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 15

Выполнил:

студент группы ИВТ-20-2Б Чувашев Максим.

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.



**Анализ задачи**

1. Для решения задачи необходимо:
   1. Организовать структуру DATA для хранения всех данных об объекте

struct DATA

{

string Name;

string Surname;

string Middle;

int day;

int month;

int year;

int Home;

string Street;

string City;

string data;

int code;

};

* 1. Организовать структуру HashTab для инициализации хэш таблицы

struct HashTab

{

int key;

int code;

bool f;

string data;

};

* 1. Ораганизовать функцию humanINIT для инициализации объекта структуры DATA
  2. Организовать функцию void PrintH(DATA\* human, int size) для отображения элемента массива структур в консоль
  3. Организовать функция void PrintHashTable(vector<HashTab> HASH) для отображения сгенерированной хэш таблицы
  4. Организовать функцию int Hashcode(string str) в которой происходит вычисление hash кода
  5. Организовать функцию void FillHashTable(DATA\* human, int size, vector<HashTab>& HASH) для заполнения хэш таблицы
  6. Организовать функцию void search(int key, int ind, vector<HashTab> HASH) для выполнения поиска по хэш таблице
  7. void SearchInHashTable(DATA\* human, int size, vector<HashTab> HASH) для вызов функции поиска по таблице hash
  8. Организовать функцию void Add(DATA\* human, vector<HashTab> HASH) для добавления элемента
  9. Организовать фукнцию bool Choice(DATA\* human, int size, int foo, bool f, vector<HashTab>\* HASH), в которой пользователь выполняет выбор действий

1. Для решения задачи данные были представлены в следующем виде:
   1. Данные генерируются в виде массива структур
2. Для операций ввода и вывода использовались следующие операторы и функции:
   1. Операторы ввода и вывода cin, cout

cout << "Введите размер:\t";

cin >> size;

1. Поставленные задачи будут решены следующими действиями:
   1. В функции void humanINIT(DATA\* human, int size) происходит генерация объекта структуры DATA

void humanINIT(DATA\* human, int size)

{

string name[] = { "Александр", "Алексей", "Анатолий", "Андрей", "Антон", "Аркадий", "Арсений", "Артём", "Артур", "Борис", "Вадим", "Валентин", "Валерий", "Василий", "Виктор", "Виталий", "Владимир", "Владислав", "Вячеслав", "Георгий", "Глеб", "Григорий", "Даниил", "Денис", "Дмитрий", "Евгений", "Егор", "Иван", "Игорь", "Илья", "Кирилл", "Константин", "Лев", "Леонид", "Максим", "Марк", "Матвей", "Михаил", "Никита", "Николай", "Олег", "Павел", "Пётр", "Роман", "Руслан", "Сергей", "Степан", "Тимур", "Фёдор", "Юрий", "Ярослав" };

string surname[] = { "Пантелеев", "Чувашев", "Галинов", "Ананина", "Масылюк", "Вострокнутова", "Сабуров", "Исламов", "Шамай", "Филатов", "Сафронов", "Шарпов", "Карелов", "Брейкин", "Копытов", "Солдатов", "Кузнецов", "Пажгин", "Фотин", "Бадретдинов", "Механошин", "Булдаков", "Тулинов", "Тедеев", "Колпаков", "Черных", "Нефедов", "Рябцев", "Пепеляев", "Тарасов", "Аркадьев" };

string middle[] = { "Романович", "Иосифович", "Аникитевич", "Прохорович", "Архипович", "Александрович", "Мартьянович", "Богданович", "Никифорович", "Георгиевич" , "Прокофиевич" , "Валериевич" , "Владиславович" , "Данилевич" , "Ульянович" , "Кондратиевич" , "Валериевич" , "Алексеевич" , "Кондратович" , "Евграфович" , "Куприянович" , "Кондратович" , "Тихонович" , "Игнатиевич" , "Алексеевич" , "Венедиктович" , "Мартьянович" , "Родионович" , "Тимурович" , "Самуилович" , "Климентович" , "Натанович" , "Леонович" , "Игоревич" , "Захарович" , "Андронович" , "Платонович" , "Несторович" , "Модестович" , "Ерофеевич" , "Адамович" };

string street[] = { "Ленина", "Авиаторов", "Авиационная", "Гоголя", "Горчакова", "Капотня", "Кедрова", "Луховицкая", "Малая Бронная", "Малая Калужская", "Малевича", " Новорублевская", "Новостроевская", "Новотетерки", "Обручева", "Оборонная" };

string city[] = { "Коноха", "Токио", "Киото", "Киров", "Пермь", "Москва", "Зувека", "Калуга", "Мурманск", "Сочи", "Адлер", "Рим", "Бостон", "Хьюстон", "Лос-Анджелес", "Сан-Франциско", "Нагоя", "Окадзаки", "Саппоро", "Никко", "Китами", "Тояма", "Хиросима", "Иокогама", "Осака", "Нагоя", "Фукуока", "Сендай", "Ниигата", "Кумамото", "Химедзи" };

