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

Doubly Linked List

4.3.2 Doubly Link List - Mô tả

- Danh sách liên kết đôi là một dạng biến thể của danh sách liên kết đơn.

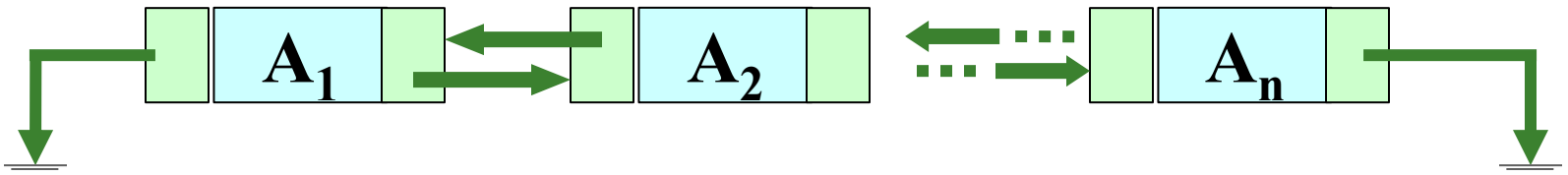
- Ưu điểm so với danh sách liên kết đơn:

- Có thể duyệt danh sách liên kết đôi theo cả hai cách.
- Dễ dàng duyệt theo hướng từ trái sang hoặc từ phải sang.

- Ví dụ: Nút **Back** và nút **Forward** của trình duyệt.

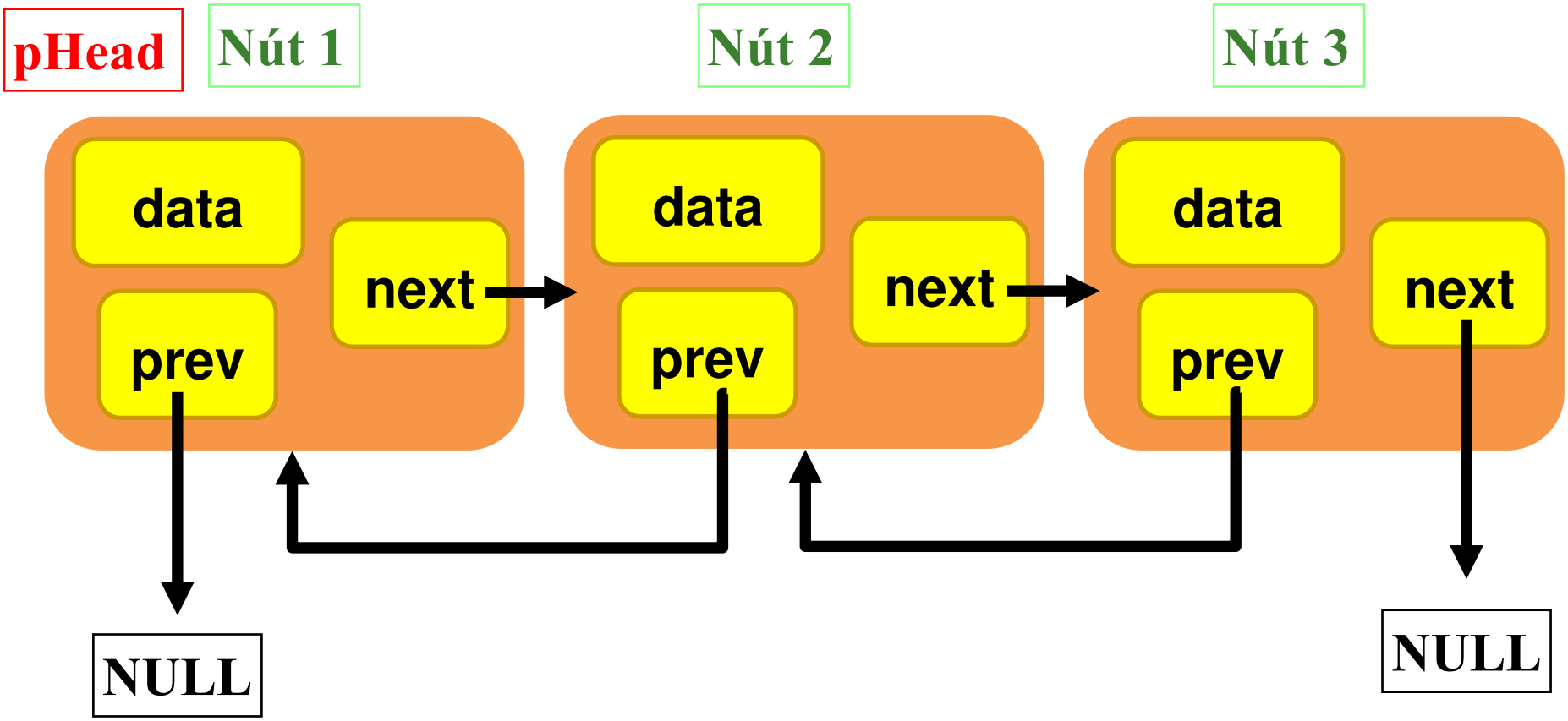
4.3.2 Doubly Link List - Mô tả

- Cho phép di chuyển 2 chiều đến nút trước và sau.
  - Liên kết nút trước là: **prev**
  - Liên kết nút sau là: **next**
- Trường **prev** của nút đầu bằng NULL
- Trường **next** của nút cuối bằng NULL



4.3.2 Doubly Linked List - Mô tả

Mô tả Danh sách liên kết kép



4.3.2 Doubly Linked List – Cài đặt

- Khai báo

```
struct Nut
{
        int  data;
        struct Nut *  prev;
        struct Nut *  next;
};
typedef  struct  Nut  Node;

Node  *pHead;
```

→ trỏ đến nút trước

→ trỏ đến nút sau

→ pHead quản lý ds kép

4.3.2 DLL - Các thao tác

- Các thao tác cơ bản
  - listInit
  - creatNode

Phần minh hoạ sẽ dùng

DataType là int

- In danh sách: printList, printReverse
- Bổ sung 1 phần tử mới vào danh sách
  - insertBegin
  - insertEnd
  - insertAfter
  - insertBefore
- Loại bỏ 1 phần tử khỏi danh sách
  - deleteFirst
  - deleteLast
  - deleteNodeP
  - deletePosK
  - deleteDataX
- Các thao tác khác
  - searchValue
  - sortList

4.3.2 DLL – Các thao tác

- **listInit**: Khởi tạo danh sách
- // list initailization

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

4.3.2 DLL – Duyệt danh sách

- **printList -** In danh sách
  - Duyệt từ đầu danh sách
  - Đến khi nào hết danh sách thì dừng

4.3.2 DLL – Các thao tác cơ bản

- **printList:** In danh sách

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

4.3.2 DLL – Duyệt danh sách

- **printReverse -** In danh sách từ cuối về đầu
  - Duyệt ngược từ cuối danh sách.
  - Khi nào đến đầu danh sách thì dừng

4.3.2 DLL – Duyệt danh sách

- **printReverse -** In danh sách từ cuối về đầu

```
1. void printReverse(Node *pHead)
2. {  if (pHead == NULL) return;
3.    Node *p = pHead;
4.    while (p->next != NULL)
5.          p = p->next;
6.    do
7.    {     cout<<p->data;  cout<<"\t";
8.          p = p->prev;
9.    }while (p != NULL);
10.}
```

4.3.2 DLL – Các thao tác

- **creatNode:** Tạo nốt mới có nội dung x

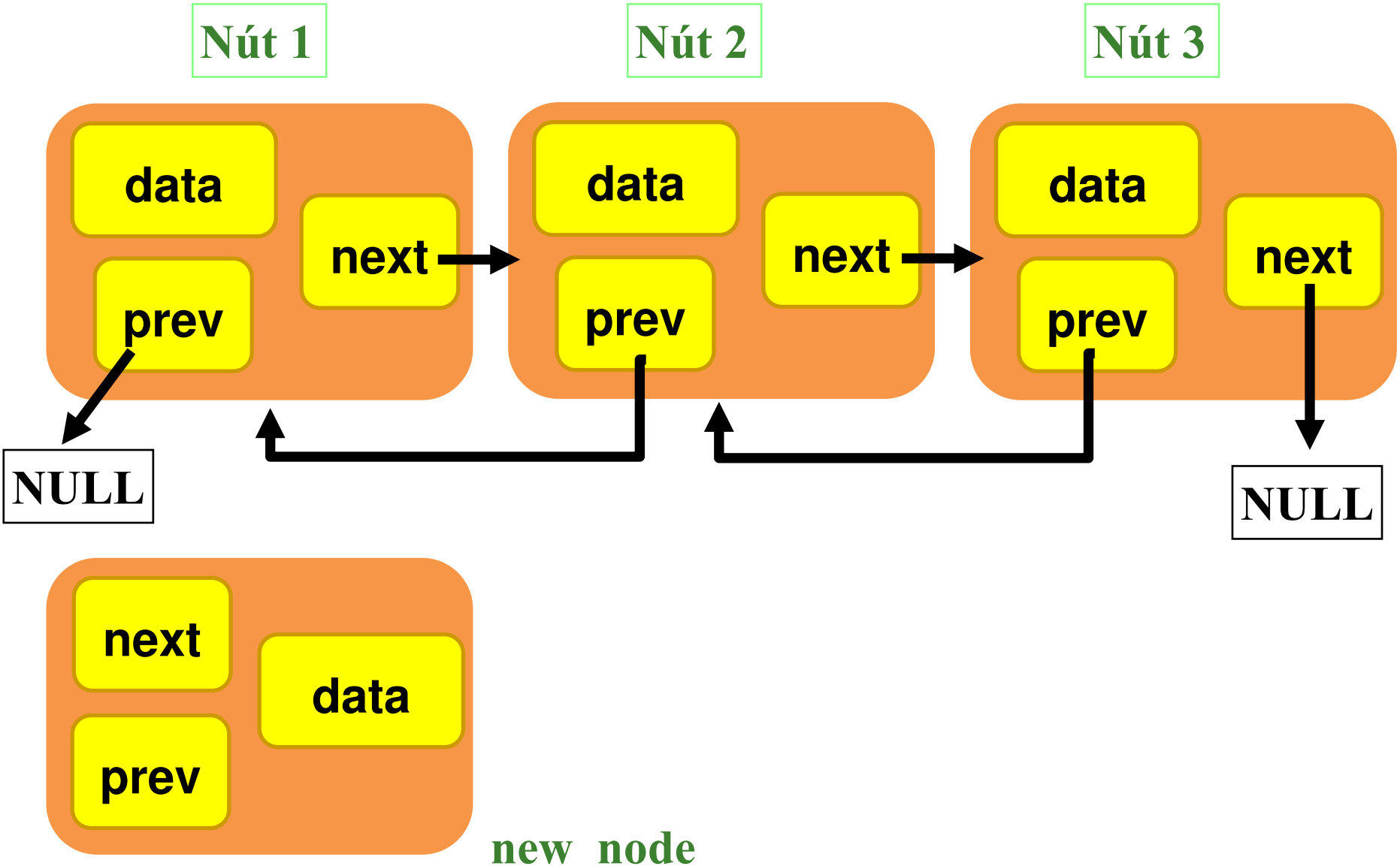
```
1. Node* creatNode(int x)
2. {
3.     Node* new_node;
4.     new_node = new Node;
5.     new_node->data = x;
6.     new_node->next = NULL;
7.     new_node->prev = NULL;
8.     return new_node;
9. }
```

4.3.2 DLL – Các thao tác

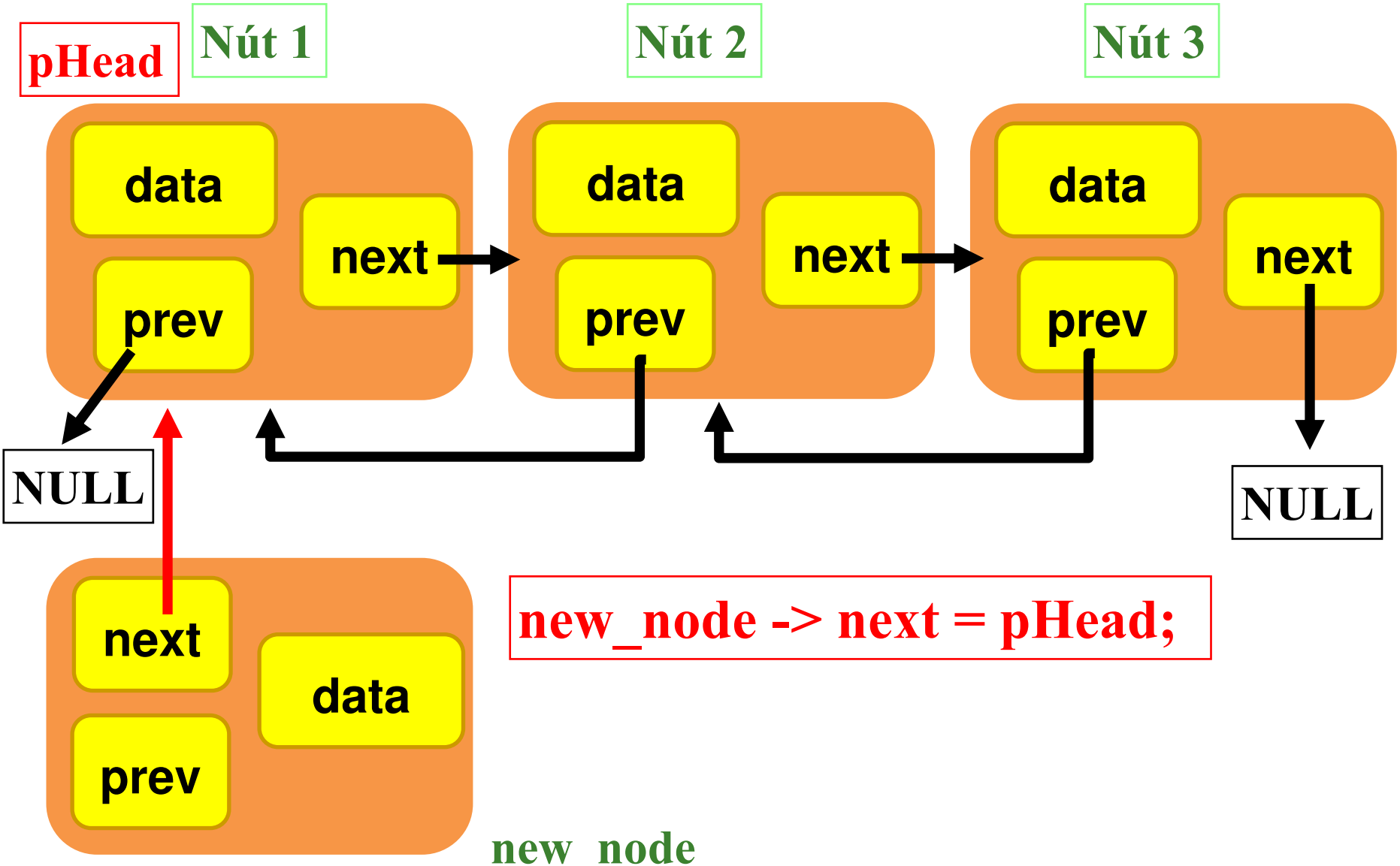
- **addFirst:** Kết nạp phần tử đầu tiên vào danh sách

