + 1]) {

int arr1 = tempArr[j];

tempArr[j] = tempArr[j + 1];

tempArr[j + 1] = arr1;

}

}

}

end = chrono::high\_resolution\_clock::now();

durationTime = end - start;

cout << "Полученный массив: ";

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

cout << tempArr[i] << " ";

}

cout << '\n' << '\n' << "Затраченное время: " << durationTime.count() << " секунд" << '\n' << '\n';

cout << "Шелл сорт\n";

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

tempArr[i] = arr[i];

}

start = chrono::high\_resolution\_clock::now();

shell(tempArr, N);

end = chrono::high\_resolution\_clock::now();

durationTime = end - start;

cout << "Полученный массив: ";

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

cout << tempArr[i] << " ";

}

cout << '\n' << '\n' << "Затраченное время: " << durationTime.count() << " секунд" << '\n' << '\n';

break;

}

}

}