Министерство образования РФ

Пермский национальный исследовательский политехнический университет

Кафедра ИТАС

ЛАБОРАТОРНАЯ РАБОТА № 11

ПО ИНФОРМАТИКЕ ЗА I СЕМЕСТР

Вариант 5

|  |
| --- |
| Выполнил студент:  Отинов Иван Юрьевич  Группа РИС-20-1бз  Шифр 20-ЭТФ-635  Кафедра ИТАС:  Полякова Ольга Андреевна |

ПЕРМЬ 2020

# Лабораторная работа №11 "Поиск данных с помощью хэш-таблиц”

**Вариант №5**

**Цель:**

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

**Задача:**

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.

**Вариант задания:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 5 | ФИО, №паспорта, адрес | ФИО | H(k)=k mod M | Метод цепочек |

**Текст программы**

#include <iostream>

#include <fstream>

#include <string>

#include <vector>

using namespace std;

struct Node {

string name;

string pasport;

};

struct Bucket {

int hash;

vector<Node> buckets;

};

void printHashTable(vector<Bucket>& hashTable);

void insert(vector<Bucket>& hashTable, string name, string pasport);

void remove(vector<Bucket>& hashTable, string key);

void find(vector<Bucket>& hashTable, string key);

void loadFromFile(char\* databaseName, vector<Bucket>& hashTable);

void save(char\* fileName, vector<Bucket>& hashTable);

void countConflict(vector<Bucket>& hashTable);

int main() {

vector<Bucket> hashTable;

int size = 20;

char fileName[] = "D:\\lab11.txt";

for (int i = 0; i < 20; i++) {

Bucket\* bucket = new Bucket();

bucket->hash = i;

hashTable.push\_back(\*bucket);

}

int help;

loadFromFile(fileName, hashTable);

printHashTable(hashTable);

cout << "\nInput N from insert: ";

cin >> help;

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

string name, pasport;

cout << "\nCreate new Record\nInput name: ";

cin >> name;

cout << "Input pasport: ";

cin >> pasport;

insert(hashTable, name, pasport);

}

printHashTable(hashTable);

cout << "\nInput N from delete: ";

cin >> help;

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

string key;

cout << "\nDelete Record\nInput key: ";

cin >> key;

remove(hashTable, key);

}

printHashTable(hashTable);

cout << "\nInput N from search: ";

cin >> help;

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

string key;

cout << "\n\Search Record\nInput key: ";

cin >> key;

find(hashTable, key);

}

countConflict(hashTable);

save(fileName, hashTable);

}

void insert(vector<Bucket> &hashTable, string name, string pasport) {

Node\* node = new Node();

node->name = name;

node->pasport = pasport;

int hash = (node->name.length()) % (hashTable.size());

int isContains = 0;

for (int i = 0; i < hashTable.at(hash).buckets.size(); i++) {

if (hashTable.at(hash).buckets.at(i).name == name) {

isContains = 1;

break;

}

}

if (!isContains) {

hashTable.at(hash).buckets.push\_back(\*node);

}

}

void remove(vector<Bucket> &hashTable, string key) {

int hash = key.length() % 20;

for (int i = 0; i < hashTable.at(hash).buckets.size(); i++) {

if (key == hashTable.at(hash).buckets.at(i).name) {

hashTable.at(hash).buckets.erase(hashTable.at(hash).buckets.begin() + i);

}

}

}

void find(vector<Bucket>& hashTable, string key) {

int hash = key.length() % 20;

for (int i = 0; i < hashTable.at(hash).buckets.size(); i++) {

if (key == hashTable.at(hash).buckets.at(i).name) {

cout << " { " << hashTable.at(hash).buckets.at(i).name << " : " << hashTable.at(hash).buckets.at(i).pasport << " }";

}

}

}

void printHashTable(vector<Bucket>& hashTable) {

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

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

cout << "\nHASH = " << hashTable.at(i).hash << " ";

for (int j = 0; j < hashTable.at(i).buckets.size(); j++) {

cout << " { " << hashTable.at(i).buckets.at(j).name << " : " << hashTable.at(i).buckets.at(j).pasport << " } ->";

}

}

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

}

void loadFromFile(char\* fileName, vector<Bucket>& hashTable) {

ifstream in(fileName);

string line, name, pasport;

int count = 0;

if (in.is\_open()) {

while (getline(in, line)) {

if (count == 0) {

name = line;

count++;

} else {

pasport = line;

count = 0;

insert(hashTable, name, pasport);

}

}

}

in.close();

}

void save(char\* fileName, vector<Bucket>& hashTable) {

ofstream out(fileName, ios::trunc);

if (out.is\_open()) {

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

for (int j = 0; j < hashTable.at(i).buckets.size(); j++) {

out << hashTable.at(i).buckets.at(j).name << endl;

out << hashTable.at(i).buckets.at(j).pasport << endl;

}

}

}

out.close();

}

void countConflict(vector<Bucket>& hashTable) {

int count = 0;

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

if (hashTable.at(i).buckets.size() > 1) {

count += hashTable.at(i).buckets.size() - 1;

}

}

cout << "\nFind conflicts: " << count << "\n";

}

**Результаты работы программы:**

