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 лабораторной работе №12**

Дисциплина: основы теории алгоритмов и структуры данных

Тема: Алгоритмы поиска. Работа со списками. Метод интерполяционного поиска

Вариант: 20

Выполнил работу

студент группы РИС-20-1б

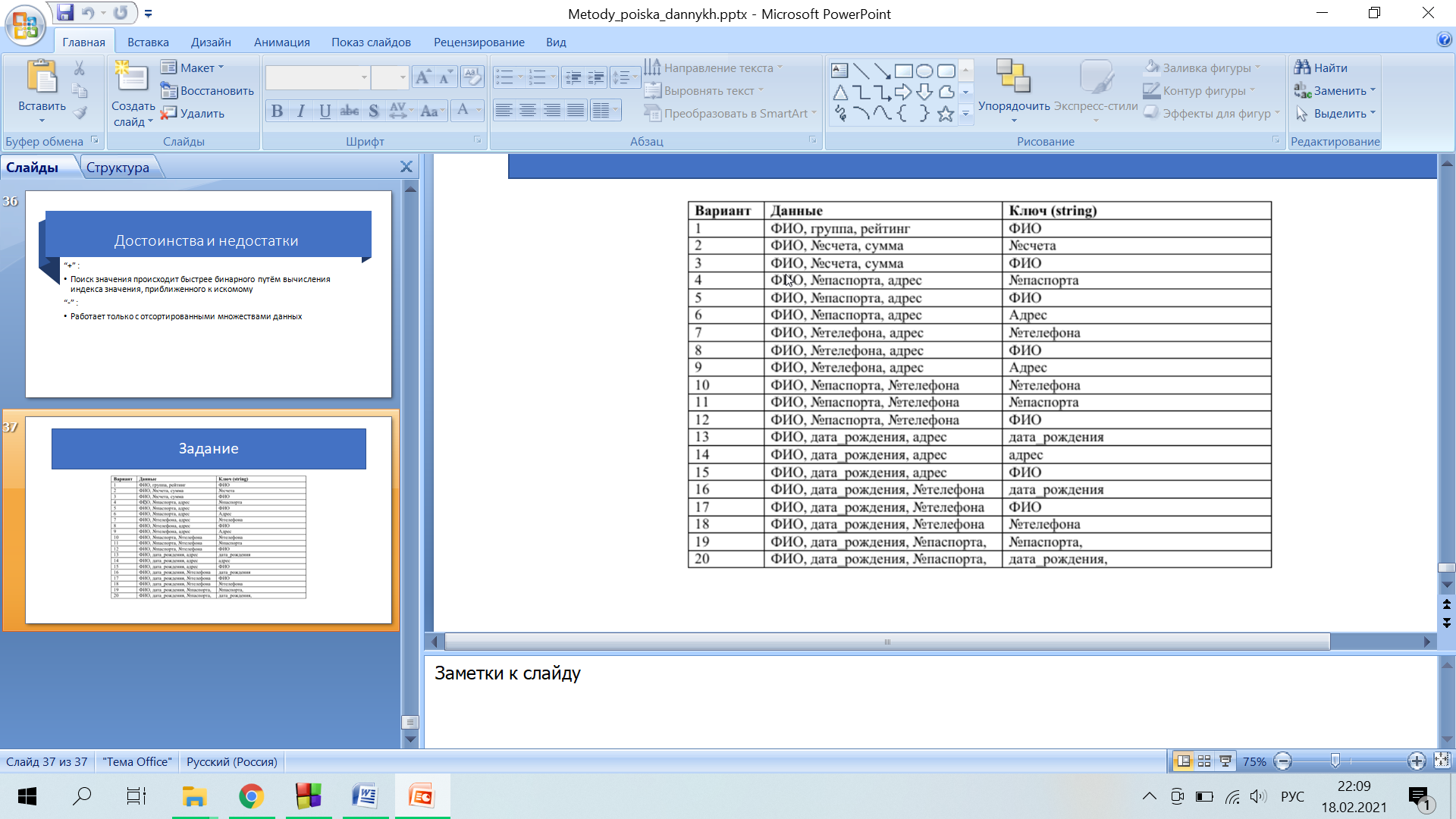
Рябов Никита Андреевич

Проверила

Доцент кафедры ИТАС Полякова О.А.

