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 алгоритмизации и программирования

Тема: “Поиск данных в динамических структурах по заданному ключу”

Вариант 16

Выполнил:

студент группы ИВТ-20-2Б Тедеев А.З.

Проверила: доцент кафедры ИТАС

Полякова О.А.

Пермь, 2021

**Постановка задачи**

1. Создать динамический массив из записей (в соответствии с вариантом), содержащий не менее 100 элементов. Для заполнения элементов массива использовать ГСЧ.

2. Предусмотреть сохранение массива в файл.

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 линейный, интерполяционный и подстроки в строке.

2

**Анализ задачи**

**1.** Для решения задачи необходимо:

**1.1.** Организовать функцию LinearSearch(), которая отвечает за линейный поиск.

**1.2.** Организовать функцию SearchInSubstrInString(), которая отвечает за поиск подстроки в строке.

**1.3.** Организовать функцию Interpolation(), которая отвечает за интерполяционный поиск.

**1.4.** Организовать функции Addition() и Addition2() для добавления элементов в массив структур.

**1.5.** Организовать функции Deletion() и Deletion2() для удаления элементов из массива структур.

**1.6.** Организовать функцию SaveToFile(), которая будет сохранять список в текстовом файле.

**1.7.** Организовать функцию, которая будет проверять, найден ли результат после вызова одной из функций поиска – CheckingResult().

**1.8.** Организовать структуру Human.

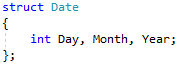
**1.9.** Организовать структуру Date.

**2.** В ходе работы были использованы следующие типы данных:

**2.1.** enum для повышения читаемости кода, так как используются функции switch.

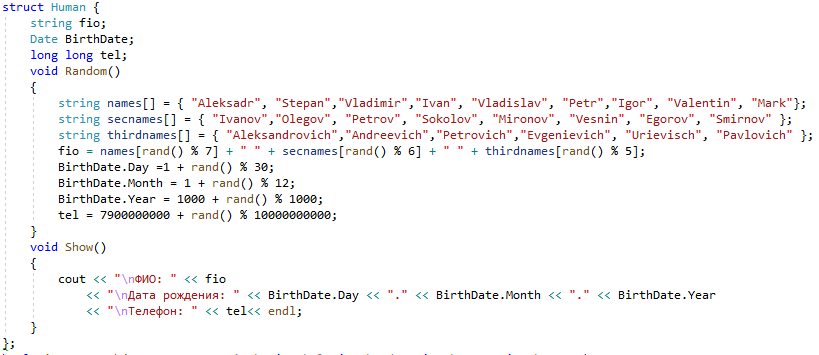


**2.2.** Структура Date для хранения дат рождения.



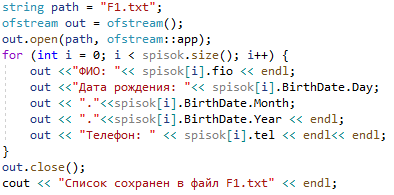
**2.3.** Структура Human для хранения данных людях.

3



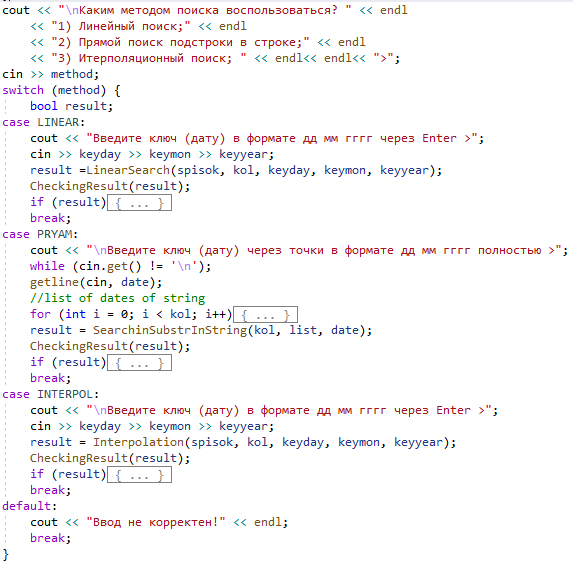
Поле fio типа string для хранения имени, фамилии и отчества. Поле BirthDate типа Date для хранения даты рождения (в свою очередь структура Date хранит дни, недели, годы в разных полях типа int) и поле tel типа long long для хранения номера телефона. Метод Random(), в котором генерируются случайным образом в каждом из полей данные: под поле fio используются три массива строк, в которые соответственно вписаны имена, фамилии и отчества, при заполнении поля fio используется функция rand(), которая и перебирает данные.

**2.4.** В функции SaveToFile используются команды из библиотеки fstream для сохранения списка в файл.



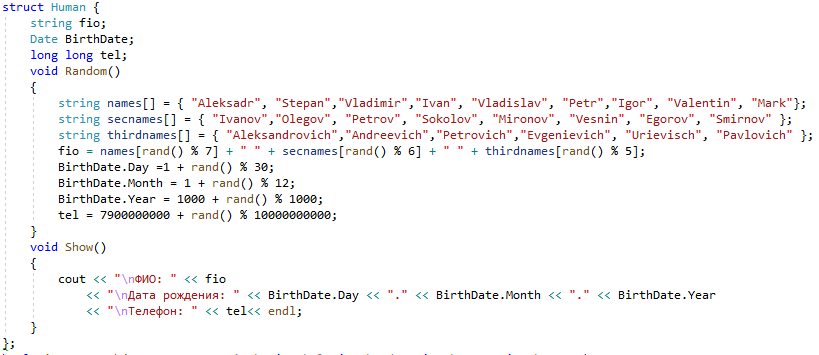
**2.5.** В главной функции используется многократно функция switch под «меню», например, под выбор метода поиска, выбора удаления или добавления элемента и т.д.

4



**3.** Для решения задачи данные были представлены в следующем виде:

**3.1.** Для обработки данных используется массив структур Human.



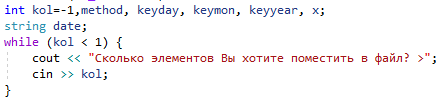
5

**3.2.** Для хранения данных используется вектор типа Human

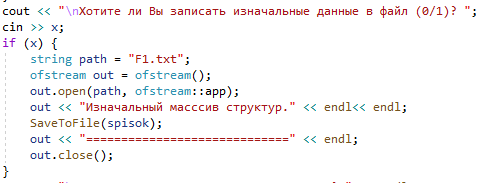
vector<Human> spisok(kol);

**4.** Для операций ввода и вывода использовались следующие операторы и функции:

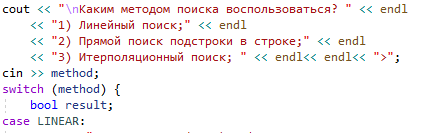
**4.1.** Для ввода количества элементов используется переменная kol. Ввод осуществляется командой cin.



**4.2.** Временная переменная x используется для предоставления пользователю выбора, сохранять массив структур в файл или нет. Ввод осуществляется командой cin.



**4.3.** Переменная method необходима для предоставления пользователю выбора метода поиска в массиве структур. Ввод осуществляется командой cin.

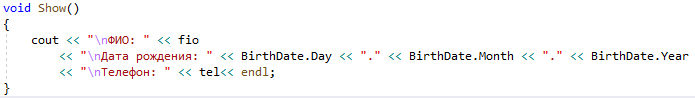


**4.4.** Переменные keyday, keymon, keyyear типа int необходимы для хранения ключа (т.е. даты). Вводятся через функцию cin по порядку.



