Министерство высшего образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«Пермский национальный исследовательский политехнический университет» (ПНИПУ)**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

ОТЧЁТ

по лабораторной работе

Тема: «Поиск данных с помощью хэш-таблиц»

Выполнил

Студент группы РИС-22-2б

Скорюпин Д.А.

Проверил доц. Кафедры ИТАС

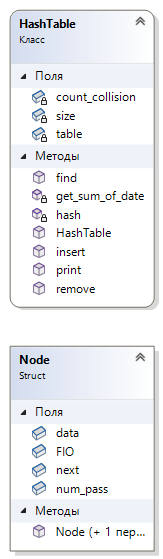
Полякова Ольга Андреевна

Пермь 2023

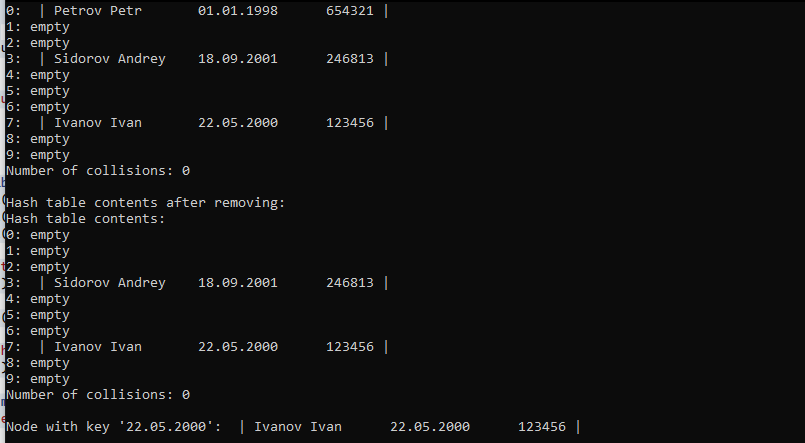
# Постановка задачи

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

**UML таблица**



**Скриншоты работы программы**



**Код программы**

#include <iostream>

#include <vector>

#include <string>

#include <fstream>

#include <stack>

using namespace std;

struct Node {

string FIO;

string data;

long int num\_pass;

Node\* next;

Node(string fio, string dat, int num) : FIO(fio), data(dat), num\_pass(num), next(nullptr) {}

Node(string data) : data(data), FIO("NULL"), num\_pass(0), next(nullptr) {}

friend ostream& operator<<(ostream& out, const Node& node)

{

if (node.data == "not find") { out << "not find"; }

else if (node.data == "delete") { out << "delete"; }

else

out << " | " << node.FIO << "\t" << node.data << "\t" << node.num\_pass << " | ";

return out;

}

};

class HashTable {

private:

vector<Node\*> table;

int size;

int count\_collision = 0;

int hash(string str)

{

double a = ((sqrt(5) - 1) / 2) \* get\_sum\_of\_date(str);

double c = size \* (a - int(a));

return int(c);

}

int get\_sum\_of\_date(string str)

{

return stoi(str.substr(0, 2)) + stoi(str.substr(3, 2)) + stoi(str.substr(6, 4));

}

public:

HashTable(int s) : size(s)

{

table.resize(size, nullptr);

}

void insert(string FIO, string key, int num) {

int index = hash(key);

Node\* newNode = new Node(FIO, key, num);

if (table[index] == nullptr) {

table[index] = newNode;

}

else {

count\_collision++;

Node\* current = table[index];

while (current->next != nullptr) {

current = current->next;

}

current->next = newNode;

}

}

void remove(string key) {

int index = hash(key);

if (table[index] == nullptr) {

return;

}

else if (table[index]->data == key) {

Node\* temp = table[index];

table[index] = temp->next;

delete temp;

}

else {

Node\* current = table[index];

while (current->next != nullptr && current->next->data != key) {

current = current->next;

}

if (current->next != nullptr && current->next->data == key) {

Node\* temp = current->next;

current->next = temp->next;

delete temp;

}

}

}

Node find(string key) {

int index = hash(key);

Node\* current = table[index];

while (current != nullptr) {

if (current->data == key) {

return \*current;

}

current = current->next;

}

return Node("not find");

}

void print() {

cout << "Hash table contents:\n";

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

cout << i << ": ";

if (table[i] == nullptr) {

cout << "empty\n";

}

else {

Node\* current = table[i];

while (current != nullptr) {

cout << \*current;

current = current->next;

}

cout << "\n";

}

}

cout << "Number of collisions: " << count\_collision << "\n";

}

};

int main() {

HashTable myTable(10);

myTable.insert("Ivanov Ivan", "22.05.2000", 123456);

myTable.insert("Petrov Petr", "01.01.1998", 654321);

myTable.insert("Sidorov Andrey", "18.09.2001", 246813);

cout << "Hash table contents after inserting:\n";

myTable.print();

myTable.remove("01.01.1998");

cout << "\nHash table contents after removing:\n";

myTable.print();

Node result = myTable.find("22.05.2000");

cout << "\nNode with key '22.05.2000': " << result << endl;

return 0;

}