**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. | Каекбердин Д. Р. |  |
| 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 был написан алгоритм для заполнения массива случайными числами, не выходящими за пределы [-99; 99]. Для второго пункта была реализована сортировка “Insert sort”. Для третьего пункта алгоритм поиска максимального и минимального элементов. Четвертый пункт выполнен с помощью алгоритмов нахождения среднего арифметического и поиска индексов элементов с массива, которые равные среднему значению. В пятом пункте написан алгоритм, который считает элементы отсортированного массива меньше числа введенного пользователем. В шестом пункте почти такой же алгоритм, но он ищет елементы больше введенного числа. Седьмой пункт не реализован. В восьмом пункте пользователь поменять элементы отсортированного и не отсортированного массивов, введя индексы этих элементов.

Блок скриншотов работы программы



**Выводы.**

Изучен навык работы с одномерными статическими массивами. Изучены базовые алгоритмы сортировок и поиска.

Приложение А

рабочий код

#include <iostream>  
#include <chrono>  
  
using namespace std;  
using namespace chrono;  
  
  
int main() {  
 *// Выбор среды для локализации русского языка* while (true)  
 {  
 int Environment;  
 cout << "1) Clion\n2) Visual Studio\nSelect the development environment number: ";  
 cin >> Environment;  
 if (Environment == 1) {  
 setlocale(**LC\_ALL**, "ru\_RU.UTF-8");  
 system("chcp 65001");  
 break;  
 }  
 if (Environment == 2) {  
 setlocale(0, "Rus");  
 break;  
 }  
 else  
 cout << "\n???\n";  
 }  
  
 time\_point<steady\_clock, duration<**\_\_int64**, ratio<1, 1000000000>>> start, end;  
 nanoseconds result;  
 int Point = 10;  
 int x, j, Max, Min, Mid, Quantity, a, b, index\_1, index\_2;  
 int Nums[100], Sort\_Nums[100];  
  
 while (true)  
 {  
 *// Задание 1* if (Point == 10 || Point == 1)  
 {  
 srand(time(0));  
  
 for (int i = 0; i < 100; i++) {  
 Nums[i] = rand() % 199 - 99; *// диапазон [-99; 99]* Sort\_Nums[i] = Nums[i];  
 }  
 cout << "Задание 1\nЦелочисленный масcив: [";  
 for (int i = 0; i < 100; i++) {  
 cout << Nums[i] << " ";  
 }  
 cout << "]\n\n";  
 }  
  
 *// Задание 2* if (Point == 10 || Point == 2)  
 {  
 start = steady\_clock::now(); *// Начальный момент времени  
  
 // Cортировка "Insert sort"* for (int i = 1; i < 100; i++) {  
 x = Sort\_Nums[i];  
 j = i;  
 while ((j > 0) && (x < Sort\_Nums[j - 1])) {  
 Sort\_Nums[j] = Sort\_Nums[j - 1];  
 j--;  
 }  
 Sort\_Nums[j] = x;  
 }  
  
 end = steady\_clock::now(); *// Конечный момент времени* result = duration\_cast<nanoseconds>(end - start);  
 cout << "Задание 2\nОтсортированный масcив: [";  
 for (int i = 0; i < 100; i++) {  
 cout << Sort\_Nums[i] << " ";  
 }  
 cout << "]\n";  
 cout << "Время сортировки: " << result.count() << " (наносек.)\n\n";  
 }  
  
 *// Задание 3* if (Point == 10 || Point == 3)  
 {  
 Min = Max = Sort\_Nums[0];  
 start = steady\_clock::now();  
 for (int i = 1; i < 100; i++) { *// Поиск максимального и минимального значений* if (Max < Sort\_Nums[i])  
 Max = Sort\_Nums[i];  
 if (Min > Sort\_Nums[i])  
 Min = Sort\_Nums[i];  
 }  
 cout << "Задание 3\nМаксимальный элемент: " << Max << "\nМинимальный элемет: " << Min;  
 end = steady\_clock::now();  
 result = duration\_cast<nanoseconds>(end - start);  
 cout << "\nВремя поиска макс. и мин. элементов в отсортированном массиве: " << result.count()  
 << " (наносек.)\n\n";  
  
 Min = Max = Nums[0];  
  
 start = steady\_clock::now();  
 for (int i = 1; i < 100; i++) {  
 if (Max < Nums[i])  
 Max = Nums[i];  
 if (Min > Nums[i])  
 Min = Nums[i];  
 }  
 cout << "Максимальный элемент: " << Max << "\nМинимальный элемет: " << Min;  
 end = steady\_clock::now();  
 result = duration\_cast<nanoseconds>(end - start);  
 cout << "\nВремя поиска макс. и мин. элементов в не отсортированном массиве: " << result.count()  
 << " (наносек.)\n\n";  
 }  
  
