**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

|  |  |  |
| --- | --- | --- |
| Студентка гр. 4372 |  | Хальметова Ю.Р. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

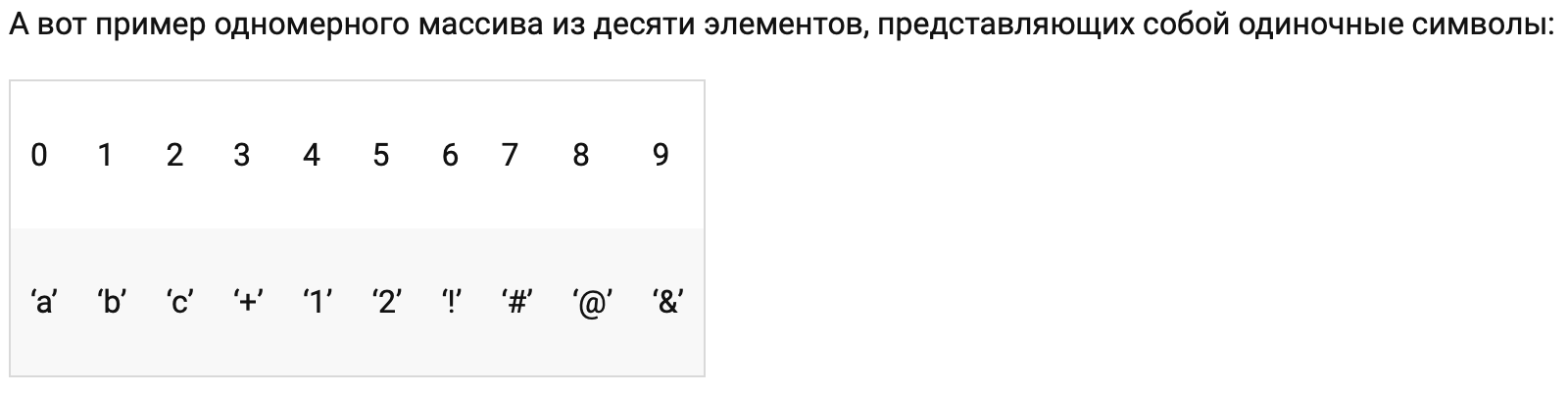
2024

**Цель работы.**

Разработать алгоритм и написать программу, которая позволяет создать целочисленный массив размером N = 100, заполнив его случайными значениями в диапазоне от -99 до 99. Затем массив нужно отсортировать методом сортировки от меньшего к большему, при этом важно измерить время, затраченное на сортировку, с использованием библиотеки chrono. После этого нужно найти максимальный и минимальный элементы массива, подсчитать время их поиска как в отсортированном, так и в неотсортированном массиве. Далее следует вычислить среднее значение максимального и минимального элементов, вывести индексы всех элементов, равных этим значениям, и их количество, также подсчитав время поиска. Затем нужно определить, сколько элементов в отсортированном массиве меньше заданного пользователем числа a и сколько элементов больше числа b, также введенного пользователем. Кроме того, необходимо проверить наличие введенного пользователем числа в отсортированном массиве с помощью алгоритма бинарного поиска и сравнить его скорость работы с обычным перебором. В завершение, следует реализовать обмен местами элементов массива по введенным пользователем индексам и вывести время, затраченное на этот обмен, используя библиотеку chrono.

**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.



Объявление в программах одномерных массивов выполняется в соответствии со следующим правилом:

<Базовый тип элементов> <Идентификатор массива> [<Количество элементов>]

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.

Многомерные массивы определяются аналогично одномерным массивам. Количество элементов по каждому измерению указывается отдельно в квадратных скобках:

int  A1 [5] [3];                 //  Двумерный массив, элементами которого являются

                                         //   значения типа int

double D [10] [15] [3];   //  Трехмерный массив, элементами которого являются

                                 //   значения типа double

Простейший циклический алгоритм вывода значений элементов некоторого одномерного массива выглядит так:

const int n = 10;

short A[n];

//  Для использования setw() необходимо включить #include <iomanip>

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

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

cout << endl;

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.

