Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Пермский национальный исследовательский**

**политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**О Т Ч Ё Т**

**по лабораторным работам №12,14**

Дисциплина: основы алгоритмизации и программирования

Тема: “Методы поиска: линейный, интерполяционный, прямой поиск подстроки в строке, Бойера-Мура, Кнута-Морриса-Пратта"

**Вар.15**

Выполнила:

студент группы ИВТ-20-1б

Лыкова Анна Викторовна

Проверил:

Ст. пр. кафедры ИТАС

к.т.н Яруллин Д. В.

Пермь, 2021

**Цель задачи**

1) Получить практические навыки работы с поисками;

2) Получить практические навыки работы со списками;

3) Получить практические навыки работы с меню;

Для достижения поставленной цели необходимо решить следующие задачи:

* Разработать структуру человека.
* Организовать ввод-вывод данных.
* Реализовать функцию удаления элементов из списка.
* Реализовать меню.
* Реализовать функцию отмены последнего удаления элемента.
* Разработать функцию, заполняющую массив объектов структуры случайными числами.
* Реализовать функции поиска.
* Реализовать функцию, сохраняющую измененный массив в файл.
* Разработать программу.

**Постановка задачи**

1. Создать динамический массив из записей (в соответствии с вариантом), содержащий не менее 100 элементов. Для заполнения элементов массива использовать ДСЧ.2. Предусмотреть сохранение массива в файл и загрузку массива из файла.

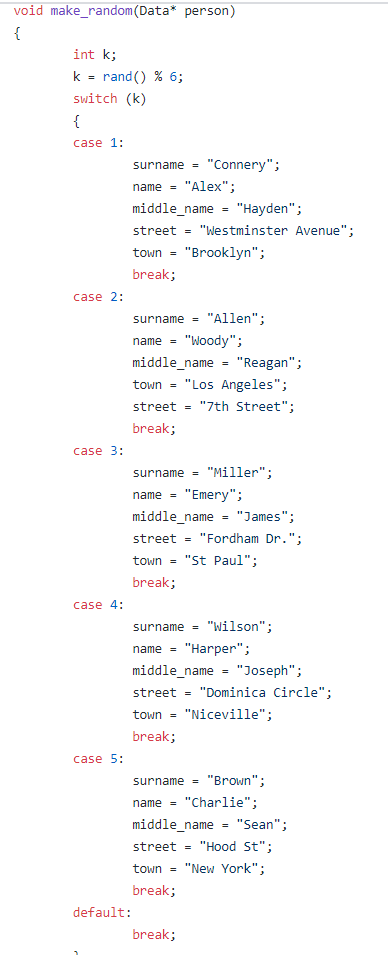
3. Предусмотреть возможность добавления и удаления элементов из массива (файла).

4. Выполнить поиск элемента в массиве по ключу в соответствии с вариантом. Для поиска использовать метод линейного поиска, метод Прямого поиска подстроки в строке и интерполяционный метод.

