

---

# Chương 4:

## Danh sách liên kết (tt)



## 4.3 Danh sách liên kết mở rộng

### 4.3.1 Danh sách liên kết vòng

- Mô tả
- Cài đặt
- Thao tác

### 4.3.2 Danh sách liên kết kép

- Mô tả
- Cài đặt
- Thao tác

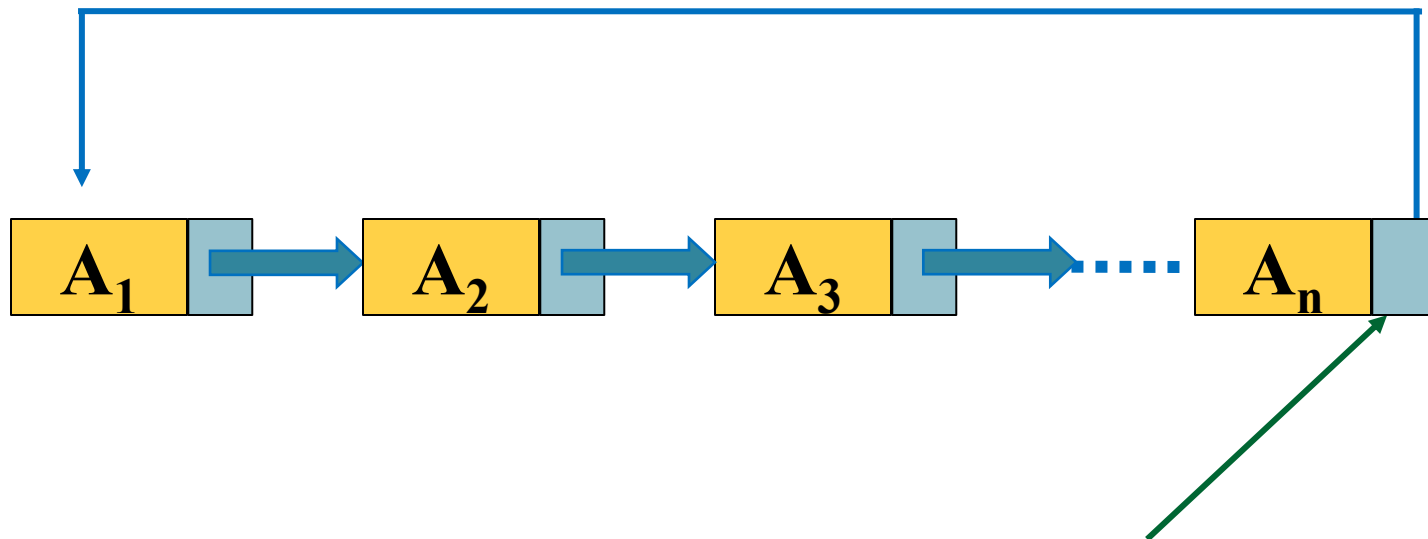
### 4.3.3 Danh sách liên kết đôi vòng

- Mô tả
- Cài đặt
- Thao tác

# Circular Linked List

## 4.3.1 Circular Linked List - Mô tả

- Tương tự như danh sách liên kết đơn.
- Trường next của nút cuối chỉ đến đầu danh sách



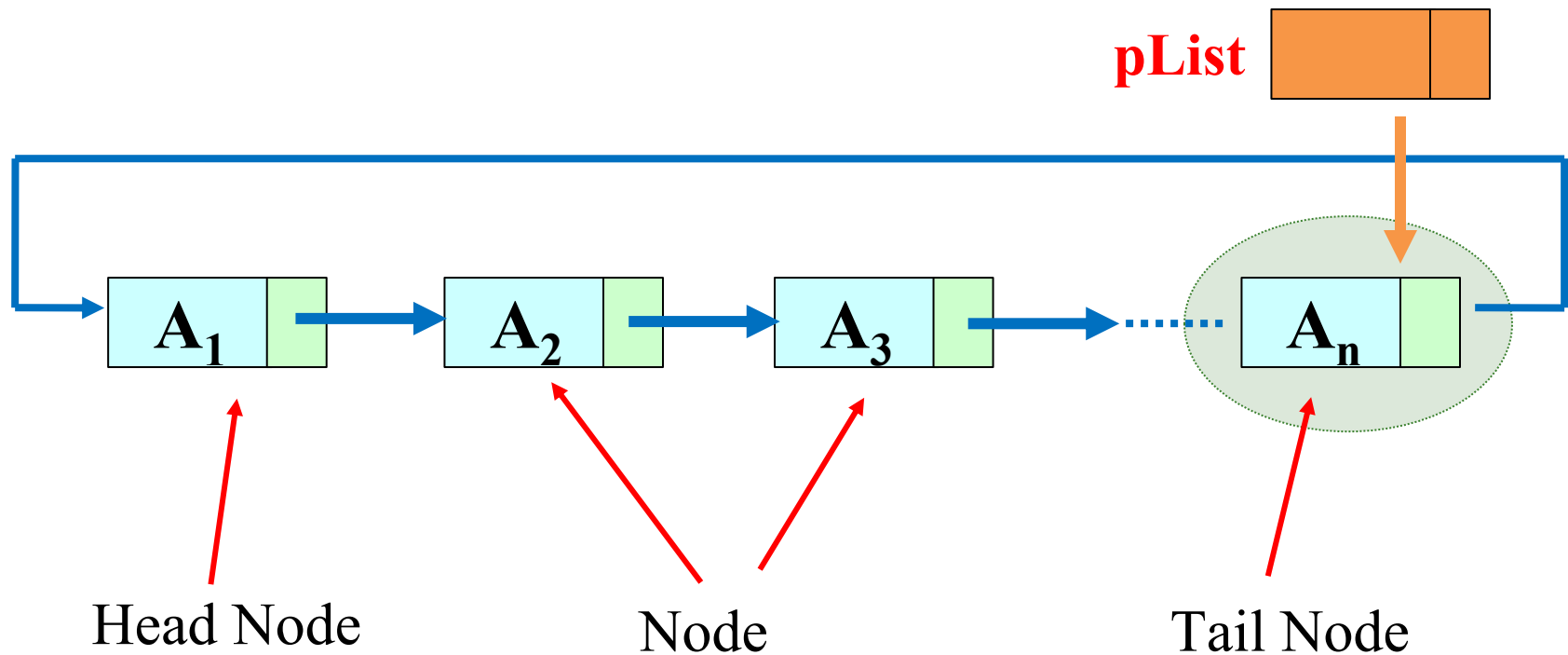
Trường Next của nút cuối ko còn trở đến NULL, mà trở đến nút đầu



## 4.3.1 CLL - Mô tả

### Mô tả Danh sách liên kết đơn vòng


Sử dụng **pList** trỏ đến phần tử cuối của danh sách



## 4.3.1 CLL - Cài đặt

### □ Khai báo

```
struct Nut
{
    DataType data;
    Nut*      next;
};
typedef struct Nut    Node;
```

Node\* pList;       pList trỏ nút cuối ds



## 4.3.1 CLL - Các thao tác

### □ In danh sách

- listInit
- createNode
- printList, printReverse



Phần minh  
họa sẽ dùng  
DataType là  
int

### □ Bổ sung 1 phần tử mới vào danh sách

- insertBegin
- insertAfter
- insertEnd
- insertBefore

### □ Loại bỏ 1 phần tử khỏi danh sách

- deleteFirst
- deleteAfter
- DeleteNode
- deleteLast
- deleteBefore

### □ Các thao tác khác

- searchValue
- sortList
- clearList



## 4.3.1 CLL – Các thao tác

□ **listInit**: Khởi tạo danh sách      // **List Initialization**

```
1. void listInit(Node* &pList)
2. {
3.     pList = new Node;
4.     pList = NULL;
5.     cout<<"Danh sach duoc khoi tao!";
6. }
```



## 4.3.1 CLL - Các thao tác cơ bản

### □ **printList** - In danh sách

- Dùng 1 con trỏ duyệt từ đầu danh sách.
- In thành phần dữ liệu, sau đó cho con trỏ duyệt sang node kế tiếp.
- Thực hiện lặp thao tác trên đến khi nào quay lại phần tử đầu thì dừng

## 4.3.1 CLL - Các thao tác cơ bản

### □ **printList**: In danh sách

```
1. void printList (Node* &pList)
2. {
3.     if (pList == NULL ) return;
4.     Node *p = pList->next;
5.     do
6.     {      cout<< p->data      <<"\t";
7.           p = p->next;          //chuyen nut sau
8.     } while (p != pList->next);
9. }
```

## 4.3.1 CLL - Các thao tác

□ **creatNode**: Tạo node mới có nội dung x

```
1. Node* creatNode (int x)
2. {
3.     Node* new_node;
4.     new_node = new Node;
5.     new_node->data = x;
6.     new_node->next = NULL;
7.     return new_node;
8. }
```

## 4.3.1 CLL - Các thao tác

□ **addFirst**: Kết nạp node đầu tiên vào danh sách

```
1. void addFirst (Node* &pList, int x)
2. {
3.     Node *new_node;
4.     new_node = creatNode (x) ;
5.     pList = new_node;
6.     pList->next = pList;
7. }
```

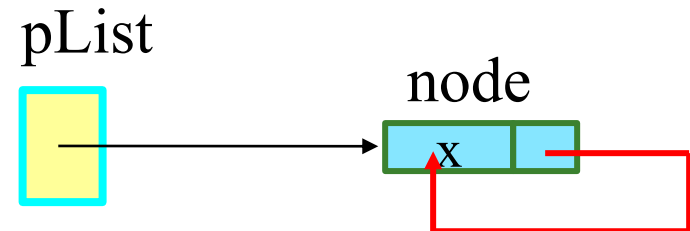
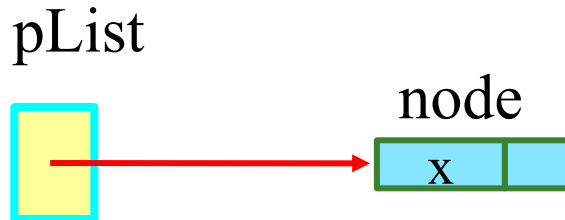
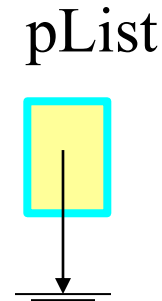
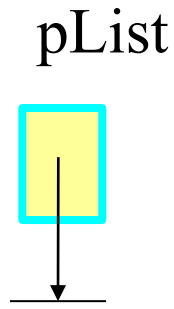
## 4.3.1 CLL - Bổ sung vào đầu ds

### □ **insertBegin**: Bổ sung vào đầu danh sách

- Nếu danh sách trống: Gọi hàm **addFirst**
- Nếu danh sách không trống:
  - + Tạo node mới
  - + Chèn vào trước node đầu tiên.