Обмен элементов массива осуществляется через буферную переменную либо через функцию swap(a, b).

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.

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.

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.

Сортировка вставками (insert sort) – алгоритм сортировки, в котором элементы массива просматриваются по одному, и каждый новый элемент размещается в подходящее место среди ранее упорядоченных элементов.

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.**

1. Создать целочисленный массив размером N = 100, заполнить его случайными значениями в диапазоне от -99 до 99.

2. Отсортировать массив методом сортировки от меньшего к большему, измеряя время, затраченное на сортировку, с использованием библиотеки chrono.

3. Найти максимальный и минимальный элементы массива, подсчитывая время их поиска как в отсортированном, так и в неотсортированном массиве.

4. Вычислить среднее значение максимального и минимального элементов, вывести индексы всех элементов, равных этим значениям, и их количество, также подсчитывая время поиска.

5. Определить количество элементов в отсортированном массиве, которые меньше заданного пользователем числа a, и количество элементов, которые больше числа b, также введённого пользователем.

6. Проверить наличие введённого пользователем числа в отсортированном массиве с помощью алгоритма бинарного поиска, сравнить его скорость работы с обычным перебором.

7. Реализовать обмен местами элементов массива по введённым пользователем индексам и вывести время, затраченное на этот обмен, с использованием библиотеки chrono.

**Выполнение работы.**

| Ввод пользователем и обработка данных | | Работа алгоритма и вывод на экран |
| --- | --- | --- |
| Меню | | |
| При запуске программы перед пользователем появляется, выбор задания. | | Меню: |
| Вывод целочисленного массива размерности 100 . | | |
| При вводе пользователем корректного значения пункта меню программа выполняется. | | Создает целочисленный массив размерности N = 100. Элементы массивы должны принимать случайное значение в диапазоне от -99 до 99. |
| Вывод на экран созданного отсортированного массива. | | |
| При вводе пользователем корректного значения пункта меню программа выполняется. | | Сортирует массив и выводит на экран время сортировки в наносекундах |
| Вывод на экран минимального и максимального элемента массива с замером времени | | |
| При вводе пользователем корректного значения пункта выполняется поиск наименьшего и наибольшего элементов массива | Ищет наибольший и наименьший элементы массива, замеряет время поиска и выводит на экран | |
| Вывод на экран среднего значения (если необходимо, число нужно округлить) максимального и минимального значения в отсортированном и неотсортированном. Вывод индексов всех элементов, которые равны этому значению, и их количество | | |
| При вводе пользователем корректного значения подсчет среднего значения максимального и минимального элементов | Считает среднее значение макс. и мин. элементов и выводит на экран индексы всех элементов, которые равны этому значению. Замеряет время подсчета и поиска и выводит на экран время в наносекундах. | |
| Вывод на экран количества элементов в отсортированном массиве, которые меньше числа a, которое инициализируется пользователем. | | |
| При вводе пользователем корректного значения пользователь вводит число a. | Производит подсчет чисел в массиве, которые меньше введенного пользователем числа, вывод на экран | |
| Вывод на экран количества элементов в отсортированном массиве, которые больше числа b, которое инициализируется пользователем. | | |
| При вводе пользователем корректного значения пользователь вводит число b. | Производит подсчет чисел в массиве, которые больше введенного пользователем числа, вывод на экран | |
| Вывод на экран информации о том, есть ли введенное пользователем число в отсортированном массиве. | | |
| При вводе пользователем корректного значения пользователь вводит число для поиска его в массиве. | Производит поиск введенного пользователем числа при помощи бинарного поиска и поиска перебором. Замеряет время подсчета и поиска и выводит на экран время в наносекундах. | |
| Обмен местами элементов массива, индексы которых вводит пользователь | | |
| При вводе пользователем корректного значения пользователь вводит число. | Производит обмен местами элементов, индексы которых вводит пользователь. Производит подсчет времени и выводит на экран. | |

**Выводы.**

В процессе выполнения работы были изучены одномереные статические массивы, библиотека сhrono и типы сортировок.