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

{

human[i].year = 1903 + rand() % 117;

human[i].month = 1 + rand() % 12;

int a = human[i].month;

switch (a)

{

case 1: human[i].day = 1 + (rand() % 31); break;

case 2: human[i].day = 1 + (rand() % 28); break;

case 3: human[i].day = 1 + (rand() % 31); break;

case 4: human[i].day = 1 + (rand() % 30); break;

case 5: human[i].day = 1 + (rand() % 31); break;

case 6: human[i].day = 1 + (rand() % 30); break;

case 7: human[i].day = 1 + (rand() % 31); break;

case 8: human[i].day = 1 + (rand() % 31); break;

case 9: human[i].day = 1 + (rand() % 30); break;

case 10: human[i].day = 1 + (rand() % 31); break;

case 11: human[i].day = 1 + (rand() % 30); break;

case 12: human[i].day = 1 + (rand() % 31); break;

default:

break;

}

int g = rand() % (sizeof(name) / sizeof(name[0]));

human[i].Name = name[g];

g = rand() % (sizeof(surname) / sizeof(surname[0]));

human[i].Surname = surname[g];

g = rand() % (sizeof(middle) / sizeof(middle[0]));

human[i].Middle = middle[g];

g = rand() % (sizeof(street) / sizeof(street[0]));

human[i].Street = street[g];

g = rand() % (sizeof(city) / sizeof(city[0]));

human[i].City = city[g];

human[i].Home = rand() % 100;

}

}

* 1. В функции void PrintH(DATA\* human, int size) происходит вывод объекта структуры в консоль

void PrintH(DATA\* human, int size)

{

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

{

cout << "[" << i << "]" << endl;

cout << human[i].Surname << endl;

cout << human[i].Name << endl;

cout << human[i].Middle << endl;

cout << human[i].day << "." << human[i].month << "." << human[i].year << endl;

cout << human[i].City << ", " << human[i].Street << ", " << human[i].Home << endl << endl;

}

}

* 1. В функции void PrintHashTable(vector<HashTab> HASH) происходит печать сгенерированной хэш таблицы в консоль

void PrintHashTable(vector<HashTab> HASH)

{

int count = 1;

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

{

if (HASH[i].code != NULL)

{

cout << count << " элемент хеш-таблицы находится под индексом: " << i << endl;

cout << "хеш:\t " << HASH[i].code % 256 << endl;

cout << HASH[i].data << endl << endl;

count++;

}

}

}

* 1. В функции int Hashcode(string str) происходит инициализация хэш кода для элемента

int Hashcode(string str)

{

int H = 0;

int Help = 1;

for (int j = 0; j < str.length(); j++)

{

H += int(str[j]) \* Help;

if (j % 3 == 0)

{

Help += 21;

}

Help += 2;

}

return abs(H);

}

* 1. В функции void FillHashTable(DATA\* human, int size, vector<HashTab>& HASH) происходит заполнение хэш таблицы

void FillHashTable(DATA\* human, int size, vector<HashTab>& HASH)

{

vector <HashTab> Hash2(size \* 3);

int N = 256;

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

{

string str = human[i].Surname + " " + human[i].Name + " " + human[i].Middle;

int H = Hashcode(str);

human[i].code = H;

human[i].data = str + "\n" + to\_string(human[i].day) + "." + to\_string(human[i].month) + "." + to\_string(human[i].year) + "\n" + human[i].City + ", " + human[i].Street + ", " + to\_string(human[i].Home) + "\n";

if (Hash2[human[i].code % N].code == NULL)

{

Hash2[human[i].code % N].code = human[i].code;

Hash2[human[i].code % N].data = human[i].data;

Hash2[human[i].code % N].key = human[i].code % N;

Hash2[human[i].code % N].f = true;

}

else if (Hash2[human[i].code % N].code != NULL)

{

int a = human[i].code % N;

while (Hash2[a].code != NULL)

{

a++;

}

Hash2[human[i].code % N].code = human[i].code;

Hash2[human[i].code % N].data = human[i].data;

Hash2[human[i].code % N].f = true;

}

}

HASH = Hash2;

}

* 1. В функции void search(int key, int ind, vector<HashTab> HASH) происходится поиск элемента по таблице исходя из полученного ключа

void search(int key, int ind, vector<HashTab> HASH)

{

if (HASH[ind].code != NULL)

{

while (HASH[ind].code != key && HASH[ind].code != NULL)

{

ind++;

}

if (HASH[ind].code == NULL)

{

cout << "Элемент не найден!" << endl;

}

else

{

cout << HASH[ind].data << endl;

}

}

else

{

cout << "Элемент не найден!" << endl;

}

}

* 1. В фукнции void SearchInHashTable(DATA\* human, int size, vector<HashTab> HASH) так же производится поиск по хэш таблице

void SearchInHashTable(DATA\* human, int size, vector<HashTab> HASH)