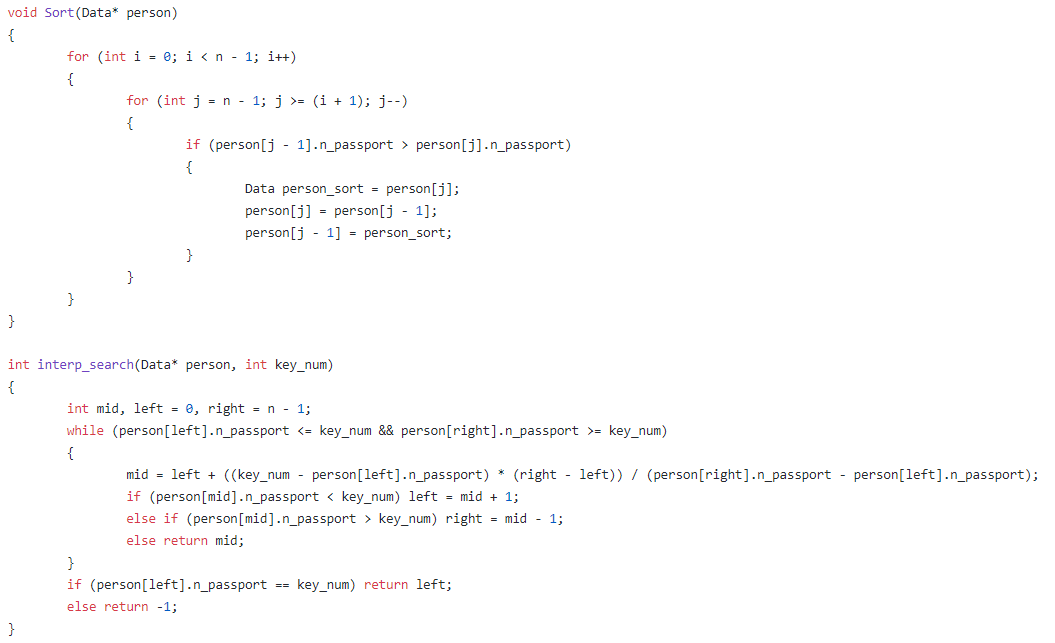


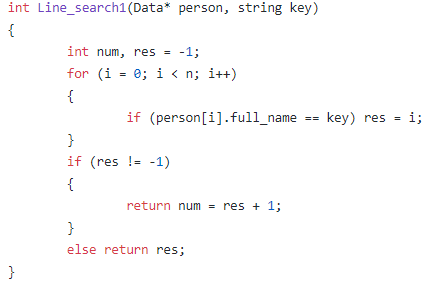
**Анализ задачи**

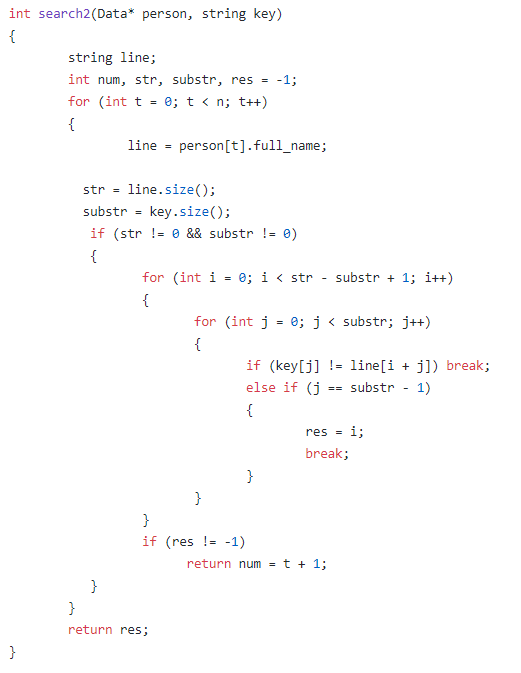
1. Для решения задачи необходимо:
   1. Реализовать функцию получения корректного размера массива от пользователя.
   2. Разработать функцию меню.
   3. Разработать функцию, заполняющую массив с помощью ДСЧ.
   4. Реализовать функцию удаления элемента из массива.
   5. Разработать функции поиска по ключу.
   6. Реализовать структуру Data с полями full\_name, DOB, address, n\_passport.
   7. Разработать функцию записи массива в файл.
   8. Реализовать функцию сортировки массива для Интерполяционного поиска.
   9. Разработать функцию переворота строки для поиска Бойера-Мура.
   10. Разработать префикс-функцию для метода КМП.
2. В ходе работы были использованы следующие типы данных:
   1. Поля структуры Data типа string, поле n\_passport типа int.
   2. Для записи массива в файл используется класс ofstream.
   3. Функции поиска принимают в качестве параметров массив и ключ.
3. Для решения задачи данные были представлены в следующем виде:
   1. Для хранения информации о человеке используется структура Data с 3-4 полями (см.п.1.6 и п.2.1).
4. Структуры, которые использовались при решении задачи:
   1. Для заполнения полей структуры Data была создана функция make\_random(), для заполнения поля full\_name используется 3 алфавита (имен, фамилий, отчеств). Номер паспорта - случайное число от 1 до 6000. Аналогично, заполняются поле DOB и address.



