|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ | | | | | | | | | | | | |
| Федеральное государственное автономное образовательное учреждение  высшего образования | | | | | | | | | | | | |
| **Дальневосточный федеральный университет** | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
| **ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ** | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
| **Кафедра прикладной математики, механики, управления и программного обеспечения** | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
| **О Т Ч Е Т** | | | | | | | | | | | | |
| по лабораторной работе №2.2  дисциплина «Фундаментальные структуры данных и алгоритмы» | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | |  |  | Студент гр. Б9120-09.03.04прогин | | | |
|  |  |  | | О. Г. Александрович | |
|  | | | | | | |  |  | (подпись) | |  | |
|  | | | | | | | | | | | | |
|  | | | | | | |  |  | Руководитель | | |  |
|  | | | | | | |  |  | ст. преподаватель | | | |
|  | | | |  |  | |  |  |  |  | О.А. Крестникова | |
|  | | | |  |  | |  |  | (подпись) |  | (И.О. Фамилия) | |
|  |  |  |  | | |  |  |  |  | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
| г. Владивосток | | | | | | | | | | | | |
| 2022 | | | | | | | | | | | | |

Оглавление

[Постановка задачи 2](#_Toc105278079)

[Описание алгоритмов поиска 2](#_Toc105278080)

[Линейный поиск 2](#_Toc105278081)

[Рекурсивный бинарный 2](#_Toc105278082)

[Тесты 3](#_Toc105278083)

[Входные и выходные файлы 3](#_Toc105278084)

[Входной файл для тестов (Текстовый файл №1) 3](#_Toc105278085)

[Дополнительный файл, с одинаковыми записями (Текстовый файл №2) 3](#_Toc105278086)

[Таблица с тестами 3](#_Toc105278087)

[Текст программы 6](#_Toc105278088)

[Основной файл 6](#_Toc105278089)

[Заголовочный файл класса Search\_01 6](#_Toc105278090)

[Файл с кодом методов, для класса Search\_01 7](#_Toc105278091)

# Постановка задачи

Алгоритмы поиска (по цифровому полю)

**Постановка задачи:**

**Входные данные:** ключ поиска; текстовый файл, каждая из строк которого содержит данные задания 1.3.

**Выходные данные:** номер строки в исходном файле, содержащей заданный ключ поиска; количество шагов поиска (линейный поиск + см. свой вариант задания) (без учета сортировки любым алгоритмом задания 1.3).

# Описание алгоритмов поиска

## Линейный поиск

Последовательный (линейный) поиск – это простейший вид поиска заданного элемента на некотором множестве, осуществляемый путем последовательного сравнения очередного рассматриваемого значения с искомым до тех пор, пока эти значения не совпадут.

Идея этого метода заключается в следующем. Множество элементов просматривается последовательно в некотором порядке, гарантирующем, что будут просмотрены все элементы множества (например, слева направо). Если в ходе просмотра множества будет найден искомый элемент, просмотр прекращается с положительным результатом; если же будет просмотрено все множество, а элемент не будет найден, алгоритм должен выдать отрицательный результат.

## Рекурсивный бинарный

Бинарный поиск — тип поискового алгоритма, который последовательно делит пополам заранее отсортированный массив данных, чтобы обнаружить нужный элемент.

Двоичный поиск заключается в том, что на каждом шаге множество объектов делится на две части и в работе остаётся та часть множества, где находится искомый объект. Или же, в зависимости от постановки задачи, мы можем остановить процесс, когда мы получим первый или же последний индекс вхождения элемента.

