Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

**«Пермский национальный исследовательский**

**политехнический университет»**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

направление подготовки: 09.03.01 - «Информатика и вычислительная техника»

**О Т Ч Е Т**

**по лабораторной работе Хэш функции**

**по дисциплине**

**«Основы алгоритмизации и программирования» семестр 2**

Выполнил студент гр. ИВТ-21-1б

Ипатов Дмитрий Сергеевич

Проверил:

Ст. Преподаватель кафедры ИТАС

Яруллин Д.В.

(оценка) (подпись)

г. Пермь-2022

**Постановка задачи:**

1. Создать динамический массив из записей (в соответствии с вариантом),

содержащий не менее 100 элементов. Для заполнения элементов массива

использовать ДСЧ.

1. Выполнить поиск элемента в массиве по ключу в соответствии с вариантом. Для поиска использовать хеш-таблицу.
2. Подсчитать количество коллизий при размере хеш-таблицы 40, 75 и 90

элементов.

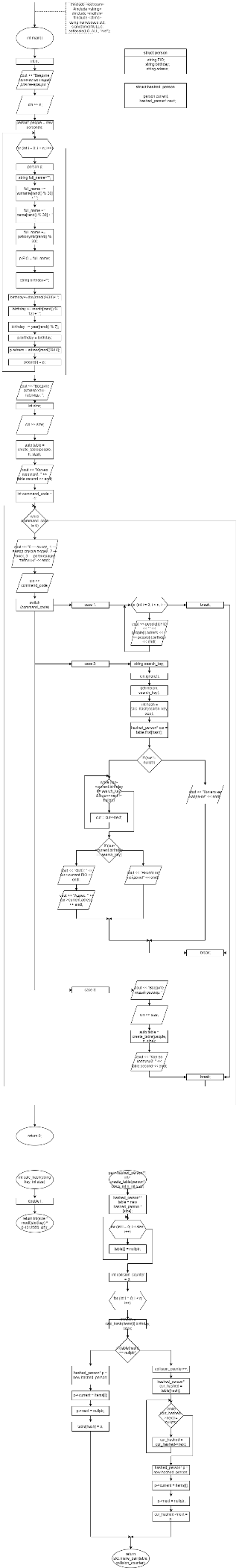
Анализ задачи:

Инициализируем глобальные массивы строкового типа. Добавляем прототипы функций.

Создаем людей по параметрам из массивов при помощи отдельных функций для каждого параметра. Считаем индекс методом открытой адресации также в отдельной функции. Добавляем элементы в хэш-таблицу через отдельную функцию также. Вывод на экран осуществляется аналогично-через отдельные функции.

Не забываем про структуры, где указываем параметры человека. Основные функции вызываются из главной. Считается количество коллизий, для этого инициализирован счетчик как глобальная переменная

**Блок схема:**

****

**Код программы:**

#include <iostream>

#include <string>

#include <math.h>

#include <ctime>

using namespace std;

string name[30] = { "Иван", "Андрей", "Антон", "Амин", "Роберт", "Георгий", "Борис", "Владислав", "Дмитрий", "Никита", "Данила", "Артем", "Парвиз", "Сергей", "Илья", "Григорий", "Артур", "Денис", "Алексей", "Александр", "Валерий", "Дамир", "Михаил", "Константин", "Николай", "Герман", "Иннокентий", "Богдан", "Станислав", "Сурман" };

string patronymic[30] = { "Иванович", "Андреевич", "Антонович", "Аминович", "Робертович", "Георгиевич", "Борисович", "Владиславович", "Дмитриевич", "Никитович","Данилович", "Артемович", "Парвизович", "Сергеевич", "Ильич", "Григорьевич", "Артурович", "Денисович", "Алексеевич", "Александрович", "Валерьевич", "Дамирович", "Михайлович", "Константинович", "Николаевич", "Германович", "Иннокентьевич", "Богданович", "Станиславович", "Сурманович" };

string surname[30] = { "Карпов", "Афанасьев", "Власов","Маслов","Исаков","Тихонов","Аксёнов","Гаврилов","Родионов","Котов","Горбунов","Кудряшов","Быков","Зуев","Третьяков","Савельев","Панов","Рыбаков","Суворов","Абрамов","Воронов","Мухин","Архипов","Трофимов","Мартынов","Емельянов","Горшков","Чернов","Овчинников","Селезнёв" };

string adress[15] = { "Пушкина 17", "Фрунзе 10", "Екатерининская 168", "Льва Толстого 8", "Чкалова 9", "Ленина 103", "Профессора Дедюкина 18", "Братиславская 30", "Первомайская 35", "Октябрьская 5", "Корнилова 10", "Пермская 3", "Лужников 4", "Петропавловская 107", "Краснова 4" };

string day[31] = {"01","02", "03", "04", "05", "06", "07", "08", "09", "10", "11", "12", "13", "14", "15", "16", "17", "18", "19", "20", "21", "22", "23", "24", "25", "26", "27", "28", "29", "30", "31"};

string month[12] = {"01","02", "03", "04", "05", "06", "07", "08", "09", "10", "11", "12"};

string year[7] = {"1997","1998", "1999", "2000", "2001", "2002", "2003"};

struct person

{

string FIO;

string birthday;

string adress;

