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 17

Выполнил:

Студент группы РИС-20-1б

Томилов Владислав Алексеевич

Проверила:

Доцент кафедры ИТАС

Полякова О. А.

**Пермь, 2021**

**Цель работы**

1. Изучить построение функции хеширования и алгоритмов хеширования данных.

2. Научиться разрабатывать алгоритмы открытого и закрытого хеширования при решении задач на языке C+

**Постановка задачи**

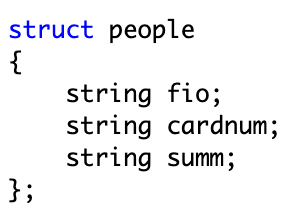
1. Создать динамический массив из записей (в соответствии с вариантом), содержащий не менее 100 элементов. Для заполнения элементов массива использовать ДСЧ.

2. Выполнить поиск элемента в массиве по ключу в соответствии с вариантом. Для поиска использовать хеш-таблицу.

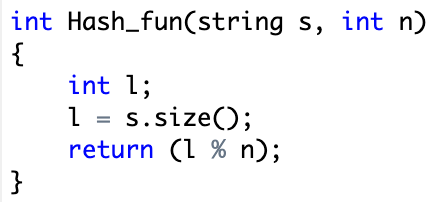
3. Подсчитать количество коллизий при размере хеш-таблицы 40, 75 и 90 элементов.

**Анализ задачи**

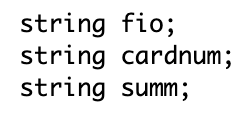
**1.** Для решения задачи необходимо:

**1.1.** Создать структуру для реализации алгоритма хеширования данных.

.

**1.2.** Разработать функцию хеширования Hash\_fun в соответствии с заданным вариантом.

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

**2.1.** Структура имеет информационные поля типа string.

**2.2.** Для функции Hash\_fun, используются следующие аргументы:

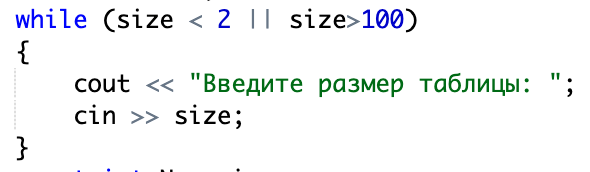
****

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

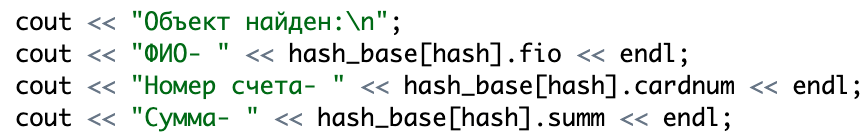
**3.1.** Для заполнения массива структур используется заполнение значениями из файла.

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

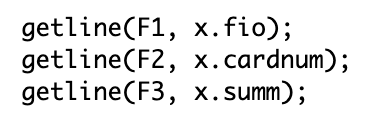
**4.1.** Ввод данных реализован с помощью оператора cin.



