**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. 2372 |  | Соколовский В.Д. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2022

**Цель работы.**

Изучение одномерных статических массивов и основных алгоритмов работы с ними. Написание программы, заполняющий массив размера, указанного пользователем, случайными числами и сортирующей его пятью разными алгоритмами. Программа должна так же искать различную информацию о массиве, такую как: максимальный и минимальный элементы, их среднее значение и индексы элементов, равные ему; содержится ли в массиве число, введённое пользователем с клавиатуры. Для всех функций необходимо программно рассчитать затраченное время.

**Основные теоретические положения.**

Сортировка пузырьком заключается в попарном сравнении элементов и последующем их обмене. 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 гарантирует, что в конце массива находится максимальный элемент. Хуже всего алгоритм работает, когда на вход подается массив, отсортированный в обратную сторону, быстрее всего – с отсортированным массивом на входе.

Шейкер-сортировка – модификация пузырьковой сортировки. Принцип аналогичен, но имеется существенное отличие. Как только максимальный элемент становится на свое место, алгоритм запускает сортировку в обратную сторону. После выполнения первой итерации минимальный и максимальный элементы будут в начале и конце массива соответственно. За счёт того, что сортировка работает в обе стороны, массив сортируется на порядок быстрее. Но в работе с большими массивами преимущество шейкер-сортировки уменьшается из-за использования двух циклов.

Сортировка вставками. Элементы массива просматриваются по одному, и каждый новый элемент размещается в подходящее место среди ранее упорядоченных элементов. Сортировка вставками отделяет отсортированную часть массива. Лучше всего работает при обработке почти отсортированных массивов. В массивах с длиной меньше 10 она работает быстрее других.

Сортировка расческой – модификация сортировки пузырьком. В этой сортировке переставляются элементы, стоящие на расстоянии. Алгоритм разработан для случаев, когда минимальные элементы стоят слишком далеко от начала массива или максимальные слишком близко к его началу. Эта сортировка неустойчива.

Быстрая сортировка, разработанная в 1960 году английским ученым Чарльзом Хоаром, отличается от других операцией разбиения массива на две части относительно опорного элемента. Вне зависимости от выбора опорного элемента, массив будет отсортирован, но лучшей ситуацией считается та, в которой по сторонам от опорного элемента оказывается примерно равное количество элементов.

**Постановка задачи.**

Разработать алгоритм и написать программу, которая:

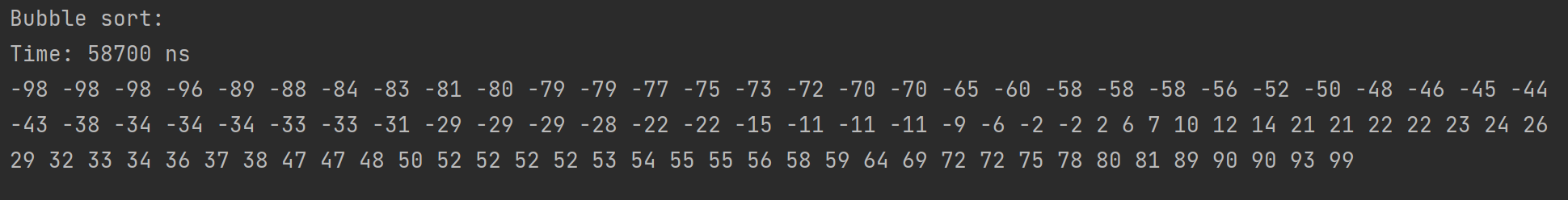
1. Создает целочисленный массив размерности N и заполняет его случайными числами в диапазоне от -99 до 99 включительно.
2. Сортирует полученный массив по возрастанию пятью разными сортировками (bubble sort, shaker sort, comb sort, insert sort, quick sort) и определяет время, затраченное на сортировку.
3. Находит максимальный и минимальный элемент массива и определяет время, затраченное на поиск этих элементов в отсортированном и неотсортированном массивах.
4. Находит среднее значение максимального и минимального значения. Выводит индексы элементов, которые равны этому значению, и определяет время, затраченное на поиск.
5. Выводит количество элементов в отсортированном массиве, которые меньше числа, инициализируемого пользователем.
6. Выводит количество элементов в отсортированном массиве, которые больше числа, инициализируемого пользователем.
7. Находит число, заданное пользователем, методом бинарного поиска и методом перебора значений и определяет время, затраченное на поиск.
8. Меняет местами элементы массива, индексы которых вводит пользователь и определяет время обмена.