};

struct hashed\_person

{

person current;

hashed\_person\* next;

};

int calc\_hash(string key, int size)

{

double f;

return int(size \* modf(stoi(key) \* 0.4312556, &f));

}

pair<hashed\_person\*\*, int> create\_table(person\* items, int n, int size)

{

hashed\_person\*\* table = new hashed\_person \* [size];

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

{

table[i] = nullptr;

}

int collision\_counter = 0;

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

{

int hash = calc\_hash(items[i].birthday, size);

if (table[hash] == nullptr)

{

hashed\_person\* p = new hashed\_person;

p->current = items[i];

p->next = nullptr;

table[hash] = p;

}

else

{

collision\_counter++;

hashed\_person\* cur\_hashed = table[hash];

while (cur\_hashed->next != nullptr)

{

cur\_hashed = cur\_hashed->next;

}

hashed\_person\* p = new hashed\_person;

p->current = items[i];

p->next = nullptr;

cur\_hashed->next = p;

}

}

return std::make\_pair(table, collision\_counter);

}

int main()

{

srand(time(NULL));

setlocale(LC\_ALL, "rus");

int n;

cout << "Введите количество людей для генерации ";

cin >> n;

person\* people = new person[n];

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

{

person p;

string full\_name="";

full\_name += surname[rand() % 30] + ' ';

full\_name += name[rand() % 30] + ' ';

full\_name += patronymic[rand() % 30];

p.FIO = full\_name;

string birthday="";

birthday+=day[rand()%30]+'.';

birthday += month[rand() % 12] + '.';

birthday += year[rand() % 7];

p.birthday = birthday;

p.adress = adress[rand()%14];

people[i] = p;

}

cout << "Введите размер хэш-таблицы: ";

int size;

cin >> size;

auto table = create\_table(people, n, size);

cout << "Кол-во коллизий: " << table.second << endl;

int command\_code = -1;

while (command\_code != 0)

{

cout << "0 — выход, 1 — вывод списка людей, 2 — поиск, 3 — регенерация таблицы" << endl;

cin >> command\_code;

switch (command\_code)

{

case 1: {

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

{

cout << people[i].FIO << ' ' << people[i].adress << ' ' << people[i].birthday << endl;

}

break;

}

case 2: {

string search\_key;

cin.ignore();

getline(cin, search\_key);

int hash = calc\_hash(search\_key, size);

hashed\_person\* cur = table.first[hash];

if (cur != nullptr)

{

while (cur->current.birthday != search\_key && cur->next != nullptr)

{

cur = cur->next;

}

if (cur->current.birthday == search\_key)

{

cout << "ФИО: " << cur->current.FIO << endl;

cout << "Адрес: " << cur->current.adress << endl;

}

else

{

cout << "Ничего не найдено!" << endl;

}

}

else

{

cout << "Ничего не найдено!" << endl;

}

break;

}

case 3: {

cout << "Введите новый размер: ";

cin >> size;

auto table = create\_table(people, n, size);

cout << "Кол-во коллизий: " << table.second << endl;

}

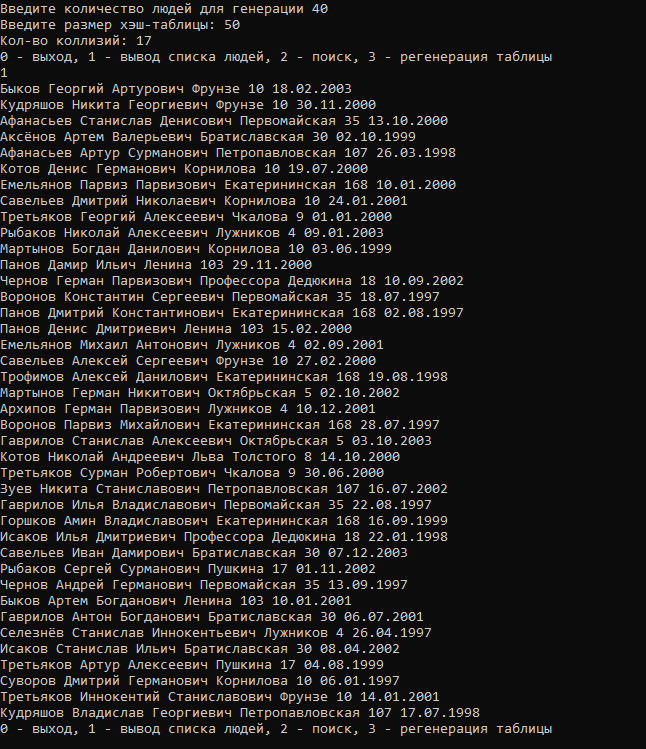
}

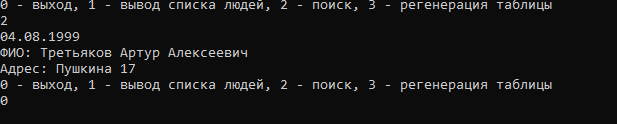
}

return 0;

}

**Скриншоты результатов:**

****

****