**4.2.** Вывод данных на консоль реализован с помощью оператора cout.

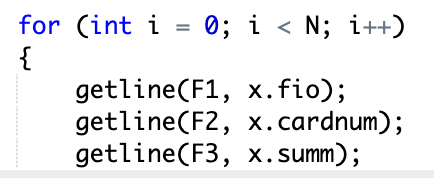


**4.3.** Ввод данных из файла реализован с помощью оператора getline.

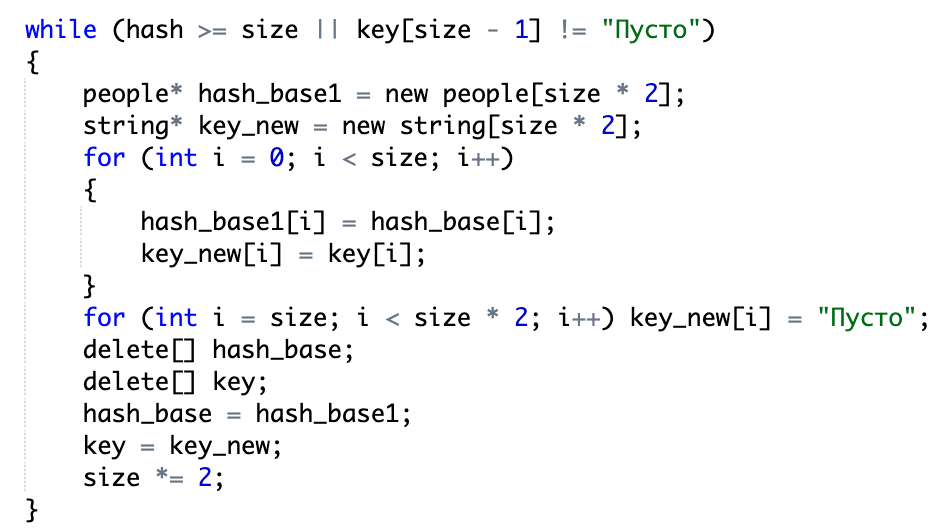
****

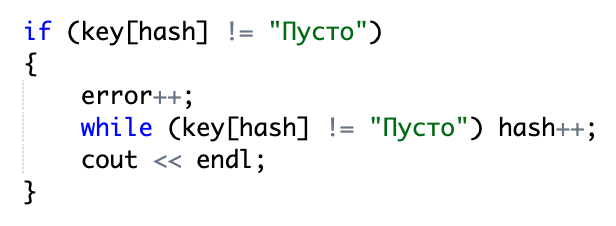
**5.** Поставленные задачи будут решены следующими действиями:

**5.1.** Для заполнения массива используется цикл for:

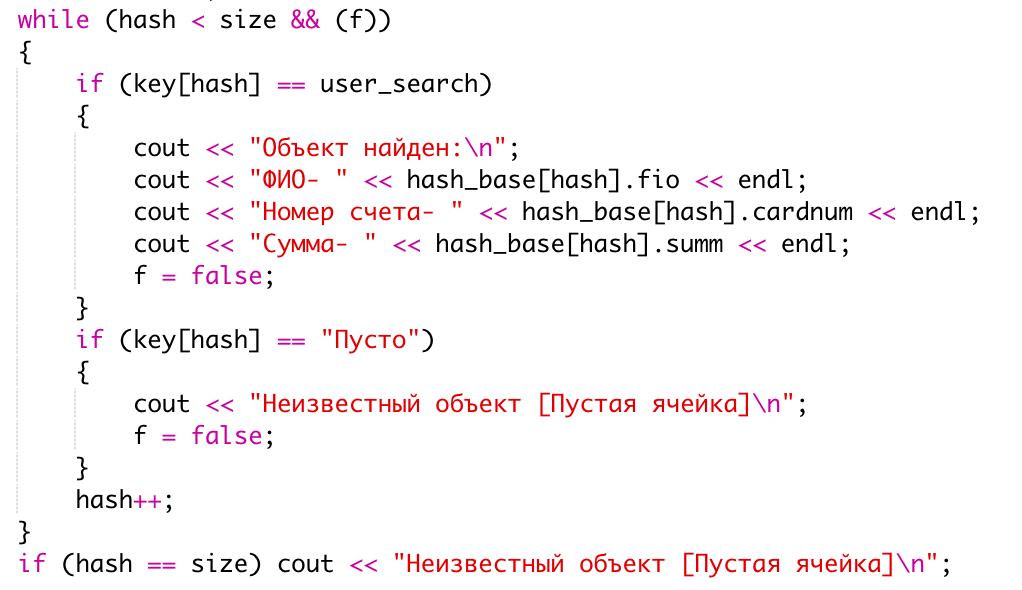


**5.2** Для создание хеш-таблицы нужного размера используется цикл while.

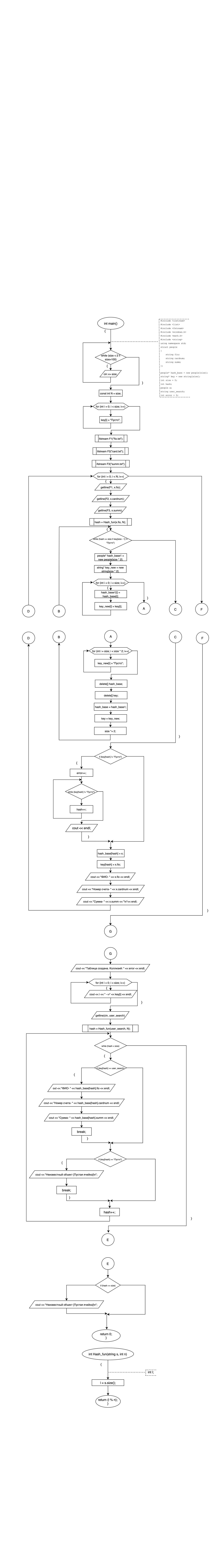


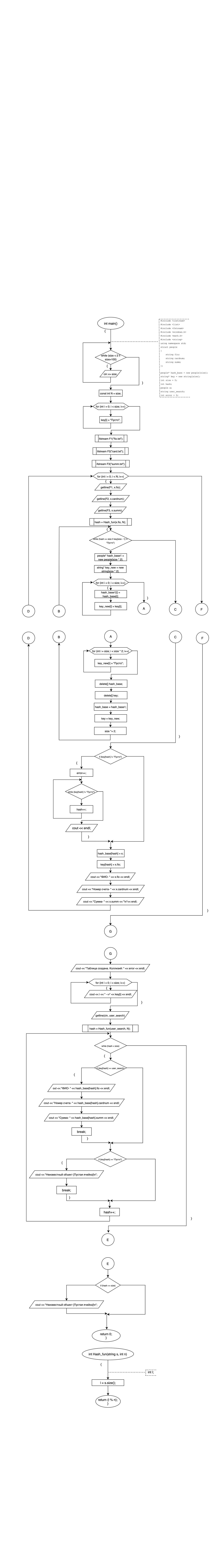
** 5.3** Способ исключения коллизий реализован методом открытой адресации

