



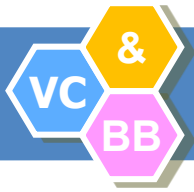
Bộ môn Công nghệ phần mềm
Khoa Công nghệ thông tin
Trường Đại học Khoa học Tự nhiên

KỸ THUẬT LẬP TRÌNH

ThS. Đặng Bình Phương
dbphuong@fit.hcmus.edu.vn

DANH SÁCH LIÊN KẾT





Nội dung

1

Các hình thức tổ chức danh sách

2

Các loại danh sách liên kết

3

Thao tác trên DSLK đơn

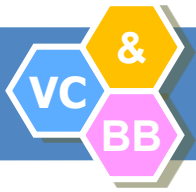
4

Các ứng dụng của DSLK đơn



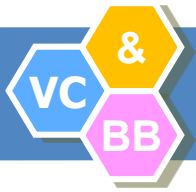
Các hình thức tổ chức danh sách

- ❖ Mỗi liên hệ giữa các phần tử được ngầm hiểu
 - Mỗi phần tử có một chỉ số và ngầm hiểu rằng x_{i+1} nằm sau x_i . Do đó các phần tử phải **nằm cạnh nhau trong bộ nhớ**.
 - **Số lượng phần tử cố định**. Không có thao tác thêm và hủy mà chỉ có thao tác dời chỗ.
 - **Truy xuất ngẫu nhiên** đến từng phần tử nhanh chóng.
 - **Phí bộ nhớ** do không biết trước kích thước.
 - Ví dụ: **mảng một chiều**.



Các hình thức tổ chức danh sách

- ❖ Mỗi liên hệ giữa các phần tử rõ ràng
 - Mỗi phần tử ngoài thông tin bản thân còn có thêm **liên kết** (địa chỉ) **đến phần tử kế tiếp**.
 - Các phần tử **không** cần phải sắp xếp cạnh nhau trong bộ nhớ.
 - Việc **truy xuất** đến một phần tử này đòi hỏi phải **thông qua một phần tử khác**.
 - Tùy nhu cầu, các phần tử sẽ liên kết theo nhiều cách khác nhau tạo thành danh sách liên kết **đơn, kép, vòng**.



Danh sách liên kết

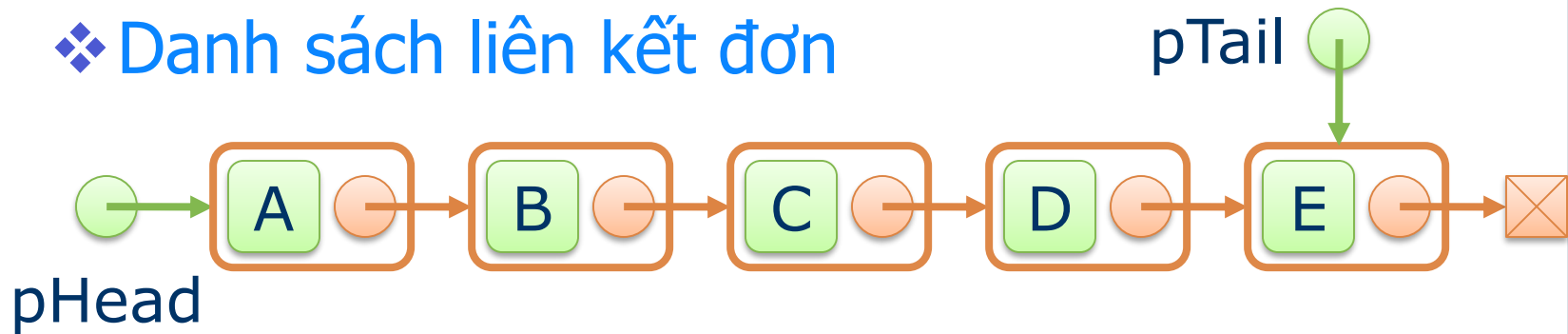
❖ Nhận xét

- **Số nút không cố định**, thay đổi tùy nhu cầu nên đây là cấu trúc động.
- Thích hợp thực hiện các thao tác **chèn** và **hủy** vì không cần phải dời nút mà **chỉ cần sửa các liên kết cho phù hợp**. Thời gian thực hiện không phụ thuộc vào số nút danh sách.
- Tổn bộ nhớ chứa con trỏ liên kết pNext.
- **Truy xuất tuần tự** nên mất thời gian.



