#ifndef HASHNODE\_H

#define HASHNODE\_H

#include<string>

using namespace std;

class hashNode

{

public:

hashNode();

hashNode(string fullName, string stuId, string stuEmail, string phoneNum);

virtual ~hashNode();

string getFullName(){ return fullName;}

string getStudentID(){ return stuId; }

string getStudentEmail(){ return stuEmail;}

string getPhoneNumber(){ return phoneNum;}

void setNext(hashNode \*val){ next = val;}

hashNode\* getNext(){ return next; }

protected:

private:

// key: fullName, value: stuId, stuEmail, phoneNum

string fullName;

string stuId;

string stuEmail;

string phoneNum;

hashNode \*next;

};

#endif // HASHNODE\_H

#include "hashNode.h"

hashNode::hashNode()

{

//ctor

}

hashNode::hashNode(string fullName, string stuId, string stuEmail, string phoneNum)

{

this->fullName = fullName;

this->stuId = stuId;

this->stuEmail = stuEmail;

this->phoneNum = phoneNum;

this->next = nullptr;

}

hashNode::~hashNode()

{

//dtor

}

#ifndef HASHTABLE\_H

#define HASHTABLE\_H

#include "hashNode.h"

class hashTable

{

public:

hashTable(int sz);

virtual ~hashTable();

void insertNode(hashNode \*p);

bool deleteNode(string fullName);

hashNode\* searchNode(string fullName);

void display();

protected:

private:

hashNode \*\*table; // dung de quan ly mang con tro hashNode

int TABLE\_SIZE;

int hashFunc(string fullName);

};

#endif // HASHTABLE\_H

#include "hashTable.h"

#include <iostream>

#include<iomanip>

using namespace std;

hashTable::hashTable(int sz) : TABLE\_SIZE(sz)

{

//ctor

table = new hashNode \* [TABLE\_SIZE];

for(int i = 0; i < TABLE\_SIZE; i++)

{

table[i] = nullptr;

}

}

hashTable::~hashTable()

{

//dtor

}

int hashTable::hashFunc(string fullName)

{

int sum = 0;

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

sum += (int)tolower(fullName[i]);

return sum % TABLE\_SIZE;

}

void hashTable::insertNode(hashNode \*p)

{

int hashIndex = hashFunc(p->getFullName());

hashNode \*tmp = p;

// insert first linked list

// không cần kiểm tra dsach rỗng(head == nulll) như dslk vì hashtable không có tail

// nên ko cần cập nhập tail trong điều kiện danh sách rỗng

tmp->setNext(table[hashIndex]);

table[hashIndex] = tmp;

}

bool hashTable::deleteNode(string fullName)

{

int hashIndex = hashFunc(fullName);

// Xoa ptu dau

if(table[hashIndex]->getFullName().compare(fullName) == 0)

{

hashNode \*p = table[hashIndex];

table[hashIndex] = table[hashIndex]->getNext();

delete p;

return true;

}

// Xoa ptu giua va cuoi

hashNode \*prev = table[hashIndex];

hashNode \*curr = table[hashIndex]->getNext();

while(curr != nullptr)

{

if(curr->getFullName().compare(fullName) == 0)

{

prev->setNext(curr->getNext());

delete curr;

return true;

}

prev = curr;

curr = curr->getNext();

}

return false;

}

hashNode\* hashTable::searchNode(string fullName)

{

int hashIndex = hashFunc(fullName);

hashNode\* curr = table[hashIndex];

while(curr != nullptr)

{

if(curr->getFullName().compare(fullName) == 0)

{

return curr;

}

curr = curr->getNext();

}

return nullptr;

}

void hashTable::display()

{

// Traverse the entire hash table

for (int i=0; i < TABLE\_SIZE; ++i) {

// cout << " +--------+--------+" << endl;

cout << i << " |";

hashNode\* p = table[i];

if (p == NULL ) {

// NULL record, print empty

cout << "[" << setw(6) << "" << " ]";// | " << setw(6) << "" << " |";

}

if(p != nullptr) {

// Print the record from the table

cout << "[ " << p->getFullName() << " | " << p->getStudentEmail() << ", " << p->getPhoneNumber() << ", " << p->getStudentID() << " ]";

// Traverse and print the chain

for (p = p->getNext(); p != nullptr ; p = p->getNext()) {

cout << " --> " << "[ " << p->getFullName() << " | " << p->getStudentEmail() << ", " << p->getPhoneNumber() << ", " << p->getStudentID() << " ]";

}

}

cout << endl;

}

cout << endl;

}

#include <iostream>

#include"hashNode.h"

#include"hashTable.h"

using namespace std;

int main()

{

hashTable \*st = new hashTable(20);

st->insertNode(new hashNode("Lan", "001", "lan@gmail.com", "0922222222"));

st->insertNode(new hashNode("Phong", "002", "phong@gmail.com", "093333333"));

st->insertNode(new hashNode("Thuy", "003", "thuy@gmail.com", "098273732"));

st->insertNode(new hashNode("Ha", "004", "ha@gmail.com", "0944444444"));

st->insertNode(new hashNode("Anh", "005", "anh@gmail.com", "0955555555"));

st->insertNode(new hashNode("Ban", "006", "ban@gmail.com", "0966666666"));

st->insertNode(new hashNode("Aan", "007", "aan@gmail.com", "0966666666"));

st->insertNode(new hashNode("Cuong", "008", "cuong@gmail.com", "097777777"));

st->insertNode(new hashNode("Khag", "009", "khag@gmail.com", "098888888"));

st->insertNode(new hashNode("Yem", "010", "yem@gmail.com", "09999999999"));

st->display();

st->deleteNode("Lan");

st->display();

cout << st->searchNode("Phong")->getStudentEmail();

return 0;

}