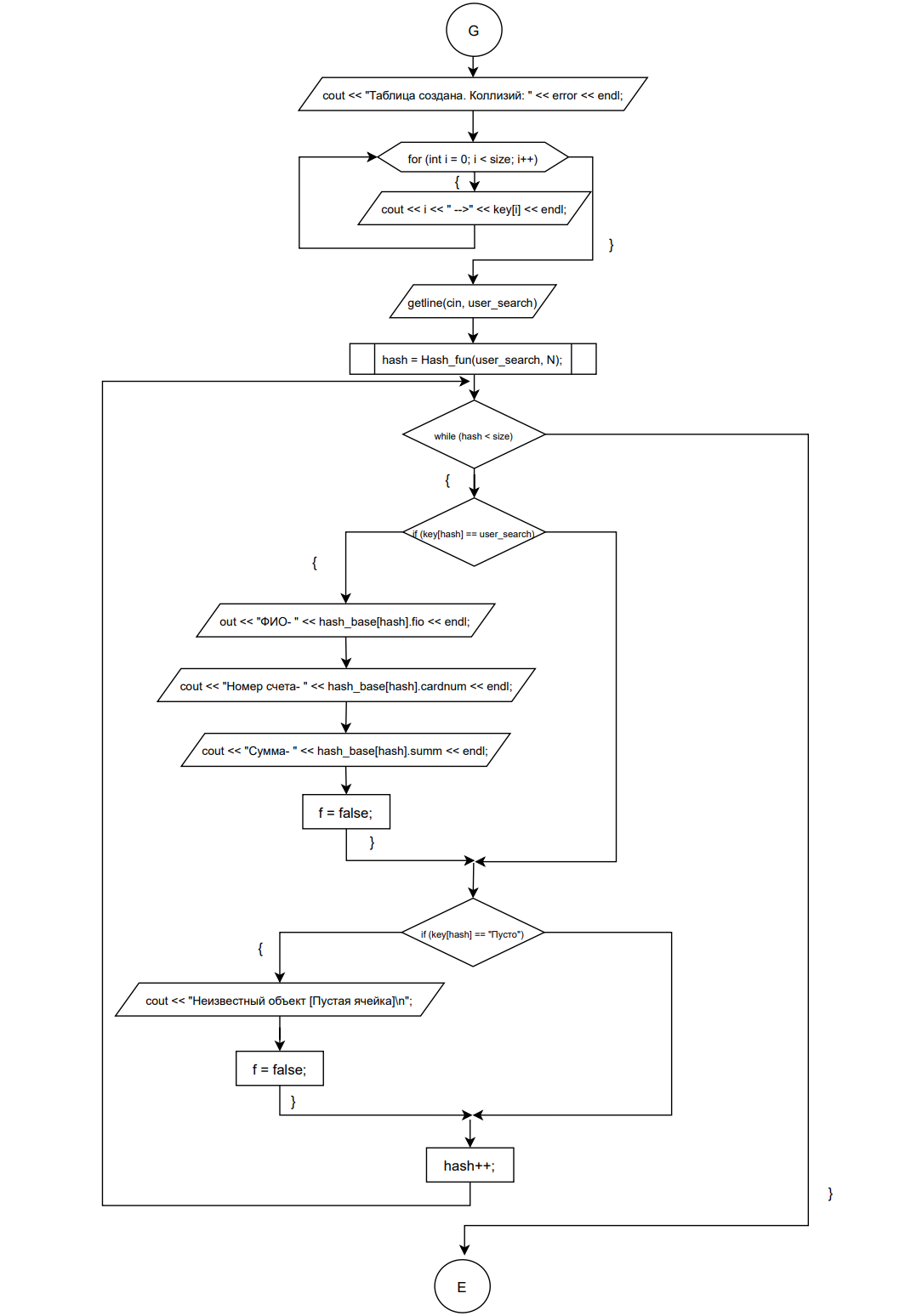
**5.4** Поиск значений в хеш-таблице реализован с помощью цикла while и вложенных циклов if.

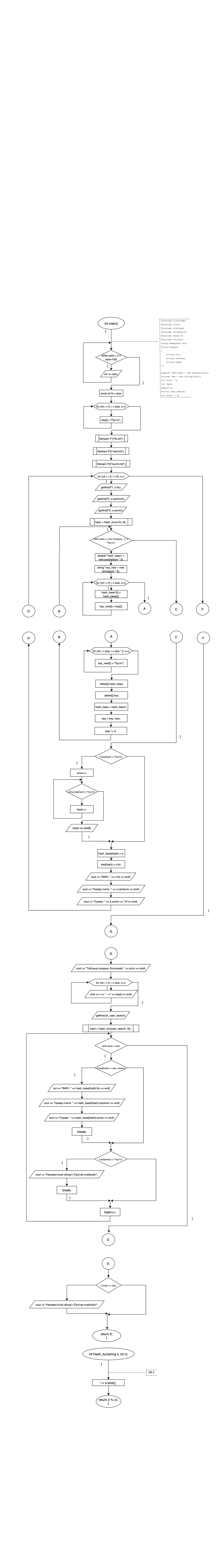
****

**Блок-схема**









**Решение**

#include <iostream>

#include <list>

#include <fstream>

#include <windows.h>

#include <math.h>

#include <string>

using namespace std;

struct people

{

string fio;

string cardnum;

string summ;

};

int Hash\_fun(string s, int n)

{

int l;

l = s.size();

return (l % n);

}

int main()

{

SetConsoleOutputCP(1251);

SetConsoleCP(1251);

int size = 0;

int hash;

people x;

string user\_search;

int error = 0;

while (size < 2 || size>100)

{

cout << "Введите размер таблицы: ";

cin >> size;

}

const int N = size;

cin.get();

people\* hash\_base = new people[size];

string\* key = new string[size];

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

key[i] = "Пусто";

ifstream F1("fio.txt");

ifstream F2("card.txt");

ifstream F3("summ.txt");