Пермь, 2021

**Постановка задачи**



**Анализ задачи**

*Какие предстоит выполнить действия.*

Считываем длину массива и считываем массив соответствующей длины

while (len<2 || len>100)

{

cout << "Введите размер таблицы\n";

cin >> len;

}

cin.get();

CITIZEN \*data\_base=new CITIZEN[len];

ifstream F1("F1.txt");

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

{

getline(F1,data\_base[i].fio);

getline(F1,data\_base[i].birthday);

getline(F1,data\_base[i].passport);

}

Сортируем его

sort(data\_base,data\_base+len,comp);

Запрашиваем у пользователя ключ поиска и получаем ответ от функций

cout << "Введите запрос\n";

cin >> user\_search;

int pos1=Linear(data\_base,user\_search);

int pos2=Binary(data\_base,user\_search);

int pos3=Interpol(data\_base,user\_search);

Вывод результата

if (pos1==-1 || pos2==-1 || pos3==-1) cout << "Элемент не найден\n";

else

{

cout << "Результат линейного поиска:\n";

Print(data\_base[pos1]);

cout << "Результат бинарного поиска:\n";

Print(data\_base[pos2]);

cout << "Результат интерполяционного поиска:\n";

Print(data\_base[pos3]);

cout << endl;

}

Функция для превращения строковой даты рождения в число

int Dig(string a)

{

return (int)(a[6]+a[7]+a[3]+a[4]+a[0]+a[1]);

}

Функция для сортировки

bool comp(CITIZEN a, CITIZEN b)

{

return (Dig(a.birthday)<Dig(b.birthday));

}

Функция, печатающая элемент структуры

void Print(CITIZEN a)

{

cout << "ФИО : " << a.fio << endl;

cout << "Дата рождения : " << a.birthday << endl;

cout << "Номер пасспорта: " << a.passport << endl;

}

Просто пробегает по массиву до первого совпадения. Если таких нет – выводит -1

int Linear(CITIZEN arr[], string x)

{

for (int i=0;i<len;i++) if (arr[i].birthday==x) return i;

return -1;

}

Несмотря на название, функция реализует прямой поиск подстроки в строке. Последовательно проверяет все возможные варианты расположения шаблона в строке.

int Binary(CITIZEN arr[], string x)

{

string mas="";

bool flag;

for (int i=0;i<len;i++) mas+=arr[i].birthday;

for (int i=0;i<len\*8;i++)

{

flag=true;

for (int k=0;k<8;k++)

{

if (x[k]!=mas[i+k])

{

flag=false;

break;

}

}

if (flag) return i/8;

}

return -1;

}

Функция интерполяционного поиска. Устанавливает границы поиска и сужает их с каждой итерацией. В зависимости от положения искомого элемента, по специальной формуле.

int Interpol(CITIZEN arr[], string x)

{

int L=0;

int R=len-1;

while (R-L>1)

{

int mid=L+(Dig(x)-Dig(arr[L].birthday))\*(R-L)/(Dig(arr[R].birthday)-Dig(arr[L].birthday));

if (Dig(x)<Dig(arr[mid].birthday)) R=mid;

else if (Dig(x)>Dig(arr[mid].birthday)) L=mid;

else return mid;

}

if (Dig(x)==Dig(arr[L].birthday)) return L;

if (Dig(x)==Dig(arr[R].birthday)) return R;

return -1;