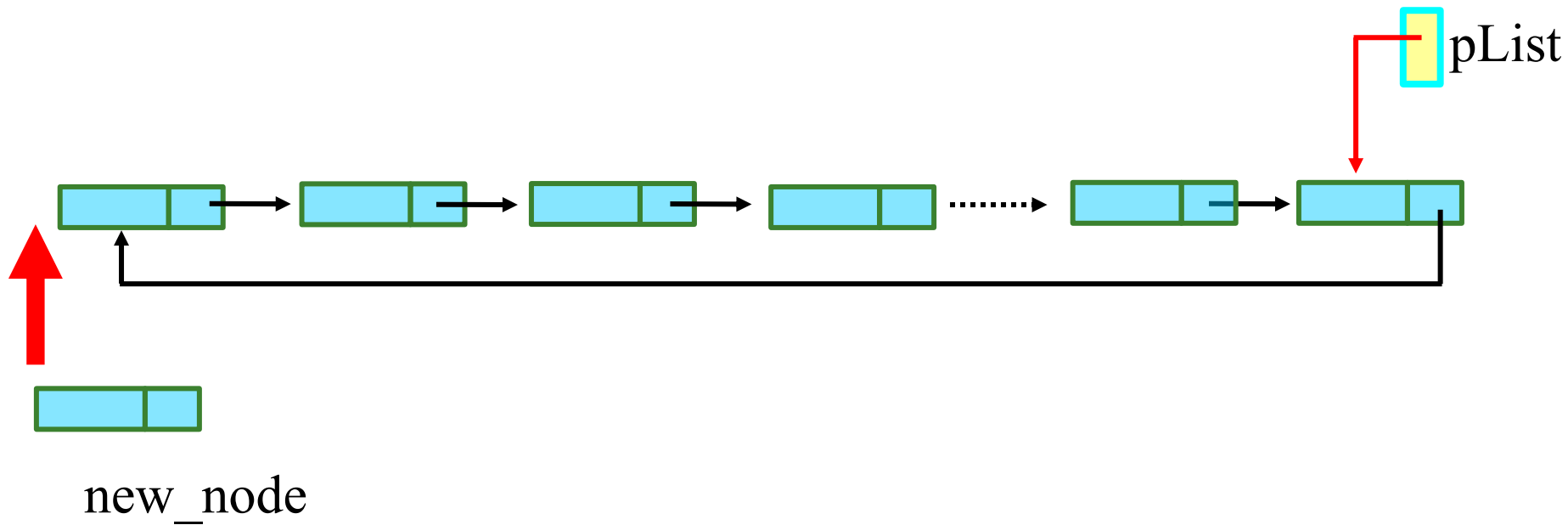
## 4.3.1 CLL - Bổ sung vào đầu ds

□ **insertBegin:**  $pList == \text{NULL}$



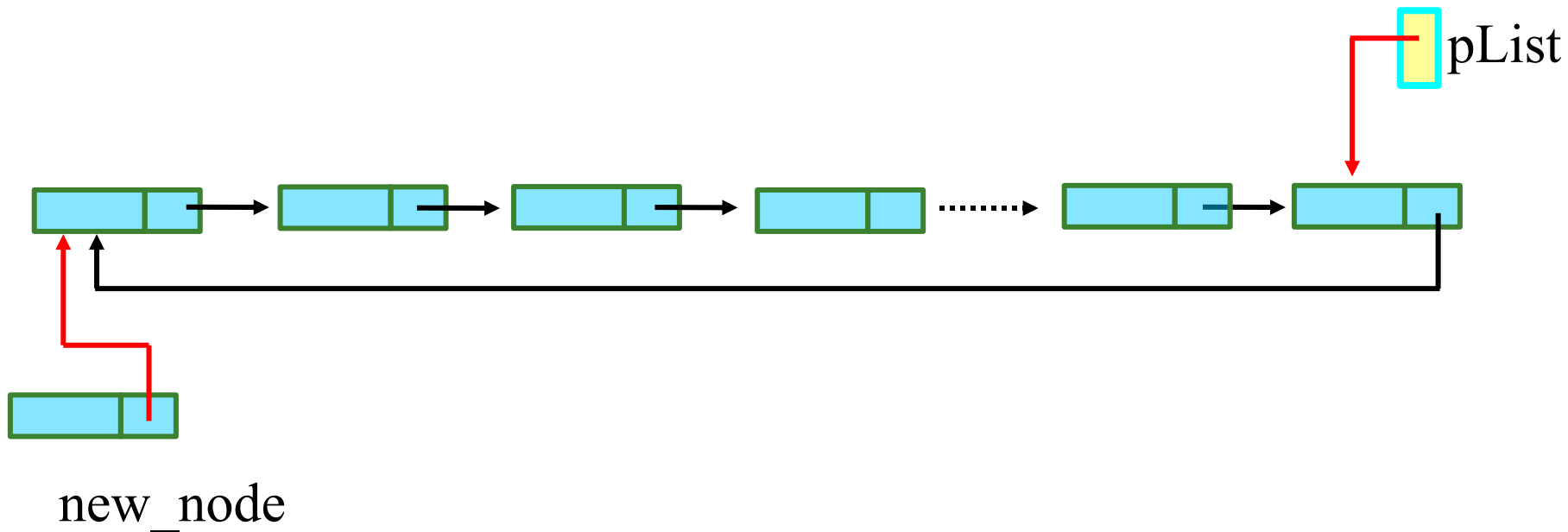
## 4.3.1 CLL - Bổ sung vào đầu ds

□ **insertBegin:**  $pList \neq \text{NULL}$



## 4.3.1 CLL - Bổ sung vào đầu ds

□ **insertBegin:**  $pList \neq \text{NULL}$



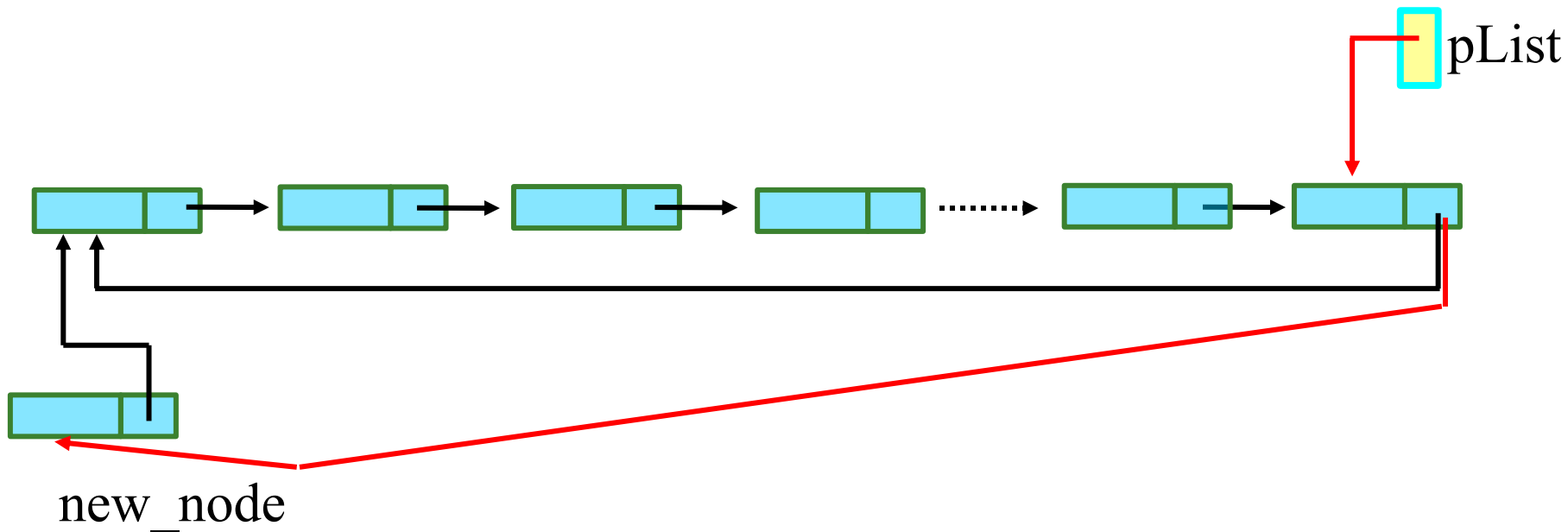
Cho **new\_node** -> next trở tới node đầu:

**$\text{new\_node} \rightarrow \text{next} = \text{pList} \rightarrow \text{next};$**



## 4.3.1 CLL - Bổ sung vào đầu ds

□ **insertBegin:**  $pList \neq \text{NULL}$

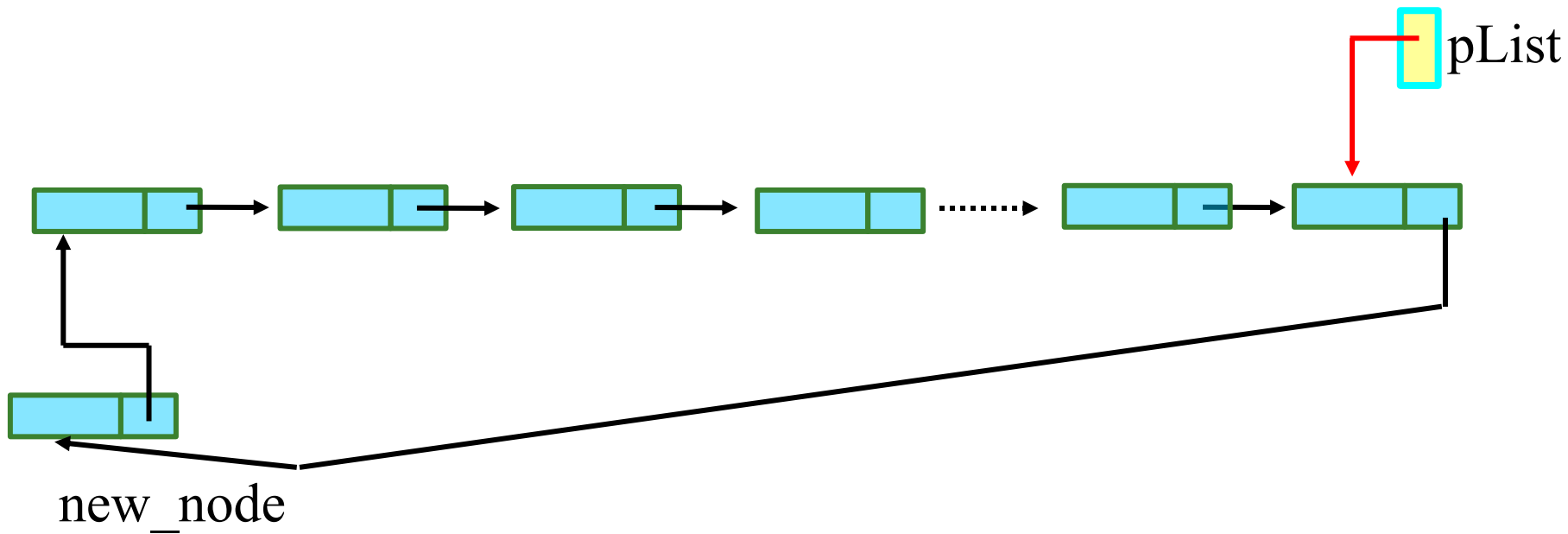


Cho  $pList \rightarrow \text{next}$  trở tới **new\_node**:

**$pList \rightarrow \text{next} = \text{new\_node};$**

## 4.3.1 CLL - Bổ sung vào đầu ds

□ **insertBegin:**  $pList \neq \text{NULL}$



## 4.3.1 CLL - Bổ sung vào đầu ds

```
1. void insertBegin (Node* &pList, int x)
2. {
3.     if (pList == NULL)
4.         addFirst (pList, x);
5.     else
6.     { Node new_node;
7.         new_node = creatNode (x);
8.         new_node->next = pList->next;
9.         pList->next = new_node;
10.    }
11. }
```

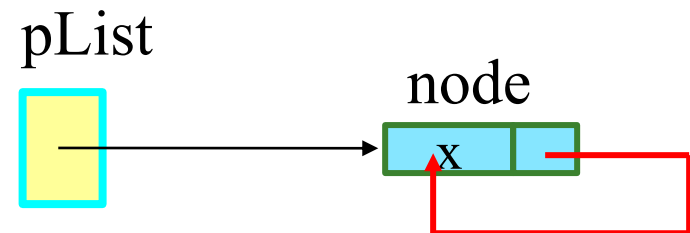
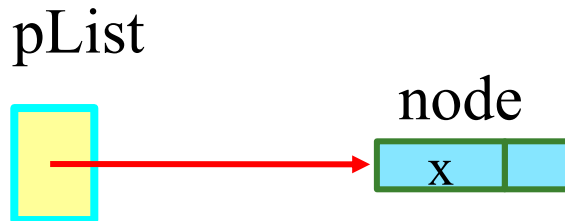
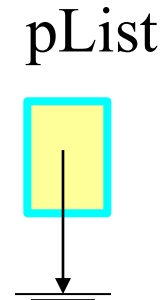
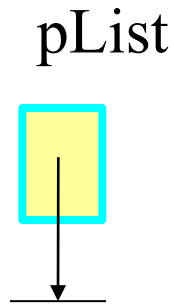
## 4.3.1 CLL - BỔ sung vào cuối ds

### □ **insertEnd**: BỔ sung vào cuối danh sách

- Nếu danh sách trống: Gọi hàm **addFirst**
- Nếu danh sách không trống:
  - + Tạo node mới
  - + Chèn vào sau pList (trước node đầu tiên).
  - + Dịch chuyển pList về node mới chèn.

