|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 №2.1  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](#_Toc101052059)

[Описание алгоритмов поиска 3](#_Toc101052060)

[БМХ - Бойера-Мура-Хорспула 3](#_Toc101052061)

[Р-К - Рабина-Карпа 3](#_Toc101052062)

[Тесты 5](#_Toc101052063)

[Входные и выходные файлы 5](#_Toc101052064)

[Входной файл для тестов (Текстовый файл №1) 5](#_Toc101052065)

[Выходной файл, с пустыми результатами (Текстовый файл №2) 5](#_Toc101052066)

[Выходной файл, с найденными результатами (Текстовый файл №3) 6](#_Toc101052067)

[Таблица с тестами 6](#_Toc101052068)

[Текст программы 7](#_Toc101052069)

[Основной файл 7](#_Toc101052070)

[Заголовочный файл класса Search\_01 7](#_Toc101052071)

[Файл с кодом методов, для класса Search\_01 9](#_Toc101052072)

[Выводы 27](#_Toc101052073)

# Постановка задачи

Алгоритмы поиска в строке

**Входные данные:**

* Текстовый файл, каждая из строк которого содержит: ФИО, Должность, Дата приёма, стаж.
* Шаблон для поиска в поле Должность, k - количество для этого поля
* Шаблон для поиска в поле ФИО, k - количество для этого поля

**Выходные данные:** два текстовых файла, содержащие результаты поиска, по алгоритмам БМХ и РК, шаблон содержится в каждом из 2х полей, не более k раз. **Ограничения реализации:**

1. Модифицировать пакет подпрограмм для работы с массивом задания 1.3, в котором элементы представлены в виде структуры.
2. Добавить функцию поиска (на вход - элемент массива).
3. Каждый алгоритм поиска оформить в виде отдельной подпрограммы.

# Описание алгоритмов поиска

## БМХ - Бойера-Мура-Хорспула

Описание: Создаёт таблицу сдвигов для паттерна

Принимает: Паттерн

Возвращает: Таблицу сдвигов в виде динамического массива из int

**int\* BMH\_CreateTablezOfShifts(string pat);**

Описание: Ищет одновременно и по должности и по ФИО

Принимает: Указатель на массив из структуры, где хранятся все данные, паттерн и k для Должности, таблицу сдвигов шаблона для Должности, паттерн и k для ФИО, таблицу сдвигов шаблона для ФИО

Возвращает: Указатель на массив из структуры, где записаны результаты, совпадающие по 2м критериям поиска одновременно данные, из начального массива

Алгоритм работы: Сначала создаёт таблицу сдвигов для шаблона, и потом в цикле сравнивает символы с конца шаблона, и символы в строке. Если не найдёт совпадение, то смотрит на последний символ, который он проверил, смотрит на него в исходной строке, и берёт для этого символа число из таблицы сдвигов, и, далее сдвигает на это число вправо. Если такого символа нет в таблице сдвигов, то сдвигает на всю длину шаблона.

**Search\_01::AQ\* BMH\_Search\_Post\_And\_FIO(Search\_01::AQ\* massNS, string patternPost, int\* rtPost, int InkPost, string patternFio, int\* rtFio, int InkFio);**

Описание: Процедура, которая записывает выходные данные в файл

Принимает: Массив с результатами

Возвращает: True, если удалось записать в файл

**bool PrintToFileBMH(Search\_01::AQ\* mass);**

Описание: Основная процедура БМХ-поиска, со вводом и обработкой данных от пользователя

