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: “Хеш-функции”

**Вар.20**

Выполнил работу:

студент группы ИВТ-20-2Б

Кузнецов Н.Д.

Проверил:

Доцент кафедры ИТАС

Полякова О.А.

Пермь, 2020

**Цель задачи**

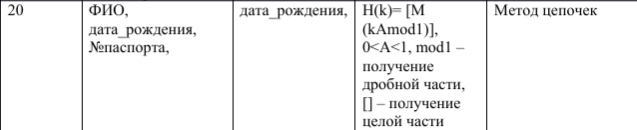
Получить практические навыки работы с внешними сортировками;

Для достижения поставленной цели необходимо решить следующие задачи:

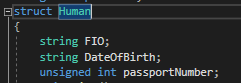
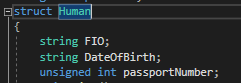
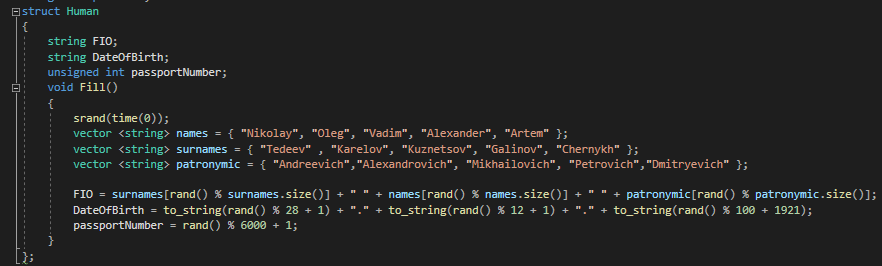
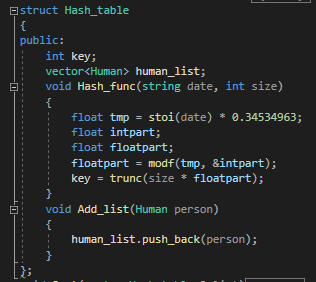
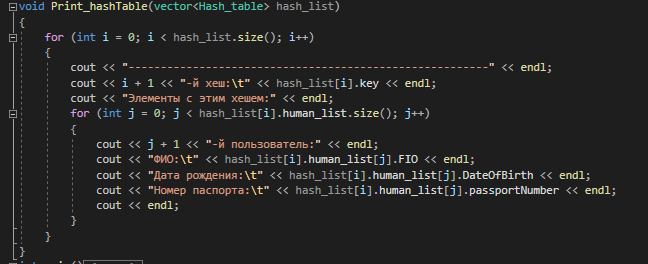
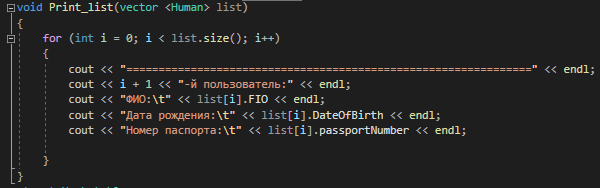
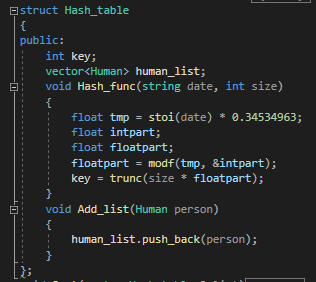
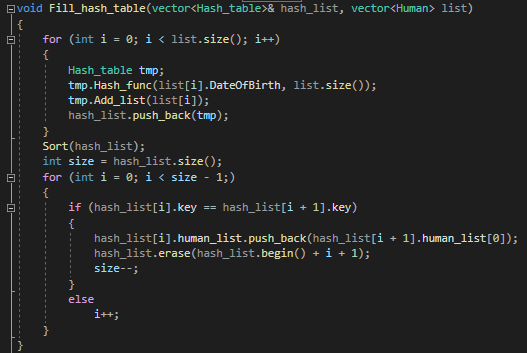
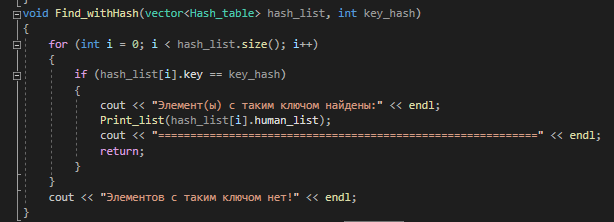
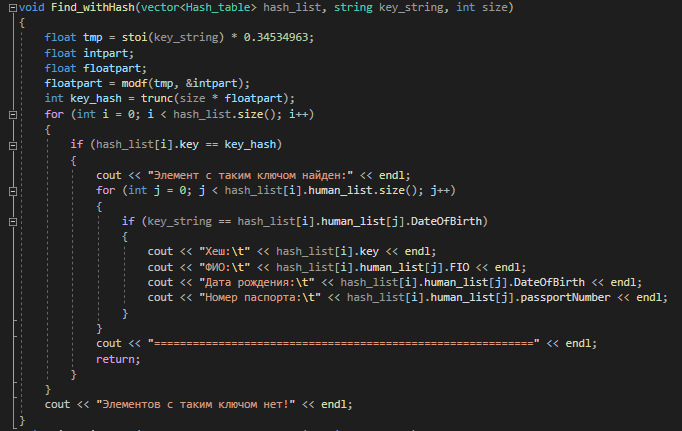
* Организовать ввод-вывод данных.
* Реализовать функцию хеширования.
* Реализовать функции поиска по ключу и по хешу.
* Реализовать структуру человека и структуру таблицы хешей.
* Разработать функцию, заполняющую массив случайными числами.
* Разработать программу.