1. Для операций ввода и вывода использовались следующие операторы и функции:
   1. cin и cout.
   2. Getline, для ввода ключа (ФИО пользователя).
2. Поставленные задачи будут решены следующими действиями:
   1. Функция Interpol\_search преобразует массив объектов структуры в массив дат, приведенных к восьмизначному числу. Ключ преобразуется по такому же принципу. Для сортировки массива используется сортировка Шелла. С помощью цикла while, который выполняется до тех пор, пока значение левой границы меньше равно ключу и значение правой границы больше равно ключу. Внутри цикла рассчитывается индекс серединного элемента. Если значение серединного элемента меньше ключа, то левой границе присваивается инкрементированный индекс середины. Если же значение серединного элемента больше ключа, то правой границе присваивается декрементированный индекс середины. Если же значение mid совпадает с ключом, то функция возвращает true, в случае если элемента в массиве нет, то функция возвращает значение false.



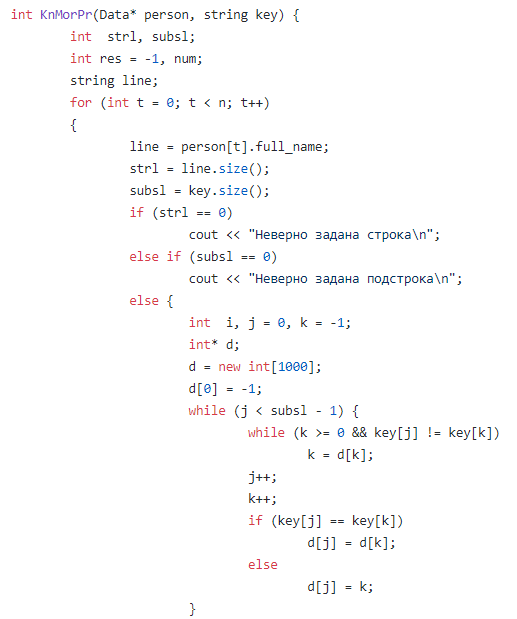
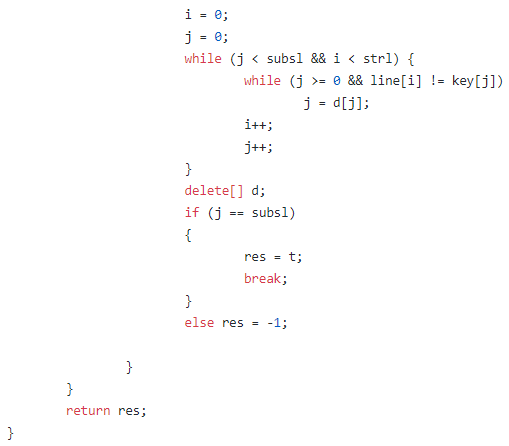
* 1. Функция Line\_search преобразует ключ и массив по тому же принципу что и функция Interpol\_search (см.п.6.1). С помощью цикла for осуществляется проход по всем элементам массива, в случае удовлетворения элемента массива ключу, функция возвращает res = i + 1, иначе res =-1.
  2. Функция search2 преобразует массив объектов структуры в строку фио, ключ так же преобразовывается в строку. Проход осуществляется с помощью цикла for, который выполняется до тех пор, пока индекс не станет равен длине строки без подстроки. Внутри цикла for используется цикл while, который считает количество совпавших символов, если количество совпавших символов равно длине подстроки, то возвращается res = i + 1, иначе res =-1.



