**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра наименование кафедры**

отчет

**по практической работе №2**

**по дисциплине «Программирование»**

Тема: Одномерные статические массивы

|  |  |  |
| --- | --- | --- |
| Студентка гр. 2372 |  | Мельникова М.А. |
| Преподаватель |  | Глущенко А.Г. |

Санкт-Петербург

2022

**Цель работы.**

Познакомиться с массивами, их объявлением, с различными типами сортировок: bubble sort, shaker sort, comb sort, insert sort, quick sort, с бинарным поиском.

**Основные теоретические положения.**

Массив – это индексированная последовательность однотипных элементов с заранее определенным количеством элементов. Массивы используются для обработки большого количества однотипных данных.

Имя массива является указателем.

Элемент массива – это отдельная ячейка данных массива. Элементы массива нумеруются с нуля. При описании массива используются те же модификаторы (класс памяти, const и инициализатор), что и для простых переменных.

Алгоритм – это совокупность четко сформулированных инструкций, которые решают точно обозначенную задачу.

Алгоритм бинарного поиска – классический алгоритм поиска в отсортированном массиве, который использует дробление массива на половины.

**Постановка задачи.**

Необходимо разработать алгоритм и написать программу, которая позволяет:

1) Создает целочисленный массив размерности N = 100. Элементы массивы должны принимать случайное значение в диапазоне от -99 до 99.

2) Отсортировывает заданный в пункте 1 массив сортировкой (от меньшего к большему). Определяет время, затраченное на сортировку.

3) Находит максимальный и минимальный элемент массива. Выводит время поиска этих элементов в отсортированном массиве и неотсортированном.

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, их количество и 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. Реализовать алгоритм бинарного поиска. Сравнить скорость его работы с обычным перебором.

8) Меняет местами элементы массива, индексы которых вводит пользователь. Выведите скорость обмена.

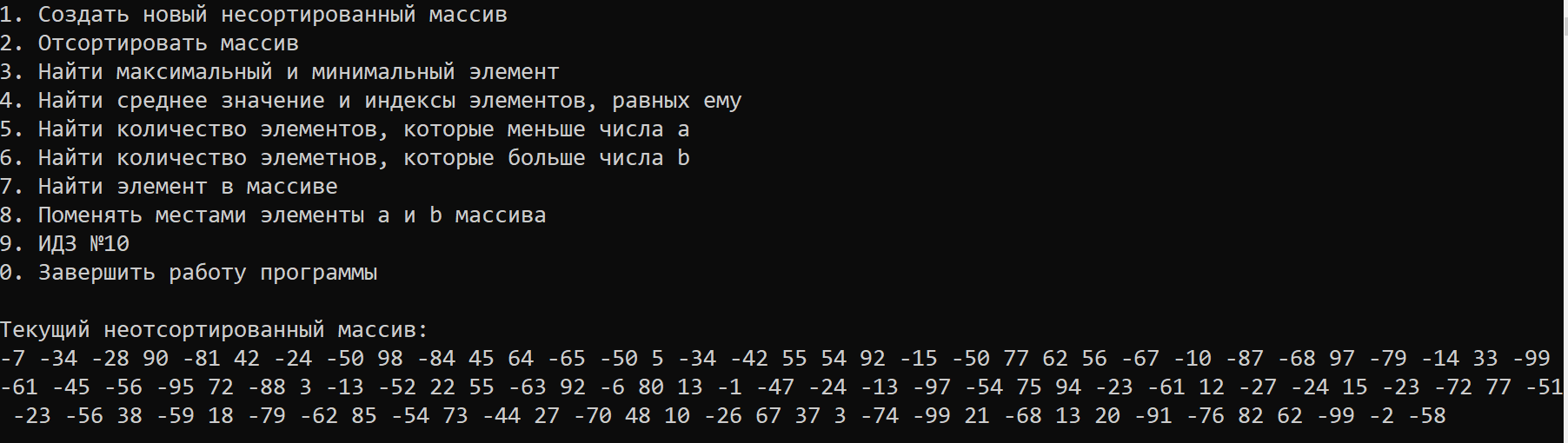
**Выполнение работы.**

Для выполнения поставленной задачи работы использован язык программирования С++. Итоговый код программы представлен в приложении.