{

string str;

cout << "Введите желаемое ФИО:\t";

cin.ignore();

getline(cin, str);

cout << endl;

int key = Hashcode(str);

int ind = key % 256;

search(key, ind, HASH);

}

* 1. В функции void Add(DATA\* human, vector<HashTab> HASH) происходит добавление элемента в список

void Add(DATA\* human, vector<HashTab> HASH)

{

DATA append;

cout << "Введите фамилию:\t";

cin >> append.Surname;

cout << "Введите имя: \t";

cin >> append.Name;

cout << "Введите Отчество: \t";

cin >> append.Middle;

append.year = 1903 + rand() % 117;

append.month = 1 + rand() % 12;

int a = append.month;

switch (a)

{

case 1: append.day = 1 + (rand() % 31); break;

case 2: append.day = 1 + (rand() % 28); break;

case 3: append.day = 1 + (rand() % 31); break;

case 4: append.day = 1 + (rand() % 30); break;

case 5: append.day = 1 + (rand() % 31); break;

case 6: append.day = 1 + (rand() % 30); break;

case 7: append.day = 1 + (rand() % 31); break;

case 8: append.day = 1 + (rand() % 31); break;

case 9: append.day = 1 + (rand() % 30); break;

case 10: append.day = 1 + (rand() % 31); break;

case 11: append.day = 1 + (rand() % 30); break;

case 12: append.day = 1 + (rand() % 31); break;

default:

break;

}

cout << "Дата рождения сгенерирована автоматически!" << endl;

cout << "Введите город проживания:\t";

cin >> append.City;

cout << "Введите улицу проживания:\t";

cin >> append.Street;

cout << "Введите номер дома:\t";

cin >> append.Home;

while (append.Home < 1 || append.Home > 200)

{

cout << "Введите корректный номер дома!" << endl;

cout << "Введите номер дома:\t";

cin >> append.Home;

}

}

* 1. В функции bool Choice(DATA\* human, int size, int foo, bool f, vector<HashTab>\* HASH) пользователь производит выбор действия

bool Choice(DATA\* human, int size, int foo, bool f, vector<HashTab>\* HASH)

{

switch (foo)

{

case 1: humanINIT(human, size); break;

case 2: PrintH(human, size); break;

case 3: FillHashTable(human, size, \*HASH); break;

case 4: SearchInHashTable(human, size, \*HASH); break;

case 5: PrintHashTable(\*HASH); break;

case 6: f = true; break;

default:

break;