**Выполнение работы.**

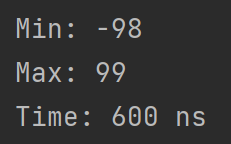
Код программы представлен в приложении А

1. Следующий шаг зависит от введенной команды, если пользователь ввёл:

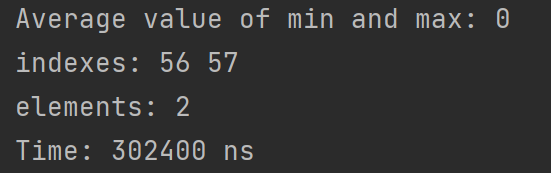
* 1. Выполняется сортировка, указанная в программе, а также время, затраченное на сортировку. (одна из сортировок)



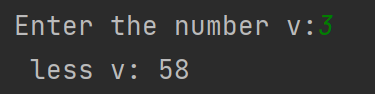
* 1. Выполняется поиск максимального и минимального элементов в отсортированном и неотсортированном массивах. Выводится время, затраченное на поиски.



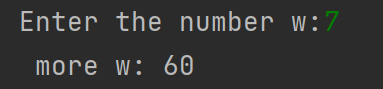
* 1. Выполняется поиск чисел, равных среднему значению минимального и максимального элементов массива. Выводятся индексы элементов и время, затраченное на поиск.



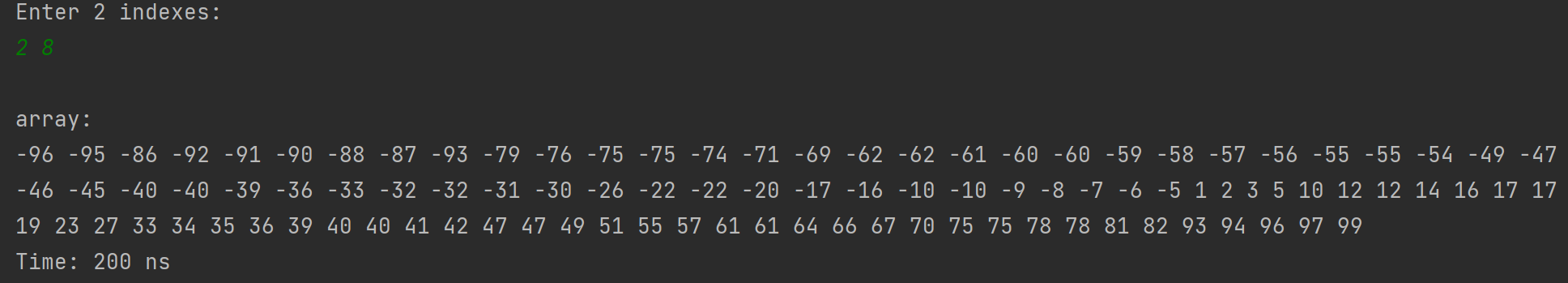
* 1. В массиве ищется количество чисел, значение которых меньше числа, введенного пользователем.



* 1. В массиве ищется количество чисел, значение которых больше числа, введенного пользователем.



* 1. Выполняется обмен местами чисел, индексы которых вводит пользователь. Подсчитывается время обмена.



**Выводы.**

В ходе работы были изучены одномерные статические массивы и методы работы с ними. Изучены алгоритмы сортировки массивов, алгоритм бинарного поиска.