Для создания массива случайных значений в заданном диапазоне используем функцию srand(). Отсортируем массив сортировками bubble sort, shaker sort, comb sort, insert sort, quick sort, представленными функциями типа void, которые не возвращают никакое значение. Bubble sort реализуется как простой перебор, двумя вложенными циклами. Shaker sort – как перебор, идущий в две стороны. Comb sort модификация сортировки bubble sort, где сравниваются элементы из разных концов массива и из итерации в итерацию уменьшается размер шага. Insert sort – алгоритм сортировки, в котором элементы массива просматриваются по одному, и каждый новый элемент размещается в подходящее место среди ранее упорядоченных элементов. Quick sort – рекурсивный алгоритм, дробящий массив на каждом этапе на массивы больше и меньше опорного элемента.

Бинарный поиск реализовывается с помощью цикла while, с условием пока left - левая граница массива, равная нулю меньше или равна right - правой границе массива, равной последнему индексу массива. Массив делится пополам и проверяется, в какой части находится искомый элемент. Потом с этой половиной проводится то же действие.

При запуске программа выводит следующее:



*Рис.1*

При вызове второго задания предлагается выбор сортировки:



*Рис.2*

**Выводы.**

В процессе работы было получено представление о работе с массивами, сортировками, бинарным поиском. Написана программа, генерирующая массив, изменяющая массив, выводящая некоторую информацию о массиве.

Приложение

Полный код программы

#include <iostream>

#include <chrono>

#include <ctime>

using namespace std;

using namespace chrono;

void random(int Arr[], const int n) {

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

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

}

}

void print\_arr(int Arr[], const int n) {

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

cout << Arr[i] << " ";

}

}

void bubble\_sort(int Arr[], const int n)

{

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

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

if (Arr[i] > Arr[i + 1]) {

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

}

}

}

}

void shaker\_sort(int Arr[], const int n)

{

bool flag = true;

int right = n - 1;

int left = 1;

while (flag) {

flag = false;

for (int i = left; i <= right; i++) {

if (Arr[i - 1] > Arr[i]) {

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

flag = true;

}

}

right--;

for (int i = right; i >= left; i--) {

if (Arr[i] < Arr[i - 1]) {

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

flag = true;

}

}

left++;

}

}

void comb\_sort(int Arr[], const int n)

{

float k = 1.247;

float S = n - 1;

while (S >= 1) {

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

if (Arr[i] > Arr[int(i + S)]) {

swap(Arr[int(i + S)], Arr[i]);

}

}

S /= k;

}

int count = 0;

while (true) {

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

if (Arr[i] > Arr[i + 1]) {

swap(Arr[int(i + S)], Arr[i]);

}

else count++;

}

if (count == n - 1)

break;

else

count = 0;

}

}

void insert\_sort(int Arr[], const int n)

{

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

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

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

if (Arr[i] > Arr[i + 1]) {

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

}

}

}

}

}

void quick\_sort(int Arr[], int end, int start)

{

int mid, s = start, e = end;

mid = Arr[(s + e) / 2];

while (s < e)

{

while (Arr[s] < mid) s++;

while (Arr[e] > mid) e--;

if (s <= e)

{

swap(Arr[s], Arr[e]);

s++;

e--;

}

}

if (start < e) quick\_sort(Arr, e, start);

if (s < end) quick\_sort(Arr, end, s);

}

int main()

