**Московский Авиационный Институт**

(Национальный Исследовательский Университет)

Институт №8 «Информационные технологии и прикладная математика»

Кафедра 806 «Вычислительная математика и программирование»

Курсовой проект

по дисциплине «Практикум программирования»

2 семестр

Задание 9.

Тема: «Сортировка и поиск»

|  |  |
| --- | --- |
| Студент: | Доянова В. А. |
| Группа: | М8О-111Б-22 |
| Преподаватель: | Аносова Н. П. |
| Подпись: |  |
| Оценка: |  |

Москва, 2023

Содержание:

**Задание3**

Формулировка3

**Работа4**

Код программы4

Описание алгоритма7

Протокол выполнения программы8

**Вывод11**

**Задание**

**Формулировка**

Составить программу на языке Си с использованием процедур и функций для сортировки таблицы заданным методом и двоичного поиска по ключу в таблице.

Программа должна **вводить** значения элементов неупорядоченной таблицы и проверять работу процедуры

сортировки в трех случаях: (1) элементы таблицы с самого начала упорядочены; (2) элементы таблицы расставлены в обратном порядке; (3) элементы таблицы не упорядочены.

Для каждого вызова процедуры сортировки необходимо печатать исходное состояние таблицы и результаты сортировки. После выполнения сортировки программа должна вводить ключи и для каждого из них выполнять поиск в упорядоченной таблице с помощью процедуры двоичного поиска и печатать найденные элементы, если они присутствуют в таблице.

Метод сортировки: 11. Быстрая сортировка Хоара.

Тип ключа: целый.

**Работа**

**Код программы**

Файл head.h:   
#ifndef QUICK\_SORT\_H

#define QUICK\_SORT\_H

#include <vector>

#include <string>

using namespace std;

struct Record {

string value;

int key;

};

void quickSort(vector<Record>& arr, int left, int right);

bool is\_ordered(const vector<Record>& arr);

string find\_record(const vector<Record>& arr, int key);

#endif

Файл functions.cpp:

#include "head.h"

#include <iostream>

using namespace std;

int partition(vector<Record>& arr, int left, int right) {

int p = arr[(left + right) / 2].value.length();

int i = left - 1;

int j = right + 1;

while (true) {

do {

i++;

} while (arr[i].value.length() < p);

do {

j--;

} while (arr[j].value.length() > p);

if (i >= j) {

return j;

}

swap(arr[i], arr[j]);

}

}

void quickSort(vector<Record>& arr, int left, int right) {

if (left < right) {

int p = partition(arr, left, right);

quickSort(arr, left, p);

quickSort(arr, p + 1, right);

}

}

bool is\_ordered(const vector<Record>& arr) {

for (int i = 1; i < arr.size(); i++) {

if (arr[i].value.length() < arr[i - 1].value.length()) {

return false;

}

}

return true;

}

string find\_record(const vector<Record>& arr, int key) {

string s;

for (const auto& r : arr) {

if (r.key == key) {

s += r.value + " ";

}

}

return s;

}

Файл main.cpp:  
#include <fstream>

#include <iostream>

#include <algorithm>

#include <vector>

#include <iomanip>

#include "head.h"

using namespace std;

int main() {

setlocale(LC\_ALL, "Russian");

ifstream inputFile("input.txt");

vector<Record> arr;

while (true) {

Record r;

inputFile >> r.value >> r.key;

if (!inputFile.good()) break;

arr.push\_back(r);

}

cout << "Исходная таблица:" << endl;

cout << "|\tЗначение" << setw(32) << "| Ключ\t|" << endl;

cout << "|---------------------------------------|-------|\n";

for (const auto& r : arr) {

cout << "|\t" << setw(32) << left << r.value << "|" << r.key << "\t|" << endl;

}

cout << "|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_|\n";

vector<Record> reversed\_arr(arr);

reverse(reversed\_arr.begin(), reversed\_arr.end());

if (is\_ordered(arr)) {

cout << "Исходная таблица уже отсортирована\n" << endl;

