Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное

учреждение

высшего образования

**«Пермский национальный исследовательский**

**политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЁТ**

**по лабораторной работе №17**

Дисциплина: «Основы теории алгоритмов и структуры данных»

Тема: Методы быстрого поиска. Хеширование.

Вариант 21

Выполнил:

Студент группы ИВТ-20-2б

Галинов О.Ю.

Проверила:

Доцент кафедры ИТАС

Полякова О. А.

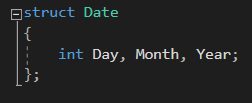
Пермь, 2021

**Постановка задачи**

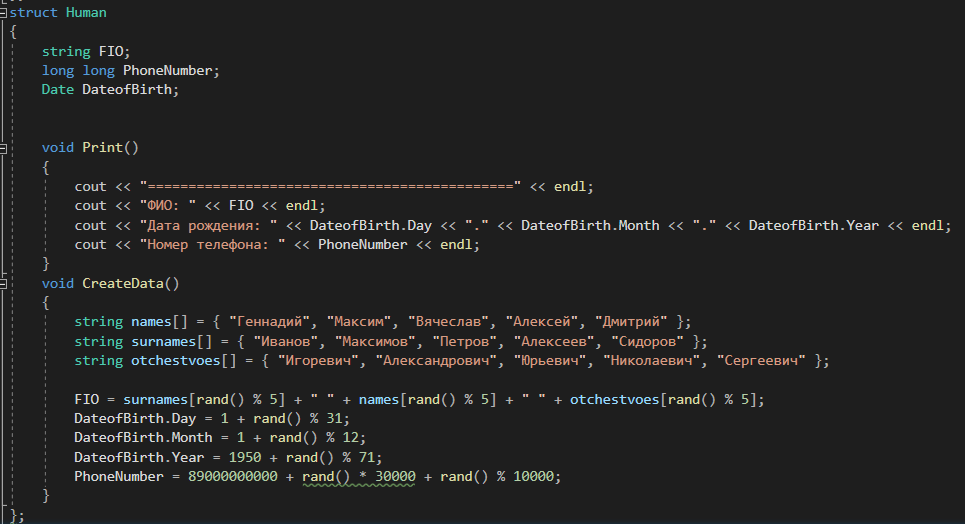
Создать динамический массив. Выполнить поиск элемента по ключу с помощью хеш-таблицы. Метод устранения коллизий – метод открытой адресации.

**Анализ задачи**

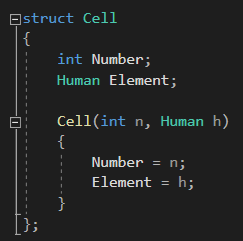
1. В программе используются типы:
   1. Int
   2. Date
   3. Human
   4. String
   5. Long long
   6. String[]
   7. Vector<Human>
   8. Double
   9. Cell
   10. Bool
   11. Vector<Cell>
2. Действия над данными
   1. Int, long long, double – арифметические операции
   2. Vector<> и []- добавление, удаление и изменение элементов контейнера.
3. Виды данных в программе
   1. Vector<> - динамический одномерный массив
   2. String[] – статический одномерный массив типа string
4. Структуры
   1. Date – используется для хранения даты.



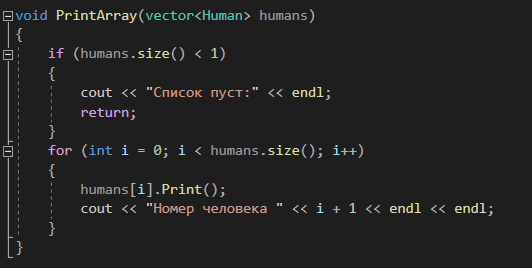
* 1. Human – используется для хранения сведений об одном человеке. Имеет поля FIO, DateOfBirth, PassportNumber. Описаны методы для печати сведений и генерировании случайной информации.



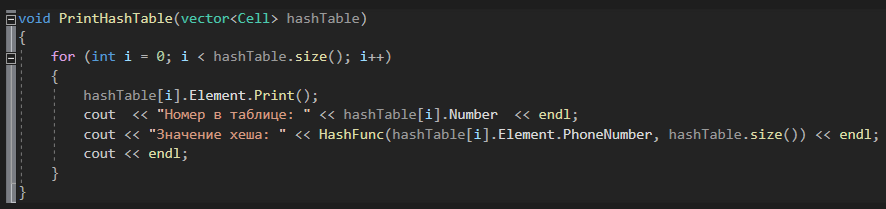
* 1. Cell – одна ячейка хеш-таблицы. Имеет поля Number и Element, а также метод – конструктор.