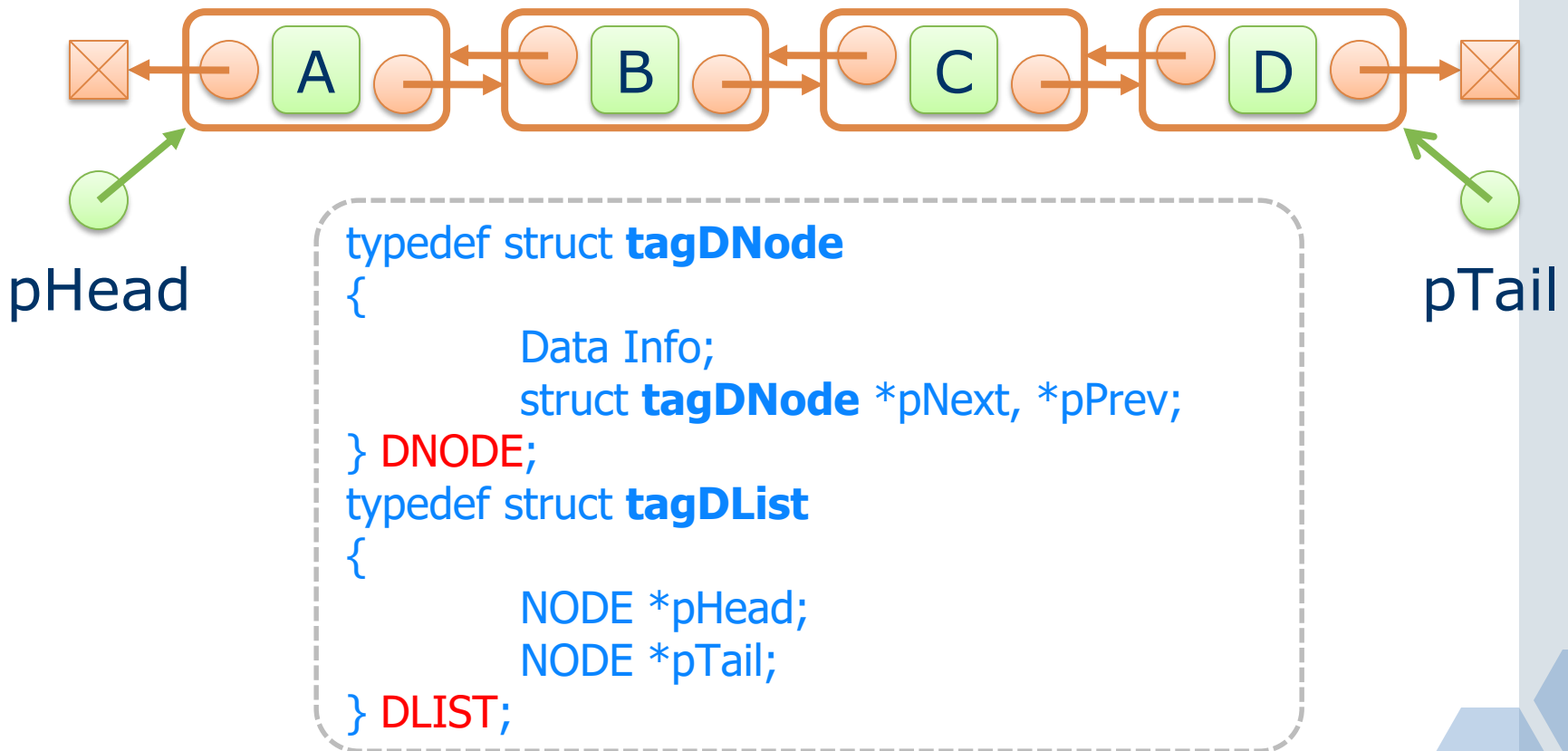
Các loại danh sách liên kết

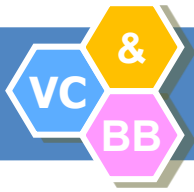
❖ Danh sách liên kết đơn



```
typedef struct tagNode
{
    Data Info;
    struct tagNode *pNext;
} NODE;
typedef struct tagList
{
    NODE *pHead;
    NODE *pTail;
} LIST;
```