Приложение А

рабочий код

#include <iostream>

#include <set>

#include <chrono>

#include <vector>

#include <unordered\_set>

#include <unistd.h>

using namespace std;

using namespace chrono;

int binarySearch(int arr[], float number, int start, int end) {

if (end >= start) {

int mid = start + (end - start) / 2;

if (arr[mid] == number) {

return mid;

}

if (arr[mid] > number) {

return binarySearch(arr, number, start, mid - 1);

} else {

return binarySearch(arr, number, mid + 1, end);

}

}

return -1;

}

void quickSort(int arr[],int n ,int end, int start){

int mid;

int f = start;

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--;

}

}

}

void insertionSort(int arr[], int n){

int i, j, key;

for (i = 1; i < n; i++){

key = arr[i];

j = i - 1;

while (j >=0 && arr[j] > key){

arr[j + 1] = arr[j];

j--;

}

arr[j + 1] = key;

}

}

void combSort(int arr[], int n){

float k = 1.247;

int s;

s = n-1;

while (s >= 1){

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

if (arr[i] > arr[i + s]){

swap(arr[i], arr[i + s]);

}

}

s/=k;

}

}

void cocktailSort(int arr[], int n){

bool swapped = true;

int start = 0;

int end = n-1;

while (swapped){

swapped = false;

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

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

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

swapped = true;

}

}

if (!swapped){

break;

}

end--;

for(int i = end - 1;i >= start; i--){

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

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

swapped = true;

}

}

start++;

}

}

void bubbleSort(int arr[], int n) {

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

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

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

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

}

}

}

}

int main(){

const int n = 100;

int arr[n];

int originalArr[n];

srand(static\_cast<unsigned>(time(0)));

int count\_task = 0;

int max\_number = arr[0];

int min\_number = arr[0];

bool flag = true;

bool flag\_sort = false;

system("clear");

while (flag){

int task;

cout << "\n\nВыберите номер задания(1-8) (9 - завершение программы) (10 - идз): ";

cin >> task;

switch(task){

case 1:{

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

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

originalArr[i] = arr[i];

}

cout << "\nМассив до сортировки: ";

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

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

}

count\_task++;

max\_number = 0;

min\_number = 99;

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

if (arr[i] > max\_number) max\_number = arr[i];

if (arr[i] < min\_number) min\_number = arr[i];

}

break;

}

case 2:{

if (count\_task == 0){

cout << "\nМассив ещё не был сгенерирован! Попробуйте еще раз!";

break;

}

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

cocktailSort(arr,n);

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

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

cout << "\n\nМассив после сортировки: ";

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

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

}

cout <<"\n\nВремя сортировки в наносекундах = " <<result\_sort.count();

flag\_sort = true;

break;

}

case 3:{

if (count\_task == 0){

cout << "\nМассив ещё не был сгенерирован! Попробуйте еще раз!";

break;

}

cout << "\n\nВыберите в каком массиве производить поиск(1-сорт/0-несорт)";

int choice;

cin >> choice;

if (choice == 1){

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

max\_number = arr[99];

min\_number = arr[0];

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

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

cout << "\n\nНаибольший элемент массива(сорт): "<< max\_number;

cout << "\nНаименьший элемент массива(сорт): "<< min\_number;

cout << "\n\nВремя поиска макс и мин элементов в отсортированном массиве = " << result\_search\_sort.count() << " наносекунд";

}

else{

/\*

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

cout << originalArr[i]<<" ";

}

\*/

int max\_number = originalArr[0];

int min\_number = originalArr[0];

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

for (int number : originalArr) {

if (number > max\_number) max\_number = number;

if (number < min\_number) min\_number = number;

}

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

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

cout << "\n\nНаибольший элемент массива(несорт): "<< max\_number;

cout << "\nНаименьший элемент массива(несорт): "<< min\_number;

cout << "\n\nВремя поиска макс и мин элементов в отсортированном массиве = " << result\_search\_sort.count() << " наносекунд";

