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 №17**

Дисциплина: основы алгоритмизации и программирования

Тема: “Поиск данных с помощью хеш-таблиц”

Вариант 19

Выполнил:

Студент группы РИС-20-2Б

Пономарёв Артём Викторович

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 ДСЧ.

2 Выполнить поиск элемента в массиве по ключу в соответствии с вариантом. Для

поиска использовать хеш-таблицу.

3 Подсчитать количество коллизий при размере хеш-таблицы 40, 75 и 90

элементов.

2

**Анализ задачи**

**1.** Для решения задачи необходимо:

**1.1.** Организовать структуру Human для хранения фамилии, имени, отчества, номера телефона и даты рождения через соответствующие поля.

**1.2.** Организовать структуру Date для хранения даты рождения в полях Day, Month, Year типа int.

**1.3.** Организовать структуру HashTaB для формирования хеш таблицы с полями key типа string, Hash типа int и вектором типа Human – human\_list, а также методом Add\_List для инициализации вектора human\_list.

**1.4.** Организовать функцию MakeArr() типа vector<Human> для формирования массива структур типа Human – result, с помощью ДСЧ.

**1.5.** Организовать функцию MakeHashTab() типа void для формирования хеш-таблицы.

**1.6.** Организовать функцию InsertionSort() для сортировки методом вставки хеш-таблицы по хешу, который вычисляется в методе hashForming() структуры Human.

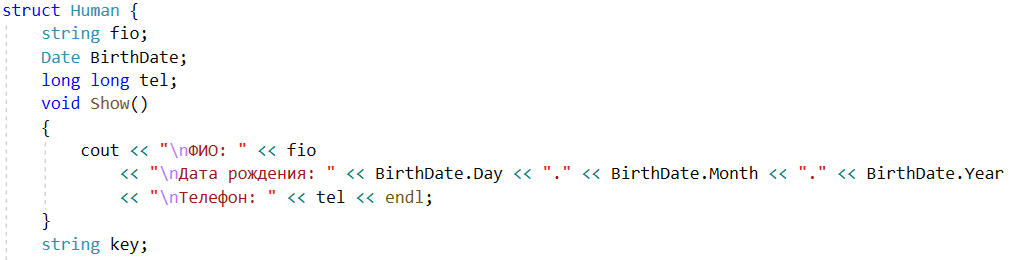
**1.7.** Организовать функцию Show() типа void для вывода хеш-таблицы в консоль.

**1.8.** Организовать функцию ConvertKey() типа string для преобразования введённого пользователем ключа для дальнейшей обработки в коде.

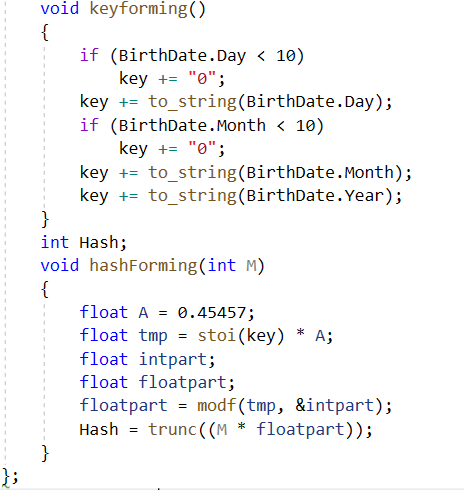
**1.9.** Организовать функцию Seeking() типа Human\* для поиска введённого пользователя ключа в хеш-таблице.

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

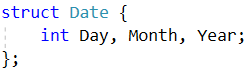
**2.1.** Структура Human для хранения фамилии, имени, отчества, номера телефона и даты рождения через соответствующие поля.



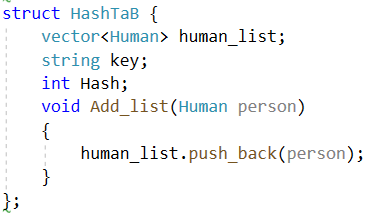
3



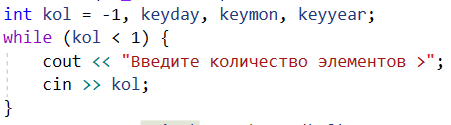
**2.2.** Организовать структуру Date для хранения даты рождения в полях Day, Month, Year типа int.



