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)

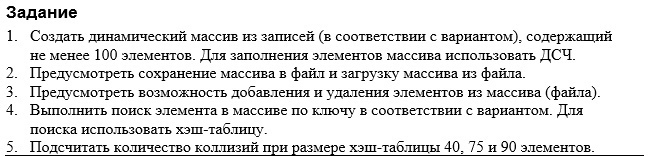
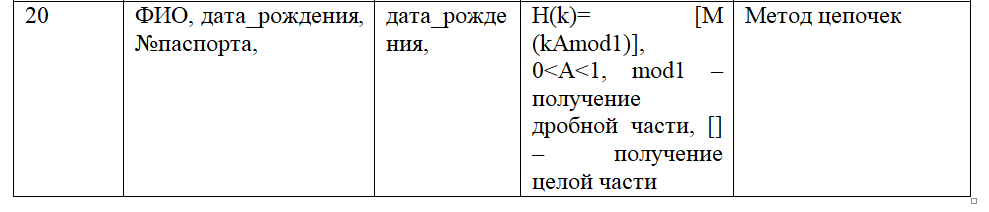
Проверила:

Полякова Ольга Андреевна

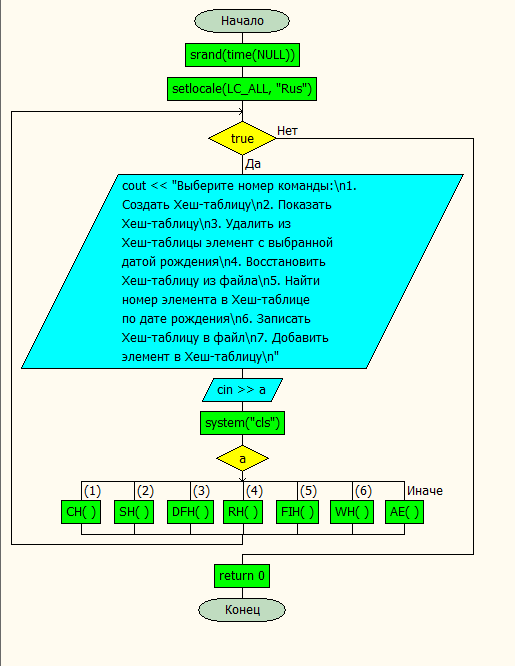
(дата, подпись)

Пермь 2024

**Задание**

****

**Блок-схема**



**Программное решение**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <string>

#include <cstring>

#include <ctime>

using namespace std;

struct Human {

char fio[255];

char dateofbirth[255];

char number[255];

};

struct Node {

Human human;

Node\* next;

};

struct LinkedList {

Node\* last, \* current, \* head;

};

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;

LinkedList\* m = new LinkedList;

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;

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

int collision = 0;

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

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

int l = GPK();

bool occupied = t[l].head != NULL;

t[l].current = t[l].head;

if (occupied) {

while (t[l].current->next != NULL) {

t[l].current = t[l].current->next;

}

}

Node\* q = new Node;

CV(q->human.fio, f[RNS()] + ' ' + i[RNS()] + ' ' + o[RNS()]);

CV(q->human.dateofbirth, date);//работает :)

CV(q->human.number, p[RNS()]);

if (occupied) {

collision++;

t[l].current->next = q;

t[l].current->next->next = NULL;

}

else {

t[l].head = q;

t[l].head->next = NULL;

}

}

delete[] m;

m = t;

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

}

void DFH() {//DeleteFromHash

if (n) {

cout << "Введите дату: ";

cin.getline(date, 255);

cin >> date;

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

m[k].current = m[k].head;

while (m[k].current != NULL) {

if (!strcmp(m[k].current->human.dateofbirth, date)) {

CV(m[k].current->human.fio, "NULL");

CV(m[k].current->human.dateofbirth, "NULL");

CV(m[k].current->human.number, "NULL");

}

m[k].current = m[k].current->next;

}

}

}

else {

cout << "Хеш-таблица пуста!\n";

}

}

void FIH() {//FindIntoHash

if (n) {

cin.get();

cout << "Введите дату: ";

cin.getline(date, 255);

int l = GPK();

int k = 0;

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

m[l].current = m[l].head;

while (m[l].current != NULL) {

if (!strcmp(m[l].current->human.dateofbirth, date)) {

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

return;

}

m[l].current = m[l].current->next;

}

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;

Node\* q = new Node;///генерирую новый элемент

CV(q->human.fio, f[RNS()] + ' ' + i[RNS()] + ' ' + o[RNS()]);//

CV(q->human.dateofbirth, d[RNS()]);//

CV(q->human.number, p[RNS()]);///

if (l > n) {//если новый элемент должен находиться в ещё несуществующем ключе, то нужно изменить размер массива

LinkedList\* t = new LinkedList[l]{};//создаём новый массив

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

t[k] = m[k];//передаём значения из старого

}

delete[] m;

m = t;

n = l;

q->next = NULL;//должен указывать на NULL, так как первый элемент под данным ключом

m[l - 1].head = q;//задаём новый элемент в получивщийся массив, в новую ячейку

}

else if (m[l - 1].head != NULL && strcmp(m[l - 1].head->human.fio, "NULL")) {//если первый элемент под нужным ключом не NULL и не не помечен как удалённый ("NULL")

m[l - 1].current = m[l - 1].head;//устанавливаем текущий на голову

while (m[l - 1].current->next != NULL && strcmp(m[l - 1].current->next->human.fio, "NULL")) {//пока следующий не NULL или "NULL"

m[l - 1].current = m[l - 1].current->next;//получаю следующий

}

if (m[l - 1].current->next == NULL) {//если слудующий NULL

q->next = NULL;//обращаться должен на NULL, так как конец списка

}

else {//если "NULL"

q->next = m[l - 1].current->next->next;//обращается на следующий->следующий элемент

}

m[l - 1].current->next = q;//меняем значение следующего элемента

}

else {//если сразу NULL или "NULL"

if (m[l - 1].head == NULL) {

q->next = NULL;//обращается на NULL, так как конец списка

m[l - 1].head = q;//задаём голове значение

}

else {//если "NULL"

q->next = m[l - 1].head->next;//задаём значение слудующего

m[l - 1].head = q;//задаём значение голове

}

}

}

void SH() {//ShowHash

if (n) {

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

if (m[k].head != NULL) {

cout << "Ключ " << k << ":\n";

m[k].current = m[k].head;

while (m[k].current != NULL) {

cout << m[k].current->human.fio << ' ' << m[k].current->human.dateofbirth << ' ' << m[k].current->human.number << '\n';

m[k].current = m[k].current->next;

}

}

}

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

}

else {

cout << "Хеш-таблица пуста!\n";

}

}

void WH() {//WriteHash

if (n) {

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

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

char space = ' ';

char newline = 'h';

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

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

m[k].current = m[k].head;

while (m[k].current != NULL) {

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

fwrite(&m[k].current->human, sizeof(Human), 1, file);

m[k].current = m[k].current->next;

}

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

}

fclose(file);

}

else {

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

}

}

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);

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

char c;

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

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

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

if (c == ' ') {

Node\* q = new Node;

fread(&q->human, sizeof(Human), 1, file);

t[k].current = t[k].head = q;

t[k].current->next = NULL;

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

while (c != 'h') {

Node\* q = new Node;

fread(&q->human, sizeof(Human), 1, file);

t[k].current->next = q;

t[k].current = t[k].current->next;

t[k].current->next = NULL;

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

}

}

}

delete[] m;

m = t;

fclose(file);

}

else {

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

fclose(file);

}

}

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;

}

