TRƯỜNG ĐẠI HỌC SÀI GÒN

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO MÔN HỌC

CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

BÁO CÁO BÀI TẬP:

THỰC HÀNH DANH SÁCH LIÊN KẾT

SINH VIÊN THỰC HIỆN: Nguyễn Đỗ Huy – 3121411085

LỚP: DCT124C7

GVHD: ĐỖ NHƯ TÀI

Thành phố Hồ Chí Minh , Tháng 3 Năm 2025

MỤC LỤC

Bài 1. Danh sách liên kết số nguyên

(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

(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

Bài làm

Ý tưởng

- Chương trình sử dụng danh sách liên kết đơn để quản lý các số nguyên. Các thao tác chính gồm:
  - Thêm phần tử vào đầu hoặc cuối danh sách.
  - Xóa phần tử theo giá trị bất kỳ.
  - Gộp hai danh sách và giữ nguyên thứ tự phần tử.
  - Xuất danh sách ra màn hình.
  - Giải phóng bộ nhớ khi kết thúc chương trình.
- Trong main(), chương trình:
  - Nhập danh sách số nguyên từ bàn phím.
  - Hiển thị danh sách.
  - Xóa một phần tử theo yêu cầu.
  - Nhập danh sách thứ hai và gộp vào danh sách đầu tiên.
  - Giải phóng bộ nhớ trước khi kết thúc.

Cài đặt chương trình

```
#include <iostream>
using namespace std;

// Cấu trúc Node cho danh sách số nguyên
```

```
struct Node {
    int info;
    Node* next;
};

struct ListInt {
    Node* head;
    void Init() { head = NULL; }
    bool IsEmpty() { return head == NULL; }
    void InsertFirst(int x);
    void InsertLast(int x);
    void Remove(int x);
    void Merge(ListInt& other);
    void Show();
    void Clear();
};

// Chèn phần tử vào đầu danh sách
void ListInt::InsertFirst(int x) {
    Node* p = new Node{x, head};
    head = p;
}

// Chèn phần tử vào cuối danh sách
void ListInt::InsertLast(int x) {
    Node* p = new Node{x, NULL};
    if (IsEmpty()) {
        head = p;
    } else {
        Node* temp = head;
        while (temp->next != NULL)
            temp = temp->next;
        temp->next = p;
    }
}

// Xóa phần tử có giá trị x
void ListInt::Remove(int x) {
    Node *p = head, *prev = NULL;
    while (p && p->info != x) {
        prev = p;
        p = p->next;
    }
    if (p) {
        if (prev) prev->next = p->next;
        else head = p->next;
        delete p;
```

```
    }
}

// Gộp danh sách `other` vào danh sách hiện tại, giữ nguyên thứ tự
void ListInt::Merge(ListInt& other) {
    if (other.head == NULL) return;
    Node* p = other.head;
    while (p) {
        InsertLast(p->info);
        p = p->next;
    }
}

// Hiển thị danh sách
void ListInt::Show() {
    if (IsEmpty()) {
        cout << "Danh sách rỗng.\n";
        return;
    }
    for (Node* p = head; p; p = p->next)
        cout << p->info << " -> ";
    cout << "NULL\n";
}

// Giải phóng bộ nhớ của danh sách
void ListInt::Clear() {
    while (head) {
        Node* p = head;
        head = head->next;
        delete p;
    }
}

int main() {
    ListInt list1, list2;
    list1.Init();
    list2.Init();

    cout << "\n===== NHẬP DANH SÁCH 1 =====\n";
    int num;
    for (int i = 0; i < 10; i++) {
        cout << "Nhập số thứ " << i + 1 << ": ";
        cin >> num;
        list1.InsertLast(num);
    }
    list1.Show();
```

```
    cout << "\nNhập số cần xóa: ";
    int k;
    cin >> k;
    list1.Remove(k);
    list1.Show();

    cout << "\n===== NHẬP DANH SÁCH 2 =====\n";
    for (int i = 0; i < 5; i++) {
        cout << "Nhập số thứ " << i + 1 << ": ";
        cin >> num;
        list2.InsertLast(num);
    }
    list2.Show();

    cout << "\nGhép danh sách 2 vào danh sách 1:\n";
    list1.Merge(list2);
    list1.Show();

    // Giải phóng bộ nhớ
    list1.Clear();
    list2.Clear();

    return 0;
}
```

Kiểm thử chương trình

1. Khởi tạo danh sách rỗng

Test case	Input	Expected Output
Kiểm tra danh sách rỗng	IsEmpty(list)	true

2. Thêm phần tử vào danh sách

Test case	Input	Expected Output	Test case	Input
Thêm 5 vào danh sách rỗng	InsertFirst(list, 5)	5 -> NULL	Thêm 5 vào danh sách rỗng	InsertFirst(list, 5)
Thêm 3 vào danh sách	InsertFirst(list, 3)	3 -> 5 -> NULL	Thêm 3 vào danh sách	InsertFirst(list, 3)

