Министерство науки и высшего образования Российской Федерации  
Федеральное государственное автономное учреждение высшего образования  
«Пермский национальный исследовательский политехнический университет»

Электротехнический факультет  
Кафедра «Информационные технологии и автоматизированные системы»

**О Т Ч Ё Т  
по лабораторной работе**

Дисциплина: «Основы алгоритмизации и программирования»

Выполнил  
студент группы ИВТ-21-1б  
Мифтахов И.Г

Проверил  
ст. п. кафедры ИТАС  
Яруллин Д.В.

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.

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.

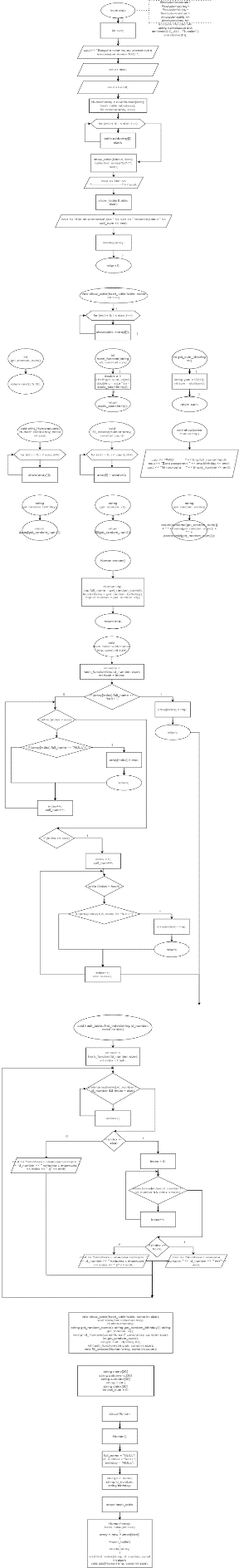
Анализ задачи:

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<cstring>

#include<string>

#include<random>

#include<stdlib.h>

#include<time.h>

#include <functional>

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 ID[30] = { "5717","5321","2323","8321","3232","9999","9210","0105","6521","8891","1118","5617","9596","3490","9085","7254","7152","2015","8047","0001","3281","2131","8328","3727","2814","9348","2388","4211","3284","5521" };

string dates[30] = { "12.12.2011","18.05.1030","09.01.2000","21.03.2003","19.10.2010","23.09.2009","24.10.2010","25.11.2011","26.12.2012","27.01.1988","28.02.1967","29.04.1999","30.05.1867","31.12.1999","01.06.1976","02.07.1980","03.08.1982","04.09.1955","05.10.1948","06.11.2014","16.05.1977","29.06.2004","19.02.2001","09.11.1996","27.03.2002","30.10.2000","27.08.2022","28.02.2019","29.10.1932","30.09.2014" };

int coll\_num = 0;

struct Human

{

Human()

{

full\_name = "NULL";

id\_number = "NULL";

birthday = "NULL";

}

string full\_name;

string id\_number;

string birthday;

};

struct hash\_table

{

Human\*array;

hash\_table(int size)

{

array = new Human[size];

}

~hash\_table()

{

delete[]array;

}

void find\_index(string id\_number, const int size);

void add(Human tmp, const int size);

};

void show\_table(hash\_table\*table, const int size);

void show(const Human tmp);

Human creator();

string get\_random\_name(); string get\_random\_birthday(); string get\_random\_id();

void print\_Humans(const Human\* const array, const int size);

int get\_random\_num();

int get\_sum\_id(string str);

int hash\_function(string str, const int size);

void fill\_creator(Human\*array, const int count);

void hash\_table::find\_index(string id\_number, const int size)

{

int hash = hash\_function(id\_number, size);

int index = hash;

while (array[index].id\_number != id\_number && index < size)

{

index++;

}

if (index >= size)

{

index = 0;

while (array[index].id\_number != id\_number && index < hash)

{

index++;

}

if (index >= hash)

{

cout << "Челобаки с номером паспорта: " << id\_number << " нет" << endl;

}

else

{

cout << "Челобака с номером паспорта: " << id\_number << " найдена с индексом: " << index << " ))" << endl;

}

}

else

{

cout << "Челобака с номером паспорта: " << id\_number << " найдена с индексом: " << index << " ))" << endl;

}

}

void hash\_table::add(Human tmp, const int size)

{

int index = hash\_function(tmp.id\_number, size);

int hash = index;

if (array[index].full\_name == "NULL")

{

array[index] = tmp;

return;

}

else

{

while (index < size)

{

if (array[index].full\_name == "NULL")

{

array[index] = tmp;

return;

}

index++;

coll\_num++;

}

if (index >= size)

{

index = 0;

coll\_num++;

while (index < hash)

{

if (array[index].full\_name == "NULL")

{

array[index] = tmp;

return;

}

index++;

coll\_num++;

}

}

}

}

Human creator()

{

Human tmp;

tmp.full\_name = get\_random\_name();

tmp.birthday = get\_random\_birthday();

tmp.id\_number = get\_random\_id();

return tmp;

}

string get\_random\_name()

{

return(surname[get\_random\_num()] + " " + name[get\_random\_num()] + " " + patronymic[get\_random\_num()]);

}

string get\_random\_birthday()

{

return dates[get\_random\_num()];

}

string get\_random\_id()

{

return ID[get\_random\_num()];

}

void show(const Human tmp)

{

cout << "ФИО: " << tmp.full\_name<<endl;

cout << "Дата рождения: " << tmp.birthday << endl;

cout << "№ паспорта: " << tmp.id\_number << endl;

}

void print\_Humans(const Human\* const array, const int size)

{

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

{

show(array[i]);

}

}

void fill\_creator(Human\*array, const int count)

{

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

{

array[i] = creator();

}

}

int get\_random\_num()

{

return rand() % 30;

}

int get\_sum\_id(string str)

{

string yam = ID[11];

int sum = stoi(yam);

return sum;

}

int hash\_function(string str, const int size)

{

double a = 0.618\*get\_sum\_id(str);

double c = size \* (a - static\_cast<int>(a));

return static\_cast<int>(c);

}

void show\_table(hash\_table\*table, const int size)

{

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

{

show(table->array[i]);

}

}

int main()

{

srand(time(0));

setlocale(LC\_ALL, "Rus");

int size;

cout << "Введите количество элементов в массиве(не менее 100): ";

cin >> size;

cout << endl;

Human\*array = new Human[size];

hash\_table table(size);

fill\_creator(array, size);

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

{

table.add(array[i], size);

}

show\_table(&table, size);

table.find\_index("5617", size);

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

show\_table(&table, size);

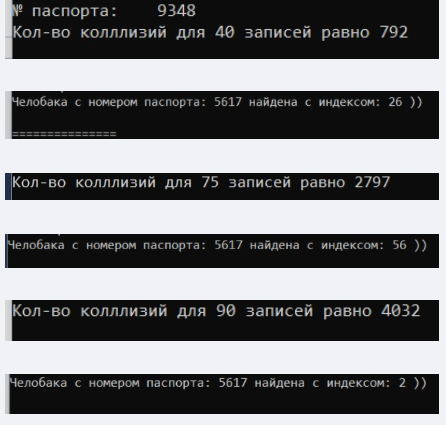
cout << "Кол-во колллизий для " << size << " записей равно " << coll\_num << endl;

delete[] array;

return 0;

}

Скриншоты результатов:



Анализ результатов:

Программа работает корректно. Выводятся количества коллизий и индекс человека с нужной серией паспорта.