Принимает: Указатель на массив из структуры, где хранятся все данные

**void BMH(Search\_01::AQ\* massNS);**

Описание: Основная процедура БМХ-поиска. Выводит результаты в файл

Принимает: Имя входного файла, паттерн и k для Должности, паттерн и k для ФИО

**void BMH(string NameFile, string PattPost, int KPost, string PattFio, int KFio);**

## Р-К - Рабина-Карпа

Описание: Находит хеш для одного символа

Принимает: Символ

Возвращает: Его хеш, в виде int числа

**int HeshSimv(char str);**

Описание: Хеш в строке со стартовой позицией и заданной длинной

Принимает: Строку, начальную позицию, и длину хеширования

Возвращает: Хеш набора символов, в виде числа int

**int HeshSimv(string str, int indStart, int length);**

Описание: Прямой поиск для РК-поиска

Принимает: Строку, паттерн, длину паттерна, стартовую позицию

Возвращает: True, если шаблон совпал со строкой, с индекса стартовой позиции

**bool ForwardSearchToRK(string str, string pat, int length, int startPos);**

Описание: РК-поиск по должности и ФИО одновременно

Принимает: Указатель на массив из структуры, где хранятся все данные, паттерн и k для Должности, паттерн и k для ФИО

Возвращает: Указатель на массив из структуры, где записаны результаты, совпадающие по 2м критериям поиска одновременно данные, из начального массива

Алгоритм работы: Сначала создаётся хеш для шаблона, и один раз для фрагмента строки, равная длине шаблона. Потом в цикле, если исходная строка длине чем шаблон, с i-го места удаляет из хеша хеш i-1-го символа, прибавляет i + длинна шаблона-й символ, и сравнивает хеш, что у нас получился, и хеш шаблона. Если хеши совпали, то, далее сравнивает строки прямым поиском – для того, что бы удостовериться, что он не ошибся.

**Search\_01::AQ\* RK\_Search\_Post\_And\_FIO(Search\_01::AQ\* massNS, string patternPost, int InkPost, string patternFio, int InkFio);**

Описание: Процедура, которая записывает выходные данные в файл

Принимает: Массив с результатами

Возвращает: True, если удалось записать в файл

**bool PrintToFileRK(Search\_01::AQ\* mass);**

Описание: Основная процедура РК-поиска, со вводом и обработкой данных от пользователя

Принимает: Указатель на массив из структуры, где хранятся все данные

**void RK(Search\_01::AQ\* massNS);**

Описание: Основная процедура БМХ-поиска. Выводит результаты в файл

Принимает: Имя входного файла, паттерн и k для Должности, паттерн и k для ФИО

**void RK(string NameFile, string PattPost, int KPost, string PattFio, int KFio);**

# Тесты

## Входные и выходные файлы

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

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

### Выходной файл, с пустыми результатами (Текстовый файл №2)

Изображение выглядит как текст

Автоматически созданное описание

### Выходной файл, с найденными результатами (Текстовый файл №3)

Изображение выглядит как текст

Автоматически созданное описание

## Таблица с тестами

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Название тестовой ситуации | Входные данные: | | | | | Выходные данные |
| Текстовый файл | Паттерн для Должности | k для должности | Паттерн для ФИО | k для ФИО |
| Совпадение только в поле ФИО | Текстовый файл №1 | о | 1 | Зайцева | 1 | Текстовый файл №2 |
| Совпадение только в поле Должность | Текстовый файл №1 | Врач | 1 | ё | 1 | Текстовый файл №2 |
| Совпадение одновременно в 2х полях | Текстовый файл №1 | Врач | 1 | Зайцева | 1 | Текстовый файл №3 |
| k = 5 для ФИО | Текстовый файл №1 | а | 1 | а | 5 | Текстовый файл №3 |
| k > 5 для ФИО | Текстовый файл №1 | а | 1 | а | 6 | Текстовый файл №3 |
| k < 5 для ФИО | Текстовый файл №1 | а | 1 | а | 4 | Текстовый файл №2 |
| Шаблон не содержится ни в одном из полей | Текстовый файл №1 | ё | 1 | ё | 1 | Текстовый файл №2 |

# Текст программы

## Основной файл

#include <iostream>

#include <cmath>

#include <time.h>

#include <fstream>

#include "Search\_01.h"