{

setlocale(0, "Rus");

srand(time(0));

const int N = 100;

int Array[N], SortArray[N];

cout << "1. Создать новый несортированный массив\n2. Отсортировать массив\n3. Найти максимальный и минимальный элемент\n4. Нaйти среднее значение и индексы элементов, равных ему\n5. Найти количество элементов, которые меньше числа a\n6. Найти количество элеметнов, которые больше числа b\n7. Найти элемент в массиве\n8. Поменять местами элементы a и b массива\n0. Завершить работу программы\n\n";

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

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

SortArray[i] = Array[i];

}

cout << "Текущий неотсортированный массив:\n";

print\_arr(Array, N);

cout << "\n\n";

while (true) {

int ind;

cin >> ind;

if (ind == 1) {

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

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

SortArray[i] = Array[i];

}

cout << "Текущий неотсортированный массив:\n";

print\_arr(Array, N);

cout << "\n\n";

}

else if (ind == 2) {

cout << "\t1. Bubble sort\n\t2. Shaker sort\n\t3. Comb sort\n\t4. Insert sort\n\t5. Quick sort\n";

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

SortArray[i] = Array[i];

}

int ind2;

cin >> ind2;

if (ind2 == 1) {

cout << "\nНесортированный массив\n";

print\_arr(SortArray, N);

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

bubble\_sort(SortArray, N);

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

auto result = duration\_cast<nanoseconds>(end - start);

cout << "\nОтсортированный массив\n";

print\_arr(SortArray, N);

cout << "\nВремя сортитовки массива из 100 элеметнов сортитовкой \"bubble sort\": ";

cout << result.count() << " наносекунд\n\n";

}

else if (ind2 == 2) {

cout << "\nНесортированный массив\n";

print\_arr(SortArray, N);

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

shaker\_sort(SortArray, N);

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

auto result = duration\_cast<nanoseconds>(end - start);

cout << "\nОтсортированный массив\n";

print\_arr(SortArray, N);

cout << "\nВремя сортитовки массива из 100 элеметнов сортитовкой \"bubble sort\": ";

cout << result.count() << " наносекунд\n\n";

}

else if (ind2 == 3) {

cout << "\nНесортированный массив\n";

print\_arr(SortArray, N);

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

comb\_sort(SortArray, N);

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

auto result = duration\_cast<nanoseconds>(end - start);

cout << "\nОтсортированный массив\n";

print\_arr(SortArray, N);

cout << "\nВремя сортитовки массива из 100 элеметнов сортитовкой \"bubble sort\": ";

cout << result.count() << " наносекунд\n\n";

}

else if (ind2 == 4) {

cout << "\nНесортированный массив\n";

print\_arr(SortArray, N);

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

insert\_sort(SortArray, N);

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

auto result = duration\_cast<nanoseconds>(end - start);

cout << "\nОтсортированный массив\n";

print\_arr(SortArray, N);

cout << "\nВремя сортитовки массива из 100 элеметнов сортитовкой \"bubble sort\": ";

cout << result.count() << " наносекунд\n\n";

}

else if (ind2 == 5) {

cout << "\nНесортированный массив\n";

print\_arr(SortArray, N);

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

int s = 0, e = N - 1;

quick\_sort(SortArray, e, s);

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

auto result = duration\_cast<nanoseconds>(end - start);

cout << "\nОтсортированный массив\n";

print\_arr(SortArray, N);

cout << "\nВремя сортитовки массива из 100 элеметнов сортитовкой \"bubble sort\": ";

cout << result.count() << " наносекунд\n\n";

}

}

else if (ind == 3) {

comb\_sort(SortArray, N);

// Поиск max, min элементов в неотсортированном массиве

auto start1 = steady\_clock::now();

int min1 = Array[0], max1 = Array[0];

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

if (Array[i] < min1) {

min1 = Array[i];

}

else if (Array[i] > max1) {

max1 = Array[i];

}

}

auto end1 = steady\_clock::now();

auto result1 = duration\_cast<nanoseconds>(end1 - start1);