}

break;

}

case 4:

if (count\_task == 0) {

cout << "nМассив ещё не был сгенерирован! Попробуйте еще раз!";

break;

}

cout << "Выберите в каком массиве хотите произвести поиск(1-сорт/0-несорт): ";

int choice\_4;

cin >> choice\_4;

if (choice\_4 == 1) {

int mid\_num\_sort = (max\_number + min\_number) / 2.0;

cout << "\n\nСреднее значение макс и мин элементов в отсортированном массиве = " << mid\_num\_sort;

vector<int> indices;

auto start\_index\_search\_sort = steady\_clock::now();

int found\_index = binarySearch(arr, mid\_num\_sort, 0, n - 1);

if (found\_index == -1) {

cout << "\nЭлемент не найден в массиве!" << endl;

break;

}

for (int i = found\_index - 1; i >= 0 && arr[i] == mid\_num\_sort; i--) {

indices.push\_back(i);

}

indices.push\_back(found\_index);

for (int i = found\_index + 1; i < n && arr[i] == mid\_num\_sort; i++) {

indices.push\_back(i);

}

auto end\_index\_search\_sort = steady\_clock::now();

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

cout << "\n\nИндексы элементов среднего значения(сорт) = ";

for (int ind : indices) {

cout << ind << " ";

}

cout << "\nИх кол-во = " << indices.size() << endl;

cout << "\n\nВремя поиска индексов среднего значения в отсортированном массиве = " << result\_index\_search\_sort.count() << " наносекунд";

break;

}

if (choice\_4 == 0){

float mid\_num\_unsort = (max\_number + min\_number) / 2.0;

cout << "\n\nСреднее значение макс и мин элементов в неотсортированном массиве = " << mid\_num\_unsort;

vector<int> indeces;

auto start\_index\_search\_unsort = steady\_clock::now();

int count\_mid = 0;

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

if (mid\_num\_unsort == originalArr[i]){

indeces.push\_back(i);

count\_mid++;

}

}

auto end\_index\_search\_unsort = steady\_clock::now();

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

cout << "\n\nИндексы элементов среднего значения(несорт) = ";

for (int ind : indeces) {

cout << ind << " ";

}

cout << "\nИх кол-во = "<<count\_mid << endl;

cout << "\n\nВремя поиска индексов среднего значения в неотсортированном массиве = " << result\_index\_search\_unsort.count() << " наносекунд";

break;

}

case 5:{

if (count\_task == 0){

cout << "\nМассив ещё не был сгенерирован! Попробуйте еще раз!";

break;

}

int a;

cout << "\nВведите число a ";

cin >> a;

int count\_a = 0;

unordered\_set<int> unique\_numbers;

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

if (arr[i] < a) {

unique\_numbers.insert(arr[i]);

}

}

count\_a = unique\_numbers.size();

cout << "\n\nКол-во чисел в массиве, которые меньше введенного числа a = " << count\_a<< endl;

break;

}

case 6:{

if (count\_task == 0){

cout << "\nМассив ещё не был сгенерирован! Попробуйте еще раз!";

break;

}

int b;

cout << "\nВведите число b ";

cin >> b;

int count\_b = 0;

unordered\_set<int> unique\_numbers;

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

if (arr[i] > b) {

unique\_numbers.insert(arr[i]);

}

}

count\_b = unique\_numbers.size();

cout << "\n\nКол-во чисел в массиве, которые больше введенного числа b = " << count\_b << endl;

break;

}

case 7:{

if (count\_task == 0){

cout << "\nМассив ещё не был сгенерирован! Попробуйте еще раз!";

break;

