**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра информационных систем**

отчет

**по практической работе №1**

**по дисциплине «Программирование»**

Тема: ОДНОМЕРНЫЕ СТАТИЧЕСКИЕ МАССИВЫ

|  |  |  |
| --- | --- | --- |
| Студентка гр. 4373 | Некрасова А.В. |  |
| Преподаватель | Глущенко А. Г. |  |

Санкт-Петербург

2024

**Цель работы.**

Изучение работы массивов и их внутреннего представления в памяти; сортировки; получение практических навыков работы с массивами.

**Основные теоретические положения.**

При использовании простых переменных каждой области памяти для хранения данных соответствует свое имя. Если с группой величин одинакового типа требуется выполнить однообразные действия, им дают одно имя, а различают по порядковому номеру (индексу). Это дает возможность компактно записать множество операций с использованием циклов.

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.

Все массивы можно разделить на две группы: одномерные и многомерные. Описание массива в программе отличается от объявления обычной переменной наличием размерности массива, которая задается в квадратных скобках после имени.

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.

**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 в отсортированном и неотсортированном. Выводит индексы всех элементов, которые равны этому значению, и их количество. Подсчитайте время поиска.

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.**

**#include <iostream>**

**#include <chrono>**

**#include <cmath>**

**#include <vector>**

**using namespace std;**

**void BubbleSort(int arr[], int n) {**

**int start = 0, end = n - 1;**

**for (int i = start; i < end; ++i)**

**for (int j = start; j < end - i; ++j)**

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

**swap(arr[j], arr[j + 1]);**

**}**

**void ShakerSort(int arr[], int n) {**

**bool swapped = true;**

**int start = 0, end = n - 1;**

**while (swapped) {**

**swapped = false;**

**for (int i = start; i < end; ++i) { // Проход слева направо**

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

**swap(arr[i], arr[i + 1]);**

**swapped = true;**

**}**

**}**

**if (!swapped) {**

**break;**

**}**

**swapped = false;**

**--end;**

**for (int i = end - 1; i >= start; --i) { // Проход справа налево**

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

**swap(arr[i], arr[i + 1]);**

**swapped = true;**

**}**

**}**

**++start;**

**}**

**}**

**void CombSort(int arr[], int n) {**

**int start = 0, end = n - 1, count = 0;**

**float k = 1.247;**

**while (end >= 1)**

**{**

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

**{**

**if (arr[i] > arr[int(i + end)])**

**swap(arr[int(i + end)], arr[i]);**

**}**

**end /= k;**

**}**

**while (true){**

**for (int i = 0; i < 99; i++){**

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

**swap(arr[int(i + end)], arr[i]);**

**}**

**else count++;**

**}**

**if (count == 99)**

**break;**

**else**

**count = 0;**

**}**

**}**

**void InsertionSort(int arr[], int n) {**

**int i, key, j;**

**for (i = 1; i < n; i++) {**

**key = arr[i];**

**j = i - 1;**

**while (j >= 0 && arr[j] > key) {**

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

**j = j - 1;**

**}**

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

**}**

**}**

**void QuickSort(int\* arr, int end, int begin) {**

**int mid;**

**int f = begin;**

**int l = end;**

**mid = arr[(f + l) / 2];**

**while (f < l){**

**while (arr[f] < mid) f++;**

**while (arr[l] > mid) l--;**

**if (f <= l){**

**swap(arr[f], arr[l]);**

**f++;**

**l--;**

**}**

**}**

**if (begin < l) QuickSort(arr, l, begin);**

**if (f < end) QuickSort(arr, end, f);**

**}**

**void PrintArray(int arr[], int n) {**

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

**std::cout << arr[i] << " ";**

**}**

**std::cout << std::endl;**

**}**

**void ShellSort(int arr[]) {**

**int n = 100;**

**for (int gap = n / 2; gap > 0; gap /= 2) {**

**for (int i = gap; i < n; i++) {**

**int temp = arr[i];**

**int j;**

**for (j = i; j >= gap && arr[j - gap] > temp; j -= gap) {**

**arr[j] = arr[j - gap];**

**}**

**arr[j] = temp;**

**}**

**}**

**}**

**int main()**

**{**

**setlocale(0, "");**

**srand(time(0)); //чтобы случайные числа были каждый раз разные**

**int massiv[100], massiv0[100], sort, loop = 1;**

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

**massiv[i] = rand() % (199) - 99; //присваиваем массиву рандомные значения**

**cout << "Массив: ";**

**PrintArray(massiv, 100);**

**copy(massiv, massiv + 100, massiv0);**

**do {**

**cout << "\nВыберете сортировку массива: 1) BubbleSort 2) ShakerSort 3) CombSort 4) InsertionSort 5) QuickSort 6) ShellSort(идз)\nВаш выбор: ";**