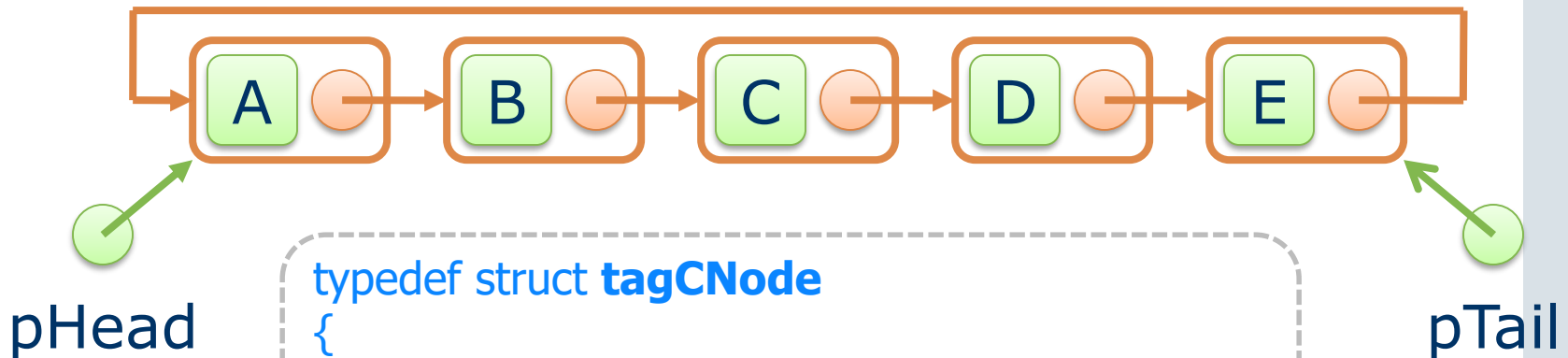
❖ Danh sách liên kết kép (Doubly Linked List)



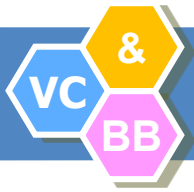


Các loại danh sách liên kết

❖ Danh sách liên kết đơn vòng (Circular Linked List)

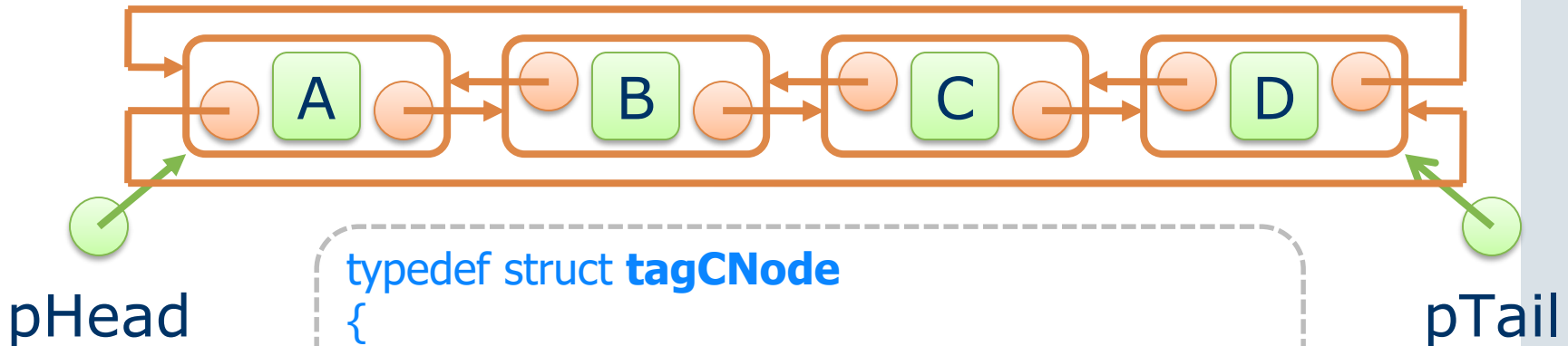


```
typedef struct tagCNode
{
    Data Info;
    struct tagCNode *pNext;
} CNODE;
typedef struct tagCList
{
    NODE *pHead;
    NODE *pTail;
} CLIST;
```

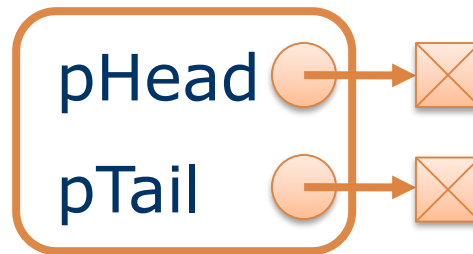
Các loại danh sách liên kết

❖ Danh sách liên kết kép vòng (Circular Linked List)

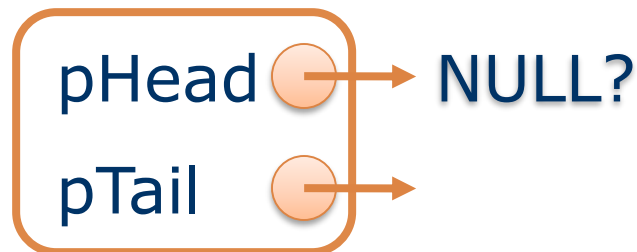


```
typedef struct tagCNode
{
    Data Info;
    struct tagCNode *pNext, *pPrev;
} CNODE;
typedef struct tagCList
{
    NODE *pHead;
    NODE *pTail;
} CLIST;
```

❖ Khởi tạo danh sách



❖ Kiểm tra danh sách có rỗng hay không



❖ Tạo một nút mới



❖ Xác định con trỏ của nút thứ i trong danh sách

- $p = \text{pHead}$
- $p = p \rightarrow \text{pNext}$ i lần trong khi $p \neq \text{NULL}$ rồi return lại con trỏ p hiện tại

