Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования

Пермский национальный исследовательский политехнический университет

Электротехнический факультет

Кафедра информационных технологий и автоматизированных систем

**ОТЧЕТ**

**Тема:** Реализация Хеш-таблицы методом открытой цепочек

Семестр: 2

Выполнил студент ИВТ-23-2б:

Синицын Владислав Сергеевич

(дата, подпись)

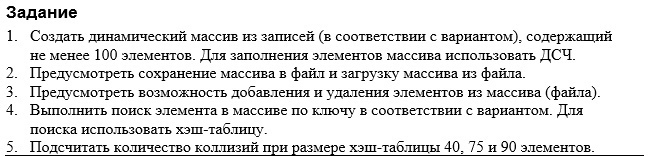
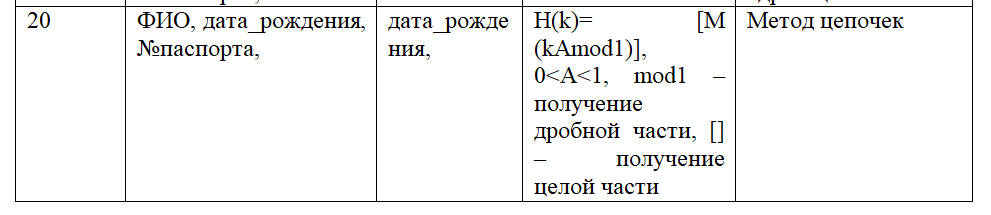
Проверила:

Полякова Ольга Андреевна

(дата, подпись)

Пермь 2024

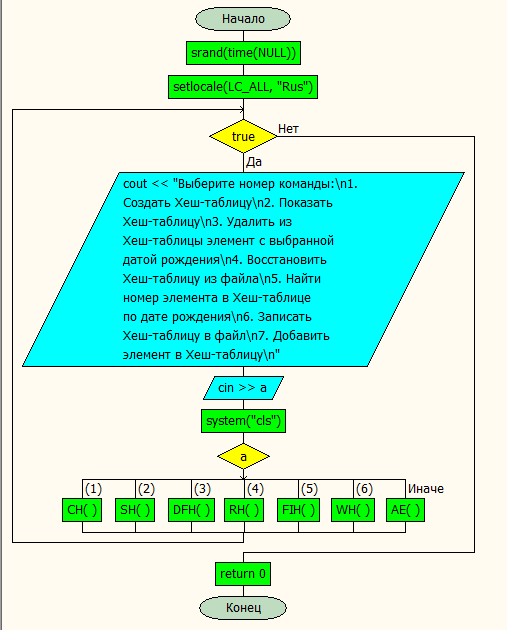
**Задание**

****

**Анализ задачи**

1. Функция CV – замена значений в массиве знаком, значениями из другого массива знаков
2. Функция RNS – датчик случайных чисел
3. Функция GPK – получение возможного ключа
4. Функция CH – создание хеш-таблицы
5. Функция SH – вывод хеш-таблицы
6. Функция DFH – удаление элемента из хеш-таблицы
7. Функция FIH – найти ключ элемента в хеш-таблице
8. Функция AE – добавить элемент в хеш-таблицу
9. Функция RH – восстановление хеш-таблицы
10. Функция WH – запись хеш-таблицы в файл

**Блок-схема**



**Программное решение**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <string>

#include <cstring>

#include <ctime>

#include <fstream>

using namespace std;

string f[5] = {"Синицын", "Мельников", "Серебряков", "Дудашвили", "Куклин"};

string i[5] = { "Тимофей", "Влад", "Егор", "Андрей", "Александр" };

string o[5] = { "Владимирович", "Сергеевич", "Андреевич", "Егорович", "Николаевич" };

string d[5] = { "11.05.01", "01.12.05", "02.04.06", "07.05.06", "03.02.07" };

string p[5] = { "677364", "783783", "777741", "764376", "366553" };

int n = 0;

char date[255];

FILE\* file;

struct Human {

char fio[255];

char dateofbirth[255];

char number[255];

};

Human\* m = new Human;

void CV(char\* c, string r) { //ChangeValue

for (int k = 0; k < r.length(); k++) {

c[k] = r[k];

}

c[r.length()] = '\0';

}

int RNS() { //RandomNuberSensor

return rand() % 5;

}

int GPK() { //GetPosibleKey

double key = 0;

for (int k = 0; k < 9; k += 3) {

string s = "";

s += date[k];

s += date[k + 1];

key += stoi(s);

}

key \*= 0.75;

key -= (int)key;

return (int)(n \* key);

}

void CH() { //CreateHash

cout << "Введите количество элементов: ";

cin >> n;

Human\* t = new Human[n]{};

int collision = 0;

for (int k = 0; k < n; k++) {

CV(date, d[RNS()]);

int l = GPK();

if (t[l].fio[0] != NULL) {

collision++;

}

while (t[l].fio[0] != NULL) {

l = (l + 1) % n;

}

CV(t[l].fio, f[RNS()] + ' ' + i[RNS()] + ' ' + o[RNS()]);