#include <string>

# include <limits.h>

# include <string.h>

# include <stdio.h>

using namespace std;

int const AllCount = 20;

void main()

{

Search\_01 Search\_01;

setlocale(LC\_ALL, "rus");

srand(time(0));

Search\_01.BMH("OutFile", "а", 1, "а", 5);

Search\_01.RK("OutFile", "а", 1, "а", 5);

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 Data

{

int dd;

int mm;

int gg;

};

Data data;

struct Dann

{

string FIO;

string Proff;

Data Data;

int Staj;

};

Dann dann;

struct AQ // Стандартная структура нужных нам данных

{

string d; // Должность

int s; // Стаж

string fio; // ФИО

int id = 0; // id

};

bool isCreateNewDataFile = false; // Если true, то создастся новый файл данных, и уже он будет отсотрирован

// Если false - то будет использоватся имеющийся файл данных. Если его не найдётся - выведется ошибка.

// Генерирует значения, в текстовый файл, используя 4 других файла

bool mainCreateData(int AllCount);

// Загружает значения в массивы программы, из текстового файла

bool Processing\_data\_from\_the\_input\_file(Dann\* mass);

// 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);

// Загружает значения в программу (Основная процедура)

Dann\* DataCreator();

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

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

//// ----- БМХ-поиск:

// Создаём таблицу сдвигов для паттерна

int\* BMH\_CreateTablezOfShifts(string pat);

// Тут мы ищем одновременно и по должности и по ФИО

Search\_01::AQ\* BMH\_Search\_Post\_And\_FIO(Search\_01::AQ\* massNS, string patternPost, int\* rtPost, int InkPost, string patternFio, int\* rtFio, int InkFio);

// Процедура, которая записывает выходные данные в файл

bool PrintToFileBMH(Search\_01::AQ\* mass);

// Основная процедура БМХ-поиска, со вводом и обработкой данных от пользователя

void BMH(Search\_01::AQ\* massNS);

// Основная процедура БМХ-поиска, с Inside-Code вводом данных

void BMH(string NameFile, string PattPost, int KPost, string PattFio, int KFio);

////

//// ----- РК-поиск:

int HeshSimv(string str); // Хеш для строки

int HeshSimv(char str); // Для одиночных символов

int HeshSimv(string str, int indStart, int length); // Хеш в строке со стартовой позицией и заданной длинной

// Прямой поиск для РК-поиска

bool ForwardSearchToRK(string str, string pat, int length, int startPos);

// РК-поиск по должности и ФИО одновременно

Search\_01::AQ\* RK\_Search\_Post\_And\_FIO(Search\_01::AQ\* massNS, string patternPost, int InkPost, string patternFio, int InkFio);

// Вывод в файл результатов РК-поиска

bool PrintToFileRK(Search\_01::AQ\* mass);

// Основная процедура РК-поиска, со вводом и обработкой данных от пользователя

void RK(Search\_01::AQ\* massNS);

// Основная процедура РК-поиска, с Inside-Code вводом данных

void RK(string NameFile, string PattPost, int KPost, string PattFio, int KFio);

////

};

/\*

Создатель кода: Gogik Ortey (Орлов Георгий)

https://vk.com/gog.ortey

Этот код предоставляется на условиях открытого доступа.

Все, кому нужно, могут скопировать и использовать его когда и где захотят.

\*/

## 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;

// Генерируем данные текстового файла из 4х других текстовых файлов

bool Search\_01::mainCreateData(int AllCount)

{

const int size = 250; // Максимальная длинна строки (Имени или профессии)

string S[size];

string str;

int n = 0;

int r;

bool genderMan = true;

fstream OutFile("OutFile.txt", ios::out); // открываем файл для записи в конец //app - в конец файла

cout << "count = " << AllCount << endl;

int turCount = 0;

while (turCount < AllCount)

