**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра Информационных систем**

отчет

**по лабораторной работе №2**

**по дисциплине «Программирование»**

Тема: Типы данных и их внутреннее представление в памяти

|  |  |  |
| --- | --- | --- |
| Студент гр. 0323 |  | Кольцов К.Э |
| Преподаватель |  | Глущенко А.Г. |

Санкт-Петербург

2020

**Цель работы.**

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 элементы массива быстрой 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.

Должна присутствовать возможность запуска каждого пункта многократно.

**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.

Аналогом одномерного массива из математики может служить последовательность некоторых элементов с одним индексом: a\_i*ai*​ при  i = 0, 1, 2, … n – одномерный вектор. Каждый элемент такой последовательности представляет собой некоторое значение определенного типа данных.

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.

На практике массив обычно делят на две части: «меньше опорного» и «равные и большие» или «меньше опорного или равные» и «большие». Такой поход в общем случае эффективнее, ведь упрощается алгоритм разделения.

Бинарный поиск работает только в топ случае, если массив отсортирован. Например, если бы искомое минимальное значение стояло не на своем положенном месте, а на месте максимального элемента, то мы бы откинули его на первой же итерации. Сам алгоритм имеет вид:

1)    Определение значения в середине массива (или иной структуры данных). Полученное значение сравнивается с ключом (значением, которое необходимо найти).

2)    Если ключ меньше значения середины, то необходимо осуществлять поиск в первой половине элементов, иначе – во второй.

3)    Поиск сводится к тому, что вновь определяется значение серединного элемента в выбранной половине и сравнивается с ключом.

4)    Процесс продолжается до тех пор, пока не будет определен элемент, равный значению ключа или не станет пустым интервал для поиска.

**Экспериментальные результаты.**

Программа выполняет команды, которые пишет пользователь, всего команд 7:

s: заново создать массив;

b: найти все элементы больше вашего числа;

l: найти все элементы меньше вашего числа;

m: найти индексы среднего элемента;

w: поменять местам пару элементов в массиве;

f: найти индекс элемента, равного вашему числу;

e: завершить работу программы.

Команда h выводит список команд, команда e завершает программу, остальные команды вызывают соответствующие методы в коде программы.

Команда s позволяет заново сгенерировать массив, через функцию newArr, которая вызывается в начале программы для создания массива.

Команда b считывает число с ввода и выдает количество элементов больше заданного. Команда l работает аналогично для чисел меньше заданного.

Команда m находит количество и индексы элементов в массиве, равных половине суммы максимального и минимального элементов массива.

Команда w запрашивает пару чисел и меняет местами два элемента массива с соответствующими индексами. Если индексы не совпадают, то массив становится неотсортированным, из-за чего перед каждой функцией, требующей порядка в массиве проводится сортировка.

Команда f находит число, введённое пользователем, в массиве бинарным поиском и простым перебором, для сравнения скорости работы.

Работу каждой функции можно просмотреть отдельно (см. рис. 1, 2, 3, 4)