CV(t[l].dateofbirth, date);

CV(t[l].number, p[RNS()]);

}

delete[] m;

m = t;

cout << "Хеш-таблица создана " << "Коллизий: " << collision << '\n';

}

void SH() {

if (n) {

for (int k = 0; k < n; k++) {

cout << m[k].fio << ' ' << m[k].dateofbirth << ' ' << m[k].number << '\n';

}

cout << "-------------------------------\n";

}

else {

cout << "Хеш-таблица пуста!\n";

}

}

void DFH() {//DeleteFromHash

if (n) {

cout << "Введите дату: ";

cin.getline(date, 255);

cin >> date;

for (int k = 0; k < n; k++) {

if (!strcmp(m[k].dateofbirth, date)) {

CV(m[k].fio, "NULL");

CV(m[k].dateofbirth, "NULL");

CV(m[k].number, "NULL");

}

}

}

else {

cout << "Хеш-таблица пуста\n";

}

}

void FIH() {//FindIntoHash

if (n) {

cin.get();

cout << "Введите дату: ";

cin.getline(date, 255);

int l = GPK();

for (int k = 0; k < n; k++) {

if (!strcmp(m[l].dateofbirth, date)) {

cout << l + 1 << '\n';

return;

}

l = (l + 1) % n;

}

cout << "-1\n";

}

else {

cout << "Хеш-таблица пуста, операция не возможна\n";

}

}

void AE() {//AddElement

int l;

cout << "Введите номер нового элемента: ";

cin >> l;

if ((l <= n && strcmp(m[l-1].dateofbirth, "NULL")) || l > n) {

Human\* t = new Human[n >= l ? n + 1 : l];

for (int k = 0; k < l - 1 && k < n; k++) {

t[k] = m[k];

}

for (int k = l; k <= n; k++) {

t[k] = m[k - 1];

}

for (int k = n; k < l; k++) {

CV(t[k].fio, "NULL");

CV(t[k].dateofbirth, "NULL");

CV(t[k].number, "NULL");

}

n = n >= l ? n + 1 : l;

delete[] m;

m = t;

}

CV(m[l - 1].fio, f[RNS()] + ' ' + i[RNS()] + ' ' + o[RNS()]);

CV(m[l - 1].dateofbirth, d[RNS()]);

CV(m[l - 1].number, p[RNS()]);

}

void RH() {//ReadHash

if ((file = fopen("Hash.txt", "rb")) == NULL) {

cout << "Ошибка чтения файла\n";

}

else if (!feof(file)) {

fread(&n, sizeof(n), 1, file);

Human\* t = new Human[n];

for (int k = 0; k < n; k++) {

fread(&t[k], sizeof(Human), 1, file);

}

delete[] m;

m = t;

fclose(file);

}

else {

cout << "Файл пуст, операция не возможна\n";

fclose(file);

}

}

void WH() {

if (n) {

file = fopen("Hash.txt", "wb");

fwrite(&n, sizeof(n), 1, file);

for (int k = 0; k < n; k++) {

fwrite(&m[k], sizeof(Human), 1, file);

}

fclose(file);

}

else {

cout << "Хеш-таблица пуста, операция не возможна\n";

}

}

int main() {

srand(time(NULL));

setlocale(LC\_ALL, "Rus");

int a;

while (true) {

cout << "Выберите номер команды:\n1. Создать Хеш-таблицу\n2. Показать Хеш-таблицу\n3. Удалить из Хеш-таблицы элемент с выбранной датой рождения\n4. Восстановить Хеш-таблицу из файла\n5. Найти номер элемента в Хеш-таблице по дате рождения\n6. Записать Хеш-таблицу в файл\n7. Добавить элемент в Хеш-таблицу\n";

cin >> a;

system("cls");

switch (a) {

case(1):

CH();

break;

case(2):

SH();

break;

case(3):

DFH();

break;

case(4):

RH();

break;

case(5):

FIH();

break;

case(6):

WH();

break;

default:

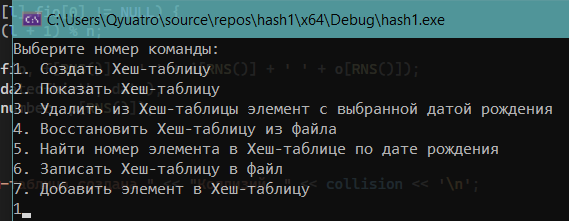
AE();

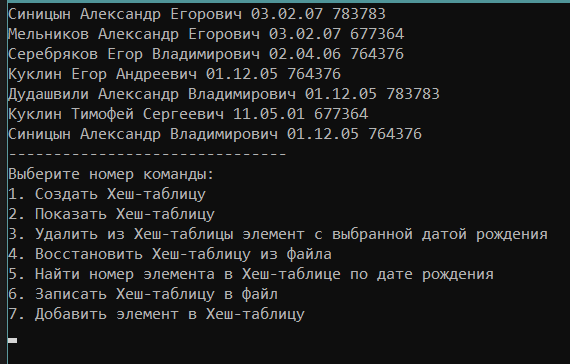
}

}

return 0;

}



****