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

{

getline(F1, x.fio);

getline(F2, x.cardnum);

getline(F3, x.summ);

hash = Hash\_fun(x.fio, N);

while (hash >= size || key[size - 1] != "Пусто")

{

people\* hash\_base1 = new people[size \* 2];

string\* key\_new = new string[size \* 2];

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

{

hash\_base1[i] = hash\_base[i];

key\_new[i] = key[i];

}

for (int i = size; i < size \* 2; i++) key\_new[i] = "Пусто";

delete[] hash\_base;

delete[] key;

hash\_base = hash\_base1;

key = key\_new;

size \*= 2;

}

if (key[hash] != "Пусто")

{

error++;

while (key[hash] != "Пусто") hash++;

cout << endl;

}

hash\_base[hash] = x;

key[hash] = x.fio;

cout << "ФИО- " << x.fio << endl;

cout << "Номер счета- " << x.cardnum << endl;

cout << "Сумма- " << x.summ << "\n"<< endl;

}

cout << "Таблица создана. Коллизий: " << error << endl;

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

{

cout << i << " -->" << key[i] << endl;

}

cout << "Введите запрос\n";

getline(cin, user\_search);

hash = Hash\_fun(user\_search, N);

bool f = true;

while (hash < size && (f))

{

if (key[hash] == user\_search)

{

cout << "Объект найден:\n";

cout << "ФИО- " << hash\_base[hash].fio << endl;

cout << "Номер счета- " << hash\_base[hash].cardnum << endl;

cout << "Сумма- " << hash\_base[hash].summ << endl;

f = false;

}

if (key[hash] == "Пусто")

{

cout << "Неизвестный объект [Пустая ячейка]\n";

f = false;

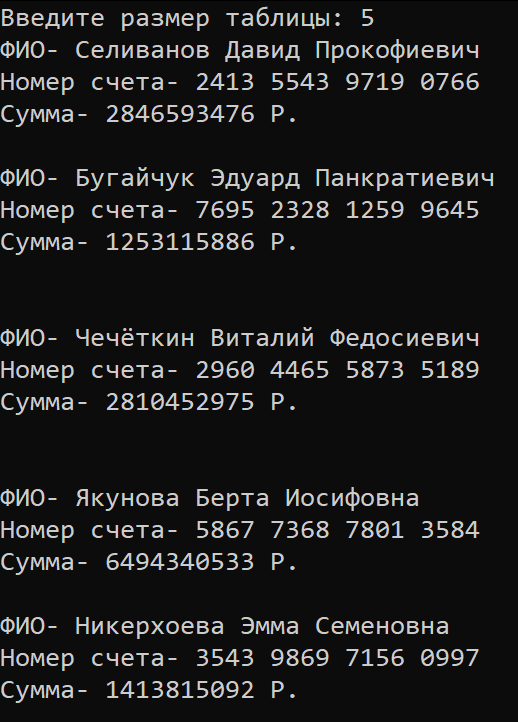
}

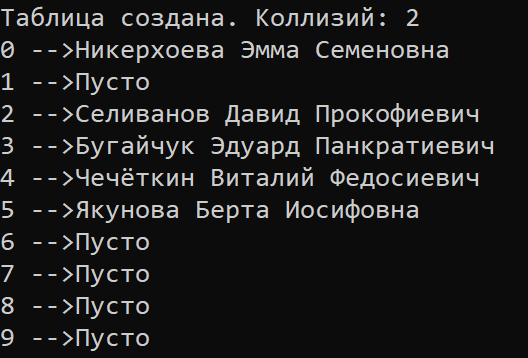
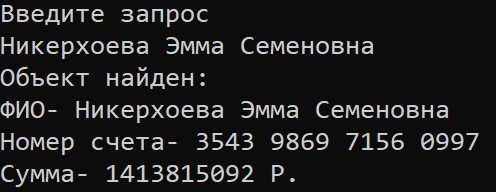
hash++;

}

if (hash == size) cout << "Неизвестный объект [Пустая ячейка]\n";

}

**Скриншоты результатов работы программы**

****