# Тесты

## Входные и выходные файлы

### Входной файл для тестов (Текстовый файл №1)

Новикова Ольга Нинаовна, Руководитель, 2.11.2021, 28 лет

Морозова Вера Елизаветаовна, Продюсер, 17.11.2021, 13 лет

Соколова Юлия Иринаовна, Астроном, 29.11.2021, 1 лет

Соколова Юлия Любовьовна, Дизайнер ландшафта, 10.11.2021, 20 лет

Соловьёва Кира Вераовна, Микробиолог, 4.11.2021, 26 лет

Петрова Кира Викторияовна, Переводчик, 10.11.2021, 20 лет

Зайцева Екатерина Нинаовна, Врач, 12.11.2021, 18 лет

Семёнова Кира Юлияовна, Исследователь, 28.11.2021, 2 лет

Иванова Полина Ангелинаовна, Генетик, 9.11.2021, 21 лет

Семёнова Екатерина Надеждаовна, Хирург, 9.11.2021, 21 лет

Волкова Ангелина Софияовна, Авиадиспетчер, 3.11.2021, 27 лет

Соколова Ольга Анастасияовна, Дизайнер, 29.11.2021, 1 лет

Новикова Ольга Надеждаовна, Системный программист, 4.11.2021, 26 лет

Богданова Наталья Любовьовна, Врач, 4.11.2021, 26 лет

Соловьёва Ирина Светланаовна, Исследователь, 3.11.2021, 27 лет

Богданова Кристина Юлияовна, Пожарный, 15.11.2021, 15 лет

Волкова Нина Вераовна, Повар, 16.11.2021, 14 лет

Козлова Елизавета Василисаовна, Программист, 28.11.2021, 2 лет

Петрова Елизавета Александраовна, Руководитель, 2.11.2021, 28 лет

Смирнова Любовь Викторияовна, Пожарный, 14.11.2021, 27 лет

### Дополнительный файл, с одинаковыми записями (Текстовый файл №2)

Зайцева Екатерина Ниновна, Врач, 12.11.2021, 18 лет

Зайцева Екатерина Ниновна, Врач, 12.11.2021, 18 лет

Зайцева Екатерина Ниновна, Врач, 12.11.2021, 18 лет

Зайцева Екатерина Ниновна, Врач, 12.11.2021, 18 лет

Зайцева Екатерина Ниновна, Врач, 12.11.2021, 18 лет

Зайцева Екатерина Ниновна, Врач, 12.11.2021, 18 лет

Зайцева Екатерина Ниновна, Врач, 12.11.2021, 18 лет

Зайцева Екатерина Ниновна, Врач, 12.11.2021, 18 лет

Зайцева Екатерина Ниновна, Врач, 12.11.2021, 18 лет

Зайцева Екатерина Ниновна, Врач, 12.11.2021, 18 лет

Зайцева Екатерина Ниновна, Врач, 12.11.2021, 18 лет

Зайцева Екатерина Ниновна, Врач, 12.11.2021, 18 лет

Зайцева Екатерина Ниновна, Врач, 12.11.2021, 18 лет

Зайцева Екатерина Ниновна, Врач, 12.11.2021, 18 лет

Зайцева Екатерина Ниновна, Врач, 12.11.2021, 18 лет

Зайцева Екатерина Ниновна, Врач, 12.11.2021, 18 лет

Зайцева Екатерина Ниновна, Врач, 12.11.2021, 18 лет

Зайцева Екатерина Ниновна, Врач, 12.11.2021, 18 лет

Зайцева Екатерина Ниновна, Врач, 12.11.2021, 18 лет

Зайцева Екатерина Ниновна, Врач, 12.11.2021, 18 лет

## Таблица с тестами

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Описание тестовой ситуации** | **Алгоритм поиска** | **Входные данные** | | **Результат выполнения алгоритма** |
| **Текстовый файл** | **Номер для поиска** |
| Ключ поиска содержится один раз, разное количество шагов до нахождения ответа у алгоритмов | Линейный | Текстовый файл №1 | 14 | Изображение выглядит как текст  Автоматически созданное описание |
| Бинарный |
| Ключ поиска содержится несколько раз в исходном файле, разные ответы у алгоритмов | Линейный | Текстовый файл №1 | 27 |  |
| Бинарный |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Результат поиска при 20 одинаковых записях, алгоритмы вернули разные строки, в которых нашли совпадения | Линейный | Текстовый файл №2 | 18 |  |
| Бинарный |
| Когда ключа не содержится в исходных данных | Линейный | Текстовый файл №1 | 0 |  |
| Бинарный |

