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:

CÀI CÁC TÁC VỤ DANH SÁCH LIÊN KẾT ĐƠN

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

1. Giới thiệu
   - Danh sách liên kết (linked list) là một danh sách mà các phần tử nối kết với nhau dựa vào vùng liên kết của chúng: vùng liên kết của phần tử ai chứa tham chiếu đến phần tử a_{i+1}.
   - Báo cáo này trình bày cách cài đặt danh sách liên kết đơn bằng ngôn ngữ lập trình C++ và các thao tác cơ bản như thêm, xóa, tìm kiếm, duyệt và sắp xếp danh sách.

2. Cấu trúc dữ liệu
   - Danh sách liên kết đơn bao gồm các phần tử (nút - Node), mỗi nút chứa hai thành phần:
     - Info: lưu trữ giá trị của nút.
     - Next: con trỏ trỏ đến nút tiếp theo trong danh sách.
   - Cấu trúc của một nút được khai báo như sau:

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

3. Các tác vụ của danh sách liên kết đơn
   - Khởi tạo danh sách rỗng (Init).
   - Kiểm tra danh sách có rỗng không (IsEmpty).
   - Tạo một nút mới (CreateNode).
   - Thêm một phần tử vào đầu danh sách (InsertFirst).
   - Chèn một phần tử sau một nút (InsertAfter).
   - Chèn phần tử vào danh sách có thứ tự (InsertOrder).
   - Xóa phần tử đầu tiên (DeleteFirst).
   - Xóa phần tử sau một nút (DeleteAfter).
   - Xóa phần tử theo giá trị (Remove).
   - Tìm kiếm phần tử theo giá trị (Find).
   - Hiển thị danh sách (ShowList).
   - Sắp xếp danh sách theo thứ tự tăng dần (SelectionSort).
   - Xóa toàn bộ danh sách (ClearList).

4. Cài đặt chương trình
   - Khởi tạo danh sách rỗng

```
void Init(Node*& pHead) {
    pHead = NULL;
}
```

   - Kiểm tra danh sách rỗng

```
bool IsEmpty(Node* pHead) {
    return (pHead == NULL);
}
```

   - Tạo một Node mới

```
Node* CreateNode(int x) {
    Node* p = new Node;
    p->info = x;
    p->next = NULL;
    return p;
}
```

   - Duyệt danh sách

```
void ShowList(Node* pHead) {
    if (IsEmpty(pHead)) {
        cout << "Danh sách rỗng\n";
        return;
    }
    Node* p = pHead;
    while (p != NULL) {
        cout << p->info << " -> ";
        p = p->next;
    }
    cout << "NULL\n";
}
```

   - Thêm phần tử vào đầu danh sách

```
void InsertFirst(Node*& pHead, int x) {
    Node* p = CreateNode(x);
    p->next = pHead;
    pHead = p;
}
```

   - Thêm phần tử sau một Node p đã biết

```
void InsertAfter(Node* p, int x) {
    if (p != NULL) {
        Node* q = CreateNode(x);
        q->next = p->next;
        p->next = q;
    }
```

```
}
```

- Thêm phần tử có thứ tự

```
void InsertOrder(Node*& pHead, int x) {
    Node* tp = NULL;
    Node* p = pHead;
    while (p != NULL && p->info < x) {
        tp = p;
        p = p->next;
    }
    if (tp == NULL)
        InsertFirst(pHead, x);
    else
        InsertAfter(tp, x);
}
```

- Tìm kiếm phần tử trong danh sách không có thứ tự

```
Node* Find(Node* pHead, int x) {
    Node* p = pHead;
    while (p != NULL) {
        if (p->info == x)
            return p;
        p = p->next;
    }
    return NULL;
}
```

- Tìm kiếm phần tử trong danh sách có thứ tự

```
Node* FindOrder(Node* pHead, int x) {
    Node* p = pHead;
    while (p != NULL) {
        if (p->info == x)
            return p;
        else if (p->info > x) // Nếu danh sách có thứ tự, không cần tìm tiếp
            return NULL;
        p = p->next;
    }
    return NULL;
}
```