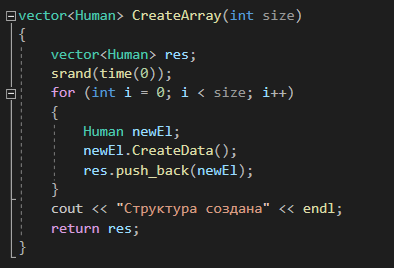
1. Организация ввода-вывода
   1. Печать вектора. Присутствует проверка на размер.



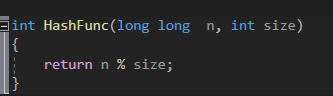
* 1. Печать таблицы хэшей.



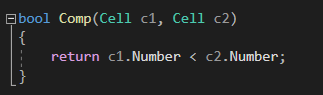
1. Действия для решения задачи
   1. Создание вектора. Все элементы вектора генерируются случайным образом.



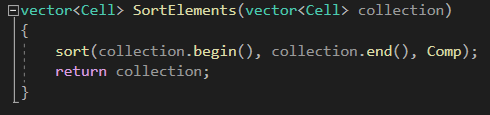
* 1. Хеш-функция.



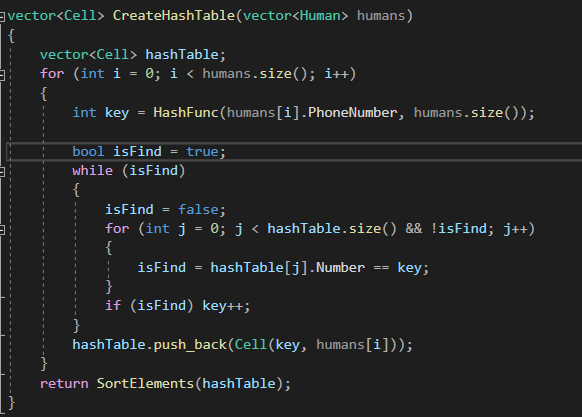
* 1. Метод сравнения для вызова в методе сортировки.



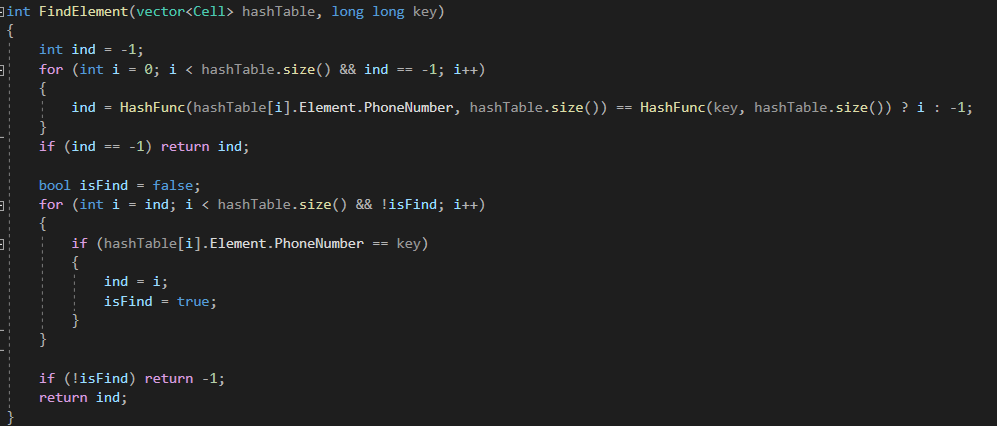
* 1. Метод сортировки.



* 1. Создание хеш-таблицы. На вход подается вектор с типом Human. В качестве ключевого поля используется поле PhoneNumber.



* 1. Поиск элемента в хеш-таблице. Так как элементы в таблице отсортированы по ключу (точнее по номеру, так как коллизий много), то сначала поиск осуществляется до указанного номера. То есть цикл работает, пока не найдет номер в таблице, соответствующий значению хеш-функции от ключа. Если такого номера нет, то функция сразу вернет -1. Затем элементы начинают сравниваться уже каждый по отдельности соответствующими полями.



