VÕ TIẾN
Thảo luận kiến thức CNTT trường BK về KHMT(CScience), KTMT(CEngineering)
https://www.facebook.com/groups/khmt.ktmt.cse.bku

Kỹ Thuật Lập Trình (Cơ bản và nâng cao C++)

KTLT2 - HK242

TASK 10 Linked List

Thảo luận kiến thức CNTT trường BK
về KHMT(CScience), KTMT(CEngineering)
https://www.facebook.com/groups/khmt.ktmt.cse.bku

TP. HỒ CHÍ MINH, THÁNG 3/2025

Mục lục

1 Danh sách liên kết đơn (Singly Linked List)

Khái niệm Danh sách liên kết đơn (singly linked list) là một cấu trúc dữ liệu động gồm các **nút (node)** được liên kết theo một chiều, trong đó mỗi nút chứa dữ liệu và một con trỏ (hoặc tham chiếu) đến nút tiếp theo. Đây là một trong những cấu trúc dữ liệu cơ bản nhất, thuộc nhóm **danh sách liên kết (linked list)**, khác với mảng ở chỗ không yêu cầu bộ nhớ liền kề và có kích thước linh hoạt.

1. **Lý thuyết cơ bản**:
   - Mỗi nút trong danh sách liên kết đơn có hai thành phần:
     - **Dữ liệu (data)**: Giá trị được lưu trữ (có thể là bất kỳ kiểu nào).
     - **Con trỏ next**: Trỏ đến nút tiếp theo hoặc nullptr nếu là nút cuối cùng.
   - Danh sách có một **nút đầu (head)** để truy cập và thường có một con trỏ đến **nút cuối (tail)** nếu cần tối ưu thêm/xóa nhanh.
   - Kích thước danh sách không cố định, có thể thay đổi trong runtime bằng cách thêm hoặc xóa nút.
2. **Đặc điểm**:
   - Chỉ có thể duyệt theo một chiều (từ đầu đến cuối).
   - Không hỗ trợ truy cập ngẫu nhiên (random access) như mảng.
   - Dùng bộ nhớ động (heap) để cấp phát các nút.

```cpp
struct Node {
    int data;
    Node* next;
    Node(int val) : data(val), next(nullptr) {}
};

int main() {
    Node* head = new Node(1);
    head->next = new Node(2);
    head->next->next = new Node(3);

    Node* current = head;
    while (current != nullptr) {
        cout << current->data << " ";  // Output: 1 2 3
        current = current->next;
    }

    // Giải phóng bộ nhớ
    while (head != nullptr) {
        Node* temp = head;
        head = head->next;
        delete temp;
    }
    return 0;
}
```

1.1 Cách triển khai (Lý thuyết và cơ chế)

Cấu trúc cơ bản

- **Lý thuyết**:
  - Danh sách liên kết đơn thường được triển khai bằng một lớp/struct Node và một lớp quản lý danh sách (LinkedList).
  - Lớp Node chứa dữ liệu và con trỏ next.
- **Cơ chế bên dưới**:
  - Mỗi nút được cấp phát động bằng new, liên kết qua con trỏ next.
  - Khi xóa, cần giải phóng bộ nhớ bằng delete để tránh rò rỉ.

Các thao tác cơ bản

1. **Thêm nút (Insertion)**:
   - Thêm vào đầu (O(1)).
     (a) Tạo nút mới với dữ liệu.
     (b) Gán next của nút mới trỏ đến head hiện tại.
     (c) Cập nhật head thành nút mới.
   - Thêm vào cuối (O(n) nếu không có con trỏ tail).
     (a) Tạo nút mới.
     (b) Nếu danh sách rỗng (head = nullptr), gán head là nút mới.
     (c) Nếu không, duyệt đến nút cuối và gán next của nút cuối trỏ đến nút mới.
   - Thêm vào giữa (O(n) để tìm vị trí).
     (a) Tạo nút mới.
     (b) Duyệt đến nút trước vị trí cần chèn.
     (c) Gán next của nút mới trỏ đến nút sau vị trí.
     (d) Gán next của nút trước trỏ đến nút mới.
2. **Xóa nút (Deletion)**:
   - Xóa đầu (O(1)).
     (a) Kiểm tra danh sách rỗng (head = nullptr).
     (b) Lưu nút đầu vào biến tạm.
     (c) Cập nhật head trỏ đến nút thứ hai.
     (d) Giải phóng nút tạm.
   - Xóa cuối hoặc giữa (O(n) để tìm).
     (a) Nếu xóa đầu, dùng cách trên.
     (b) Nếu không, duyệt đến nút trước nút cần xóa.
     (c) Điều chỉnh con trỏ next và giải phóng nút.
3. **Duyệt danh sách (Traversal)**: Duyệt từ đầu đến cuối (O(n)).
   (a) Bắt đầu từ head.
   (b) Dùng vòng lặp để di chuyển qua từng nút (current = current->next) cho đến khi gặp nullptr.
4. **Tìm kiếm (Search)**: Tìm phần tử theo giá trị (O(n)).

