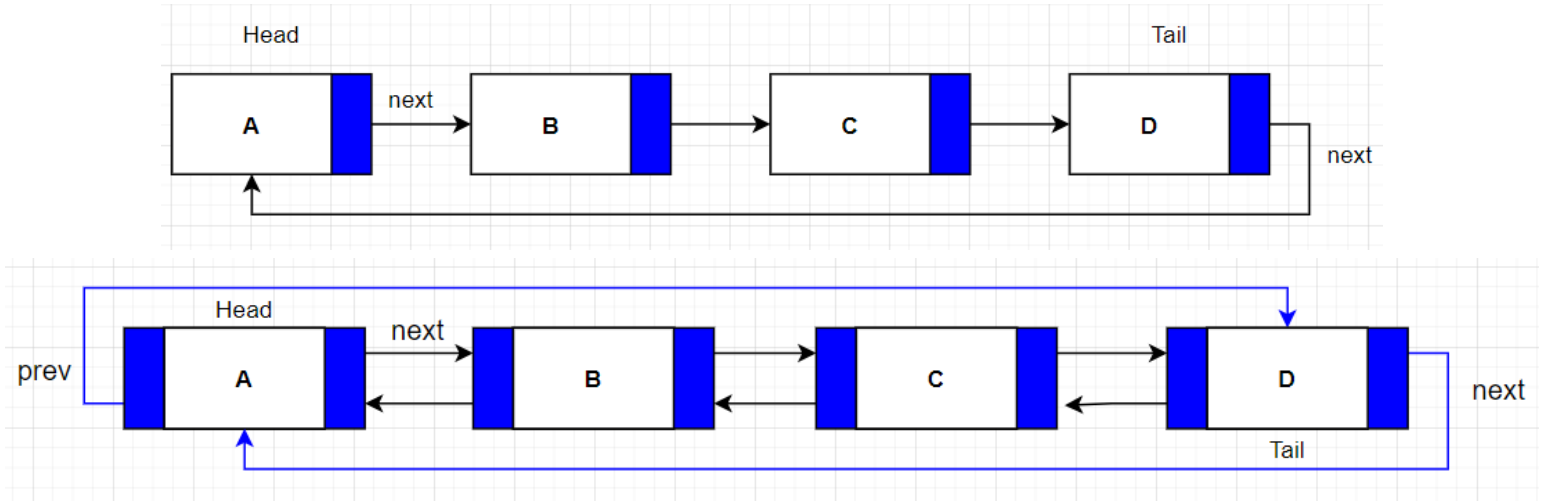
Bài 3.9: Danh sách liên kết vòng

- ✓ Khái niệm và đặc điểm
- ✓ Thêm node vào DSLK vòng
- ✓ Xóa node khỏi DSLK vòng
- ✓ Duyệt DSLK vòng
- ✓ Ví dụ minh họa
- ✓ Bài tập thực hành

Khái niệm

- Danh sách liên kết vòng là biến thể của danh sách liên kết.
- Nếu là danh sách liên kết đơn vòng, next của node tail sẽ trỏ tới node head.
- Nếu là danh sách liên kết đôi vòng, next của node tail trỏ tới node head. Node prev của head sẽ trỏ tới node tail.



Đặc điểm

- Bất kì node nào trong danh sách cũng có thể được sử dụng làm node bắt đầu.
- Hữu ích khi triển khai hàng đợi.
- Áp dụng trong các ứng dụng cần chạy lặp đi lặp lại như xử lý đa nhiệm trong hệ điều hành.

Thêm node vào danh sách

- Tạo node p với data cho trước.
- Nếu danh sách rỗng:
  - Gán head = tail = p;
  - Gán p->next = head;
  - Gán p->prev = head;
- Nếu danh sách không rỗng:
  - Gán p->next = head;
  - Gán head->prev = p;
  - Gán p->prev = tail;
  - Gán tail->next = p;
  - Cập nhật lại head: head = p;

Ví dụ

```
void add(T data) {
    Node<T>* p = new Node<T>(data); // tạo node mới p
    if (isEmpty()) { // kiểm tra xem danh sách rỗng không
        head = tail = p; // gán giá trị cho head, tail
        p->next = head;
        p->prev = head;
    }
    else { // nếu danh sách không rỗng
        p->next = head; // cập nhật node next của p
        head->prev = p; // cập nhật prev của head
        p->prev = tail; // cập nhật prev của p
        tail->next = p; // cập nhật next của tail
        head = p; // cập nhật lại node head
    }
}
```

Xóa node khỏi danh sách

- Nếu danh sách rỗng, kết thúc.
- Nếu danh sách chỉ có 1 node:
  - Lưu lại head vào biến r;
  - Gán head = tail = null;
  - Xóa node r;
- Nếu danh sách có nhiều node:
  - Lưu lại head vào biến r;
  - Cập nhật head: head = head->next;
  - Cập nhật prev của head: head->prev = tail;
  - Cập nhật next của tail: tail->next = head;
  - Xóa node r.

Ví dụ

```
bool remove() {
    if (isEmpty()) {
        return false;
    }
    if (head == tail) {
        Node<T>* r = head;
        head = nullptr;
        tail = nullptr;
        delete r;
        return true;
    }
    else {
        Node<T>* r = head;
        head = head->next;
        head->prev = tail;
        tail->next = head;
        delete r;
        return true;
    }
}
```

Duyệt danh sách

- Nếu danh sách rỗng, kết thúc.
- Nếu danh sách không rỗng:
- B1. Khởi tạo p = head;
- B2. Lặp vô hạn:
- B3. Hiển thị p->data;
- B4. Cập nhật p: p = p->next;
- B5. Nếu p == head, break;

Ví dụ

```
void showList() { // hàm hiển thị danh sách liên kết
    if (isEmpty()) {
        cout << "Danh sach rong.\n";
    }
    else {
        Node<T>* p = head; // khởi đầu từ node head
        while (true)
        {
            cout << p->data << endl; // in dữ liệu của node p
            p = p->next; // cập nhật p
            if (p == head) { // nếu p là node head, kết thúc 1 vòng
                break;
            }
        }
        cout << endl;
    }
}
```

Kết quả minh họa

```
LinkedList<BankAccount> accounts;
accounts.add(BankAccount{ 100, "TK001", 50000 });
accounts.add(BankAccount{ 200, "TK002", 10000 });
accounts.add(BankAccount{ 300, "TK003", 30000 });
accounts.add(BankAccount{ 400, "TK004", 20000 });
accounts.add(BankAccount{ 500, "TK005", 40000 });
accounts.add(BankAccount{ 700, "TK007", 80000 });
accounts.add(BankAccount{ 600, "TK006", 70000 });
cout << "Danh sach ban dau: \n";
accounts.showList();
auto result = accounts.remove();
if (result) {
    cout << "Xoa thanh cong!\n";
}
else {
    cout << "Xoa that bai!\n";
}
cout << "Danh sach sau khi xoa: \n";
accounts.showList();
```

```
Microsoft Visual Studio Debug Console
Danh sach ban dau:
[600, TK006, 70000]
[700, TK007, 80000]
[500, TK005, 40000]
[400, TK004, 20000]
[300, TK003, 30000]
[200, TK002, 10000]
[100, TK001, 50000]

Xoa thanh cong!
Danh sach sau khi xoa:
[700, TK007, 80000]
[500, TK005, 40000]
[400, TK004, 20000]
[300, TK003, 30000]
[200, TK002, 10000]
[100, TK001, 50000]


C:\Users\trieu\source\re
```

Nội dung tiếp theo

Giới thiệu lớp array