}

*С каким типом данных действуем*

Динамический массив структур

*В каком виде представлены данные*

Строки, содержащие любые символы

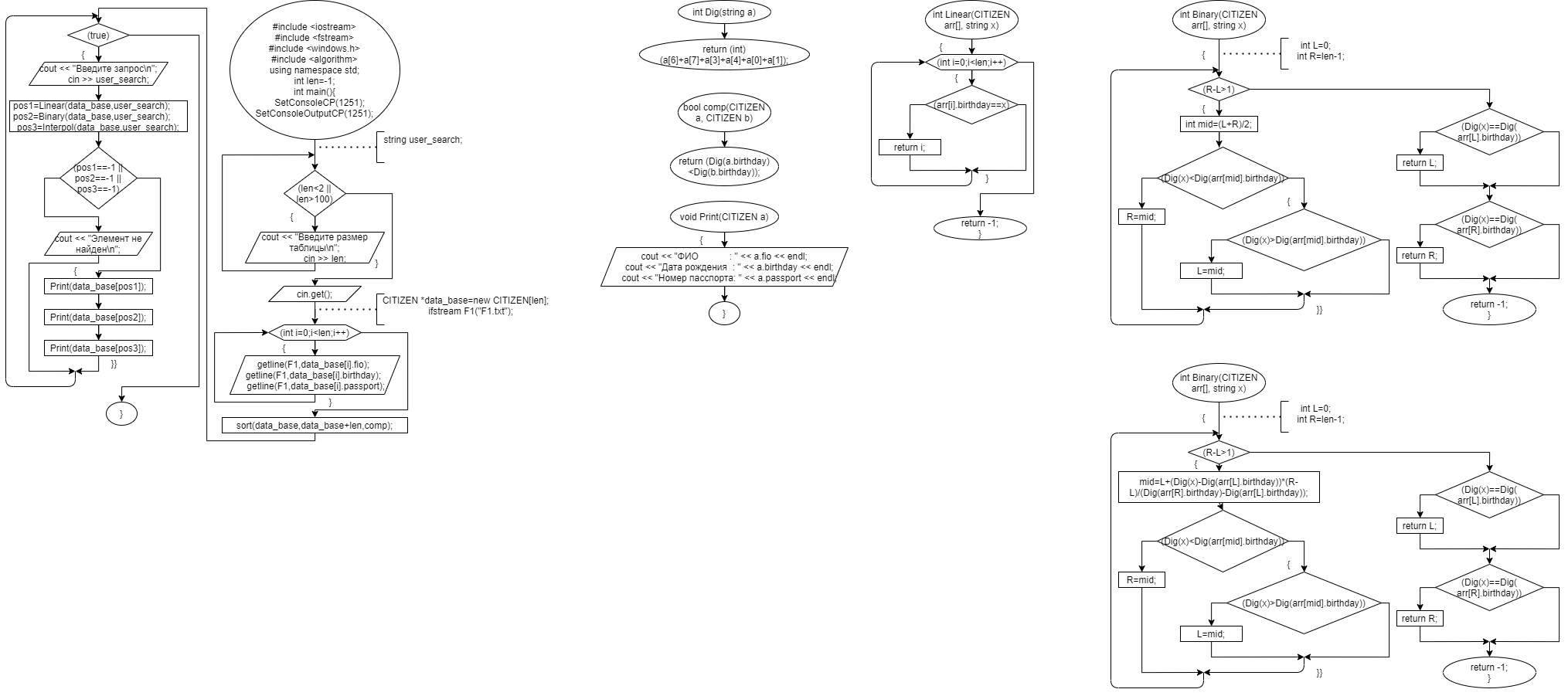