**Код программы**

#include <iostream>

#include <vector>

#include <string>

#include <list>

#include <ctime>

#include <cmath>

#include<algorithm>

using namespace std;

struct Date

{

int Day, Month, Year;

};

struct Human

{

string FIO;

long long PhoneNumber;

Date DateofBirth;

void Print()

{

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

cout << "ФИО: " << FIO << endl;

cout << "Дата рождения: " << DateofBirth.Day << "." << DateofBirth.Month << "." << DateofBirth.Year << endl;

cout << "Номер телефона: " << PhoneNumber << endl;

}

void CreateData()

{

string names[] = { "Геннадий", "Максим", "Вячеслав", "Алексей", "Дмитрий" };

string surnames[] = { "Иванов", "Максимов", "Петров", "Алексеев", "Сидоров" };

string otchestvoes[] = { "Игоревич", "Александрович", "Юрьевич", "Николаевич", "Сергеевич" };

FIO = surnames[rand() % 5] + " " + names[rand() % 5] + " " + otchestvoes[rand() % 5];

DateofBirth.Day = 1 + rand() % 31;

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

DateofBirth.Year = 1950 + rand() % 71;

PhoneNumber = 89000000000 + rand() \* 30000 + rand() % 10000;

}

};

vector<Human> CreateArray(int size)

{

vector<Human> res;

srand(time(0));

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

{

Human newEl;

newEl.CreateData();

res.push\_back(newEl);

}

cout << "Структура создана" << endl;

return res;

}

void PrintArray(vector<Human> humans)

{

if (humans.size() < 1)

{

cout << "Список пуст:" << endl;

return;

}

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

{

humans[i].Print();

cout << "Номер человека " << i + 1 << endl << endl;

}

}

int GetSize()

{

int n;

cout << "Введите количество: ";

cin >> n;

while (n < 1)

{

cout << "Введите число корректно: ";

cin >> n;

}

return n;

}

int HashFunc(long long n, int size)

{

return n % size;

}

struct Cell

{

int Number;

Human Element;

Cell(int n, Human h)

{

Number = n;

Element = h;

}

};

bool Comp(Cell c1, Cell c2)

{

return c1.Number < c2.Number;

}

vector<Cell> SortElements(vector<Cell> collection)

{

sort(collection.begin(), collection.end(), Comp);

return collection;

}

vector<Cell> CreateHashTable(vector<Human> humans)

{

vector<Cell> hashTable;

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

{

int key = HashFunc(humans[i].PhoneNumber, humans.size());

bool isFind = true;

while (isFind)

{

isFind = false;

for (int j = 0; j < hashTable.size() && !isFind; j++)

{

isFind = hashTable[j].Number == key;

}

if (isFind) key++;

}

hashTable.push\_back(Cell(key, humans[i]));

}

return SortElements(hashTable);

}

void PrintHashTable(vector<Cell> hashTable)

{

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

{

hashTable[i].Element.Print();

cout << "Номер в таблице: " << hashTable[i].Number << endl;

cout << "Значение хеша: " << HashFunc(hashTable[i].Element.PhoneNumber, hashTable.size()) << endl;

cout << endl;

}

}

int FindElement(vector<Cell> hashTable, long long key)

{

int ind = -1;

for (int i = 0; i < hashTable.size() && ind == -1; i++)

{

ind = HashFunc(hashTable[i].Element.PhoneNumber, hashTable.size()) == HashFunc(key, hashTable.size()) ? i : -1;

}

if (ind == -1) return ind;

bool isFind = false;

for (int i = ind; i < hashTable.size() && !isFind; i++)

{

if (hashTable[i].Element.PhoneNumber == key)

{

ind = i;

isFind = true;

}

}

if (!isFind) return -1;

return ind;

}

int main()

{

setlocale(LC\_ALL, "rus");

int size;

size = GetSize();

; vector<Human>humans;

humans = CreateArray(size);

vector<Cell> hashTable = CreateHashTable(humans);

PrintHashTable(hashTable);

long long key;

cout << endl << "Введите номер телефона для поиска: ";

cin >> key;

int ind = FindElement(hashTable, key);

if (ind != -1)

