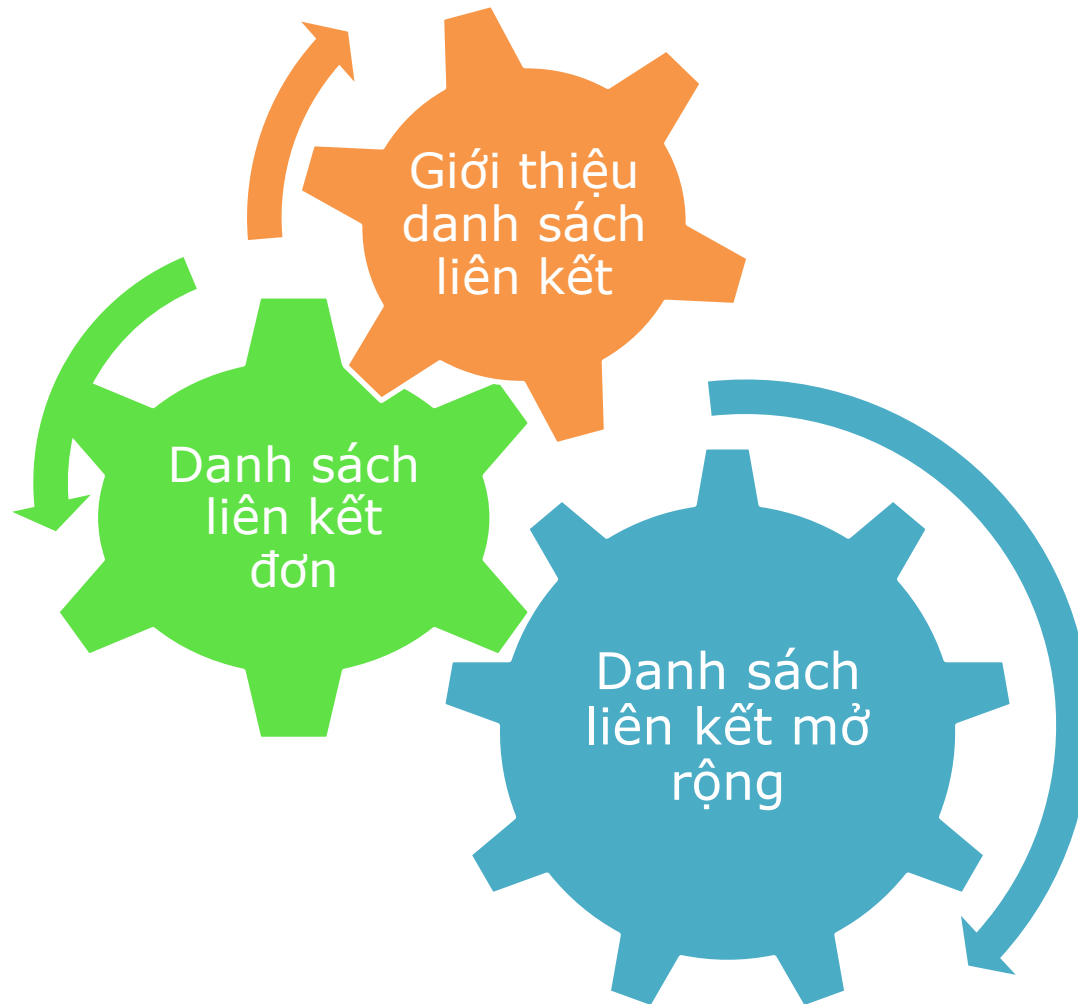

Chương 4

Danh sách liên kết



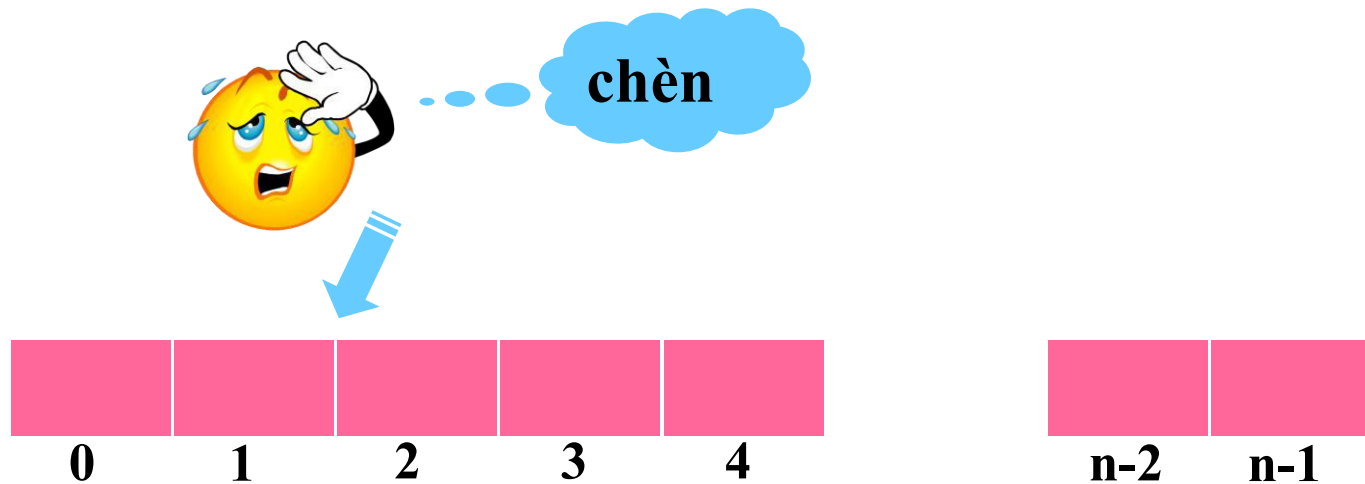
Nội dung



4.1 Giới thiệu danh sách liên kết

□ Mảng 1 chiều

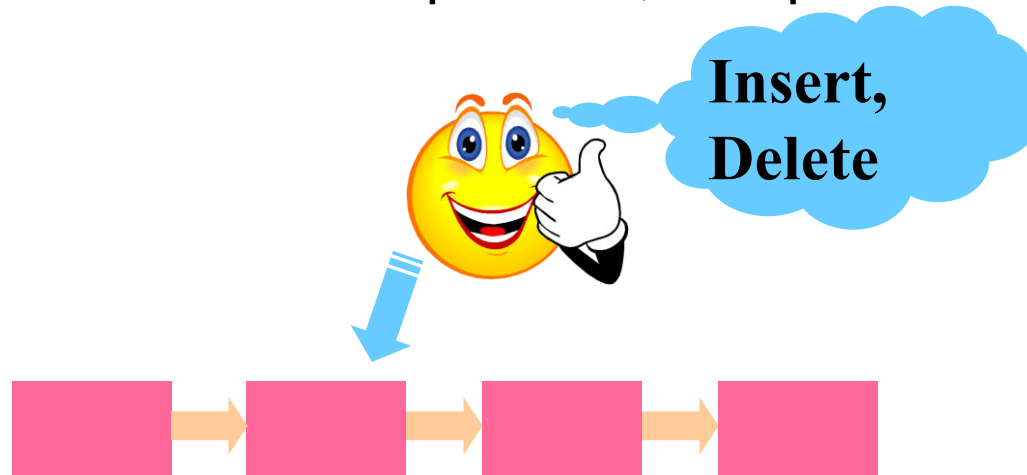
- Kích thước cố định (fixed size)
- Chèn 1 phần tử vào mảng rất khó
- Các phần tử tuần tự theo chỉ số 0, 1, ... $n-1$
- Truy cập ngẫu nhiên (random access)



