THỰC HÀNH CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

Họ tên: Khương Thanh Bình

MSSV:3123411034

Bài Tập Danh sách liên kết đơn (tt)

**Câu 1:**

(a) Cài đặt CTDL ListInt dùng để chứa các số nguyên, trong đó:

• Sử dụng danh sách liên kết

• Cài đặt các phép toán: khởi tạo, hủy, thêm phần tử, xóa phần tử, thêm một danh sách phần tử, xuất danh sách

CODE

#include <iostream>

using namespace std;

typedef struct Node { //Tạo con trỏ

int info;

Node\* next;

} Node;

struct ListInt { //Tạo danh sách liên kết

Node\* pHead;

// Khởi tạo danh sách rỗng

ListInt() { pHead = NULL; }

// Thêm số nguyên vào cuối danh sách

void themSo(int x) {

Node\* p = new Node;

p->info = x;

p->next = NULL;

if (pHead == NULL) {

pHead = p;

}

else {

Node\* q = pHead;

while (q->next != NULL) {

q = q->next;

}

q->next = p;

}

}

// Xuất danh sách số nguyên

void xuatDanhSach() {

Node\* p = pHead;

while (p) {

cout << p->info << " ";

p = p->next;

}

cout << endl;

}

// Xóa một số k trong danh sách

void xoaSo(int k) {

while (pHead && pHead->info == k) {

Node\* p = pHead;

pHead = pHead->next;

delete p;

}

Node\* p = pHead;

while (p && p->next) {

if (p->next->info == k) {

Node\* nodeCanXoa = p->next;

p->next = p->next->next;

delete nodeCanXoa;

}

else {

p = p->next;

}

}

}

// Thêm một danh sách khác vào danh sách hiện tại

void themDanhSach(ListInt& dsKhac) {

if (dsKhac.pHead == NULL) return;

if (pHead == NULL) {

pHead = dsKhac.pHead;

}

else {

Node\* p = pHead;

while (p->next != NULL) {

p = p->next;

}

p->next = dsKhac.pHead;

}

dsKhac.pHead = NULL; // Tránh trùng lặp vùng nhớ

}

// Giải phóng bộ nhớ danh sách

void xoaDanhSach() {

while (pHead) {

Node\* p = pHead;

pHead = pHead->next;

delete p;

}

}

};

(b) Viết thành chương trình:

• Nhập 10 số nguyên và đưa vào danh sách

• In danh sách ra màn hình

• Nhập một số k và xóa số k trong danh sách

• In danh sách sau khi xóa phần tử

• Nhập 5 số nguyên vào một danh sách thứ hai

• Thêm danh sách thứ hai vào danh sách thứ nhất

• In danh sách thứ nhất ra màn hình

CODE

int main() {

ListInt danhSach1, danhSach2;

int so, k;

// Nhập 10 số nguyên vào danh sách 1

cout << "Nhap 10 so nguyen: ";

for (int i = 0; i < 10; i++) {

cin >> so;

danhSach1.themSo(so);

}

// Xuất danh sách 1

cout << "Danh sach vua nhap: ";

danhSach1.xuatDanhSach();

// Nhập một số k và xóa khỏi danh sách

cout << "Nhap so can xoa: ";

cin >> k;

danhSach1.xoaSo(k);

// Xuất danh sách sau khi xóa

cout << "Danh sach sau khi xoa " << k << ": ";

danhSach1.xuatDanhSach();

// Nhập 5 số nguyên vào danh sách 2

cout << "Nhap 5 so nguyen cho danh sach 2: ";

for (int i = 0; i < 5; i++) {

cin >> so;

danhSach2.themSo(so);

}

// Thêm danh sách 2 vào danh sách 1

danhSach1.themDanhSach(danhSach2);

// Xuất danh sách sau khi ghép

cout << "Danh sach sau khi them danh sach 2: ";

danhSach1.xuatDanhSach();

danhSach1.xoaDanhSach();//Giải phóng bộ nhớ

danhSach2.xoaDanhSach();

return 0;

}

TEST CODE

Kết quả:

A number on a black background

AI-generated content may be incorrect.

