TRƯỜNG ĐẠI HỌC SÀI GÒN

**KHOA CÔNG NGHỆ THÔNG TIN**

Báo cáo học phần Cấu trúc dữ liệu và giải thuật

**BTTH DANH SÁCH LIÊN KẾT ĐƠN**

|  |  |
| --- | --- |
| **Sinh viên thực hiện:** | Nguyễn Hoài Quân MSSV: 3124411249 |

Thành phố Hồ Chí Minh, tháng 03 năm 2025

**1. Mục tiêu**

* Hiểu và cài đặt danh sách liên kết đơn.
* Thực hiện các thao tác cơ bản: thêm, xóa, tìm kiếm, sắp xếp.
* Sử dụng bộ nhớ động hiệu quả.

**2. Nội dung thực hành**

**2.1. Cấu trúc dữ liệu**

Danh sách liên kết đơn gồm các nút (Node), mỗi nút chứa:

* info: Giá trị phần tử.
* next: Con trỏ trỏ đến nút kế tiếp.

**2.2. Các thao tác**

* **Khởi tạo:** Init(Node\*& pHead) đặt pHead = NULL.
* **Kiểm tra rỗng:** IsEmpty(Node\* pHead).
* **Tạo nút:** CreateNode(int x).
* **Thêm phần tử:** InsertFirst, InsertAfter, InsertOrder.
* **Xóa phần tử:** Remove, RemoveOrder, DeleteAfter.
* **Tìm kiếm:** Find(Node\* pHead, int x).
* **Duyệt danh sách:** ShowList(Node\* pHead).
* **Sắp xếp:** SelectionSort(Node\*& pHead).
* **Giải phóng bộ nhớ:** ClearList(Node\*& pHead).

