**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. |  | Коваленко М. А. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2022

**Цель работы.**

Научиться работать с одномерными статическими массивами, применять для них разные способы сортировок, определять время, которое требуется для различных операций, находить максимальные и минимальные значения массива.

**Постановка задачи.**

Требуется выполнить следующие задачи:

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.

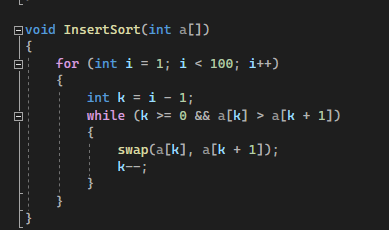
5)Выводит количество элементов в отсортированном массиве, которые меньше числа *a*, которое инициализируется пользователем.

6)Выводит количество элементов в отсортированном массиве, которые больше числа *b*, которое инициализируется пользователем.

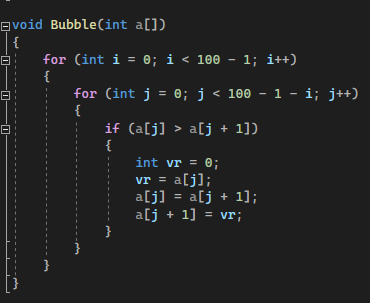
7)Выводит информацию о том, есть ли введенное пользователем число в отсортированном массиве. Реализуйте алгоритм бинарного поиска. Сравните скорость его работы с обычным перебором. (\*)

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

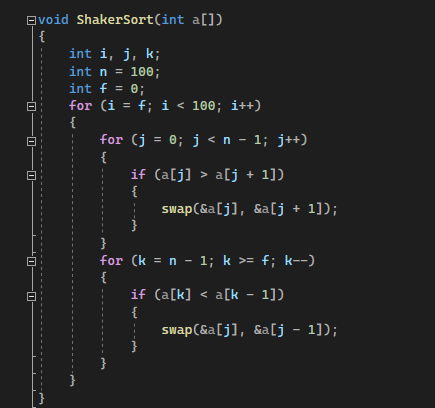
**Основные теоретические положения.**



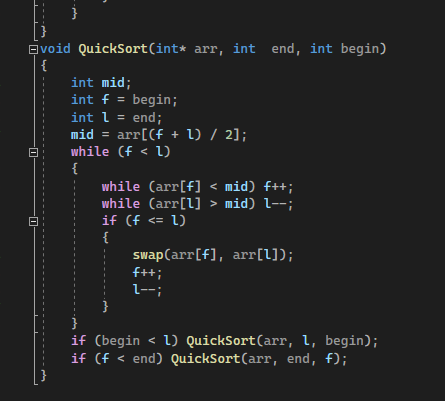
Алгоритм InsertSort.



Алгоритм BubbleSort.

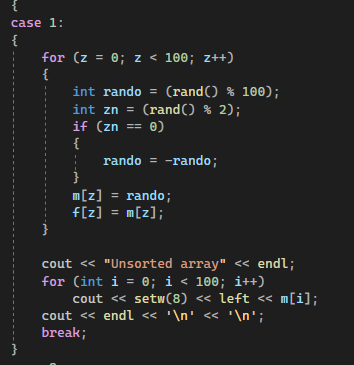


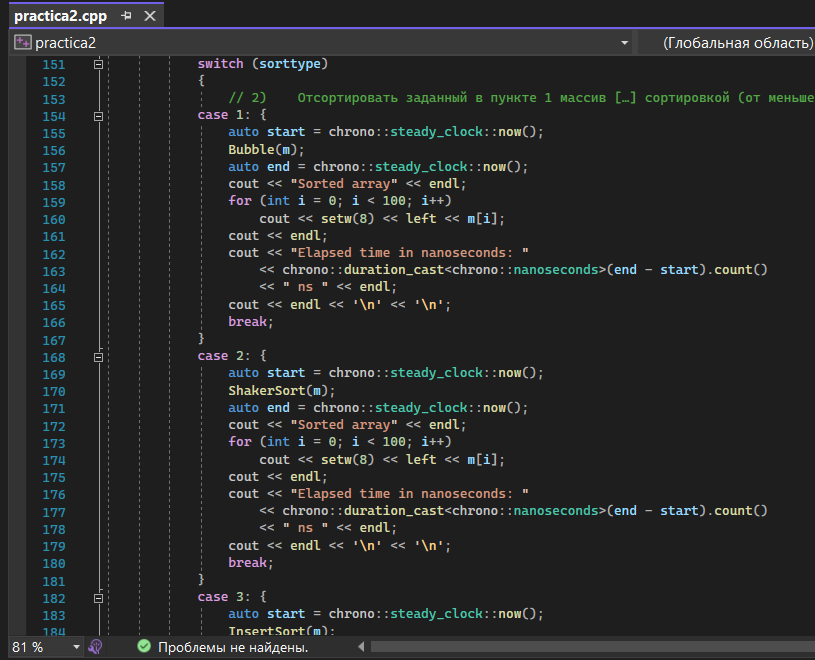
Алгоритм ShakerSort.