**Câu 2:** (a) Cài đặt CTDL SinhVien để quản lý một sinh viên gồm có: họ tên (50 ký tự), địa chỉ (70 ký tự), lớp (10 ký tự), khóa (số nguyên). Cài đặt 4 hàm so sánh 2 sinh viên theo từng tiêu chí. Viết hàm nhập một sinh viên, hàm xuất một sinh viên.

CODE

#include <iostream>

#include <cstring>

using namespace std;

typedef struct SinhVien {

char hoTen[50];

char diaChi[70];

char lop[10];

int khoa;

SinhVien\* next;

} SinhVien

// Hàm so sánh theo tên

bool soSanhTheoTen(SinhVien\* a, SinhVien\* b) {

return strcmp(a->hoTen, b->hoTen) < 0;

}

// Hàm so sánh theo địa chỉ

bool soSanhTheoDiaChi(SinhVien\* a, SinhVien\* b) {

return strcmp(a->diaChi, b->diaChi) < 0;

}

// Hàm so sánh theo lớp

bool soSanhTheoLop(SinhVien\* a, SinhVien\* b) {

return strcmp(a->lop, b->lop) < 0;

}

// Hàm so sánh theo khóa

bool soSanhTheoKhoa(SinhVien\* a, SinhVien\* b) {

return a->khoa < b->khoa;

}

(b) Cài đặt CTDL ListSV dùng để chứa các sinh viên, trong đó:

• Sử dụng danh sách liên kết

• Cài đặt các phép toán: khởi tạo, hủy, thêm phần tử, xóa phần tử, thêm một danh sách phần tử, xuất danh sách, sắp xếp danh sách sử dụng selection sort và con trỏ hàm so sánh.

CODE

struct ListSV {

SinhVien\* head;

// Khởi tạo danh sách rỗng

ListSV() { head = NULL; }

// Hàm nhập một sinh viên

void nhapSV(SinhVien& sv) {

cout << "Nhap ho ten: ";

cin.getline(sv.hoTen, 50);

cout << "\nNhap dia chi: ";

cin.getline(sv.diaChi, 70);

cout << "\nNhap lop: ";

cin.getline(sv.lop, 10);

cout << "\nNhap khoa: ";

cin >> sv.khoa;

}

// Hàm xuất một sinh viên

void xuatSV(SinhVien\* sv) {

cout << "\nHo ten: " << sv->hoTen << "\nDia chi: " << sv->diaChi

<< "\nLop: " << sv->lop << "\nKhoa: " << sv->khoa << endl;

}

// Thêm sinh viên vào danh sách

void themSV(const char\* ten, const char\* diachi, const char\* lop, int khoa) {

SinhVien\* p = new SinhVien;

strcpy(p->hoTen, ten); //Hàm sao chép chuỗi

strcpy(p->diaChi, diachi);

strcpy(p->lop, lop);

p->khoa = khoa;

p->next = NULL;

if (head == NULL) {

head = p;

}

else {

SinhVien\* q = head;

while (q->next != NULL) {

q = q->next;

}

q->next = p;

}

}

// Xuất danh sách sinh viên

void xuatDanhSach() {

SinhVien\* p = head;

while (p) {

xuatSV(p);

p = p->next;

}

cout << endl;

}

// Xóa sinh viên theo tên

void xoaSVTheoTen(const char\* ten) {

while (head && strcmp(head->hoTen, ten) == 0) {

SinhVien\* p = head;

head = head->next;

delete p;

}

SinhVien\* p = head;

while (p && p->next) {

if (strcmp(p->next->hoTen, ten) == 0) {

SinhVien\* nodeCanXoa = p->next;

p->next = p->next->next;

delete nodeCanXoa;

}

else {

p = p->next;