**Постановка задачи**

1. Изучить построение функции хеширования и алгоритмов хеширования данных
2. Научиться разрабатывать алгоритмы открытого и закрытого хеширования при решении задач на языке C++.



**Анализ задачи**

1. Для решения задачи необходимо:
   1. Реализовать структуру хеш-таблицы и функцию хеширования.
   2. Реализовать функцию, разрешающую коллизии.
   3. Реализовать структуру человека с заданными полями.
   4. Создать динамический массив объектов структуры, заполнение с помощью ДСЧ.
2. В ходе работы были использованы следующие типы данных:
   1. String - поля структуры человека.  
      
   2. Unsigned int - поле номера паспорта человека.  
      
   3. Int - размер динамического массива.
   4. В качестве динамического массива используется класс vector.
3. Для решения задачи данные были представлены в следующем виде:
   1. Динамический массив (vector) типа Human.  
      
4. Для решения задачи использовались следующие структуры:
   1. Human - поля string (ФИО и дата рождения), поле unsigned int - номер паспорта).  
      
   2. Hash\_table - поле int (key - хеш), поле vector <Human> (массив людей, чья дата рождения соответствует хешу key).  
      
5. Для операций ввода и вывода использовались следующие операторы и функции:
   1. cout для вывода таблицы хешей и массива людей в функциях Print\_hashTable и Print\_list.  
      
   2. Cin для ввода ключа поиска.
6. Поставленные задачи будут решены следующими действиями:
   1. Хеш-функция (метод структуры Hash\_table) преобразует строку date, передаваемую как параметр, в значение типа int (хеш). Суть самого алгоритма заключается в том, чтобы привести строку к целочисленному типу и умножить на некую константу (0 < const < 1). Далее выделяется дробная часть от полученного числа, умножается на длину исходного массива (массив типа Human) и дробная часть отбрасывается.  
      
   2. Функция заполнения хеш-таблицы заполняет ее и разрешает коллизии. Функция принимает в качестве параметров ссылки на хеш-таблицу и массив людей. С помощью цикла с автоматическим перебором переменных происходит заполнение таблицы (поле key i-го элемента - результат работы метода Hash\_func, в который передается строка i-го элемента массива людей и размер этого массива; поле human\_list – i-ый человек в массиве людей). Далее для разрешения коллизий хеш-таблица сортируется по полю key.(сортировка Шелла). Алгоритм разрешения коллизий прост: с помощью цикла for осуществляется сравнение текущего элемента таблицы и следующего, если они равны, то в поле human\_list текущего элемента добавляется человек из поля human\_list следующего элемента таблицы, следующий элемент удаляется из таблицы, иначе управляющая переменная инкрементируется.  
      