{

turCount++;

fstream Famil("Rus Famil.txt", fstream::in);

fstream ManName("Rus MenName.txt", fstream::in);

fstream WomenName("Rus FaName.txt", fstream::in);

fstream Prof("Proffesion.txt", fstream::in);

if (Famil.is\_open() && ManName.is\_open() && WomenName.is\_open() && Prof.is\_open())

{

if ((rand() % 2 + 0) == 1)

{

genderMan = false;

}

r = rand() % 20 + 0;

n = 0;

while (!Famil.eof())

{

getline(Famil, S[n]);

if (n == r)

{

if (genderMan == true) OutFile << S[n] << " ";

else OutFile << S[n] << "а" << " ";

break;

}

n++;

}

if (genderMan == true)

{

r = rand() % 25 + 0;

n = 0;

while (!ManName.eof())

{

getline(ManName, S[n]);

if (n == r)

{

OutFile << S[n] << " ";

break;

}

n++;

}

ManName.close(); WomenName.close();

fstream ManName("Rus MenName.txt", fstream::in);

fstream WomenName("Rus FaName.txt", fstream::in);

r = rand() % 25 + 0;

n = 0;

while (!ManName.eof())

{

getline(ManName, S[n]);

if (n == r)

{

OutFile << S[n] << "ович";

break;

}

n++;

}

}

else

{

r = rand() % 25 + 0;

n = 0;

while (!WomenName.eof())

{

getline(WomenName, S[n]);

if (n == r)

{

OutFile << S[n] << " ";

break;

}

n++;

}

ManName.close(); WomenName.close();

fstream ManName("Rus MenName.txt", fstream::in);

fstream WomenName("Rus FaName.txt", fstream::in);

r = rand() % 25 + 0;

n = 0;

while (!WomenName.eof())

{

getline(WomenName, S[n]);

if (n == r)

{

OutFile << S[n] << "овна";

break;

}

n++;

}

}

OutFile << ", ";

r = rand() % 25 + 0;

n = 0;

while (!Prof.eof())

{

getline(Prof, S[n]);

if (n == r)

{

OutFile << S[n];

break;

}

n++;

}

//22.08.2020

int dd, mm, gg;

dd = rand() % 29 + 1;

mm = 11; //rand() % 11 + 1;

gg = 2021; //rand() % 20 + 2000;

OutFile << ", " << dd << "." << mm << "." << gg;

int MainDat = 2022 \* 12 \* 30;

int AllDat = gg \* 12 \* 30 + mm \* 30 + dd;

int currStaj = MainDat - AllDat;

OutFile << ", " << currStaj << " лет";

OutFile << endl;

Famil.close(); ManName.close(); WomenName.close(); Prof.close();

}

else return false;

}

OutFile.close();

return true;

}

// 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)

{

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

Search\_01 Search\_01;

string S[AllCount];

int n = 0;

if (Search\_01.isCreateNewDataFile == true)

{

if (Search\_01.mainCreateData(AllCount)) // Генерируем записи

{

cout << "Данные успешно сгенерированы" << endl;

cout << endl;

}

else

{

//return false;

cout << "Ошибка!" << endl;

exit(000);

}

}

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];

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);

n++;

intMass++;

}

}

else

{

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

exit(000);

}

DataFile.close();

return mass;

}

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

bool Search\_01::Processing\_data\_from\_the\_input\_file(Dann\* mass)

{

string S[AllCount];

int n = 0;

if (isCreateNewDataFile == true)

{

if (mainCreateData(AllCount)) // Генерируем записи

{

cout << "Данные успешно сгенерированы" << endl;

cout << endl;

}

else

{

return false;

}

}

fstream DataFile("OutFile.txt", 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].Proff = str;

str = "";

r = input[y];

while (r != '.')

{

r = input[y];

if (r != '.')

{

str = str + r;

}

y++;

}

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

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

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

mass[intMass].Data.dd = atoi(strng);

str = "";

r = input[y];

while (r != '.')

{

r = input[y];

if (r != '.')