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

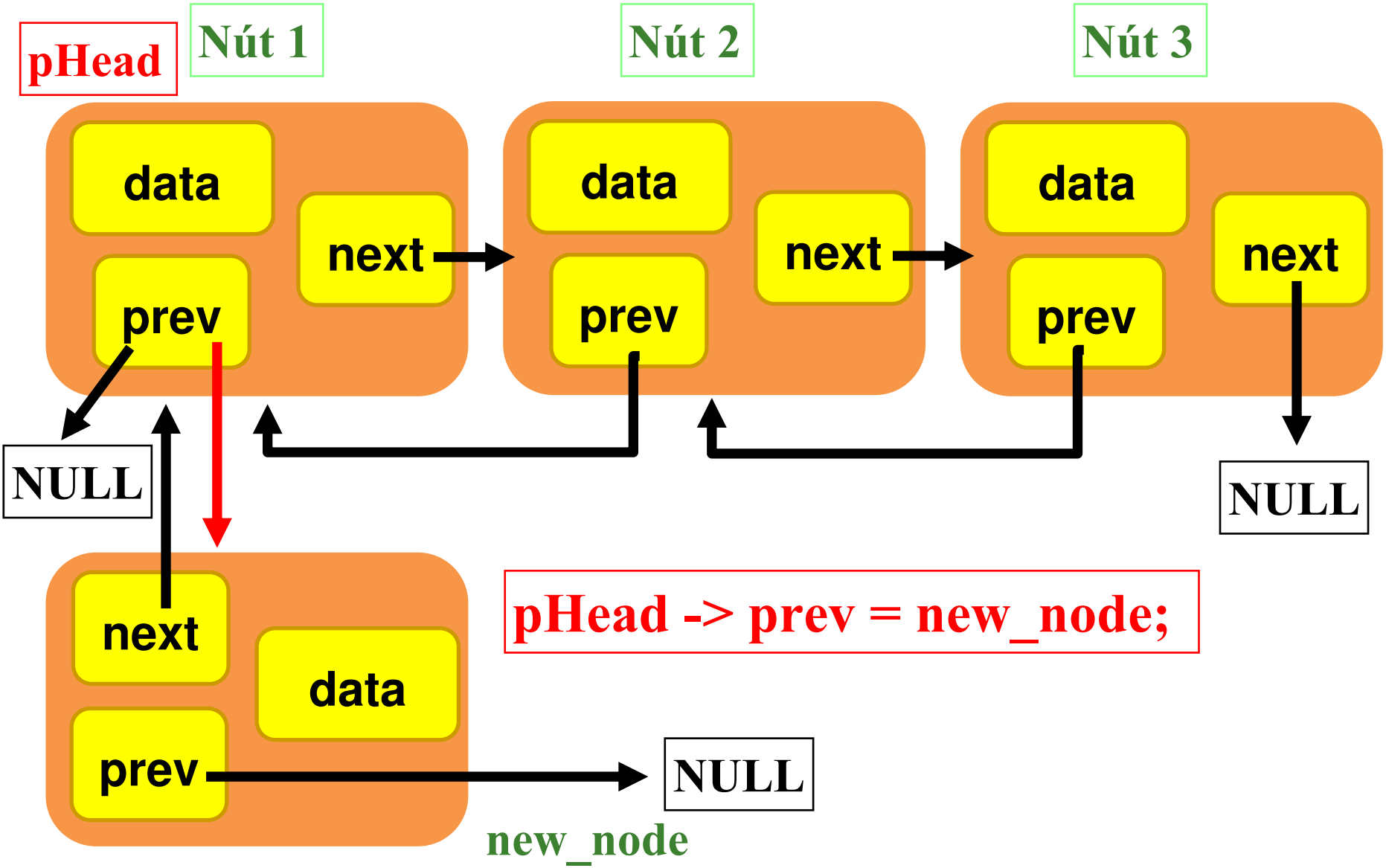
4.3.2 DLL – Thêm node vào đầu



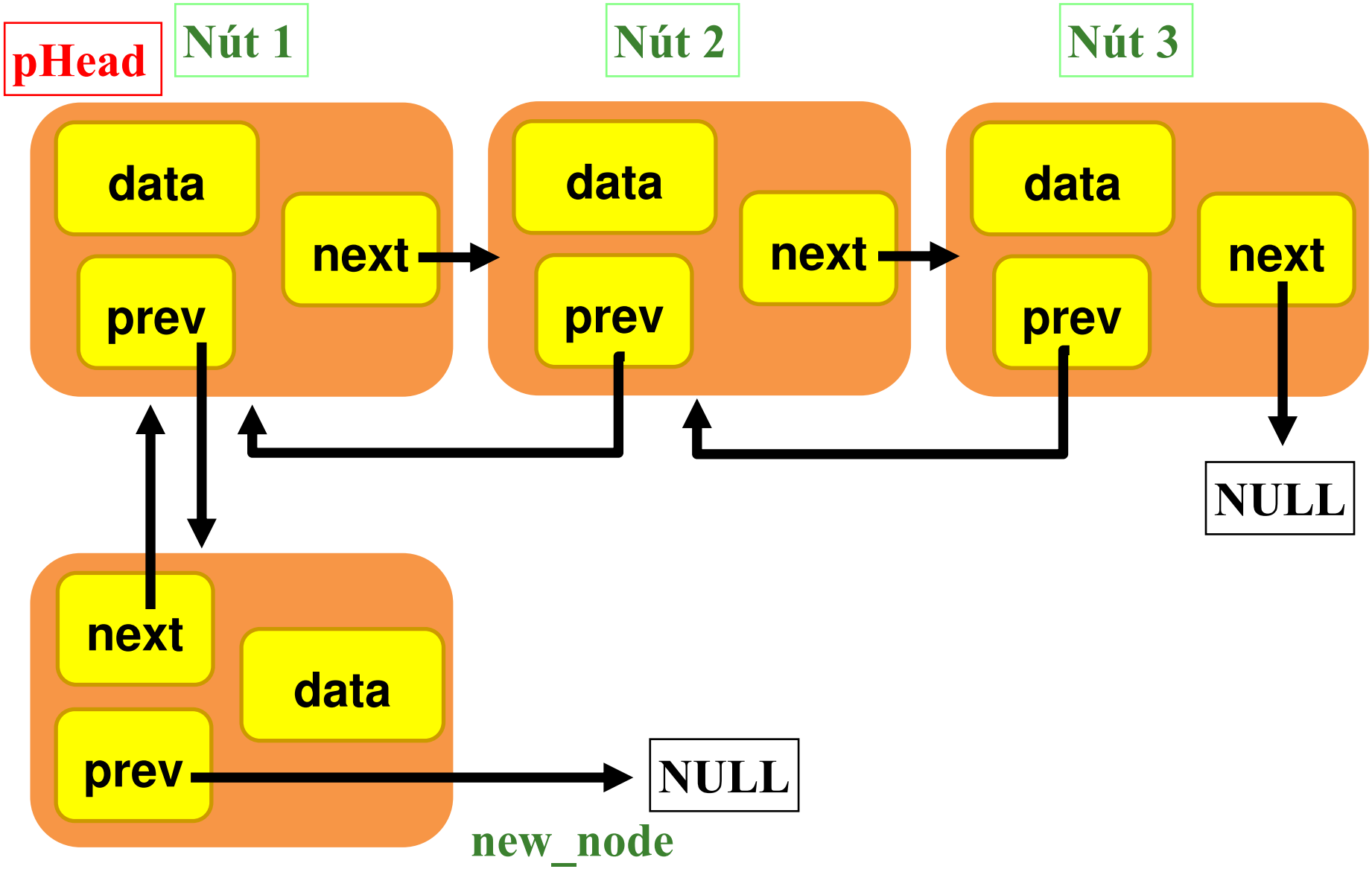
4.3.2 DLL – Thêm node vào đầu



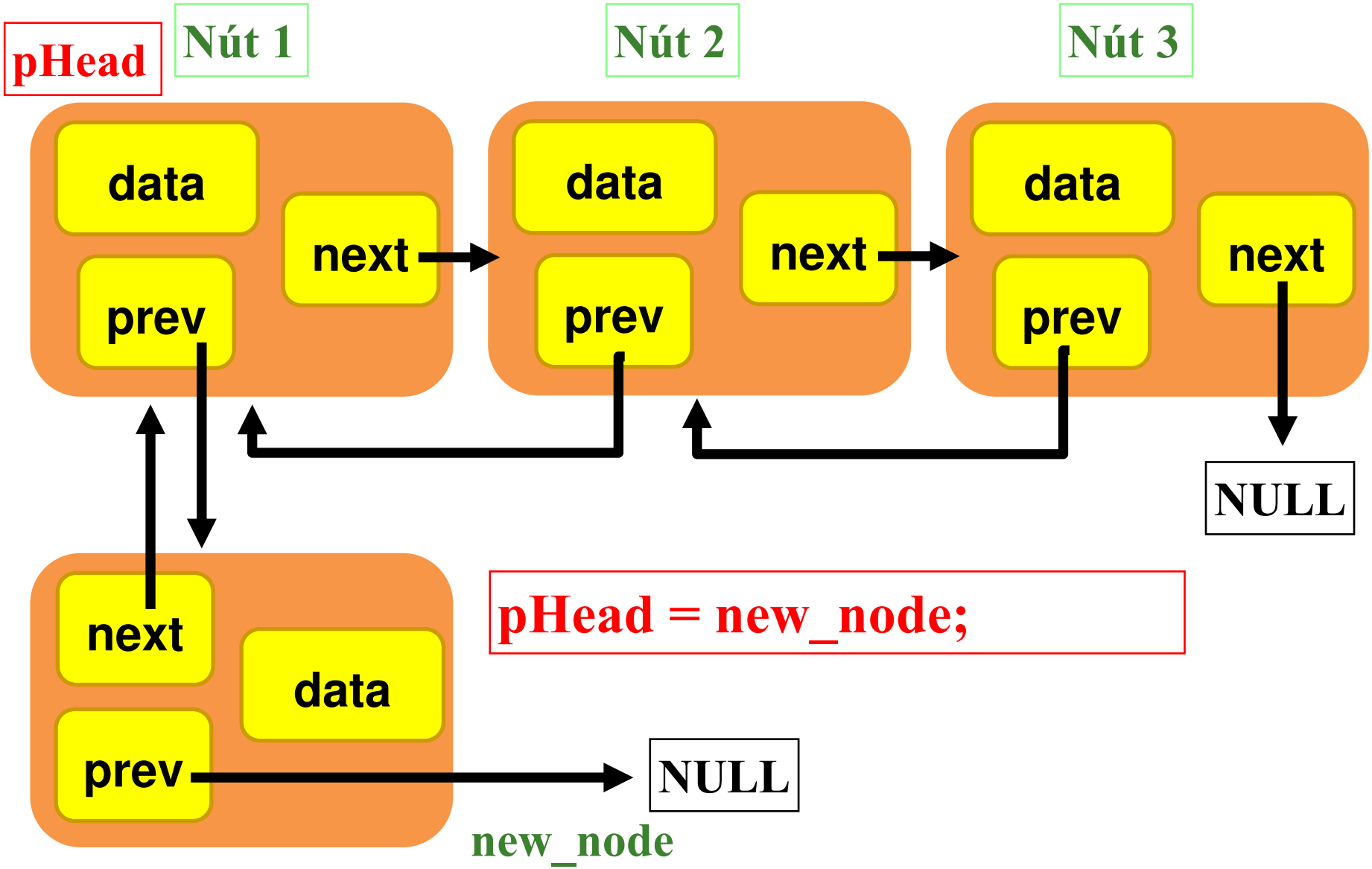
4.3.2 DLL – Thêm node vào đầu



4.3.2 DLL – Thêm node vào đầu



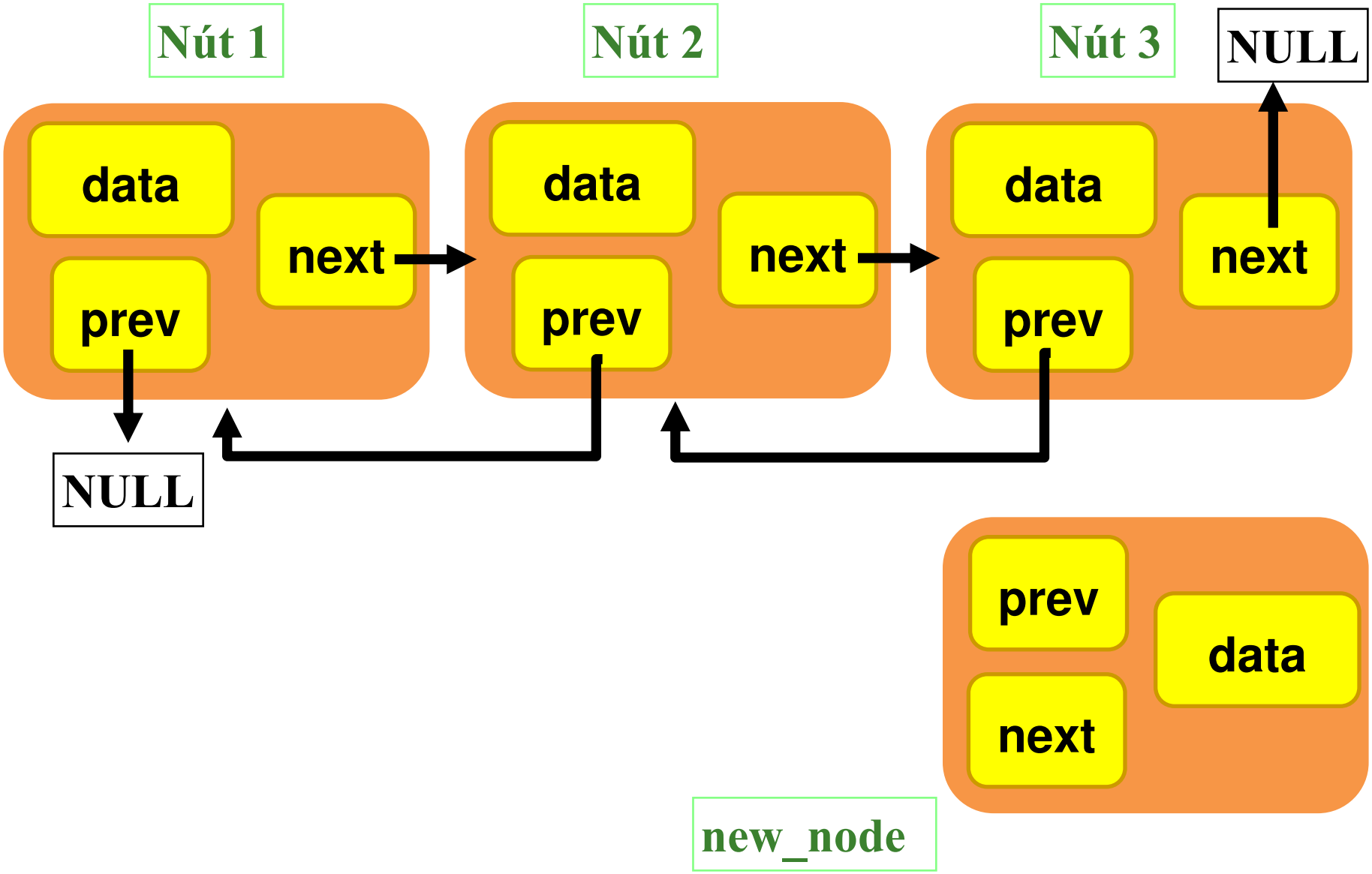
4.3.2 DLL – Thêm node vào đầu



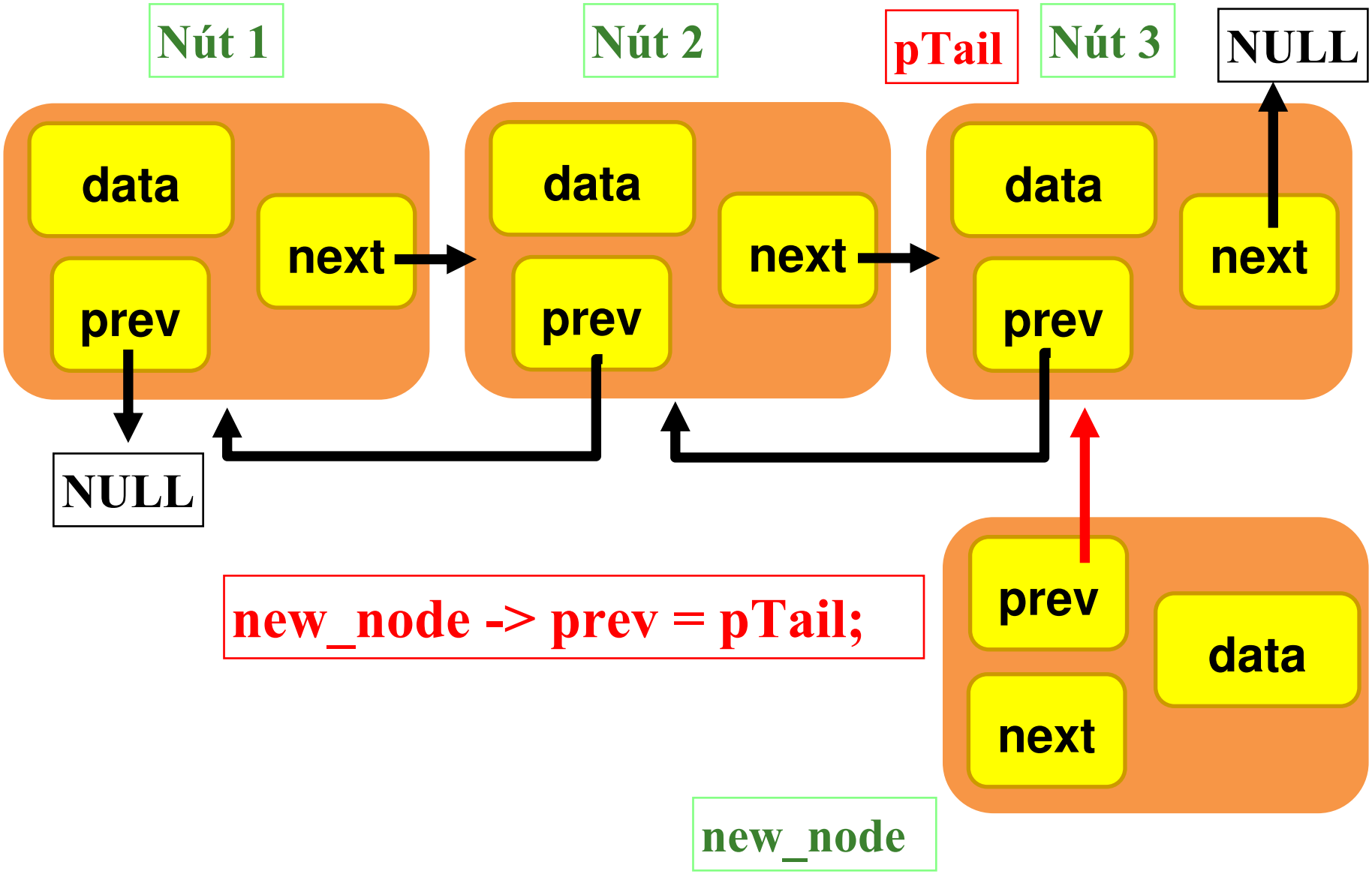
4.3.2 DLL - Thêm node vào đầu

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

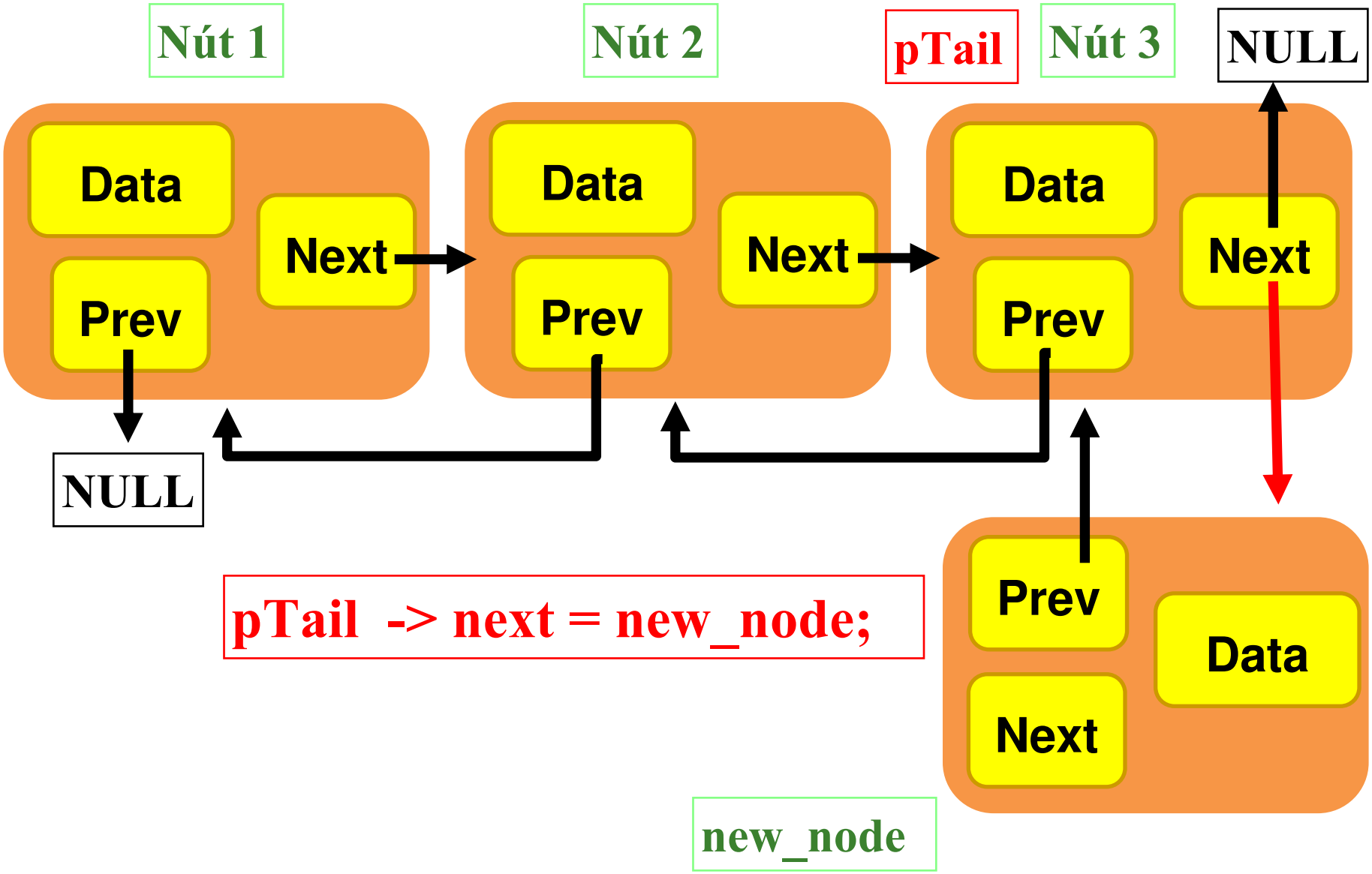
4.3.2 DLL - Thêm node vào cuối



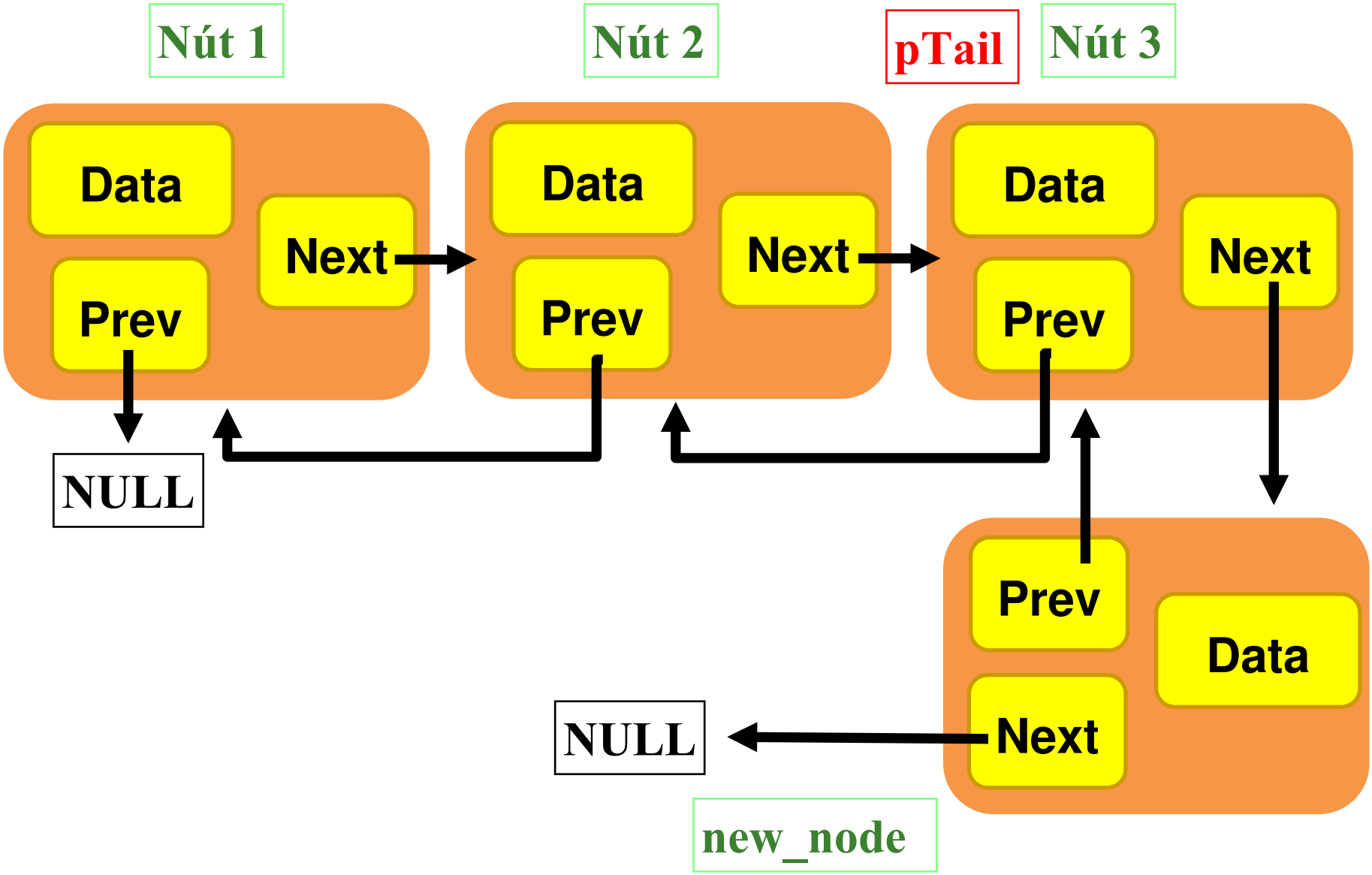
4.3.2 DLL - Thêm node vào cuối



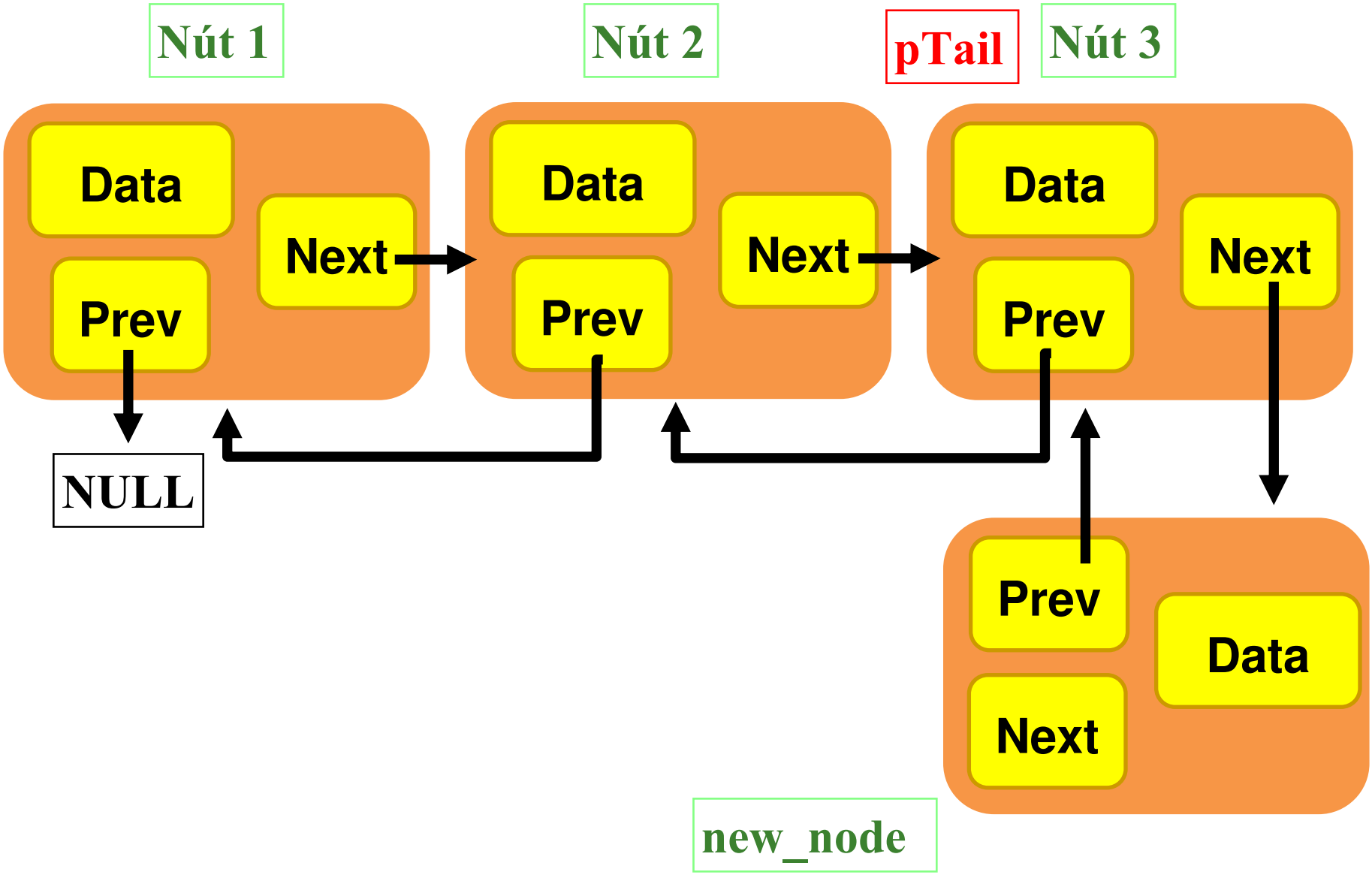