   3. Функция Find\_withHash перегружена (для поиска по хешу и дате). В случае если функция принимает дату в виде строки, то она хешируется и осуществляется поиск по хешу, далее в подмассиве людей, удовлетворяющему найденному хешу осуществляется поиск по ключу. Если же в функцию передается хеш, то в случае нахождения такого хеша в таблице выводится весь подмассив людей.   
      

**Код программы.**

#include <iostream>

#include <vector>

#include <string>

#include <list>

#include <ctime>

#include <cmath>

using namespace std;

struct Human

{

string FIO;

string DateOfBirth;

unsigned int passportNumber;

void Fill()

{

srand(time(0));

vector <string> names = { "Nikolay", "Oleg", "Vadim", "Alexander", "Artem" };

vector <string> surnames = { "Tedeev" , "Karelov", "Kuznetsov", "Galinov", "Chernykh" };

vector <string> patronymic = { "Andreevich","Alexandrovich", "Mikhailovich", "Petrovich","Dmitryevich" };

FIO = surnames[rand() % surnames.size()] + " " + names[rand() % names.size()] + " " + patronymic[rand() % patronymic.size()];

DateOfBirth = to\_string(rand() % 28 + 1) + "." + to\_string(rand() % 12 + 1) + "." + to\_string(rand() % 100 + 1921);

passportNumber = rand() % 6000 + 1;

}

};

vector<Human> CreateArray(int size)

{

srand(time(0));

vector <Human> result;

vector <string> names = { "Nikolay", "Oleg", "Vadim", "Alexander", "Artem" };

vector <string> surnames = { "Tedeev" , "Karelov", "Kuznetsov", "Galinov", "Chernykh" };

vector <string> patronymic = { "Andreevich","Alexandrovich", "Mikhailovich", "Petrovich","Dmitryevich" };

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

{

Human NewPerson;

NewPerson.FIO = surnames[rand() % surnames.size()] + " " + names[rand() % names.size()] + " " + patronymic[rand() % patronymic.size()];

NewPerson.DateOfBirth = to\_string(rand() % 28 + 1) + "." + to\_string(rand() % 12 + 1) + "." + to\_string(rand() % 100 + 1921);

NewPerson.passportNumber = rand() % 6000 + 1;

result.push\_back(NewPerson);

}

return result;

}

void Print\_list(vector <Human> list)

{

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

{

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

cout << i + 1 << "-й пользователь:" << endl;

cout << "ФИО:\t" << list[i].FIO << endl;

cout << "Дата рождения:\t" << list[i].DateOfBirth << endl;

cout << "Номер паспорта:\t" << list[i].passportNumber << endl;

}

}

struct Hash\_table

{

public:

int key;

vector<Human> human\_list;

void Hash\_func(string date, int size)

{

float tmp = stoi(date) \* 0.34534963;

float intpart;

float floatpart;

floatpart = modf(tmp, &intpart);

key = trunc(size \* floatpart);

}

void Add\_list(Human person)

{

human\_list.push\_back(person);

}

};

void Sort(vector<Hash\_table>& list)

{

for (int step = list.size() / 2; step > 0; step /= 2)

{

for (int i = step; i < list.size(); i++)

{

for (int j = i; j >= step; j -= step)

{

if (list[j].key < list[j - step].key)

swap(list[j], list[j - step]);

}

}

}

}

void Fill\_hash\_table(vector<Hash\_table>& hash\_list, vector<Human> list)

{

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

{

Hash\_table tmp;

tmp.Hash\_func(list[i].DateOfBirth, list.size());

tmp.Add\_list(list[i]);

hash\_list.push\_back(tmp);

}

Sort(hash\_list);

int size = hash\_list.size();

for (int i = 0; i < size - 1;)

{

if (hash\_list[i].key == hash\_list[i + 1].key)

{

hash\_list[i].human\_list.push\_back(hash\_list[i + 1].human\_list[0]);

hash\_list.erase(hash\_list.begin() + i + 1);

size--;