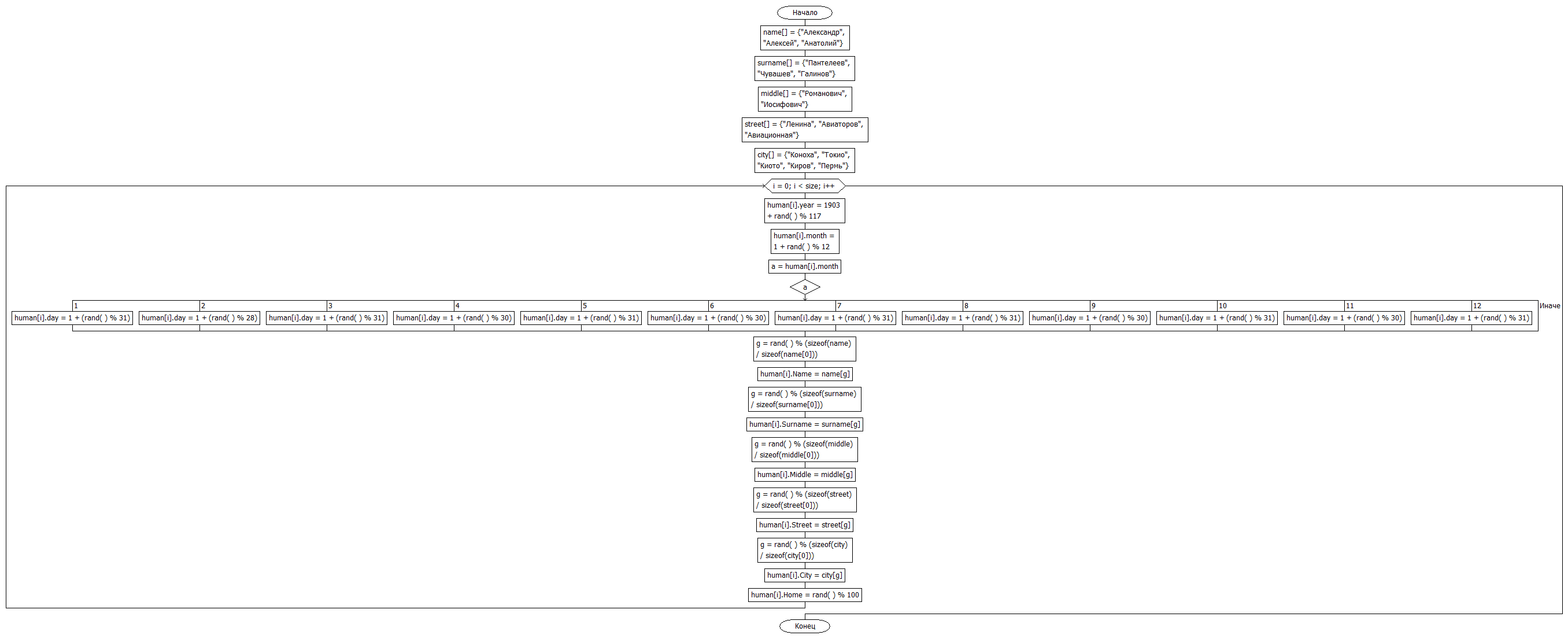
}

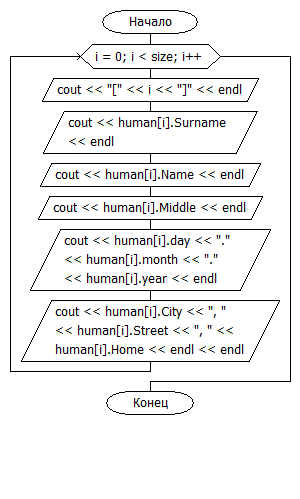
return f;

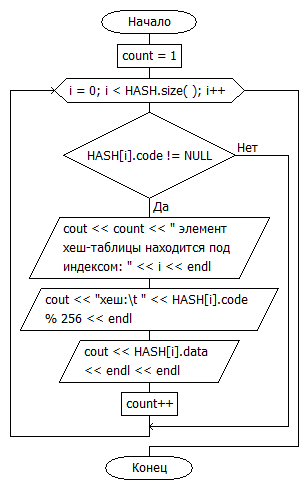
}

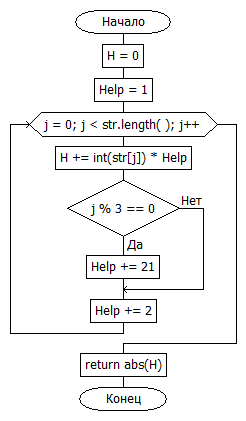
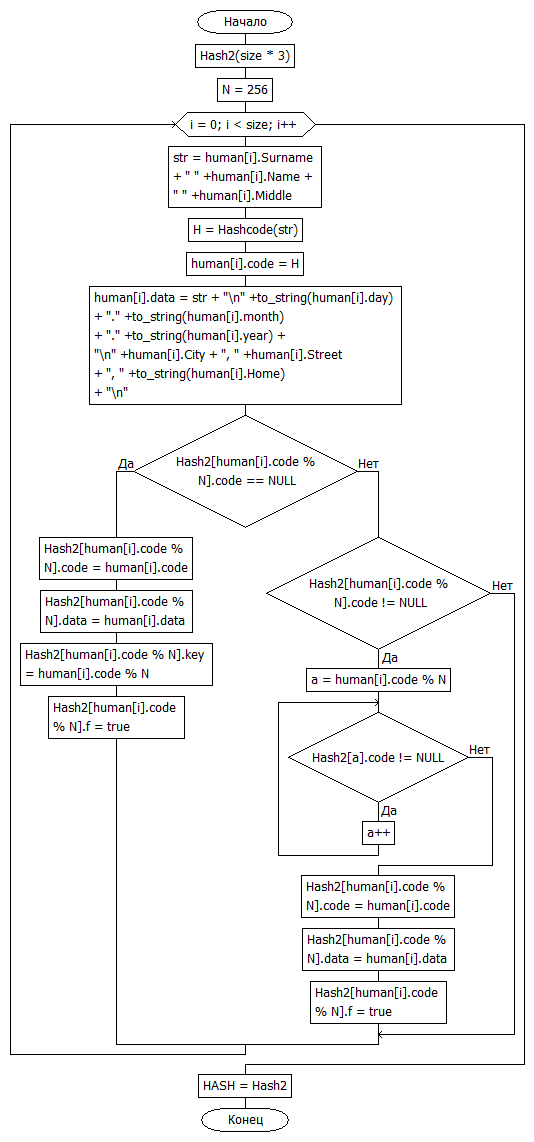
**Блок-схема**