# Текст программы

## Основной файл

#include <iostream>

#include <cmath>

#include <time.h>

#include <fstream>

#include <string>

#include <limits.h>

#include <string.h>

#include <stdio.h>

#include "Search\_01.h"

using namespace std;

void main()

{

Search\_01 Search\_01;

setlocale(LC\_ALL, "rus");

srand(time(0));

Search\_01.LinearSearch("OutFile", 2);

Search\_01.BinarySearch("OutFile", 2);

cout << "Поиск успешно завершён" << endl;

}

## Заголовочный файл класса Search\_01

#pragma once

#include <iostream>

#include <cmath>

#include <time.h>

#include <fstream>

using namespace std;

class Search\_01

{

public:

struct AQ // Структура нужных нам данных

{

string d; // Должность

int s; // Стаж

string fio; // ФИО

int id = 0; // id

};

// Получаем на вход имя файла, и потом загружаем из него во внутренние массивы программы данные

Search\_01::AQ\* Processing\_data\_from\_the\_input\_file(string NameFile);

void coutM(AQ\* mass, int n); // Выводим данные в консоль

void coutQ(AQ mass); // Выводим в консоль только нужную нам строку данных

//// ----- Линейный поиск:

// Процедура, для пользователя

int LinearSearch(string NameFile, int pattStaj);

// Основная процедура запуска линейного поиска

int LinearSearchInsideProc(Search\_01::AQ\* massNS, int pattStaj);

// Печать результатов в файл

void PrintFromOutFile\_LinearSearch(int i, int pattStaj, Search\_01::AQ\* massNS);

//// ----- Бинарный (рекурсивная реализация):

// Основная процедура запуска бинарного поиска

int BinRecSearch(Search\_01::AQ\* massNS, int pattStaj);

// Алгоритм бинарного рекурсивного поиска

int binaryS(Search\_01::AQ\* massNS, int l, int r, int x);

// Процедура, для пользователя

int BinarySearch(string NameFile, int pattStaj);

// Печать результатов в файл

void PrintFromOutFile\_BinarySearch(int i, int pattStaj, Search\_01::AQ\* massNS);

// Сортировка пузырьком

Search\_01::AQ\* BubbleSort(Search\_01::AQ\* massNS);

};

## Файл с кодом методов, для класса Search\_01

#include "Search\_01.h"

#include <iostream>

#include <cmath>

#include <time.h>

#include <fstream>

#include <stdlib.h>

#include <iomanip>

#include <string>

#include <Windows.h>

#include <stdlib.h>

#include <string.h>

#include <stdio.h>

#include <errno.h>

using namespace std;

// Если вам мешают уведомления об ошибках, то просто отключите их))

#pragma warning(disable : 4996)

// Главный счётчик для размеров всех массивов

// Столько записей мы будем создавать и сортировать

int const AllCount = 20;

// Распознаём и записываем данные из файла во внутреннй массив программы, для дальнейшей сортировки

Search\_01::AQ\* Search\_01::Processing\_data\_from\_the\_input\_file(string NameFile)

{

int const AllCount = 20;

Search\_01::AQ\* mass = new Search\_01::AQ[AllCount];

Search\_01 Search\_01;

string S[AllCount];

int n = 0;

NameFile = NameFile + ".txt";

fstream DataFile(NameFile, fstream::in);

if (DataFile.is\_open())

{

string str = "";

string nam, fam, oth;

int intMass = 0;

n = 0;

while (n < AllCount) //(!DataFile.eof())

{

getline(DataFile, S[n]);