}

else

i++;

}

}

void Find\_withHash(vector<Hash\_table> hash\_list, string key\_string, int size)

{

float tmp = stoi(key\_string) \* 0.34534963;

float intpart;

float floatpart;

floatpart = modf(tmp, &intpart);

int key\_hash = trunc(size \* floatpart);

for (int i = 0; i < hash\_list.size(); i++)

{

if (hash\_list[i].key == key\_hash)

{

cout << "Элемент с таким ключом найден:" << endl;

for (int j = 0; j < hash\_list[i].human\_list.size(); j++)

{

if (key\_string == hash\_list[i].human\_list[j].DateOfBirth)

{

cout << "Хеш:\t" << hash\_list[i].key << endl;

cout << "ФИО:\t" << hash\_list[i].human\_list[j].FIO << endl;

cout << "Дата рождения:\t" << hash\_list[i].human\_list[j].DateOfBirth << endl;

cout << "Номер паспорта:\t" << hash\_list[i].human\_list[j].passportNumber << endl;

}

}

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

return;

}

}

cout << "Элементов с таким ключом нет!" << endl;

}

void Find\_withHash(vector<Hash\_table> hash\_list, int key\_hash)

{

for (int i = 0; i < hash\_list.size(); i++)

{

if (hash\_list[i].key == key\_hash)

{

cout << "Элемент(ы) с таким ключом найдены:" << endl;

Print\_list(hash\_list[i].human\_list);

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

return;

}

}

cout << "Элементов с таким ключом нет!" << endl;

}

void Print\_hashTable(vector<Hash\_table> hash\_list)

{

for (int i = 0; i < hash\_list.size(); i++)

{

cout << "--------------------------------------------------------" << endl;

cout << i + 1 << "-й хеш:\t" << hash\_list[i].key << endl;

cout << "Элементы с этим хешем:" << endl;

for (int j = 0; j < hash\_list[i].human\_list.size(); j++)

{

cout << j + 1 << "-й пользователь:" << endl;

cout << "ФИО:\t" << hash\_list[i].human\_list[j].FIO << endl;

cout << "Дата рождения:\t" << hash\_list[i].human\_list[j].DateOfBirth << endl;

cout << "Номер паспорта:\t" << hash\_list[i].human\_list[j].passportNumber << endl;

cout << endl;

}

}

}

int main()

{

setlocale(LC\_ALL,"ru");

int size;

cin >> size;

vector <Human> list = CreateArray(size);

vector<Hash\_table> hash\_list;

Print\_list(list);

Fill\_hash\_table(hash\_list, list);

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

cout << "Таблица хешей:" << endl;

Print\_hashTable(hash\_list);

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

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

string Key;

cout << "Введите дату, по которой собираетесь искать (через точку, без нулей):\t";

cin >> Key;

Find\_withHash(hash\_list, Key, list.size());

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

int key;

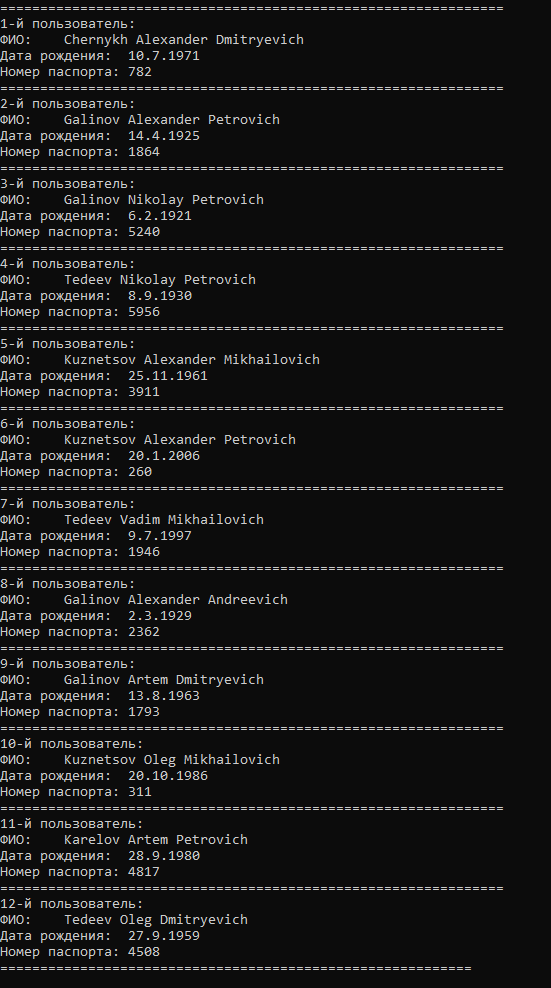
cout << "Введите хеш, по которому собираетесь искать:\t";