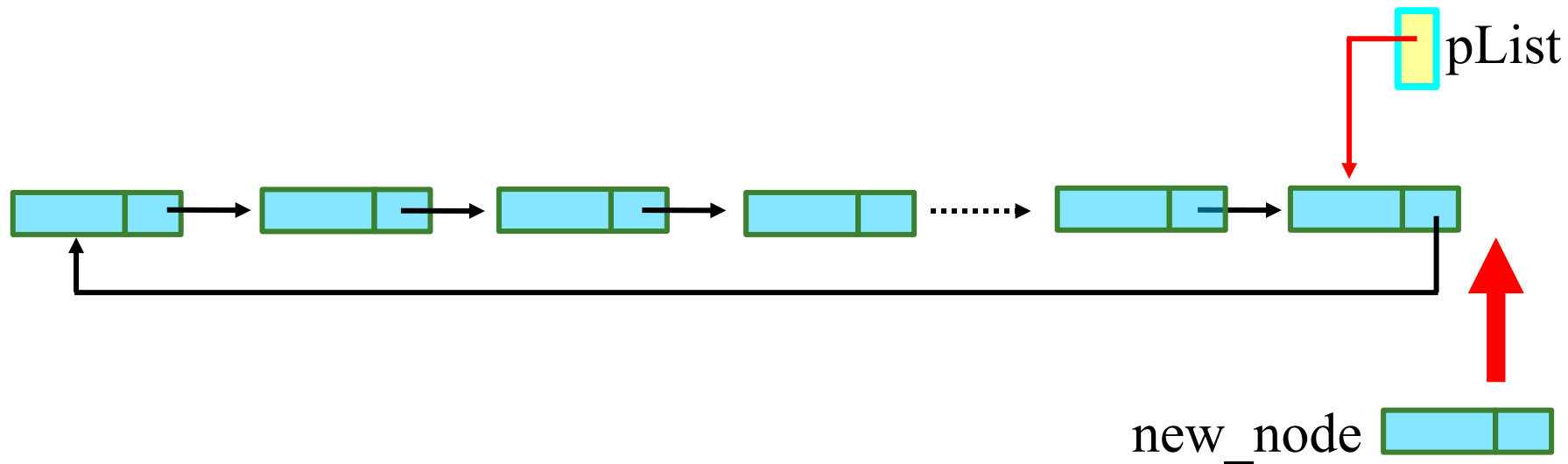
## 4.3.1 CLL - Bổ sung vào cuối ds

□ **insertEnd:**  $pList == \text{NULL}$



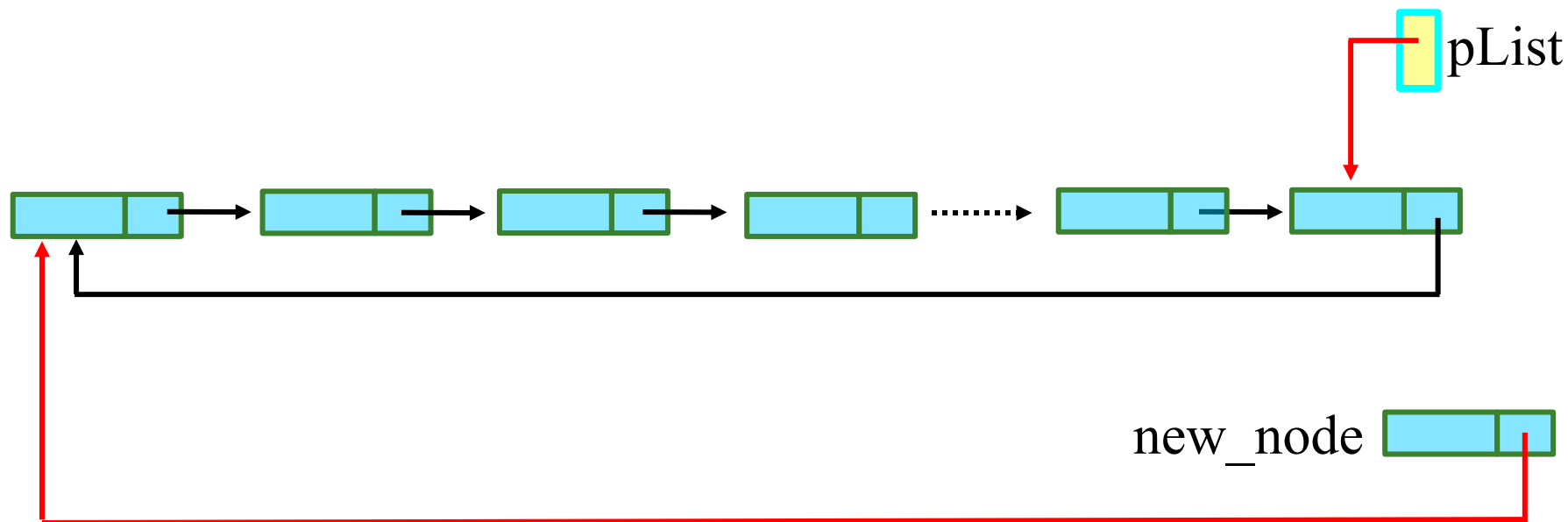
## 4.3.1 CLL - Bổ sung vào cuối ds

□ **insertEnd:** pList  $\neq$  NULL



## 4.3.1 CLL - Bổ sung vào cuối ds

□ **insertEnd:**  $pList \neq \text{NULL}$

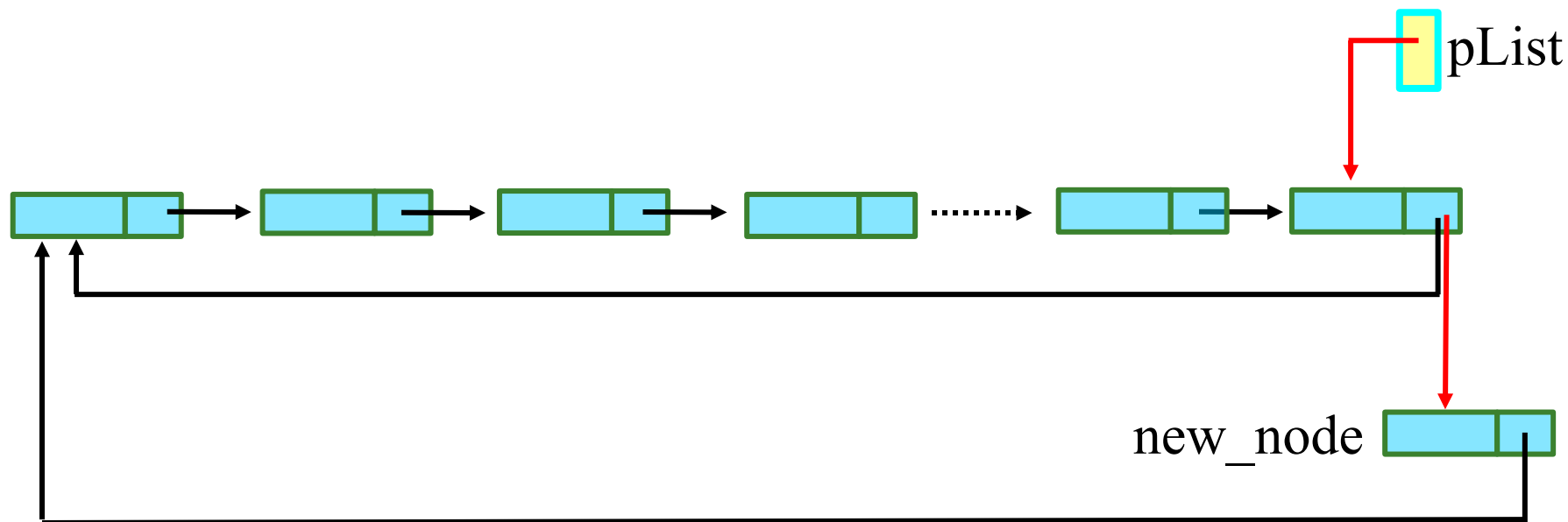


Cho `new_node -> next` trở tới node đầu:

**`new_node->next = pList->next;`**

## 4.3.1 CLL - Bổ sung vào cuối ds

□ **insertEnd:**  $pList \neq \text{NULL}$



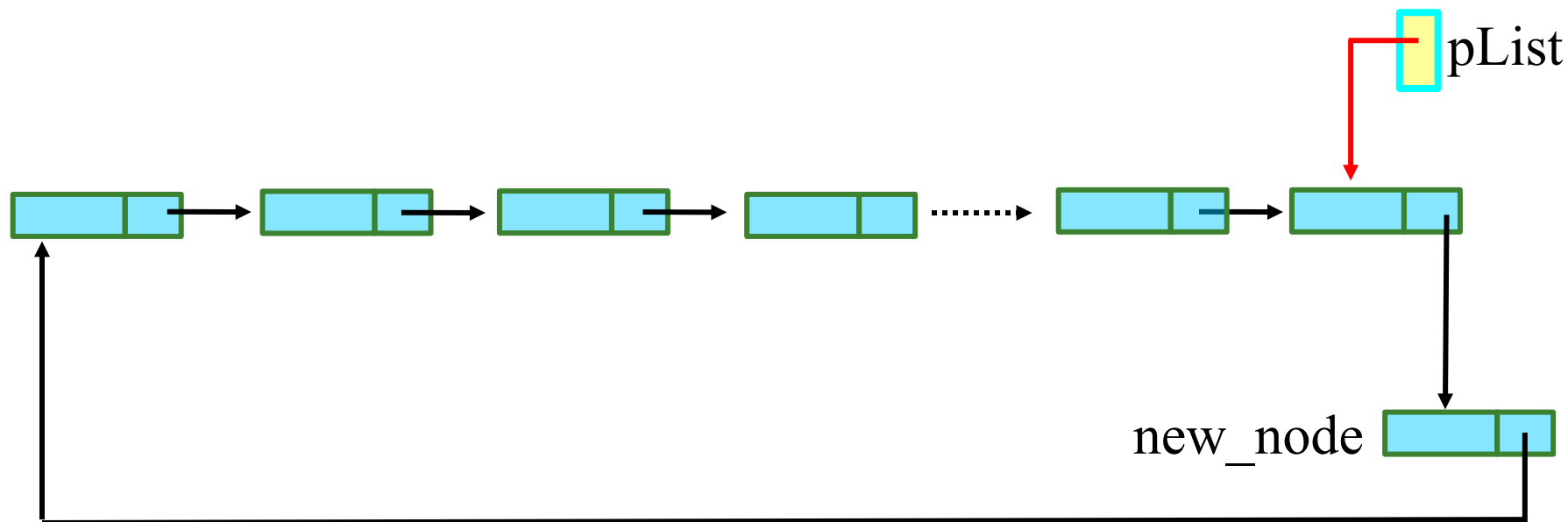
Cho  $pList \rightarrow \text{next}$  trở tới **new\_node**:

**$pList \rightarrow \text{next} = \text{new\_node};$**



## 4.3.1 CLL - Bổ sung vào cuối ds

□ **insertEnd:**  $pList \neq \text{NULL}$



Cho pList dịch chuyển tới new\_node:

**pList = new\_node;**

## 4.3.1 CLL – Bổ sung vào cuối ds

```
1. void insertEnd(Node* &pList, int x)
2. { if (pList == NULL)
3.     addFirst(pList, x);
4.     else
5.     { Node *new_node;
6.       new_node = createNode(x);

7.       new_node->next = pList->next;
8.       pList->next = new_node;
9.       pList = new_node;
10.    }
11. }
```

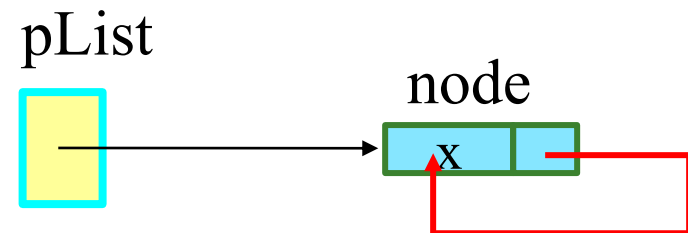
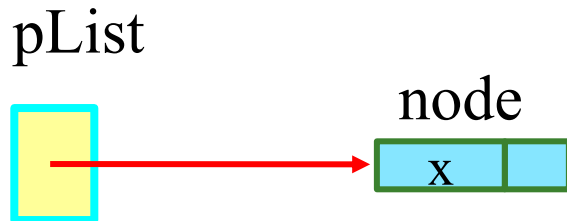
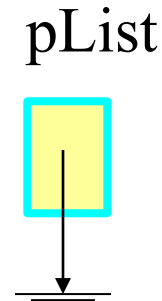
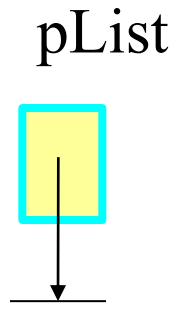
## 4.3.1 CLL - BỔ sung vào sau nút p

### □ **insertAfter**: BỔ sung vào sau node p

- Nếu danh sách trống: Gọi hàm **addFirst**
- Nếu  $p == pList$ : Gọi hàm **insertLast** để chèn vào cuối danh sách
- Nếu danh sách không trống và  $p \neq pList$ :
  - + Tạo node mới
  - + Chèn vào sau node p.

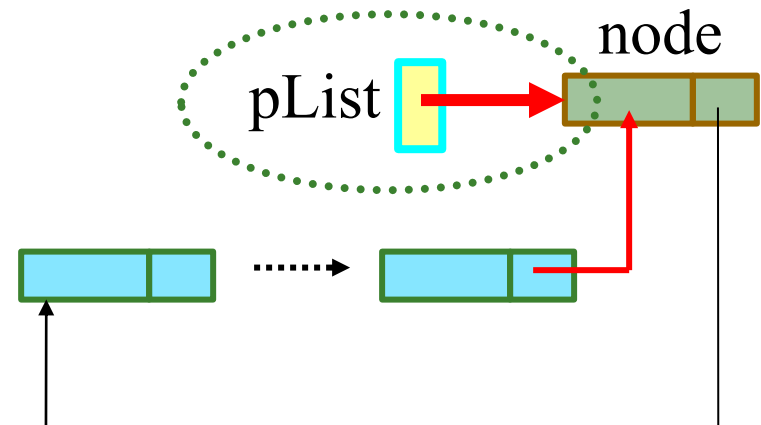
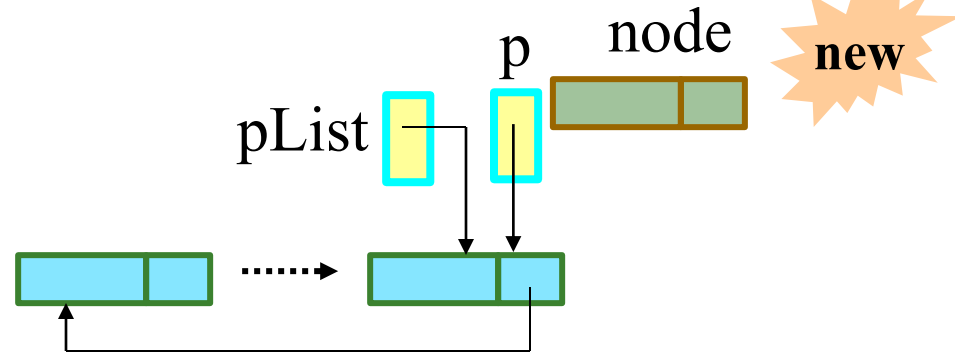
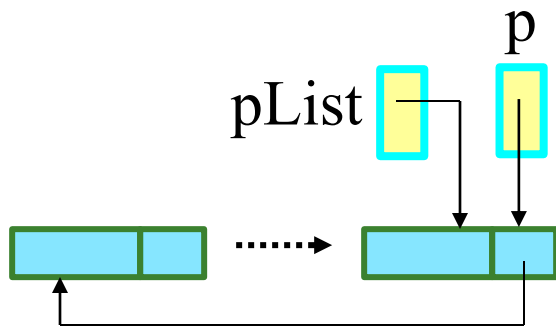
## 4.3.1 CLL - Bổ sung vào sau node p

□ **insertAfter:** pList == NULL



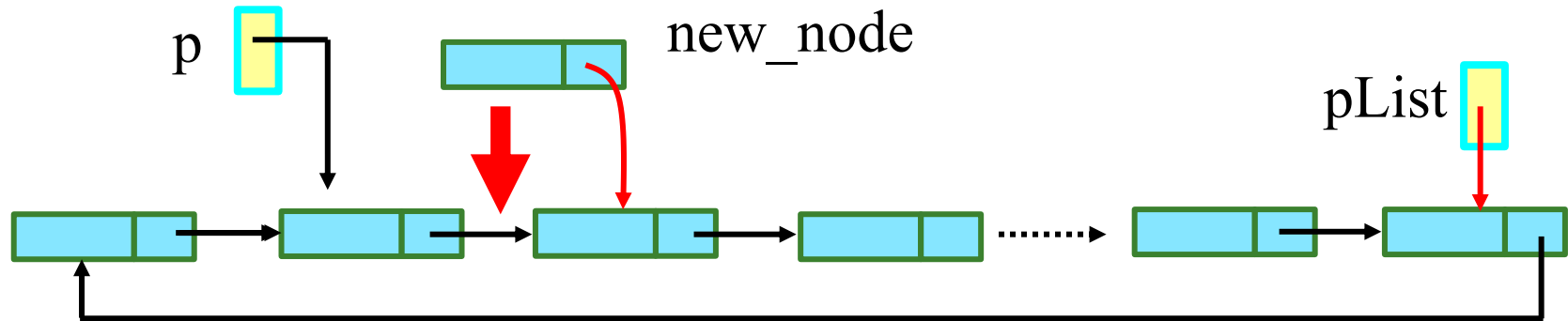
## 4.3.1 CLL - Bổ sung vào sau node p

□ **insertAfter:**  $p == pList$



## 4.3.1 CLL - Bổ sung vào sau nút p

□ **insertAfter:** `pList != NULL && p != pList`

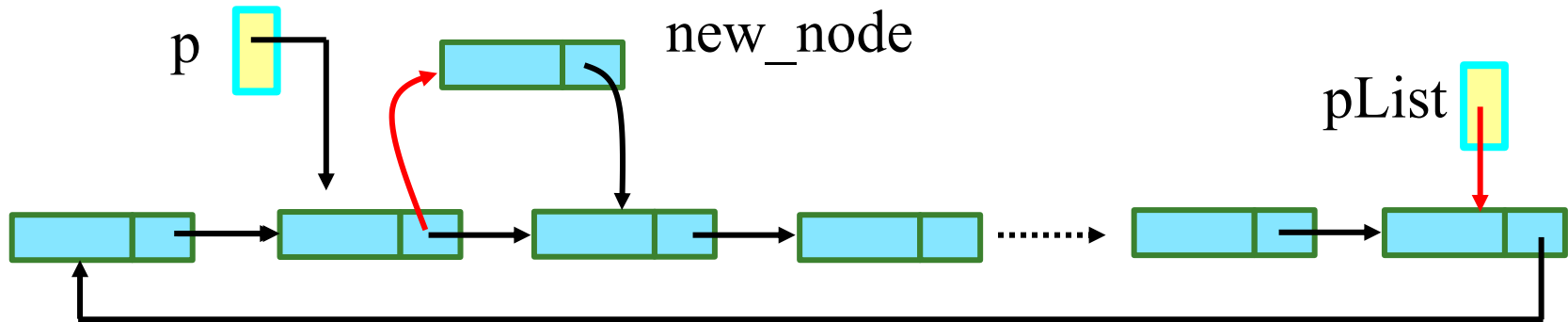


Cho `new_node->next` trở tới nút sau p:

**`new_node->next = p->next;`**

## 4.3.1 CLL - Bổ sung vào sau nút p

□ **insertAfter:** `pList != NULL && p != pList`



Cho `p->next` trở tới node mới (`new_node`):

**`p->next = new_node->next;`**

## 4.3.1 CLL – BỔ sung vào sau nút p

```
1. void insertAfter(Node* &pList, Node *p, int x)
2. { if (pList == NULL)
3.     addFirst(pList, x);
4.     else if (p == pList)
5.         insertEnd(pList, x);
6.     else
7.     { Node* new_node;
8.       new_node = createNode(x);
9.
10.      new_node->next = p->next;
11.      p->next = new_node;
12.    }
13. }
```



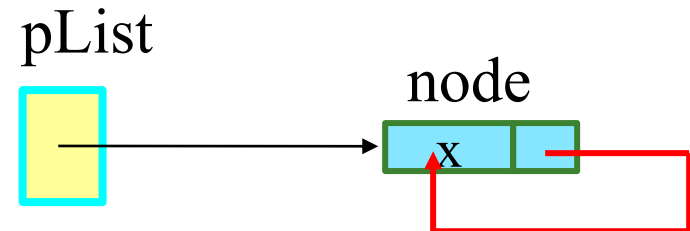
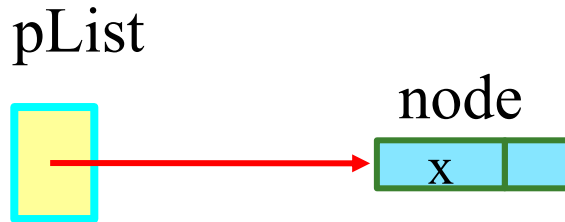
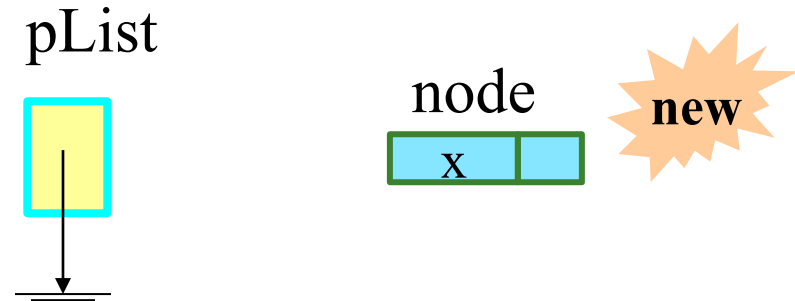
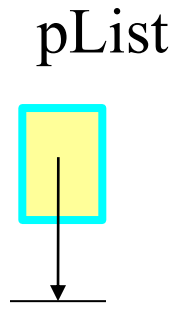
## 4.3.1 CLL - BỔ sung vào trước nút p

### □ **insertBefore**: BỔ sung vào trước nút p

- Nếu danh sách trống: Gọi hàm **addFirst**
- Nếu danh sách không trống:
  - + Tạo node mới
  - + Chèn vào trước node p.