Приложение А

рабочий код

#include <iostream>  
#include <chrono>  
#include <ctime>  
using namespace std;  
using namespace chrono;  
int main() {  
 setlocale(LC\_ALL, "en");  
  
 srand(time(0));  
 int arr[100];  
 int min;  
 int max;  
 auto start = steady\_clock::now(); // chrono  
 cout << endl;  
 auto end = steady\_clock::now();  
 auto result = duration\_cast<nanoseconds>(end - start);  
  
 cout << "Unsorted array: " << "\n"; // unsorted array  
 start = steady\_clock::now();  
 for (int i = 0; i < 100; i++){  
 arr[i] = rand() % 199 - 99;  
 }  
  
 end = steady\_clock::now();  
 result = duration\_cast<nanoseconds>(end - start); // fide time unsorted array  
 cout << "Time: " << result.count() << " ns" << "\n";  
  
 for (int i = 0; i < 100; i++) {  
 cout << arr[i] << " ";  
 }  
 cout << "\n" << "\n";  
  
  
 cout << "Bubble sort: " << "\n"; // bubble sort  
 start = steady\_clock::now();  
 for (int i = 0; i < 100; i++) {  
 for (int j = 0; j < 99; j++) {  
 if (arr[j] > arr[j + 1]) {  
 int b = arr[j];  
 arr[j] = arr[j + 1];  
 arr[j + 1] = b;  
 }  
 }  
 }  
 end = steady\_clock::now();  
 result = duration\_cast<nanoseconds>(end - start); // fide time sorted array(bubble sort)  
 cout << "Time: " << result.count() << " ns" << "\n";  
 for (int i = 0; i < 100; i++) {  
 cout << arr[i] << " ";  
 }  
  
 cout << "\n" << "\n";  
  
 start = steady\_clock::now();  
 for(int i = 0; i < 100; i++) // min and max(sorted array)  
 {  
 if(arr[i + 1] < arr[i]){  
 max = arr[i];  
 }  
 if(arr[i - 1] > arr[i]) {  
 min = arr[i];  
 }  
 }  
 end = steady\_clock::now();  
 result = duration\_cast<nanoseconds>(end - start);  
  
 cout << "Min: " << min << endl;  
 cout << "Max: " << max << endl;  
 cout << "Time: " << result.count() << " ns" << "\n";  
  
 cout << "\n";  
  
 cout << "Average value of min and max: "; // average value of min and max(sorted array)  
 float sr = (min + max) / 2;  
 cout << sr << "\n";  
  
 cout << "indexes: "; // prints the indexes of all elements that are equal to this value  
 start = steady\_clock::now();  
 int d = 0;  
 for (int i = 0; i < 100; i++) {  
 if (arr[i] == int(sr)) {  
 cout << i << " ";  
 d += 1;  
 }  
 }  
 end = steady\_clock::now();  
 result = duration\_cast<nanoseconds>(end - start);  
 cout << endl << "elements: " << d << "\n"; // prints the number of indexes of all elements  
 cout << "Time: " << result.count() << " ns" << "\n";  
  
 cout << "Enter the number v: ";  
 int v; /\* Returns the number of elements in the sorted array that are  
 less than the number V, which is initialized by the user \*/  
 int q = 0;  
 cin >> v;  
 for (int i = 0; i < 100; i++)  
 if (arr[i] < v)  
 q += 1;  
 cout << "less v: " << q;  
  
 cout << "\n";  
  
 cout << "Enter the number w: "; /\*Returns the number of elements in the sorted array that are greater  
 than the number W, which is initialized by the user\*/  
 int w;  
 int l = 0;  
 cin >> w;  
 for (int i = 0; i < 100; i++)  
 if (arr[i] < w)  
 l += 1;  
 cout << "more w: " << l;  
  
 cout << "\n";  
  
 cout << "\n" << "\n";  
  
