**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. |  | Симаков Г.О. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2022

**Цель работы.**

Ознакомиться со структурой одномерного массива, обработкой данных

одномерных массивов; изучить различные виды сортировок; провести

временную оценку различных действий с массивами.

**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.

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.

Размерность массива вместе с типом его элементов определяет объем памяти, необходимый для размещения массива, которое выполняется на этапе компиляции, поэтому размерность должна быть задана целой положительной константой или константным выражением.

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.

Затем процесс повторяется до тех пор, пока вся последовательность не будет упорядочена. Важно заметить, что после первого прохода по массиву, уже имеется один упорядоченный элемент, он стоит на своем месте, и менять его не надо. Таким образом на следующем шаге будут сравниваться *N*-1 элемент.

Shaker sort – модификация пузырьковой сортировки. Принцип работы этой сортировки аналогичен bubble sort: попарное сравнение элементов и последующий обмен местами. Но имеется существенное отличие. Как только максимальный элемент становится на свое место, алгоритм не начинает новую итерацию с первого элемента, а запускает сортировку в обратную сторону. Алгоритм гарантирует, что после выполнения первой итерации, минимальный и максимальный элемент будут в начале и конце массива соответственно.

Затем процесс повторяется до тех пор, пока массив не будет отсортирован. За счет того, что сортировка работает в обе стороны, массив сортируется на порядок быстрее. Очевидным примером этого был бы случай, когда в начале массива стоит максимальный элемент, а в конце массива – минимальный. Shaker sort справится с этим за 1 итерацию, при условии, что другие элементы стоят на правильном месте.

Comb sort (сортировка расческой) – ещё одна модификация сортировки пузырьком. Алгоритм был разработан специально для случаев, когда минимальные элементы стоят слишком далеко, или максимальные – слишком близко к началу массива. В сортировке расческой переставляются элементы, стоящие на расстоянии.

Оптимально изначально взять расстояние равным длине массива, а далее уменьшать его на определенный коэффициент, который примерно равен 1.247. Когда расстояние станет равно 1, выполняется обычная сортировка пузырьком.

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 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.

**Выполнение работы.**

Код программы представлен в приложении А.

**Выводы.**

Выполняя работу, я научился работать со структурой одномерного массива, обрабатывать

данные одномерных массивов; изучил различные виды сортировок;

реализовал сортировку массива; произвел временную оценку различных

действий с массивами.

Приложение А

рабочий код

//функция createArray заполняет целочисленный массив размерности N = 100 случайными значениями в диапазоне от –99 до 99.

//функция sort сортирует элементы массива. Определяется время, затраченное на сортировку, используетсябиблиотека chrono.

//функция quicksort сортирует элементы массива. Определяется время, затраченное на сортировку, используетсябиблиотека chrono.

//функция findMaxMin находит макс. и мин. элементы массива. Определяется время поиска этих элементов в отсортированном и неотсортированном массиве.

//функция findAvarege выводит СрЗнач макс. и мин. значения и выводит индексы всех элементов, которые равны этому значению, и их количество.

//функция findSmaller выводит количество элементов в отсортированном массиве, которые меньше числа a, которое инициализируется пользователем.

//функция findBigger выводит количество элементов в отсортированном массиве, которые больше числа b, которое инициализируется пользователем.

//функция binSearch выводит информацию о том, есть ли введенное пользователем число в отсортированном массиве. Реализуется алгоритм бинарного поиска. Сравнивается скорость его работы с обычным перебором.

//функция replaceElements меняет местами элементы массива,индексы которых вводит пользователь. Выводится скорость обмена.

#include<iostream>

#include<ctime>

#include<chrono>

using namespace std;

using namespace chrono;

const int N = 100;

int arr[N];

int arr\_sort[N];

void createArray()

{

setlocale(LC\_ALL, "ru");

srand(time(0));

cout << "Входной массив:" << endl;

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

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

arr\_sort[i] = arr[i];

cout << arr[i] << " ";

}

cout << endl;

}

void sort()

{

setlocale(0, "rus");

cout << "\nBubble sort" << "\n";

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

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

bool flag = true;

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

if (arr\_sort[j] > arr\_sort[j + 1]) {

flag = false;

swap(arr\_sort[j], arr\_sort[j + 1]);

}

}

if (flag) {

break;

}

}

cout << "Отсортированный массив:\n";

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

cout << arr\_sort[i] << ' ';

arr\_sort[i] = arr[i];

}

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

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

cout << "\nВремя сортировки:" << result.count() << " ns" << endl;

cout << "\nComb sort\n";

float k = 1.247, S = N - 1;

int count = 0;

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

while (S >= 1)

{

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

{

if (arr\_sort[i] > arr\_sort[int(i + S)])

{

swap(arr\_sort[i], arr\_sort[int(i + S)]);

}

}

S /= k;

}

while (true)

{

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

{

if (arr\_sort[i] > arr\_sort[i + 1])

{

swap(arr\_sort[i + 1], arr\_sort[i]);

}

else count++;

}

if (count == N - 1)

break;

else

count = 0;