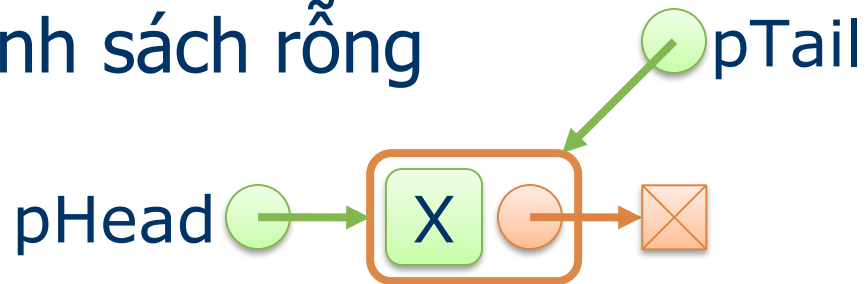
❖ Xác định vị trí của nút p trong danh sách

- Tương tự như trên nhưng trả lại vị trí

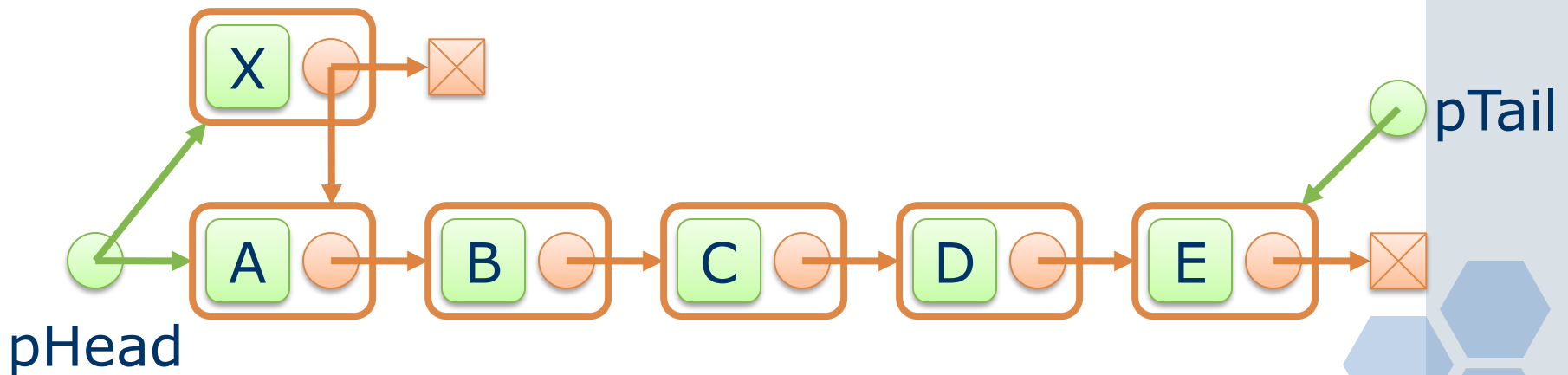
Danh sách liên kết đơn

❖ Chèn một nút vào đầu danh sách

- Danh sách rỗng



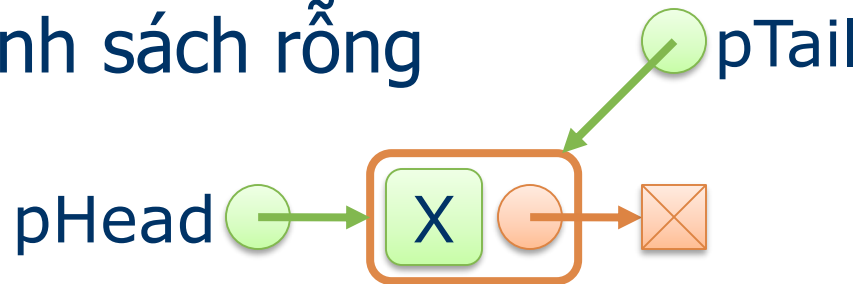
