

# DANH SÁCH LIÊN KẾT (LINKED LISTS)



# Nội dung

2

- Giới thiệu
- Danh sách liên kết đơn (**Single Linked List**)
- Danh sách liên kết đôi (**Double Linked List**)
- Danh sách liên kết vòng (**Circular Linked List**)

# DSLK đơn

3

## □ Các thao tác cơ bản

- Tạo danh sách rỗng
- Thêm một phần tử vào danh sách
- Duyệt danh sách
- Tìm kiếm một giá trị trên danh sách
- Xóa một phần tử ra khỏi danh sách
- Hủy toàn bộ danh sách
- ...

# DSLK đơn – Các thao tác cơ sở

4

- Xóa một node của danh sách
  - Xóa node đầu của danh sách
  - Xóa node cuối danh sách
  - Xóa node có khoá k
  - .....

# DSLK đơn – Các thao tác cơ sở

5

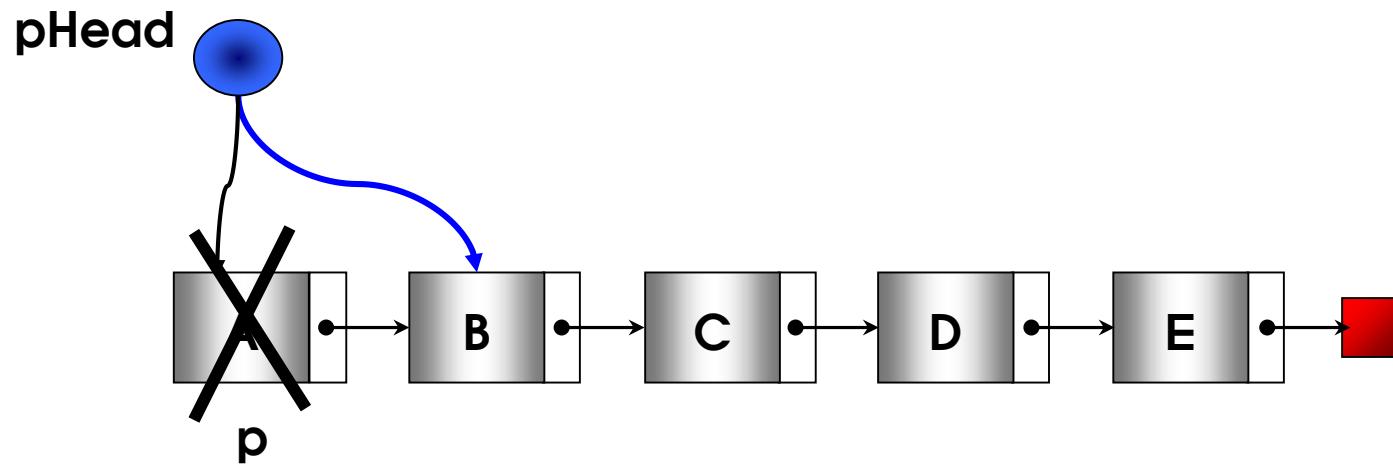
## Xóa node đầu của danh sách

- ❑ Gọi p là node đầu của danh sách (pHead)
- ❑ Cho pHead trỏ vào node sau node p (là p->pNext)
- ❑ Giải phóng vùng nhớ mà p trỏ tới

# DSLK đơn – Các thao tác cơ sở

6

- Xóa một node của danh sách



**pHead = pHead->pNext;**

# DSLK đơn – Các thao tác cơ sở

7

```
node DelHead(pNode pHead){  
    if(pHead == NULL){  
        printf("\n Khong co phan tu de xoa");  
    }else{  
        pHead = pHead->next;  
    }  
    return pHead;  
}
```

# DSLK đơn – Các thao tác cơ sở

8

- Xóa một node của danh sách
  - Xóa node đầu của danh sách
  - Xóa node cuối trong danh sách
  - Xóa node có khoá k/tại vị trí chỉ định
  - .....