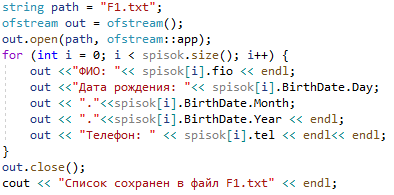
6

**4.5.** Вывод на консоль осуществляется с помощью метода Show() структуры Human в цикле for.



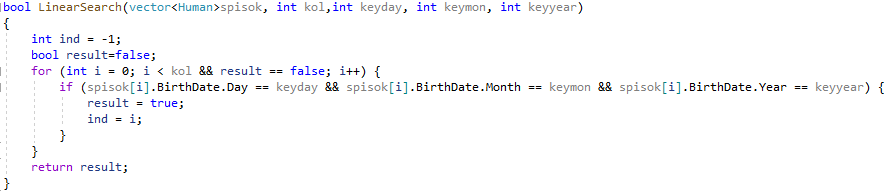


**4.6.** Вывод в файл осуществляется через функцию SaveToFile().



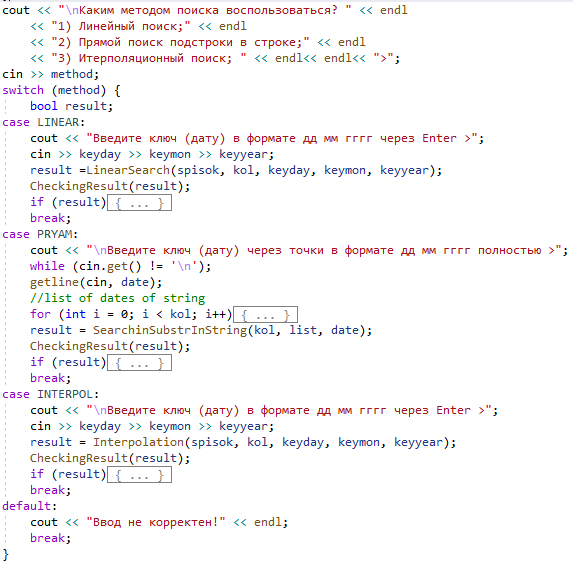
**5.** Поставленные задачи будут решены следующими действиями:

**5.1.** Вызов функции LinearSearch() типа bool присваивается переменной result. В функции LinearSearch() производится поиск элемента по ключу методом линейного поиска.

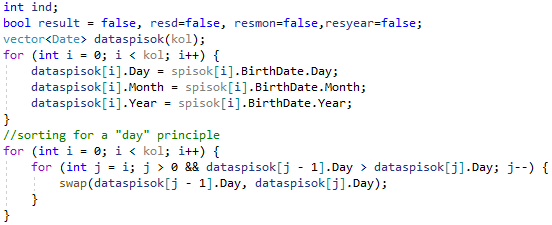


7

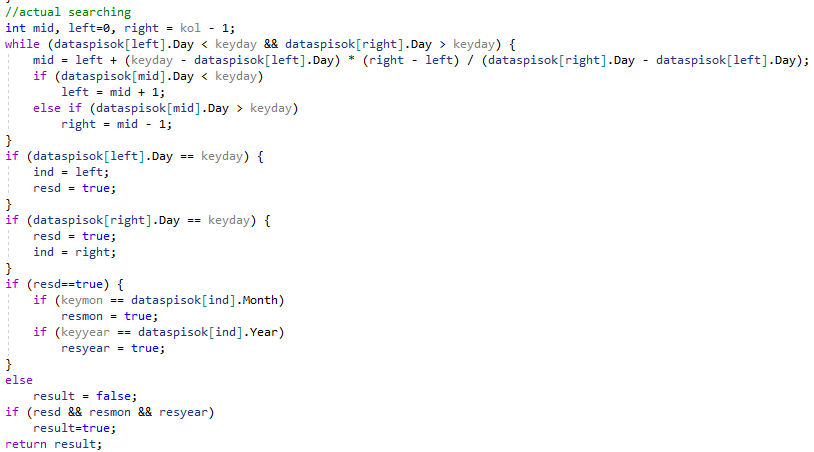
**5.2.** Меню с выбором метода поиска выполняется с помощью функции switch



**5.3.** Вызов функции Interpolation() также осуществляется при присвоении переменной result. В данной функции сначала заводится дополнительный вектор, который хранит даты рождения из списка, затем этот же дополнительный массив сортируется по дням (так как в структуре Date все данные распределены по переменным Day, Month, Year) методом вставки. Далее в отсортированном векторе уже производится поиск интерполяционный, где если нужный день (по ключу) находится, то переменная resd получает «статус» true и запоминается номер этого элемента в дополнительном векторе. Далее в ветвлениях уже идет проверка конкретного элемента, индекс которого был вычислен выше. Переменная result становится true, когда resd, resmon и resyear становятся true.

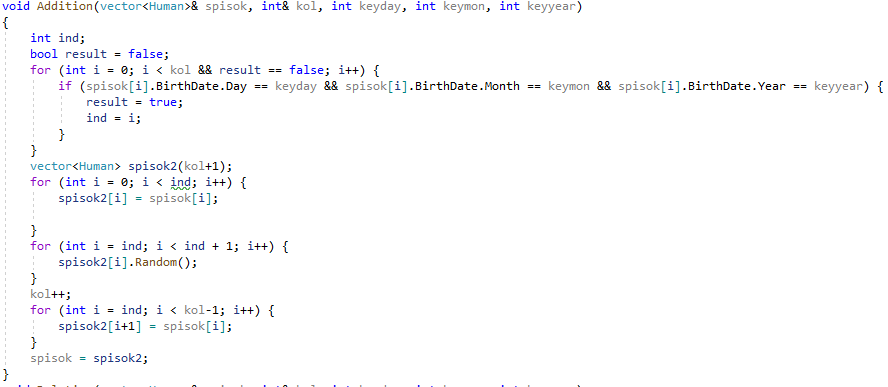


8



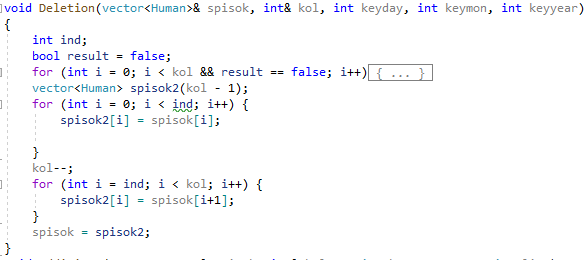
**5.4.** Если результат поиска положителен, то после выполнения функции CheckingResult() пользователь выбирает функцию добавления или удаления по ключу.

**5.5.** Добавление осуществляется в функции Addition(). Функция использует линейный поиск для нахождения номера нужного элемента, а затем в 3 циклах for производится обмен элементами между дополнительным вектором и изначальным.



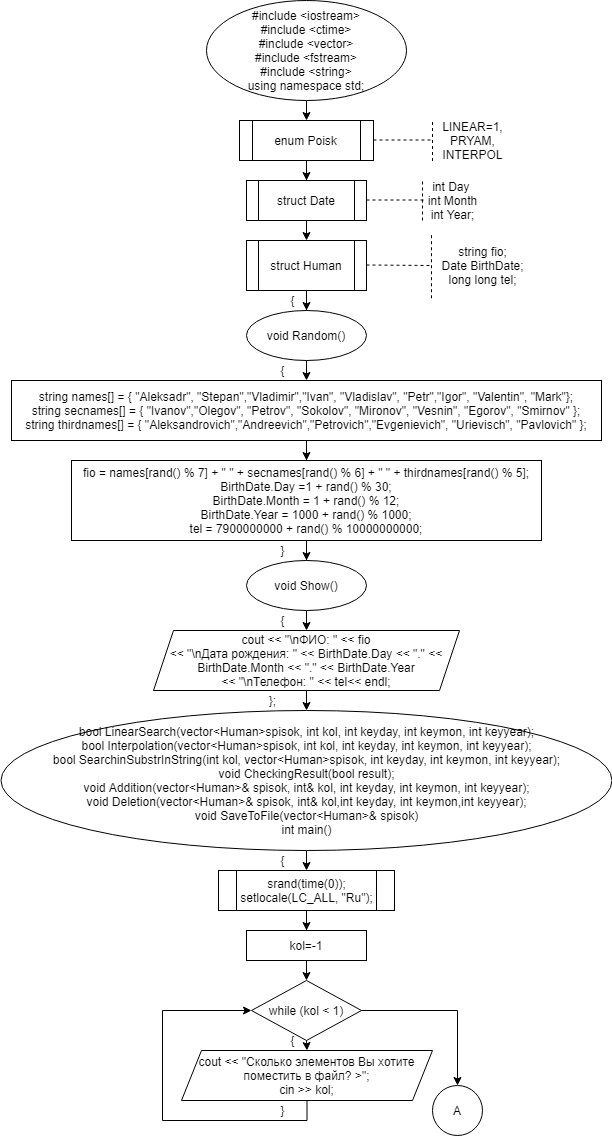
**5.5.** Удаление осуществляется в функции Deletion(). Функция использует линейный поиск для нахождения номера нужного элемента, а затем в 2 циклах for производится обмен элементами между дополнительным вектором и изначальным.