4.1 Giới thiệu danh sách liên kết

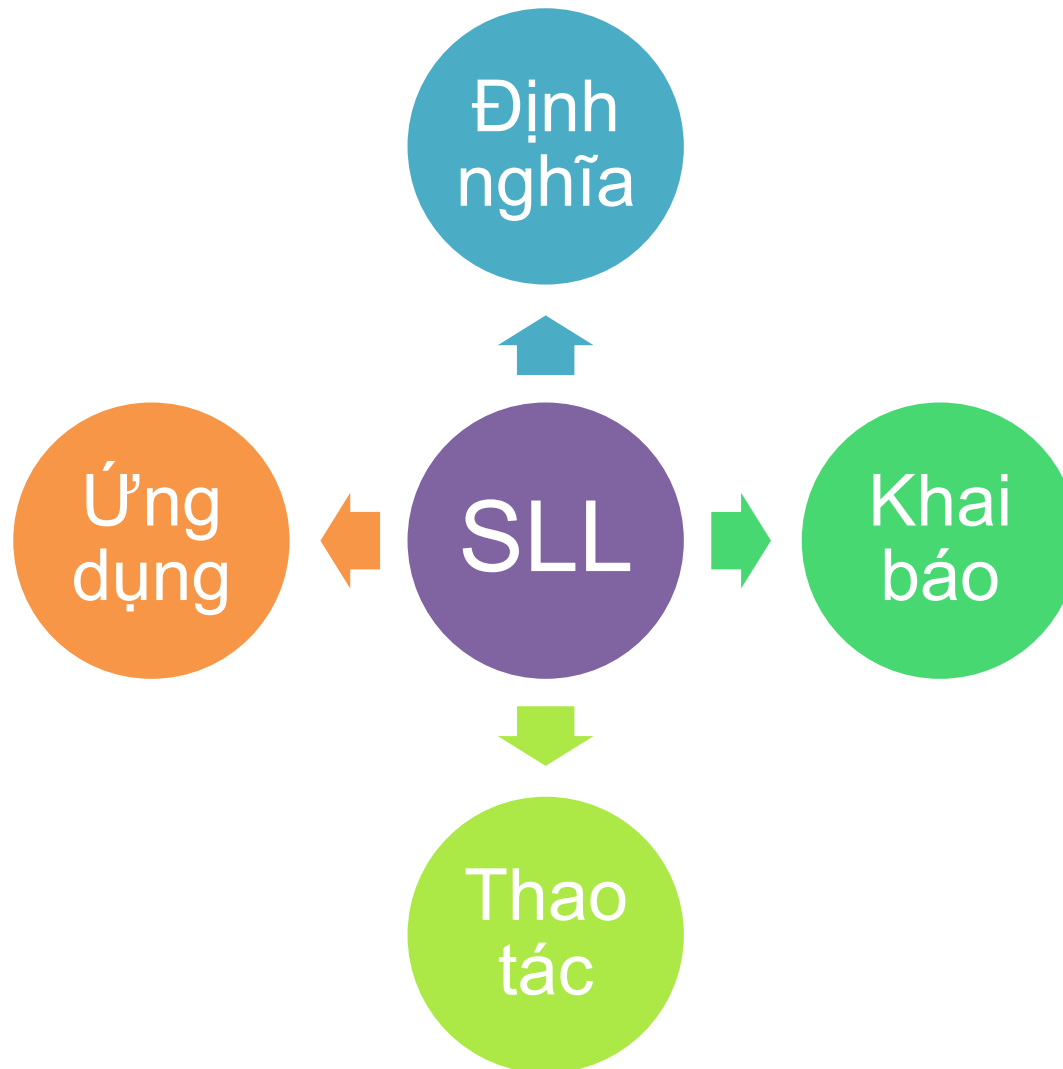
□ Danh sách liên kết

- Cấp phát động lúc chạy chương trình
- Các phần tử nằm rải rác ở nhiều nơi trong bộ nhớ
- Kích thước danh sách chỉ bị giới hạn do RAM
- Thao tác chèn thêm phần tử, xóa phần tử đơn giản



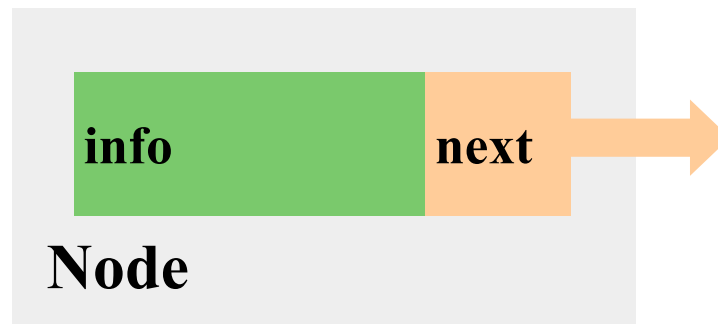
Singly Linked List

4.2 Danh sách liên kết đơn - SLL



4.2.1 SLL - định nghĩa

- ❑ DSLK đơn là chuỗi các node, được tổ chức theo thứ tự tuyến tính
- ❑ Mỗi node gồm 2 phần:
 - Phần Data, information => data
 - Phần link hay con trỏ trỏ đến node kế tiếp => next



SLL – Ôn pointer

□ Nhắc lại pointer

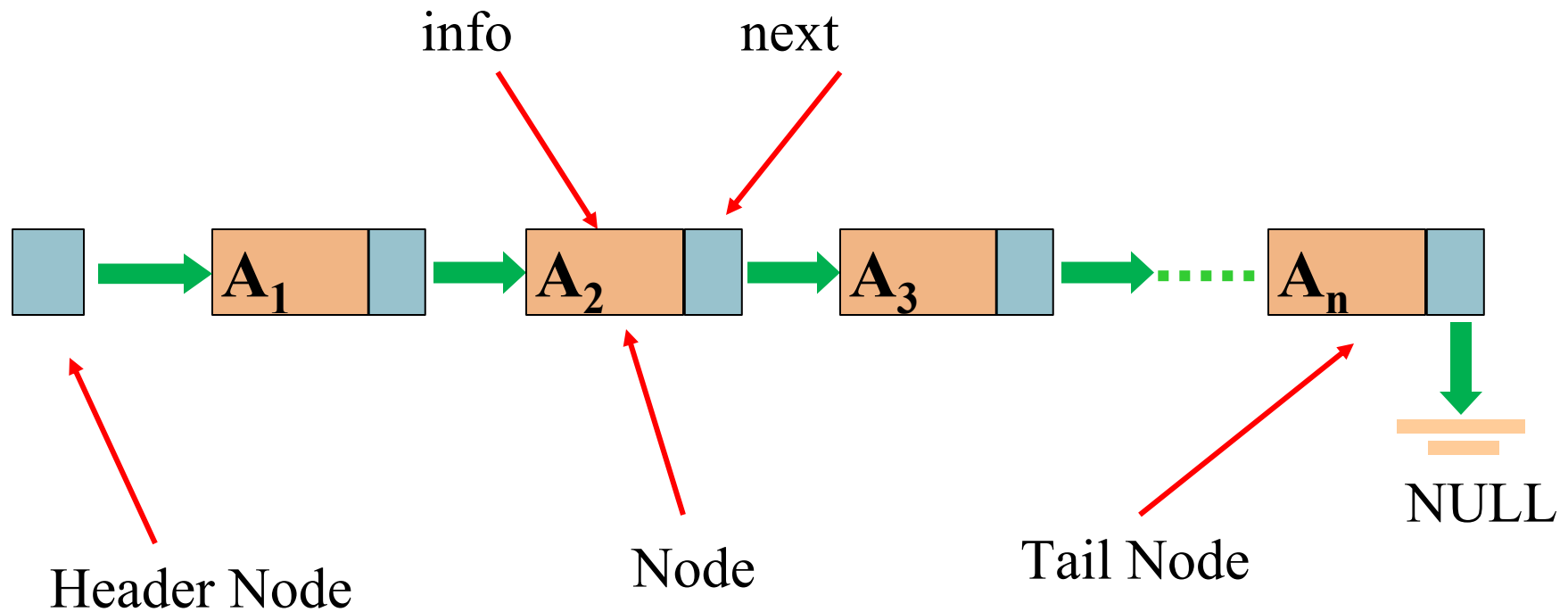
`int i, *pi;` **i** **1FF0** **?** **pi** **1FF4** **?**