- Danh sách không rỗng



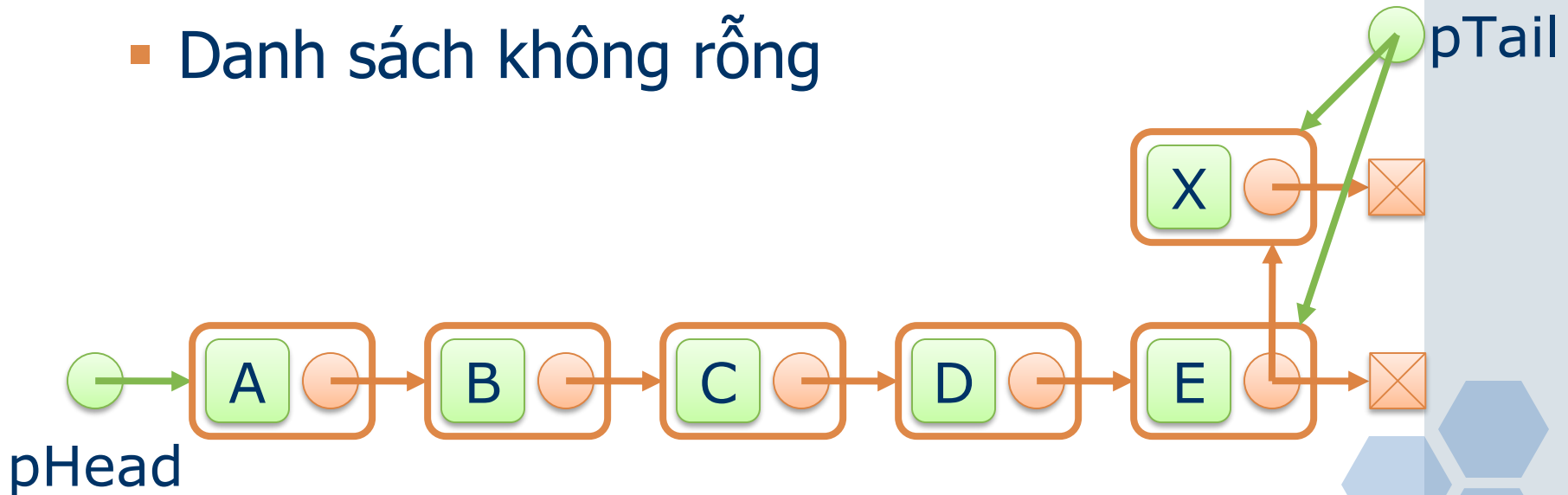
Danh sách liên kết đơn

❖ Thêm một nút vào cuối danh sách

- Danh sách rỗng



- Danh sách không rỗng

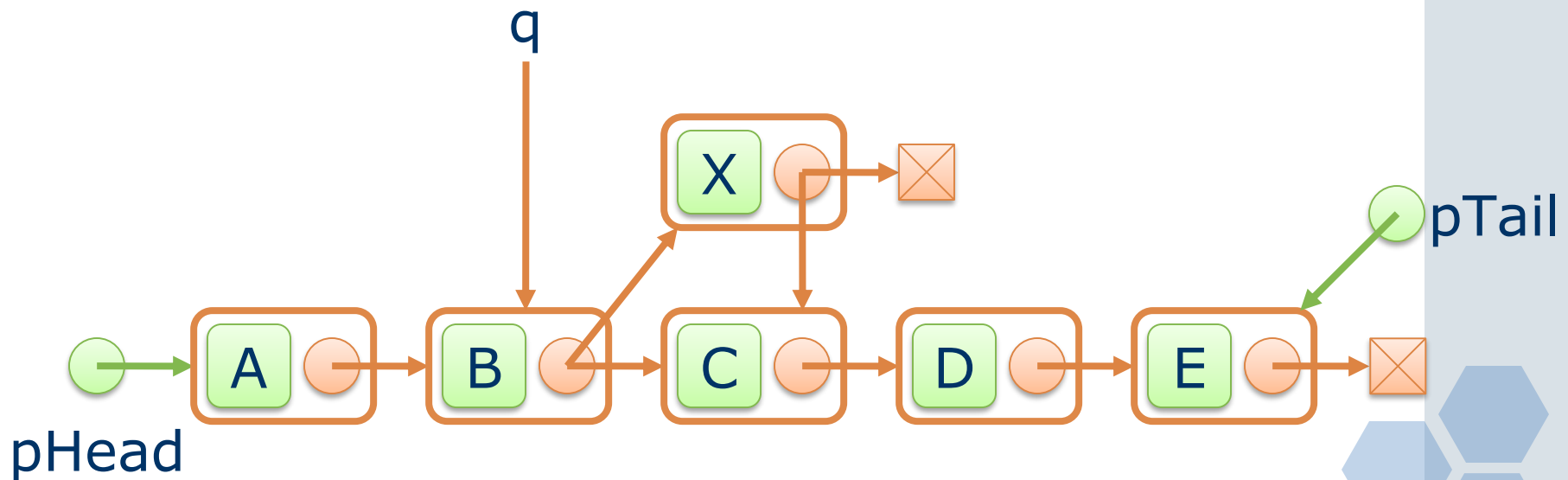




Danh sách liên kết đơn

❖ Thêm một nút vào sau nút q

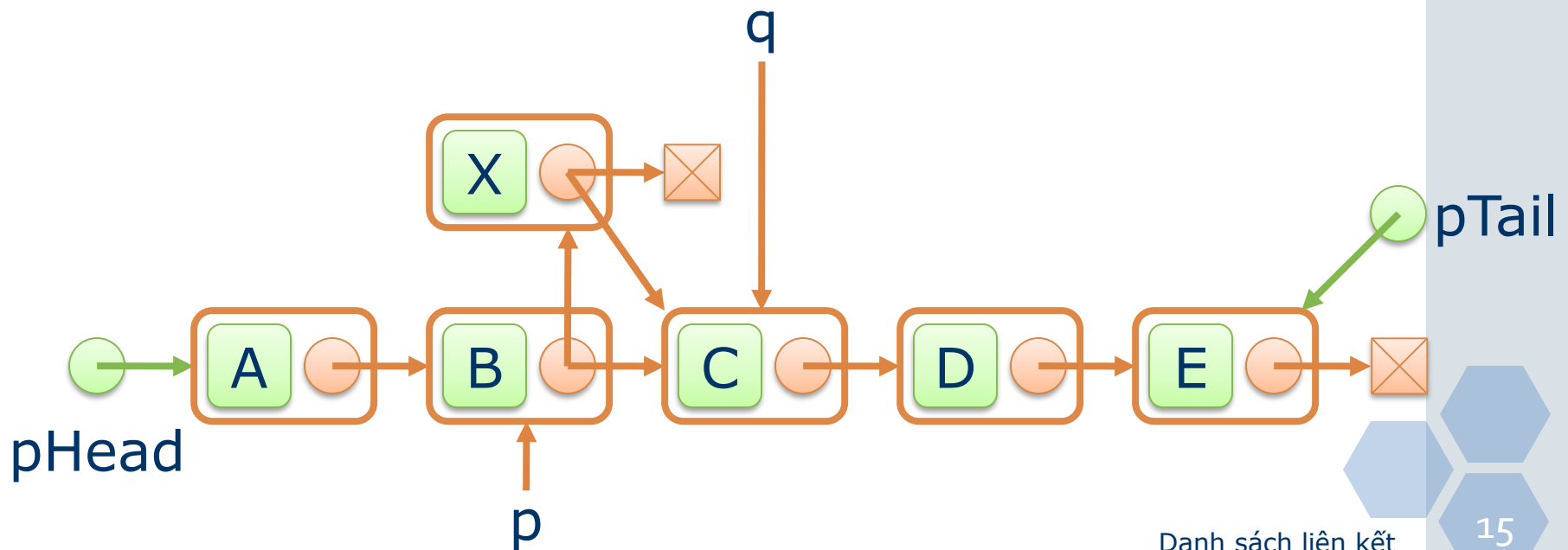