9

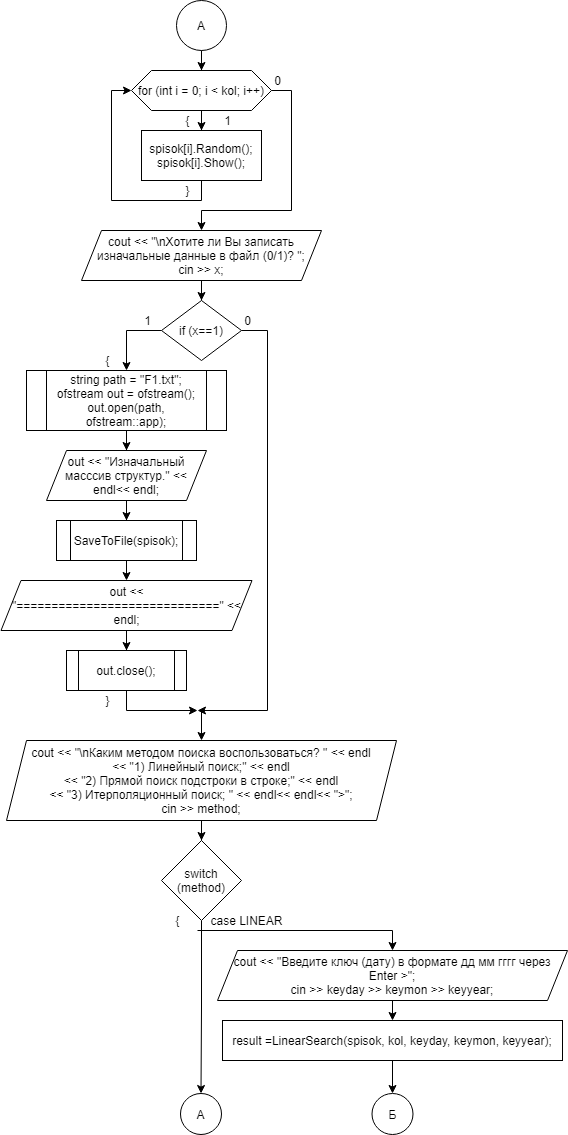


10

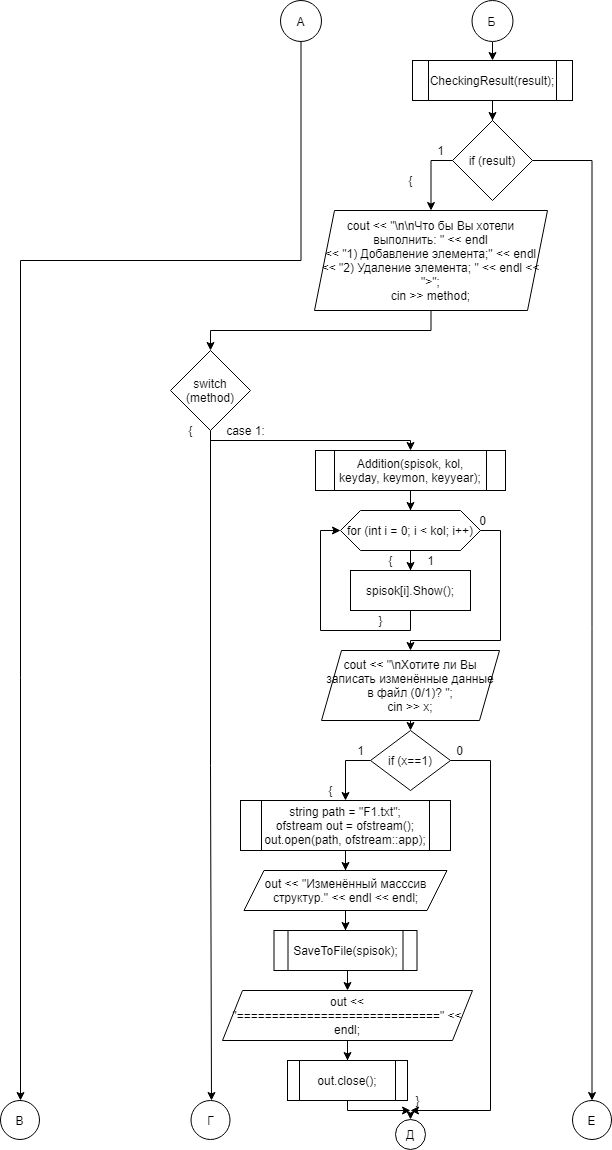
**Блок-схема**



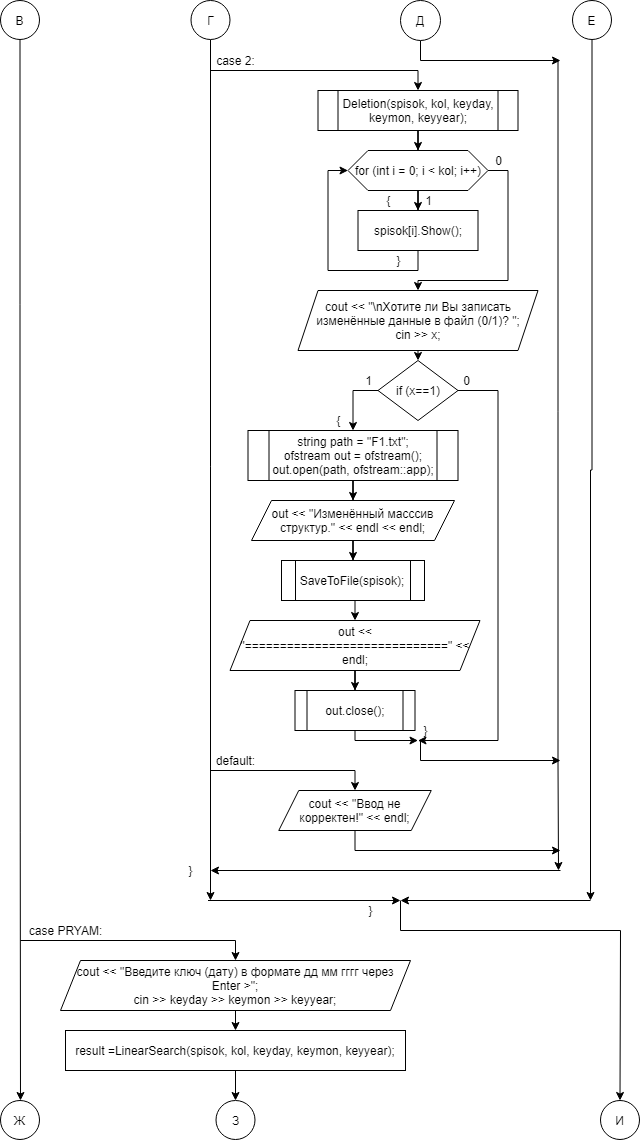
11

****

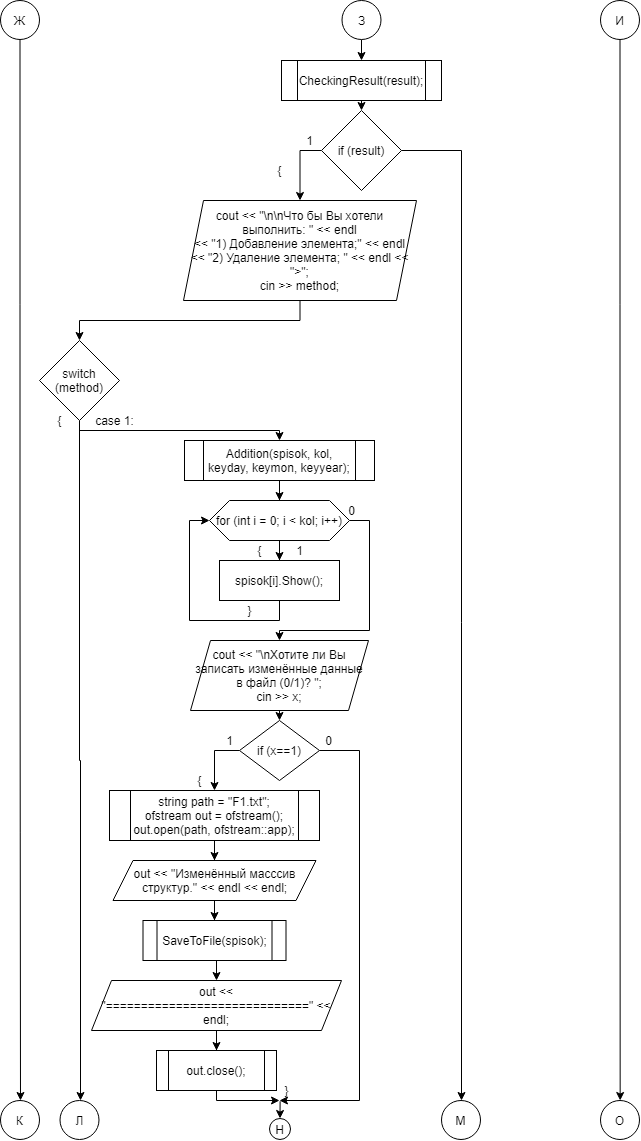
12



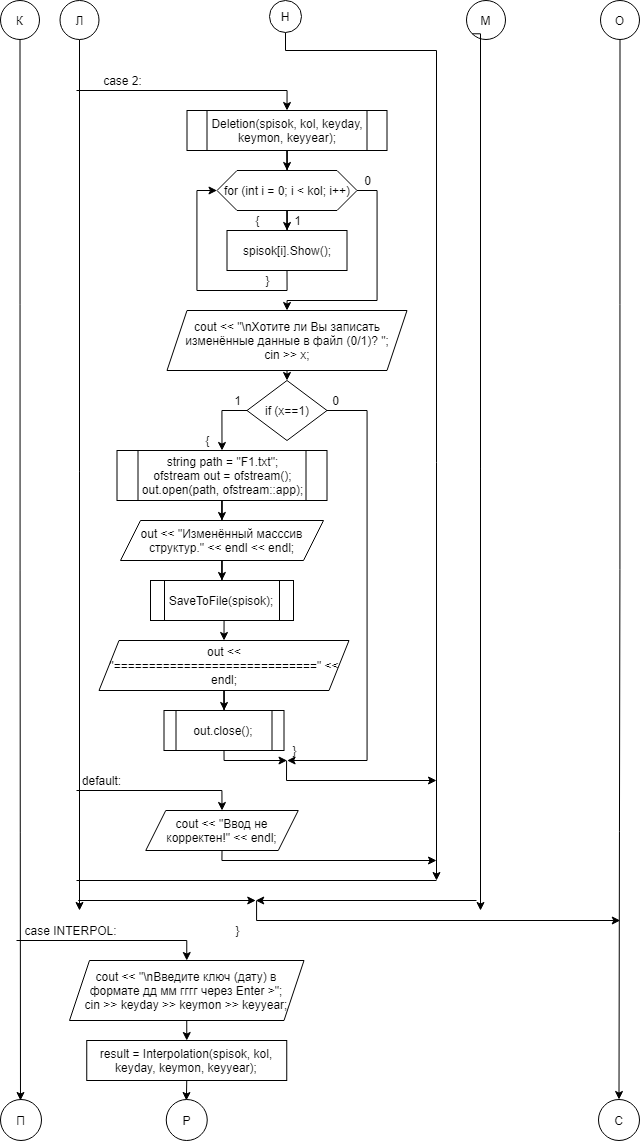
13

****

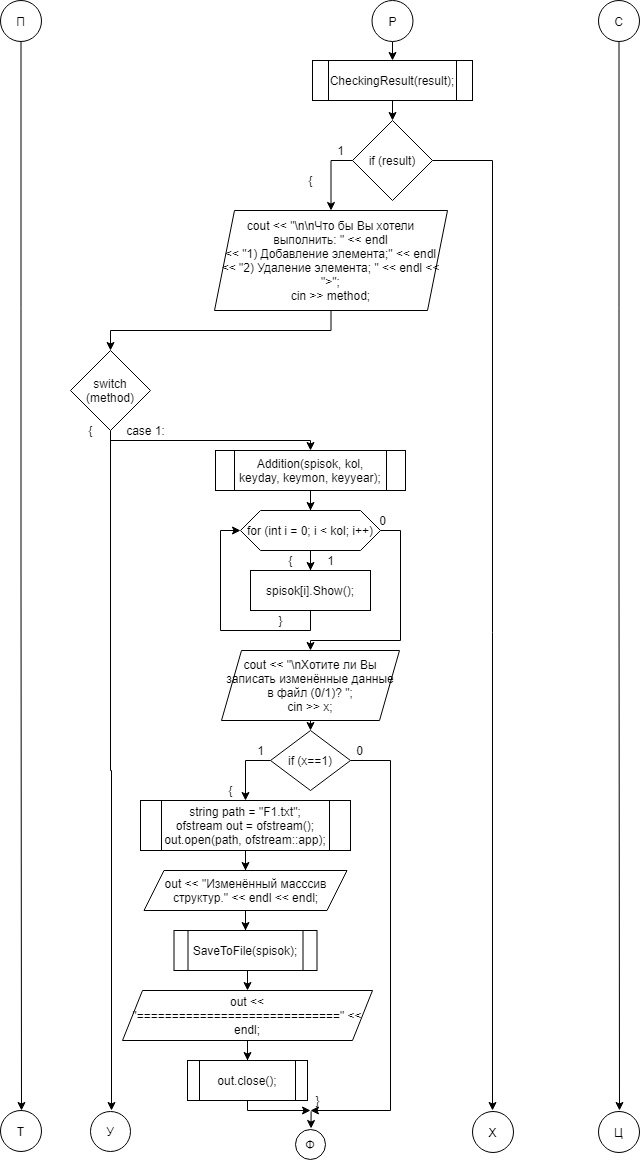
14

****

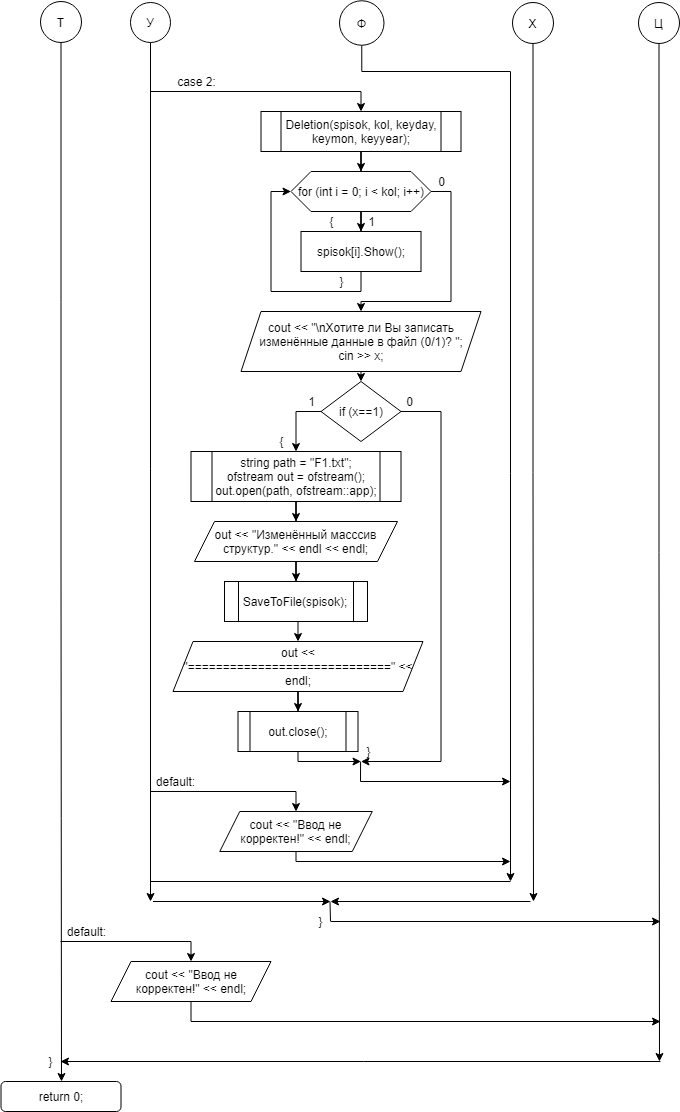
15



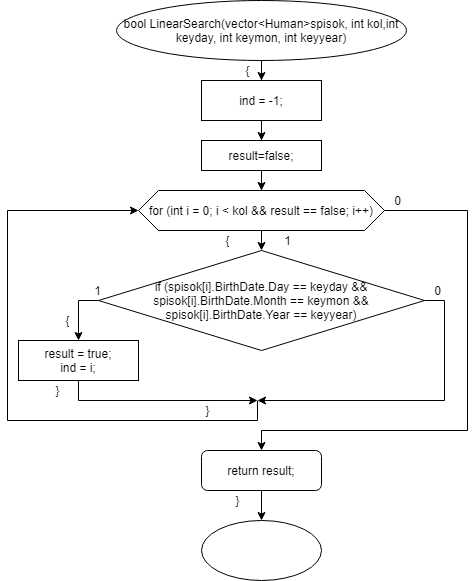
16



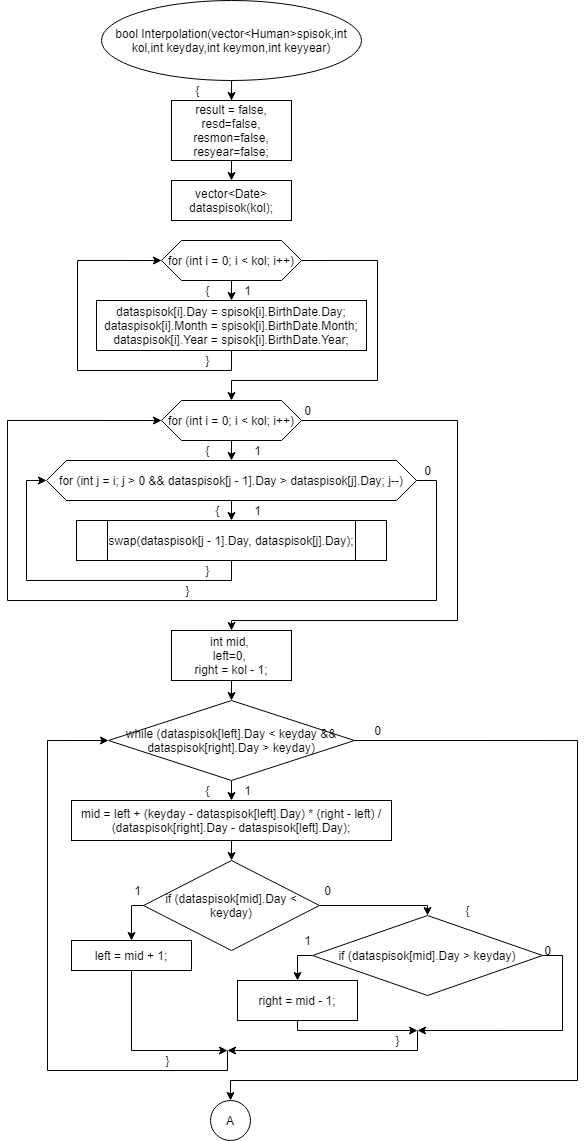
17

****

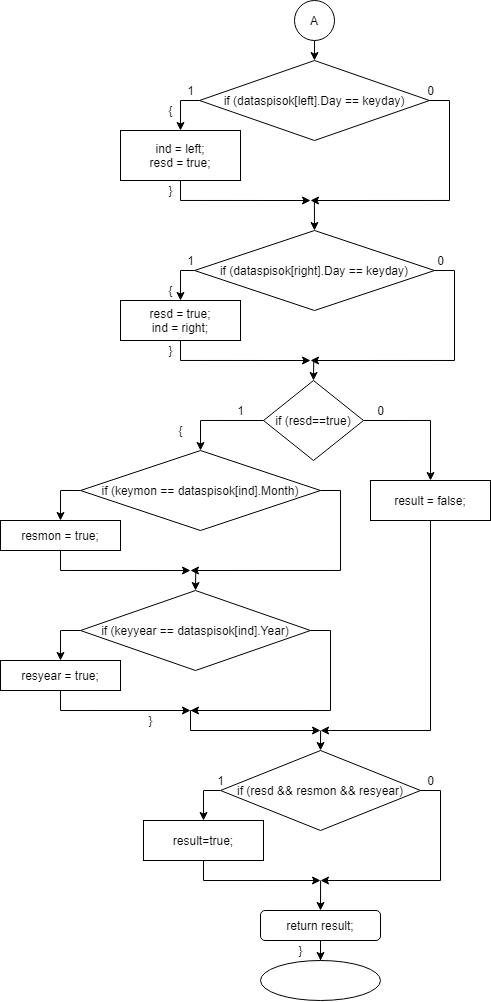
18

****

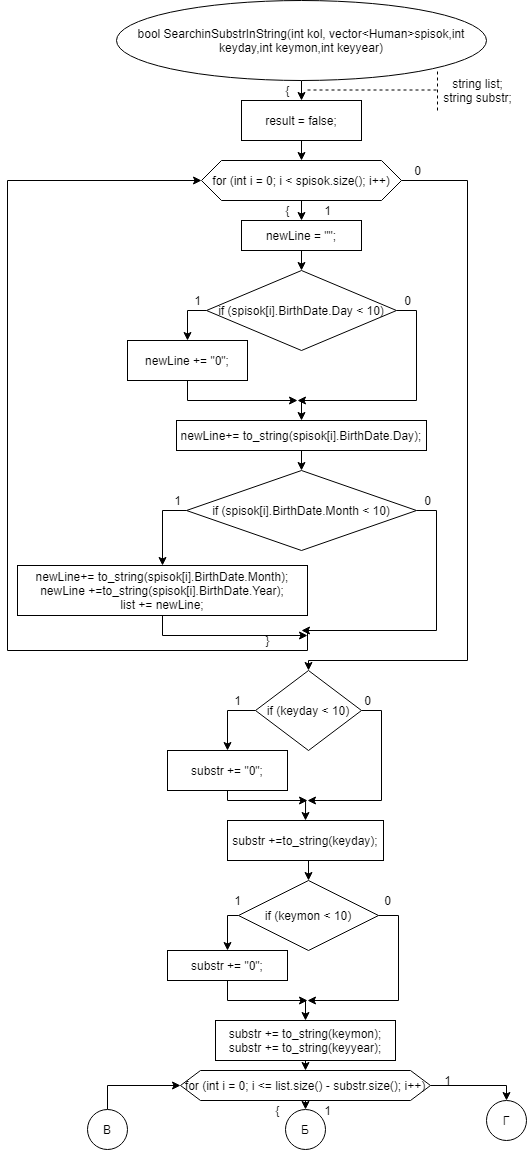
19

****

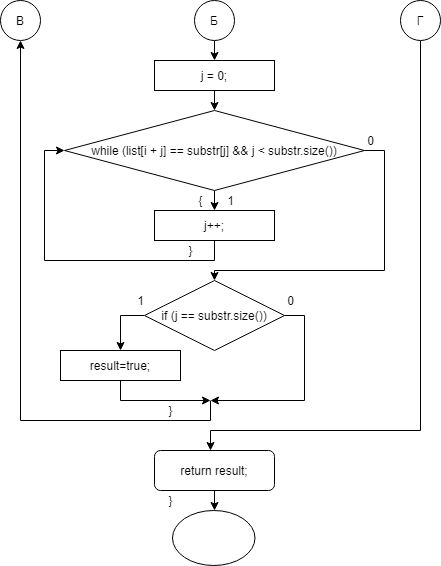
