Министерство образования РФ

Пермский государственный технический университет

Кафедра ИТАС

ЛАБОРАТОРНАЯ РАБОТА № 11

ПО ИНФОРМАТИКЕ ЗА I СЕМЕСТР

Вариант 9

Выполнил студент:

Главатских Максим Николаевич

Группа РИС-20-1бз

Шифр 20-ЭТФ-659

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:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 9 | ФИО, №телефона, адрес | Адрес | H(k)=k mod M | Метод цепочек |

**Текст программы**

#include <vector> // Lib importing for storage <Node> objects

#include <iostream> // Lib for work with information out

using namespace std; // For using cout, endl, vector withoud prefix std::

#define DEFAULT 256 // Declare size of HashTable - YOU CAN SET IT AS YOU WANT

class Node

{

friend class HashTable;

Node\* next;

int data;

public:

Node()

{}

Node(int d, Node\* n)

{

data = d;

next = n;

}

~Node()

{}

};

class HashTable

{

public:

int size;

vector<Node\*>\* Nodes;

HashTable(int s)

{

size = s;

Nodes = new vector<Node\*>(size);

}

HashTable()

{

size = DEFAULT;

Nodes = new vector<Node\*>(size);

}

~HashTable()

{}

void insert(int data)

{

int index = getHashByDivision(data);

if (Nodes->at(index) == NULL)

{

Nodes->at(index) = new Node(data, NULL);

return;

}

Node\* p = Nodes->at(getHashByDivision(data));

while (p != NULL) {

if (p->next == NULL)

{

p->next = new Node(data, NULL);

break;

}

p = p->next;

}

}

int getSize()

{

return size;

}

int getHashByDivision(int key)

{

return (key % getSize());

}

void print(int start)

{

int lineCount = 0;

for (int i = start; i < Nodes->size(); i++) {

cout << i << "\t";

Node\* p = Nodes->at(i);

while (p != NULL) {

cout << p->data << " ";

p = p->next;

}

cout << endl;

}

}

};

int main(int argc, const char\* argv[]) {

int n = 10;

int A[] = { 40, 12, 79, 35, 43, 52, 83, 66, 89, 79 };

HashTable\* hash = new HashTable(n);

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

hash->insert(A[i]);

}

hash->print(0);

system("pause");

return 0;

}