  
  
  
 cout << "Unsorted array: " << "\n";  
 for (int i = 0; i < 100; i++){  
 arr[i] = rand() % 199 - 99;  
 }  
 for (int i = 0; i < 100; i++) {  
 cout << arr[i] << " ";  
 }  
  
 cout << "\n" << "\n";  
  
 cout << "Shaker sort: " << "\n"; // shaker sort  
 start = steady\_clock::now();  
 bool s = true;  
 int t = 0;  
 int nd = 99;  
 while (s) {  
 s = false;  
 for (int i = t; i < nd; ++i) {  
 if (arr[i] > arr[i + 1]) {  
 swap(arr[i], arr[i + 1]);  
 s = true;  
 }  
 }  
 s = false;  
 nd--;  
 for (int i = nd - 1; i >= t; --i) {  
 if (arr[i] > arr[i + 1]) {  
 swap(arr[i], arr[i + 1]);  
 s = true;  
 }  
 }  
 t++;  
 }  
  
 end = steady\_clock::now();  
 result = duration\_cast<nanoseconds>(end - start); // fide time sorted array(shaker sort)  
 cout << "Time: " << result.count() << " ns" << "\n";  
  
 for (int i = 0; i < 100; i++) {  
 cout << arr[i] << " ";  
 }  
  
 cout << "\n";  
  
 cout << "\n" << "\n";  
  
  
  
  
 cout << "Unsorted array: " << "\n";  
 for (int i = 0; i < 100; i++){  
 arr[i] = rand() % 199 - 99;  
 }  
 for (int i = 0; i < 100; i++) {  
 cout << arr[i] << " ";  
 }  
  
 cout << "\n" << "\n";  
  
 cout << "Comb sort: " << "\n"; // comb sort  
 start = steady\_clock::now();  
 int a;  
 int b;  
 while (true){  
 for (int i = 0; i < 99; i++){  
 if (arr[i] > arr[i + 1])  
 {  
 b = arr[i + 1];  
 arr[i + 1] = arr[i];  
 arr[i] = b;  
 }  
 else a++;  
 }  
 if (a == 99){  
 break;  
 }  
 else {  
 a = 0;  
 }  
 }  
  
 end = steady\_clock::now();  
 result = duration\_cast<nanoseconds>(end - start); // fide time sorted array(comb sort)  
 cout << "Time: " << result.count() << " ns" << "\n";  
  
 for (int i = 0; i < 100; i++) {  
 cout << arr[i] << " ";  
 }  
 cout << "\n";  
  
 cout << "\n" << "\n";  
  
  
  
 cout << "Unsorted array: " << "\n";  
 for (int i = 0; i < 100; i++){  
 arr[i] = rand() % 199 - 99;  
 }  
 for (int i = 0; i < 100; i++) {  
 cout << arr[i] << " ";  
 }  
  
 cout << "\n" << "\n";  
  
 cout << "Insert sort: " << "\n"; // insert sort  
 start = steady\_clock::now();  
 int x;  
 for (int i = 0; i < 100; i++) {  
 x = arr[i];  
 int j = i;  
 while ((j > 0) && (x < arr[j - 1])) {  
 arr[j] = arr[j - 1];  
 j--;  
 }  
  
 arr[j] = x;  
 }  
  
 end = steady\_clock::now();  
 result = duration\_cast<nanoseconds>(end - start); // fide time sorted array(insert sort)  
 cout << "Time: " << result.count() << " ns" << "\n";  
  
 for (int i = 0; i < 100; i++) {  
 cout << arr[i] << " ";  
 }  
 cout << endl << "\n";  
  
 cout << "array: " << "\n"; // Swaps the elements of an array whose indexes are entered by the user  
 for (int i = 0; i < 100; i++) {  
 cout << arr[i] << " ";  
 }  
 cout << "\nEnter 2 indexes: " << "\n";  
 int m, k;  
 cin >> m >> k;  
 cout << "\narray: " << "\n";  
 start = steady\_clock::now();  
 int ind = arr[m];  
 arr[m] = arr[k];  
 arr[k] = ind;  
  
 end = steady\_clock::now();  
 result = duration\_cast<nanoseconds>(end - start);  
  
 for (int i = 0; i < 100; i++) {  
 cout << arr[i] << " ";  
 }  
 cout << "\nTime: " << result.count() << " ns\n"; // exchange rate  
  
 cout << "\n" << "\n";  
  
  
 return 0;  
}