}

else if (is\_ordered(reversed\_arr)) {

cout << "Исходная таблица отсортирована в обратном порядке\n" << endl;

quickSort(arr, 0, arr.size() - 1);

cout << "Отсортированная таблица:" << endl;

cout << "|\tЗначение" << setw(32) << "| Ключ\t|" << endl;

cout << "|---------------------------------------|-------|\n";

for (const auto& r : arr) {

cout << "|\t" << setw(32) << left << r.value << "|" << r.key << "\t|" << endl;

}

cout << "|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_|\n";

}

else {

cout << "Исходная таблица не отсортирована\n" << endl;

quickSort(arr, 0, arr.size() - 1);

cout << "Отсортированная таблица:" << endl;

cout << "|\tЗначение\t\t\t" << setw(32) << "| Ключ\t|" << endl;

cout << "|---------------------------------------|-------|\n";

for (const auto& r : arr) {

cout << "|\t" << setw(32) << left << r.value << "|" << r.key << "\t|" << endl;

}

cout << "|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_|\n";

}

int key;

cout << "Введите ключ, чтобы найти элемент: ";

cin >> key;

string found\_records = find\_record(arr, key);

if (!found\_records.empty()) {

cout << "Элемент с ключом " << key << ": " << found\_records << endl;

}

else {

cout << "Элемент с ключом " << key << " не найден" << endl;

}

return 0;

}

**Описание алгоритма**

Файл head.h: это заголовочный файл. В нем указаны использованные функции, а также определена структура Record, которая состоит из строки value (значение в таблице) и целого значения key (ключа таблицы).

Файл functions.cpp: это файл с функциями.

* Функция partition: эта функция необходима для сортировки таблицы (сортировка производится на основе длины элементов, так как они представляют из себя строки). Она выбирает значение из середины списка, а затем сравнивает с соседними элементами. В результате сравнения элементы меняются местами таким образом, чтобы список был отсортирован по возрастанию длины элементов.
* Функция quickSort: эта функция помещает элементы в предыдущую функцию для получения сравнения.
* Функция is\_ordered: функция проверяет, отсортирована ли таблица. Она в цикле идет по списку и проверяет упорядоченность элементов.
* Функция find\_record: данная функция получает на вход ключ и ищет в таблице элемент, соответствующий полученному ключу.

Файл main.cpp: эта основная часть программы. Изначально выводится исходная таблица. Затем она проверяется на упорядоченность и в результате выводится один из вариантов: таблица уже отсортирована (выводится еще раз исходная таблица), таблица не отсортирована (выводится отсортированная таблица), таблица отсортирована в обратном порядке (также выводится уже отсортированная таблица).   
Далее пользователь вводит ключ, а программа выводит элемент(-ы), соответствующие этому ключу.

**Протокол выполнения программы**

1. Входные данные:

aaaaaeaaaa 1

bbbbbbbbbbbbbbbbb 2

ccccccccccccc 3

ggsfogospsgggffoogks 89

gagaagagagag 5

dddd 4

eeeee 5

ffffff 6

ggggggg 7

etherth6 0

odofogkokfkdlgk 87

dkgokdjdsjjo 134

Вывод программы:

Изображение выглядит как снимок экрана, текст, Мультимедийное программное обеспечение, Графическое программное обеспечение

Автоматически созданное описание

1. Входные данные:

dddd 1

eeeee 2

ffffff 3

ggggggg 7

etherth6 0

aaaaaeaaaa 4

gagaagagagag 89

dkgokdjdsjjo 134

ccccccccccccc 5

odofogkokfkdlgk 87

bbbbbbbbbbbbbbbbb 5

ggsfogospsgggffoogks 6

Вывод программы:

Изображение выглядит как текст, снимок экрана, Мультимедийное программное обеспечение, программное обеспечение