{

cout << "Элемент найден" << endl;

hashTable[ind].Element.Print();

}

else

{

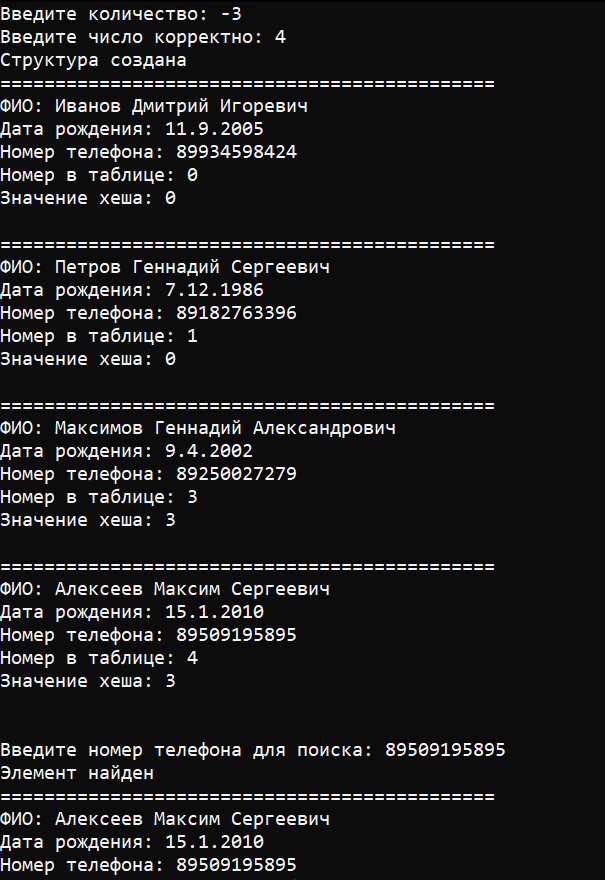
cout << "Элемент не найден" << endl;

}

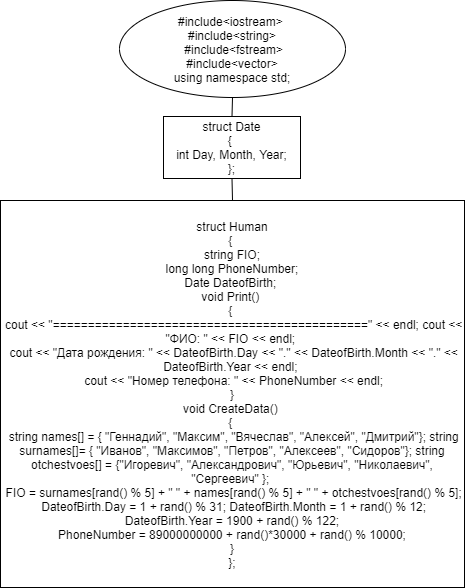
system("pause");