## 4.3.1 CLL - Bổ sung vào trước nút p

□ **insertBefore:** pList == NULL



## 4.3.1 CLL - Bổ sung vào trước nút p

□ **insertBefore**: thêm node có nội dung x trước node p

```
1. void insertBefore(Node* &pList, Node *p, int x)
2. {   if (pHead == NULL)
3.         cout<<"Danh sach trong!";
4.     else
5.     {   Node* new_node;
6.         new_node = creatNode(x);
7.         Node *q = pList->next;
8.         while(q->next != p)    //q trở node trước p
9.             q = q->next;
10.        new_node->next = p;
11.        q->next = new_node;
12.    }
13. }
```

## 4.3.1 CLL - Đếm số node

□ **countNode**: Đếm số node trong danh sách

```
1. int countNode (Node* pList)
2. { if (pHead == NULL)    return 0;
3.   else
4.     {      Node* p = pList->next;
5.         int dem = 0;
6.         do
7.             {      dem++;
8.                 p = p->next;
9.             } while (p != pList->next);
10.    return dem;
11.  }
12. }
```

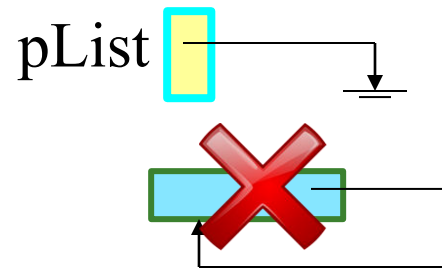
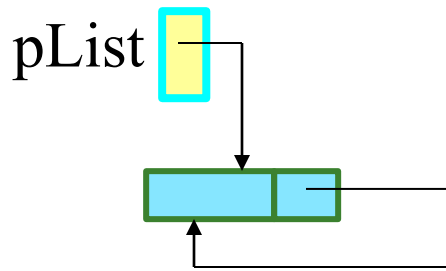
## 4.3.1 CLL - Xóa bỏ node đầu ds

### □ **deleteFirst**: Xóa bỏ node đầu danh sách

- Nếu danh sách trống: Không xóa được
- Nếu danh sách có 1 node: Xóa pList
- Nếu danh sách có nhiều hơn 1 node:
  - + Cho pList->next trở về node thứ 2.
  - + Xóa node đầu.

## 4.3.1 CLL - Loại bỏ node đầu

□ Danh sách có 1 nút:       $pList \rightarrow next = pList$

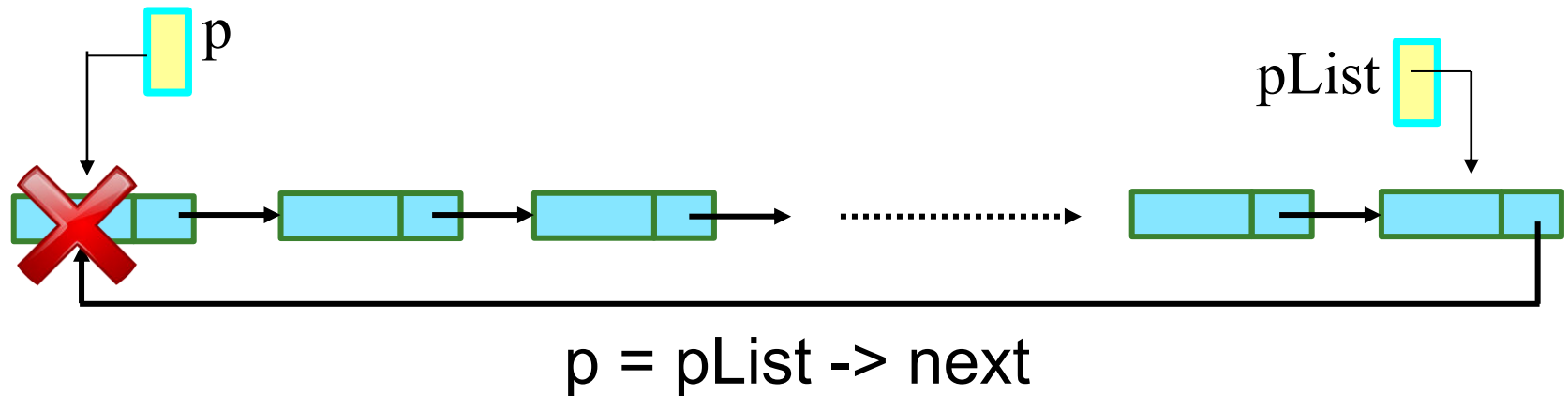


**delete pList;**

**pList = NULL;**

## 4.3.1 CLL - Loại bỏ node đầu

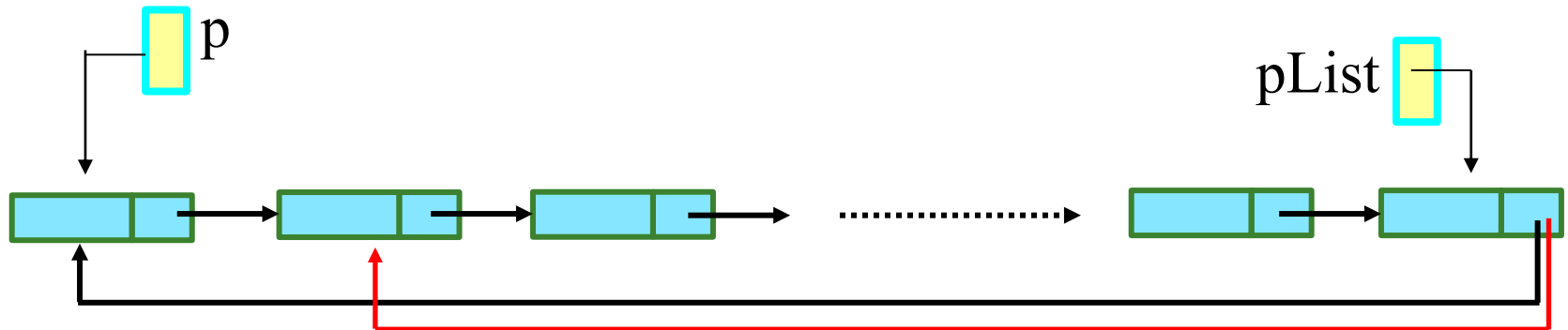
□ Danh sách có nhiều hơn 1 nút:  $pList \rightarrow next \neq pList$



**Loại bỏ nút đầu**

## 4.3.1 CLL - Loại bỏ node đầu

□ Danh sách có nhiều hơn 1 nút: **pList -> next != pList**



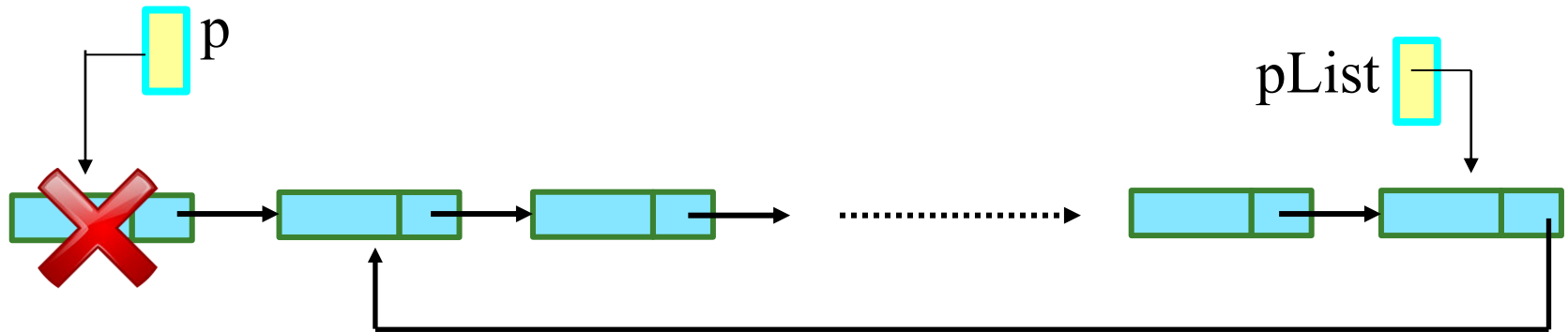
Cho **pList->next** trở tới node kế sau node **p**:

**pList->next = p->next;**



## 4.3.1 CLL - Loại bỏ node đầu

□ Danh sách có nhiều hơn 1 nút: **pList -> next != pList**



Xóa node p:

**delete p;      p = NULL;**

## 4.3.1 CLL - Loại bỏ node đầu

```
1. void deleteFirst (Node* &pList)
2. { if (pList == NULL)
3.     cout<<"Danh sach trong!";
4.     else if (pList->next == pList) //co 1 nut
5.     { delete pList;
6.       pList = NULL;
7.     }
8.     else //co nhieu hon 1 nut
9.     { Node *p = pList->next;
10.      pList->next = p->next;
11.      delete p; p = NULL;
12.    }
13. }
```

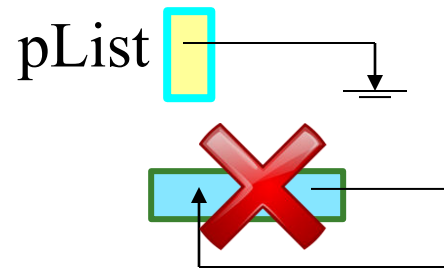
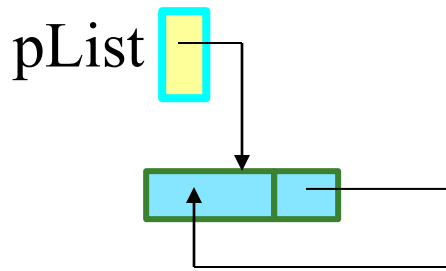
## 4.3.1 CLL - Xóa bỏ node cuối ds

### □ deleteLast: Xóa bỏ node cuối danh sách

- Nếu danh sách trống: Không xóa được
- Nếu danh sách có 1 node: Xóa pList
- Nếu danh sách có nhiều hơn 1 node:
  - + Cho thành phần next của node kề trước pList trở về đầu danh sách.
  - + Dịch chuyển pList trở về node kề trước.
  - + Xóa node cuối.