**2.3.** Организовать структуру HashTaB для формирования хеш таблицы с полями key типа string, Hash типа int и вектором типа Human – human\_list, а также методом Add\_List для инициализации вектора human\_list.



**2.4.** Переменные типа int: kol, keyday, keymon, keyyear, где kol – количество элементов в хеш-таблице, а keyday, keymon, keyyear – ввод ключа (даты) для поиска в хеш-таблице.



**2.5.** Вектор spisok типа Human для хранения сгенерированных элементов с помощью ДСЧ. Вектор Table типа HashTaB для хранения.

vector<Human> spisok = MakeArr(kol);

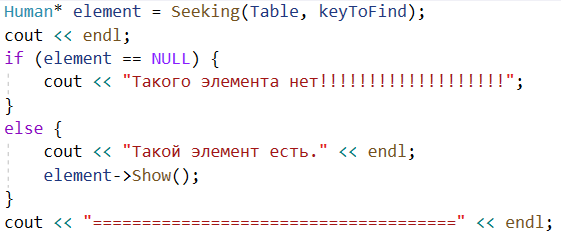
vector<HashTaB>Table;

4

**2.6.** Переменная keyToFind типа string для ввода ключа.

string keyToFind;

**2.7.** Указатель element типа для вывода найденного элемента по ключу, который получает значение из функции Seeking().

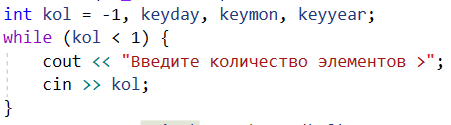


**3.** Для решения задачи данные были представлены в следующем виде:

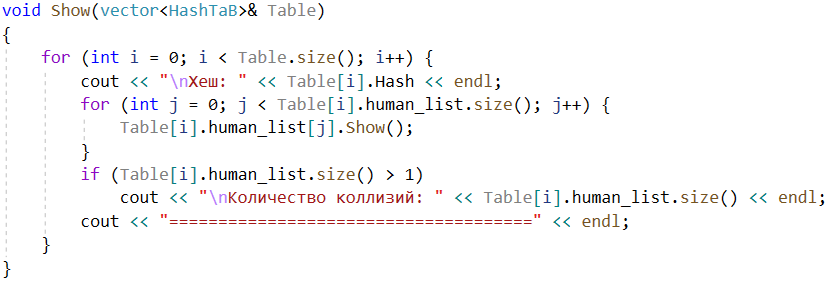
**3.1.** Данные генерируются в виде массива структур (тип Spisok и полями типа string).

**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** Вывод на консоль хеш-таблицы осуществляется функцией Show(), параметром в которую передаётся вектор Table.