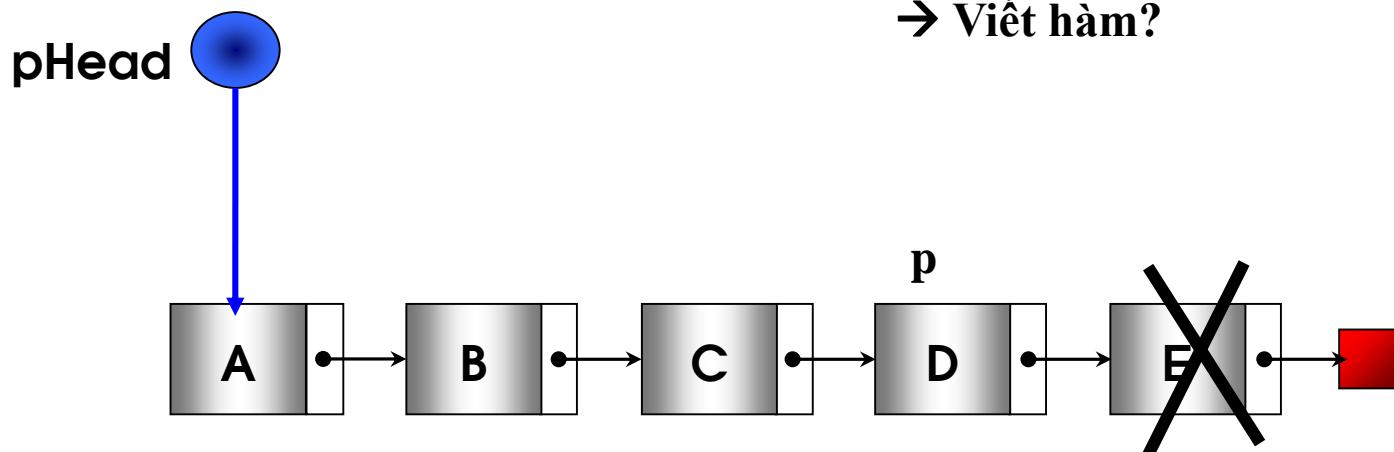
□ Xóa phần tử cuối danh sách

\_ TH1: ds rỗng

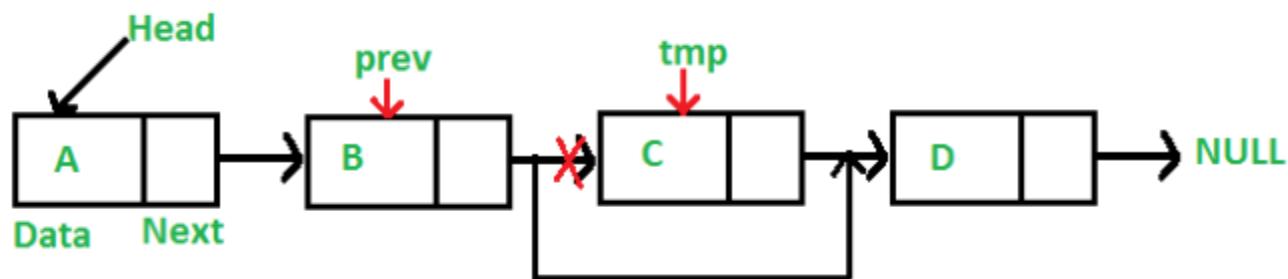
\_ TH2: ds có 1 phần tử

\_ TH3: ds có nhiều phần tử

→ Viết hàm?



- Xóa phần tử ở vị trí pos
- → Quy trình tìm vị trí pos giống chèn thêm phần tử ở vị trí pos
- → Lệnh xóa?