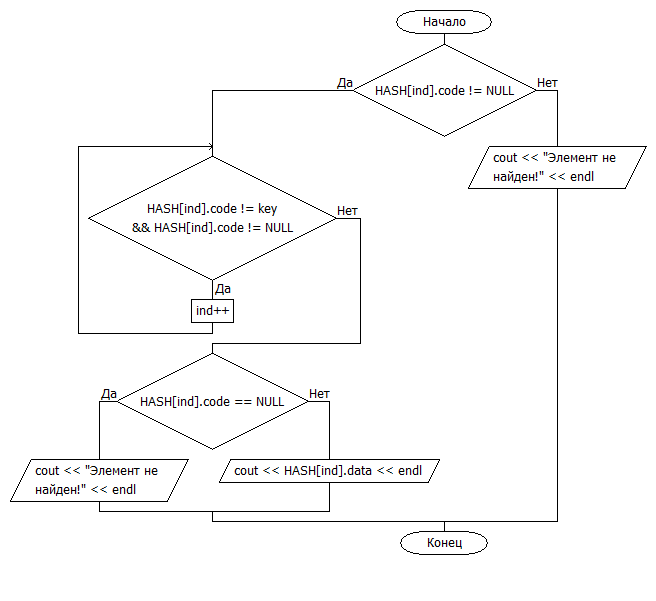


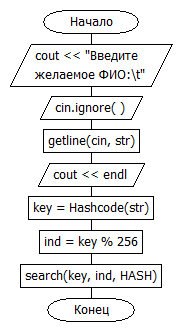


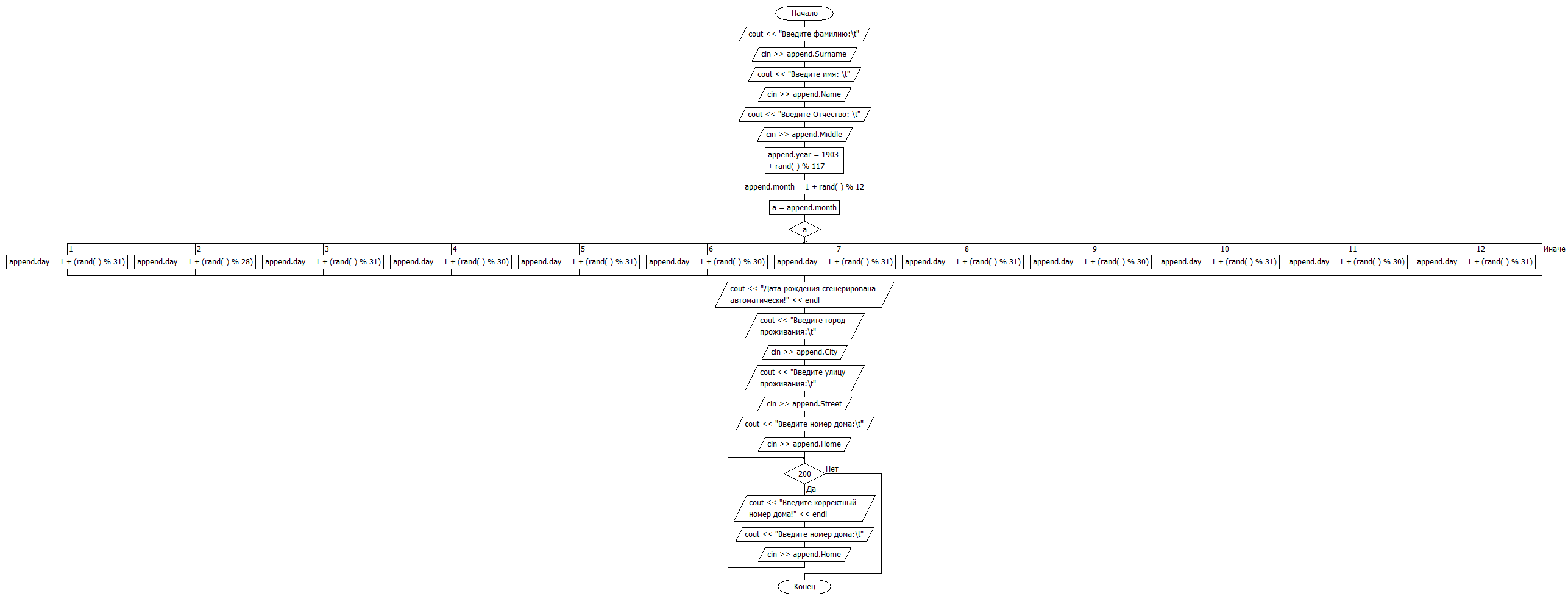


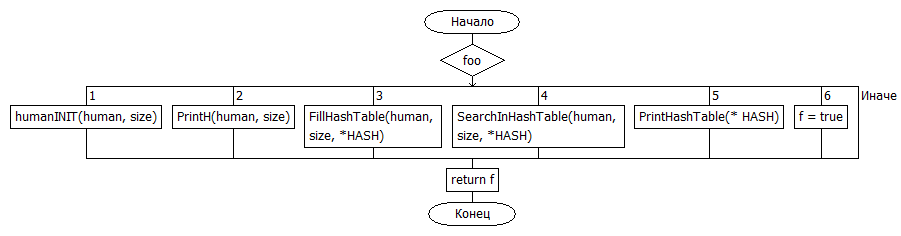


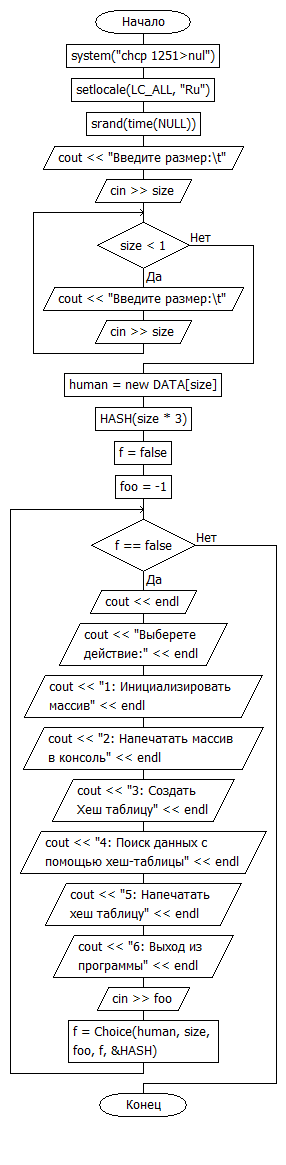
 











**Код**

#include <iostream>

#include <string>

#include <list>

#include <ctime>

#include <fstream>

#include <vector>

using namespace std;

struct DATA

{

string Name;

string Surname;

string Middle;

int day;

int month;

int year;

int Home;

string Street;

string City;

string data;

int code;

};

struct HashTab

{

int key;

int code;

bool f;

string data;

};

void humanINIT(DATA\* human, int size)

{

string name[] = { "Александр", "Алексей", "Анатолий", "Андрей", "Антон", "Аркадий", "Арсений", "Артём", "Артур", "Борис", "Вадим", "Валентин", "Валерий", "Василий", "Виктор", "Виталий", "Владимир", "Владислав", "Вячеслав", "Георгий", "Глеб", "Григорий", "Даниил", "Денис", "Дмитрий", "Евгений", "Егор", "Иван", "Игорь", "Илья", "Кирилл", "Константин", "Лев", "Леонид", "Максим", "Марк", "Матвей", "Михаил", "Никита", "Николай", "Олег", "Павел", "Пётр", "Роман", "Руслан", "Сергей", "Степан", "Тимур", "Фёдор", "Юрий", "Ярослав" };

string surname[] = { "Пантелеев", "Чувашев", "Галинов", "Ананина", "Масылюк", "Вострокнутова", "Сабуров", "Исламов", "Шамай", "Филатов", "Сафронов", "Шарпов", "Карелов", "Брейкин", "Копытов", "Солдатов", "Кузнецов", "Пажгин", "Фотин", "Бадретдинов", "Механошин", "Булдаков", "Тулинов", "Тедеев", "Колпаков", "Черных", "Нефедов", "Рябцев", "Пепеляев", "Тарасов", "Аркадьев" };