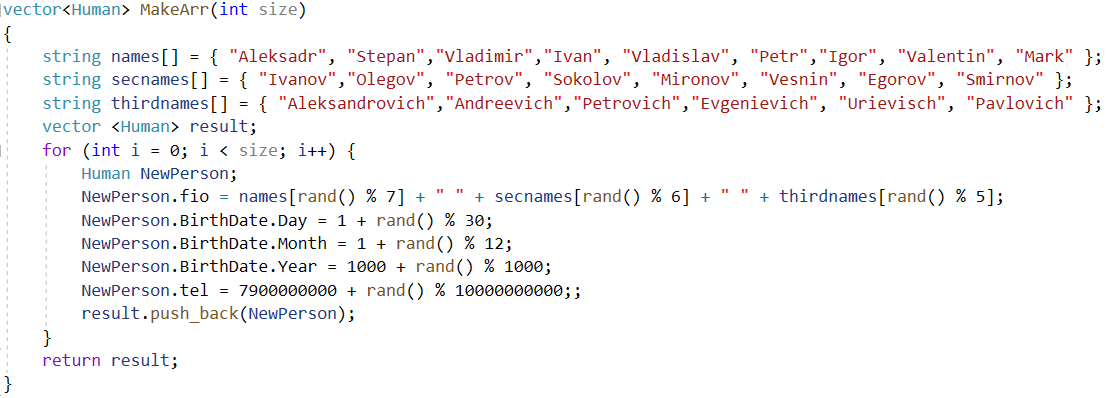
5

**4.3.** Ввод ключа производится функцией cin через переменные keyday, keymonth, keyyear.

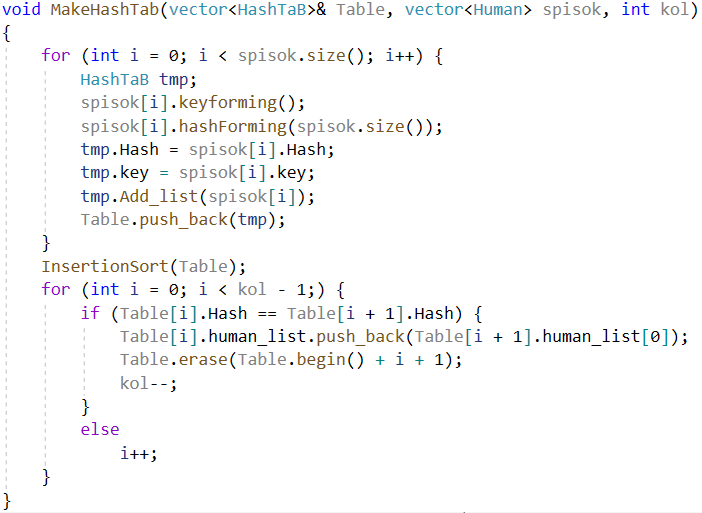
cin >> keyday >> keymon >> keyyear;

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

**5.1.** В функции MakeArr() передаётся переменная kol как размерность массива (вектора) структур Human. Затем описаны массивы типа string для имени, фамилии, отчества и вектор структур Human – result, который будет хранить данные «людей» и возвращать этот массив в функцию main(). В цикле for производится присваивание значений случайным образом с помощью функции rand().

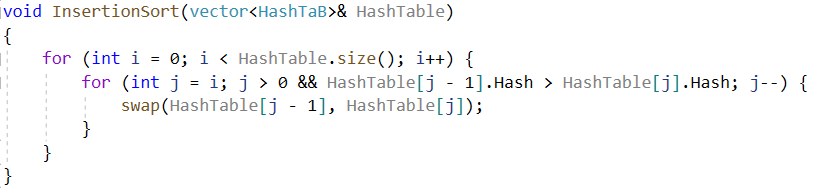


**5.2.** В функции MakeHashTab() передаётся переменная kol как размерность массива (вектора) структур Human, вектор spisok и вектор Table. Затем в цикле for с помощью переменной tmp типа HashTaB запоминается вычисленный хеш для ключа и сам ключ через соответствующие методы (key, Hash). С помощью функции push\_back() заносим заполненный элемент tmp в хеш-таблицу Table.



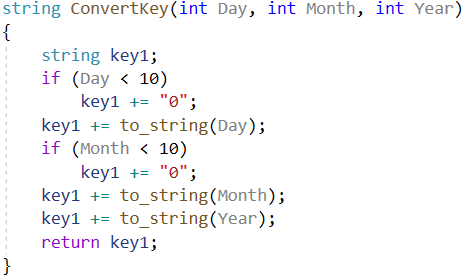
6

Далее использовать функцию InsertionSort() для сортировки хеш-таблицы методом вставки.



Затем в следующем цикле for идёт преобразование коллизий: если следующий элемент имеет тот же хеш – произошла коллизия и поэтому данный (следующий) элемент через функцию push\_back() присваивается к предыдущему (метод устранения коллизий – это метод цепочек) и этот элемент удаляется из изначального вектора Table через метод erase().

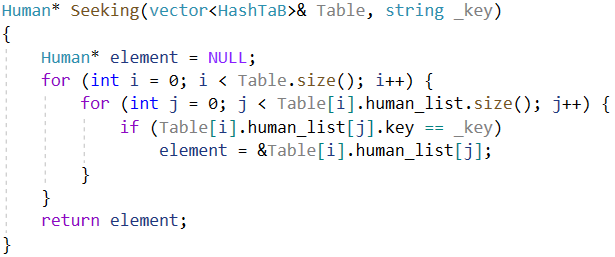
**5.3.** В функции ConvertKey() необходима для преобразования введённого пользователем ключа в строковый тип данных для последующей обработки, так как в хеш-таблице ключ хранится в соответствующем типе.