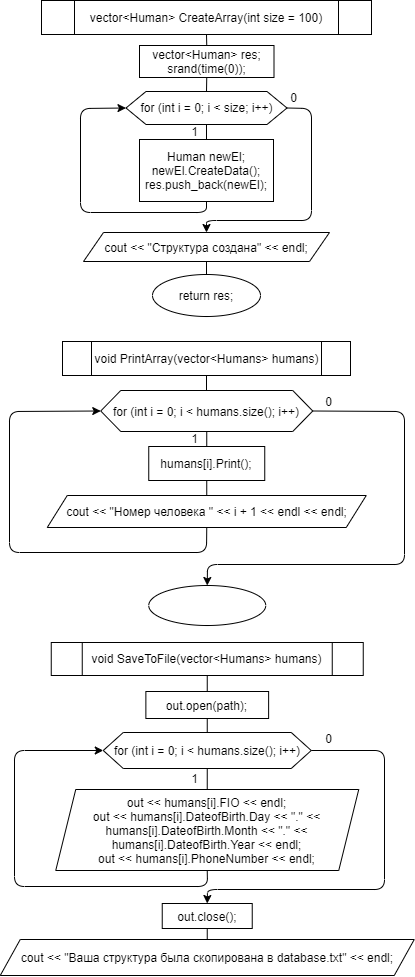
}

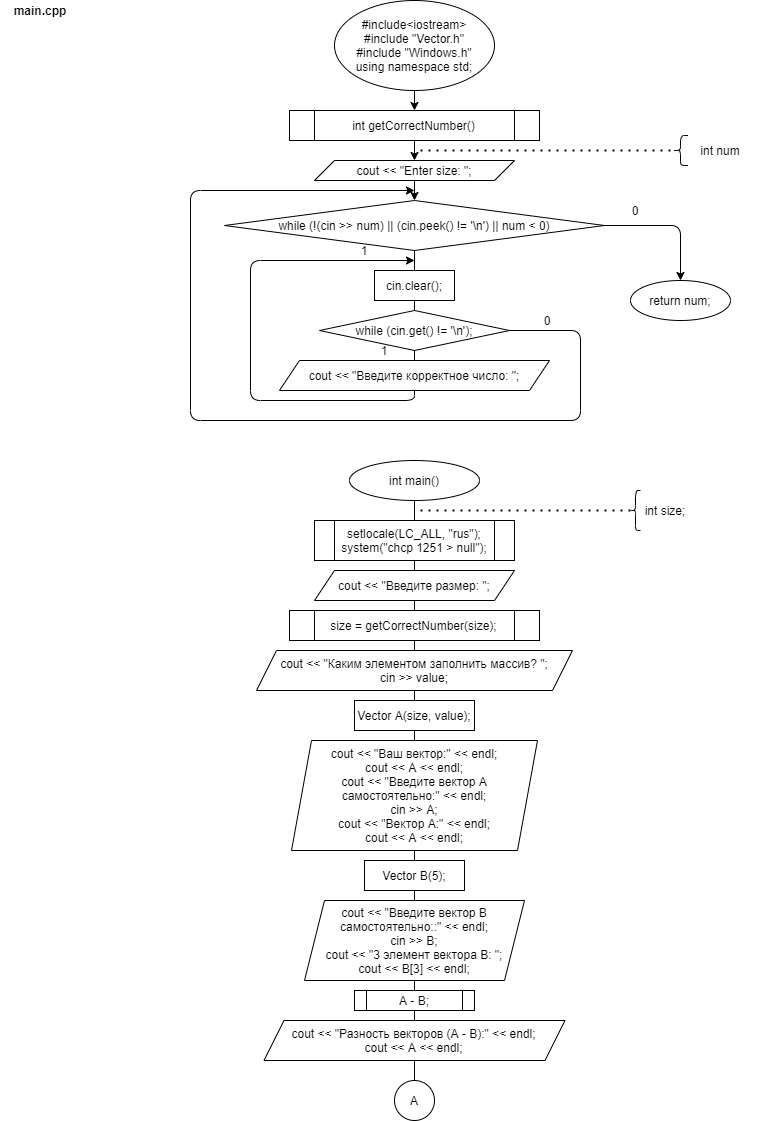
**Скриншоты выполнения**