string middle[] = { "Романович", "Иосифович", "Аникитевич", "Прохорович", "Архипович", "Александрович", "Мартьянович", "Богданович", "Никифорович", "Георгиевич" , "Прокофиевич" , "Валериевич" , "Владиславович" , "Данилевич" , "Ульянович" , "Кондратиевич" , "Валериевич" , "Алексеевич" , "Кондратович" , "Евграфович" , "Куприянович" , "Кондратович" , "Тихонович" , "Игнатиевич" , "Алексеевич" , "Венедиктович" , "Мартьянович" , "Родионович" , "Тимурович" , "Самуилович" , "Климентович" , "Натанович" , "Леонович" , "Игоревич" , "Захарович" , "Андронович" , "Платонович" , "Несторович" , "Модестович" , "Ерофеевич" , "Адамович" };

string street[] = { "Ленина", "Авиаторов", "Авиационная", "Гоголя", "Горчакова", "Капотня", "Кедрова", "Луховицкая", "Малая Бронная", "Малая Калужская", "Малевича", " Новорублевская", "Новостроевская", "Новотетерки", "Обручева", "Оборонная" };

string city[] = { "Коноха", "Токио", "Киото", "Киров", "Пермь", "Москва", "Зувека", "Калуга", "Мурманск", "Сочи", "Адлер", "Рим", "Бостон", "Хьюстон", "Лос-Анджелес", "Сан-Франциско", "Нагоя", "Окадзаки", "Саппоро", "Никко", "Китами", "Тояма", "Хиросима", "Иокогама", "Осака", "Нагоя", "Фукуока", "Сендай", "Ниигата", "Кумамото", "Химедзи" };

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

{

human[i].year = 1903 + rand() % 117;

human[i].month = 1 + rand() % 12;

int a = human[i].month;

switch (a)