`pi = &i;` **i** **1FF0** **?** **pi** **1FF4** **1FF0**

`i = 10 or *pi = 10` **i** **1FF0** **10** **pi** **1FF4** **1FF0**

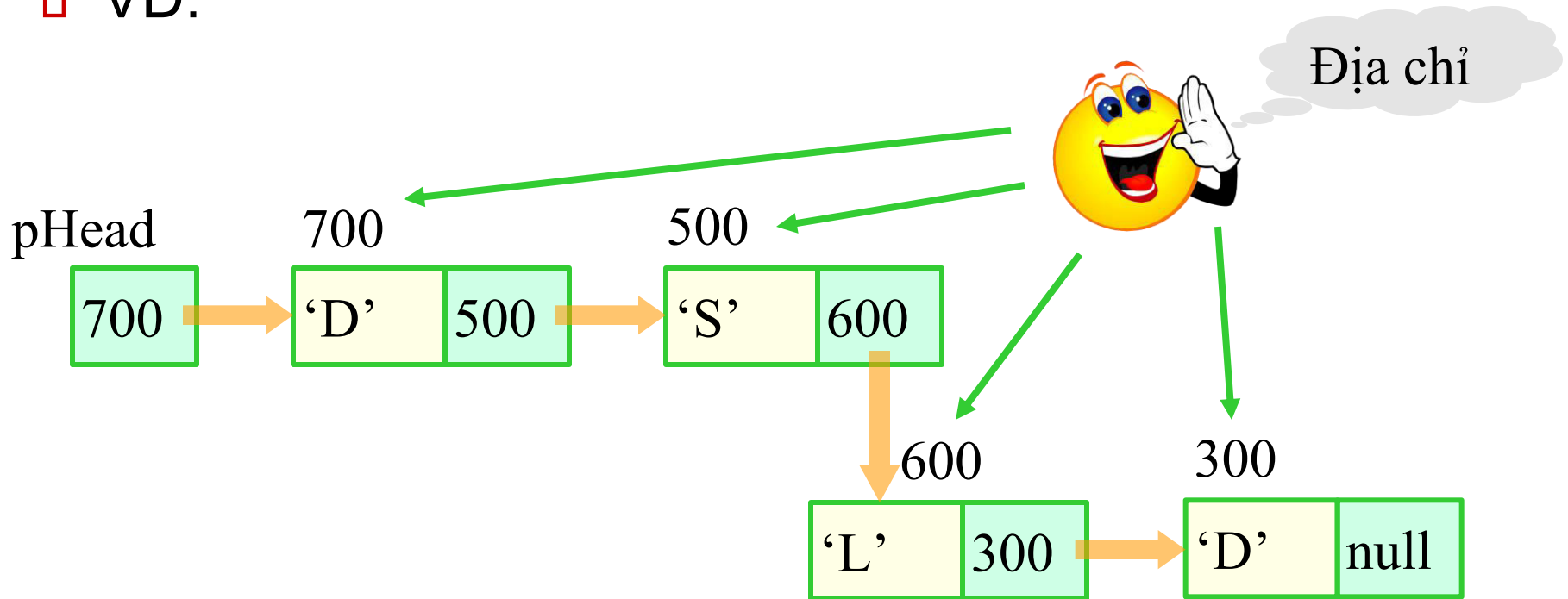
SLL – Minh hoạ

□ Mô tả DSLK



SLL – Minh họa

VD:



4.2.2 SLL – Khai báo

□ Khai báo DSLK - DataType

- Kiểu dữ liệu định nghĩa trước
- Chứa dữ liệu, thông tin của từng node



```
typedef struct
{
    char    Ten[30];
    char    MaSv[10];
    int     Gioitinh;
    float   diem;
} SinhVien;
```

4.2.2 SLL – Khai báo

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

Cấu trúc
node

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

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

4.2.2 SLL – Khai báo

□ Khai báo và khởi tạo danh sách liên kết đơn

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

```
Node *pHead;
pHead = NULL;
```

Phần minh họa
sẽ dùng
DataType là int

pHead quản lý danh sách

Khởi tạo dslk

4.2.3. SLL – Các thao tác

□ Các thao tác cơ bản

- init
- isEmpty
- printList



Phần minh
họa sẽ dùng
DataType là
int

□ Bổ sung 1 phần tử mới vào danh sách

- insertFirst
- insertAfter
- insertLast
- insertBefore

□ Loại bỏ 1 phần tử khỏi danh sách

- deleteFirst
- deleteAfter
- deleteNode
- deleteLast
- deleteBefore

□ Các thao tác khác

- searchValue
- sortList

4.2.3 SLL – Các thao tác

□ **init:** Khởi tạo danh sách

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

4.2.3 SLL - Các thao tác

□ **isEmpty**: kiểm tra danh sách rỗng

```
1. int isEmpty (Node* &pHead)
2. {
3.     if (pHead == NULL)
4.         return 1;
5.     else
6.         return 0;
7. }
```


4.2.3 CLL – Các thao tác cơ bản

□ **printList** - In (toàn bộ) danh sách

- Duyệt từ đầu danh sách, in nội dung của nút đầu
- Thực hiện tương tự với các nút tiếp theo
- Khi nào đến cuối danh sách thì dừng

4.2.3 CLL – 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;           //chuyen nut sau
8.     } while (p != NULL) ;
9. }
```

4.2.3 SLL – 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.     return new_node;
8. }
```