3. Xóa phần tử trong danh sách

Test case	Input	Expected Output
Xóa phần tử 8	Remove(list, 8)	1 -> 3 -> 5 -> NULL
Xóa phần tử 1	Remove(list, 1)	3 -> 5 -> NULL
Xóa phần tử không tồn tại (10)	Remove(list, 10)	Không thay đổi danh sách

4. Ghép danh sách

Test case	Input	Expected Output
Ghép danh sách {20 -> 21 -> NULL} vào {1 -> 3 -> 5 -> NULL}	AppendList(list1, list2)	1 -> 3 -> 5 -> 20 -> 21 -> NULL

5. Xóa toàn bộ danh sách

Test case	Input	Expected Output
Xóa danh sách	ClearList(list)	Danh sách rỗng

Bài 2. Danh sách liên kết sinh viên

(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.

(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.

(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

Bài làm

Ý tưởng

- Chương trình sử dụng danh sách liên kết đơn để quản lý danh sách sinh viên. Các chức năng chính gồm:
    - Thêm sinh viên vào danh sách.
    - Xóa sinh viên theo tên hoặc địa chỉ.
    - Hiển thị danh sách sinh viên.
    - Sắp xếp danh sách theo tiêu chí (tên, địa chỉ, lớp, khóa) bằng Selection Sort và con trỏ hàm.
    - Giải phóng bộ nhớ khi kết thúc chương trình.
- Trong main(), chương trình thực hiện:
    - Nhập danh sách sinh viên từ bàn phím.
    - Hiển thị danh sách.
    - Sắp xếp danh sách theo tên và hiển thị lại.
    - Xóa sinh viên có tên "Nguyen Van Teo".
    - Xóa sinh viên có địa chỉ "Nguyen Van Cu".
    - Thêm sinh viên "Tran Thi Mo" vào danh sách.
    - Hiển thị danh sách sau khi cập nhật.

- Giải phóng bộ nhớ trước khi kết thúc.