}

cout << "Отсортированный массив:\n";

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

{

cout << arr\_sort[i] << ' ';

arr\_sort[i] = arr[i];

}

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

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

cout << "\nВремя сортировки:" << result1.count() << " ns" << endl;

cout << "\nShaker sort\n";

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

bool sort\_or\_not = true;

do {

sort\_or\_not = true;

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

if (arr\_sort[i] > arr\_sort[i + 1]) {

swap(arr\_sort[i], arr\_sort[i + 1]);

sort\_or\_not = false;

}

}

for (int i = 99; i >= 1; i--) {

if (arr\_sort[i] < arr\_sort[i - 1]) {

swap(arr\_sort[i], arr\_sort[i - 1]);

sort\_or\_not = false;

}

}

} while (sort\_or\_not == false);

cout << "Отсортированный массив:\n";

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

{

cout << arr\_sort[i] << " ";

arr\_sort[i] = arr[i];

}

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

auto result3 = duration\_cast<nanoseconds>(end3 - start3);

cout << "\nВремя сортировки:" << result3.count() << " ns" << endl;

cout << "\nInsert sort\n";

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

for (int j = 1; j < N; j++)

{

int x = arr\_sort[j];

int i = j - 1;

while ((i >= 0) && (x < arr\_sort[i]))

{

arr\_sort[i + 1] = arr\_sort[i];

i--;

}

arr\_sort[++i] = x;

}

cout << "Отсортированный массив:\n";

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

{

cout << arr\_sort[i] << " ";

}

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

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

cout << "\nВремя сортировки:" << result2.count() << " ns" << endl;

}

void quicksort(int\* arr\_sort, int end, int begin)

{

int mid;

int f = begin;

int l = end;

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

while (f < l)

{

while (arr\_sort[f] < mid) f++;

while (arr\_sort[l] > mid) l--;

if (f <= l)

{

swap(arr\_sort[f], arr\_sort[l]);

f++;

l--;

}

}

if (begin < l) quicksort(arr\_sort, l, begin);

if (f < end) quicksort(arr\_sort, end, f);

}

void findMaxMin()

{

setlocale(0, "rus");

cout << "Arr\_unsort" << "\nMaxMin:";

int max = -100;

int min = 199;

auto start4 = high\_resolution\_clock::now();

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

if (arr[i] > max) {

max = arr[i];

}

if (arr[i] < min) {

min = arr[i];

}

}

cout << max << " " << min;

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

auto result4 = duration\_cast<nanoseconds>(end4 - start4);

cout << "\nВремя поиска Max\_min:" << result4.count() << " ns" << endl;

cout << "\nArr\_sort" << "\nMaxMin\_sort:";

int max\_sort = arr\_sort[0];

int min\_sort = 199;

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

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

if (arr\_sort[i] > max\_sort) {

max\_sort = arr\_sort[i];

}

if (arr\_sort[i] < min\_sort) {

min\_sort = arr\_sort[i];

}

}

cout << max\_sort << " " << min\_sort;

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

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

cout << "\nВремя поиска MaxMin\_sort:" << result2.count() << " ns" << endl;

}

void findAvarege()

{

setlocale(LC\_ALL, "ru");

int max\_sort = arr\_sort[0];

int min\_sort = 199;

int n;

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

if (arr\_sort[i] > max\_sort) {

max\_sort = arr\_sort[i];

}

if (arr\_sort[i] < min\_sort) {

min\_sort = arr\_sort[i];

}

}

int max = -100;

int min = 199;

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

if (arr[i] > max) {

max = arr[i];

}

if (arr[i] < min) {

min = arr[i];

}

}

cout << "Arr\_sort\n";

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

n = (max\_sort + min\_sort) / 2;

int cnt1 = 0;

cout << "СрЗнач:" << " " << n << endl;

cout << "Индексы:" << " ";

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

if (arr\_sort[i] == n) {

cnt1++;

cout << i << " ";

}

}

cout << "\nКоличество:" << " " << cnt1;

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

auto result3 = duration\_cast<nanoseconds>(end3 - start3);

cout << "\nВремя выполнения:" << result3.count() << " ns" << endl;

cout << "\n\nArr\_unsort\n";

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

n = (max + min) / 2;

int cnt2 = 0;

cout << "СрЗнач:" << " " << n << endl;

cout << "Индексы:" << " ";

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

if (arr[i] == n) {

cnt2++;

cout << i << " ";

}

}

cout << "\nКоличество:" << " " << cnt2;

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

auto result5 = duration\_cast<nanoseconds>(end5 - start5);

cout << "\nВремя выполнения:" << result5.count() << " ns" << endl;

}

void findSmaller()

{

setlocale(LC\_ALL, "ru");

int a;

int cnt3 = 0;

cout << "\nВведите число a:";

cin >> a;

cout << "кол-во элементов меньше а:";

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

if (arr\_sort[i] < a) {

cnt3++;

}

}

cout << cnt3 << endl;

}

void findBigger()