{

case 1: human[i].day = 1 + (rand() % 31); break;

case 2: human[i].day = 1 + (rand() % 28); break;

case 3: human[i].day = 1 + (rand() % 31); break;

case 4: human[i].day = 1 + (rand() % 30); break;

case 5: human[i].day = 1 + (rand() % 31); break;

case 6: human[i].day = 1 + (rand() % 30); break;

case 7: human[i].day = 1 + (rand() % 31); break;

case 8: human[i].day = 1 + (rand() % 31); break;

case 9: human[i].day = 1 + (rand() % 30); break;

case 10: human[i].day = 1 + (rand() % 31); break;

case 11: human[i].day = 1 + (rand() % 30); break;

case 12: human[i].day = 1 + (rand() % 31); break;

default:

break;

}

int g = rand() % (sizeof(name) / sizeof(name[0]));

human[i].Name = name[g];

g = rand() % (sizeof(surname) / sizeof(surname[0]));

human[i].Surname = surname[g];

g = rand() % (sizeof(middle) / sizeof(middle[0]));

human[i].Middle = middle[g];

g = rand() % (sizeof(street) / sizeof(street[0]));

human[i].Street = street[g];

g = rand() % (sizeof(city) / sizeof(city[0]));

human[i].City = city[g];

human[i].Home = rand() % 100;

}

}

void PrintH(DATA\* human, int size)

{

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

{

cout << "[" << i << "]" << endl;

cout << human[i].Surname << endl;

cout << human[i].Name << endl;

cout << human[i].Middle << endl;

cout << human[i].day << "." << human[i].month << "." << human[i].year << endl;

cout << human[i].City << ", " << human[i].Street << ", " << human[i].Home << endl << endl;

}

}

void PrintHashTable(vector<HashTab> HASH)

{

int count = 1;

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

{

if (HASH[i].code != NULL)

{

cout << count << " элемент хеш-таблицы находится под индексом: " << i << endl;

cout << "хеш:\t " << HASH[i].code % 256 << endl;

cout << HASH[i].data << endl << endl;

count++;

}

}

}

int Hashcode(string str)

{

int H = 0;

int Help = 1;

for (int j = 0; j < str.length(); j++)

{

H += int(str[j]) \* Help;

if (j % 3 == 0)

{

Help += 21;

}

Help += 2;

}

return abs(H);

}

void FillHashTable(DATA\* human, int size, vector<HashTab>& HASH)

{

vector <HashTab> Hash2(size \* 3);

int N = 256;

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

{

string str = human[i].Surname + " " + human[i].Name + " " + human[i].Middle;

int H = Hashcode(str);

human[i].code = H;

human[i].data = str + "\n" + to\_string(human[i].day) + "." + to\_string(human[i].month) + "." + to\_string(human[i].year) + "\n" + human[i].City + ", " + human[i].Street + ", " + to\_string(human[i].Home) + "\n";

if (Hash2[human[i].code % N].code == NULL)

{

Hash2[human[i].code % N].code = human[i].code;

Hash2[human[i].code % N].data = human[i].data;

Hash2[human[i].code % N].key = human[i].code % N;

Hash2[human[i].code % N].f = true;

}

else if (Hash2[human[i].code % N].code != NULL)

{

int a = human[i].code % N;

while (Hash2[a].code != NULL)

{

a++;

}

Hash2[human[i].code % N].code = human[i].code;

Hash2[human[i].code % N].data = human[i].data;

Hash2[human[i].code % N].f = true;

}

}

HASH = Hash2;

}

void search(int key, int ind, vector<HashTab> HASH)

{

if (HASH[ind].code != NULL)

{

while (HASH[ind].code != key && HASH[ind].code != NULL)

{

ind++;

}

if (HASH[ind].code == NULL)

{

cout << "Элемент не найден!" << endl;

}

else

{

cout << HASH[ind].data << endl;

}

}

else

{

cout << "Элемент не найден!" << endl;

}

}

void SearchInHashTable(DATA\* human, int size, vector<HashTab> HASH)

{

string str;

cout << "Введите желаемое ФИО:\t";

cin.ignore();

getline(cin, str);

cout << endl;

int key = Hashcode(str);

int ind = key % 256;

search(key, ind, HASH);

}

void Add(DATA\* human, vector<HashTab> HASH)

{

DATA append;

cout << "Введите фамилию:\t";

cin >> append.Surname;

cout << "Введите имя: \t";

cin >> append.Name;

cout << "Введите Отчество: \t";

cin >> append.Middle;

append.year = 1903 + rand() % 117;

append.month = 1 + rand() % 12;

int a = append.month;

switch (a)

{

case 1: append.day = 1 + (rand() % 31); break;

case 2: append.day = 1 + (rand() % 28); break;

case 3: append.day = 1 + (rand() % 31); break;

case 4: append.day = 1 + (rand() % 30); break;

case 5: append.day = 1 + (rand() % 31); break;

case 6: append.day = 1 + (rand() % 30); break;

case 7: append.day = 1 + (rand() % 31); break;

case 8: append.day = 1 + (rand() % 31); break;

case 9: append.day = 1 + (rand() % 30); break;

case 10: append.day = 1 + (rand() % 31); break;

case 11: append.day = 1 + (rand() % 30); break;

case 12: append.day = 1 + (rand() % 31); break;

default:

break;