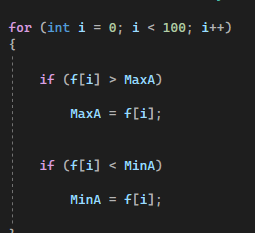


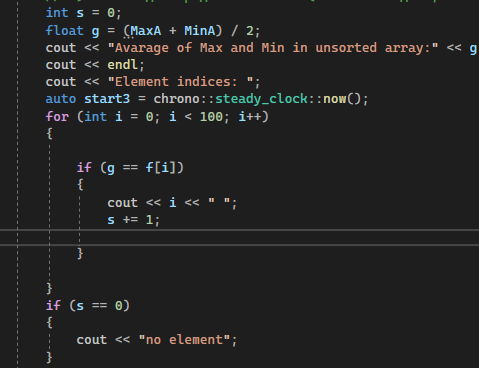
Алгоритм QuickSort.

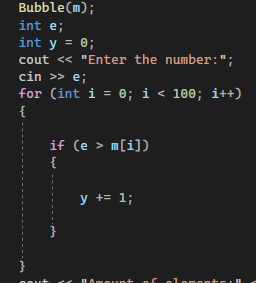
**Выполнение работы.**

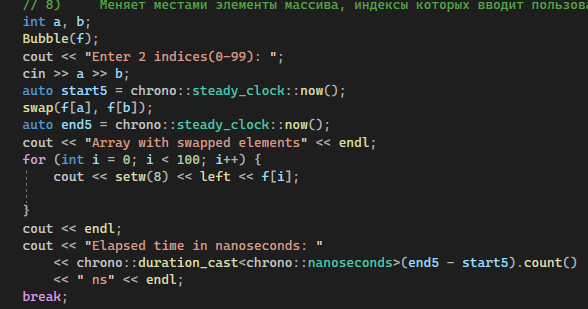
Создаём массив.

Сортируем массив разными видами сортировок.

Находим максимальное и минимальное значение массива.

Находим среднее значение максимального и минимального, и выводим индексы и количество элементов.

Вводим число, и у нас выводится количество элементов, которые больше или меньше данного числа.

 Меняем местами 2 элемента массива, которые сами выбираем.

**Выводы.**

Данная практическая работа выполняется исключительно с помощью разных видов сортировок, с помощью них мы нашли максимальное и минимальное значение массива и нашли их среднее значение.

Приложение А

Полный код программы

#include <iostream>

#include <stdlib.h>

#include <iomanip>

#include <chrono>

#include <clocale>

using namespace std;

void swap(int\* a, int\* b)

{

int temp;

temp = \*a;

\*a = \*b;

\*b = temp;

}

void InsertSort(int a[])

{

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

{

int k = i - 1;

while (k >= 0 && a[k] > a[k + 1])

{

swap(a[k], a[k + 1]);

k--;

}

}

}

void Bubble(int a[])

{

for (int i = 0; i < 100 - 1; i++)

{

for (int j = 0; j < 100 - 1 - i; j++)

{

if (a[j] > a[j + 1])

{

int vr = 0;

vr = a[j];

a[j] = a[j + 1];

a[j + 1] = vr;

}

}

}

}

void ShakerSort(int a[])

{

int i, j, k;

int n = 100;

int f = 0;

for (i = f; i < 100; i++)

{

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

{

if (a[j] > a[j + 1])

{

swap(&a[j], &a[j + 1]);

}

}

for (k = n - 1; k >= f; k--)

{

if (a[k] < a[k - 1])

{

swap(&a[j], &a[j - 1]);

}

}

}

}

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);

}

int main()

{

setlocale(LC\_ALL, "Russian");

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

srand(time(NULL));

int m[100];

int f[100];

int z;

int MaxA = -100;

int MinA = 100;

while (true) {

int task;

cout << "Select task:\n"

"1. Create array\n"

"2. Sort array\n"

"3. Minimum and Maximum\n"

"4. Avarage of Minimum and Maximum\n"

"5. The number of elements, are less \"a\"\n"

"6. The number of elements, are more \"b\"\n"

"8. Element replacement \"a\" & \"b\"\n"

"9. Close programm\n";

cin >> task;

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

switch (task)

{

case 1:

{

for (z = 0; z < 100; z++)

{

int rando = (rand() % 100);

int zn = (rand() % 2);

if (zn == 0)

{

rando = -rando;

}

m[z] = rando;

f[z] = m[z];

}

cout << "Unsorted array" << endl;

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

cout << setw(8) << left << m[i];

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

break;

}

case 2:

{

int sorttype;

cout << "Choice sort:\n" <<

" 1. Bubble Sort\n" <<

" 2. Shaker Sort\n" <<

" 3. Insertion Sort\n" <<

" 4. Quick Sort\n";

cin >> sorttype;

switch (sorttype)

{

// 2) Отсортировать заданный в пункте 1 массив […] сортировкой (от меньшего к большему). Определить время, затраченное на сортировку, используя библиотеку chrono.

case 1: {

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

Bubble(m);

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

cout << "Sorted array" << endl;