4.2.3 SLL – Các thao tác

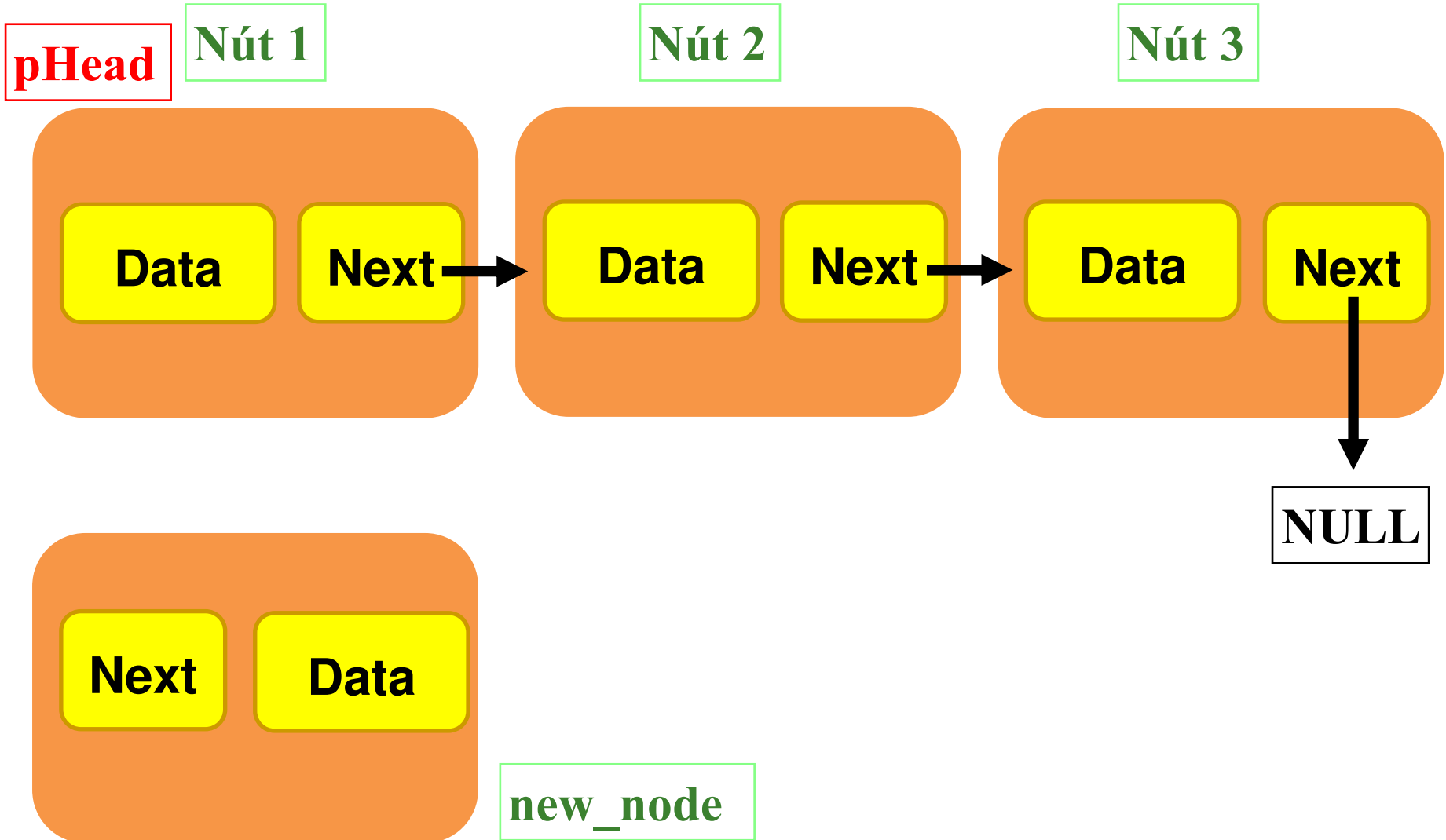
□ **addHead**: Kết nạp phần tử đầu tiên vào danh sách

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

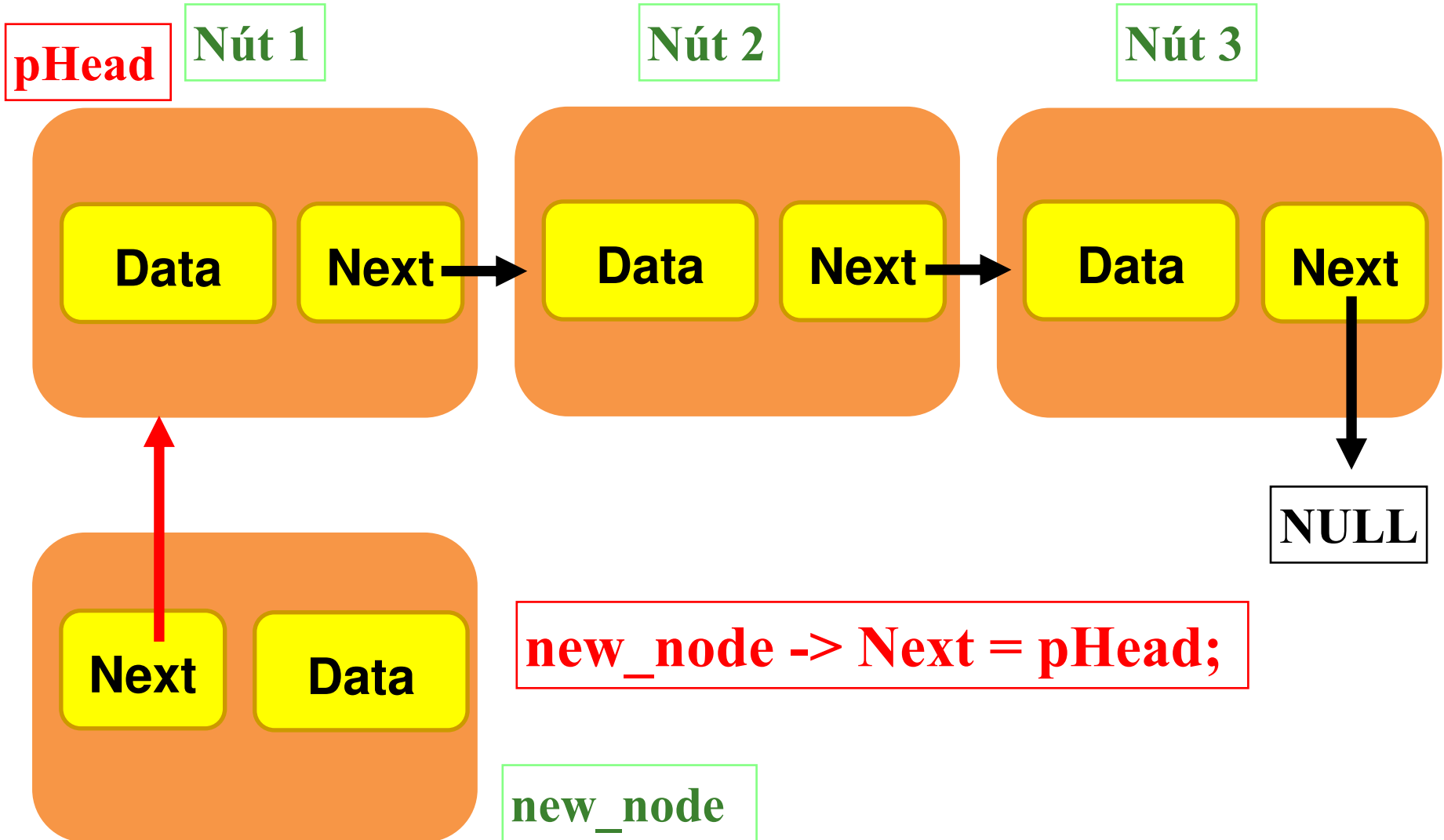
4.2.3 SLL – Các thao tác

- **insertFirst**: Thêm nút có nội dung x vào đầu danh sách
 - Tạo nút mới
 - Nếu danh sách trống: Gọi hàm **addHead**
 - Nếu danh sách không trống: Chèn vào trước nút đầu tiên.