{

str = str + r;

}

y++;

}

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

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

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

mass[intMass].Data.mm = atoi(strng);

str = "";

r = input[y];

while (r != ',')

{

r = input[y];

if (r != ',')

{

str = str + r;

}

y++;

}

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

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

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

mass[intMass].Data.gg = atoi(strng);

str = "";

y++;

r = input[y];

while (r != ' ')

{

r = input[y];

if (r != ' ')

{

str = str + r;

}

y++;

}

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

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

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

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

n++;

intMass++;

}

}

else return false;

DataFile.close();

return true;

}

// Процедура для управления первыми двумя

// Возвращает NULL, если распознавание дало сбой, или что-то ещё произошло

Search\_01::Dann\* Search\_01::DataCreator()

{

setlocale(LC\_ALL, "rus");

srand(time(0));

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

Dann\* mass = new Dann[AllCount]; // Массив для храниения записей как структуру полей

if (Processing\_data\_from\_the\_input\_file(mass)) // Загружаем во внутренний массив записи из файла

{

cout << endl;

cout << "Данные успешно обработаны" << endl;

return mass;

}

else

{

cout << "Файл, необходимый для обработки записей, не открывается." << endl;

return NULL;

}

}

// 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;

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;

}

//// ----- БМХ-поиск:

// Создаём таблицу сдвигов для паттерна

int\* Search\_01::BMH\_CreateTablezOfShifts(string pat)

{

int s = 255; // Размер алфавита

int m = pat.length();

int\* rt = new int[s]; // Таблица всех символов

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

{

rt[i] = m;

}

//cout << "m-1 = " << m - 1 << endl;

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

{

// Преобразуем код символа в числовой, процедурой (int)(pat[i])

rt[(int)(pat[i]) + 127] = m - i - 1;

//cout << "Числовой код символа " << pat[i] << " = " << (int)(pat[i])+127 << endl;

}

return(rt);

}

// Тут мы ищем одновременно и по должности и по ФИО

Search\_01::AQ\* Search\_01::BMH\_Search\_Post\_And\_FIO(Search\_01::AQ\* massNS, string patternPost, int\* rtPost, int InkPost, string patternFio, int\* rtFio, int InkFio)

{

Search\_01 Search\_01;

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

int m = patternPost.length(); // Длинна шаблона

int mff = patternFio.length(); // Длинна шаблона

int CurkPost = 0;

int CurkFio = 0;

int nnn = 0;

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

{

int n = massNS[a].d.length(); // Длинна входящей последовательности

int i = 0;

while (i <= (n - m))

{

int j = m - 1;

while ((j >= 0) && (massNS[a].d[i + j] == patternPost[j]))

{

j = j - 1; // Идём справа налево по шаблону

}

if (j < 0) // Если дошли до конца шаблона, помещаем результат в ответ

{

CurkPost++;

//cout << "[Должность]: Найдено совпадение в поле " << massNS[a].d << endl;

}

char c = massNS[a].d[i + m - 1];

i = i + rtPost[(int)(c)+127]; // Шагаем дальше по последовательности, прибавляя нужное число символов

}

int nff = massNS[a].fio.length(); // Длинна входящей последовательности

int iff = 0;

while (iff <= (nff - mff))

{

int jff = mff - 1;

while ((jff >= 0) && (massNS[a].fio[iff + jff] == patternFio[jff]))

{

jff = jff - 1; // Идём справа налево по шаблону

}

if (jff < 0) // Если дошли до конца шаблона, помещаем результат в ответ

{

CurkFio++;

//cout << "[ФИО]: Найдено совпадение в поле " << massNS[a].fio << endl;

}

//cout << "i + m - 1 = " << i + m - 1 << endl;

char c = massNS[a].fio[iff + mff - 1];

iff = iff + rtFio[(int)(c)+127]; // Шагаем дальше по последовательности, прибавляя нужное число символов