4.3.2 DLL - Thêm node vào cuối



4.3.2 DLL - Thêm node vào cuối



4.3.2 DLL - Thêm node vào cuối

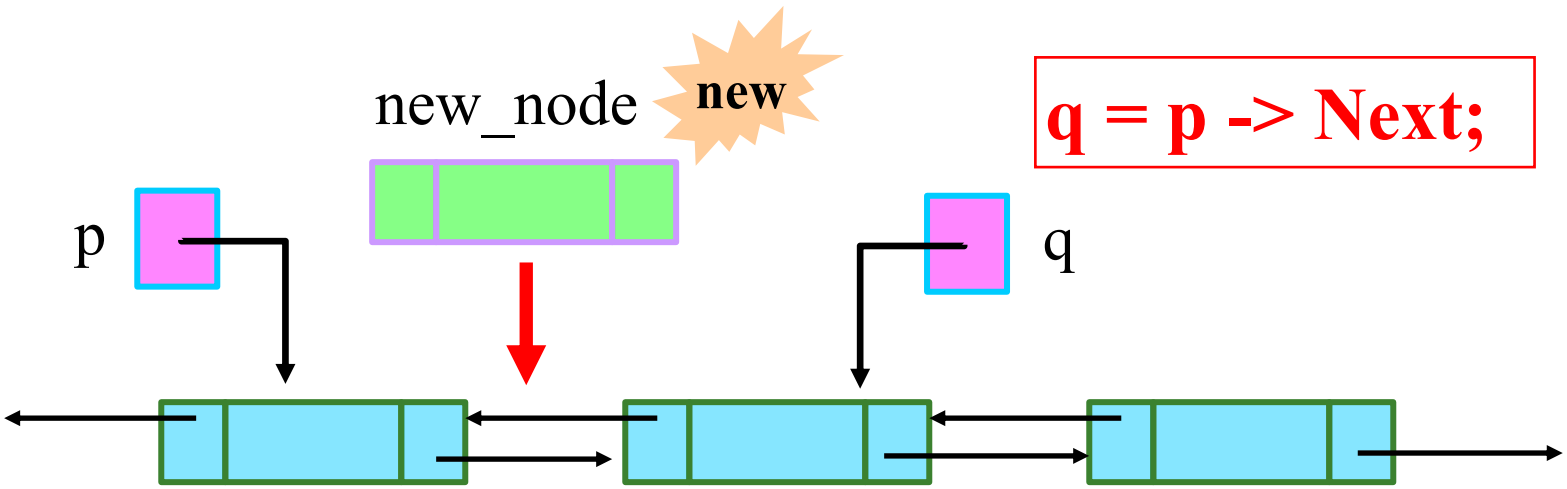


4.3.2 DLL - Thêm node vào cuối

```
1. void insertEnd(Node* &pHead, int x)
2. {  if (pHead == NULL)
3.          addFirst(pHead, x);
4.    else
5.    {     Node *new_node;
6.          new_node = creatNode(x);
7.          Node *pTail = pHead;
8.          while (pTail->next != NULL)
9.               pTail = pTail->next;
10.         new_node->prev = pTail;
11.         pTail->next = new_node;
12.   }
13.}
```

4.3.2 DLL - Bổ sung vào sau node p

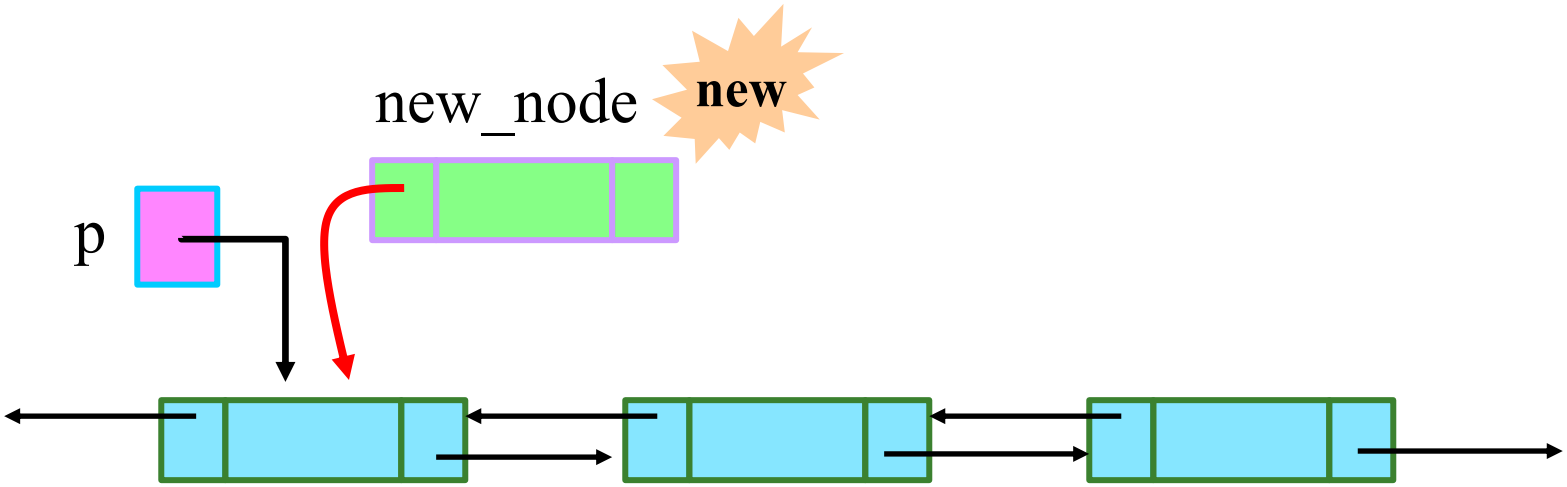
- **insertAfter**: Bổ sung vào sau node p