 *// Задание 4* if (Point == 10 || Point == 4)  
 {  
 Quantity = 0;  
 Mid = (Max + Min); *// Поиск среднего значения* if (Mid % 2 > 0) {  
 Mid += 1;  
 }  
 Mid /= 2;  
 cout << "Задание 4\nСреднее значение макс. и мин. элементов массива:" << Mid  
 << "\nИндексы всех элементов, равные среднему значению: [ ";  
  
 for (int i = 20; i < 100; i++) { *// Поиск индексов элементов и их кол-ва* if (Sort\_Nums[i] == Mid) {  
 cout << i << " ";  
 Quantity += 1;  
 }  
 if (Sort\_Nums[i] > Mid) {  
 break;  
 }  
 }  
 cout << "] \nКоличество эелементов: " << Quantity << "\n\n";  
 }  
  
 *// Задание 5* if (Point == 10 || Point == 5)  
 {  
 Quantity = 0;  
 cout << "Задание 5\nВведите целое число: ";  
 cin >> a;  
  
 for (int i = 0; i < 100; i++) { *// Поиск кол-ва элементов, которые меньше а* if (Sort\_Nums[i] < a) {  
 Quantity += 1;  
 }  
 if (Sort\_Nums[i] >= a) {  
 break;  
 }  
 }  
 cout << "\nКоличество элементов, которые меньше введенного числа: " << Quantity << "\n\n";  
 }  
  
 *// Задание 6* if (Point == 10 || Point == 6)  
 {  
 Quantity = 0;  
 cout << "Задание 6\nВведите целое число: ";  
 cin >> b;  
  
 for (int i = 0; i < 100; i++) { *// Поиск кол-ва элементов, которые больше b* if (Sort\_Nums[i] > b) {  
 Quantity += 1;  
 }  
 }  
 cout << "\nКоличество элементов, которые больше введенного числа: " << Quantity << "\n\n";  
 }  
  
 *// Задание 7* if (Point == 10 || Point == 7)  
 {  
 cout << "Введите индексы элементов массива, которые хотите поменять: ";  
 cin >> index\_1 >> index\_2;  
  
 *// Замена элементов у не отсортированного массива* start = steady\_clock::now();  
 x = Nums[index\_1];  
 Nums[index\_1] = Nums[index\_2];  
 Nums[index\_2] = x;  
 end = steady\_clock::now();  
 result = duration\_cast<nanoseconds>(end - start);  
 cout << "\nВремя замены у не отсортированного массива: " << result.count() << " (наносек.)\n";  
 cout << "Итог: [";  
 for (int i = 0; i < 100; i++) {  
 cout << Nums[i] << " ";  
 }  
 cout << "]\n\n";  
  
 *// Замена элементов у отсортированного массива* start = steady\_clock::now();  
 x = Sort\_Nums[index\_1];  
 Sort\_Nums[index\_1] = Sort\_Nums[index\_2];  
 Sort\_Nums[index\_2] = x;  
 end = steady\_clock::now();  
 result = duration\_cast<nanoseconds>(end - start);  
 cout << "\nВремя замены у отсортированного массива: " << result.count() << " (наносек.)\n";  
 cout << "Итог: [";  
 for (int i = 0; i < 100; i++) {  
 cout << Sort\_Nums[i] << " ";  
 }  
 cout << "]\n\n";  
 }  
  
 *// Меню* cout << "\t\tМеню\n1) Создать новый массив.\n2) Отсортировать созданный массив.\n3) Найти максимальный и мнимальный элементы в двух массивах.\n4) Среднее значение максимального и минимального значений и индексы элементов равные ему.\n5) Вывести количество элементов, которые меньше введенного числа.\n6) Вывести количество элементов, которые больше введенного числа.\n7) Поменять местами элементы массивов.\n0) Выход.\n";  
 while (true) {  
 cout << "\nВведите номер пункта, который хотите запустить: ";  
 cin >> Point;  
 if (Point >= 0 & Point <= 7)  
 {  
 cout << "\nЗапускаю пункт под номером: " << Point << "\n\n";  
 break;  
 }  
 else  
 cout << "\nНе понимаю...\n";  
 }  
  
 *// Выход* if (Point == 0)  
 {  
 cout << "\nВыхожу";  
 break;  
 }  
 }  
}