*Какие поля*

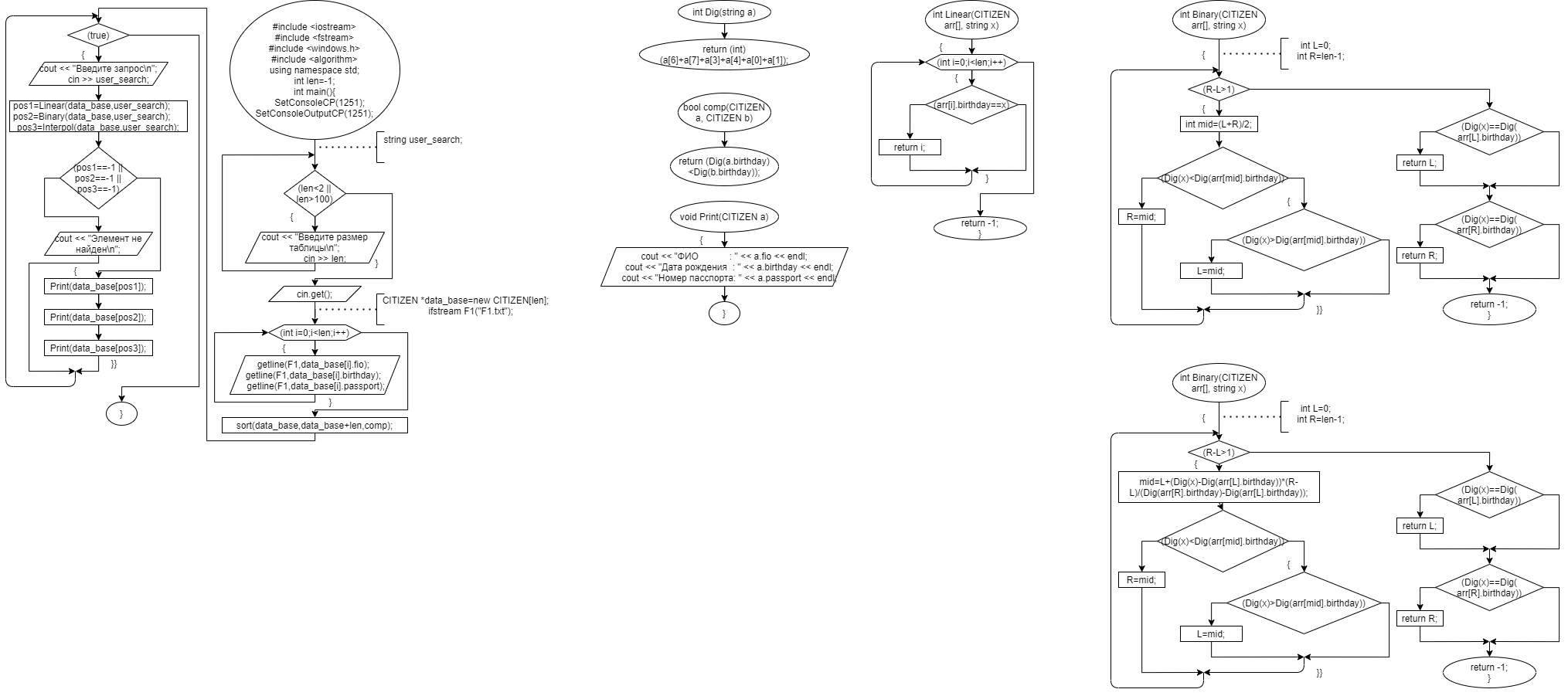
Структура содержит поле string для ФИО, поле string для даты рождения и поле string для номера пасспорта