## 4.3.1 CLL - Loại bỏ node cuối

□ Danh sách có 1 nút:  $pList \rightarrow next = pList$

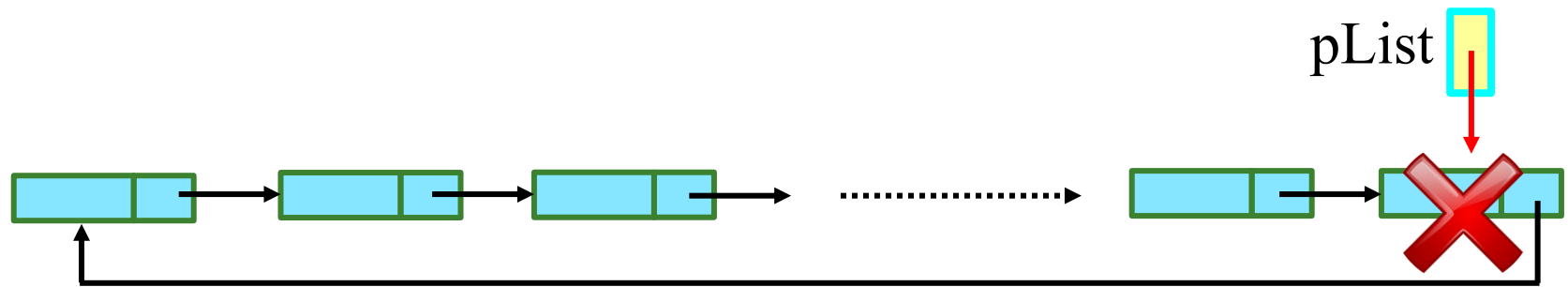


**delete pList;**

**pList = NULL;**

## 4.3.1 CLL - Loại bỏ node cuối

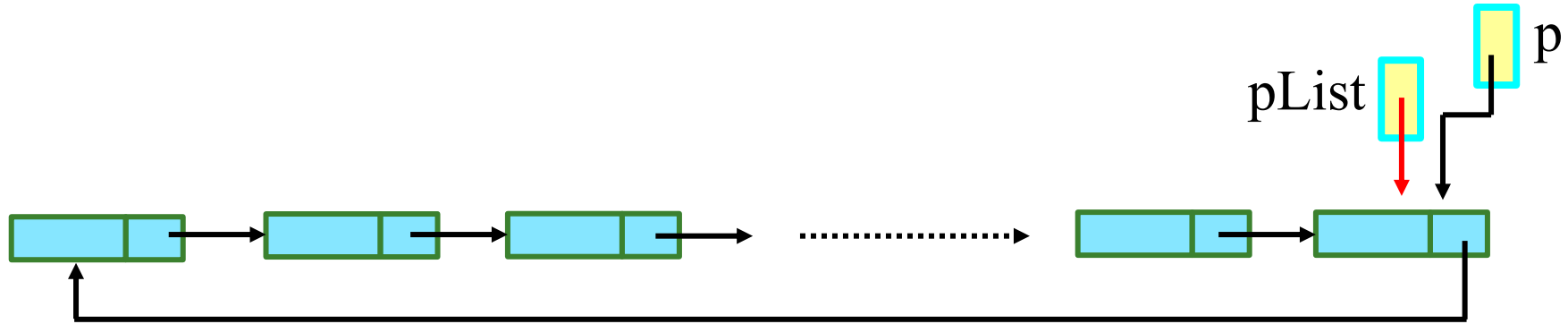
□ Danh sách có nhiều hơn 1 nút: **pList -> next != pList**



**Loại bỏ nút cuối**

## 4.3.1 CLL - Loại bỏ node cuối

□ Danh sách có nhiều hơn 1 nút: **pList -> next != pList**

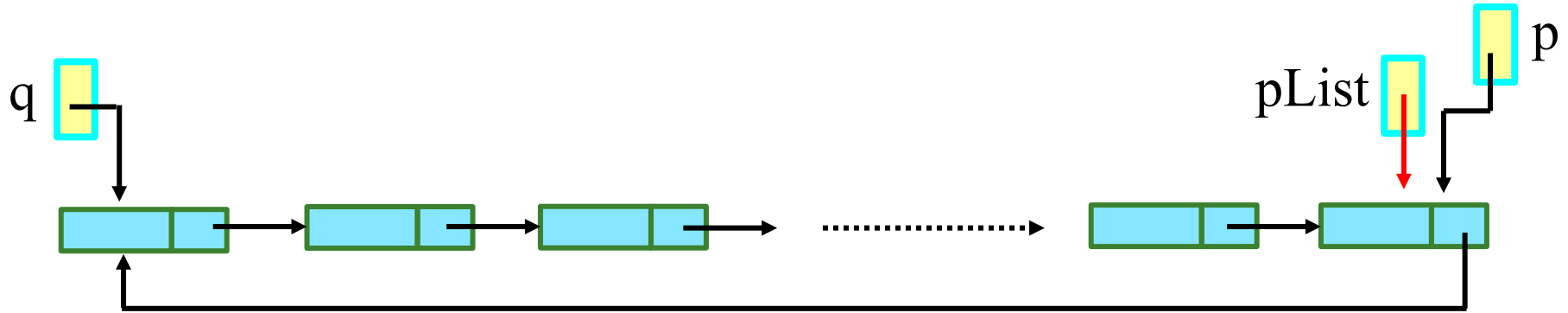


Cho p trở vào cuối danh sách cùng pList:

**p = pList;**

## 4.3.1 CLL - Loại bỏ node cuối

□ Danh sách có nhiều hơn 1 nút: **pList -> next != pList**

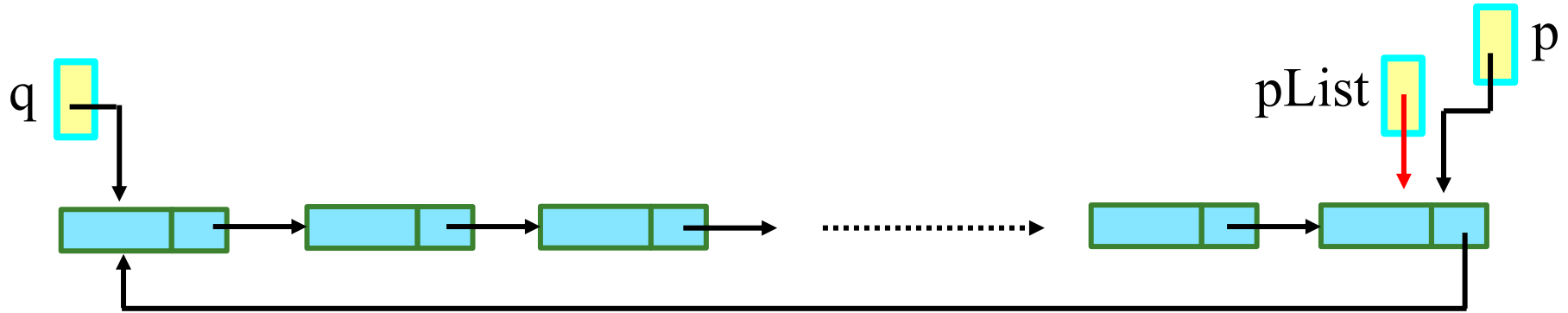


Khai báo con trỏ q, q trỏ vào đầu danh sách:

**Node \*q = pList ->next;**

## 4.3.1 CLL - Loại bỏ node cuối

□ Danh sách có nhiều hơn 1 nút: **pList -> next != pList**



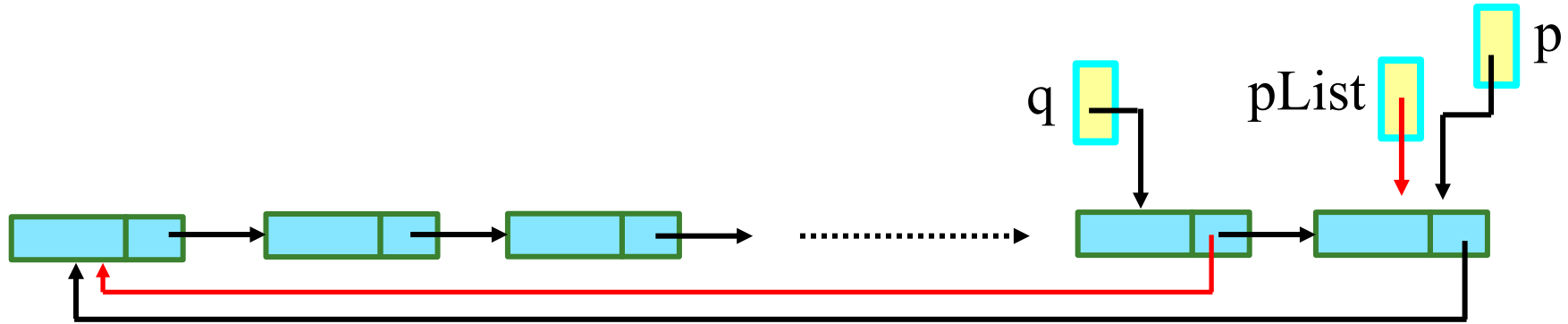
Dịch chuyển q: về trở node trước node p:

**while (q->next != p) q = q->next;**



## 4.3.1 CLL - Loại bỏ node cuối

□ Danh sách có nhiều hơn 1 nút: **pList -> next != pList**

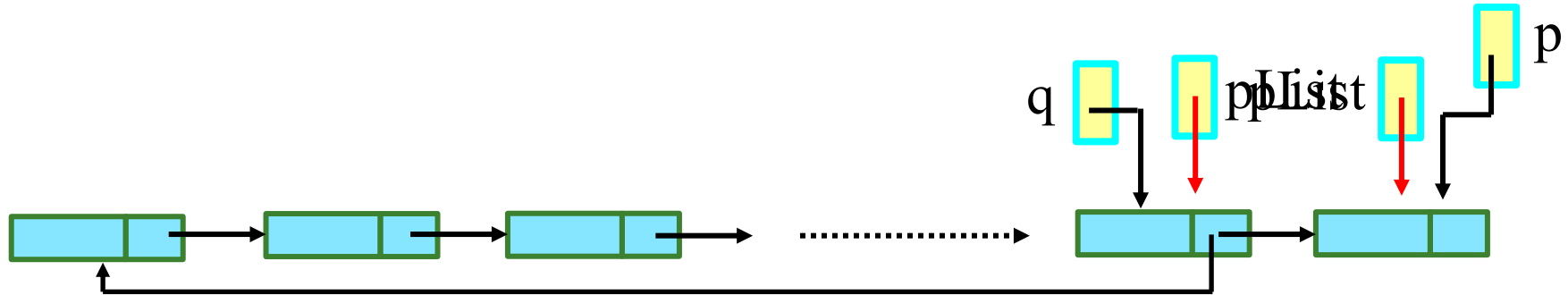


Cho  $q \rightarrow \text{next}$  trở node đầu danh sách:

**$q \rightarrow \text{next} = \text{pList} \rightarrow \text{next};$**

## 4.3.1 CLL - Loại bỏ node cuối

□ Danh sách có nhiều hơn 1 nút: **pList -> next != pList**

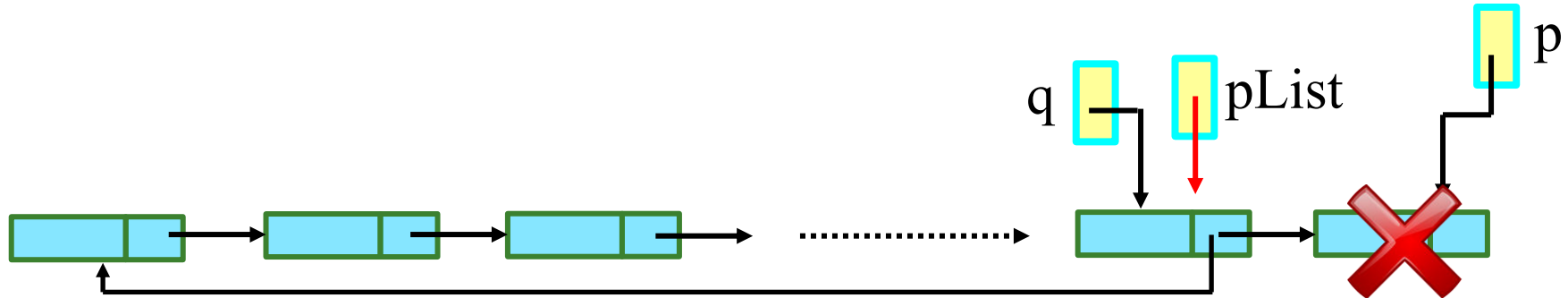


Dịch chuyển pList trở node cùng q:

**pList = q;**

## 4.3.1 CLL - Loại bỏ node cuối

□ Danh sách có nhiều hơn 1 nút: **pList -> next != pList**



Xóa node p:

**delete p;      p = NULL;**

## 4.3.1 CLL - Loại bỏ node cuối

```
1. void deleteLast (Node* &pList)
2. {
3.     if (pList == NULL)           //ds trống
4.         return;
5.     else if (pList->next == pList) //co 1 nut
6.     {
7.         delete pList;
8.         pList = NULL;
9.     }
```

## 4.3.1 CLL – Loại bỏ node cuối

9.       **else**

10.     {       Node \*p = pList;       //p trở nút cuối

11.       Node \*q = pList->next;

12.       **while** (q->next != p)

13.               q = q->next;       //q trở trước p

14.

15.       q->next = pList->next;

16.       pList = q;       //Dịch chuyển pList

17.       **delete** p;

18.       p = **NULL**;

19.     }

20. }

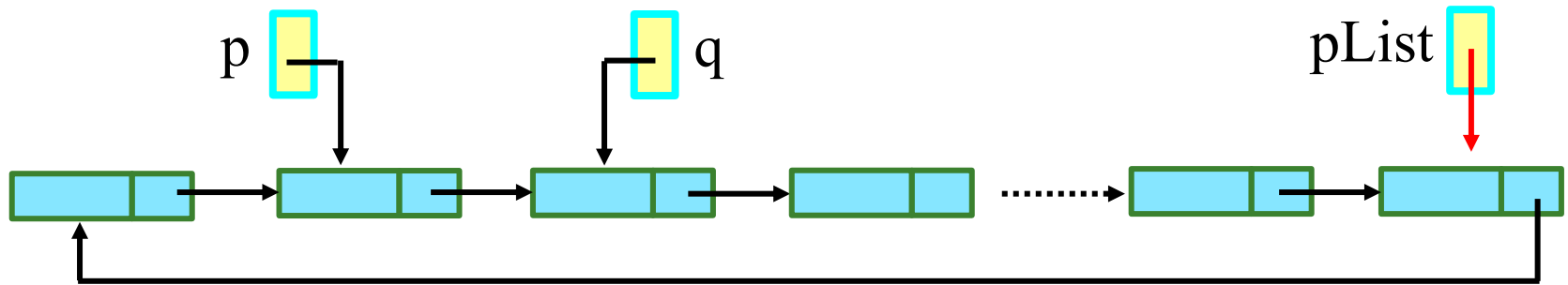
## 4.3.1 CLL - Xóa bỏ node kề sau p

### □ **deleteAfter**: Xóa bỏ node kề sau p

- Nếu danh sách trống: Không xóa được
- Nếu `pList->next == pList`: Không có node sau p.
- Nếu không thuộc 1 trong các trường hợp trên:
  - + Khai báo con trỏ q, trỏ vào node kề sau p.
  - + Cho thành phần next của p trỏ vào node sau q.
  - + Xóa node q.

## 4.3.1 CLL - Xóa bỏ node kế sau p

□ Danh sách có nhiều hơn 1 nút: **pList -> next != pList**

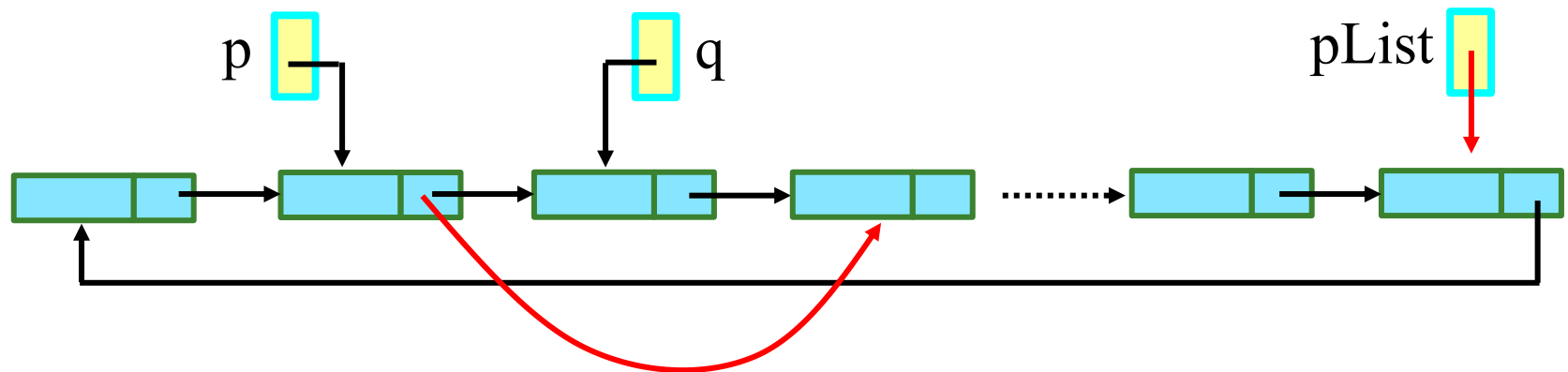


Khai báo con trỏ q trỏ node kế sau p:

**q = p->next;**

## 4.3.1 CLL - Xóa bỏ node kế sau p

□ Danh sách có nhiều hơn 1 nút: **pList -> next != pList**



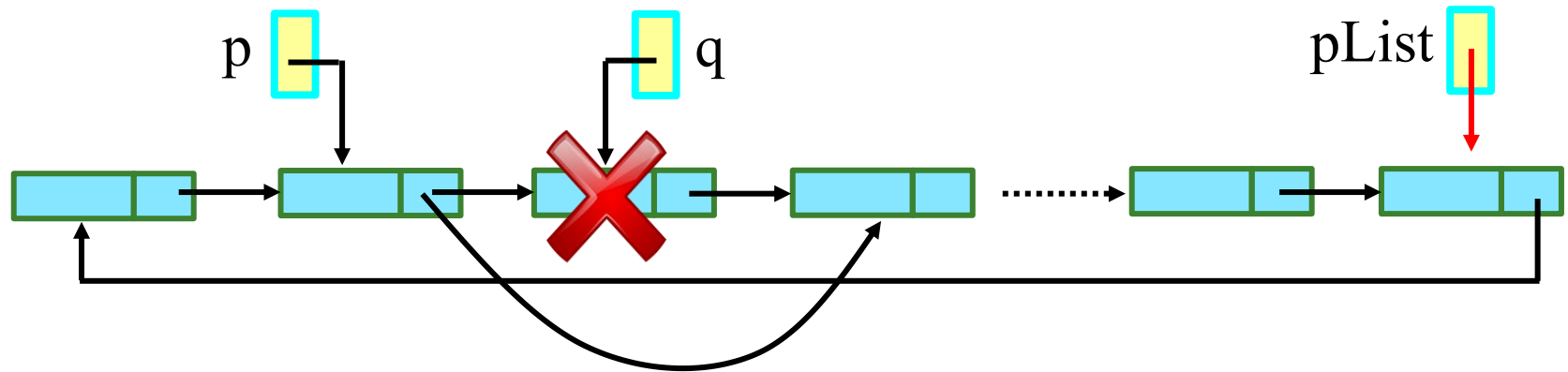
Cho thành phần next của p trở node kế sau q:

**p->next = q->next;**



## 4.3.1 CLL - Xóa bỏ node kế sau p

□ Danh sách có nhiều hơn 1 nút:  $pList \rightarrow next \neq pList$



Xóa node q:

**delete q;      q = NULL;**

## 4.3.1 CLL - Xóa bỏ nút kề sau p

□ **deleteAfter**: Xóa node kề sau node p trong danh sách

```
1. void DeleteAfter(Node* &pList, Node* &p)
2. {   if (pList == NULL)
3.         cout<<"Danh sach trong!";
4.     else if (pList->next == pList)
5.     {   cout<<"Khong co nut sau p!";
6.         return;
7.     }
8.     else
9.     {   Node *q = pList->next;
10.        p->next = q->next;
11.        delete q; q = NULL;
12.    }
```

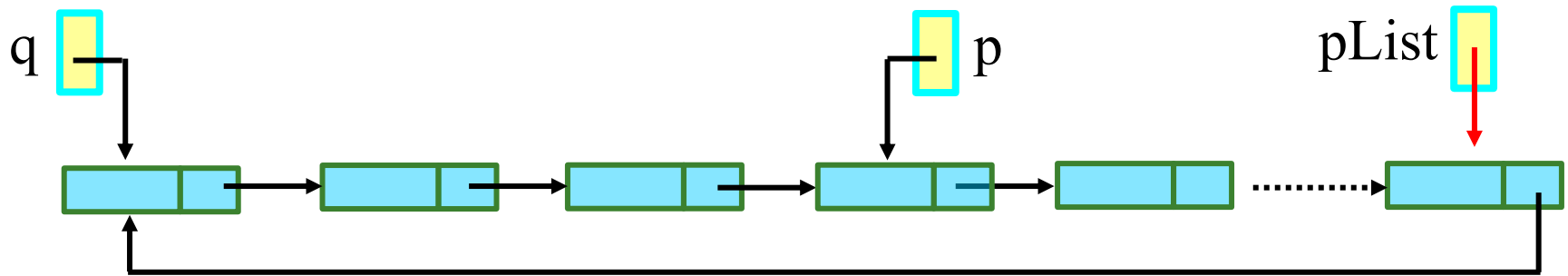
## 4.3.1 CLL - Xóa bỏ node p

□ **deleteNode**: Xóa bỏ node p trong danh sách

- Nếu danh sách trống: Không xóa được
- Nếu  $p == pList \rightarrow next$ : Gọi hàm **deleteFirst** xóa node đầu
- Nếu  $p = pList$ : Gọi hàm **deleteLast** xóa node cuối
- Nếu không thuộc 1 trong các trường hợp trên:
  - + Khai báo con trỏ q, trỏ vào node kề trước p.
  - + Cho thành phần next của q trỏ vào node sau q.
  - + Xóa node p.