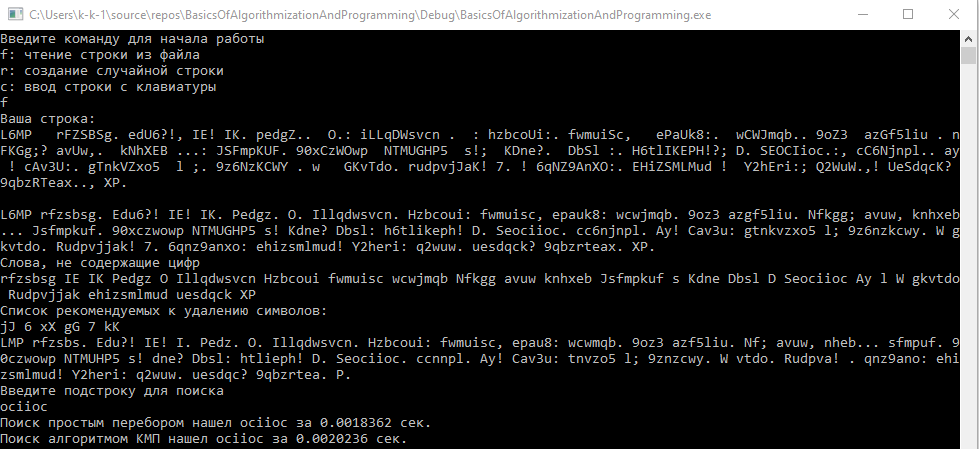


Рисунок 1 – Начало работы программы

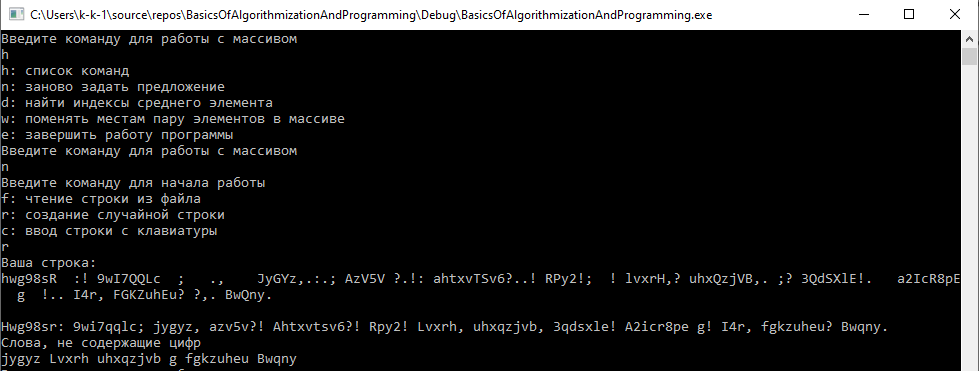


Рисунок 2 – Команды h и n

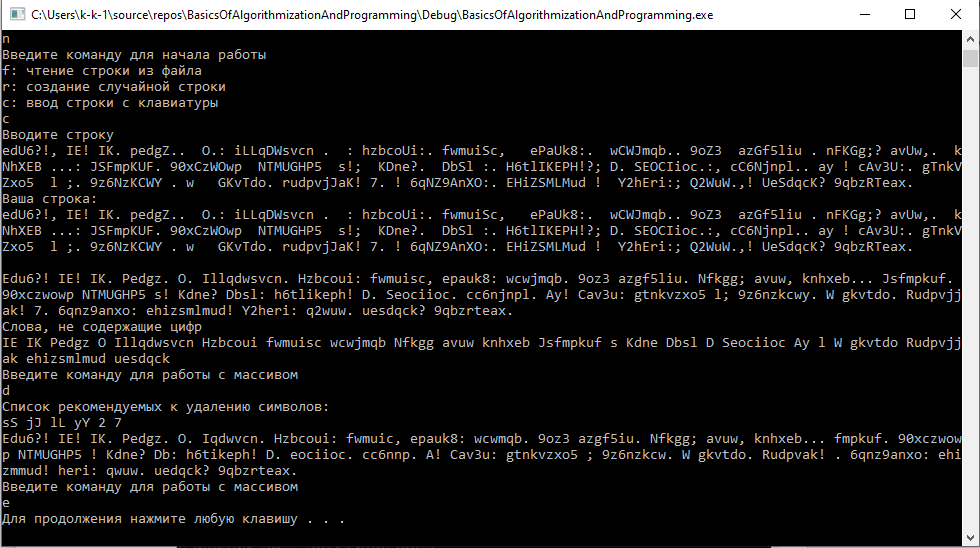


Рисунок 3 – Варианты команды n и выход из программы

**Обработка результатов эксперимента.**

Программа выводит результат корректной сортировки и правильно находит элементы в отсортированном массиве.

**Выводы.**

В ходе данной лабораторной работы я изучил различные способы сортировки массивов и работы с ними.

ПРИЛОЖЕНИЕ А

Листинг программы

#include <iostream>

#include <ctime>

#include <chrono>

#define N 100

bool wasArrSort;

void newArr(int\*, int\*);

void quicksort(int\*, int, int);

int findElem(int\*, int, int, int);

int simpFindElem(int\*, int, int, int);

void findAllMed(int\*, int, int);

double findPlace(int\*, int, int, int);

void userFindLess(int\*);

void userFindMore(int\*);

void userSwap(int\*);

void userFindNumber(int\*);

int main(){

std::cout.setf(std::ios::fixed);

std::cout.precision(7);

std::srand(time(0));

setlocale(LC\_ALL, "Russian");

int arrUnsort[N], arrSort[N];

newArr(arrUnsort, arrSort);

char answer;

bool flag = true;

while (flag) {

std::cout << "Введите команду для работы с массивом\n";

std::cin >> answer;

switch (answer)

{

case 'h':

std::cout << "h: список команд\n" \

"s: заново создать массив\n" \

"b: найти все элементы больше вашего числа\n" \

"l: найти все элементы меньше вашего числа\n" \

"m: найти индексы среднего элемента\n" \

"w: поменять местам пару элементов в массиве\n" \

"f: найти индекс элемента, равного вашему числу\n" \

"e: завершить работу программы";

break;

case 's':

newArr(arrUnsort, arrSort);

break;

case 'b':

userFindMore(arrSort);

break;

case 'l':

userFindLess(arrSort);

break;

case 'm':

findAllMed(arrSort, 0, N - 1);

break;

case 'w':

userSwap(arrSort);

break;

case 'f':

userFindNumber(arrSort);

break;

case'e':

flag = false;

break;

default:

std::cout << "Вы указали несуществующую команду";

break;