4.3.2 DLL - Bổ sung vào sau node p

- **insertAfter**: Bổ sung vào sau node p

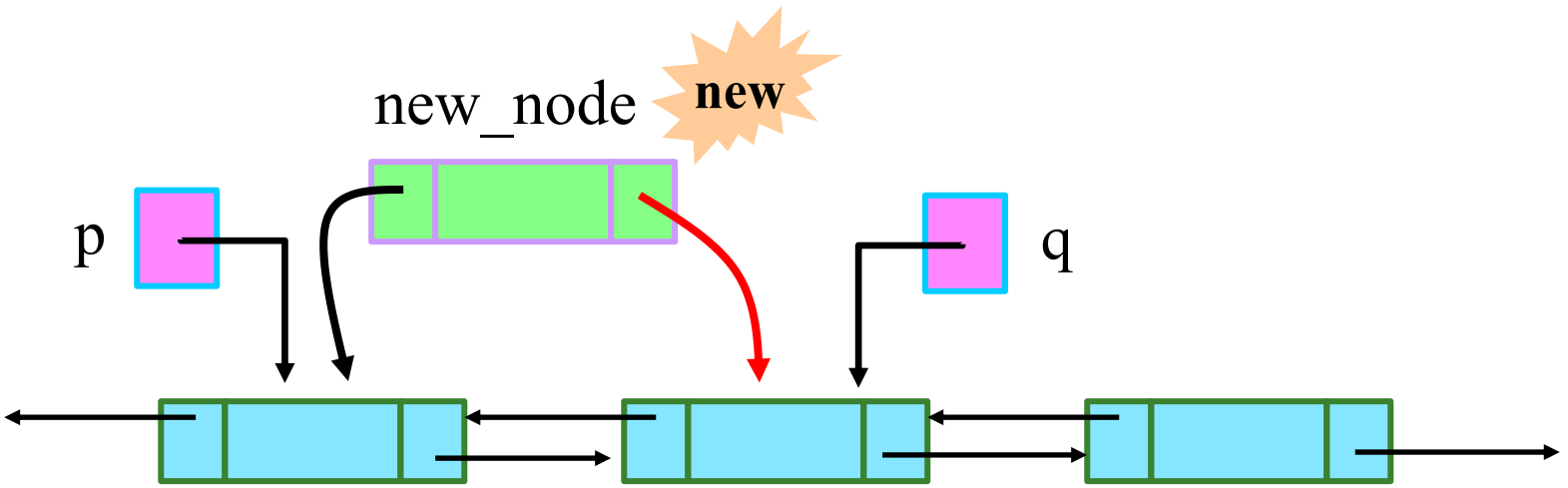
```
new_node -> prev = p;
```



4.3.2 DLL - Bổ sung vào sau node p

- **insertAfter**: Bổ sung vào sau node p

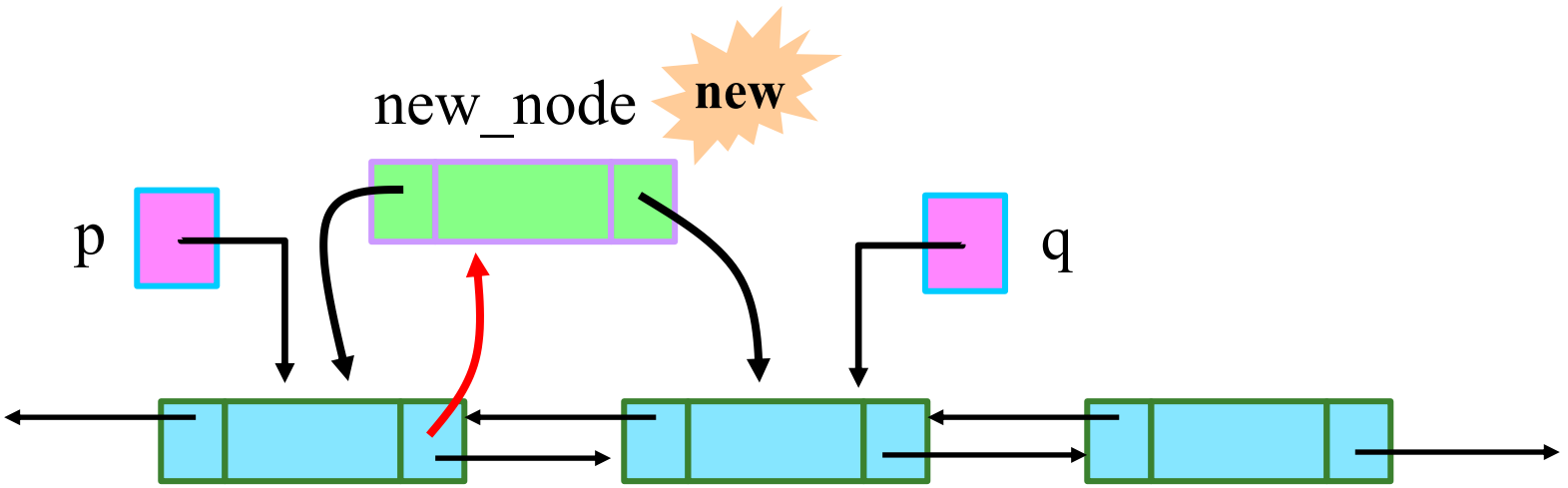
```
new_node -> next = q;
```



4.3.2 DLL - Bổ sung vào sau node p

- **insertAfter**: Bổ sung vào sau node p

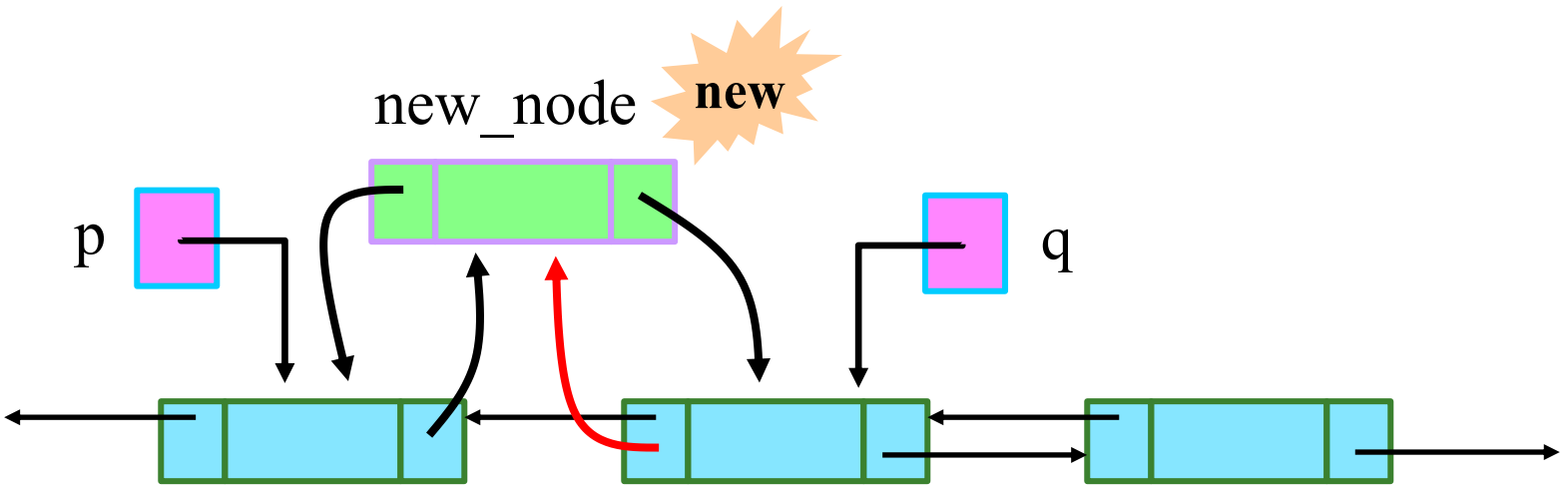
```
p -> next = new_node ;
```



4.3.2 DLL - Bổ sung vào sau node p

- **insertAfter**: Bổ sung vào sau node p

```
q -> prev = new_node ;
```



4.3.2 DLL - Bổ sung vào sau node p

```
1. void insertAfter(Node* &pHead, Node *p, int x)
2. {
3.    if (pHead == NULL)
4.          addFirst(pHead, x);
5.    else if (p->Next == NULL)
6.          insertEnd(pHead, x);
7.    else
8.    {
9.          Node *new_node, *q;
```

4.3.2 DLL - Bổ sung vào sau node p

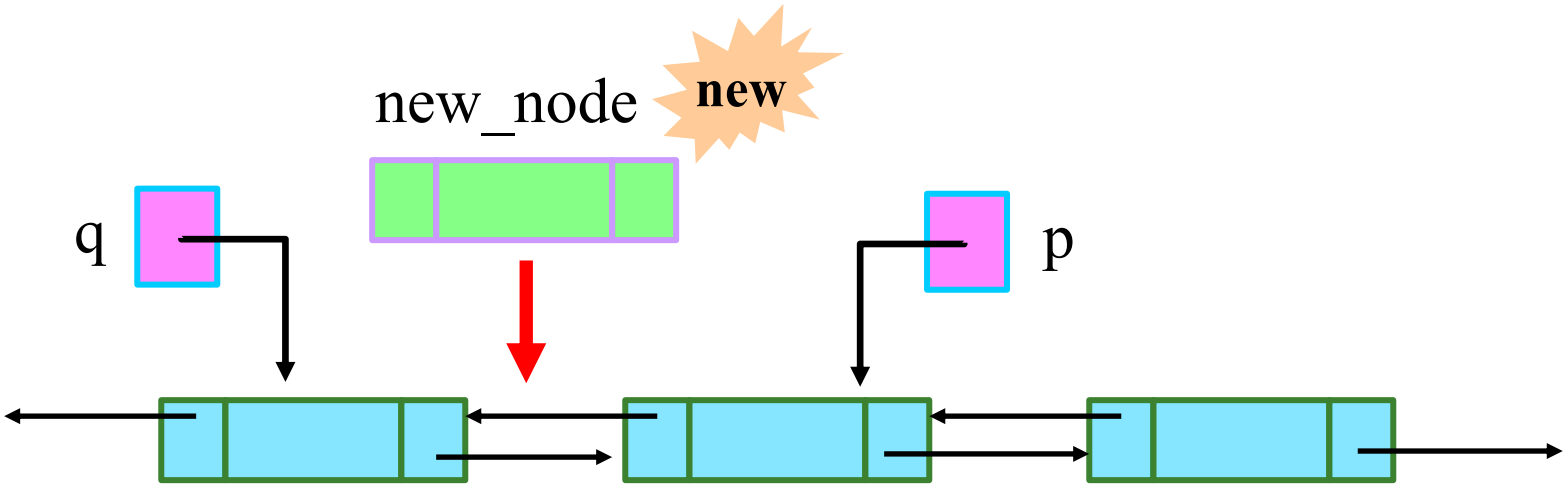
```
10.         new_node = creatNode(x);
11.         q = p->next;
12.         //noi new_node voi p, q
13.         new_node->prev = p;
14.         new_node->next = q;
15.         //noi p, q voi new_node
16.         p->next = new_node;
17.         q->prev = new_node;
18.   }
19.}
```

4.3.2 DLL - Bổ sung vào trước node p

- **insertBefore**: Bổ sung vào trước node p

Khai báo con trỏ q, cho q trỏ vào node trước p:

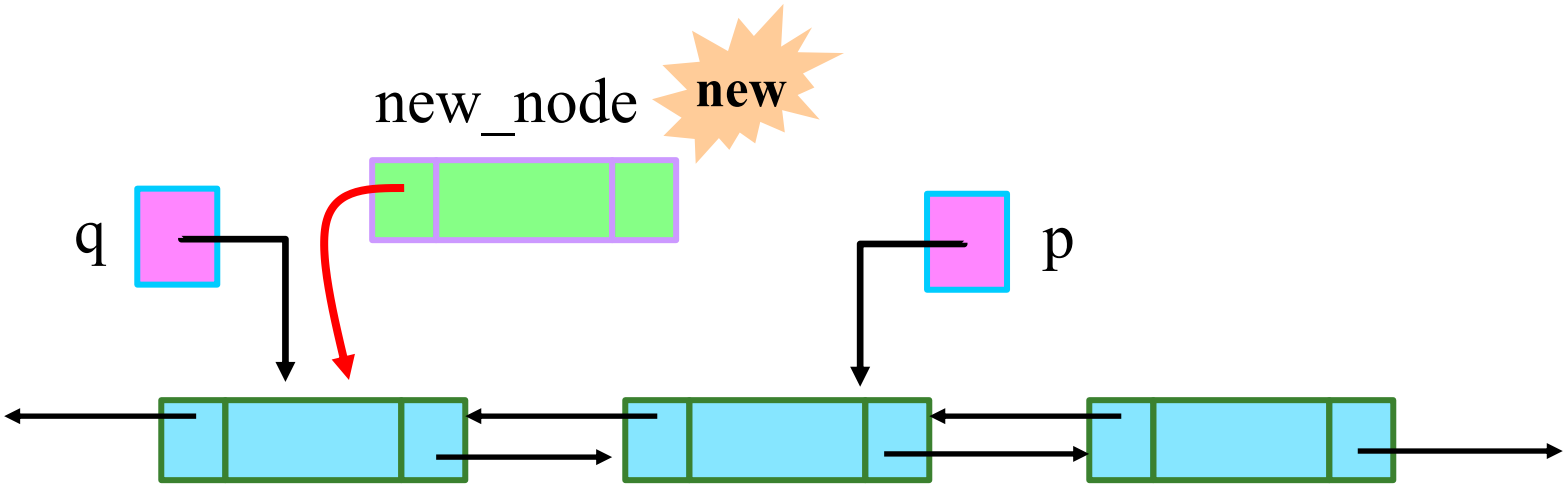
```
q = p -> prev;
```



4.3.2 DLL - Bổ sung vào trước node p

- **insertBefore**: Bổ sung vào trước node p

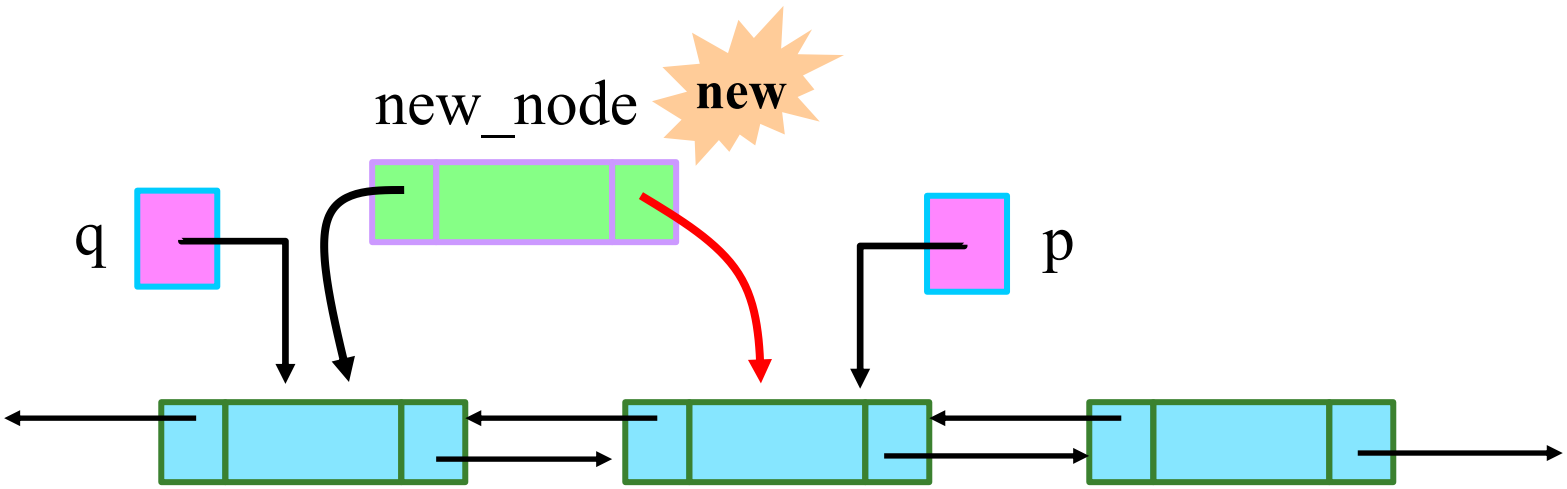
```
new_node -> prev = q;
```



4.3.2 DLL - Bổ sung vào trước node p

- **insertBefore**: Bổ sung vào trước node p

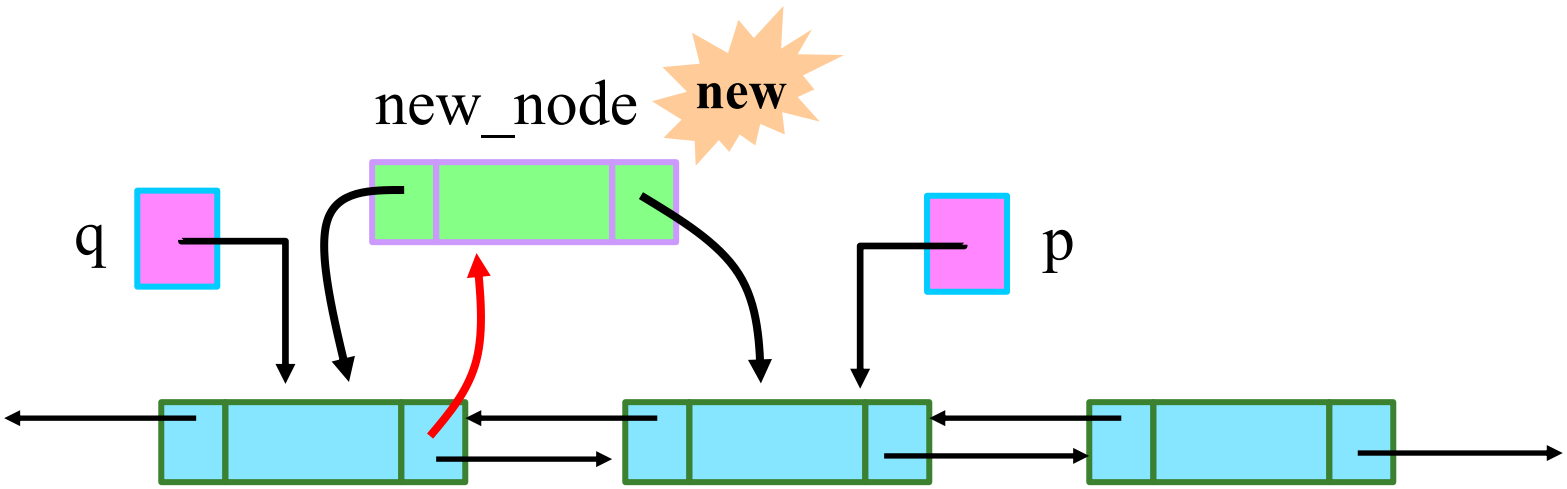
```
new_node -> next = p;
```



4.3.2 DLL - Bổ sung vào trước node p

- **insertBefore**: Bổ sung vào trước node p

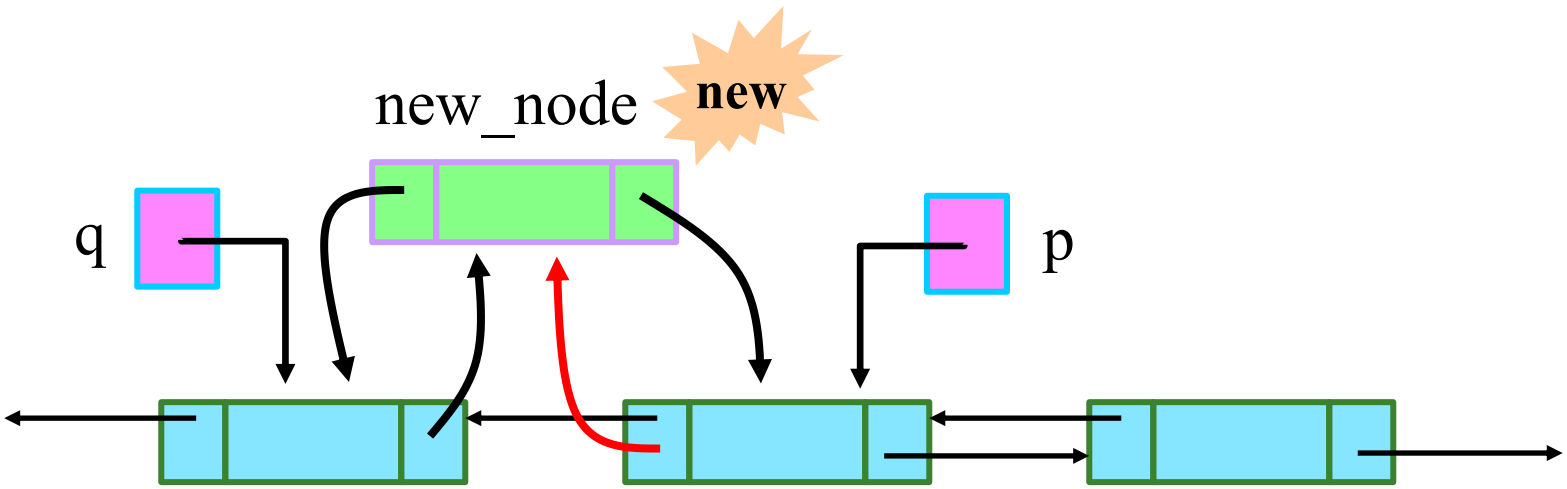
```
q -> next = new_node ;
```



4.3.2 DLL - Bổ sung vào trước node p

- **insertBefore**: Bổ sung vào trước node p

```
p -> prev = new_node ;
```



4.3.2 DLL - Bổ sung vào trước node p

```
1. void insertBefore(Node* &pHead,Node* p, int x)
2. {  if (pHead == NULL)
3.         addFirst(pHead, x);
4.    else if (p->prev == NULL)
5.         insertBegin(pHead, x);
6.    else
7.    {
8.         Node *new_node, *q;
9.         new_node = creatNode(x);
```

4.3.2 DLL - Bổ sung vào trước node p

```
10.   q = p->prev;
11.   //noi new_node voi p, q
12.         new_node->prev = q;
13.         new_node->next = p;
14.   //noi p, q voi new_node
15.         q->next = new_node;
16.         p->prev = new_node;
17.   }
18.}
```

4.3.2 DLL – Đếm số node

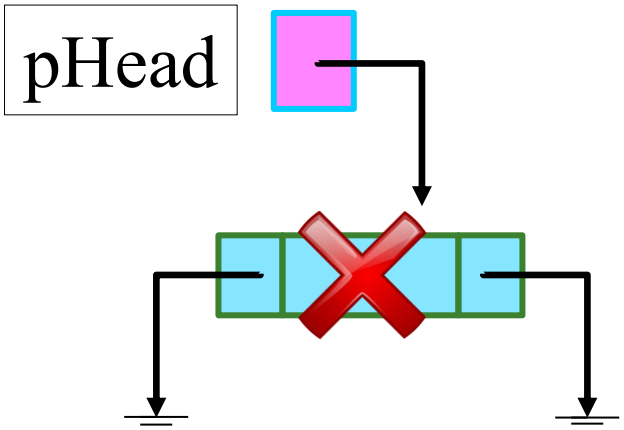