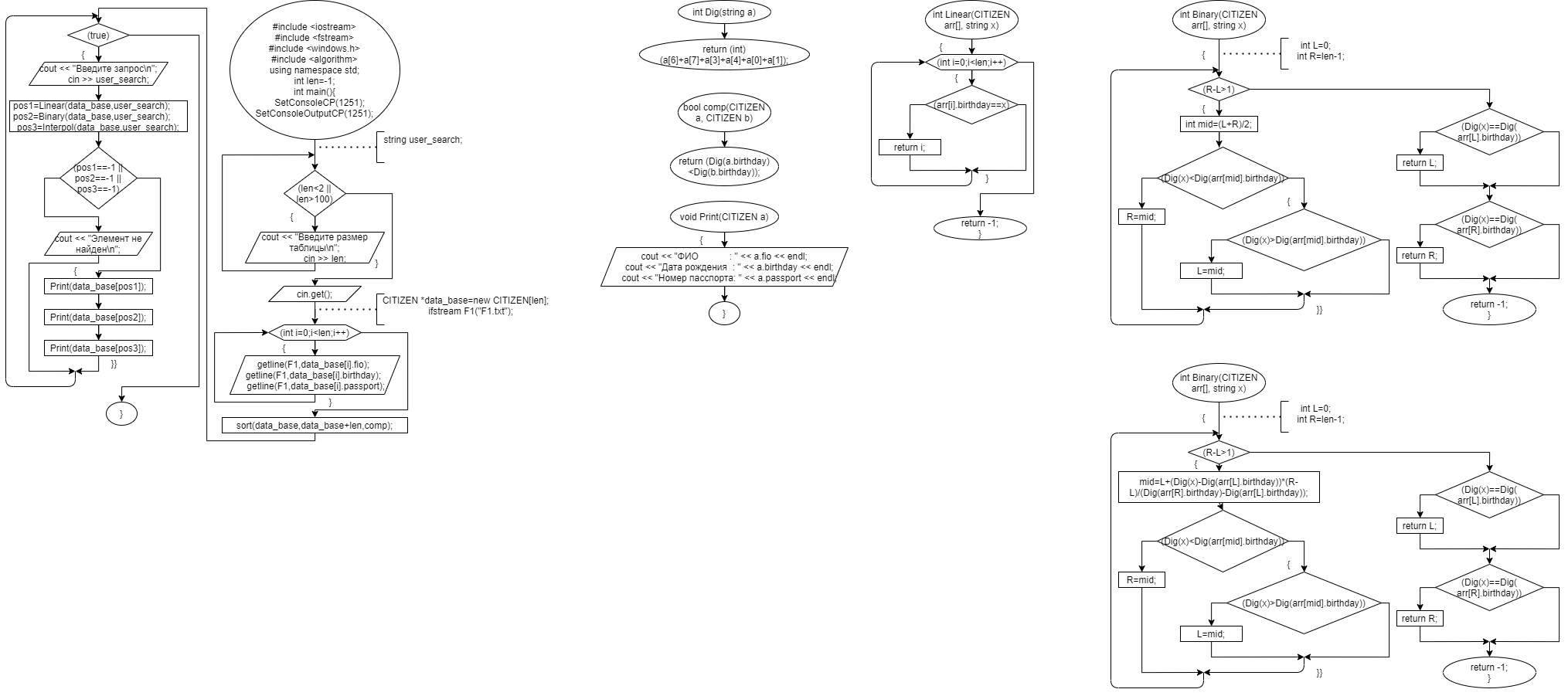
*Операторы ввода/вывода*

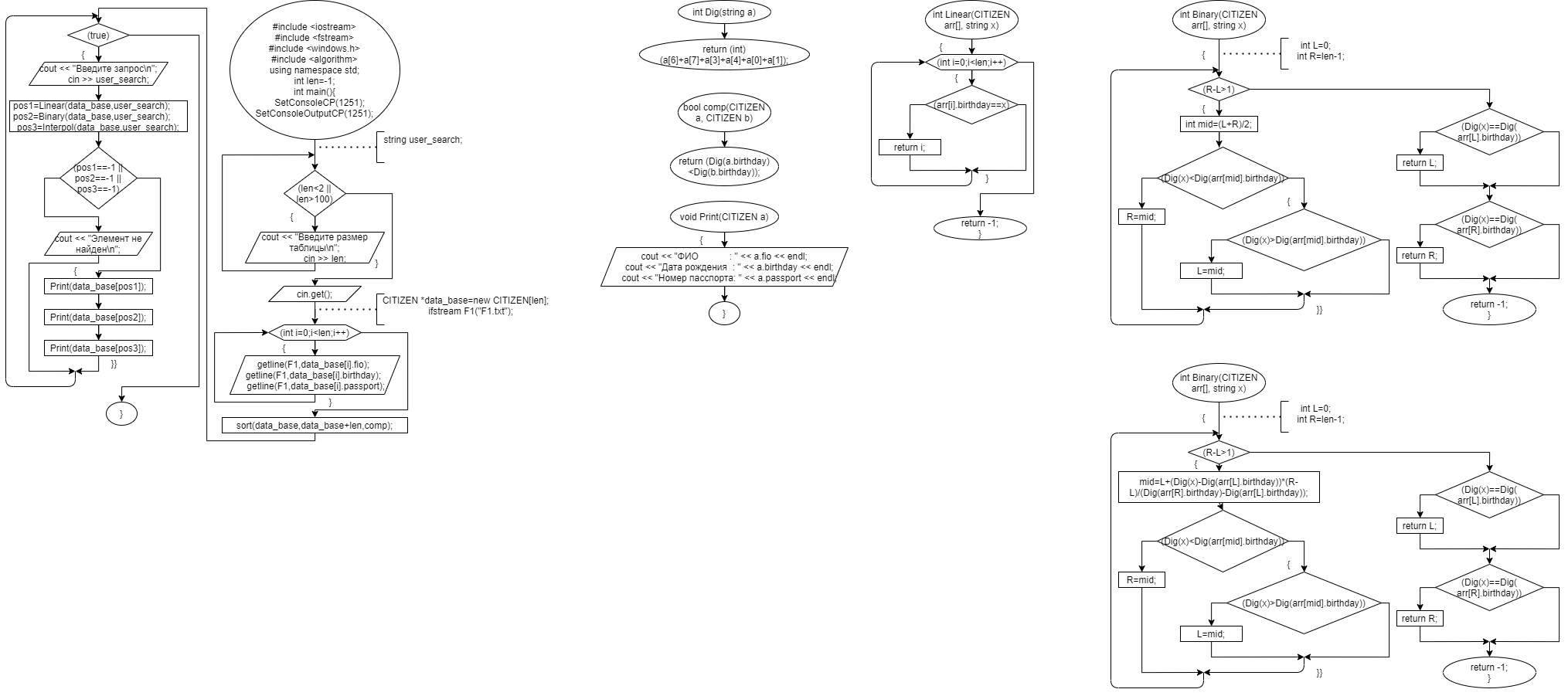
В основном используются операторы cin/cout. Исключение составляет ввод строк. Они могут содержать пробелы, и к ним применяется метод getline().