* 1. Функция Boyer\_Moor.



* 1. Функция KnMorPr аналогично пунктам 6.4 и 6.3 осуществляет преобразование ключа и массива в строки.



**Код программы**

**Лаб. 12:**

#include <iostream>

#include <windows.h>

#include <fstream>

#include <string>

#include <ctime>

using namespace std;

string key;

string surname, name, middle\_name, street, town;

int n, i, j;

struct Data

{

string full\_name;

string DOB;

string address;

};

void make\_random(Data\* person)

{

int k;

k = rand() % 6;

switch (k)

{

case 1:

surname = "Connery";

name = "Alex";

middle\_name = "Hayden";

street = "Westminster Avenue";

town = "Brooklyn";

break;

case 2:

surname = "Allen";

name = "Woody";

middle\_name = "Reagan";

town = "Los Angeles";

street = "7th Street";

break;

case 3:

surname = "Miller";

name = "Emery";

middle\_name = "James";

street = "Fordham Dr.";

town = "St Paul";

break;

case 4:

surname = "Wilson";

name = "Harper";

middle\_name = "Joseph";

street = "Dominica Circle";

town = "Niceville";

break;

case 5:

surname = "Brown";

name = "Charlie";

middle\_name = "Sean";

street = "Hood St";

town = "New York";

break;

default:

break;

}

}

void to\_form(Data\* person)

{

int a = 0, d, m, y;

string surname1, name1, middle\_name1, street1, town1;

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

make\_random(person);

surname1 = surname;

make\_random(person);

name1 = name;

make\_random(person);

middle\_name1 = middle\_name;

person[i].full\_name = surname1 + " " + name1 + " " + middle\_name1;

d = 1 + rand() % 31;

m = 1 + rand() % 12;

y = rand() % 122 + 1950;

auto day = to\_string(d);

auto month = to\_string(m);

auto year = to\_string(y);

person[i].DOB = month + "." + day + "." + year;

a = rand() % 1000;

auto Apt = to\_string(a);

make\_random(person);

street1 = street;

make\_random(person);

town1 = town;

person[i].address = Apt + " " + street1 + ", " + town1;

}

}

void print\_to\_file(Data\* person)

{

ofstream fout;

fout.open("Data+.txt");

if (!fout.is\_open())

{

perror("Ошибка при открытии файла");

exit(0);

}

else {

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

fout << i + 1 << endl;

fout << person[i].full\_name << endl;

fout << person[i].DOB << endl;

fout << person[i].address << endl << endl;

}

}

fout.close();

}

int Line\_search1(Data\* person, string key)

{

int num, res = -1;

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

{

if (person[i].full\_name == key) res = i;

}

if (res != -1)

{

return num = res + 1;

}

else return res;

}

int search2(Data\* person, string key)

{

string line;

int num, str, substr, res = -1;

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

{

line = person[t].full\_name;

str = line.size();

substr = key.size();

if (str != 0 && substr != 0)

{

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

{

for (int j = 0; j < substr; j++)

{

if (key[j] != line[i + j]) break;

else if (j == substr - 1)

{

res = i;

break;

}

}

}

if (res != -1)

return num = t + 1;

}

}

return res;

}

void del(Data\* person, int num)

{

Data\* person2 = new Data[n];

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

person2[i] = person[i];

}

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

person2[i] = person[i + 1];

}

n--;

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

{

person[i] = person2[i];

}

print\_to\_file(person);

}

void add(Data\* person, int num)

{

Data\* person2 = new Data[n];

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

{

person2[i] = person[i];

}

cin.get();

getline(cin, person[num].full\_name);

getline(cin, person[num].DOB);

getline(cin, person[num].address);

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

{

person2[i+1] = person[i];

}

n++;

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

//{

// person[i] = person2[i];

//}

print\_to\_file(person2);

}

void add\_del(Data\* person, bool click, int num)