Если в переменной Day хранится значение меньшее 10, то в ключ сначала добавляется 0, а затем через функцию to\_string() преобразуется int значение переменной Day и так далее, пока дата не будет преобразована к общему стандарту.

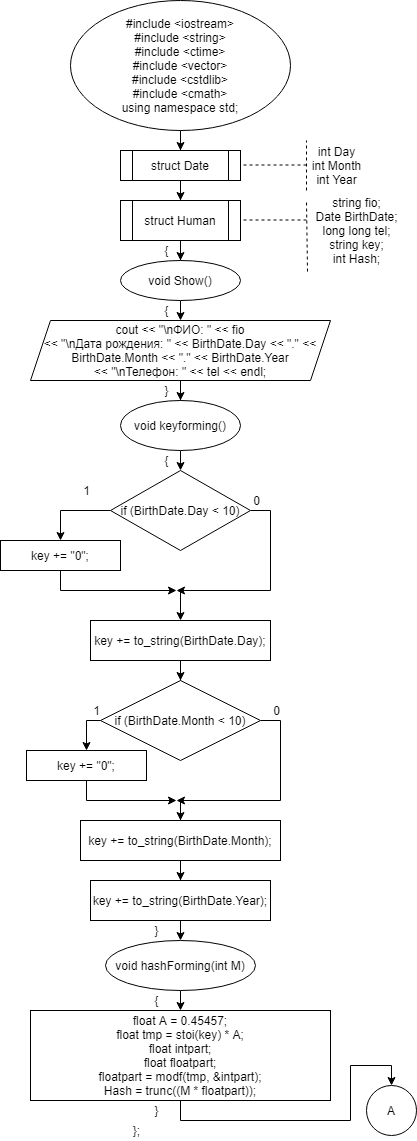
**5.4.** В функции Seeking() типа Human\* передаётся хеш-таблица и сам ключ, который по которому будет вестись поиск. Поиск происходит линейным методом в двух циклах for, где сравнивается элемент из хеш-таблицы через метод key и сам ключ. Если элемент был найден, то он возвращается в функцию main() через указатель element и выводится на экран через метод Show структуры Human. Если элемент не был найден, то возвращается NULL.