- $q == \text{NULL} \rightarrow$ chèn vào đầu danh sách
- $q \neq \text{NULL}$



Danh sách liên kết đơn

❖ Thêm một nút vào trước nút q

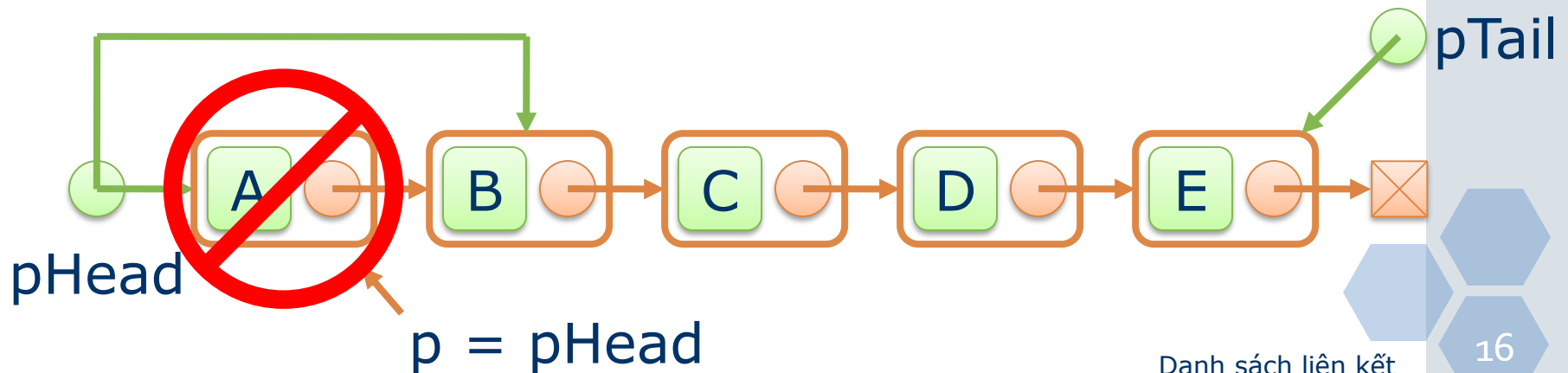
- $q == \text{NULL}$ → chèn vào đầu danh sách
- $q \neq \text{NULL}$ → Tìm nút p trước q rồi thêm vào sau nút p này.