}

if ((CurkPost <= InkPost) && (CurkFio <= InkFio) && (CurkPost > 0) && (CurkFio > 0))

{

massFS[nnn] = massNS[a];

nnn++;

}

CurkPost = 0;

CurkFio = 0;

}

return(massFS);

}

// Процедура, которая записывает выходные данные в файл

bool Search\_01::PrintToFileBMH(Search\_01::AQ \* mass)

{

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

(OutSortFile1).close();

int n = AllCount;

fstream OutSortFile("ResultBMHSearch.txt", ios::app); // Изменить на другие разные названия файлов

OutSortFile << "БМХ-Поиск: " << endl;

OutSortFile << "Количество записей: " << n << endl;

OutSortFile << endl;

OutSortFile << "Записи, которые есть одновременно в двух запросах: " << endl;

OutSortFile << endl;

int f = 0;

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

{

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

{

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

f++;

}

}

if (f == 0)

{

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

}

OutSortFile << endl;

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

(OutSortFile).close();

return(true);

}

// Основная процедура БМХ-поиска, со вводом и обработкой данных от пользователя

void Search\_01::BMH(Search\_01::AQ\* massNS)

{

Search\_01 Search\_01;

string patternPost = "";

string patternFio = "";

int PostK;

int FioK;

Search\_01::AQ\* massIn = new Search\_01::AQ[AllCount]; // По обоим запросам (смежный)

cout << endl;

cout << "Запускаем БМХ-поиск:" << endl;

cout << endl;

cout << "Введите шаблон для поля Должность: ";

//cin >> patternPost;

//cin.ignore(); // !!! //

getline(cin, patternPost);

int\* PostTabl = new int[AllCount];

int\* FioTabl = new int[AllCount];

PostTabl = Search\_01.BMH\_CreateTablezOfShifts(patternPost); // Создаём таблицу сдвигов для шаблона

while (true)

{

cout << "Введите значение k для поиска этого шаблона \nДанный шаблон должен найтись в поле не более k раз: ";

cin >> PostK;

if (PostK <= 0)

{

cout << "Вы ввели неверные значения для k. Попробуйте ввести их в интевале от 1 до +inf" << endl;

}

else break;

}

cout << endl;

cout << "Введите шаблон для поля ФИО: ";

cin.ignore();

getline(cin, patternFio);

FioTabl = Search\_01.BMH\_CreateTablezOfShifts(patternFio); // Создаём таблицу сдвигов для шаблона

while (true)

// Переделать

{

cout << "Введите значение k для поиска этого шаблона: ";

cin >> FioK;

if (FioK <= 0)

{

cout << "Вы ввели неверные значения для k. Попробуйте ввести их в интевале от 1 до +inf" << endl;

}

else break;

}

massIn = Search\_01.BMH\_Search\_Post\_And\_FIO(massNS, patternPost, PostTabl, PostK, patternFio, FioTabl, FioK);

cout << endl;

cout << "---------------" << endl;

cout << endl;

cout << "Общий результат поиска: " << endl;

cout << endl;

Search\_01.coutM(massIn, AllCount);

Search\_01.PrintToFileBMH(massIn);

}

// Основная процедура БМХ-поиска, с Inside-Code вводом данных

void Search\_01::BMH(string NameFile, string PattPost, int KPost, string PattFio, int KFio)

{

Search\_01 Search\_01;

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

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

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

if (KPost < 0) KPost = 0;

if (KFio < 0) KFio = 0;

int\* PostTabl = new int[AllCount];

int\* FioTabl = new int[AllCount];

PostTabl = Search\_01.BMH\_CreateTablezOfShifts(PattPost); // Создаём таблицу сдвигов для шаблона

FioTabl = Search\_01.BMH\_CreateTablezOfShifts(PattFio); // Создаём таблицу сдвигов для шаблона

massNS = Search\_01.BMH\_Search\_Post\_And\_FIO(massNS, PattPost, PostTabl, KPost, PattFio, FioTabl, KFio);