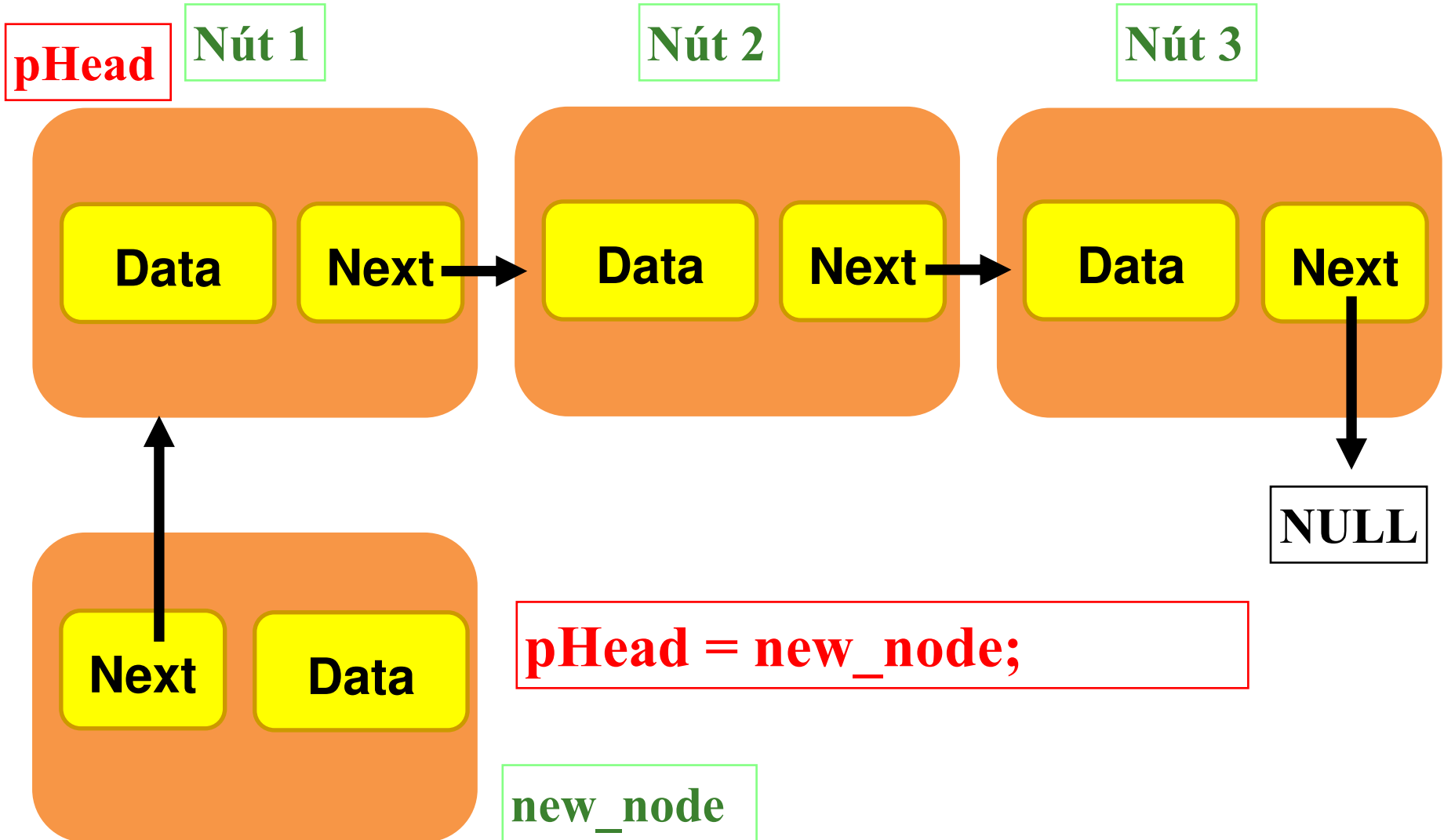
4.3.2 DLL - Bổ sung vào đầu ds



4.3.2 DLL - Bổ sung vào đầu ds



4.3.2 DLL - Bổ sung vào đầu ds



4.2.3 SLL – Các thao tác

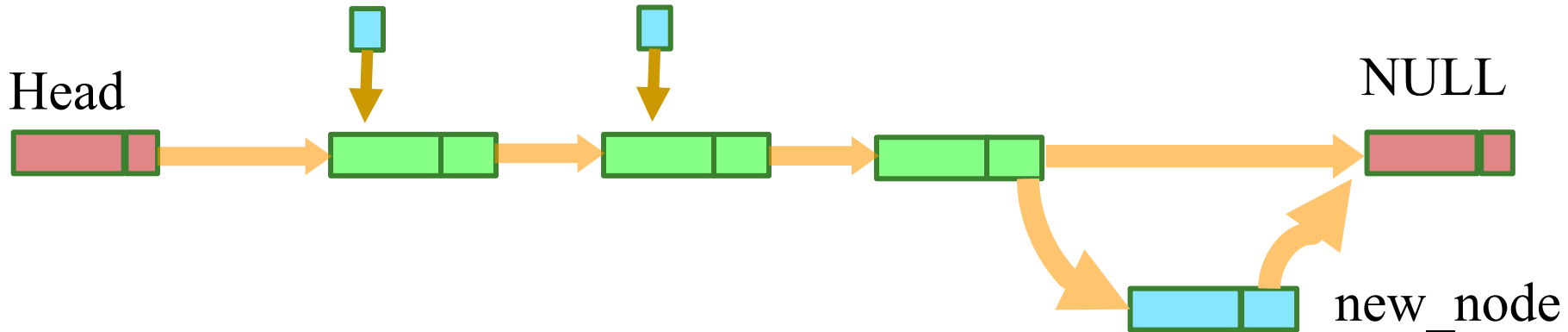
□ **insertFirst**: Thêm nút có nội dung x vào đầu danh sách

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

4.2.3 SLL – Các thao tác

Bổ sung vào cuối danh sách InsertLast

Nếu danh sách không trống: Head \neq NULL



```
while (p -> next  $\neq$  NULL)
    p = p->next;
```