```
1. int countNode(Node *pHead)
2. {
3.     int dem = 0;
4.     Node *p = pHead;
5.     while(p != NULL)
6.     {
7.         dem++;
8.         p = p->next;
9.     }
10.    return dem;
11.}
```

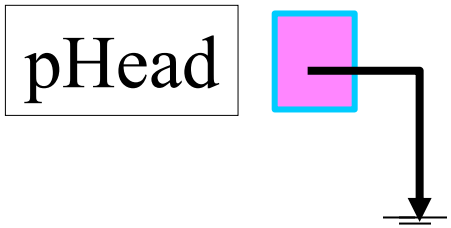
4.3.2 DLL - Thêm 1 node vào vị trí cho trước

```
1. void insertPosK(Node* &pHead)
2. {  int x;
3.    cout<<"\nNhap gia tri can chen: ";
4.    cin>>x;
5.    int vi_tri;
6.    int n = countNode(pHead);
7.    do
8.    {
9.       cout<<"Nhap vi tri can chen: ";
10.      cin>>vi_tri;
11.   }while (vi_tri <= 0 || vi_tri > n);
12.   ..........................
```

4.3.2 DLL – Loại bỏ node đầu

- Danh sách có 1 node: **pHead -> next == NULL**





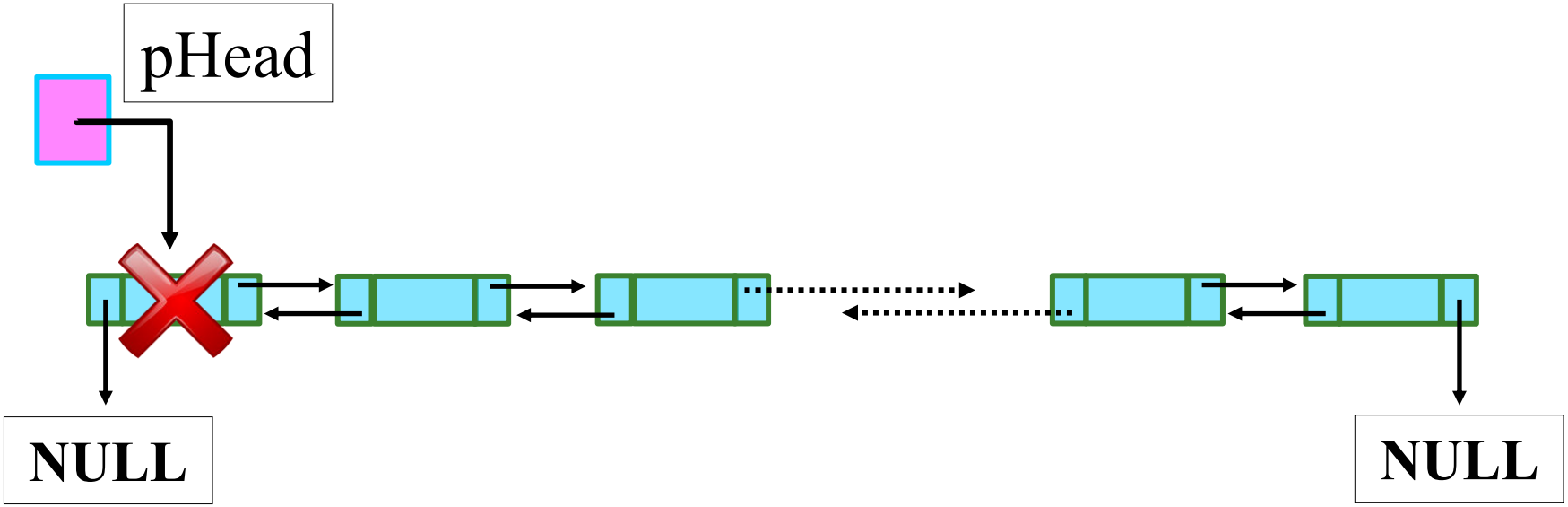
delete pHead;

pHead = NULL;

4.3.2 DLL – Loại bỏ node đầu

- Danh sách có nhiều hơn 1 node:

pHead->next != NULL

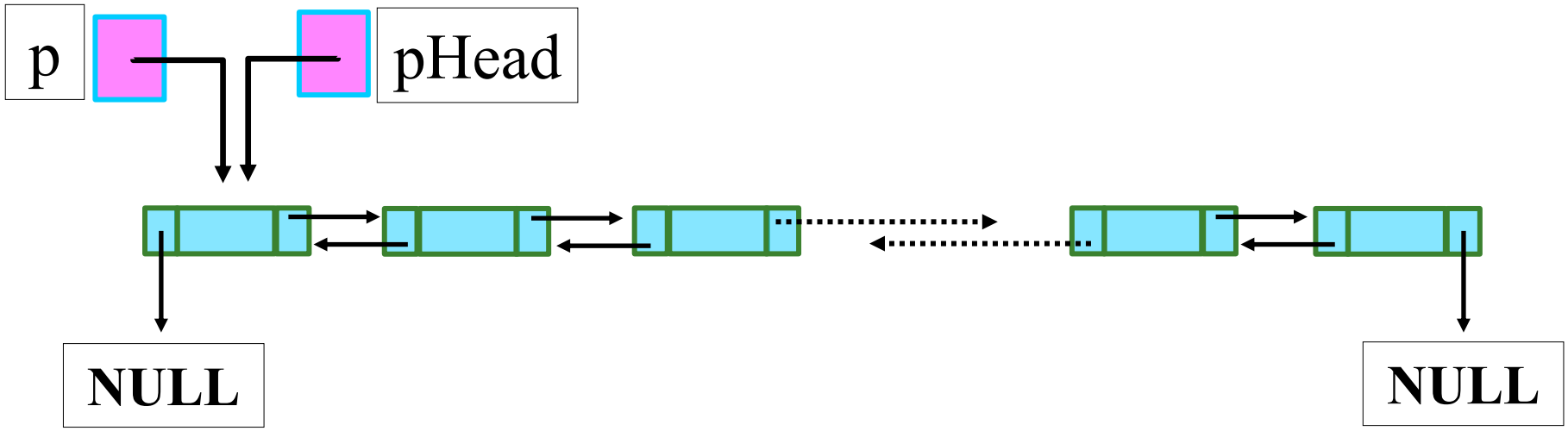


Loại bỏ node đầu

4.3.2 DLL – Loại bỏ node đầu

- Danh sách có nhiều hơn 1 node:

pHead->next != NULL



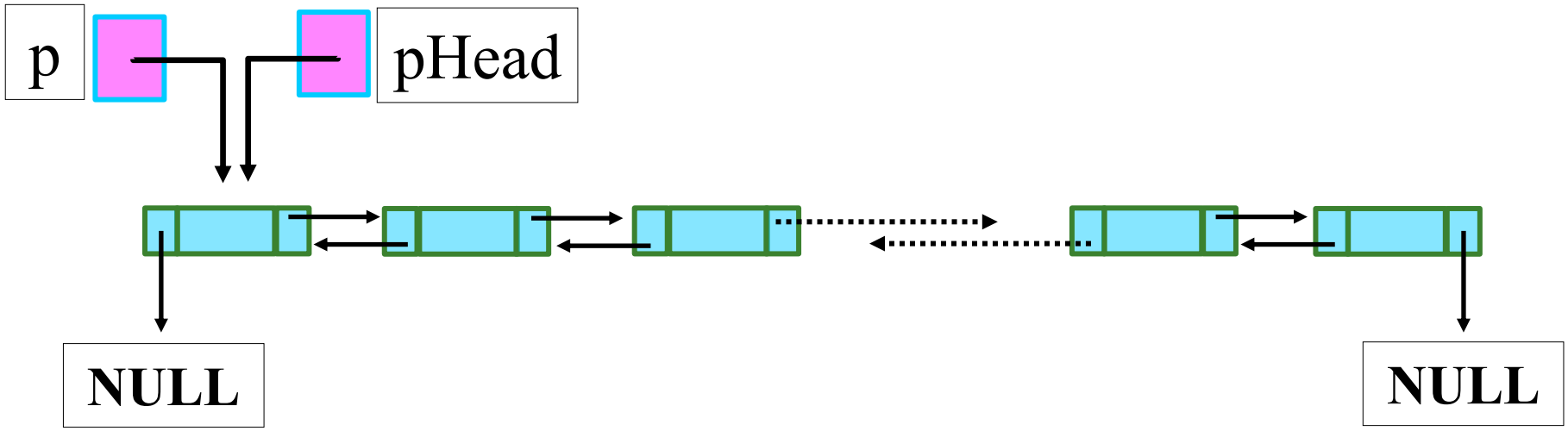
Cho con trỏ p trỏ vào đầu cùng pHead:

p = pHead;

4.3.2 DLL – Loại bỏ node đầu

- Danh sách có nhiều hơn 1 node:

pHead->next != NULL



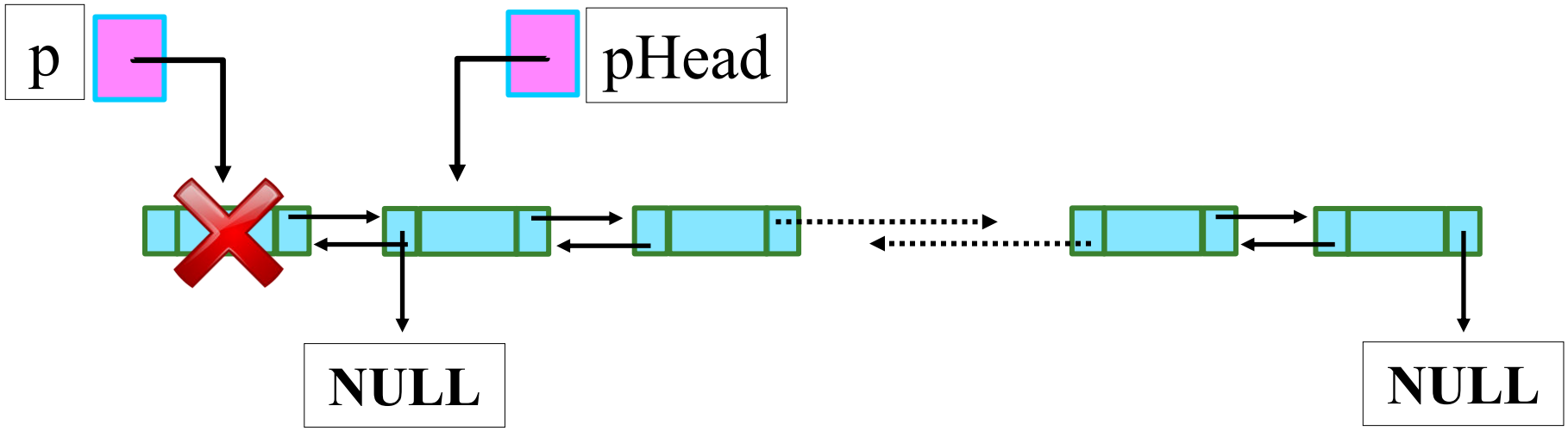
Dịch chuyển con trỏ pHead về phía sau 1 node:

pHead = pHead -> next; pHead -> prev = NULL;