{

cout << endl;

cout << "\tВыберите действие: " << endl;

cout << "0 - Удалить найденный эелемент в списке" << endl;

cout << "1 - Добавить элемент в список после найденного" << endl;

cin >> click;

if (!click) del(person, num);

else add(person, num);

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int click, close = 0;

bool click2 = 0;

srand(time(NULL));

n = rand() % 1000 - 500;

cout << "\tЗаписей в файле: - " << n;

while (true)

{

if (n > 0)

break;

else {

cout << endl;

cout << "Файл пуст!" << endl;

cout << endl;

cout << "\tЗавершить работу программы? - ";

cout << "1.Да/2.Нет" << endl;

cin >> click;

switch (click)

{

case 1:

{

cout << endl;

cout << "Программа завершила свою работу!";

cout << endl;

close = 1;

break;

}

case 2:

{

n = rand() % 1000 - 500;

cout << "\tЗаписей в файле: - " << n;

break;

}

default:

break;

}

break;

}

}

Data\* person1 = new Data[n];

if (close == 0) {

do {

cout << endl;

cout << "\tЧто вы хотите сделать?" << endl;

cout << "1.Сформировать файл" << endl;

cout << "2.Открыть готовый файл" << endl;

cout << endl;

cin >> click;

switch (click)

{

case 1:

{

to\_form(person1);

print\_to\_file(person1);

break;

}

case 2:

{

int count = 0;

string row;

ifstream fout;

fout.open("Data+.txt");

while (!fout.eof())

{

count++;

getline(fout, row);

}

n = count / 4;

person1 = new Data[n];

fout.close();

fout.open("Data+.txt");

int i = 0;

while (!fout.eof())

{

getline(fout, row);

getline(fout, person1[i].full\_name);

getline(fout, person1[i].DOB);

getline(fout, person1[i].address);

getline(fout, row);

i++;

}

fout.close();

cout << "Список загружен." << endl << endl;

break;

}

default:

break;

}

cout << endl;

cout << "\tВыберите метод поиска: " << endl;

cout << "1.Линейный поиск в файле" << endl;

cout << "2.Прямой поиск подстроки в строке в файле" << endl;

cout << "3.Выход" << endl;

cout << endl;

cin >> click;

switch (click) {

case 1:

{

cin.get();

cout << "Введите ФИО человека, данные о котором хотите найти: ";

getline(cin, key);

int elem = Line\_search1(person1, key);

if (elem != -1)

{

cout << "Элемент в списке найден. Он находится под номером " << elem << endl;

add\_del(person1, click2, elem);

}

else

cout << "Элемента в списке нет" << endl;

break;

}

case 2:

{

cin.get();

cout << "Введите фамилию человека, данные о котором хотите найти: ";

getline(cin, key);

int elem = search2(person1, key);

if (elem != -1)

{

cout << "Элемент в списке найден. Он находится под номером " << elem << endl;

add\_del(person1, click2, elem);

}

else

cout << "Элемента в списке нет" << endl;

break;

}

case 3:

{

cout << endl;

cout << "Программа завершила свою работу!";

cout << endl;

close = 1;

break;

}

default:

break;

}

} while (close != 1);

}

return 0;

}

**Лаб. 14**

#include <iostream>

#include <windows.h>

#include <fstream>

#include <string>

#include <ctime>

using namespace std;

string key;

string surname, name, middle\_name, street, town;

int n, i, j;

struct Data

{

string full\_name;

string DOB;

string address;

int n\_passport;

};

void make\_random(Data\* person)

{

int k;

k = rand() % 6;

switch (k)