}

cout << "Дата рождения сгенерирована автоматически!" << endl;

cout << "Введите город проживания:\t";

cin >> append.City;

cout << "Введите улицу проживания:\t";

cin >> append.Street;

cout << "Введите номер дома:\t";

cin >> append.Home;

while (append.Home < 1 || append.Home > 200)

{

cout << "Введите корректный номер дома!" << endl;

cout << "Введите номер дома:\t";

cin >> append.Home;

}

}

bool Choice(DATA\* human, int size, int foo, bool f, vector<HashTab>\* HASH)

{

switch (foo)

{

case 1: humanINIT(human, size); break;

case 2: PrintH(human, size); break;

case 3: FillHashTable(human, size, \*HASH); break;

case 4: SearchInHashTable(human, size, \*HASH); break;

case 5: PrintHashTable(\*HASH); break;

case 6: f = true; break;

default:

break;

}

return f;

}

int main()

{

system("chcp 1251>nul");

setlocale(LC\_ALL, "Ru");

srand(time(NULL));

int size;

cout << "Введите размер:\t";

cin >> size;

while (size < 1)

{

cout << "Введите размер:\t";

cin >> size;

}

DATA\* human = new DATA[size];

vector<HashTab> HASH(size \* 3);

bool f = false;

int foo = -1;

while (f == false)

{

cout << endl;

cout << "Выберете действие:" << endl;

cout << "1: Инициализировать массив" << endl;

cout << "2: Напечатать массив в консоль" << endl;

cout << "3: Создать Хеш таблицу" << endl;

cout << "4: Поиск данных с помощью хеш-таблицы" << endl;

cout << "5: Напечатать хеш таблицу" << endl;

cout << "6: Выход из программы" << endl;

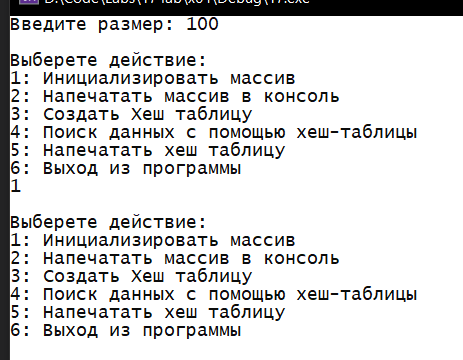
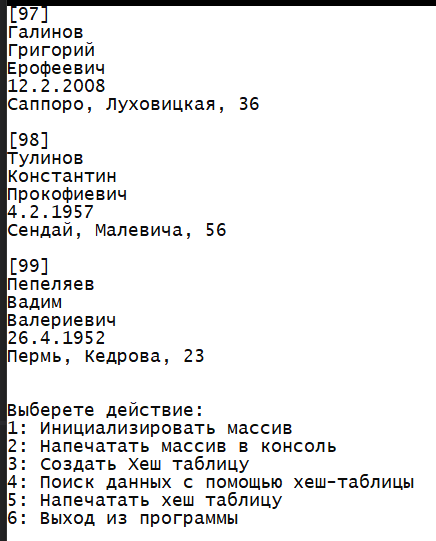
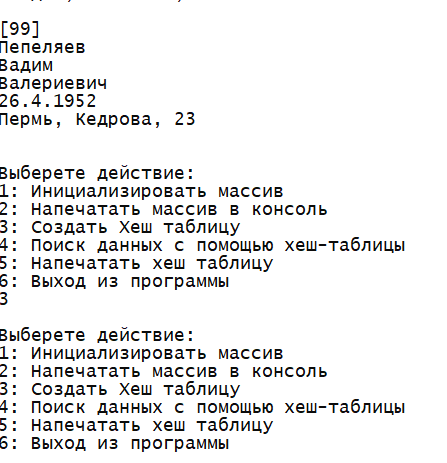
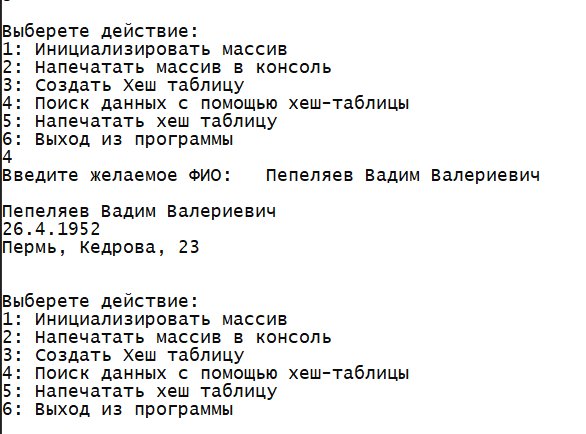
cin >> foo;

f = Choice(human, size, foo, f, &HASH);

}

}

**Скриншоты**

* 1. 
  2. 
  3. 
  4. 
  5. 