7

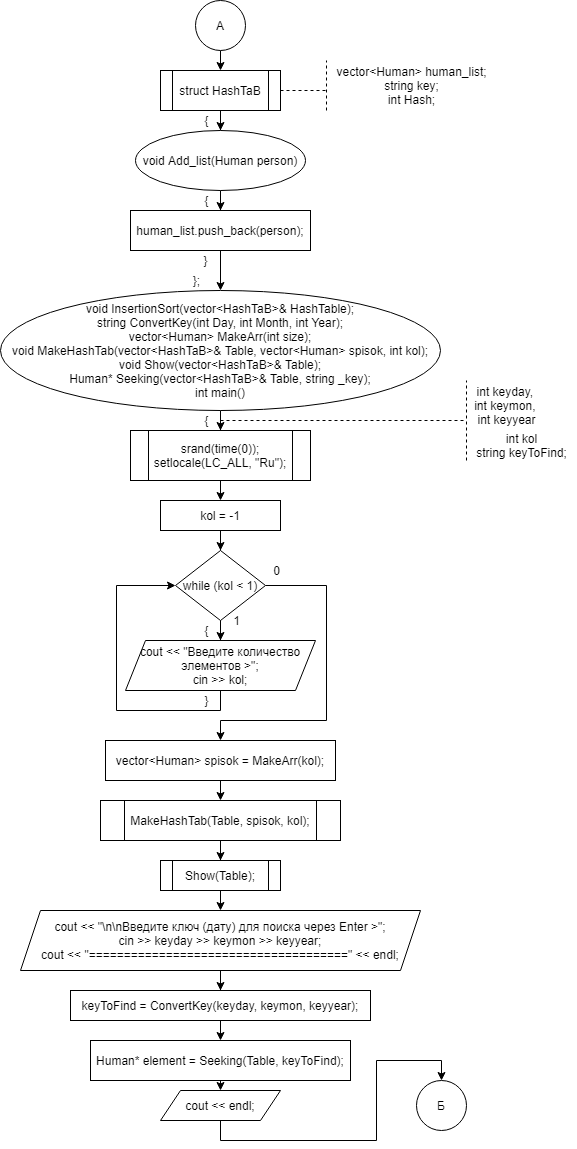


8

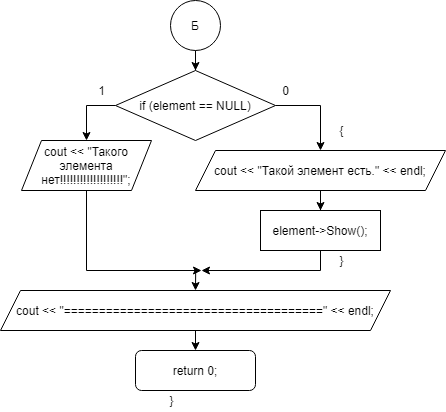
**Блок-схема**



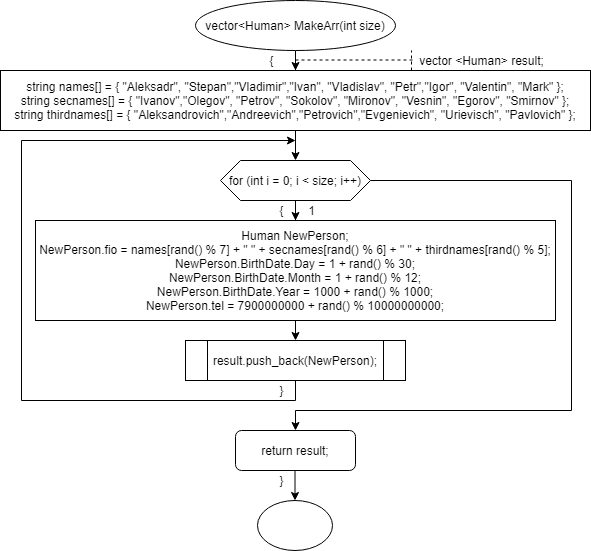
9



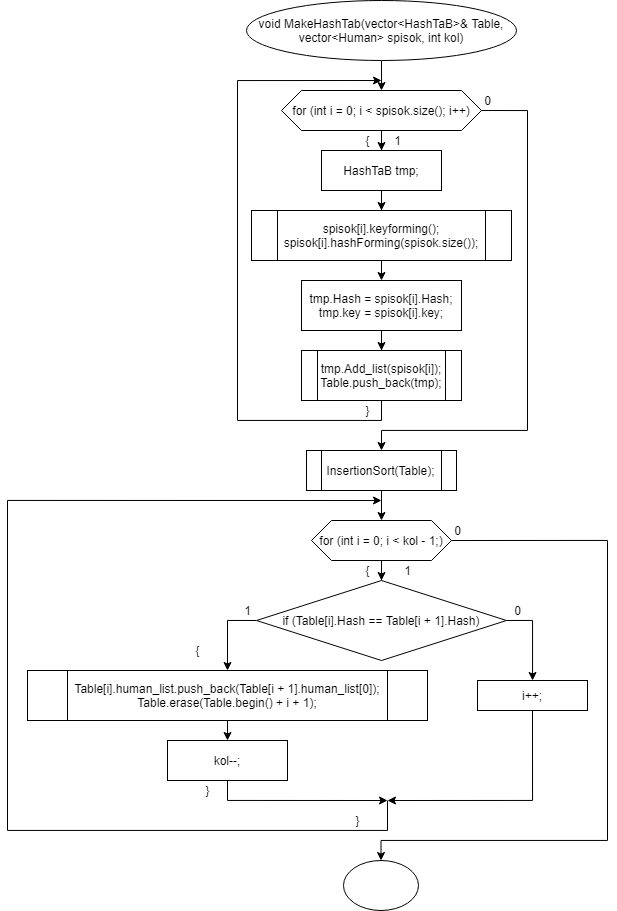
10



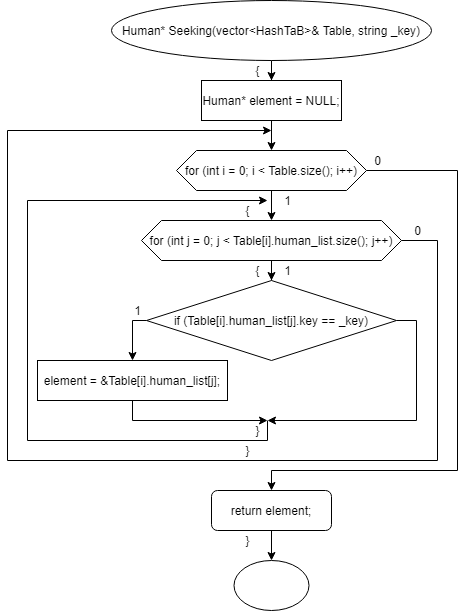
11



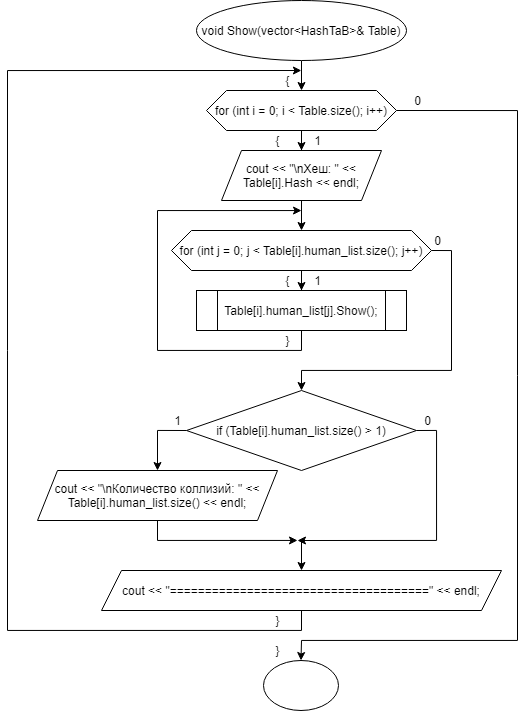
12



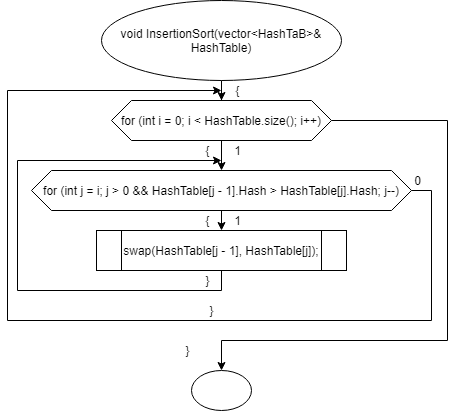
13



14



15



16

**Код**

#include <iostream>

#include <string>

#include <ctime>

#include <vector>

#include <cstdlib>

#include <cmath>

using namespace std;

struct Date {

int Day, Month, Year;

};

struct Human {

string fio;

Date BirthDate;

long long tel;

void Show()

{

cout << "\nФИО: " << fio

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

<< "\nТелефон: " << tel << endl;

}

string key;

void keyforming()

{

if (BirthDate.Day < 10)

key += "0";

key += to\_string(BirthDate.Day);

if (BirthDate.Month < 10)

key += "0";

key += to\_string(BirthDate.Month);

key += to\_string(BirthDate.Year);

}

int Hash;

void hashForming(int M)

{

float A = 0.45457;

float tmp = stoi(key) \* A;

float intpart;

float floatpart;

floatpart = modf(tmp, &intpart);

Hash = trunc((M \* floatpart));

}

};

struct HashTaB {

vector<Human> human\_list;

string key;

int Hash;

void Add\_list(Human person)

{

human\_list.push\_back(person);

}

};

void InsertionSort(vector<HashTaB>& HashTable);

string ConvertKey(int Day, int Month, int Year);

vector<Human> MakeArr(int size);

void MakeHashTab(vector<HashTaB>& Table, vector<Human> spisok, int kol);

void Show(vector<HashTaB>& Table);

Human\* Seeking(vector<HashTaB>& Table, string \_key);

int main()

{

17

srand(time(0));

setlocale(LC\_ALL, "Ru");

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

while (kol < 1) {

cout << "Введите количество элементов >";

cin >> kol;

}

vector<Human> spisok = MakeArr(kol);

vector<HashTaB>Table;

MakeHashTab(Table, spisok, kol);

