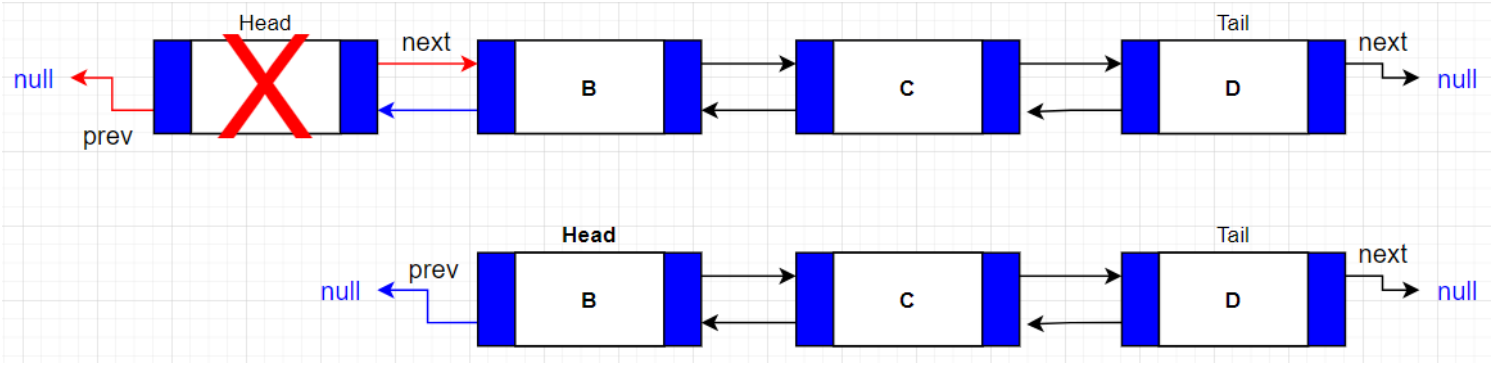
Bài 3.6: Xóa một node khỏi DSLK đôi

- ✓ Xóa node ở đầu
- ✓ Xóa node ở cuối
- ✓ Xóa node ở giữa
- ✓ Ví dụ minh họa
- ✓ Bài tập thực hành

Xóa node ở đầu DSLK

- Bước 1. Lưu node head ra biến r.
- Bước 2. Nếu head->next khác null, gán head->next->prev = null;
- Bước 3. Gán head = head->next;
- Bước 4. Xóa node r.
- Bước 5. Trả về kết quả xóa node head.

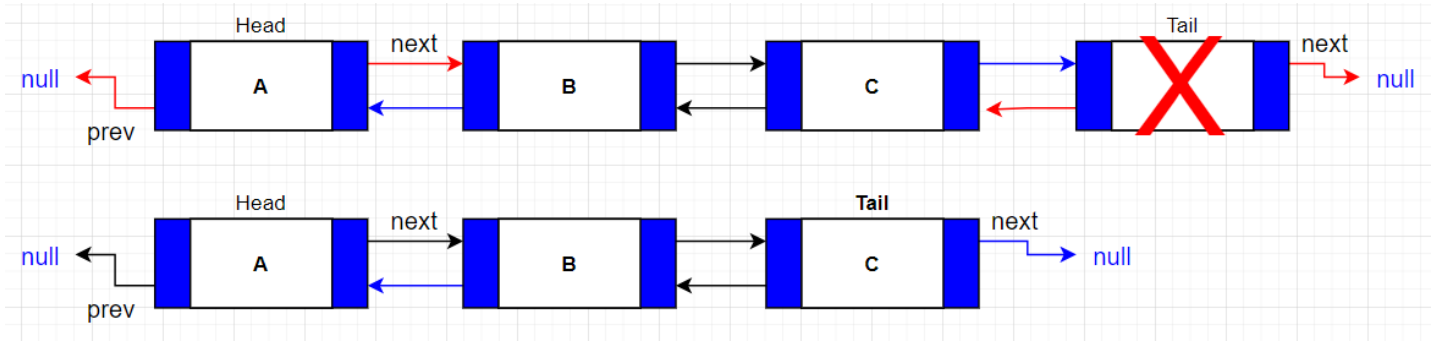


Code mẫu

```
bool removeHead() {
    Node<T>* r = head;                  // lưu lại head
    if (head->next != nullptr) {        // nếu next của head khác null
        head->next->prev = nullptr;     // cập nhật prev của next của head
    }
    head = head->next;                  // cập nhật head mới
    delete r;                           // xóa head
    return true;                        // xóa thành công
}
```