- Xóa phần tử đầu danh sách

```
void DeleteFirst(Node*& pHead) {
    if (IsEmpty(pHead)) {
        cout << "Danh sách rỗng!\n";
        return;
    }
    Node* p = pHead;
    pHead = pHead->next;
```

```
    delete p;
}
```

- Xóa phần tử sau nút p

```
void DeleteAfter(Node* p) {
    if (p == NULL || p->next == NULL) {
        cout << "Không thể xóa nút này!\n";
        return;
    }
    Node* q = p->next;
    p->next = q->next;
    delete q;
}
```

- Xóa phần tử có giá trị x (danh sách không có thứ tự)

```
void Remove(Node*& pHead, int x) {
    Node* tp = NULL;
    Node* p = pHead;
    while (p != NULL && p->info != x) {
        tp = p;
        p = p->next;
    }
    if (p == NULL) return;
    if (p == pHead) {
        pHead = p->next;
    } else {
        tp->next = p->next;
    }
    delete p;
}
```

- Xóa phần tử trong danh sách có thứ tự

```
void RemoveOrder(Node*& pHead, int x) {
    Node* tp = NULL;
    Node* p = pHead;
    while (p != NULL && p->info < x) {
        tp = p;
        p = p->next;
    }
    if (p == NULL || p->info != x) return;
    if (p == pHead) {
        pHead = p->next;
    } else {
        tp->next = p->next;
    }
    delete p;
}
```

- Sắp xếp danh sách bằng Selection Sort

```
void SelectionSort(Node*& pHead) {
    for (Node* p = pHead; p != NULL && p->next != NULL; p = p->next) {
        Node* pMin = p;
        for (Node* q = p->next; q != NULL; q = q->next) {
            if (q->info < pMin->info) {
                pMin = q;
            }
        }
        swap(p->info, pMin->info);
    }
}
```

- Xóa toàn bộ danh sách

```
void ClearList(Node*& pHead) {
    Node* p;
    while (pHead != NULL) {
        p = pHead;
        pHead = pHead->next;
        delete p;
    }
}
```

5. Kiểm thử chương trình
   - Khởi tạo danh sách rỗng

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

   - Thêm phần tử vào danh sách

Test case	Input	Expected Output
Thêm 5 vào danh sách rỗng	InsertOrder(list, 5)	5 -> NULL
Thêm 3 vào danh sách	InsertOrder(list, 3)	3 -> 5 -> NULL
Thêm 8 vào danh sách	InsertOrder(list, 8)	3 -> 5 -> 8 -> NULL
Thêm 1 vào danh sách	InsertOrder(list, 1)	1 -> 3 -> 5 -> 8 -> NULL
Thêm 7 vào danh sách	InsertOrder(list, 7)	1 -> 3 -> 5 -> 7 -> 8 -> NULL

   - Duyệt danh sách

Test case	Input	Expected Output
Hiển thị danh sách	ShowList(list)	1 -> 3 -> 5 -> 7 -> 8 -> NULL

   - Tìm kiếm phần tử

Test case	Input	Expected Output
Tìm phần tử 5	FindOrder(list, 5)	Tìm thấy
Tìm phần tử 10	FindOrder(list, 10)	Không tìm thấy

   - Xóa phần tử trong danh sách

Test case	Input	Expected Output
Xóa phần tử đầu tiên	DeleteFirst(list)	3 -> 5 -> 7 -> 8 -> NULL
Xóa phần tử 5	RemoveOrder(list, 5)	3 -> 7 -> 8 -> NULL
Xóa phần tử không tồn tại (10)	RemoveOrder(list, 10)	Không thay đổi danh sách

   - Sắp xếp danh sách

Test case	Input	Expected Output
Sắp xếp danh sách tăng dần	SelectionSort(list)	3 -> 7 -> 8 -> NULL

   - Xóa toàn bộ danh sách

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

   - Xóa phần tử sau một nút

Test case	Input	Expected Output
Thêm 2 vào danh sách	InsertOrder(list, 2)	2 -> 3 -> 7 -> 8 -> NULL
Thêm 6 vào danh sách	InsertOrder(list, 6)	2 -> 3 -> 6 -> 7 -> 8 -> NULL
Xóa phần tử sau nút có giá trị 3	DeleteAfter(FindOrder(list, 3))	2 -> 3 -> 7 -> 8 -> NULL