Search\_01.PrintToFileBMH(massNS);

delete[] massNS;

}

//// ----- РК-поиск:

// Хеш для строки

int Search\_01::HeshSimv(string str)

{

int r = 0;

for (int i = 0; i < str.length(); i++)

{

r += (int)str[i] + 127;

// В принципе то можно и не добавлять +127, но как показала практика,

// В С++ символы кодируются от -127 до 128

// И воспринимать неотрицательные коды визуально удобнее

}

return r;

}

// Для одиночных символов

int Search\_01::HeshSimv(char str) // Для одиночных символов

{

return (int)str + 127;

// Можно эту строчку писать и в коде, но так будет красивее

}

// Хеш в строке со стартовой позицией и заданной длинной

int Search\_01::HeshSimv(string str, int indStart, int length)

{

int r = 0;

for (int i = indStart; i <= length; i++)

{

r += (int)str[i] + 127;

}

return r;

}

// Прямой поиск для РК-поиска

bool Search\_01::ForwardSearchToRK(string str, string pat, int length, int startPos)

{

int y = 0;

//cout << "startPos = " << startPos << " length = " << length << endl;

for (int i = startPos; i <= length; i++)

{

if (str[i] != pat[y]) return false;

//cout << "str[i] = " << str[i] << " pat[y] = " << pat[y] << endl;

y++;

}

return true;

}

// РК-поиск по должности и ФИО одновременно

Search\_01::AQ\* Search\_01::RK\_Search\_Post\_And\_FIO(Search\_01::AQ\* massNS, string patternPost, int InkPost, string patternFio, int InkFio)

{

Search\_01 Search\_01;

int n = 0;

int CurkPost = 0;

int CurkFio = 0;

int lengthPatt = patternPost.length();

int heshPatt = Search\_01.HeshSimv(patternPost);

int lengthPattF = patternFio.length();

int heshPattF = Search\_01.HeshSimv(patternFio);

//cout << "heshPatt = " << heshPatt << endl;

//cout << "lengthPatt = " << lengthPatt << endl;

//cout << endl;

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

int CurHech = 0;

int CurHechF = 0;

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

{

CurHech = 0;

if (lengthPatt <= massNS[m].d.length())

// Если проверяемое слово вообще дилиннее шаблона

{

for (int i = 1; i < (massNS[m].d.length() - lengthPatt + 1); i++)

{

// Ищем от 1, потому что первый символ в поле - пробел

// Мы будем идти от 1 до того момента, пока длинна шаблона будет больше, чем осталось до конца

// Дальше шаблона мы всё равно не найдём

// На этот кусок кода, кончено, было потрачено много времени

if (i == 1)

{

CurHech = Search\_01.HeshSimv(massNS[m].d, 1, lengthPatt); ///

}

else

{

CurHech = CurHech - Search\_01.HeshSimv(massNS[m].d[i - 1]) + Search\_01.HeshSimv(massNS[m].d[i + lengthPatt - 1]);

}

if (heshPatt == CurHech)

{

if (Search\_01.ForwardSearchToRK(massNS[m].d, patternPost, lengthPatt, i) == true)

{

CurkPost++;

//cout << "[Должность]: Найдено совпадение в поле " + massNS[m].d << endl;

}

}

//cout << "i = " << i << " " << massNS[m].d[i] << " CurHech =" << CurHech << endl;

}

}

CurHechF = 0;

if (lengthPattF <= massNS[m].fio.length())

// Если проверяемое слово вообще дилиннее шаблона

{

for (int iF = 0; iF < (massNS[m].fio.length() - lengthPattF + 1); iF++)

{

if (iF == 0)

{

CurHechF = Search\_01.HeshSimv(massNS[m].fio, 0, lengthPattF - 1); ///

}

else

{

CurHechF = CurHechF - Search\_01.HeshSimv(massNS[m].fio[iF - 1]) + Search\_01.HeshSimv(massNS[m].fio[iF + lengthPattF - 1]);