20



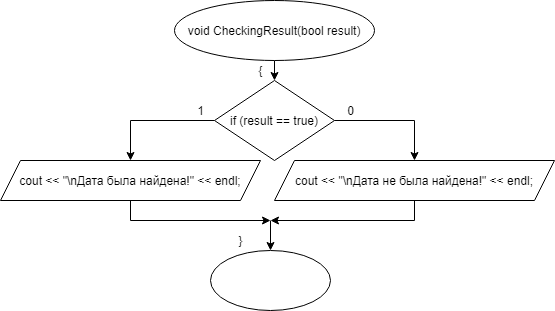
21



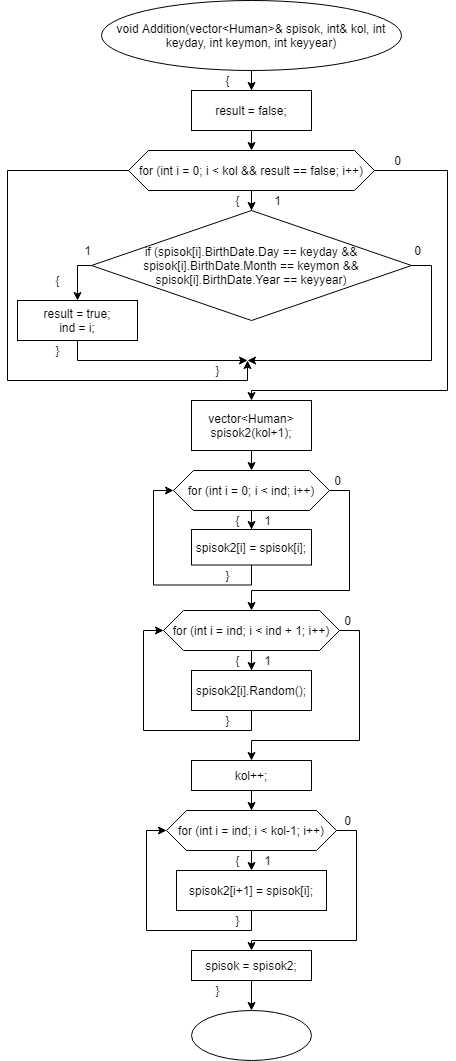
22



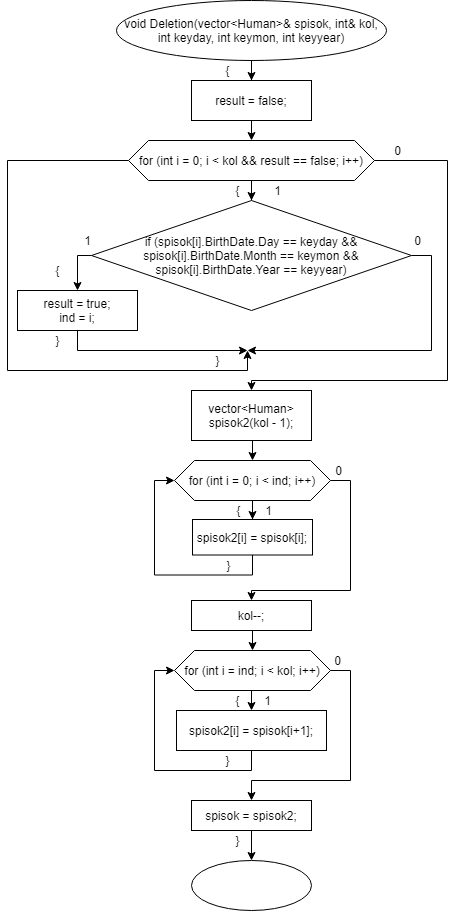
23



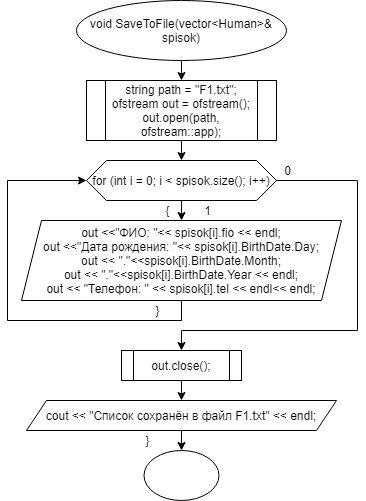
24



25



26



27

**Код**

#include <iostream>

#include <ctime>

#include <vector>

#include <fstream>

#include <string>

using namespace std;

enum Poisk{

LINEAR=1,

PRYAM,

INTERPOL

};

struct Date

{

int Day, Month, Year;

};

struct Human {

string fio;

Date BirthDate;

long long tel;

void Random()

{

string names[] = { "Aleksadr", "Stepan","Vladimir","Ivan", "Vladislav", "Petr","Igor", "Valentin", "Mark"};