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

cout << setw(8) << left << m[i];

cout << endl;

cout << "Elapsed time in nanoseconds: "

<< chrono::duration\_cast<chrono::nanoseconds>(end - start).count()

<< " ns " << endl;

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

break;

}

case 2: {

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

ShakerSort(m);

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

cout << "Sorted array" << endl;

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

cout << setw(8) << left << m[i];

cout << endl;

cout << "Elapsed time in nanoseconds: "

<< chrono::duration\_cast<chrono::nanoseconds>(end - start).count()

<< " ns " << endl;

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

break;

}

case 3: {

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

InsertSort(m);

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

cout << "Sorted array" << endl;

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

cout << setw(8) << left << m[i];

cout << endl;

cout << "Elapsed time in nanoseconds: "

<< chrono::duration\_cast<chrono::nanoseconds>(end - start).count()

<< " ns " << endl;

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

break;

}

case 4: {

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

QuickSort(m, 99, 0);

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

cout << "Sorted array" << endl;

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

cout << setw(8) << left << m[i];

cout << endl;

cout << "Elapsed time in nanoseconds: "

<< chrono::duration\_cast<chrono::nanoseconds>(end - start).count()

<< " ns " << endl;

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

break;

}

default: {

break;

}

}

break;

}

case 3: {

InsertSort(m);

// 3) Найти максимальный и минимальный элемент массива. Подсчитайте время поиска этих элементов в отсортированном массиве и неотсортированном, используя библиотеку chrono.

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

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

{

if (f[i] > MaxA)

MaxA = f[i];

if (f[i] < MinA)

MinA = f[i];

}

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

cout << "In sorted array" << endl;

cout << "Max:" << MaxA << " " << "Min:" << MinA;

cout << endl;

cout << "Elapsed time in nanoseconds: "

<< chrono::duration\_cast<chrono::nanoseconds>(end1 - start1).count()

<< " ns" << endl;

cout << endl;

cout << "In unsorted array" << endl;

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

cout << "Max:" << m[99] << " " << "Min:" << m[0];

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

cout << endl;

cout << "Elapsed time in nanoseconds: "

<< chrono::duration\_cast<chrono::nanoseconds>(end2 - start2).count()

<< " ns " << endl;

cout << endl << '\n';

break;

}

case 4: {

// 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.

int s = 0;

float g = (MaxA + MinA) / 2;

cout << "Avarage of Max and Min in unsorted array:" << g;

cout << endl;

cout << "Element indices: ";

auto start3 = chrono::steady\_clock::now();

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

{

if (g == f[i])

{

cout << i << " ";

s += 1;

}

}

if (s == 0)

{

cout << "no element";

}

auto end3 = chrono::steady\_clock::now();

cout << '\n' << "Amount of element:" << s;

cout << endl;

cout << "Elapsed time in nanoseconds: "

<< chrono::duration\_cast<chrono::nanoseconds>(end3 - start3).count()

<< " ns " << endl;

cout << endl;

int l = 0;

float q = (m[99] + m[0]) / 2;

cout << "Avarage of Max and Min in sorted array:" << q;

cout << endl;

cout << "Element indices: ";

auto start4 = chrono::steady\_clock::now();

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

{

if (q == m[i])

{

cout << i << " ";

l += 1;

}

}

if (l == 0)

{

cout << "no element";

}

auto end4 = chrono::steady\_clock::now();

cout << '\n' << "Amount of element:" << l;

cout << endl;

cout << "Elapsed time in nanoseconds: "

<< chrono::duration\_cast<chrono::nanoseconds>(end4 - start4).count()

<< " ns" << endl;

cout << endl;

cout << endl;

break;

}

case 5: {

// 5) Выводит количество элементов в отсортированном массиве, которые меньше числа a, которое инициализируется пользователем.

Bubble(m);

int e;

int y = 0;

cout << "Enter the number:";

cin >> e;

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

{

if (e > m[i])

{

y += 1;

}

}

cout << "Amount of elements:" << y;

cout << endl;

cout << endl;

break;

}

case 6: {

// 6) Выводит количество элементов в отсортированном массиве, которые больше числа b, которое инициализируется пользователем.

Bubble(m);

int n;

int p = 0;

cout << "Enter the number:";

cin >> n;

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

{

if (n < m[i])

{

p += 1;

}

}

cout << "Amount of elements:" << p;

cout << endl;

cout << endl;

break;

}

case 8: {

// 8) Меняет местами элементы массива, индексы которых вводит пользователь. Выведите скорость обмена, используя библиотеку chrono.

int a, b;

Bubble(f);

cout << "Enter 2 indices(0-99): ";

cin >> a >> b;

auto start5 = chrono::steady\_clock::now();

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

auto end5 = chrono::steady\_clock::now();

cout << "Array with swapped elements" << endl;

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

cout << setw(8) << left << f[i];

}

cout << endl;

cout << "Elapsed time in nanoseconds: "

<< chrono::duration\_cast<chrono::nanoseconds>(end5 - start5).count()

<< " ns" << endl;

break;

}

case 9: {exit(0);

break;

}

default:

{

break;

}

}

}

}