Show(Table);

string keyToFind;

cout << "\n\nВведите ключ (дату) для поиска через Enter >";

cin >> keyday >> keymon >> keyyear;

cout << "=====================================" << endl;

keyToFind = ConvertKey(keyday, keymon, keyyear);

Human\* element = Seeking(Table, keyToFind);

cout << endl;

if (element == NULL) {

cout << "Такого элемента нет!!!!!!!!!!!!!!!!!!!";

}

else {

cout << "Такой элемент есть." << endl;

element->Show();

}

cout << "=====================================" << endl;

}

vector<Human> MakeArr(int size)

{

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" };

vector <Human> result;

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

Human NewPerson;

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

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

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

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

NewPerson.tel = 7900000000 + rand() % 10000000000;

result.push\_back(NewPerson);

}

return result;

}

void MakeHashTab(vector<HashTaB>& Table, vector<Human> spisok, int kol)

{

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

HashTaB tmp;

spisok[i].keyforming();

spisok[i].hashForming(spisok.size());

tmp.Hash = spisok[i].Hash;

tmp.key = spisok[i].key;

tmp.Add\_list(spisok[i]);

Table.push\_back(tmp);

}

InsertionSort(Table);

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

if (Table[i].Hash == Table[i + 1].Hash) {

18

Table[i].human\_list.push\_back(Table[i + 1].human\_list[0]);

Table.erase(Table.begin() + i + 1);

kol--;

}

else

i++;

}

}

string ConvertKey(int Day, int Month, int Year)

{

string key1;

if (Day < 10)

key1 += "0";

key1 += to\_string(Day);

if (Month < 10)

key1 += "0";

key1 += to\_string(Month);

key1 += to\_string(Year);

return key1;

}

void InsertionSort(vector<HashTaB>& HashTable)

{

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

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

swap(HashTable[j - 1], HashTable[j]);

}

}

}

void Show(vector<HashTaB>& Table)

{

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

cout << "\nХеш: " << Table[i].Hash << endl;

for (int j = 0; j < Table[i].human\_list.size(); j++) {

Table[i].human\_list[j].Show();

}

if (Table[i].human\_list.size() > 1)

cout << "\nКоличество коллизий: " << Table[i].human\_list.size() << endl;

cout << "=====================================" << endl;

}

}

Human\* Seeking(vector<HashTaB>& Table, string \_key)

{

Human\* element = NULL;

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

for (int j = 0; j < Table[i].human\_list.size(); j++) {

if (Table[i].human\_list[j].key == \_key)

element = &Table[i].human\_list[j];

}

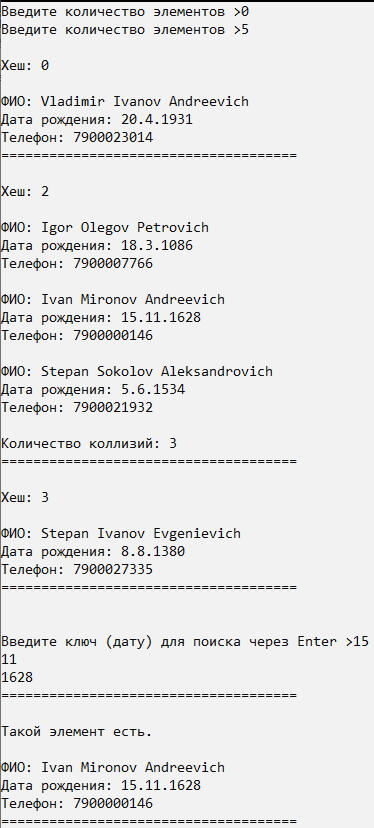
}

return element;

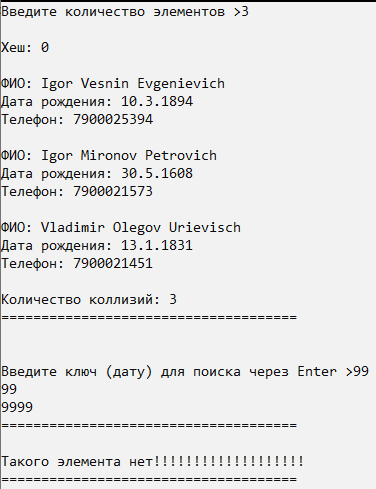
}

19

**Скриншоты**



20



21