# DSLK đơn – Các thao tác cơ sở

11

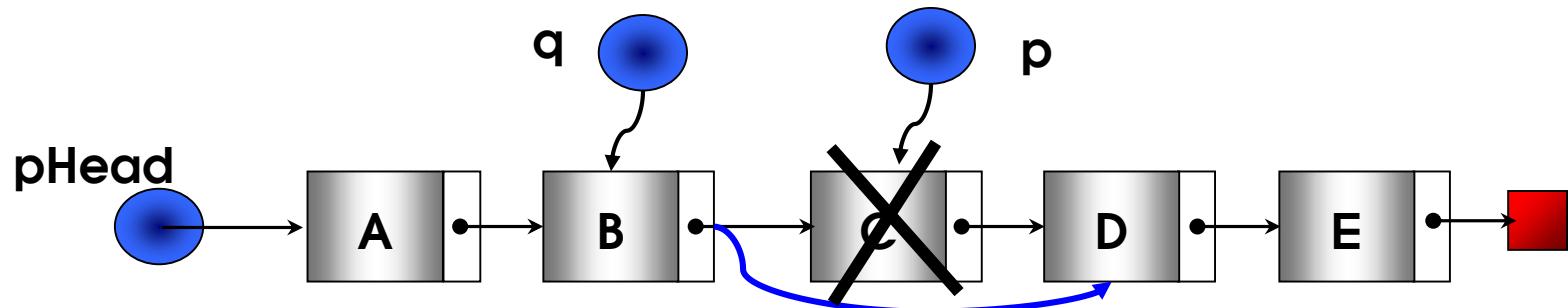
## Xóa node sau node q trong danh sách

- Điều kiện để có thể xóa được node sau q là:
  - q phải khác NULL ( $q \neq \text{NULL}$ )
  - Node sau q phải khác NULL ( $q->\text{next} \neq \text{NULL}$ )
- Có các thao tác:
  - Gọi  $p$  là node sau  $q$
  - Cho vùng  $\text{next}$  của  $q$  trỏ vào node đứng sau  $p$

# DSLK đơn – Các thao tác cơ sở

12

Xóa node sau node q trong danh sách



`q->next = p->next;`

`delete p;`

# DSLK đơn – Các thao tác cơ sở

13

- **Thuật toán: Hủy 1 phần tử có khóa k**
  - Bước 1:
    - Tìm phần tử **p** có khóa k và phần tử **q** đứng trước nó
  - Bước 2:
    - Nếu  $(p \neq \text{NULL})$  thì // tìm thấy k
      - Hủy **p** ra khỏi ds; tương tự hủy phần tử sau q;
    - Ngược lại
      - Báo không có k

# DSLK đơn – Các thao tác cơ sở

14

- Cài đặt:  
Hủy 1  
phần tử  
có khoá  
k

```
int removeNode (pNode pHead, int k)
{
    pNode p = pHead;
    pNode q = NULL;
    while (p != NULL)
    {
        if (p->data == k) break;
        q = p;
        p = p->next;
    }
    if (p == NULL) { printf("Không tìm thấy k"); return 0;}
    else if (q == NULL)
        // thực hiện xóa phần tử đầu ds là p
    else
        // thực hiện xóa phần tử p sau q
        q->next = p->next;
}
```

Tìm phần tử **p** có khóa k và phần tử **q** đứng trước nó

# Bài tập

15

- Viết chương trình đếm số phần tử trong danh sách

# Nội dung

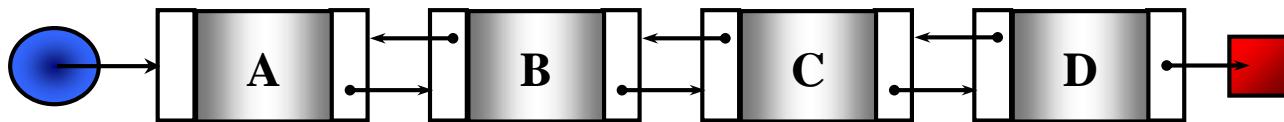
16

- Giới thiệu
- Danh sách liên kết đơn (**Single Linked List**)
- Danh sách liên kết đôi (**Double Linked List**)
- Danh sách liên kết vòng (**Circular Linked List**)

# Danh sách liên kết đôi (DSLK đôi)

17

- ☐ Là danh sách mà mỗi phần tử trong danh sách có kết nối với 1 phần tử đứng trước và 1 phần tử đứng sau nó



# DSLK đôi – Khai báo cấu trúc

18

- Dùng hai con trỏ:
  - Prev liên kết với phần tử đứng trước
  - Next liên kết với phần tử đứng sau

```
struct DNode
{
    DataType data;
    DNode* pPre;          // trỏ đến phần tử đứng trước
    DNode* pNext;         // trỏ đến phần tử đứng sau
};
```

# DSLK đôi – Tạo nút mới

19

- Hàm tạo nút:

**DNode\*** **getNode** (**DataType** x)

{

**DNode \***p;

**p = new DNode;** // Cấp phát vùng nhớ cho phần tử

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

**cout<<“Khong du bo nho”;** **return NULL;**

}

**p->data = x;** // Gán thông tin cho phần tử p

**p->Prev = p->Next = NULL;**

**return p;**

}



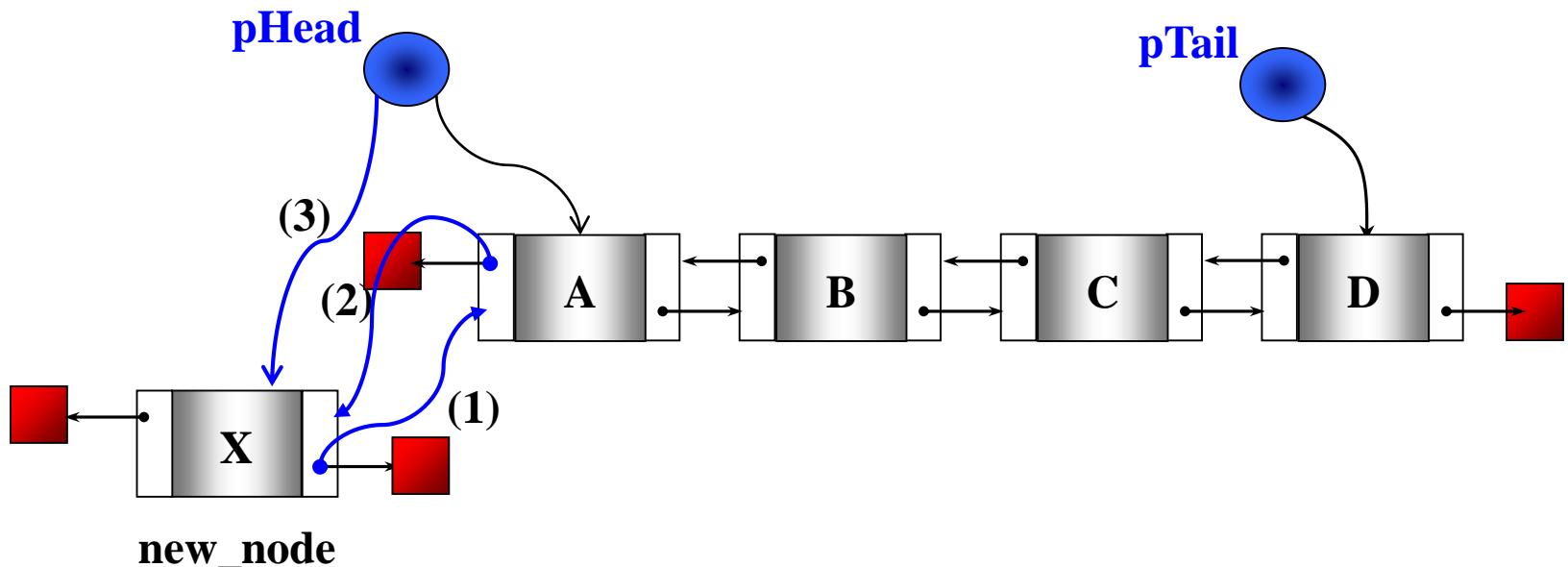
# DSLK đôi – Thêm 1 nút vào ds

20

- Có 4 loại thao tác chèn new\_node vào danh sách:
  - Cách 1: Chèn vào đầu danh sách
  - Cách 2: Chèn vào cuối danh sách
  - Cách 3 : Chèn vào danh sách sau một phần tử q
  - Cách 4 : Chèn vào danh sách trước một phần tử q