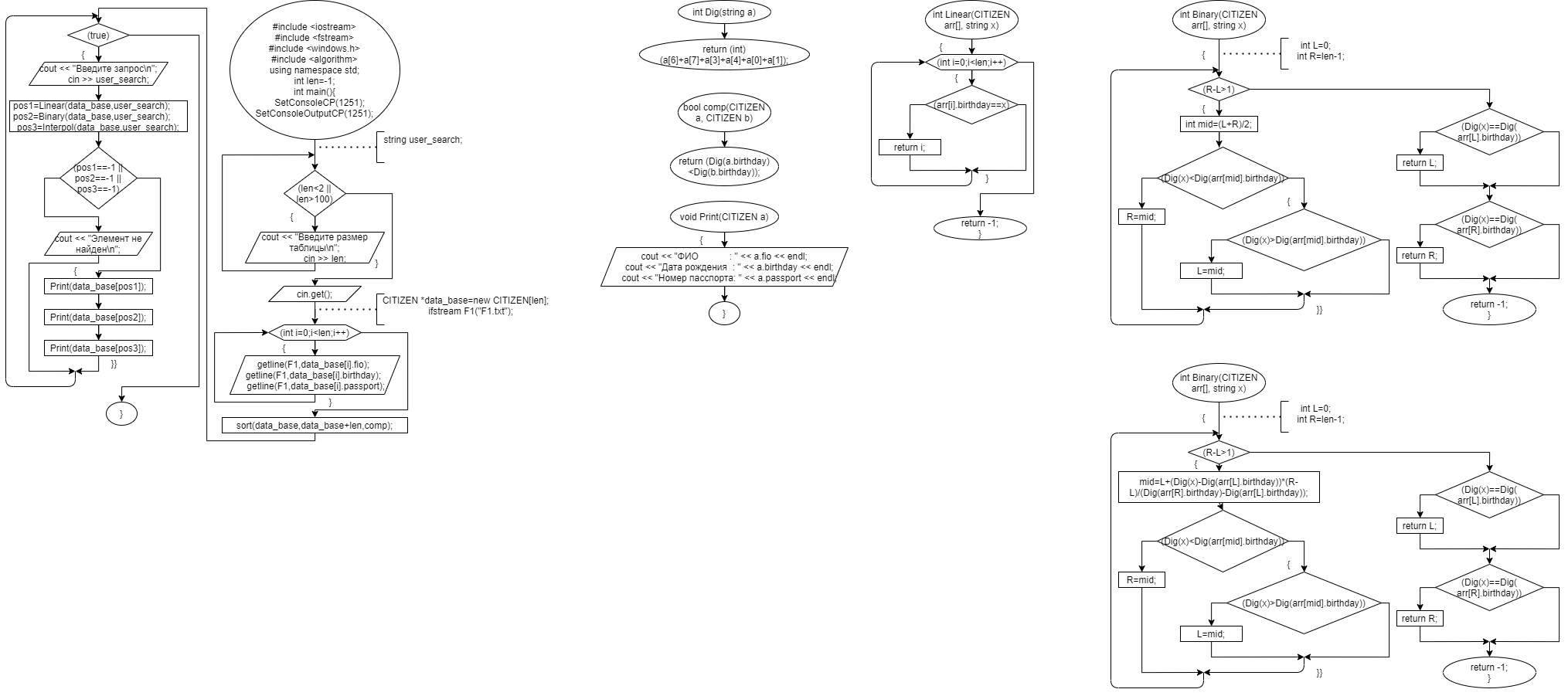
**Блок-схема**

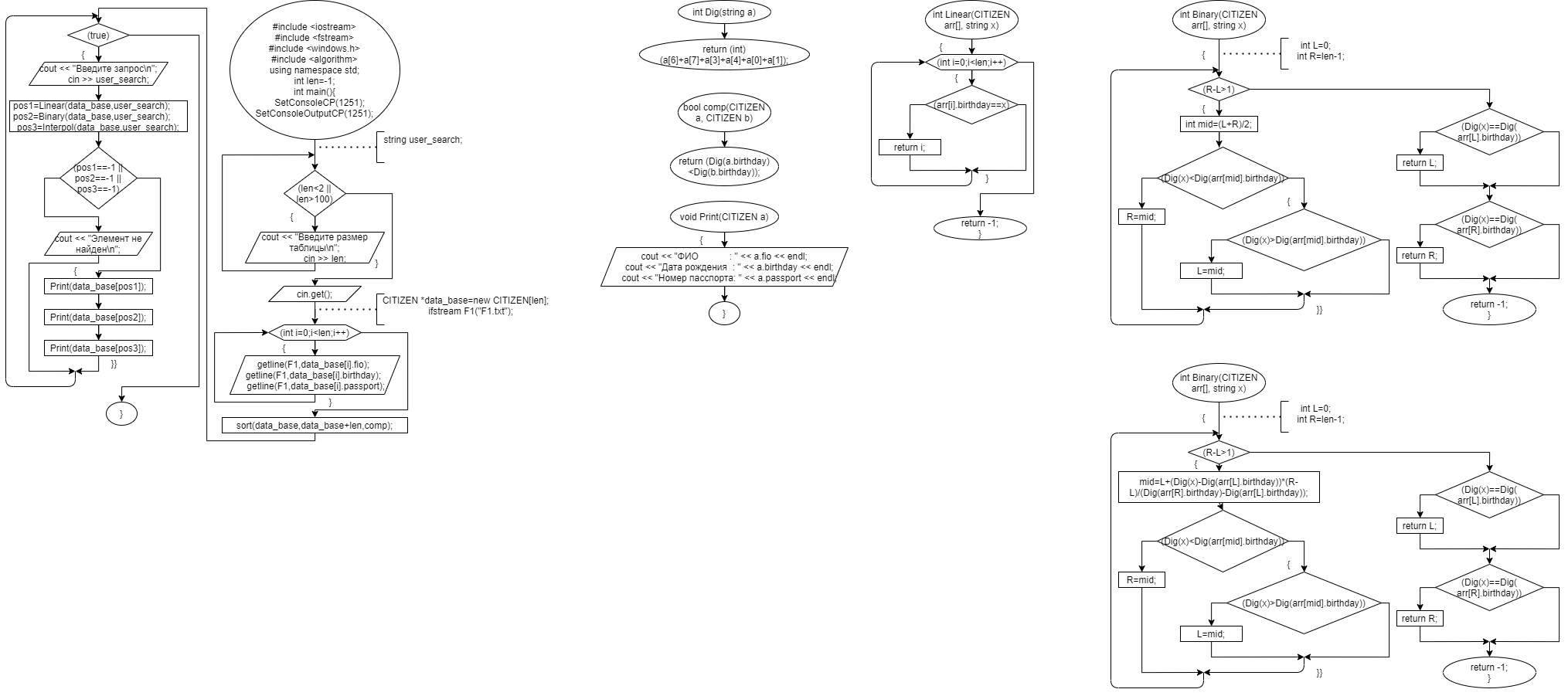
****

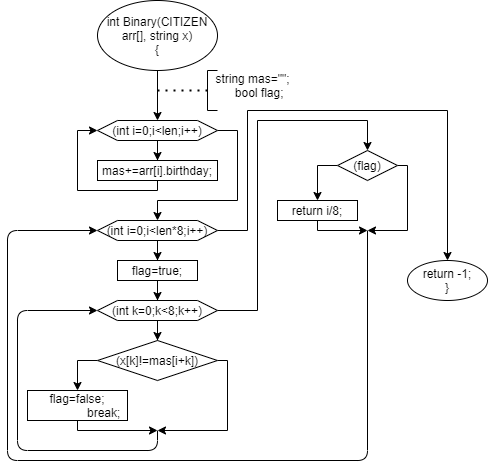
****

****

****

****

****

****

**Код**

#include <iostream>

#include <fstream>

#include <windows.h>

#include <algorithm>

using namespace std;

int len=-1;

struct CITIZEN

{

string fio;

string birthday;

string passport;

};

int Dig(string a)

{

return (int)(a[6]+a[7]+a[3]+a[4]+a[0]+a[1]);

}

bool comp(CITIZEN a, CITIZEN b)

{

return (Dig(a.birthday)<Dig(b.birthday));

}

void Print(CITIZEN a)

{

cout << "ФИО : " << a.fio << endl;

cout << "Дата рождения : " << a.birthday << endl;

cout << "Номер пасспорта: " << a.passport << endl;

}

int Linear(CITIZEN arr[], string x)

{

for (int i=0;i<len;i++) if (arr[i].birthday==x) return i;

return -1;

}

int Binary(CITIZEN arr[], string x)

{

string mas="";

bool flag;

for (int i=0;i<len;i++) mas+=arr[i].birthday;

for (int i=0;i<len\*8;i++)

{

flag=true;

for (int k=0;k<8;k++)

{

if (x[k]!=mas[i+k])

{

flag=false;

break;

}

}

if (flag) return i/8;

}

return -1;

}

int Interpol(CITIZEN arr[], string x)

{

int L=0;

int R=len-1;

while (R-L>1)

{

int mid=L+(Dig(x)-Dig(arr[L].birthday))\*(R-L)/(Dig(arr[R].birthday)-Dig(arr[L].birthday));

if (Dig(x)<Dig(arr[mid].birthday)) R=mid;

else if (Dig(x)>Dig(arr[mid].birthday)) L=mid;

else return mid;

}

if (Dig(x)==Dig(arr[L].birthday)) return L;

if (Dig(x)==Dig(arr[R].birthday)) return R;

return -1;