// Поиск max, min элементов в отсортированном массиве

auto start2 = steady\_clock::now();

int min2, max2;

min2 = SortArray[0];

max2 = SortArray[N - 1];

auto end2 = steady\_clock::now();

auto result2 = duration\_cast<nanoseconds>(end2 - start2);

cout << "min = " << min2 << "\nmax = " << max2;

cout << "\n\nВремя поиска максимального и миниманого элемента\n";

cout << "В несортированном массиве: " << result1.count() << " наносекунд\n";

cout << "В отсортированном массиве: " << result2.count() << " наносекунд\n";

}

else if (ind == 4) {

comb\_sort(SortArray, N);

int average = (SortArray[0] + SortArray[N - 1]) / 2;

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

cout << "Индексы: ";

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

int k = 0; //количество

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

if (Array[i] == average) {

cout << i << " ";

k += 1;

}

}

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

auto result = duration\_cast<nanoseconds>(end - start);

cout << "\nКоличество элеметнов: " << k << "\n";

cout << "Время: " << result.count() << " наносекунд\n";

}

else if (ind == 5) {

comb\_sort(SortArray, N);

cout << "Введите число a: ";

int a, k = 0;

bool flag = true;

cin >> a;

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

flag = false;

if (SortArray[i] < a) {

k += 1;

flag = true;

}

}

if (k > 0)

cout << "Количество элементов, которые меньше числа а: " << k << "\n\n";

else

cout << "Таких элементов нет\n\n";

}

else if (ind == 6) {

comb\_sort(SortArray, N);

cout << "Введите число b: ";

int b, k = 0;

bool flag = true;

cin >> b;

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

flag = false;

if (SortArray[N - 1 - i] > b) {

k += 1;

flag = true;

}

}

if (k > 0)

cout << "Количество элементов, которые больше числа b: " << k << "\n\n";

else

cout << "Таких элементов нет\n\n";

}

else if (ind == 7) {

comb\_sort(SortArray, N);

cout << "\nВведите число: ";

int a;

cin >> a;

bool flag1 = false, flag2 = false;

//перебор (1)

auto start1 = steady\_clock::now();

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

if (SortArray[i] == a) {

flag1 = true;

break;

}

}

auto end1 = steady\_clock::now();

auto result1 = duration\_cast<nanoseconds>(end1 - start1);

//бинарный поиск (2)

auto start2 = steady\_clock::now();

int right = N - 1, left = 0, mid;

while ((left <= right) && (flag2 != true)) {

mid = (left + right) / 2;

if (SortArray[mid] == a) flag2 = true;

if (SortArray[mid] > a) right = mid - 1;

else left = mid + 1;

}

auto end2 = steady\_clock::now();

auto result2 = duration\_cast<nanoseconds>(end2 - start2);

if (flag1 == true)

cout << "\nВведённый элемент есть в массиве\n";

else if (flag1 == false)

cout << "\nЭлемент не найден\n";

cout << "Время проверки полным перебором: " << result1.count() << " наносекунд\n\n";

if (flag2 == true)

cout << "\nВведённый элемент есть в массиве\n";

else if (flag2 == false)

cout << "\nЭлемент не найден\n";

cout << "Время проверки бинарным поиском: " << result2.count() << " наносекунд\n\n";

}

else if (ind == 8) {

int a, b;

cout << "Введите индексы элементов, которые вы хотите поменять местами: ";

cin >> a >> b;

if ((a < 0) || (b > 99)) {

cout << "\nТаких элементов нет в массиве\n\n";

}

else {

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

swap(Array[a], Array[b]);

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

auto result = duration\_cast<nanoseconds>(end - start);

cout << "\nНовый массив:\n";

print\_arr(Array, N);

cout << "\nВремя: " << result.count() << " наносекунд\n\n";

}

}

else if (ind == 0) {

break;

}

else {

cout << "Ошибка";

}

}

}