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

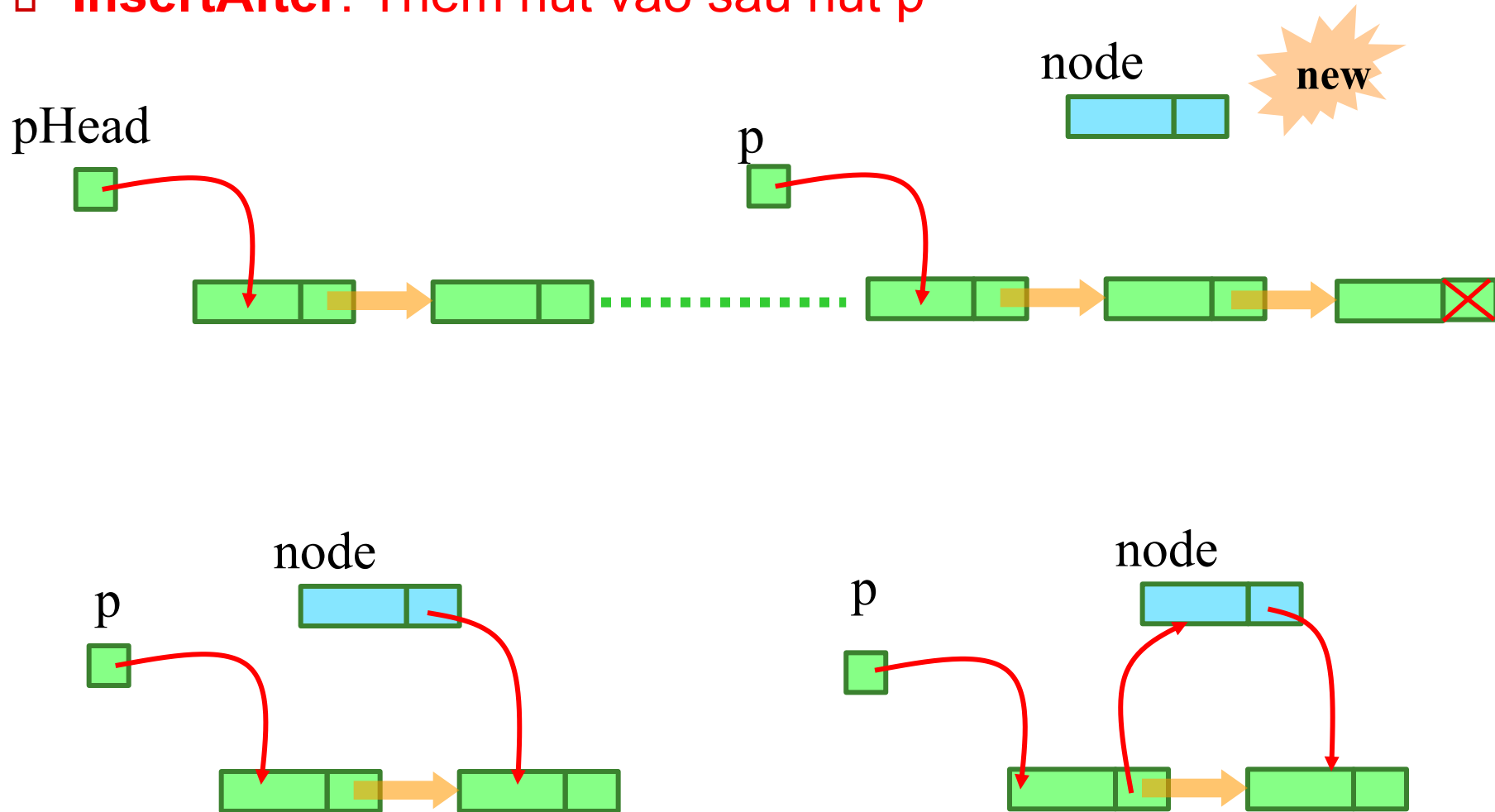
4.2.3 SLL – Các thao tác

□ **insertLast**: Thêm node có nội dung x vào cuối danh sách

```
1. void insertLast (Node* &Head, int x)
2. { Node *new_node;
3.   new_node = createNode (x) ;
4.   if (pHead == NULL)   addHead (pHead, x) ;
5.   else
6.   {   Node *p;   p = pHead;
7.       while (p -> next != NULL)
8.           p = p->next;
9.       p->next = new_node;
10.  }
```