**cin >> sort;**

**if (sort == 1) {**

**auto start = chrono::high\_resolution\_clock::now(); // Начало измерения времени**

**BubbleSort(massiv, 100);**

**auto end = chrono::high\_resolution\_clock::now(); // Конец измерения времени**

**cout << "Отсортированный массив (BubbleSort): ";**

**PrintArray(massiv, 100);**

**chrono::duration<double, std::milli> duration = end - start;**

**cout << "Время сортировки: " << duration.count() << " миллисекунд" << std::endl;**

**copy(massiv0, massiv0 + 100, massiv);**

**}**

**if (sort == 2) {**

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

**ShakerSort(massiv, 100);**

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

**cout << '\n' << "Отсортированный массив (ShakerSort): ";**

**PrintArray(massiv, 100);**

**chrono::duration<double, std::milli> duration = end - start;**

**cout << "Время сортировки: " << duration.count() << " миллисекунд" << std::endl;**

**copy(massiv0, massiv0 + 100, massiv);**

**}**

**if (sort == 3) {**

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

**CombSort(massiv, 100);**

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

**cout << '\n' << "Отсортированный массив (CombSort): ";**

**PrintArray(massiv, 100);**

**chrono::duration<double, std::milli> duration = end - start;**

**cout << "Время сортировки: " << duration.count() << " миллисекунд" << std::endl;**

**copy(massiv0, massiv0 + 100, massiv);**

**}**

**if (sort == 4) {**

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

**InsertionSort(massiv, 100);**

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

**cout << '\n' << "Отсортированный массив (InsertionSort): ";**

**PrintArray(massiv, 100);**

**chrono::duration<double, std::milli> duration = end - start;**

**cout << "Время сортировки: " << duration.count() << " миллисекунд" << std::endl;**

**copy(massiv0, massiv0 + 100, massiv);**

**}**

**if (sort == 5) {**

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

**QuickSort(massiv, 99, 0);**

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

**cout << '\n' << "Отсортированный массив (QuickSort): ";**

**PrintArray(massiv, 100);**

**chrono::duration<double, std::milli> duration = end - start;**

**cout << "Время сортировки: " << duration.count() << " миллисекунд" << std::endl;**

**copy(massiv0, massiv0 + 100, massiv);**

**}**

**if (sort == 6) {**

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

**ShellSort(massiv);**

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

**cout << '\n' << "Отсортированный массив (ShellSort): ";**

**PrintArray(massiv, 100);**

**chrono::duration<double, std::milli> duration = end - start;**

**cout << "Время сортировки: " << duration.count() << " миллисекунд" << std::endl;**

**copy(massiv0, massiv0 + 100, massiv);**

**}**

**cout << "Будем применять к циклу другие сортировки? Да - 1, нет - 0. Ваш вариант: ";**

**cin >> loop;**

**} while (loop == 1);**

**int maxmassiv = -101, minmassiv = 101, sredmassiv, kmassiv = 0, kmassiv0 = 0;**

**QuickSort(massiv, 99, 0); //РАБОТА С ОТСОРТИРОВАННЫМ МАССИВОМ (massiv)**

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

**maxmassiv = massiv[99];**

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

**chrono::duration<double, std::milli> duration1 = end1 - start1;**

**auto start2 = chrono::high\_resolution\_clock::now();**

**minmassiv = massiv[0];**

**auto end2 = chrono::high\_resolution\_clock::now();**

**chrono::duration<double, std::milli> duration2 = end2 - start2;**

**sredmassiv = (maxmassiv + minmassiv) / 2;**

**cout << "\nСреднее значение максимального и минимального чисел в массиве: " << std::round(sredmassiv) << "\n";**

**cout << "\nРАБОТА С ОТСОРТИРОВАННЫМ МАССИВОМ\nВремя нахождения максимального эл - та: " << duration1.count() << " миллисекунд" << std::endl;**

**cout << "Время нахождения минимального эл - та: " << duration2.count() << " миллисекунд" << std::endl;**

**cout << "Индексы эл - ов, которые равны среднему значению: ";**

**auto start3 = chrono::high\_resolution\_clock::now();**

**for (int i = 0; i < 100 && massiv[i] <= sredmassiv; i++) {**

**if (massiv[i] == sredmassiv) {**

**cout << i << ' ';**

**++kmassiv;**

**}**

**}**

**auto end3 = chrono::high\_resolution\_clock::now();**

**chrono::duration<double, std::milli> duration3 = end3 - start3;**

**cout << "\nВремя поиска индексов: " << duration3.count() << " миллисекунд\n";**

**cout << "Кол - во эл - ов, равных среднему значению: " << kmassiv << "\n";**

**int maxmassiv0 = massiv0[0], minmassiv0 = massiv0[0];**

**auto start4 = chrono::high\_resolution\_clock::now(); //РАБОТА С НЕОТСОРТИРОВАННЫМ МАССИВОМ (massiv0)**

**for (int i = 1; i < 100; ++i) {**

**if (massiv0[i] > massiv0[i + 1])**

**maxmassiv0 = massiv0[i];**

**}**

**auto end4 = chrono::high\_resolution\_clock::now();**

**chrono::duration<double, std::milli> duration4 = end4 - start4;**

**cout << "\nРАБОТА С НЕОТСОРТИРОВАННЫМ МАССИВОМ\nВремя нахождения максимального эл - та: " << duration4.count() << " миллисекунд" << std::endl;**