}

if (heshPattF == CurHechF)

{

if (Search\_01.ForwardSearchToRK(massNS[m].fio, patternFio, lengthPattF, iF) == true)

{

CurkFio++;

//cout << "[ФИО]: Найдено совпадение в поле " + massNS[m].fio << endl;

}

}

//cout << "i = " << i << " " << massNS[m].fio[i] << " CurHech =" << CurHech << endl;

}

}

if ((CurkPost <= InkPost) && (CurkPost > 0))

{

if ((CurkFio <= InkFio) && (CurkFio > 0))

{

massFS[n] = massNS[m];

//Search\_01.coutQ(massNS[m]);

n++;

}

}

CurkPost = 0;

CurkFio = 0;

}

return(massFS);

}

// Вывод в файл результатов РК-поиска

bool Search\_01::PrintToFileRK(Search\_01::AQ\* mass)

{

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

(OutSortFile1).close();

int n = AllCount;

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

OutSortFile << "РК-Поиск: " << endl;

OutSortFile << "Количество записей: " << n << endl;

OutSortFile << endl;

OutSortFile << "Записи, которые есть одновременно в двух запросах: " << endl;

OutSortFile << endl;

int f = 0;

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

{

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

{

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

f++;

}

}

if (f == 0)

{

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

}

OutSortFile << endl;

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

(OutSortFile).close();

return(true);

}

// Основная процедура РК-поиска, со вводом и обработкой данных от пользователя

void Search\_01::RK(Search\_01::AQ\* massNS)

{

Search\_01 Search\_01;

string patternPost = "";

string patternFio = "";

int PostK;

int FioK;

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

cout << endl;

cout << "Запускаем РК-поиск:" << endl;

cout << endl;

cout << "Введите шаблон для поля Должность: ";

//cin >> patternPost;

cin.ignore();

getline(cin, patternPost);

int\* massFS = new int[AllCount];

while (true)

{

cout << "Введите значение k для поиска этого шаблона \nДанный шаблон должен найтись в поле не более k раз: ";

cin >> PostK;

if (PostK <= 0)

{

cout << "Вы ввели неверные значения для k. Попробуйте ввести их в интевале от 1 до +inf" << endl;

}

else break;

}

cout << endl;

cout << "Введите шаблон для поля ФИО: ";

//cin >> patternFio;

cin.ignore();

getline(cin, patternFio);

while (true)

{

cout << "Введите значение k для поиска этого шаблона: ";

cin >> FioK;

if (FioK <= 0)

{

cout << "Вы ввели неверные значения для k. Попробуйте ввести их в интевале от 1 до +inf" << endl;

}

else break;

}

massIn = Search\_01.RK\_Search\_Post\_And\_FIO(massNS, patternPost, PostK, patternFio, FioK);

cout << endl;

cout << "---------------" << endl;

cout << endl;

cout << "Общий результат поиска: " << endl;

cout << endl;

Search\_01.coutM(massIn, AllCount);

Search\_01.PrintToFileRK(massIn);

}

// Основная процедура РК-поиска, с Inside-Code вводом данных

void Search\_01::RK(string NameFile, string PattPost, int KPost, string PattFio, int KFio)

{

Search\_01 Search\_01;

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

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

if (KPost < 0) KPost = 0;

if (KFio < 0) KFio = 0;

massNS = Search\_01.RK\_Search\_Post\_And\_FIO(massNS, PattPost, KPost, PattFio, KFio);

Search\_01.PrintToFileRK(massNS);

delete[] massNS;

}

# Выводы

БМХ-поиск достаточно прост в реализации, и выдаёт хорошие результаты в поиске. Но он не является самым быстрым алгоритмом.

РК-поиск является одним из самых быстрых и наименее памяти затратных алгоритмов, но только если выбран правильный алгоритм хеширования, для входных данных.