Xóa node ở cuối DSLK

- Bước 1. Lưu node tail ra biến r.
- Bước 2. Nếu tail->prev khác null, gán tail->prev->next = null;
- Bước 3. Gán tail = tail->prev;
- Bước 4. Xóa node r.
- Bước 5. Trả về kết quả xóa node tail.



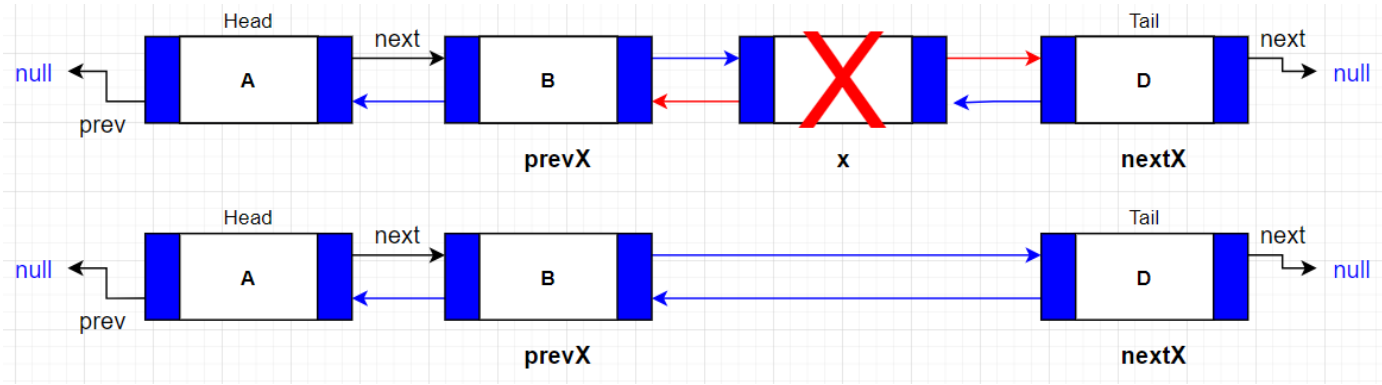
Code mẫu

```
bool removeTail(Node<T>* prevX) {
    Node<T>* r = tail;                  // lưu lại tail
    if (prevX != nullptr) {             // nếu node trước tail khác null
        prevX->next = nullptr;          // cập nhật next của prevX
    }
    tail = prevX;                       // cập nhật lại tail
    delete r;                           // xóa node r
    return true;                        // xóa thành công
}
```

Xóa node giữa trong DSLK

Gọi node cần xóa là x, node liền trước x là prevX. Giả sử đã tìm được x và prevX, ta tiến hành xóa:

- Bước 1. Lưu node x ra biến r.
- Bước 2. Gán prevX->next = x->next;
- Bước 3. Gán x = x->next;
- Bước 4. Nếu x khác null, gán x->prev = prevX;
- Bước 5. Xóa node r.
- Bước 6. Trả về kết quả xóa node x.



Xoá node bất kì

```
bool remove(T data) {
    if (isEmpty()) {
        return false;
    }
    Node<T>* x = head;
    Node<T>* prevX = head;
    while (x != nullptr) {
        if (x->data == data) {
            break;
        }
        prevX = x;
        x = x->next;
    }
    if (x == head) {
        return removeHead();
    }
    else if (x == tail) {
        return removeTail(prevX);
    }
    else if (x != nullptr) {
        Node<T>* r = x; // lưu lại node x
        prevX->next = x->next; // cập nhật next của prevX
        x = x->next; // cập nhật x
        if (x != nullptr) { // nếu x mới không null
            x->prev = prevX; // cập nhật prev của x mới
        }
        delete r; // xóa node x cũ
        return true; // xóa thành công
    }
    else {
        return false; // xóa thất bại
    }
}
```

Nội dung tiếp theo

Sắp xếp danh sách liên kết