4.3.2 DLL – Loại bỏ node đầu

- Danh sách có nhiều hơn 1 nút:

pHead->next != NULL



Xoá nút p:

delete p;

p = NULL;

4.3.2 DLL - Loại bỏ node đầu

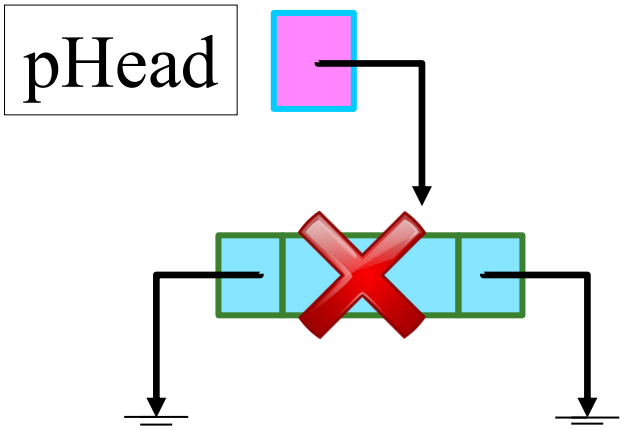
```
1. void deleteFirst (Node* &pHead)
2. {
3.    if (pHead == NULL)
4.          return;
5.    else if (pHead->next == NULL)
6.    {
7.          delete pHead;
8.          pHead = NULL;
9.    }
```

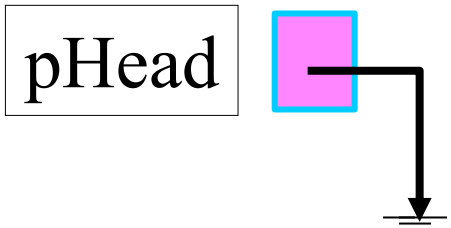
4.3.2 DLL - Loại bỏ node đầu

```
10.     else
11.     {
12.         Node *p;     // p tro nut loai bo
13.         p = pHead;   // p tro nut dau
14.         pHead = pHead->next;
15.         pHead->prev = NULL;
16.         delete p;
17.         p = NULL;
18.     }
19.}
```

4.3.2 DLL – Loại bỏ node cuối

- Danh sách có 1 node: **pHead -> next == NULL**





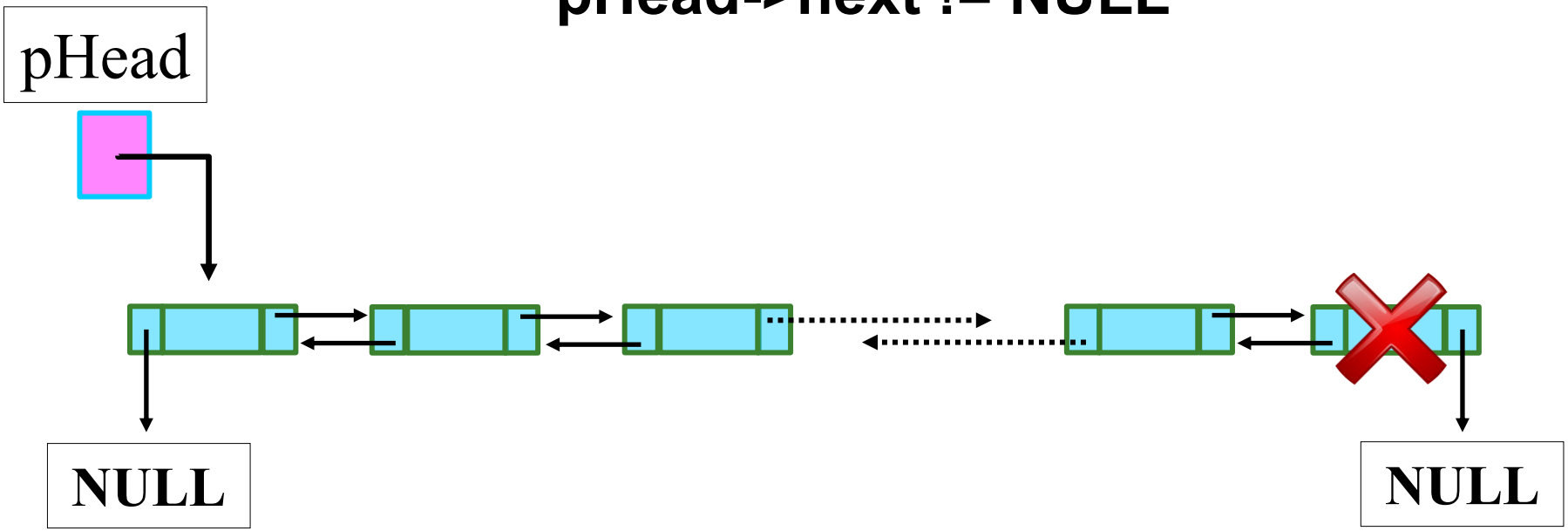
delete pHead;

pHead = NULL;

4.3.2 DLL – Loại bỏ node cuối

- Danh sách có nhiều hơn 1 node:

pHead->next != NULL

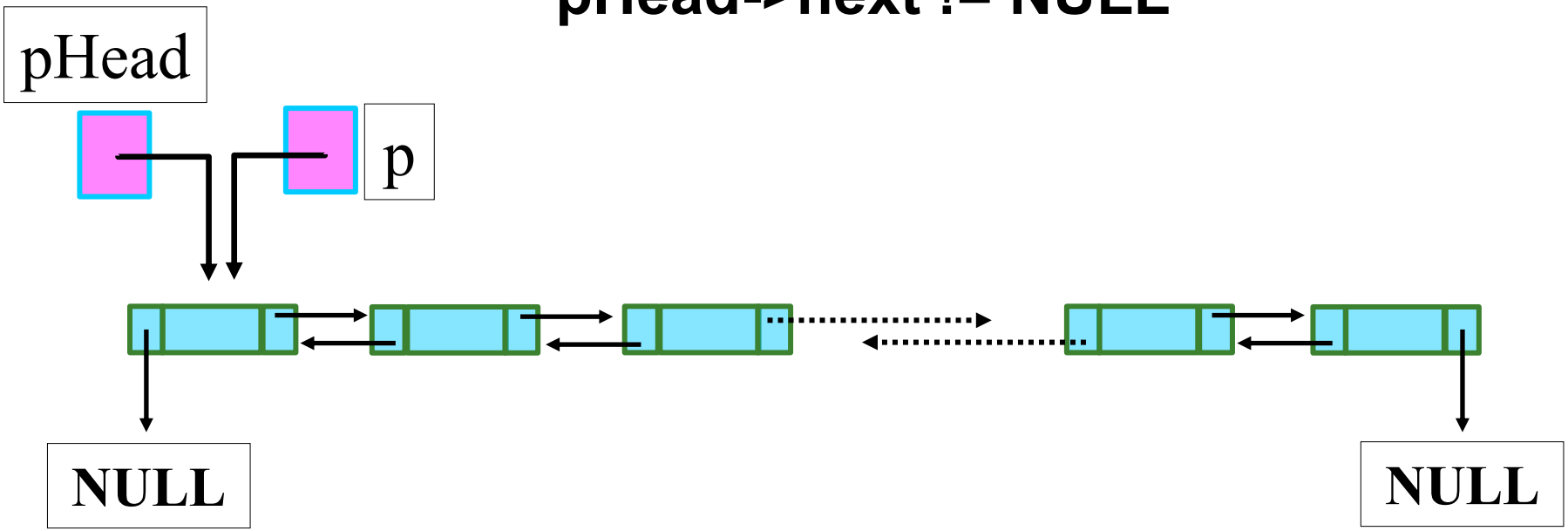


Loại bỏ node cuối

4.3.2 DLL – Loại bỏ node cuối

- Danh sách có nhiều hơn 1 node:

pHead->next != NULL



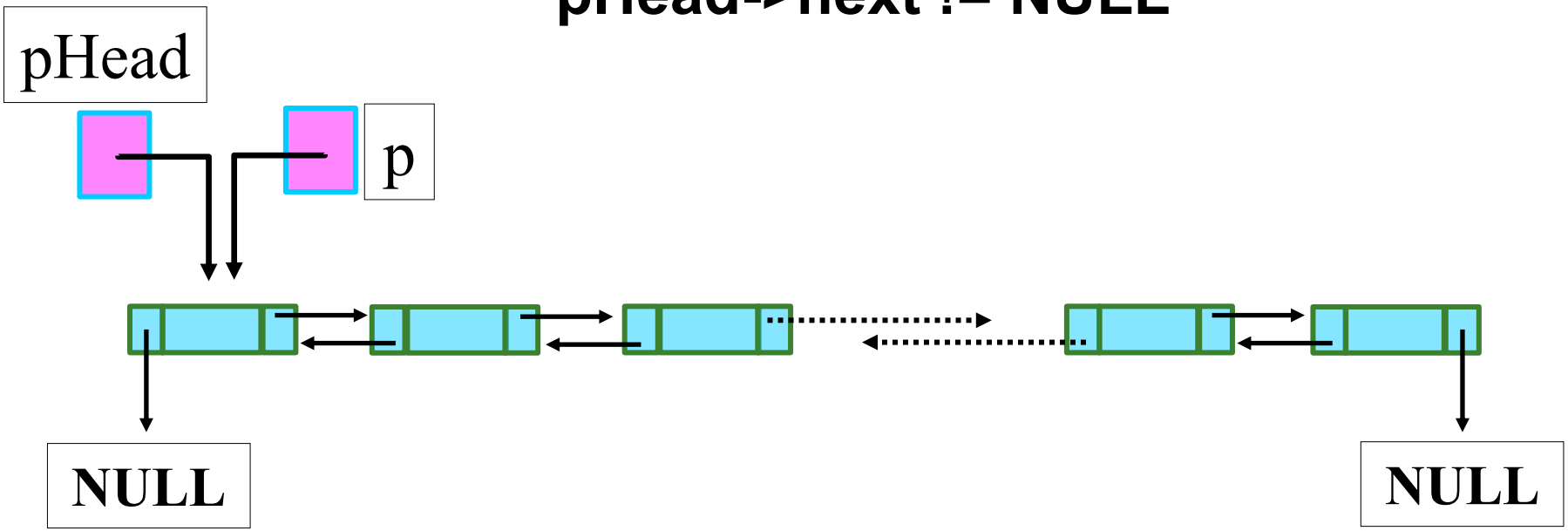
Cho con trỏ p trỏ vào node đầu cùng pHead:

p = pHead;

4.3.2 DLL – Loại bỏ node cuối

- Danh sách có nhiều hơn 1 node:

pHead->next != NULL



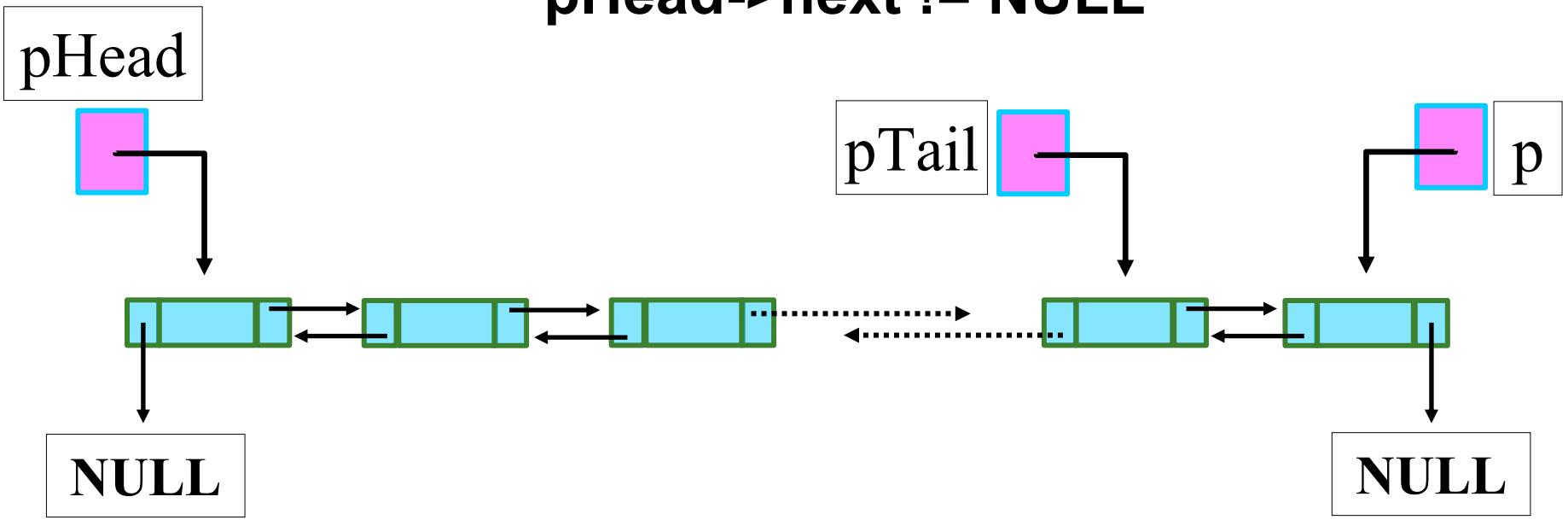
Dịch chuyển con trỏ p xuống cuối danh sách:

```
while (p->next != NULL)
        p = p->next;
```