## 4.3.1 CLL - Xóa bỏ node p

□ Danh sách có nhiều hơn 1 nút: **pList -> next != pList**

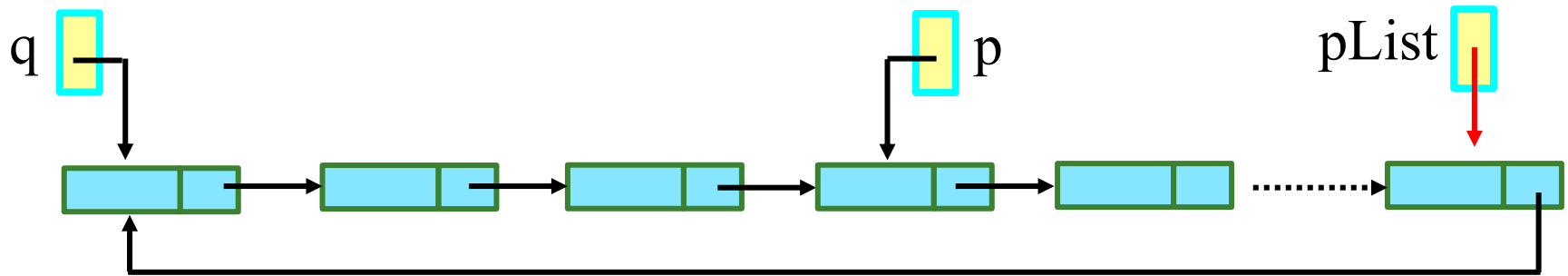


Khai báo con trỏ q trỏ node đầu danh sách:

**Node \*q = pList->next;**

## 4.3.1 CLL - Xóa bỏ node p

□ Danh sách có nhiều hơn 1 nút: **pList -> next != pList**

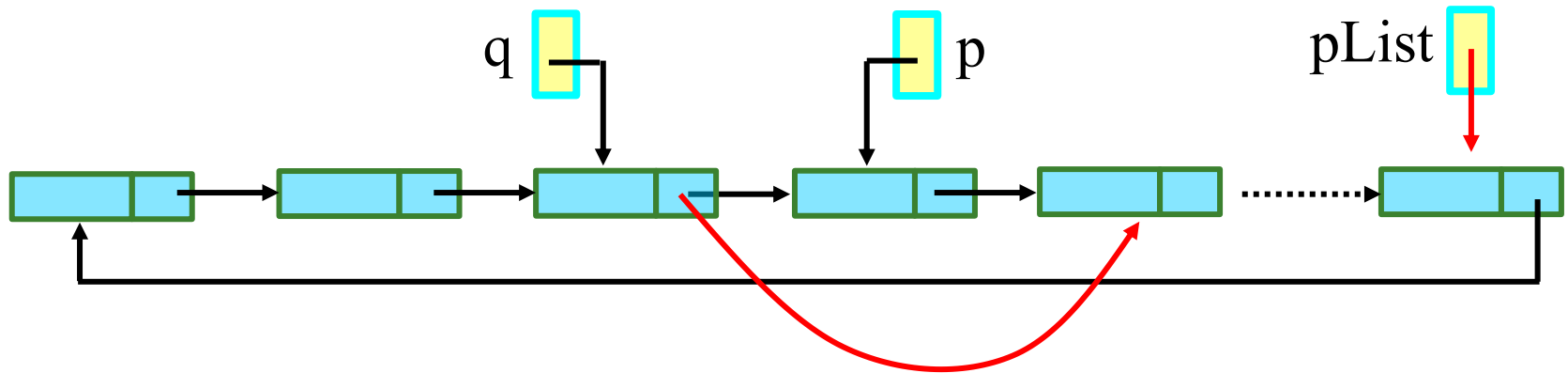


Dịch chuyển q về vị trí trở node trước p:

**while (q->next != p)    q = q->next;**

## 4.3.1 CLL - Xóa bỏ node p

□ Danh sách có nhiều hơn 1 nút: **pList -> next != pList**

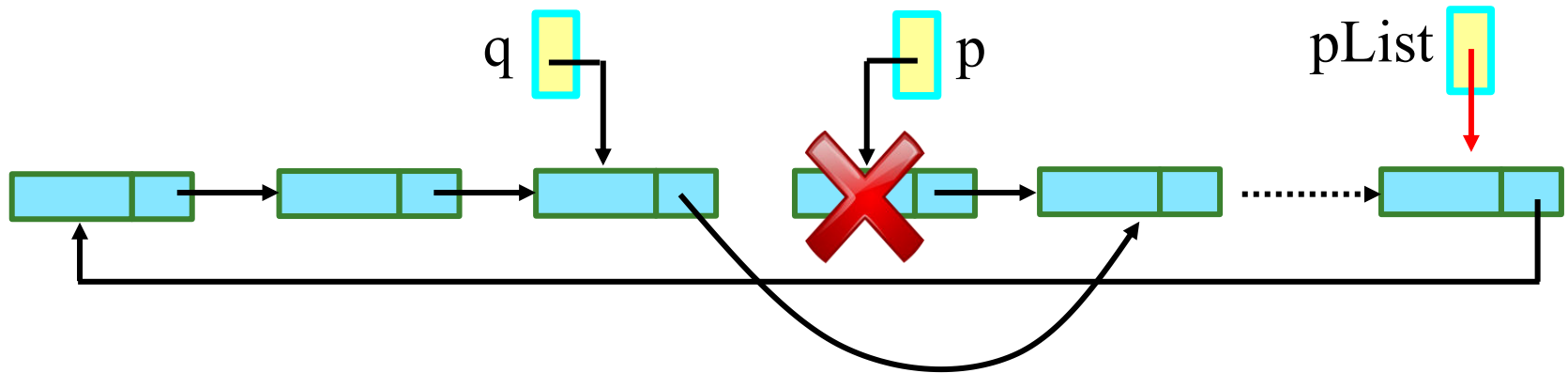


Cho thành phần next của q trở node kế sau p:

**q->next = p->next;**

## 4.3.1 CLL - Xóa bỏ node p

□ Danh sách có nhiều hơn 1 nút: **pList -> next != pList**



Xóa node p:

**delete p;      p = NULL;**

## 4.3.1 CLL - Xóa bỏ nút p

```
1. void deleteNode (Node* &pList, Node* p)
2. {   if (p == NULL)
3.     {       cout<<"Danh sach trong!";
4.           return;
5.     }
6.     else if (p == pList->next)
7.     {       deleteFirst(pList);
8.           return;
9.     }
10.    else if (p == pList)
11.    {       deleteLast(pList);
12.          return;
13.    }
```



## 4.3.1 CLL - Xóa bỏ node p

```
14.  else
15.  {
16.      Node *q = pList->next;
17.      while (q->next != p)
18.          q = q->next;
19.
20.      q->next = p->next;
21.      delete p;
22.      p = NULL;
23.  }
24. }
```

## 4.3.1 CLL - Xoá toàn bộ danh sách

□ **deleteAll**: Xoá toàn bộ danh sách

```
1. void deleteAll (Node*      &pList)
2. {
3.     Node *p;
4.     while (pList->next != NULL)
5.     {   p = pList->next;
6.         pList->next = p->next;
7.         delete p; p = NULL;
8.     }
9. }
```

## 4.3.1 CLL - Tìm kiếm

□ **searchValue**: Tìm kiếm thành phần data bằng x

□ **searchValue**:

- Xuất phát từ đầu danh sách
- Nếu tìm thấy trả về địa chỉ nút đó và dừng
- Ngược lại qua phần tử tiếp theo
- Điều kiện dừng khi quay lại phần tử đầu tiên
- Không tìm thấy trả về NULL

## 4.3.1 CLL – Tìm kiếm

□ **searchValue**: Tìm kiếm thành phần data bằng x

1. **Node\*** searchValue (Node\* &pList, int x)

2. {

3.     **if** (pList ==NULL) **return** NULL;

4.     Node \*p = pList->next;

5.     **while** ((p->next != pList->next) &&

6.             (p->data != x))

7.         p = p->next;

8.     **if** ( p->data == x)     **return** p;

9.     **return** NULL;

10. }

## 4.3.1 CLL - Sắp xếp

□ **sortLisst**: Sắp xếp theo thứ tự tăng dần của data

```
1. Node* sortList (Node* &pList)
2. {
3.     Node *p, *q;
4.     for (p = pList->next; p != pList; p = p->next)
5.         for (q = p->next; q != pList->next; q = q->next)
6.             if(p->data > q->data)
7.                 swap(p->data, q->data);
8. }
```

# Câu hỏi củng cố bài

1. Dấu hiệu nào dưới đây cho biết danh sách nối vòng có nút cuối được trỏ bởi p là có 1 nút:
- A.  $p == \text{NULL}$
  - B.  $p \rightarrow \text{next} == p$
  - C.  $p \rightarrow \text{info} == \text{NULL}$
  - D.  $p \rightarrow \text{next} == \text{NULL}$



Multiple Choice

# Câu hỏi củng cố bài

2. Khi ta thêm một node  $q$  vào đầu danh sách liên vòng có nút cuối trỏ bởi  $p$  thì:
- A.  $q \rightarrow \text{next} = p$  ;  $p = q$ ;
  - B.  $q \rightarrow \text{info} = p$ ;
  - C.  $q \rightarrow \text{next} = p$ ;
  - D.  $q \rightarrow \text{next} = p \rightarrow \text{next}$  ;  $p \rightarrow \text{next} = q$ ;



Multiple Choice

# Câu hỏi củng cố bài

3. Dấu hiệu nào dưới đây cho biết node q của một danh sách liên kết vòng có nút cuối trở bởi p không phải là node đầu:
- A.  $q \rightarrow \text{next} == \text{NULL}$
  - B.  $q == p \rightarrow \text{next}$
  - C.  $q != p \rightarrow \text{next}$
  - D.  $q \rightarrow \text{next} != \text{NULL}$



Multiple Choice



# Câu hỏi củng cố bài

4. Dấu hiệu nào dưới đây cho biết node q của một danh sách liên kết vòng có nút cuối trở bởi p không phải là node đầu:
- A.  $q \rightarrow \text{next} == \text{NULL}$
  - B.  $q == p \rightarrow \text{next}$
  - C.  $q != p \rightarrow \text{next}$
  - D.  $q \rightarrow \text{next} != \text{NULL}$



Multiple Choice

# Câu hỏi củng cố bài

5. Khi loại bỏ nút p khỏi danh sách liên kết vòng thì:
- A. Ta phải dịch chuyển về node trước node p để thực hiện
  - B. Ta phải dịch chuyển về node đầu để thực hiện
  - C. Ta phải dịch chuyển đến node cuối cùng để thực hiện
  - D. Ta phải dịch chuyển về node sau node p



Multiple Choice

# Câu hỏi củng cố bài

6. Cho 2 con trỏ p và q, p trỏ vào một nút bất kỳ trong danh sách (không phải nút đầu). Lệnh nào dưới đây là đúng để con trỏ q trỏ vào nút trước p?

A. `q = pHead;`  
`while ( p == q -> next)`  
`q = q -> next;`

B. `q = pHead;`  
`while ( p != q -> next)`  
`q = q -> next;`

C. `q = NULL;`

D. `q != NULL;`



Multiple Choice

# Câu hỏi củng cố bài

- Cho 2 con trỏ p và q, p trỏ vào một nút bất kỳ trong danh sách (không phải nút đầu). Lệnh nào dưới đây là đúng để con trỏ q trỏ vào nút trước p?  
  
A.  $q = \text{NULL};$   
  
B.  $q \neq \text{NULL};$



Multiple Choice