{

case 1:

surname = "Connery";

name = "Alex";

middle\_name = "Hayden";

street = "Westminster Avenue";

town = "Brooklyn";

break;

case 2:

surname = "Allen";

name = "Woody";

middle\_name = "Reagan";

town = "Los Angeles";

street = "7th Street";

break;

case 3:

surname = "Miller";

name = "Emery";

middle\_name = "James";

street = "Fordham Dr.";

town = "St Paul";

break;

case 4:

surname = "Wilson";

name = "Harper";

middle\_name = "Joseph";

street = "Dominica Circle";

town = "Niceville";

break;

case 5:

surname = "Brown";

name = "Charlie";

middle\_name = "Sean";

street = "Hood St";

town = "New York";

break;

default:

break;

}

}

void to\_form(Data\* person)

{

int a = 0, d, m, y;

string surname1, name1, middle\_name1, street1, town1;

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

make\_random(person);

surname1 = surname;

make\_random(person);

name1 = name;

make\_random(person);

middle\_name1 = middle\_name;

person[i].full\_name = surname1 + " " + name1 + " " + middle\_name1;

d = 1 + rand() % 31;

m = 1 + rand() % 12;

y = rand() % 122 + 1950;

auto day = to\_string(d);

auto month = to\_string(m);

auto year = to\_string(y);

person[i].DOB = month + "." + day + "." + year;

a = rand() % 1000;

auto Apt = to\_string(a);

make\_random(person);

street1 = street;

make\_random(person);

town1 = town;

person[i].address = Apt + " " + street1 + ", " + town1;

person[i].n\_passport = 10000 + rand() % 70001;

}

}

void print\_to\_file(Data\* person)

{

ofstream fout;

fout.open("Data+.txt");

if (!fout.is\_open())

{

perror("Ошибка при открытии файла");

exit(0);

}

else {

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

fout << i + 1 << endl;

fout << person[i].full\_name << endl;

fout << person[i].DOB << endl;

fout << person[i].address << endl;

fout << person[i].n\_passport << endl << endl;

}

}

fout.close();

}

int BoyerM(Data\* person, string key)

{

string line;

int stl, subsl;

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

{

line = person[t].full\_name;

stl = line.size();

subsl = key.size();

if (stl != 0 && subsl != 0 && stl >= subsl)

{

int i, p;

int b[256];

for (i = 0; i < 256; i++)

{

b[i] = subsl;

}

for (i = subsl - 2; i >= 0; i--)

{

if (b[int(unsigned char(key[i]))] == subsl)

{

b[int(unsigned char(key[i]))] = subsl - i - 1;

}

}

p = subsl - 1;

while (p < stl)

{

if (key[subsl - 1] != line[p])

{

p += b[int((unsigned char)line[p])];

}

else

{

for (i = subsl - 1; i >= 0; i--)

{

if (key[i] != line[p - subsl + i + 1])

{

p += b[int((unsigned char)line[p])];

break;

}

else if (i == 0)

{

return t;

}

}

}

}

}

}

return -1;

}

int KnMorPr(Data\* person, string key) {

int strl, subsl;

int res = -1, num;

string line;

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

{

line = person[t].full\_name;

strl = line.size();

subsl = key.size();

if (strl == 0)

cout << "Неверно задана строка\n";

else if (subsl == 0)

cout << "Неверно задана подстрока\n";

else {

int i, j = 0, k = -1;

int\* d;

d = new int[1000];

d[0] = -1;

while (j < subsl - 1) {

while (k >= 0 && key[j] != key[k])

k = d[k];

j++;

k++;

if (key[j] == key[k])

d[j] = d[k];

else

d[j] = k;

}

i = 0;

j = 0;

while (j < subsl && i < strl) {

while (j >= 0 && line[i] != key[j])

j = d[j];

i++;

j++;

}

delete[] d;

if (j == subsl)

{

res = t;

break;

}

else res = -1;

}

}

return res;

}

void Sort(Data\* person)

{

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

{

for (int j = n - 1; j >= (i + 1); j--)

{

if (person[j - 1].n\_passport > person[j].n\_passport)

{

Data person\_sort = person[j];

person[j] = person[j - 1];

person[j - 1] = person\_sort;

}

}

}

}

int interp\_search(Data\* person, int key\_num)