Cài đặt chương trình

```
#include <iostream>
#include <cstring>

using namespace std;

struct SinhVien {
    char hoTen[50];
    char diaChi[70];
    char lop[10];
    int khoa;
    SinhVien* next;
};

// Danh sách sinh viên
struct ListSV {
    SinhVien* head;
    void Init() { head = NULL; }
    void Insert(SinhVien sv);
    void RemoveByName(const char* name);
    void RemoveByAddress(const char* address);
    void Show();
    void Clear();
    void Sort(int (*compare)(SinhVien*, SinhVien*));
};

// Hàm so sánh theo tiêu chí
int CompareByName(SinhVien* sv1, SinhVien* sv2) {
    return strcmp(sv1->hoTen, sv2->hoTen);
}

int CompareByAddress(SinhVien* sv1, SinhVien* sv2) {
    return strcmp(sv1->diaChi, sv2->diaChi);
}

int CompareByClass(SinhVien* sv1, SinhVien* sv2) {
    return strcmp(sv1->lop, sv2->lop);
}

int CompareByCourse(SinhVien* sv1, SinhVien* sv2) {
    return sv1->khoa - sv2->khoa;
}
```

```
// Thêm sinh viên vào danh sách
void ListSV::Insert(SinhVien sv) {
    SinhVien* p = new SinhVien;
    strcpy(p->hoTen, sv.hoTen);
    strcpy(p->diaChi, sv.diaChi);
    strcpy(p->lop, sv.lop);
    p->khoa = sv.khoa;
    p->next = head;
    head = p;
}

// Xóa sinh viên theo tên
void ListSV::RemoveByName(const char* name) {
    SinhVien *p = head, *prev = NULL;
    while (p && strcmp(p->hoTen, name) != 0) {
        prev = p;
        p = p->next;
    }
    if (p) {
        if (prev) prev->next = p->next;
        else head = p->next;
        delete p;
    }
}

// Xóa sinh viên theo địa chỉ
void ListSV::RemoveByAddress(const char* address) {
    SinhVien *p = head, *prev = NULL;
    while (p && strcmp(p->diaChi, address) != 0) {
        prev = p;
        p = p->next;
    }
    if (p) {
        if (prev) prev->next = p->next;
        else head = p->next;
        delete p;
    }
}

// Hiển thị danh sách sinh viên
void ListSV::Show() {
    if (!head) {
        cout << "Danh sách rỗng.\n";
        return;
    }
    for (SinhVien* p = head; p; p = p->next) {
```

```
        cout << p->hoTen << " - " << p->diaChi << " - " << p->lop << " - " << p-
>khoa << "\n";
    }
}

// Xóa toàn bộ danh sách
void ListSV::Clear() {
    while (head) {
        SinhVien* p = head;
        head = head->next;
        delete p;
    }
}

// Sắp xếp danh sách bằng Selection Sort với con trỏ hàm
void ListSV::Sort(int (*compare)(SinhVien*, SinhVien*)) {
    for (SinhVien* p = head; p != NULL; p = p->next) {
        SinhVien* minNode = p;
        for (SinhVien* q = p->next; q != NULL; q = q->next) {
            if (compare(q, minNode) < 0) {
                minNode = q;
            }
        }
        // Hoán đổi nội dung thay vì con trỏ
        if (minNode != p) {
            SinhVien temp = *p;
            *p = *minNode;
            *minNode = temp;
            // Giữ nguyên con trỏ next
            swap(p->next, minNode->next);
        }
    }
}

int main() {
    ListSV svList;
    svList.Init();

    cout << "\n===== NHẬP DANH SÁCH SINH VIÊN =====\n";
    cin.ignore(); // Để loại bỏ ký tự '\n' còn lại trong bộ đệm

    for (int i = 0; i < 10; i++) {
        SinhVien sv;
        cout << "Nhập họ tên: "; cin.getline(sv.hoTen, 50);
        cout << "Nhập địa chỉ: "; cin.getline(sv.diaChi, 70);
        cout << "Nhập lớp: "; cin.getline(sv.lop, 10);
        cout << "Nhập khóa: "; cin >> sv.khoa;
```

```
        cin.ignore(); // Loại bỏ ký tự '\n' sau khi nhập số
        svList.Insert(sv);
    }
    svList.Show();

    cout << "\nSắp xếp danh sách theo tên:\n";
    svList.Sort(CompareByName);
    svList.Show();

    cout << "\nXóa sinh viên 'Nguyen Van Teo'\n";
    svList.RemoveByName("Nguyen Van Teo");
    svList.Show();

    cout << "\nXóa sinh viên có địa chỉ 'Nguyen Van Cu'\n";
    svList.RemoveByAddress("Nguyen Van Cu");
    svList.Show();

    SinhVien newSv = {"Tran Thi Mo", "25 Hong Bang", "TT0901", 2009, NULL};
    svList.Insert(newSv);

    cout << "\nDanh sách sau khi thêm sinh viên Tran Thi Mo:\n";
    svList.Show();

    // Giải phóng bộ nhớ
    svList.Clear();

    return 0;
}
```

Kiểm thử chương trình

1. Khởi tạo danh sách rỗng

Test case	**Input**	**Expected Output**
Kiểm tra danh sách rỗng	IsEmpty(list)	true

2. Thêm sinh viên vào danh sách

Test case	Input	Expected Output
Thêm sinh viên "Nguyen Van A"	Insert(svList, {"Nguyen Van A", "123 Nguyen Trai", "CNTT1", 2022})	Nguyen Van A - 123 Nguyen Trai - CNTT1 - 2022
Thêm sinh viên "Tran Van B"	Insert(svList, {"Tran Van B", "456 Le Loi", "CNTT2", 2023})	Tran Van B - 456 Le Loi - CNTT2 - 2023 Nguyen Van A - 123 Nguyen Trai - CNTT1 - 2022

3. Xóa sinh viên

Test case	Input	Expected Output
Xóa sinh viên "Nguyen Van A"	RemoveByName(svList, "Nguyen Van A")	Tran Van B - 456 Le Loi - CNTT2 - 2023
Xóa sinh viên không tồn tại "Le Thi C"	RemoveByName(svList, "Le Thi C")	Không thay đổi danh sách
Xóa sinh viên có địa chỉ "456 Le Loi"	RemoveByAddress(svList, "456 Le Loi")	Danh sách rỗng

4. Sắp xếp danh sách sinh viên

Test case	Input	Expected Output
Sắp xếp theo tên	Sort(svList, CompareByName)	Danh sách theo thứ tự tăng dần của họ tên
Sắp xếp theo địa chỉ	Sort(svList, CompareByAddress)	Danh sách theo thứ tự tăng dần của địa chỉ
Sắp xếp theo lớp	Sort(svList, CompareByClass)	Danh sách theo thứ tự tăng dần của lớp
Sắp xếp theo khóa	Sort(svList, CompareByCourse)	Danh sách theo thứ tự tăng dần của khóa

5. Xóa toàn bộ danh sách

Test case	Input	Expected Output
Xóa danh sách	ClearList(list)	Danh sách rỗng