**2.3. Chương trình minh họa**

**Mã nguồn:**

#include <iostream>

using namespace std**;**

*// Định nghĩa cấu trúc của một node trong danh sách liên kết*

struct Node {

*int* *info***;**

    Node**\*** *next***;**

}**;**

*// Kiểm tra danh sách có rỗng không*

*bool* IsEmpty(Node*\** **pHead**) {

**return** (**pHead** **==** NULL)**;**

}

*// Khởi tạo danh sách liên kết rỗng*

*void* Init(Node*\*&* **pHead**) {

**pHead** **=** NULL**;**

}

*// Tạo một node mới với giá trị x*

Node*\** CreateNode(*int* **x**) {

    Node**\*** p **=** **new** Node**;**

    p**->***info* **=** **x;**

    p**->***next* **=** NULL**;**

**return** p**;**

}

*// Hiển thị danh sách liên kết*

*void* ShowList(Node*\** **pHead**) {

**if** (**pHead** **==** NULL) {

        cout **<<** "\nDanh sách rỗng!" **<<** endl**;**

    } **else** {

**for** (Node**\*** p **=** **pHead;** p **!=** NULL**;** p **=** p**->***next*) {

            cout **<<** p**->***info* **<<** "\t"**;**

        }

        cout **<<** endl**;**

    }

}

*// Tìm kiếm một node có giá trị x trong danh sách*

Node*\** Find(Node*\** **pHead,** *int* **x**) {

    Node**\*** p **=** **pHead;**

**while** (p **!=** NULL) {

**if** (p**->***info* **==** **x**)

**return** p**;** *// Trả về node nếu tìm thấy*

        p **=** p**->***next***;**

    }

**return** NULL**;** *// Trả về NULL nếu không tìm thấy*

}

*// Thêm một node mới vào đầu danh sách*

*void* InsertFirst(Node*\*&* **pHead,** *int* **x**) {

    Node**\*** p **=** CreateNode(**x**)**;**

    p**->***next* **=** **pHead;**

**pHead** **=** p**;**

}

*// Thêm một node mới vào sau node p*

*void* InsertAfter(Node*\** **p,** *int* **x**) {

**if** (**p** **!=** NULL) {

        Node**\*** q **=** CreateNode(**x**)**;**

        q**->***next* **=** **p->***next***;** *// Liên kết node mới với phần sau của p*

**p->***next* **=** q**;** *// Cập nhật liên kết của p*

    }

}

*// Thêm một node mới vào danh sách đã có thứ tự tăng dần*

*void* InsertOrder(Node*\*&* **pHead,** *int* **x**) {

    Node**\*** p **=** **pHead;**

    Node**\*** prev **=** NULL**;**

*// Duyệt danh sách để tìm vị trí chèn phù hợp*

**while** (p **!=** NULL **&&** p**->***info* **<** **x**) {

        prev **=** p**;**

        p **=** p**->***next***;**

    }

*// Nếu chèn vào đầu danh sách*

**if** (prev **==** NULL)

        InsertFirst(**pHead,** **x**)**;**

**else**

        InsertAfter(prev**,** **x**)**;**

}

*// Xóa một node có giá trị x khỏi danh sách không có thứ tự*

Node*\** Remove(Node*\*&* **pHead,** *int* **x**) {

    Node**\*** p **=** **pHead;**

    Node**\*** prev **=** NULL**;**

**while** (p **!=** NULL **&&** p**->***info* **!=** **x**) {

        prev **=** p**;**

        p **=** p**->***next***;**

    }

**if** (p **!=** NULL) { *// Nếu tìm thấy node cần xóa*

**if** (p **==** **pHead**)

**pHead** **=** p**->***next***;**

**else**

            prev**->***next* **=** p**->***next***;**

**delete** p**;**

    }

**return** **pHead;**

}

*// Xóa một node có giá trị x trong danh sách đã có thứ tự*

Node*\** RemoveOrder(Node*\*&* **pHead,** *int* **x**) {

    Node**\*** p **=** **pHead;**

    Node**\*** prev **=** NULL**;**

**while** (p **!=** NULL **&&** p**->***info* **<** **x**) {

        prev **=** p**;**

        p **=** p**->***next***;**

    }

**if** (p **!=** NULL **&&** p**->***info* **==** **x**) { *// Nếu tìm thấy node cần xóa*

**if** (p **==** **pHead**)

**pHead** **=** p**->***next***;**

**else**

            prev**->***next* **=** p**->***next***;**

**delete** p**;**

    }

**return** **pHead;**

}

*// Xóa node ngay sau node p*

*void* DeleteAfter(Node*\** **p**) {

**if** (**p** **==** NULL **||** **p->***next* **==** NULL) {

        cout **<<** "Không thể xóa nút này!" **<<** endl**;**

    } **else** {

        Node**\*** temp **=** **p->***next***;**

**p->***next* **=** temp**->***next***;**

**delete** temp**;**

    }

}

*// Xóa toàn bộ danh sách*

*void* ClearList(Node*\*&* **pHead**) {

    Node**\*** temp**;**

**while** (**pHead** **!=** NULL) {

        temp **=** **pHead;**

**pHead** **=** **pHead->***next***;**

**delete** temp**;**

    }

}

*// Sắp xếp danh sách theo thứ tự tăng dần bằng Selection Sort*

*void* SelectionSort(Node*\*&* **pHead**) {

**if** (**pHead** **==** NULL) **return;**

**for** (Node**\*** p **=** **pHead;** p**->***next* **!=** NULL**;** p **=** p**->***next*) {

        Node**\*** minNode **=** p**;**

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

**if** (q**->***info* **<** minNode**->***info*)

                minNode **=** q**;**

        }

**if** (minNode **!=** p) {

*int* temp **=** p**->***info***;**

            p**->***info* **=** minNode**->***info***;**

            minNode**->***info* **=** temp**;**

        }

    }

}

*int* main() {

    Node**\*** pHead**;**

    Init(pHead)**;**

*// Tạo danh sách ban đầu với các giá trị: 5, 15, 25*

    InsertFirst(pHead**,** 25)**;**

    InsertFirst(pHead**,** 15)**;**

    InsertFirst(pHead**,** 5)**;**

    cout **<<** "Danh sách ban đầu:" **<<** endl**;**

    ShowList(pHead)**;**

*// Ví dụ: Chèn một node có giá trị 18 sau node có giá trị 15*

    Node**\*** pFound **=** Find(pHead**,** 15)**;**

**if** (pFound **!=** NULL) {

        InsertAfter(pFound**,** 18)**;**

    }

    cout **<<** "\nDanh sách sau khi chèn 18 sau 15:" **<<** endl**;**

    ShowList(pHead)**;**

*// Ví dụ: Xóa node có giá trị 15*

    Remove(pHead**,** 15)**;**

    cout **<<** "\nDanh sách sau khi xóa 15:" **<<** endl**;**

    ShowList(pHead)**;**

*// Ví dụ: Thêm một node vào danh sách đã được sắp xếp (chèn có thứ tự)*

*// Chèn node có giá trị 12 vào vị trí thích hợp*

    InsertOrder(pHead**,** 12)**;**

    cout **<<** "\nDanh sách sau khi chèn 12 vào danh sách có thứ tự:" **<<** endl**;**

    ShowList(pHead)**;**

*// Sắp xếp danh sách theo thứ tự tăng dần*

    SelectionSort(pHead)**;**

    cout **<<** "\nDanh sách sau khi sắp xếp:" **<<** endl**;**

    ShowList(pHead)**;**

*// Xóa toàn bộ danh sách*

    ClearList(pHead)**;**

    cout **<<** "\nDanh sách sau khi xóa toàn bộ:" **<<** endl**;**

    ShowList(pHead)**;**

**return** 0**;**

}

1. Khởi tạo danh sách với 5, 15, 25.
2. Chèn 18 sau 15.
3. Xóa 15.
4. Chèn 12 theo thứ tự.
5. Sắp xếp danh sách.
6. Giải phóng bộ nhớ.

**3. Kết quả**

* Danh sách ban đầu: 5 15 25
* Sau khi chèn 18 sau 15: 5 15 18 25
* Sau khi xóa 15: 5 18 25
* Sau khi chèn 12 theo thứ tự: 5 12 18 25
* Sau khi sắp xếp: 5 12 18 25
* Sau khi xóa danh sách: Danh sách rỗng