Danh sách liên kết đơn

❖ Hủy một nút đầu danh sách

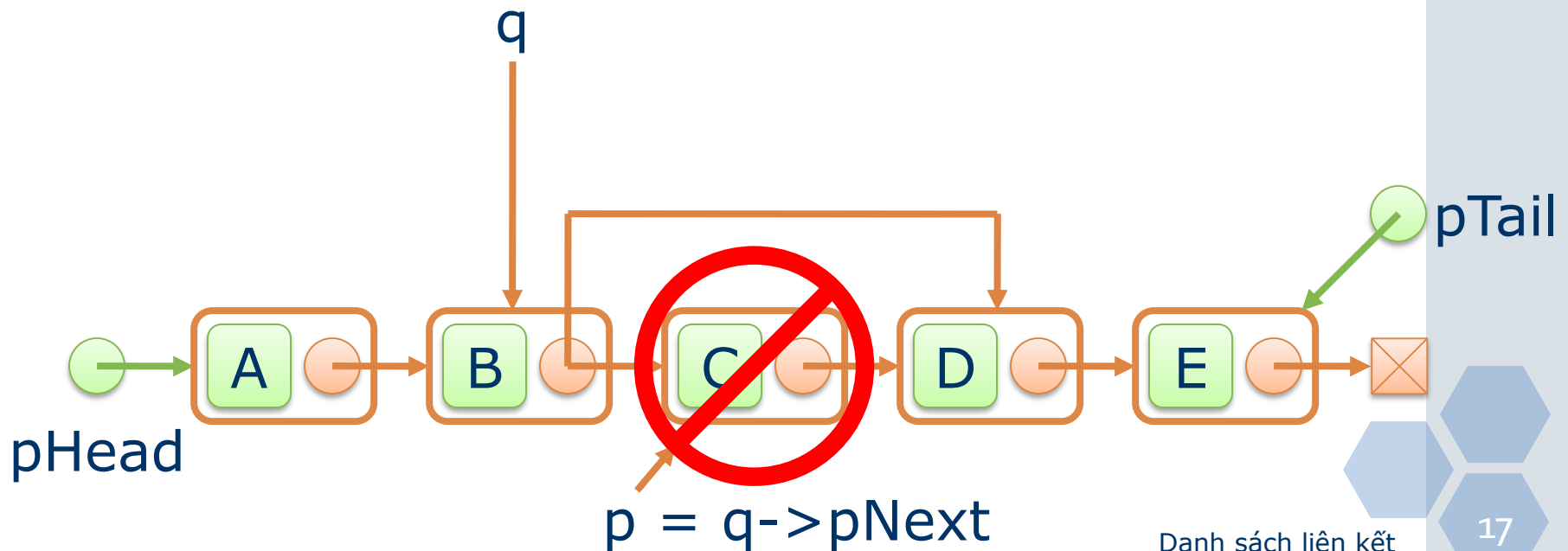
- Danh sách rỗng \rightarrow không làm gì cả
- Danh sách không rỗng (nếu sau khi hủy mà $pHead = NULL$ thì $pTail = NULL$)