}

}

system("pause");

return 0;

}

void newArr(int\* arrUnsort, int\* arrSort) {

std::chrono::system\_clock::time\_point start, end;

std::chrono::duration<double> sortTime, findTime;

int min, max;

std::cout << "Несортированный массив: ";

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

{

arrUnsort[i] = (std::rand() % 199) - 99;

arrSort[i] = arrUnsort[i];

std::cout << arrUnsort[i] << " ";

}

wasArrSort = false;

std::cout << "\n\n";

start = std::chrono::system\_clock::now();

quicksort(arrSort, 0, N - 1);

end = std::chrono::system\_clock::now();

sortTime = end - start;

wasArrSort = true;

std::cout << "Отсортированный массив: ";

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

std::cout << arrSort[i] << ' ';

std::cout << "\n\n";

std::cout << "Сортировка заняла " << sortTime.count() << " сек.\n\n";

start = std::chrono::system\_clock::now();

min = max = arrUnsort[0];

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

if (arrUnsort[i] > max)

max = arrUnsort[i];

else if (arrUnsort[i] < min)

min = arrUnsort[i];

}

end = std::chrono::system\_clock::now();

std::cout << "Минимальный элемент = " << min << ", максимальный элемент = " << max << '\n';

findTime = end - start;

std::cout << "Поиск в неотсортированном массиве занял " << findTime.count() << " сек.\n";

start = std::chrono::system\_clock::now();

min = arrSort[0];

max = arrSort[N - 1];

end = std::chrono::system\_clock::now();

findTime = end - start;

std::cout << "Поиск в отсортированном массиве занял " << findTime.count() << " сек.\n";

std::cout << "Сортировка и поиск в массиве заняли " << (findTime + sortTime).count() << " сек.\n\n";

}

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

{

int f = begin, l = end, mid = arr[(f + l) / 2];

while (f <= l)

{

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

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

if (f <= l)

{

std::swap(arr[f], arr[l]);

f++;

l--;

}

}

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

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

}

void findAllMed(int\* arr, int begin, int end) {

if (!wasArrSort) {

std::cout << "Для выполнения операции, нужно отсортировать массив\n";

wasArrSort = true;

std::cout << "Отсортированный массив: ";

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

std::cout << arr[i] << ' ';

std::cout << "\n\n";

}

int med = (arr[begin] + arr[end]) / 2;

std::cout << "Среднее значение = " << med << '\n';

int sum = 0, count = findElem(arr, begin, end, med);

if (count == -1) {

std::cout << "Средний элемент = " << med << " в массиве не найден\n\n";

}

else {

while (arr[count] == med)

count--;

count++;

std::cout << "Средний элемент = " << med << " и находится по индексу ";

while (arr[count] == med) {

std::cout << count << ", ";

count++;

sum++;

}

std::cout << "и всего их " << sum << "\n\n";

}

}

int findElem(int\* arr, int begin, int end, int elem) {

if (!wasArrSort) {

std::cout << "Для выполнения операции, нужно отсортировать массив\n";

wasArrSort = true;

std::cout << "Отсортированный массив: ";

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

std::cout << arr[i] << ' ';

std::cout << "\n\n";

}

if ((elem < arr[begin]) || (elem > arr[end]))

return -1;

double count = findPlace(arr, begin, end, elem);

if (((int)count) == ((int)(count + 0.5)))

return (int)count;

return -1;

}

int simpFindElem(int\* arr, int begin, int end, int elem) {

if (!wasArrSort) {

std::cout << "Для выполнения операции, нужно отсортировать массив\n";

wasArrSort = true;

std::cout << "Отсортированный массив: ";

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

std::cout << arr[i] << ' ';

std::cout << "\n\n";

}

if ((elem < arr[begin]) || (elem > arr[end]))

return -1;

for (int i = begin;i <= end;i++)

if (arr[i] == elem)

return i;

return -1;

}

double findPlace(int\* arr, int begin, int end, int elem) {

if ((end - begin) < 2) {

if (elem == arr[end])

return end;

if (elem == arr[begin])

return begin;

return (double)(end + begin) / 2;

}

else{

int count = (begin + end) / 2;

if (elem > arr[count])

return findPlace(arr, count, end, elem);

return findPlace(arr, begin, count, elem);

}

}

void userFindLess(int\* arr) {

if (!wasArrSort) {

std::cout << "Для выполнения операции, нужно отсортировать массив\n";

wasArrSort = true;

std::cout << "Отсортированный массив: ";