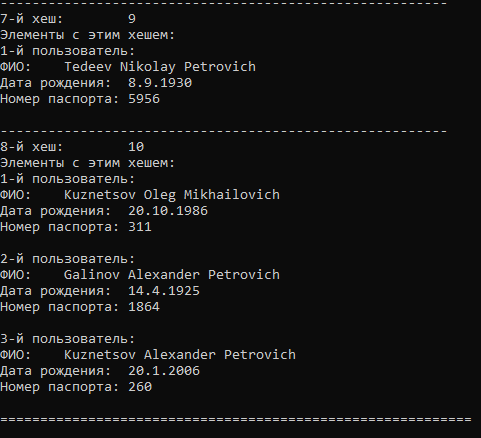
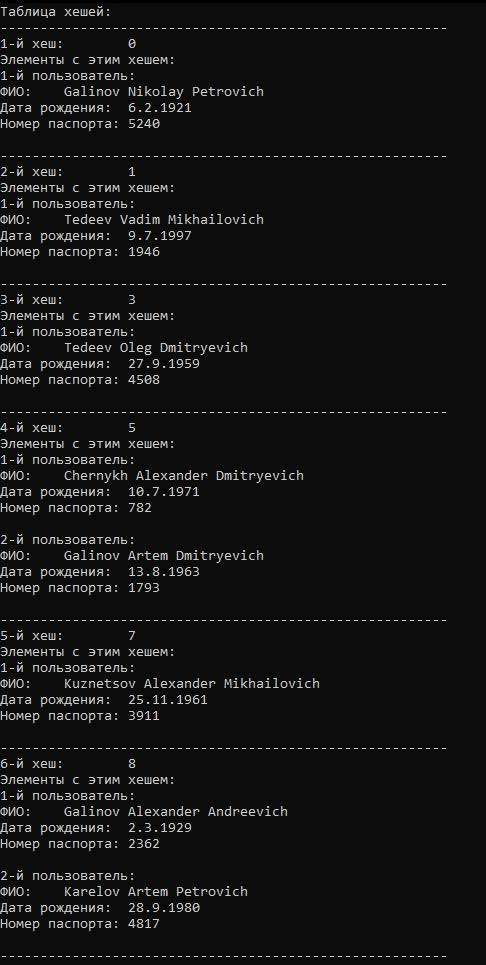
cin >> key;

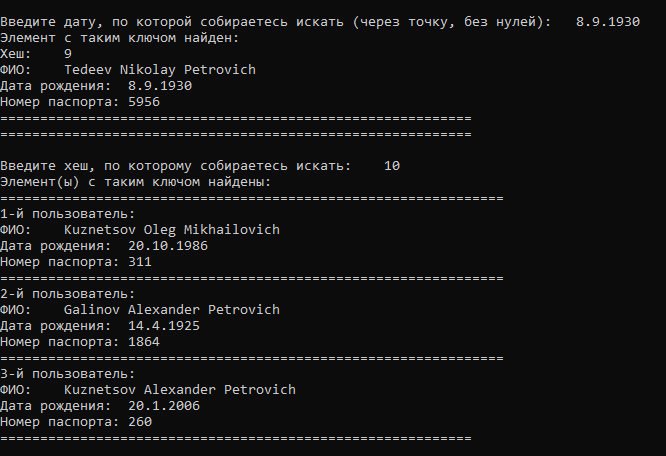
Find\_withHash(hash\_list, key);

}

**Работа кода**







**Блок-схема**