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

string user\_search;

while (len<2 || len>100)

{

cout << "Введите размер таблицы\n";

cin >> len;

}

cin.get();

CITIZEN \*data\_base=new CITIZEN[len];

ifstream F1("F1.txt");

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

{

getline(F1,data\_base[i].fio);

getline(F1,data\_base[i].birthday);

getline(F1,data\_base[i].passport);

}

sort(data\_base,data\_base+len,comp);

while (true)

{

cout << "Введите запрос\n";

cin >> user\_search;

int pos1=Linear(data\_base,user\_search);

int pos2=Binary(data\_base,user\_search);

int pos3=Interpol(data\_base,user\_search);

if (pos1==-1 || pos2==-1 || pos3==-1) cout << "Элемент не найден\n";

else

{

cout << "Результат линейного поиска:\n";

Print(data\_base[pos1]);

cout << "Результат бинарного поиска:\n";

Print(data\_base[pos2]);

cout << "Результат интерполяционного поиска:\n";

Print(data\_base[pos3]);

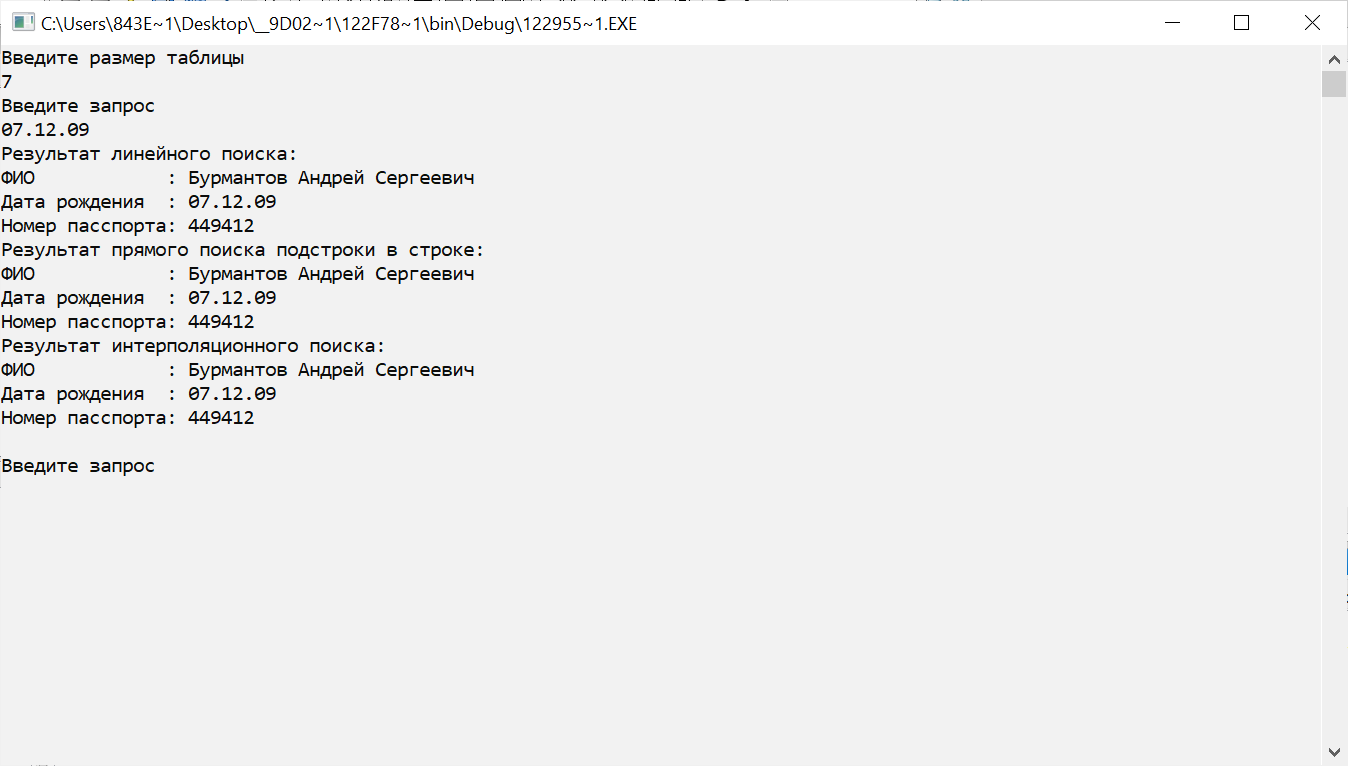
cout << endl;

}

}

}

**Скриншоты**

****