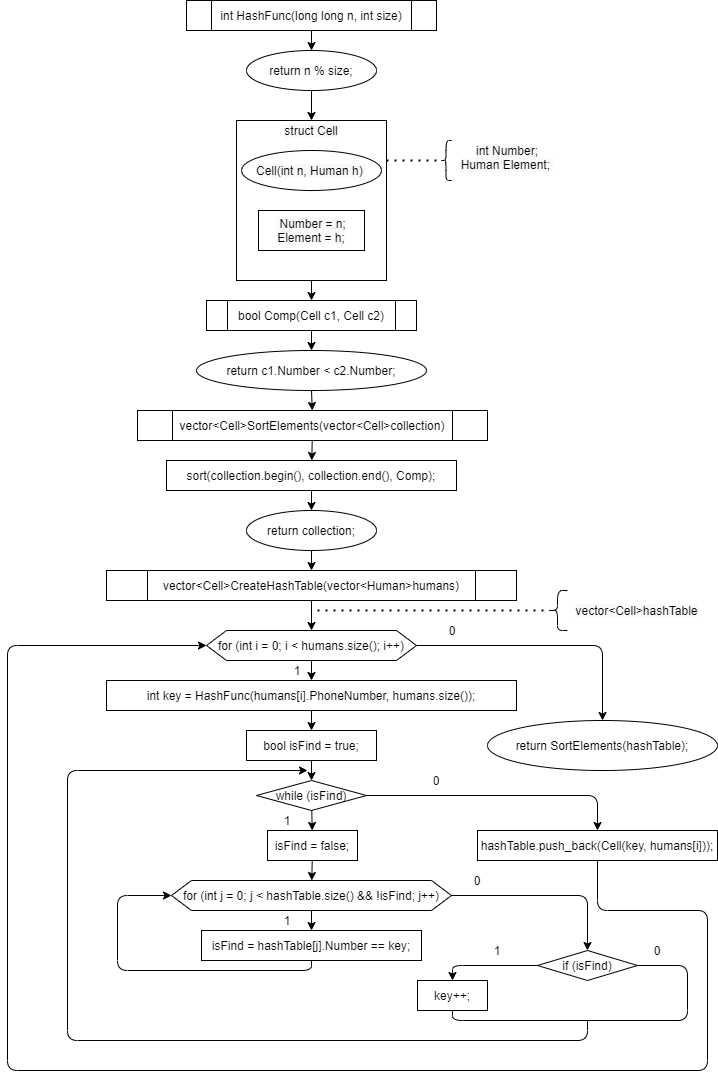
****

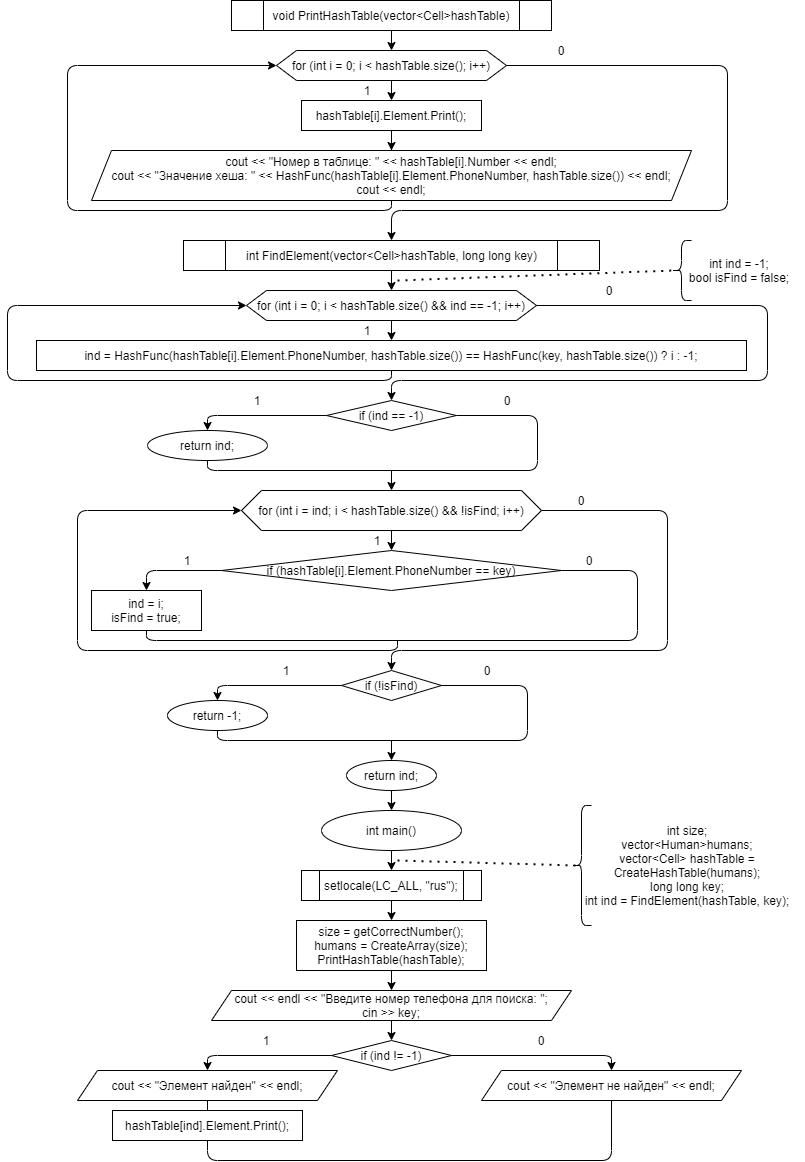
**Блок-схема**

****



****

****

****