string secnames[] = { "Ivanov","Olegov", "Petrov", "Sokolov", "Mironov", "Vesnin", "Egorov", "Smirnov" };

string thirdnames[] = { "Aleksandrovich","Andreevich","Petrovich","Evgenievich", "Urievisch", "Pavlovich" };

fio = names[rand() % 7] + " " + secnames[rand() % 6] + " " + thirdnames[rand() % 5];

28

BirthDate.Day =1 + rand() % 30;

BirthDate.Month = 1 + rand() % 12;

BirthDate.Year = 1000 + rand() % 1000;

tel = 7900000000 + rand() % 10000000000;

}

void Show()

{

cout << "\nФИО: " << fio

<< "\nДата рождения: " << BirthDate.Day << "." << BirthDate.Month << "." << BirthDate.Year

<< "\nТелефон: " << tel<< endl;

}

};

bool LinearSearch(vector<Human>spisok, int kol, int keyday, int keymon, int keyyear);

bool Interpolation(vector<Human>spisok, int kol, int keyday, int keymon, int keyyear);

bool SearchinSubstrInString(int kol, vector<Human>spisok, int keyday, int keymon, int keyyear);

void CheckingResult(bool result);

void Addition(vector<Human>& spisok, int& kol, int keyday, int keymon, int keyyear);

void Deletion(vector<Human>& spisok, int& kol,int keyday, int keymon,int keyyear);

void SaveToFile(vector<Human>& spisok);

int main()

