Министерство науки и высшего образования РФ

Федеральное бюджетное государственное образовательное учреждение высшего профессионального образования

Пермский национальный исследовательский политехнический университет (ПНИПУ)

Факультет: Электротехнический (ЭТФ)

Направление: 09.03.01 «Информатика и вычислительная техника (Автоматизированные системы обработки информации и управления)»

Кафедра: «Информационные технологии и автоматизированных систем» (ИТАС)

Теория алгоритмов

Лабораторная работа №11

Поиск данных с помощью хэш-таблиц

Студент: Балтаев Э. А.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата, подпись)

Группа: АСУ-20-1бзу

Работу проверил:

доцент кафедры ИТАС

Полякова О. А.

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Подпись:\_\_\_\_\_\_\_\_\_\_\_

Пермь 2021г.

Постановка задачи:

1. Создать динамический массив из записей (в соответствии с вариантом), содержащий не менее 100 элементов. Для заполнения элементов массива использовать ДСЧ.
2. Предусмотреть сохранение массива в файл и загрузку массива из файла.
3. Предусмотреть возможность добавления и удаления элементов из массива (файла).
4. Выполнить поиск элемента в массиве по ключу в соответствии с вариантом. Для поиска использовать хэш-таблицу.
5. Подсчитать количество коллизий при размере хэш-таблицы 40, 75 и 90 элементов

Задание варианта:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 7 | ФИО, №телефона, адрес | №телефона | H(k)=k mod M | Метод открытой адресации |

Описание класса

1) Персона

#pragma once

#include <iostream>

#include <fstream>

using namespace std;

class Person

{

private:

string fname, sname, tname;

string phone;

string adress;

public:

Person();

Person(string f, string s, string t, string p, string a);

Person(const Person& p);

~Person() {}

bool operator== (const Person& p);

int getHashCode();

static Person\* random();

friend ostream& operator<< (ostream& out, const Person& fin);

friend istream& operator>> (istream& in, Person& fin);

friend fstream& operator<< (fstream& out, const Person& fin);

friend fstream& operator>> (fstream& in, Person& fin);

};

2) Хэш-таблица

#pragma once

#include "Person.h"

class HashTable

{

private:

Person\*\* arr;

int size;

int count\_collision;

int count;

public:

HashTable();

~HashTable();

void clear();

int get\_countCollision() { return count\_collision; }

void init(int n);

void show();

void add();

void del();

void save();

void read();

};

Определение компонентных функций

1) Персона

#include "Person.h"

Person::Person()

{

fname = sname = tname = phone = adress = "";

}

Person::Person(string f, string s, string t, string p, string a)

{

fname = f;

sname = s;

tname = t;

phone = p;

adress = a;

}

Person::Person(const Person& p)

{

fname = p.fname;

sname = p.sname;

tname = p.tname;

phone = p.phone;

adress = p.adress;

}

bool Person::operator==(const Person& p)

{

return p.phone == phone;

}

int Person::getHashCode()

{

int sum = 0;

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

sum = sum + int(phone[k]);

return sum;

}

Person\* Person::random()

{

Person\* p = new Person;

string F[30] = { "Загидуллин","Панкина","Яндиева","Квартирова","Малышева","Набережный","Осинцев","Строганова",

"Клоков","Сиянскиха","Снегирёв","Педича","Яхимовича","Виноградов","Янгосярова","Турчанинова","Крайнева",

"Абоимова","Икрамов","Лисицына","Терещенко","Антипов","Богомолов","Смирновский","Янцижин","Голышева","Дюженкова","Пашина","Корбылева","Болокана" };

p->fname = F[rand() % 30];

string N[40] = { "Эльвира","Владислава","Алина","Герман","Прокл","Всеслава","Кондратий","Валерия","Раиса",

"Борис","Маргарита","Бронислав","Кира","Емельян","Богдан","Пахом","Эрнест","Марина","Майя","Юлия",

"Марк","Никита","Трофим","Прохор","Ульян","Роман","Арсений","Кондратий","Терентий","Владислав","Вера",

"Лада","Ярослава","Ника","Ираида","Наталья","Софья","Пелагея","Роза","Василиса" };

p->sname = N[rand() % 40];

string O[20] = { "Архипович","Кузьмевна","Самсонович","Вячеславович","Юлиевна","Яковович","Сергеевич","Панкратиевич","Кондратиевич","Станиславовна",

"Несторович","Ипполитович","Казимировна","Николаевна","Геннадиевна","Ульянович","Артемиевич","Станиславовна","Сергеевич","Филипповна" };

p->tname = O[rand() % 20];

string Adress[20] = { "Москва","Санкт-Петербург","Новосибирск","Пермь","Нижний-Тагил","Краснодар","Ростов","Смоленск","Тамбов","Казань","Хабаровск",