//cout << S[n] << endl;

string input = S[n];

char r;

int y = 0;

str = "";

r = input[y];

while (r != ' ')

{

r = input[y];

if (r != ' ')

{

str = str + r;

}

y++;

}

//cout << "Имя (str) = " << str << endl;

nam = str;

str = "";

r = input[y];

while (r != ' ')

{

r = input[y];

if (r != ' ')

{

str = str + r;

}

y++;

}

//cout << "Фамилия (str) = " << str << endl;

fam = str;

str = "";

r = input[y];

while (r != ',')

{

r = input[y];

if (r != ',')

{

str = str + r;

}

y++;

}

//cout << "Отество (str) = " << str << endl;

oth = str;

str = "";

r = input[y];

while (r != ',')

{

r = input[y];

if (r != ',')

{

str = str + r;

}

y++;

}

mass[intMass].fio = nam + " " + fam + " " + oth;

//cout << "Профессия (str) = " << str << endl;

mass[intMass].d = str;

str = "";

y++;

r = input[y];

while (r != ',')

{

r = input[y];

y++;

}

y++;

r = input[y];

while (r != ' ')

{

r = input[y];

if (r != ' ')

{

str = str + r;

}

y++;

}

//cout << "Стаж (str) = " << str << endl;

char\* strng = new char[str.length() + 1];

strcpy(strng, str.c\_str());

mass[intMass].s = atoi(strng);

mass[intMass].id = intMass;

n++;

intMass++;

}

}

else

{

cout << "Ошибка! Файл с именем " << NameFile << " не найден!" << endl;

exit(000);

}

DataFile.close();

return mass;

}

// Выводит данные в консоль

void Search\_01::coutM(AQ\* mass, int n)

{

int f = 0;

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

{

if (mass[i].d != "")

{

cout << "[" << mass[i].fio << ", " << mass[i].d << ", " << mass[i].s << " лет" << "]" << endl;

//cout << "[" << mass[i].fio << ", " << mass[i].d << ", " << mass[i].s << " лет" << ", id = " << mass[i].id << "]" << endl;

f++;

}

}

if (f == 0)

{

cout << "<Поиск не дал результатов>" << endl;

}

}

// Выводит в консоль только нужную нам строку данных

void Search\_01::coutQ(AQ mass)

{

cout << "[" << mass.fio << ", " << mass.d << ", " << mass.s << " лет" << "]" << endl;

//cout << "[" << mass.fio << ", " << mass.d << ", " << mass.s << " лет" << ", id = " << mass.id << "]" << endl;

}

//// ----- Бинарный рекурсивный:

int countIteration = -1; // Количесво итераций, за которые бинарный поиск найдёт нужный элемент

// Алгоритм бинарного рекурсивного поиска

int Search\_01::binaryS(Search\_01::AQ\* massNS, int l, int r, int x)

{

countIteration++;

if (r >= l)

{

int mid = l + (r - l) / 2;

//cout << "Рассматриваем строку "; coutQ(massNS[mid]);

if (massNS[mid].s == x) return massNS[mid].id; //mid;

if (massNS[mid].s > x) return Search\_01::binaryS(massNS, l, mid - 1, x);

return Search\_01::binaryS(massNS, mid + 1, r, x);

}

return -2;

}

// Сортировка пузырьком

Search\_01::AQ\* Search\_01::BubbleSort(Search\_01::AQ\* massNS)

{

Search\_01::AQ\* mass = new Search\_01::AQ[AllCount];

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

{

mass[i] = massNS[i];

}

int n = AllCount;

// Сортируем массив по возрастанию:

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

{

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

{

if (mass[r].s > mass[r + 1].s)

{

swap(mass[r], mass[r + 1]);

}

}

}

return mass;

}

// Основная процедура запуска бинарного поиска

int Search\_01::BinRecSearch(Search\_01::AQ\* massNS, int pattStaj)

{

int n = AllCount;