4.2.3 SLL – Các thao tác

□ **InsertAfter:** Thêm nút vào sau nút p



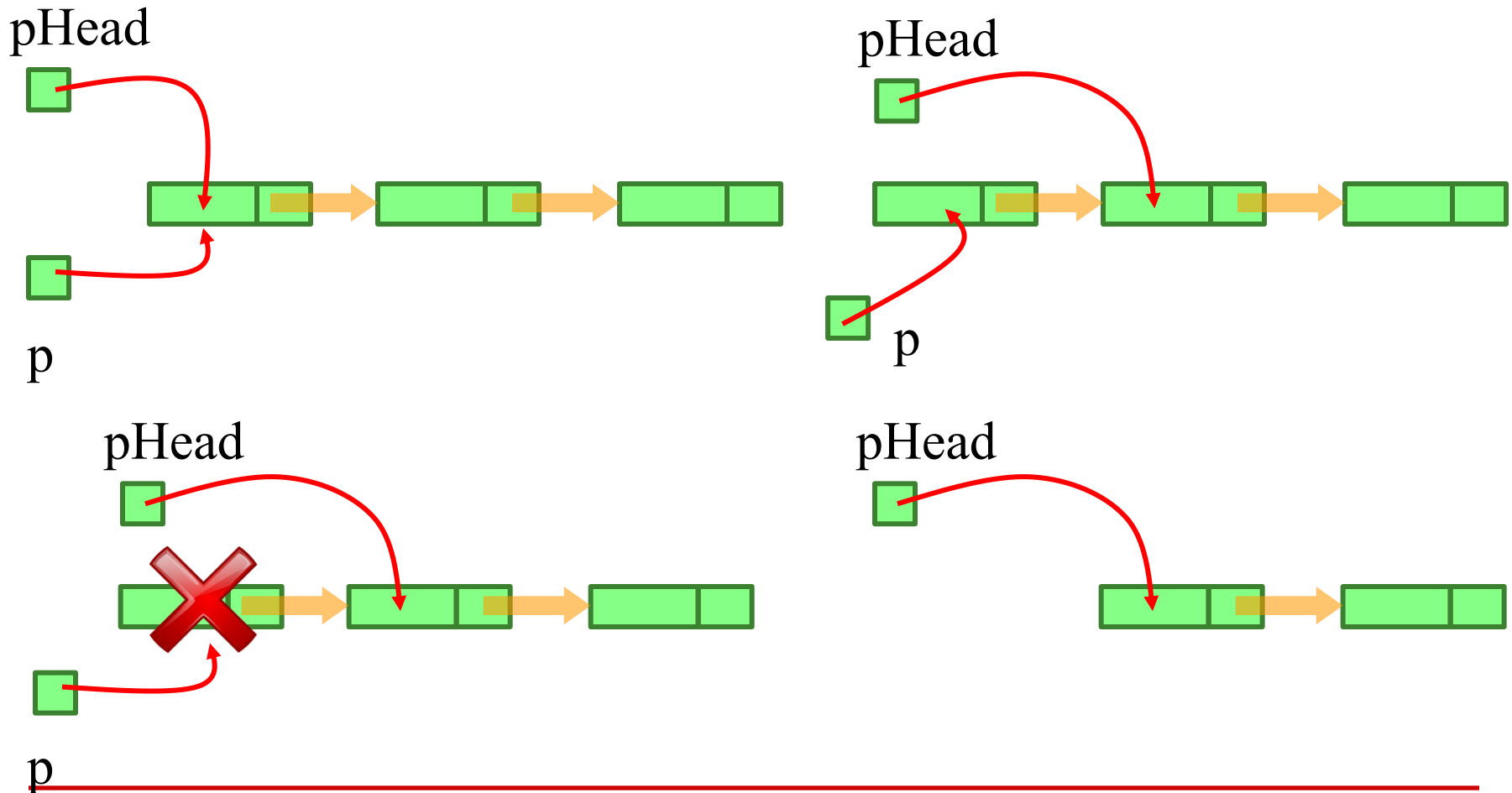
4.2.3 SLL – Các thao tác

□ **InsertAfter**: thêm node có nội dung x sau node p

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

4.2.3 SLL – Các thao tác

□ **deleteFirst:** Xóa bỏ phần tử ở đầu danh sách



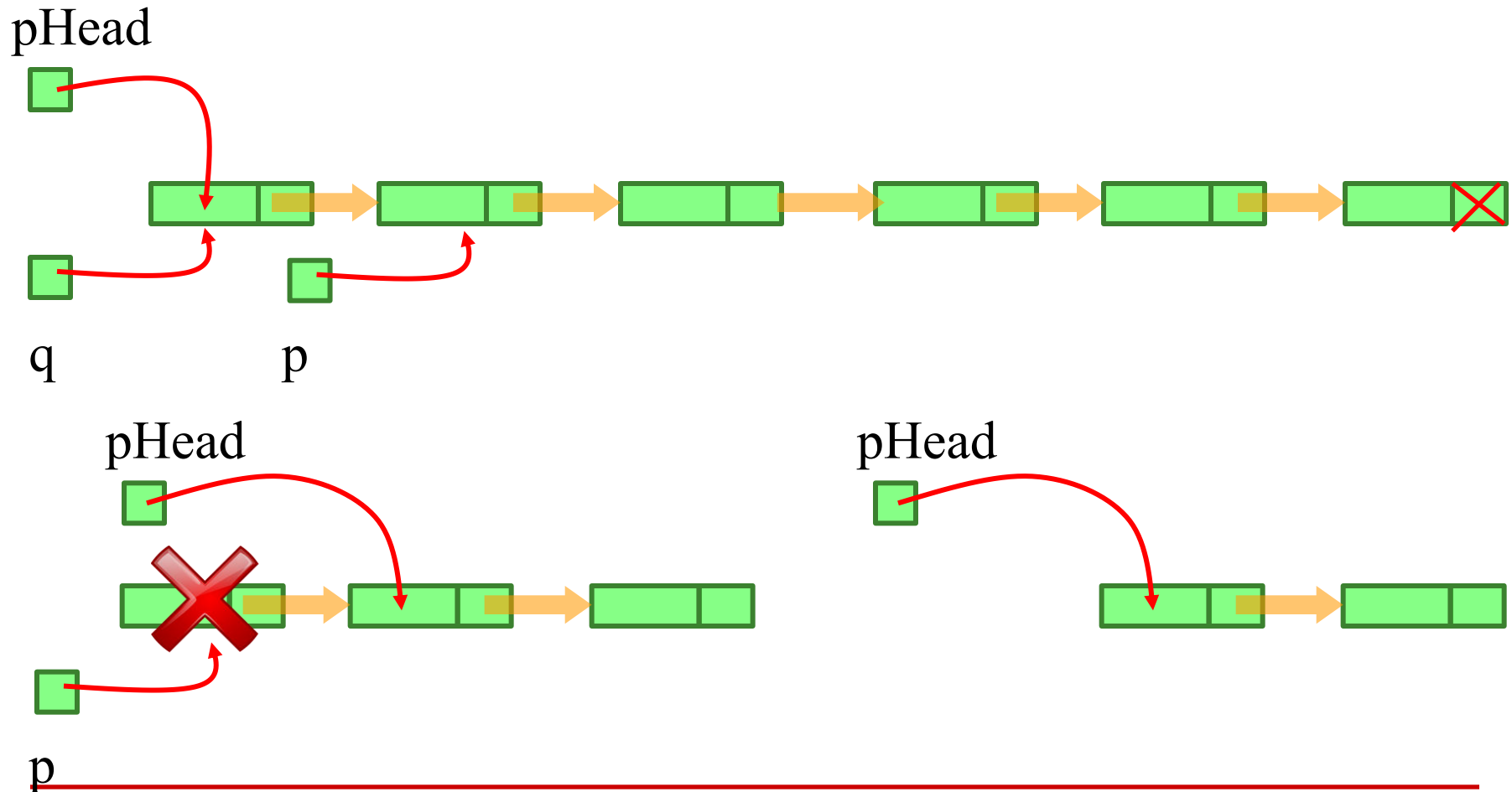
4.2.3 SLL – Các thao tác

□ **deleteFirst**: loại bỏ node đầu tiên của danh sách

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

4.2.3 SLL – Các thao tác

□ **deleteLast**: Loại bỏ phần tử cuối danh sách



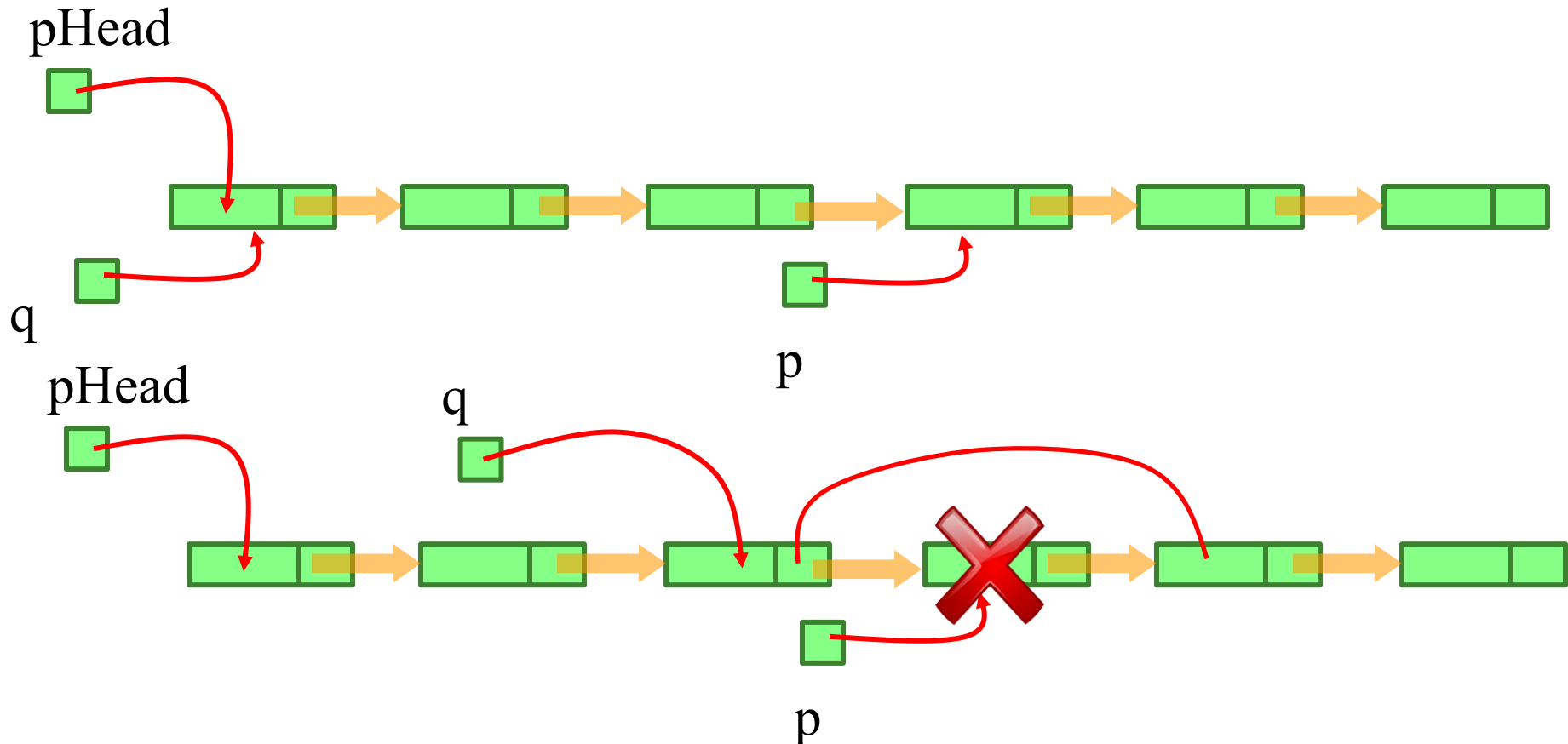
4.2.3 SLL – Các thao tác

❑ **deleteLast**: xoá node cuối trong danh sách

```
1. void DeleteLast (Node* &Head)
2. {   if (Head == NULL)
3.     cout<<"Danh sach trong!";
4.     else
5.     { Node *p = pHead;
6.         while (p->next != NULL)
7.             p = p->next;
8.         Node *q = pHead;
9.         while (q->next != p)
10.            q = q->next;
11.         q->next = NULL;
12.         delete p;      p = NULL;
13. }
```