{

setlocale(LC\_ALL, "ru");

int b;

int cnt3 = 0;

cout << "\nВведите число b:";

cin >> b;

cout << "кол-во элементов больше b:";

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

if (arr\_sort[i] > b) {

cnt3++;

}

}

cout << cnt3 << endl;

}

void binSearch()

{

setlocale(LC\_ALL, "ru");

int l = 0;

int r = N; // в этом случае присваивается именно n

int mid;

int key;

cout << "Бинарный поиск\n";

cout << "Введите ключ:";

cin >> key;

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

while (l < r) {

mid = (l + r) / 2; // считываем срединный индекс отрезка [l,r]

if (arr\_sort[mid] > key) r = mid; // проверяем, какую часть нужно отбросить с поиска

else l = mid + 1;

}

r--; // уменьшаем на один

if (arr\_sort[r] == key) cout << "Индекс элемента " << key << " в массиве равен: " << r;

else cout << "Извините, но такого элемента в массиве нет";

auto end6 = steady\_clock::now();

auto result6 = duration\_cast<nanoseconds>(end6 - start6);

cout << "\nВремя поиска:" << result6.count() << " ns" << endl;

cout << "\nПеребор\n";

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

int cnt5 = 0;

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

if (arr\_sort[i] == key)

cnt5++;

}

if (cnt5 > 0)

cout << "элемент найден";

else cout << "Извините, но такого элемента в массиве нет";

auto end7 = steady\_clock::now();

auto result7 = duration\_cast<nanoseconds>(end7 - start7);

cout << "\nВремя поиска:" << result7.count() << " ns" << endl;

if (result6.count() < result7.count()) cout << "\nБинарынй поиск быстрее перебора";

else cout << "\nПеребор быстрее бинарного поиска\n";

}

void replaceElements()

{

setlocale(LC\_ALL, "ru");

cout << "Ведите индексы от 0 до 99:" << ' ';

int z;

int y;

int temp;

cin >> z >> y;

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

if (z < N && z >= 0 && y < N && y >= 0) {

swap(arr\_sort[z], arr\_sort[y]);

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

cout << arr\_sort[i] << " ";

}

cout << endl;

}

else cout << "Индекс не найден" << endl;

auto end6 = steady\_clock::now();

auto result6 = duration\_cast<nanoseconds>(end6 - start6);

cout << "\nВремя выполнения:" << result6.count() << " ns" << endl;

}

//функция меню

int main() {

int end = N - 1, begin = 0;

for (; ; ) {

setlocale(LC\_ALL, "rus");

int choice;

cout << "\nWat du you want?\n";

cout << "1 - Create an arrey.\n";

cout << "2 - Sort arrey.\n";

cout << "3 - Find the maximum and minimum.\n";

cout << "4 - Output the avarage value.\n";

cout << "5 - Print the number smaller than the entered value.\n";

cout << "6 - Print the number bigger than the entered value.\n";

cout << "7 - Find out if ther is a number in the arrey.\n";

cout << "8 - Change arrey elements.\n";

cout << "9 - Exit.\n";

cout << "\nEnter the option number:";

cin >> choice;

switch (choice)

{

case 1:

createArray();

break;

case 2:

sort();

quicksort(arr\_sort, end, begin);

cout << endl << "Quick sort" "\nОтсортированный массив:\n";

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

cout << arr\_sort[i] << " ";

break;

case 3:

findMaxMin();

break;

case 4:

findAvarege();

break;

case 5:

findSmaller();

break;

case 6:

findBigger();

break;

case 7:

binSearch();

break;

case 8:

replaceElements();

break;

case 9:

cout << "Выход из программы...";

return 0; break;

default:

cout << "\nТакого пункта в меню нет.";

return 0; break;

}

}

}

Условный пример содержания пункта «Выполеннеие работы»

|  |  |
| --- | --- |
| Меню | |
| При запуске программы перед пользователем появляется окно с меню, где он может выбрать тип будущей вводимой последовательности | Меню: |

|  |  |
| --- | --- |
| 1 | |
| При выборе 1 пункта меню, создается и заполняется массив из 100 элементов. |  |

|  |  |
| --- | --- |
| 2 | |
| При выборе 2 пункта меню, производится сортировка массива. |  |

|  |  |
| --- | --- |
| 3 | |
| При выборе 3 пункта меню, находятся максимальный и минимальный элементы массива. | Идет вставка изображения... |

|  |  |
| --- | --- |
| 4 | |
| При выборе 4 пункта меню, выводится среднее значение. |  |

|  |  |
| --- | --- |
| 5 | |
| При выборе 5 пункта меню, находится кол-во элементов меньше заданного числа. |  |

|  |  |
| --- | --- |
| 6 | |
| При выборе 6 пункта меню, находится кол-во элементов больше заданного числа. |  |

|  |  |
| --- | --- |
| 7 | |
| При выборе 7 пункта меню, вычисляется есть ли в массиве заданное число. |  |

|  |  |
| --- | --- |
| 8 | |
| При выборе 8 пункта в меню, меняются местами элементы, индексы которых заданы пользователем. |  |