**auto start5 = chrono::high\_resolution\_clock::now();**

**for (int i = 1; i < 100; ++i) {**

**if (massiv0[i] < massiv0[i + 1])**

**minmassiv0 = massiv0[i];**

**}**

**auto end5 = chrono::high\_resolution\_clock::now();**

**chrono::duration<double, std::milli> duration5 = end5 - start5;**

**cout << "Время нахождения минимального эл - та: " << duration5.count() << " миллисекунд" << std::endl;**

**cout << "Индексы эл - ов, которые равны среднему значению: ";**

**auto start6 = chrono::high\_resolution\_clock::now();**

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

**if (massiv0[i] == sredmassiv) {**

**cout << i << ' ';**

**++kmassiv0;**

**}**

**}**

**auto end6 = chrono::high\_resolution\_clock::now();**

**chrono::duration<double, std::milli> duration6 = end6 - start6;**

**cout << "\nВремя поиска индексов: " << duration6.count() << " миллисекунд" << std::endl;**

**cout << "Кол - во эл - ов, равных среднему значению: " << kmassiv0 << "\n";**

**//ЗАДАНИЕ 5**

**int a, b, target, kolmina = 0, kolmaxb = 0, loop1 = 1, loop2 = 1, loop3 = 1;**

**do {**

**cout << "\nЗадание 5. Введите число a (выведем кол - во элементов, которые меньше числа a): ";**

**cin >> a;**

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

**if (massiv[i] < a)**

**kolmina++;**

**}**

**cout << "Кол - во эл - ов, меньших заданного числа a: " << kolmina;**

**kolmina = 0;**

**cout << "\nБудем вводить число a еще? Да - 1, нет - 0. Ваш вариант: ";**

**cin >> loop1;**

**} while (loop1 == 1);**

**//ЗАДАНИЕ 6**

**do {**

**cout << "\nЗадание 6. Введите число b (выведем кол - во элементов, которые больше числа b): ";**

**cin >> b;**

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

**if (massiv[i] > b)**

**kolmaxb++;**

**}**

**cout << "Кол - во эл - ов, больших заданного числа b: " << kolmaxb;**

**kolmaxb = 0;**

**cout << "\nБудем вводить число b еще? Да - 1, нет - 0. Ваш вариант: ";**

**cin >> loop2;**

**} while (loop2 == 1);**

**//ЗАДАНИЕ 7**

**do {**

**cout << "\nЗадание 7. Введите число для поиска: ";**

**cin >> target;**

**int left = 0, right = 99, flag = 0;**

**auto start7 = chrono::high\_resolution\_clock::now();**

**while (left <= right) {**

**int mid = left + (right - left) / 2;**

**if (massiv[mid] == target) {**

**cout << "Элемент найден";**

**flag = 1;**

**break;**

**}**

**if (massiv[mid] < target) {**

**left = mid + 1; // Ищем в правой половине**

**}**

**else**

**right = mid - 1; // Ищем в левой половине**

**}**

**if (flag == 0)**

**cout << "Элемент не найден";**

**auto end7 = chrono::high\_resolution\_clock::now();**

**chrono::duration<double, std::milli> duration7 = end7 - start7;**

**auto start8 = chrono::high\_resolution\_clock::now();**

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

**if (massiv[i] == target)**

**break;**

**}**

**auto end8 = chrono::high\_resolution\_clock::now();**

**chrono::duration<double, std::milli> duration8 = end8 - start8;**

**cout << "\nВремя методом бинарного поиска: " << duration7.count() << " миллисекунд";**

**cout << "\nВремя методом перебора: " << duration8.count() << " миллисекунд";**

**cout << "\nБудем вводить число для поиска еще? Да - 1, нет - 0. Ваш вариант: ";**

**cin >> loop3;**

**} while (loop3 == 1);**

**//ЗАДАНИЕ 8**

**int indeks1, indeks2, bober = 1, loop4;**

**do {**

**cout << "\nЗадание 8. Чтобы поменять элементы местами, введите первый индекс: ";**

**cin >> indeks1;**

**cout << "Введите второй индекс: ";**

**cin >> indeks2;**

**cout << "Массив после опреации обмена эл - ов:\n";**

**auto start9 = chrono::high\_resolution\_clock::now();**

**bober = massiv[indeks1];**

**massiv[indeks1] = massiv[indeks2];**

**massiv[indeks2] = bober;**

**auto end9 = chrono::high\_resolution\_clock::now();**

**chrono::duration<double, std::milli> duration9 = end9 - start9;**

**PrintArray(massiv, 100);**

**cout << "Время обмена эл - ов местами: " << duration9.count() << " миллисекунд";**

**cout << "\nБудем еще менять элементы местами? Да - 1, нет - 0. Ваш вариант: ";**

**cin >> loop4;**

**} while (loop4 == 1);**

**}**

**Выводы.**

Я изучила массивы и поняла, как они представлены в памяти компьютера, научилась работать с различными сортировками.