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

std::cout << arr[i] << ' ';

std::cout << "\n\n";

}

int ab;

double placeAB;

std::cout << "Введите число a, чтобы узнать, сколько чисел меньше него\n";

while (true) {

std::cin >> ab;

if (std::cin)

break;

std::cout << "Вы неправильно ввели число. Повторите ввод\n";

std::cin.clear();

std::cin.ignore(INT\_MAX, '\n');

}

if (ab < arr[0])

std::cout << "0 элементов меньше чем " << ab << "\n\n";

else {

int count;

if (ab > arr[N - 1])

std::cout << "100 элементов меньше чем " << ab << "\n\n";

else {

double countD = findPlace(arr, 0, N - 1, ab);

if (((int)countD) == ((int)(countD + 0.5)))

count = (int)countD;

else

count = (int)(countD - 0.5);

while (arr[count] == ab)

count--;

std::cout << count + 1<< " элементов меньше чем " << ab << "\n\n";

}

}

}

void userFindMore(int\* arr) {

if (!wasArrSort) {

std::cout << "Для выполнения операции, нужно отсортировать массив\n";

wasArrSort = true;

std::cout << "Отсортированный массив: ";

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

std::cout << arr[i] << ' ';

std::cout << "\n\n";

}

int ab;

double placeAB;

std::cout << "Введите число b, чтобы узнать, сколько чисел больше него\n";

while (true) {

std::cin >> ab;

if (std::cin)

break;

std::cout << "Вы неправильно ввели число. Повторите ввод\n";

std::cin.clear();

std::cin.ignore(INT\_MAX, '\n');

}

if (ab > arr[N - 1])

std::cout << "0 элементов больше чем " << ab << "\n\n";

else {

int count;

if (ab < arr[0])

std::cout << "100 элементов больше чем " << ab << "\n\n";

else {

double countD = findPlace(arr, 0, N - 1, ab);

if (((int)countD) == ((int)(countD + 0.5)))

count = (int)countD;

else

count = (int)(countD + 0.5);

while (arr[count] == ab)

count++;

std::cout << N - count << " элементов больше чем " << ab << "\n\n";

}

}

}

void userFindNumber(int\* arr) {

if (!wasArrSort) {

std::cout << "Для выполнения операции, нужно отсортировать массив\n";

wasArrSort = true;

std::cout << "Отсортированный массив: ";

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

std::cout << arr[i] << ' ';

std::cout << "\n\n";

}

int a;

std::chrono::system\_clock::time\_point start, end;

std::chrono::duration<double> binFindTime, simpFindTime;

std::cout << "Введите число, которое нужно найти в массиве\n";

while (true) {

std::cin >> a;

if (std::cin)

break;

std::cout << "Вы неправильно ввели число. Повторите ввод\n";

std::cin.clear();

std::cin.ignore(INT\_MAX, '\n');

}

int countBin, countSimp;

start = std::chrono::system\_clock::now();

countBin = findElem(arr, 0, N - 1, a);

end = std::chrono::system\_clock::now();

binFindTime = end - start;

start = std::chrono::system\_clock::now();

countSimp = simpFindElem(arr, 0, N - 1, a);

end = std::chrono::system\_clock::now();

simpFindTime = end - start;

if (countBin == -1)

std::cout << "Вашего числа нет в массиве,\n";

else

std::cout << "Ваше число соответствует элементу массива с индексом " << countBin << " и " << countSimp << '\n';

std::cout << "На поиск ответа ушло " << binFindTime.count() << " сек. бинарным поиском и " << simpFindTime.count() << " сек. простым перебором\n\n";

}

void userSwap(int\* arr) {

int a, b;

std::chrono::system\_clock::time\_point start, end;

std::chrono::duration<double> swapTime;

std::cout << "Перестановка элементов массива. Введите индексы через пробел\n";

while (true) {

std::cin >> a >> b;

if ((std::cin)&&(a>=0)&&(b>=0)&&(b<100)&&(a<100))

break;

std::cout << "Вы неправильно ввели пару чисел. Повторите ввод\n";

std::cin.clear();

std::cin.ignore(INT\_MAX, '\n');

}

if (a == b)

std::cout << "Вы ввели одинаковые числа\n";

else

wasArrSort = false;

start = std::chrono::system\_clock::now();

std::swap(arr[a], arr[b]);

end = std::chrono::system\_clock::now();

swapTime = end - start;

std::cout << "Перестановка элементов заняла " << swapTime.count() << " сек.\n";

std::cout << "Измененый массив: ";

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

std::cout << arr[i] << ' ';

std::cout << "\n\n";

}