4.3.2 DLL – Loại bỏ node cuối

- Danh sách có nhiều hơn 1 node:

pHead->next != NULL



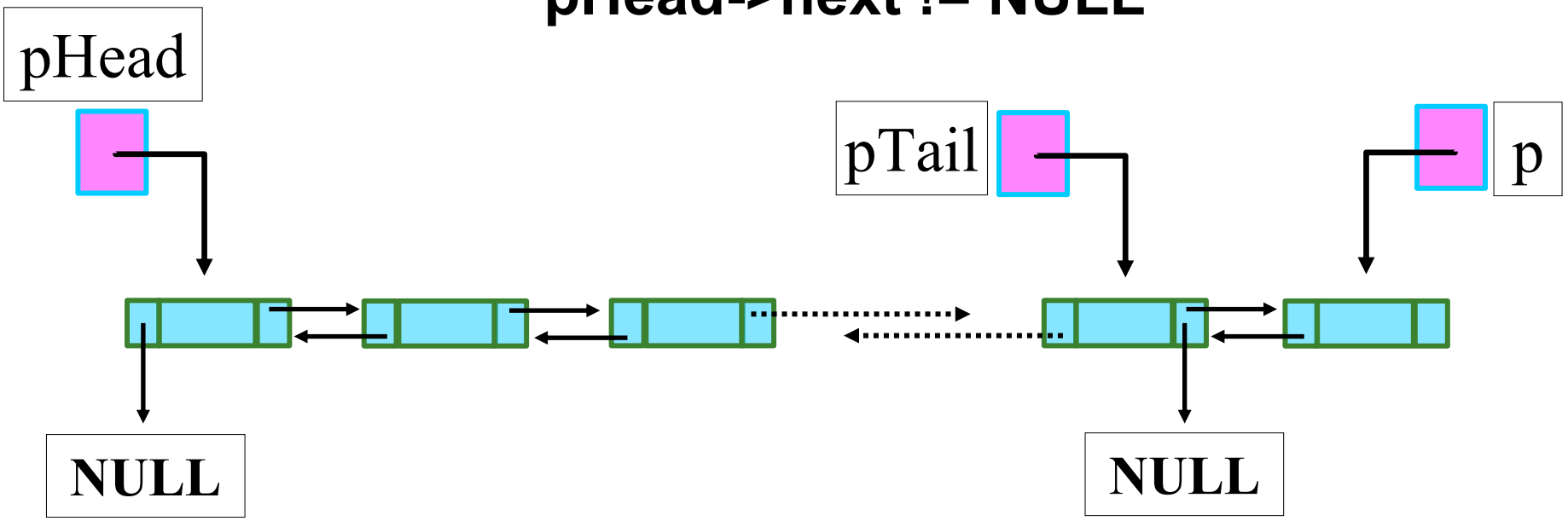
Khai báo con trỏ pTail, cho pTail trỏ vào node trước p:

pTail = p->prev; **pTail->next = NULL;**

4.3.2 DLL – Loại bỏ node cuối

- Danh sách có nhiều hơn 1 node:

pHead->next != NULL



Xoá p:

delete p; **p = NULL;**

4.3.2 DLL - Loại bỏ node cuối

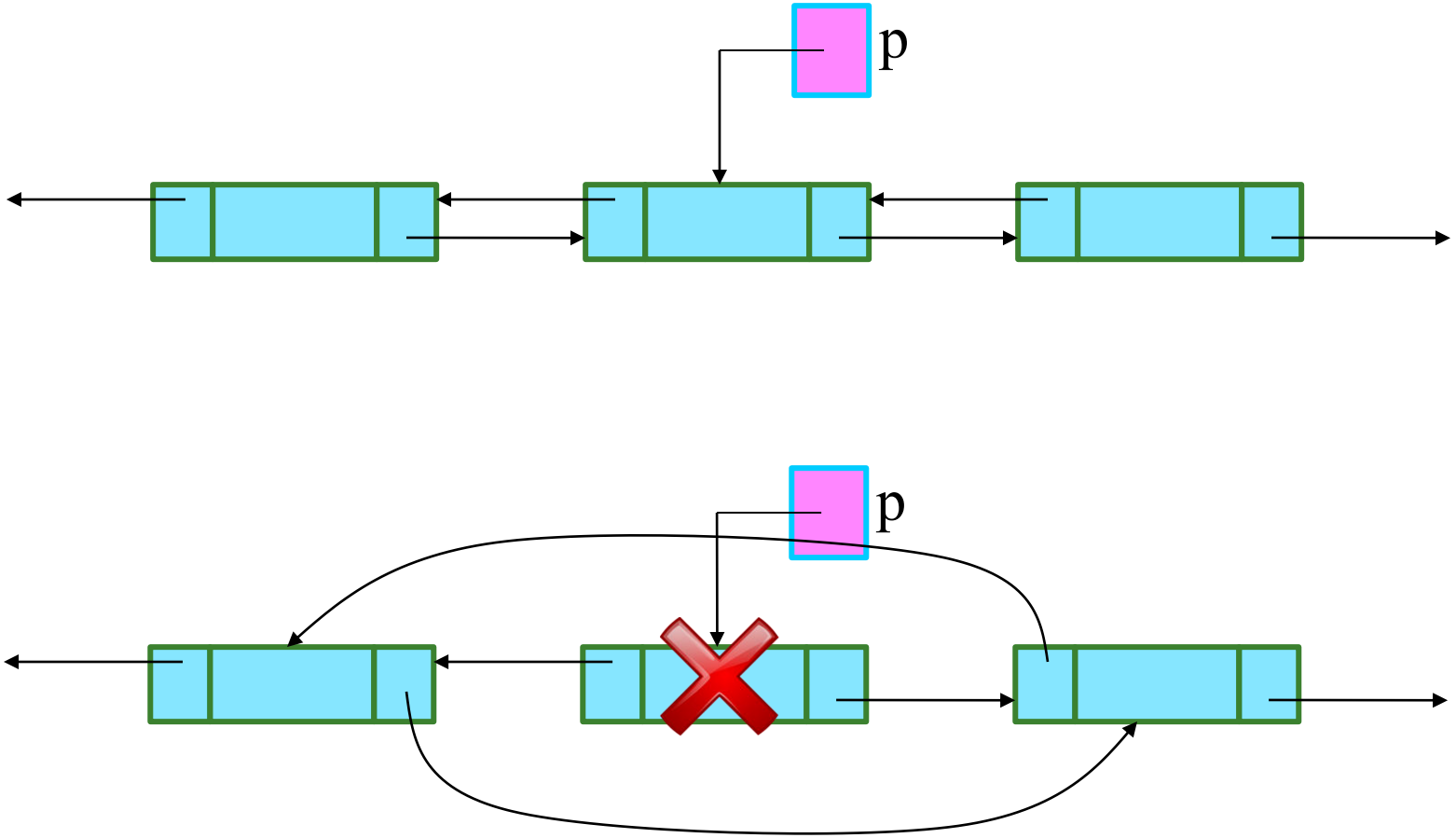
```
1. void deleteLast (Node* &pHead)
2. {  if (pHead == NULL)
3.          return;
4.    else if (pHead->next == NULL)
5.    {
6.          delete pHead;
7.          pHead = NULL;
8.    }
9.    else
10.   {     Node *p;
11.         p = pHead;
```

4.3.2 DLL - Loại bỏ node cuối

```
12.         while (p->next != NULL)
13.               p = p->next;
14.
15.         Node *pTail;
17.         pTail = p->prev;
18.         pTail->next = NULL;
19.         delete p;
20.         p = NULL;
21.   }
22.}
```

4.3.2 DLL - Loại bỏ node p

- **deleteNodeP**: Loại bỏ (xóa) node p



4.3.2 DLL - Loại bỏ node p

```
1. void deleteNodeP(Node* &pHead, Node* p)
2. {
3.     if (pHead == NULL)    // Danh sach trong
4.     {
5.         cout<<"Danh sach TRONG!";
6.         return;
7.     }
8.     else if (p->prev == NULL)   // p la nut dau
9.     {
10.        deleteFirst(pHead);
11.        return;
12.    }
13.
```

4.3.2 DLL – Loại bỏ node p

```
14.   else if (p->next == NULL)// p la nut cuoi
15.   {     deleteLast(pHead);
16.         return;
17.   }
18.   else
19.   {     Node *left, *right;
20.         left = p->prev;
21.         right = p->next;
22.         left->next = right;
23.         right->prev = left;
24.         delete p;  p = NULL;
25.   }
26.}
```

4.3.2 DLL - Xóa 1 node tại vị trí cho trước

```
1. void deleteNode(Node* &pHead)
2. {  int vi_tri;
3.    int n = countNode(pHead);
4.    do
5.    {
6.       cout<<"Nhap vi tri can xoa: ";
7.       cin>>vi_tri;
8.    }while (vi_tri <= 0 || vi_tri > n);
9.    if(vi_tri == 1)deleteFirst(pHead);
10.   else if(vi_tri == n)
11.          deleteLast(pHead);
12.   ..................
```

4.3.2 DLL - Sắp xếp

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

4.3.2 DLL – Tìm kiếm

 valueSearch: Tìm kiếm

- Xuất phát từ đầu danh sách
- Nếu pHead->data == x trả về địa chỉ node đầu
- Ngược lại qua phần tử tiếp theo, lặp lại công việc
- Dừng khi hết danh sách
- Không tìm thấy trả về NULL

4.3.2 DLL - Tìm kiếm

```
1. Node* searchValue Node* pHead, int x)
2. {
3.    if (pHead == NULL) return NULL;
4.    Node *p = pHead;
5.    while ((p != NULL) &&(p->data != x))
6.          p = p->next;
7.    return p;
8. }
```

Câu hỏi củng cố bài

1. Từ một đỉnh bất kỳ của danh sách liên kết kép:

A. Ta chỉ duyệt được các node bên phải danh sách

B. Ta duyệt được các node bên phải và bên trái danh sách

C. Ta chỉ duyệt được các node bên trái danh sách

D. Ta không duyệt được các node bên phải và bên trái danh sách

Câu hỏi củng cố bài

2. Khi loại bỏ nút p khỏi danh sách liên kết kép 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

Câu hỏi củng cố bài

3. Dấu hiệu nào dưới đây cho biết node p của một danh sách liên kết kép là node cuối:

A. p -> next != NULL

B. p -> info != NULL

C. p -> info == NULL

D. p -> next == NULL

Câu hỏi củng cố bài

4. Dấu hiệu nào dưới đây cho biết danh sách liên kết kép có nút đầu trỏ bởi p rỗng:

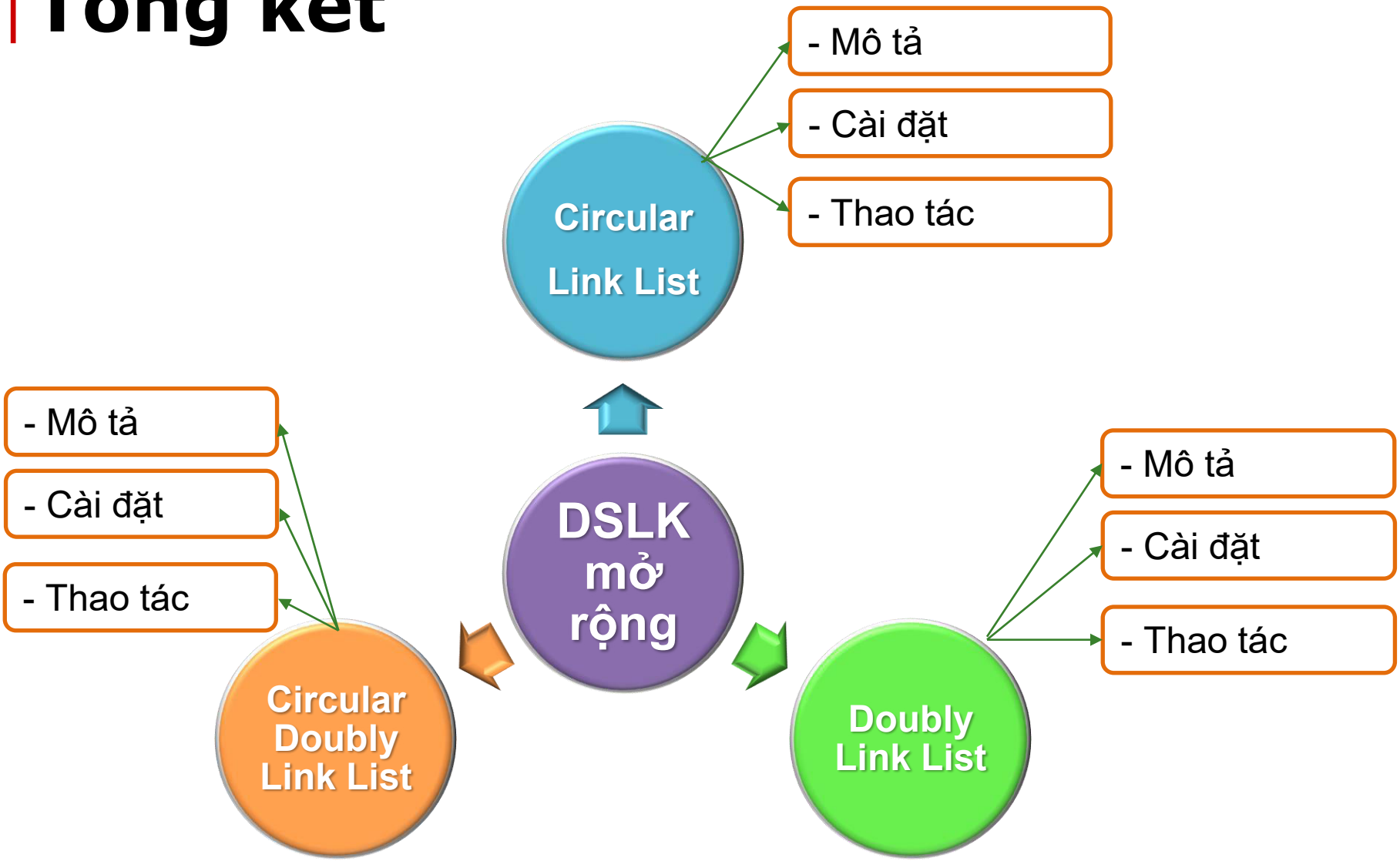
A. p->next == NULL

B. p -> next != NULL

C. p == NULL

D. p != NULL

Tổng kết


- Mô tả
- Cài đặt
- Thao tác
Circular Link List
- Mô tả
- Cài đặt
- Thao tác
DSLK mở rộng
- Mô tả
- Cài đặt
- Thao tác
Circular Doubly Link List
Doubly Link List

Tài liệu tham khảo

[1]. Giáo trình Cấu trúc dữ liệu và giải thuật – Lê Văn Vinh, NXB Đại học quốc gia TP HCM, 2013

[2]. Cấu trúc dữ liệu & thuật toán, Đỗ Xuân Lôi, NXB Đại học quốc gia Hà Nội, 2010.

[3]. Trần Thông Quế, *Cấu trúc dữ liệu và thuật toán (phân tích và cài đặt trên C/C++)*, NXB Thông tin và truyền thông, 2018

[4]. Robert Sedgewick, *Cẩm nang thuật toán*, NXB Khoa học kỹ thuật, 2004 .

[5]. PGS.TS Hoàng Nghĩa Tý, *Cấu trúc dữ liệu và thuật toán*, NXB xây dựng, 2014