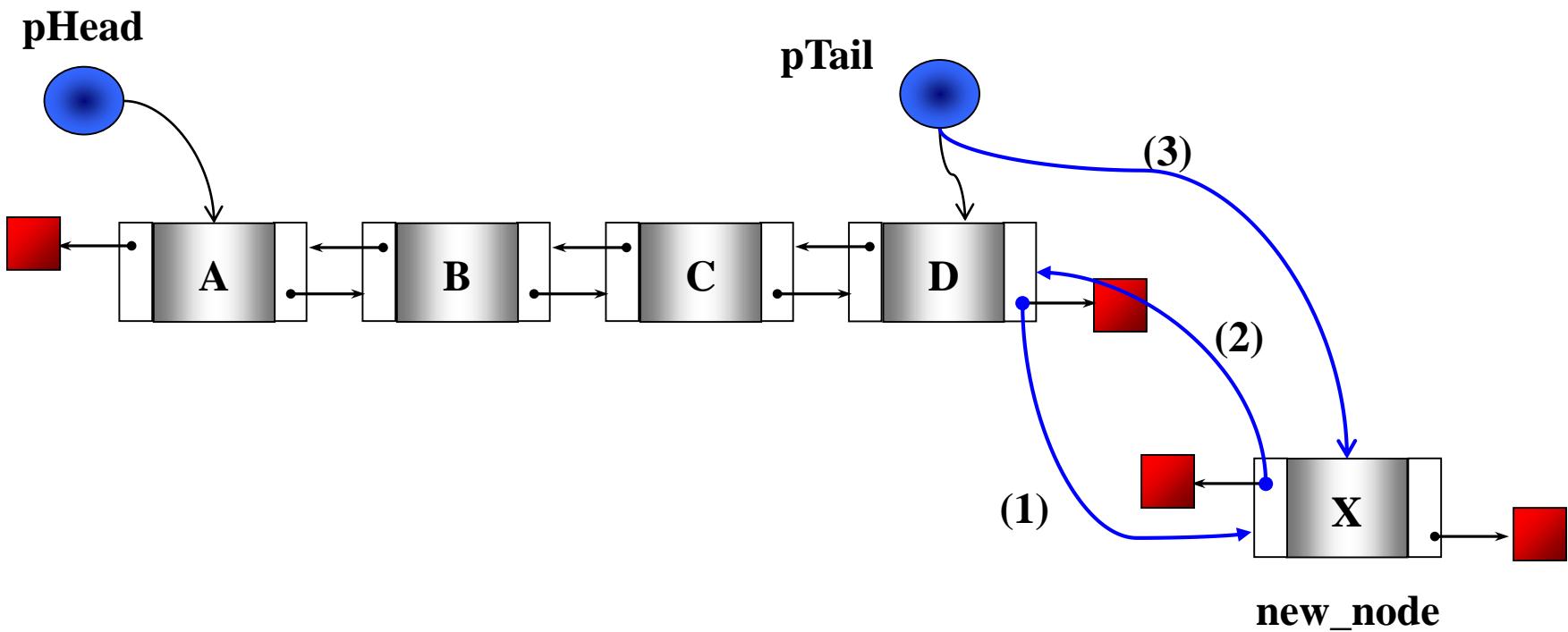
# DSLK đôi – Thêm vào đầu ds

21



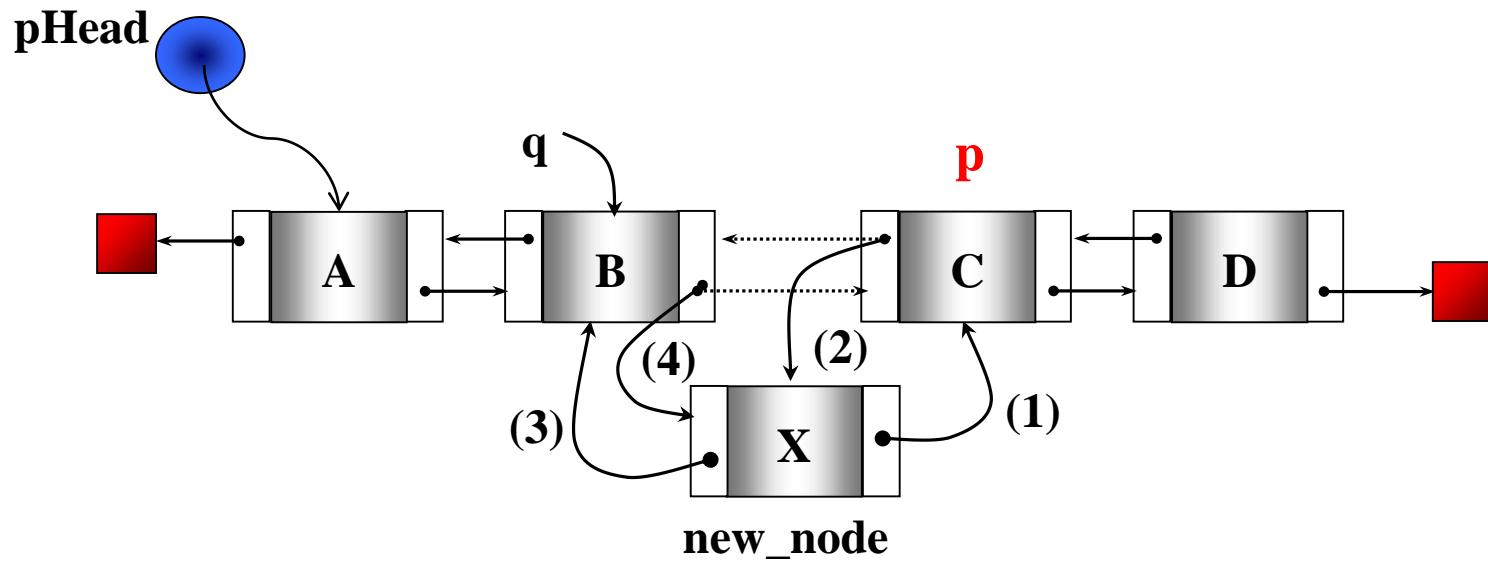
# DSLK đôi – Thêm vào cuối ds

22



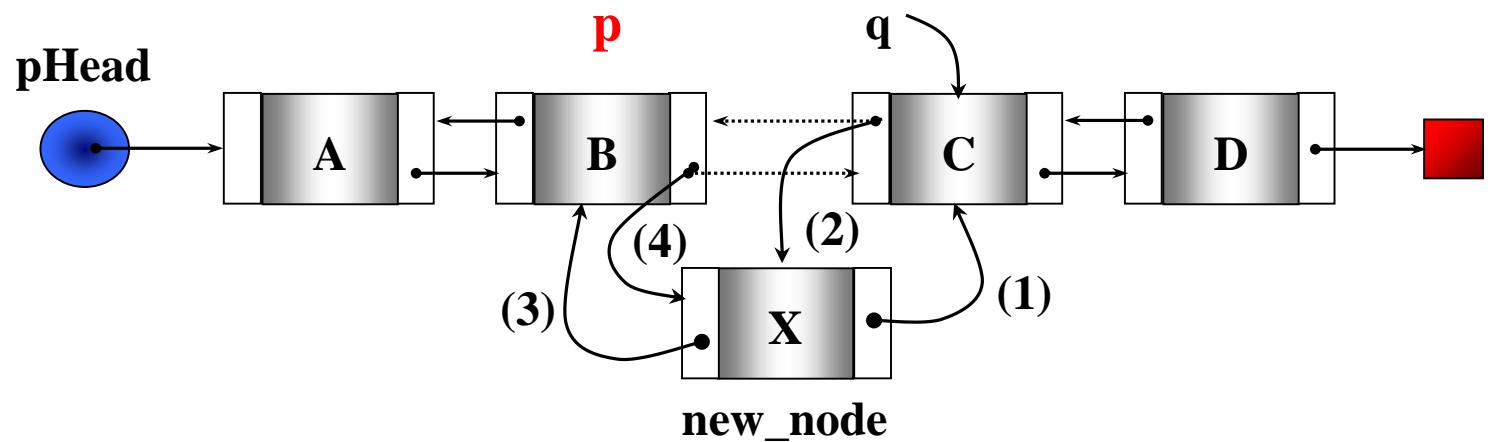
# DSLK đôi – Chèn vào sau q

23



# DSLK đôi – Chèn vào trước q

24



# DSLK đôi – Hủy phần tử

25

- Có 5 loại thao tác thông dụng hủy một phần tử ra khỏi danh sách liên kết đôi:
  - Hủy phần tử đầu ds
  - Hủy phần tử cuối ds
  - Hủy một phần tử đứng sau phần tử q
  - Hủy một phần tử đứng trước phần tử q
  - Hủy 1 phần tử có khóa k

# DSLK đôi – Nhận xét

26

- DSLK đôi về mặt cơ bản có tính chất giống như DLSK đơn
- Tuy nhiên DLSK đôi có mỗi liên kết hai chiều nên từ một phần tử bất kỳ có thể truy xuất một phần tử bất kỳ khác
- Trong khi trên DLSK đơn ta chỉ có thể truy xuất đến các phần tử đứng sau một phần tử cho trước
- Điều này dẫn đến việc ta có thể dễ dàng hủy phần tử cuối DLSK đôi, còn trên DLSK đơn thao tác này tốn chi phí  $O(n)$