"Владивосток","Самара","Тюмень","Норильск","Уфа","Челябинск","Воронеж","Сочи","Махачкала" };

p->adress = Adress[rand() % 20];

char nums[10] = { '0','1','2','3','4','5','6','7','8','9' };

p->phone = "8";

for (int i = 1; i < 11; i++)

p->phone += nums[rand() % 10];

return p;

}

ostream& operator<<(ostream& out, const Person& fin)

{

out << fin.fname << " " << fin.sname << " " << fin.tname << " " << fin.phone << " " << fin.adress;

return out;

}

istream& operator>>(istream& in, Person& fin)

{

cout << "ФИО? "; in >> fin.fname >> fin.sname >> fin.tname;

cout << "Телефон? "; in >> fin.phone;

cout << "Адрес? "; in >> fin.adress;

return in;

}

fstream& operator<<(fstream& out, const Person& fin)

{

out << fin.fname << " " << fin.sname << " " << fin.tname << " " << fin.phone << " " << fin.adress;

return out;

}

fstream& operator>>(fstream& in, Person& fin)

{

in >> fin.fname >> fin.sname >> fin.tname >> fin.phone >> fin.adress;

return in;

}

2) Хэш-таблица

#include "HashTable.h"

HashTable::HashTable()

{

arr = 0;

count = 0;

size = 0;

count\_collision = 0;

}

HashTable::~HashTable()

{

clear();

}

void HashTable::clear() {

if (count == 0)

return;

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

delete arr[i];

arr[i] = 0;

}

count\_collision = 0;

count = 0;

}

void HashTable::init(int n)

{

clear();

delete[] arr;

size = n;

arr = new Person \* [size];

// обнуляем всё

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

arr[i] = 0;

}

void HashTable::show()

{

if (count == 0)

cout << "пусто" << endl;

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

// если не пусто

if (arr[i] != 0)

cout << i + 1 << ") " << \*arr[i] << endl;

cout << endl;

}

void HashTable::add()

{

if (size == 0)

throw exception("Еще не создано");

int n;

cout << "N? "; cin >> n;

if (n < 1)

throw exception("неверное значение");

if (count + n > size)

throw exception("слишком много значений");

count += n;

Person\* p = 0;

int hashInd;

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

p = Person::random();

hashInd = p->getHashCode() % size;

// свободно

if (arr[hashInd] == 0)

arr[hashInd] = p;

// занято

else {

count\_collision++;

for (int j = i + 1; j < size; j++) {

// если нашли свободное место

if (arr[j] == 0) {

arr[j] = p;

break;

}

// если прошли круг

if ((j + 1) == i)

throw exception("Мест больше нет");

// если доходим до конца - встаем в начало

if ((j + 1) == size)

j = -1;

}

}

}

cout << "Успешно" << endl;

}

void HashTable::del()

{

if (size == 0)

throw exception("Еще не создано");

string key;

cout << "key? "; cin >> key;

Person p("", "", "", key, "");

int hashInd = p.getHashCode() % size;

if (arr[hashInd] != 0 && \*arr[hashInd] == p) {

cout << "Нашли с первого раза\n";

arr[hashInd] = 0;

count--;

return;

}

else {

for (int j = hashInd + 1, c = 1; j < size; j++, c++) {

// если нашли нужный

if (arr[j] != 0 && \*arr[j] == p) {

cout << "Нашли после " << c << " шагов на " << j + 1 << " номере" << endl;

arr[hashInd] = 0;

count--;

break;

}

// если доходим до конца - встаем в начало

if ((j + 1) == size)

j = -1;

// если прошли круг

if ((j + 1) == hashInd)

throw exception("Нет такого ключа");

}

}

}

void HashTable::save()

{

if (size == 0)

throw exception("Еще не создано");

fstream f("f.bin", ios::out);

f << count << endl;

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

if (arr[i] != 0)

f << \*arr[i] << endl;

f.close();

cout << "Успешно" << endl;

}

void HashTable::read()

{

clear();

fstream f("f.bin", ios::in);

f >> count;

init(count);

Person p;

int i = 0;

do {

f >> p;

if (f.eof())

break;

Person\* temp = new Person(p);

arr[i++] = temp;

} while (!f.eof());

f.close();

if (i == 0)

cout << "пусто";

else {

cout << "Успешно" << endl;

count = i;

}

}

Определение главного файла

#include <iostream>

#include "HashTable.h"

int main()

{

system("chcp 1251");

srand(time(0));

int x;