{

int mid, left = 0, right = n - 1;

while (person[left].n\_passport <= key\_num && person[right].n\_passport >= key\_num)

{

mid = left + ((key\_num - person[left].n\_passport) \* (right - left)) / (person[right].n\_passport - person[left].n\_passport);

if (person[mid].n\_passport < key\_num) left = mid + 1;

else if (person[mid].n\_passport > key\_num) right = mid - 1;

else return mid;

}

if (person[left].n\_passport == key\_num) return left;

else return -1;

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int click, close = 0;

srand(time(NULL));

n = rand() % 1000 - 500;

cout << "\tЗаписей в файле: - " << n;

Data\* person1 = new Data[n];

while (true)

{

if (n > 0)

break;

else {

cout << endl;

cout << "Файл пуст!" << endl;

cout << endl;

cout << "\tЗавершить работу программы? - ";

cout << "1.Да/2.Нет" << endl;

cin >> click;

switch (click)

{

case 1:

{

cout << endl;

cout << "Программа завершила свою работу!";

cout << endl;

close = 1;

break;

}

case 2:

{

n = rand() % 1000 - 500;

cout << "\tЗаписей в файле: - " << n;

break;

}

default:

break;

}

break;

}

}

if (close == 0) {

do {

cout << endl;

cout << "\tЧто вы хотите сделать?" << endl;

cout << "1.Сформировать файл" << endl;

cout << "2.Открыть готовый файл" << endl;

cout << endl;

cin >> click;

switch (click)

{

case 1:

{

to\_form(person1);

print\_to\_file(person1);

break;

}

case 2:

{

int count = 0;

string row;

ifstream fout;

fout.open("Data+.txt");

while (!fout.eof())

{

count++;

getline(fout, row);

}

n = count / 4;

person1 = new Data[n];

fout.close();

fout.open("Data+.txt");

int i = 0;

while (!fout.eof())

{

getline(fout, row);

getline(fout, person1[i].full\_name);

getline(fout, person1[i].DOB);

getline(fout, person1[i].address);

fout >> person1[i].n\_passport;

getline(fout, row);

i++;

}

fout.close();

cout << "Список загружен." << endl << endl;

break;

}

default:

break;

}

cout << endl;

cout << "\tВыберите метод поиска: " << endl;

cout << "1.Метод поиска Бойера-Мура" << endl;

cout << "2.Метод поиска Кнута-Морриса-Пратта" << endl;

cout << "3.Интерполяционный поиск в файле" << endl;

cout << "4.Выход" << endl;

cout << endl;

cin >> click;

switch (click)

{

case 1:

{

cin.get();

cout << "Введите фамилию или имя человека, данные о котором хотите найти: ";

getline(cin, key);

int elem = BoyerM(person1, key);

if (elem != -1)

cout << "Элемент найден. Номер элемента в списке: " << elem + 1 << endl;

else cout << "Элемент не найден" << endl;

break;

}

case 2:

{

cin.get();

cout << "Введите фамилию или имя человека, данные о котором хотите найти: ";

getline(cin, key);

int elem = KnMorPr(person1, key);

if (elem != -1)

cout << "Элемент найден. Номер элемента в списке: " << elem + 1 << endl;

else cout << "Элемент не найден" << endl;

break;

}

case 3:

{

int key\_num;

cout << endl;

cout << "Введите номер паспорта человека, данные о котором хотите найти: ";

cin >> key\_num;

Sort(person1);

print\_to\_file(person1);

int elem = interp\_search(person1, key\_num);

if (elem != -1)

cout << "Элемент найден. Номер элемента в отсортированном списке: " << elem + 1 << endl;

else

cout << "Элемент с введенным номером не найден" << endl;

break;

}

case 4:

cout << endl;

cout << "Программа завершила свою работу!";

cout << endl;

close = 1;

break;

default:

break;

}

} while (close != 1);

}

return 0;

}