{

srand(time(0));

setlocale(LC\_ALL, "Ru");

int kol=-1,method, keyday, keymon, keyyear, x;

while (kol < 1) {

29

cout << "Сколько элементов Вы хотите поместить в файл? >";

cin >> kol;

}

vector<Human> spisok(kol);

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

spisok[i].Random();

spisok[i].Show();

}

cout << "\nХотите ли Вы записать изначальные данные в файл (0/1)? ";

cin >> x;

if (x==1) {

string path = "F1.txt";

ofstream out = ofstream();

out.open(path, ofstream::app);

out << "Изначальный масссив структур." << endl<< endl;

SaveToFile(spisok);

out << "=============================" << endl;

out.close();

}

cout << "\nКаким методом поиска воспользоваться? " << endl

<< "1) Линейный поиск;" << endl

<< "2) Прямой поиск подстроки в строке;" << endl

<< "3) Итерполяционный поиск; " << endl<< endl<< ">";

cin >> method;

switch (method) {

bool result;

case LINEAR:

cout << "Введите ключ (дату) в формате дд мм гггг через Enter >";

30

cin >> keyday >> keymon >> keyyear;

result =LinearSearch(spisok, kol, keyday, keymon, keyyear);

CheckingResult(result);

if (result) {

cout << "\n\nЧто бы Вы хотели выполнить: " << endl

<< "1) Добавление элемента;" << endl

<< "2) Удаление элемента; " << endl << ">";

cin >> method;

switch (method) {

case 1:

Addition(spisok, kol, keyday, keymon, keyyear);

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

spisok[i].Show();

}

cout << "\nХотите ли Вы записать изменённые данные в файл (0/1)? ";

cin >> x;

if (x==1) {

string path = "F1.txt";

ofstream out = ofstream();

out.open(path, ofstream::app);

out << "Изменённый масссив структур." << endl << endl;

SaveToFile(spisok);

out << "=============================" << endl;

out.close();

}

break;

case 2:

31

Deletion(spisok, kol, keyday, keymon, keyyear);

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

spisok[i].Show();

}

cout << "\nХотите ли Вы записать изменённые данные в файл (0/1)? ";

cin >> x;

if (x==1) {

string path = "F1.txt";

ofstream out = ofstream();

out.open(path, ofstream::app);

out << "Изменённый масссив структур." << endl << endl;

SaveToFile(spisok);

out << "=============================" << endl;

out.close();

}

break;

default:

cout << "Ввод не корректен!" << endl;

break;

}

}

break;

case PRYAM:

cout << "\nВведите ключ (дату) через точки в формате дд мм гггг через Enter >";

cin >> keyday >> keymon >> keyyear;

result = SearchinSubstrInString(kol, spisok, keyday, keymon, keyyear);

CheckingResult(result);

32

if (result) {

cout << "\n\nЧто бы Вы хотели выполнить: " << endl

<< "1) Добавление элемента;" << endl

<< "2) Удаление элемента; " << endl << ">";

cin >> method;

switch (method) {

case 1:

Addition(spisok, kol, keyday, keymon, keyyear);

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

spisok[i].Show();

}

cout << "\nХотите ли Вы записать изменённые данные в файл (0/1)? ";

cin >> x;

if (x==1) {

string path = "F1.txt";

ofstream out = ofstream();

out.open(path, ofstream::app);

out << "Изменённый масссив структур." << endl << endl;

SaveToFile(spisok);

out << "=============================" << endl;

out.close();

}

break;

case 2:

Deletion(spisok, kol, keyday, keymon, keyyear);

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

spisok[i].Show();

}

33

cout << "\nХотите ли Вы записать изменённые данные в файл (0/1)? ";

cin >> x;

if (x==1) {

string path = "F1.txt";

ofstream out = ofstream();

out.open(path, ofstream::app);

out << "Изменённый масссив структур." << endl << endl;

SaveToFile(spisok);

out << "=============================" << endl;

out.close();

}

break;

default:

cout << "Ввод не корректен!" << endl;

break;

}

}

break;

case INTERPOL:

cout << "\nВведите ключ (дату) в формате дд мм гггг через Enter >";

cin >> keyday >> keymon >> keyyear;

result = Interpolation(spisok, kol, keyday, keymon, keyyear);

CheckingResult(result);

if (result) {

cout << "\n\nЧто бы Вы хотели выполнить: " << endl

<< "1) Добавление элемента;" << endl

<< "2) Удаление элемента; " << endl << ">";

cin >> method;

34

switch (method)

{

case 1:

Addition(spisok, kol, keyday, keymon, keyyear);

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

spisok[i].Show();

}

cout << "\nХотите ли Вы записать изменённые данные в файл (0/1)? ";

cin >> x;

if (x==1) {

string path = "F1.txt";

ofstream out = ofstream();

out.open(path, ofstream::app);

out << "Изменённый масссив структур." << endl;

SaveToFile(spisok);

out << "=============================" << endl;

out.close();

}

break;

case 2:

Deletion(spisok, kol, keyday, keymon, keyyear);

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

spisok[i].Show();

}

cout << "\nХотите ли Вы записать изменённые данные в файл (0/1)? ";

cin >> x;

if (x==1) {

string path = "F1.txt";

35

ofstream out = ofstream();

out.open(path, ofstream::app);

out << "Изменённый масссив структур." << endl;

SaveToFile(spisok);

out << "=============================" << endl;

out.close();

}

break;

default:

cout << "Ввод не корректен!" << endl;

break;

}

}

break;

default:

cout << "Ввод не корректен!" << endl;

break;

}

}

bool LinearSearch(vector<Human>spisok, int kol,int keyday, int keymon, int keyyear)

{

int ind = -1;

bool result=false;

for (int i = 0; i < kol && result == false; i++) {

if (spisok[i].BirthDate.Day == keyday && spisok[i].BirthDate.Month == keymon && spisok[i].BirthDate.Year == keyyear) {

result = true;

ind = i;

36

}

}

return result;

}

bool Interpolation(vector<Human>spisok,int kol,int keyday,int keymon,int keyyear)

{

int ind;

bool result = false, resd=false, resmon=false,resyear=false;

vector<Date> dataspisok(kol);

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

dataspisok[i].Day = spisok[i].BirthDate.Day;

dataspisok[i].Month = spisok[i].BirthDate.Month;

dataspisok[i].Year = spisok[i].BirthDate.Year;

}

//sorting for a "day" principle

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

for (int j = i; j > 0 && dataspisok[j - 1].Day > dataspisok[j].Day; j--) {

swap(dataspisok[j - 1].Day, dataspisok[j].Day);

}

}

//actual searching

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

while (dataspisok[left].Day < keyday && dataspisok[right].Day > keyday) {

mid = left + (keyday - dataspisok[left].Day) \* (right - left) / (dataspisok[right].Day - dataspisok[left].Day);

if (dataspisok[mid].Day < keyday)

left = mid + 1;

else if (dataspisok[mid].Day > keyday)

right = mid - 1;

37

}

if (dataspisok[left].Day == keyday) {

ind = left;

resd = true;

}

if (dataspisok[right].Day == keyday) {

resd = true;

ind = right;

}

if (resd==true) {

if (keymon == dataspisok[ind].Month)

resmon = true;

if (keyyear == dataspisok[ind].Year)

resyear = true;

}

else

result = false;

if (resd && resmon && resyear)

result=true;

return result;

}

bool SearchinSubstrInString(int kol, vector<Human>spisok,int keyday,int keymon,int keyyear)

{

bool result = false;

string list;

for (int i = 0; i < spisok.size(); i++) {

string newLine = "";

if (spisok[i].BirthDate.Day < 10) newLine += "0";

newLine+= to\_string(spisok[i].BirthDate.Day);