}

}

}

// Xóa sinh viên theo địa chỉ

void xoaSVTheoDiaChi(const char\* diachi) {

while (head && strcmp(head->diaChi, diachi) == 0) {

SinhVien\* p = head;

head = head->next;

delete p;

}

SinhVien\* p = head;

while (p && p->next) {

if (strcmp(p->next->diaChi, diachi) == 0) {

SinhVien\* nodeCanXoa = p->next;

p->next = p->next->next;

delete nodeCanXoa;

}

else {

p = p->next;

}

}

}

// Sắp xếp danh sách bằng selection sort

void sapXep(bool (ListSV::\* cmp)(SinhVien\*, SinhVien\*)) {

for (SinhVien\* p = head; p != NULL; p = p->next) {

SinhVien\* min = p;

for (SinhVien\* q = p->next; q != NULL; q = q->next) {

if ((this->\*cmp)(q, min)) {

min = q;

}

}

if (min != p) {

swap(\*p, \*min);

swap(p->next, min->next); // Giữ nguyên liên kết

}

}

}

(c) Viết thành chương trình:

• Nhập 10 sinh viên và đưa vào danh sách

• In danh sách ra màn hình

• Xóa sinh viên có tên “Nguyen Van Teo” trong danh sách

• Xóa sinh viên có địa chỉ “Nguyen Van Cu” trong danh sách

• Thêm sinh viên có tên “Tran Thi Mo”, địa chỉ “25 Hong Bang”, lớp “TT0901”, khóa 2009 vào danh sách.

• In danh sách ra màn hình

CODE

#include <iostream>

#include <cstring>

using namespace std;

typedef struct SinhVien {

char hoTen[50];

char diaChi[70];

char lop[10];

int khoa;

SinhVien\* next;

} SinhVien;

struct ListSV {

SinhVien\* head;

// Khởi tạo danh sách rỗng

ListSV() { head = NULL; }

// Hàm nhập một sinh viên

void nhapSV(SinhVien& sv) {

cout << "Nhap ho ten: ";

cin.getline(sv.hoTen, 50);

cout << "\nNhap dia chi: ";

cin.getline(sv.diaChi, 70);

cout << "\nNhap lop: ";

cin.getline(sv.lop, 10);

cout << "\nNhap khoa: ";

cin >> sv.khoa;

}

// Hàm xuất một sinh viên

void xuatSV(SinhVien\* sv) {

cout << "\nHo ten: " << sv->hoTen << " Dia chi: " << sv->diaChi

<< " Lop: " << sv->lop << " Khoa: " << sv->khoa << endl;

}

// Thêm sinh viên vào danh sách

void themSV(const char\* ten, const char\* diachi, const char\* lop, int khoa) {

SinhVien\* p = new SinhVien;

strcpy(p->hoTen, ten); //Hàm sao chép chuỗi

strcpy(p->diaChi, diachi);

strcpy(p->lop, lop);

p->khoa = khoa;

p->next = NULL;

if (head == NULL) {

head = p;

}

else {

SinhVien\* q = head;

while (q->next != NULL) {

q = q->next;

}

q->next = p;

}

}

// Xuất danh sách sinh viên

void xuatDanhSach() {

SinhVien\* p = head;

while (p) {

xuatSV(p);

p = p->next;

}

cout << endl;

}

// Xóa sinh viên theo tên

void xoaSVTheoTen(const char\* ten) {

while (head && strcmp(head->hoTen, ten) == 0) {

SinhVien\* p = head;

head = head->next;

delete p;

}

SinhVien\* p = head;

while (p && p->next) {

if (strcmp(p->next->hoTen, ten) == 0) {

SinhVien\* nodeCanXoa = p->next;

p->next = p->next->next;

delete nodeCanXoa;

}

else {

p = p->next;

}

}

}

// Xóa sinh viên theo địa chỉ

void xoaSVTheoDiaChi(const char\* diachi) {

while (head && strcmp(head->diaChi, diachi) == 0) {

SinhVien\* p = head;

head = head->next;

delete p;