HashTable table;

while (true) {

try {

cout << "\n1. Создать";

cout << "\n2. Добавить";

cout << "\n3. Удалить";

cout << "\n4. Посмотреть";

cout << "\n5. Количество коллизий";

cout << "\n6. Очистить";

cout << "\n7. Сохранить";

cout << "\n8. Загрузить";

cout << "\n9. Выйти\n>";

cin >> x;

switch (x) {

case 1: cout << "N? "; cin >> x; table.init(x); cout << "Успешно" << endl; break;

case 2: table.add(); break;

case 3: table.del(); break;

case 4: table.show(); break;

case 5: cout << "Значение = " << table.get\_countCollision(); break;

case 6: table.clear(); cout << "Успешно\n"; break;

case 7: table.save(); break;

case 8: table.read(); break;

case 9: return 0;

}

}

catch (exception& ex) {

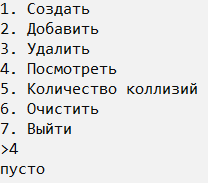
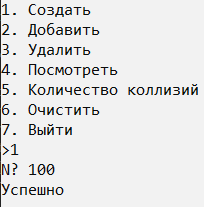
cout << "Ошибка: " << ex.what() << endl;

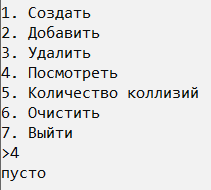
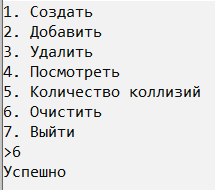
}

}

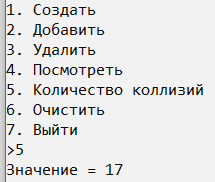
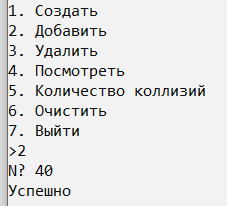
}

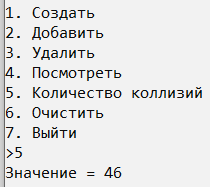
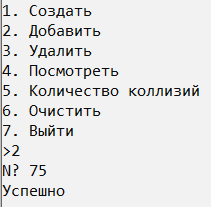
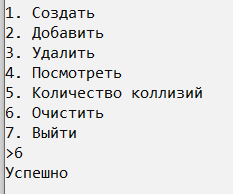
Результаты программы:

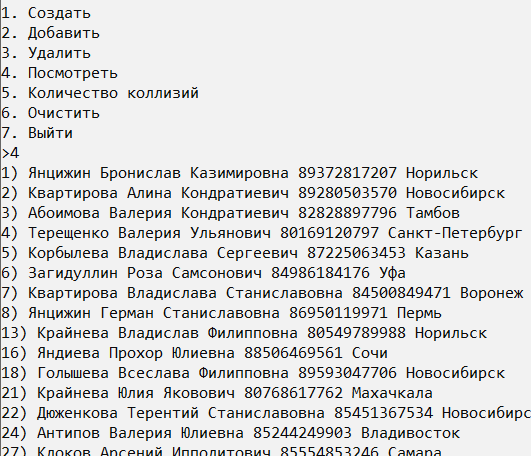


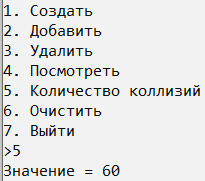
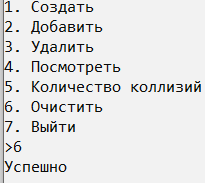


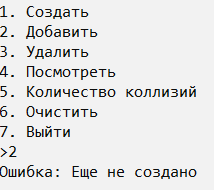
// Количество коллизий при разных значениях



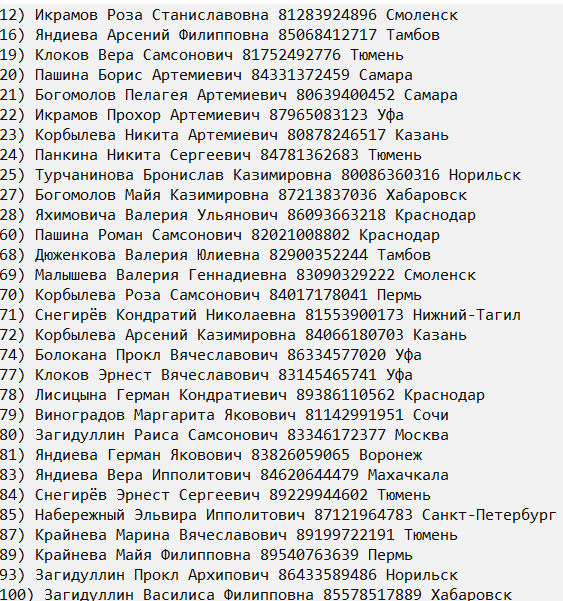


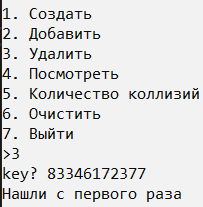
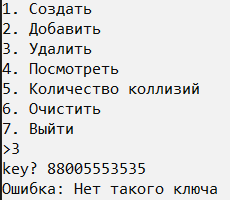






// Удаление





// Сохранение и загрузка