}

cout << "\n\nВведите число для нахождения его при помощи бинарного поиска: ";

int number;

cin >> number;

auto binary\_searchnumber = steady\_clock::now();

binarySearch(arr, number,0,n-1);

auto binary\_searchnumber\_end = steady\_clock::now();

auto binary\_searchnumber\_time = duration\_cast<nanoseconds>(binary\_searchnumber\_end - binary\_searchnumber);

if (binarySearch(arr, number,0,n-1)){

cout << "\nЭлемент есть в массиве(бинарный)!"<< endl;

}

else{

cout << "\nЭлемент не найден в массиве(бинарный)!"<< endl;

}

cout << "Время выполнения бинарного поиска = " << binary\_searchnumber\_time.count() << " наносекунд";

bool flag = false;

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

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

if (arr[i] == number){

flag = true;

break;

}

}

auto searchnumber\_end = steady\_clock::now();

auto searchnumber\_time = duration\_cast<nanoseconds>(searchnumber\_end - searchnumber);

if (flag){

cout << "\n\nЭлемент есть в массиве!"<< endl;

}

else{

cout << "\n\nЭлемент не найден в массиве!"<< endl;

}

cout << "Время выполнения переборного поиска = " << searchnumber\_time.count() << " наносекунд";

flag = false;

break;

}

case 8:{

if (count\_task == 0){

cout << "\nМассив ещё не был сгенерирован! Попробуйте еще раз!";

break;

}

cout << "\n\nВведите индексы чисел, которые стоит поменять местами: ";

int index1, index2;

cin >> index1 >> index2;

auto swap\_time = steady\_clock::now();

swap(arr[index1], arr[index2]);

auto swap\_end = steady\_clock::now();

auto swap\_duration = duration\_cast<nanoseconds>(swap\_end - swap\_time);

cout << "\n\nМассив после замены элементов: ";

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

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

}

cout << "\n\nВремя замены элементов местами = " << swap\_duration.count() << " наносекунд";

break;

}

case 9:{

flag = false;

break;}

case 10:{

if (count\_task == 0){

cout << "\nМассив ещё не был сгенерирован! Попробуйте еще раз!";

break;

}

cout << "Введите с каким массивом хотите работать: (1 - сорт/0 - несорт): ";

int answer\_idz;

cin >> answer\_idz;

if (answer\_idz == 1){

if (flag\_sort == false){

cout << "Массив еще не был отсортирован, попробуйте снова!";

break;

}

int idz\_number;

cout << "Введите на какое значение будем уменшать каждый нечетный элемент: " << endl;

cin >> idz\_number;

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

if(i % 2 != 0){

arr[i] -= idz\_number;

}

}

cout << "Массив после уменьшения: " << endl;

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

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

}

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

if (i % 2 != 0){

arr[i] = arr[i] \* (rand() % 9 + 1);

}

}

cout << '\n';

cout << "Массив после умножение: " << endl;

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

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

}

vector <int>counts(10,0); //10 элементов и все равны 0

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

for (int j = 1; j < 10; ++j){

if(arr[i] % j == 0){

counts[j]++;

}

}

}

for(int i = 1; i < 10; i++){

cout << "\nКол-во элементов, которые нацело делятся на "<< i << " равно: "<< counts[i] << endl;

}

}

if (answer\_idz == 0){

int idz\_number;

cout << "Введите на какое значение будем уменшать каждый нечетный элемент: " << endl;

cin >> idz\_number;

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

if(i % 2 != 0){

originalArr[i] -= idz\_number;

}

}

cout << "Массив после уменьшения: " << endl;

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

cout << originalArr[i] << " ";

}

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

if (i % 2 != 0){

originalArr[i] = originalArr[i] \* (rand() % 9 + 1);

}

}

cout << '\n';

cout << "Массив после умножение: " << endl;

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

cout << originalArr[i] << " ";

}

vector <int>counts(10,0);

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

for (int j = 1; j < 10; ++j){

if(originalArr[i] % j == 0){

counts[j]++;

}

}

}

for(int i = 1; i < 10; i++){

cout << endl;

cout << "\nКол-во элементов, которые нацело делятся на "<< i << " равно: "<< counts[i] << endl;

}

}

}

}

}

}