Автоматически созданное описание

1. Входные данные:

ggsfogospsgggffoogks 6

bbbbbbbbbbbbbbbbb 5

odofogkokfkdlgk 87

ccccccccccccc 5

dkgokdjdsjjo 134

gagaagagagag 89

aaaaaeaaaa 4

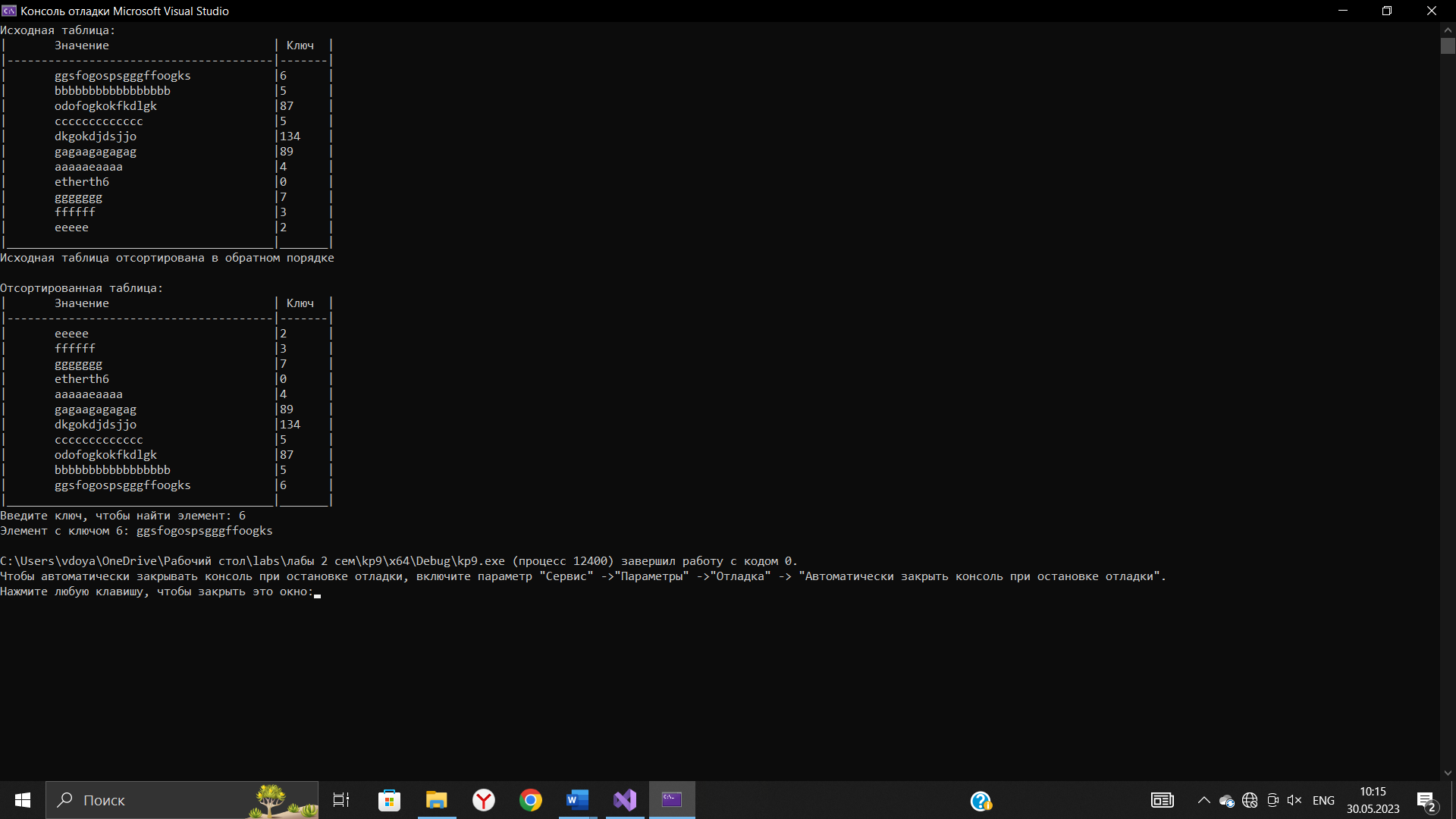
etherth6 0

ggggggg 7

ffffff 3

eeeee 2

dddd 1

Вывод программы:  


**Вывод**

В ходе выполнения данной курсовой работы я научилась осуществлять сортировку данных. Я разобралась с тем, как проверять данные на упорядоченность, как выводить их после сортировки. Также я научилась осуществлять поиск по таблице согласно заданному параметру.