(a) Bắt đầu từ head.

(b) So sánh giá trị của từng nút với giá trị cần tìm.

(c) Trả về true nếu tìm thấy, false nếu không.

```
class Node {
public:
    int data;
    Node* next;
    Node(int val) : data(val), next(nullptr) {}
};

class LinkedList {
private:
    Node* head;

public:
    LinkedList() : head(nullptr) {}
    ~LinkedList() {
        while (head != nullptr) {
            Node* temp = head;
            head = head->next;
            delete temp;
        }
    }

    // Thêm vào đầu
    void insertAtHead(int val) {
        Node* newNode = new Node(val);
        newNode->next = head;
        head = newNode;
    }

    // Thêm vào cuối
    void insertAtTail(int val) {
        Node* newNode = new Node(val);
        if (head == nullptr) {
            head = newNode;
            return;
        }
        Node* current = head;
        while (current->next != nullptr) {
            current = current->next;
        }
        current->next = newNode;
    }

    // Xóa đầu
    void deleteAtHead() {
        if (head == nullptr) return;
        Node* temp = head;
        head = head->next;
        delete temp;
    }

    // Duyệt danh sách
```

```
    void display() const {
        Node* current = head;
        while (current != nullptr) {
            cout << current->data << " ";
            current = current->next;
        }
        cout << endl;
    }

    // Tìm kiếm
    bool search(int val) const {
        Node* current = head;
        while (current != nullptr) {
            if (current->data == val) return true;
            current = current->next;
        }
        return false;
    }
};

int main() {
    LinkedList list;
    list.insertAtHead(3);
    list.insertAtHead(2);
    list.insertAtTail(4);
    list.display();      // Output: 2 3 4
    cout << list.search(3) << endl;  // Output: 1 (true)
    list.deleteAtHead();
    list.display();      // Output: 3 4
    return 0;
}
```

1.2 Danh sách liên kết đơn trong kế thừa và đa hình

- Danh sách liên kết đơn có thể được dùng để lưu trữ các đối tượng đa hình (polymorphic objects) bằng cách lưu con trỏ lớp cơ sở (Base*) thay vì kiểu cụ thể.
- Khi xóa, cần đảm bảo hàm hủy ảo trong lớp cơ sở để hủy đúng lớp dẫn xuất.

```
class Unit {
public:
    virtual void info() = 0;
    virtual ~Unit() {}
};

class Soldier : public Unit {
public:
    void info() override { cout << "Soldier" << endl; }
};

class Tank : public Unit {
public:
    void info() override { cout << "Tank" << endl; }
};

```

```cpp
struct Node {
    Unit* data;
    Node* next;
    Node(Unit* val) : data(val), next(nullptr) {}
};

class UnitList {
private:
    Node* head;

public:
    UnitList() : head(nullptr) {}
    ~UnitList() {
        while (head != nullptr) {
            Node* temp = head;
            head = head->next;
            delete temp->data;  // Hủy đối tượng Unit
            delete temp;        // Hủy nút
        }
    }

    void insert(Unit* unit) {
        Node* newNode = new Node(unit);
        newNode->next = head;
        head = newNode;
    }

    void display() const {
        Node* current = head;
        while (current != nullptr) {
            current->data->info();
            current = current->next;
        }
    }
};

int main() {
    UnitList list;
    list.insert(new Soldier());
    list.insert(new Tank());
    list.display();  // Output: Tank \n Soldier
    return 0;
}
```