38

if (spisok[i].BirthDate.Month < 10) newLine += "0";

newLine+= to\_string(spisok[i].BirthDate.Month);

newLine +=to\_string(spisok[i].BirthDate.Year);

list += newLine;

}

string substr;

if (keyday < 10)

substr += "0";

substr +=to\_string(keyday);

if (keymon < 10)

substr += "0";

substr += to\_string(keymon);

substr += to\_string(keyyear);

for (int i = 0; i <= list.size() - substr.size(); i++) {

int j = 0;

while (list[i + j] == substr[j] && j < substr.size()) j++;

if (j == substr.size()) result=true;

}

return result;

}

void CheckingResult(bool result)

{

if (result == true)

cout << "\nДата была найдена!" << endl;

else

cout << "\nДата не была найдена!" << endl;

}

void Addition(vector<Human>& spisok, int& kol, int keyday, int keymon, int keyyear)

{

39

int ind;

bool result = false;

for (int i = 0; i < kol && result == false; i++) {

if (spisok[i].BirthDate.Day == keyday && spisok[i].BirthDate.Month == keymon && spisok[i].BirthDate.Year == keyyear) {

result = true;

ind = i;

}

}

vector<Human> spisok2(kol+1);

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

spisok2[i] = spisok[i];

}

for (int i = ind; i < ind + 1; i++) {

spisok2[i].Random();

}

kol++;

for (int i = ind; i < kol-1; i++) {

spisok2[i+1] = spisok[i];

}

spisok = spisok2;

}

void Deletion(vector<Human>& spisok, int& kol, int keyday, int keymon, int keyyear)

{

int ind;

bool result = false;

for (int i = 0; i < kol && result == false; i++) {

40

if (spisok[i].BirthDate.Day == keyday && spisok[i].BirthDate.Month == keymon && spisok[i].BirthDate.Year == keyyear) {

result = true;

ind = i;

}

}

vector<Human> spisok2(kol - 1);

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

spisok2[i] = spisok[i];

}

kol--;

for (int i = ind; i < kol; i++) {

spisok2[i] = spisok[i+1];

}

spisok = spisok2;

}

void SaveToFile(vector<Human>& spisok)

{

string path = "F1.txt";

ofstream out = ofstream();

out.open(path, ofstream::app);

for (int i = 0; i < spisok.size(); i++) {

out << "ФИО: " << spisok[i].fio << endl;

out << "Дата рождения: " << spisok[i].BirthDate.Day;

out << "." << spisok[i].BirthDate.Month;

out << "." << spisok[i].BirthDate.Year << endl;

out << "Телефон: " << spisok[i].tel << endl << endl;

}

out.close();

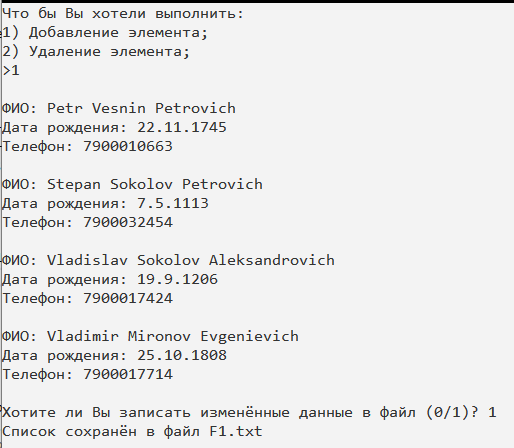
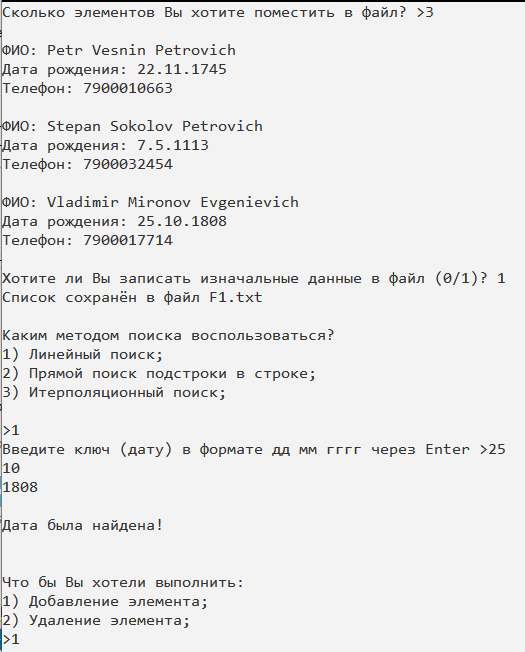
41

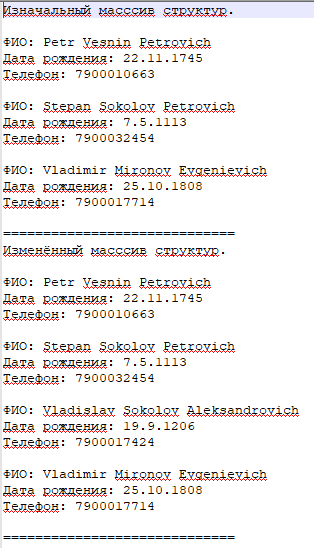
cout << "Список сохранён в файл F1.txt" << endl;

}

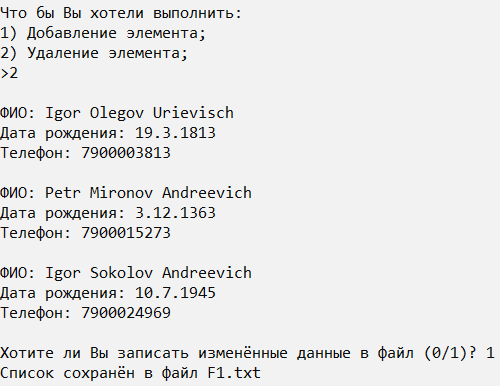
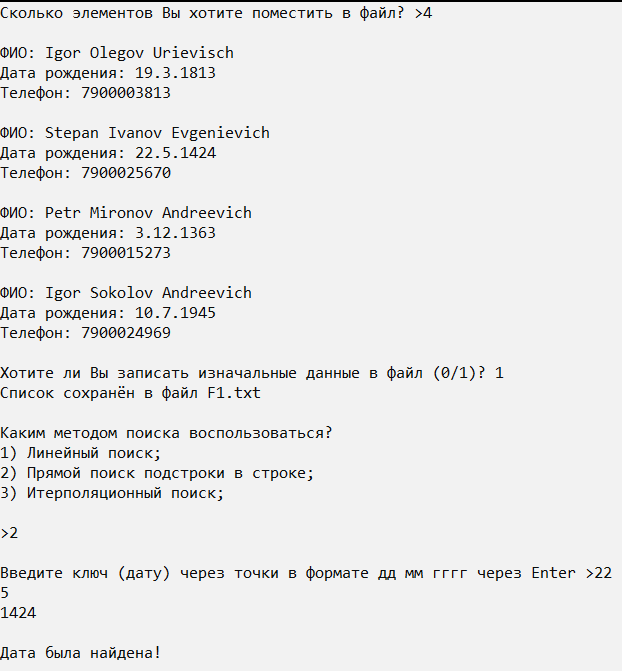
42

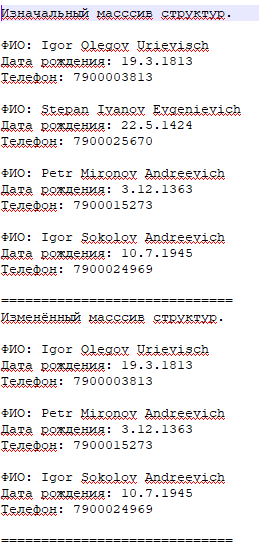
**Скриншоты**





43





44