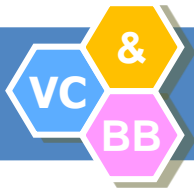


Danh sách liên kết đơn

❖ Hủy một nút sau nút q

- $q == \text{NULL} \rightarrow$ hủy nút đầu danh sách
- $q \neq \text{NULL}$





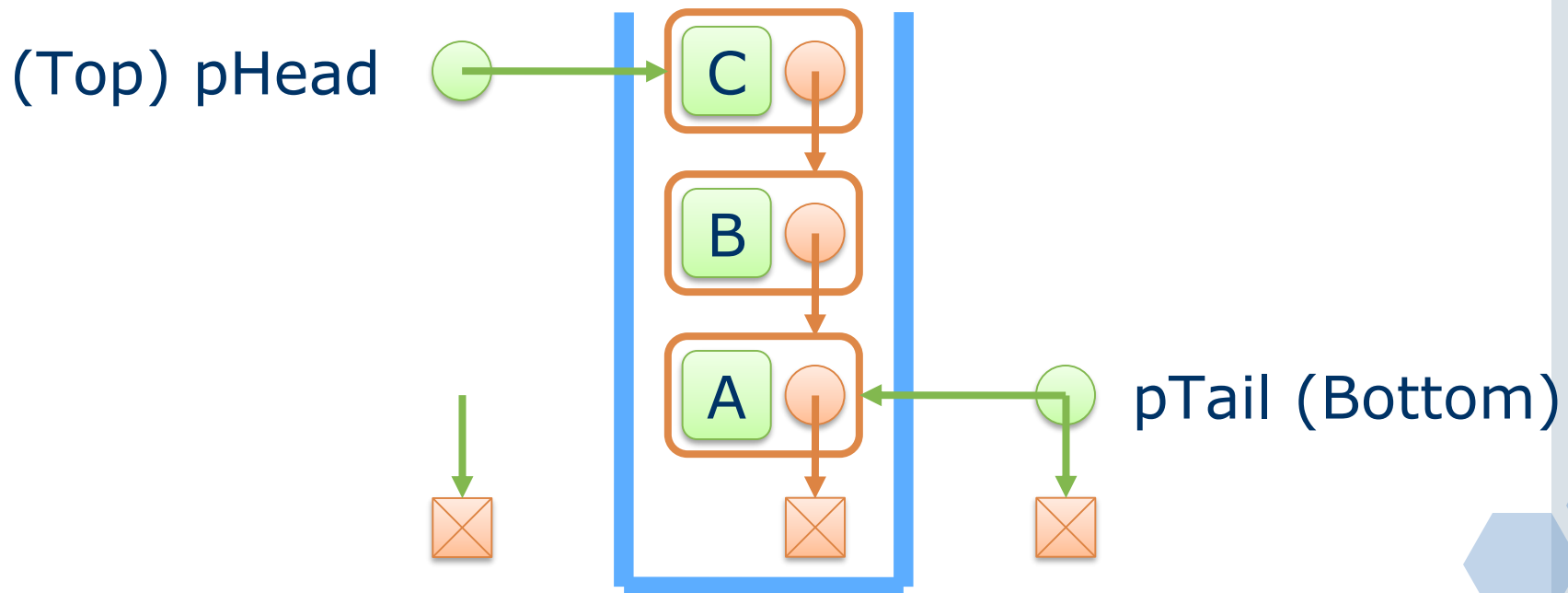
Danh sách liên kết đơn

- ❖ Hủy một nút cuối danh sách
 - Nút cuối p ($p = pTail$)
 - Tìm nút q trước nút p (nếu có)
 - Hủy nút sau nút q
- ❖ Hủy một nút có khóa k ($Info = k$)
 - Tìm nút p có khóa k và hủy nút q trước đó.
 - Hủy nút sau nút q (nếu có)
- ❖ Hủy toàn bộ danh sách
- ❖ Duyệt danh sách để in/tìm/đếm các nút



❖ Stack (Ngăn xếp)

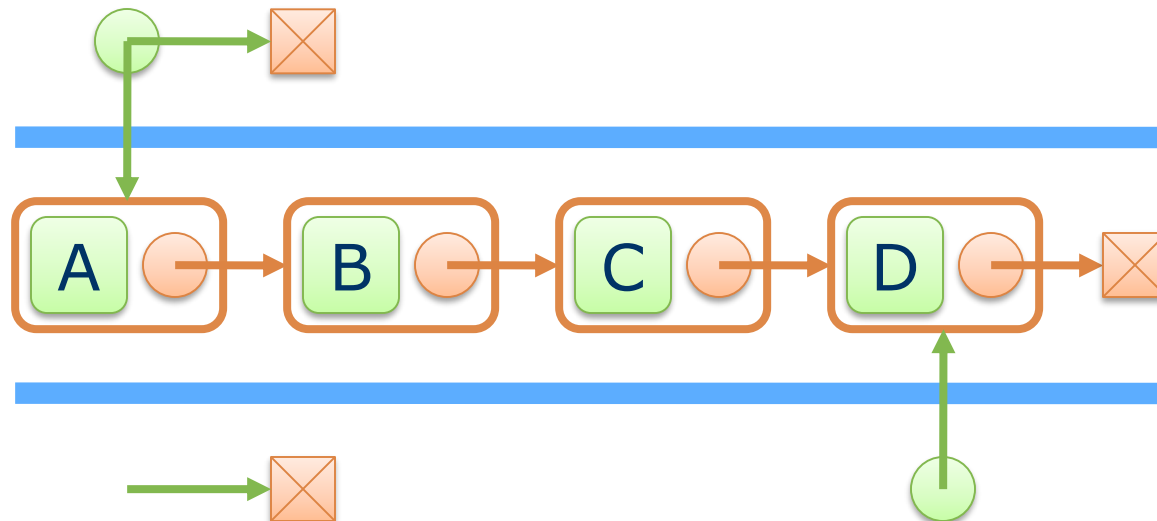
- Làm việc theo cơ chế **LIFO** (Last In First Out)



❖ Queue (Hàng đợi)

- Làm việc theo cơ chế **FIFO** (First In First Out)

pHead (Front)



pTail (Rear)