Search\_01::AQ\* massSB = new Search\_01::AQ[AllCount];

massSB = massNS;

massSB = BubbleSort(massSB);

//cout << "Неотсортированный массив: " << endl;

//coutM(massNS, AllCount);

//cout << endl;

//cout << "Отсортированный массив: " << endl;

//coutM(massSB, AllCount);

//cout << endl;

countIteration = 0;

int result = binaryS(massSB, 0, n - 1, pattStaj);

PrintFromOutFile\_BinarySearch(result+1, pattStaj, massNS);

return result + 1;

}

// Печать результатов в файл

void Search\_01::PrintFromOutFile\_BinarySearch(int i, int pattStaj, Search\_01::AQ\* massNS)

{

fstream OutSortFile1("ResultBinarySearch.txt", ios::out); // Удаляем все данные из файла

(OutSortFile1).close();

fstream OutSortFile("ResultBinarySearch.txt", ios::app);

OutSortFile << "Бинарный поиск: " << endl;

OutSortFile << endl;

if (i != -1)

{

OutSortFile << "Шаблон поиска по полю «Стаж»: " << pattStaj << endl;

OutSortFile << "Строка, в которой был найден элемент: ";

i = i - 1;

OutSortFile << "[" << massNS[i].fio << ", " << massNS[i].d << ", " << massNS[i].s << " лет" << "]" << endl;

OutSortFile << endl;

i = i + 1;

OutSortFile << "Строка, в которой находится найденный элемент: " << i << endl;

OutSortFile << "Количество шагов, за которые был получен ответ: " << countIteration << endl;

}

else

{

OutSortFile << "<Поиск не дал результатов>" << endl;

}

OutSortFile << endl;

OutSortFile << "-----------------" << endl;

(OutSortFile).close();

}

// Процедура, для пользователя

int Search\_01::BinarySearch(string NameFile, int pattStaj)

{

Search\_01 Search\_01;

Search\_01::AQ\* massNS = new Search\_01::AQ[AllCount];

massNS = Search\_01.Processing\_data\_from\_the\_input\_file(NameFile);

int ret = Search\_01::BinRecSearch(massNS, pattStaj);

return(ret); // Возвращаю для нужд пользователя

}

//// ----- Линейный поиск

int Search\_01::LinearSearch(string NameFile, int pattStaj)

{

Search\_01 Search\_01;

Search\_01::AQ\* massNS = new Search\_01::AQ[AllCount];

massNS = Search\_01.Processing\_data\_from\_the\_input\_file(NameFile);

int ret = LinearSearchInsideProc(massNS, pattStaj);

PrintFromOutFile\_LinearSearch(ret, pattStaj, massNS);

return(ret); // Возвращаю для нужд пользователя

}

void Search\_01::PrintFromOutFile\_LinearSearch(int i, int pattStaj, Search\_01::AQ\* massNS)

{

fstream OutSortFile1("ResultLinearSearch.txt", ios::out); // Удаляем все данные из файла

(OutSortFile1).close();

fstream OutSortFile("ResultLinearSearch.txt", ios::app);

OutSortFile << "Линейный поиск: " << endl;

OutSortFile << endl;

if (i != -1)

{

OutSortFile << "Шаблон поиска по полю «Стаж»: " << pattStaj << endl;

OutSortFile << "Строка, в которой был найден элемент: ";

i = i - 1;

OutSortFile << "[" << massNS[i].fio << ", " << massNS[i].d << ", " << massNS[i].s << " лет" << "]" << endl;

OutSortFile << endl;

i = i + 1;

OutSortFile << "Строка, в которой находится найденный элемент: " << i << endl;

OutSortFile << "Количество шагов, за которые был получен ответ: " << i << endl;

}

else

{

OutSortFile << "<Поиск не дал результатов>" << endl;

}

OutSortFile << endl;

OutSortFile << "-----------------" << endl;

(OutSortFile).close();

}

int Search\_01::