# DSLK đôi – Nhận xét

27

- ❑ Bù lại, xâu đôi tốn chi phí gấp đôi so với xâu đơn cho việc lưu trữ các mối liên kết. Điều này khiến việc cập nhật cũng nặng nề hơn trong một số trường hợp. Như vậy ta cần cân nhắc lựa chọn CTDL hợp lý khi cài đặt cho một ứng dụng cụ thể

# Nội dung

28

- Giới thiệu
- Danh sách liên kết đơn (**Single Linked List**)
- Danh sách liên kết đôi (**Double Linked List**)
- Danh sách liên kết vòng (**Circular Linked List**)

# Danh sách liên kết vòng (DSLK vòng)

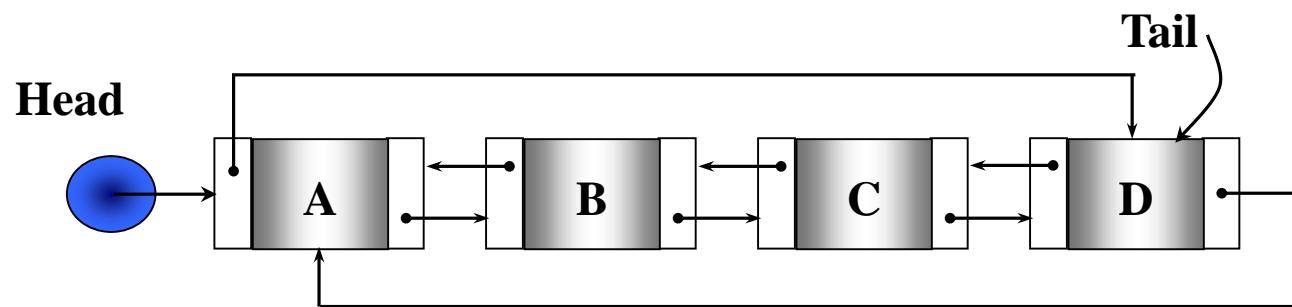
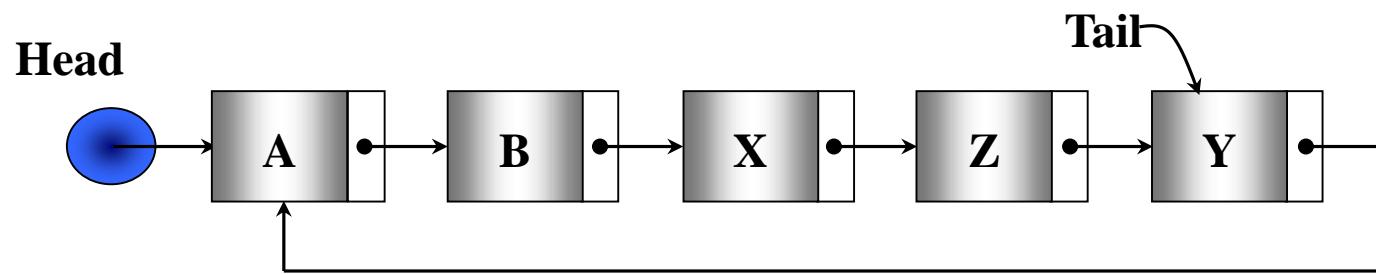
29

- Là một danh sách liên kết đơn (hoặc đôi) mà phần tử cuối danh sách, thay vì mang giá trị **NULL**, trả tới phần tử đầu danh sách
- Đối với danh sách vòng, có thể xuất phát từ một phần tử bất kỳ để duyệt toàn bộ danh sách

# DSLK vòng

30

- Để biểu diễn, có thể sử dụng các kỹ thuật biểu diễn như danh sách đơn (hoặc đôi)



# DSLK vòng – Tìm kiếm

31

- Danh sách vòng không có phần tử đầu danh sách rõ rệt, nhưng ta có thể đánh dấu một phần tử bất kỳ trên danh sách xem như phần tử đầu xâu để kiểm tra việc duyệt đã qua hết các phần tử của danh sách hay chưa