}

SinhVien\* p = head;

while (p && p->next) {

if (strcmp(p->next->diaChi, diachi) == 0) {

SinhVien\* nodeCanXoa = p->next;

p->next = p->next->next;

delete nodeCanXoa;

}

else {

p = p->next;

}

}

}

// Hàm so sánh theo tên

bool soSanhTheoTen(SinhVien\* a, SinhVien\* b) {

return strcmp(a->hoTen, b->hoTen) < 0;

}

// Hàm so sánh theo địa chỉ

bool soSanhTheoDiaChi(SinhVien\* a, SinhVien\* b) {

return strcmp(a->diaChi, b->diaChi) < 0;

}

// Hàm so sánh theo lớp

bool soSanhTheoLop(SinhVien\* a, SinhVien\* b) {

return strcmp(a->lop, b->lop) < 0;

}

// Hàm so sánh theo khóa

bool soSanhTheoKhoa(SinhVien\* a, SinhVien\* b) {

return a->khoa < b->khoa;

}

// Sắp xếp danh sách bằng selection sort

void sapXep(bool (ListSV::\* cmp)(SinhVien\*, SinhVien\*)) {

for (SinhVien\* p = head; p != NULL; p = p->next) {

SinhVien\* min = p;

for (SinhVien\* q = p->next; q != NULL; q = q->next) {

if ((this->\*cmp)(q, min)) {

min = q;

}

}

if (min != p) {

swap(\*p, \*min);

swap(p->next, min->next); // Giữ nguyên liên kết

}

}

}

// Giải phóng bộ nhớ danh sách

void xoaDanhSach() {

while (head) {

SinhVien\* p = head;

head = head->next;

delete p;

}

}

};

int main() {

ListSV danhSach;

// Thêm 10 sinh viên vào danh sách

danhSach.themSV("Nguyen Van A", "Nguyen Van Cu", "DCTC1", 2020);

danhSach.themSV("Nguyen Van Teo", "Le Loi", "DCTC2", 2021);

danhSach.themSV("Tran Thi B", "Nguyen Van Cu", "DCTC3", 2019);

danhSach.themSV("Le Thi C", "Tran Hung Dao", "DCTC6", 2022);

danhSach.themSV("Pham Van D", "Hai Ba Trung", "DCTC4", 2023);

danhSach.themSV("Bui Van E", "Hoang Dieu", "DCTC8", 2018);

danhSach.themSV("Nguyen Van F", "Vo Thi Sau", "DCTC9", 2017);

danhSach.themSV("Nguyen Van G", "Bach Dang", "DCTC5", 2016);

danhSach.themSV("Tran Van H", "Nguyen Trai", "DCTC0", 2015);

danhSach.themSV("Le Van I", "Le Lai", "DCTC7", 2014);

// Xuất danh sách sinh viên

cout << "Danh sach sinh vien ban dau:\n";

danhSach.xuatDanhSach();

cout << "Xoa sinh vien theo ten va dia chi\n";

danhSach.xoaSVTheoTen("Nguyen Van Teo");

danhSach.xoaSVTheoDiaChi("Nguyen Van Cu");

danhSach.xuatDanhSach();

cout << "Them sinh vien\n";

danhSach.themSV("Tran Thi Mo", "25 Hong Bang", "TT0901", 2009);

cout << "Danh sach sau khi them sinh vien moi:\n";

danhSach.xuatDanhSach();

// Sắp xếp danh sách theo tên

cout << "Sap xep theo ten:\n";

danhSach.sapXep(&ListSV::soSanhTheoTen);

danhSach.xuatDanhSach();

// Giải phóng bộ nhớ

danhSach.xoaDanhSach();

return 0;

}

TEST CODE

Kết quả:

A black screen with white text

AI-generated content may be incorrect.

A black screen with white text

AI-generated content may be incorrect.

A black screen with white text

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.