4.2.3 SLL – Các thao tác

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



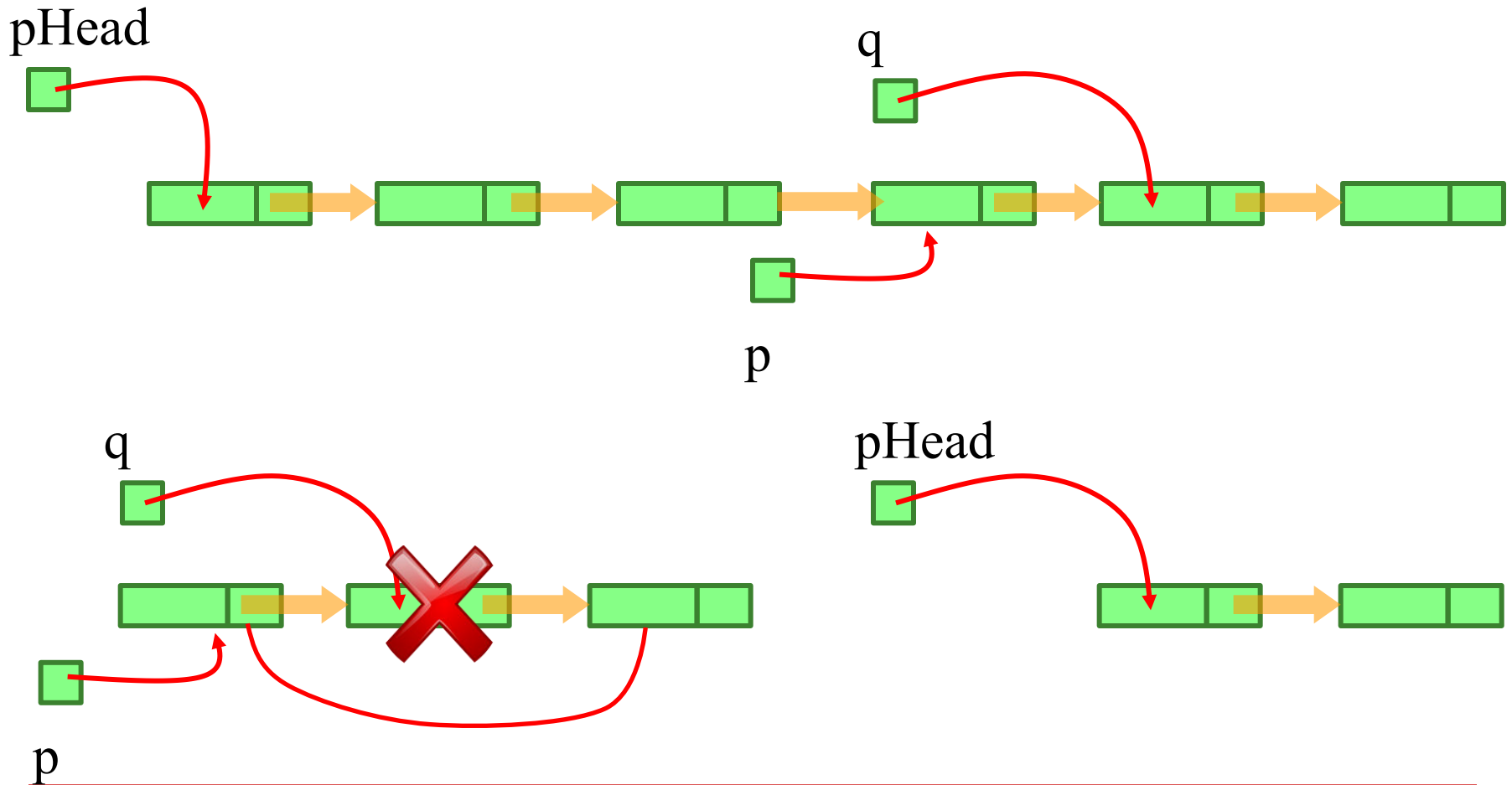
4.2.3 SLL – Các thao tác

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

```
1. void deleteNode (Node* &pHead, Node *p)
2. {   if (pHead == NULL)
3.     cout<<"Danh sach trong!";
4.     else
5.     {   Node *q= pHead;
6.         while (q->next != p)
7.             q = q->next;
8.         q->next = p->next;
9.         delete p;   p = NULL;
10.    }
11. }
```

4.2.3 SLL – Các thao tác

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



4.2.3 SLL – Các thao tác

□ **deleteAfter**: xoá node sau node p trong danh sách

```
1. void deleteAfter(Node* &Head, Node* &p)
2. {   Node* q;
3.     if (p->next == NULL)
4.         cout<<"Khong the xoa nut sau p!";
5.     else
6.     {
7.         q = p->next;
8.         p->next = q->next;
9.         delete q;           q = NULL;
10.    }
11. }
```

4.2.3 SLL – Các thao tác

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

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

4.2.3 SLL – Các thao tác

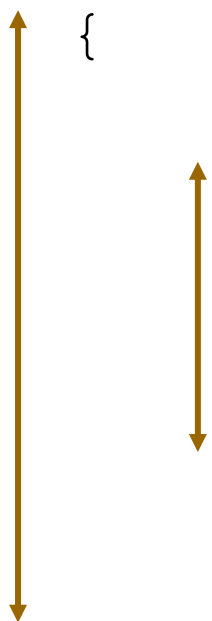
□ **searchValue**: Tìm kiếm phần tử x trong danh sách

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

4.2.3 SLL – Các thao tác

□ Sắp xếp ds theo thứ tự tăng dần, dùng Selection Sort

```
1. void sortList(Node* &Head)
2. {   Node *q, *min, *p = Head;
3.     while (p!=NULL)
4.     {   min = p; q = p -> next;
5.         while (q!=NULL)
6.         {   if (q->info < min->info)
7.             min = q;
8.             q = q->next;
9.         }
10.        swap(p->info, min->info);
11.        p = p->next;
12.    }
13. }
```



Câu hỏi củng cố bài

1. 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 (q -> next != p)

q = q -> next;

B. q = pHead;

while (q != p)

q = q -> next;

C. q = NULL;

D. q != NULL;

Câu hỏi củng cố bài

2. Cho hai danh sách nối đơn có nhiều hơn một phần tử được quản lý bởi con trỏ pHead, trường info chứa số nguyên dương. Chọn đoạn code đếm số phần tử chia hết cho 5 trong danh sách?

A.

```
int dem=0;
NodePtr p;
p = pHead;
while (p!=NULL) {
if (p->info % 5 == 0) dem++;
p = p->next; }
return dem;
```

B.

```
int dem=0;
NodePtr p;
p = pHead;
while (p!=NULL) {
if (p->info % 5 == 0) dem;
p = p->next; }
return dem;
```

C.

```
int dem=0;
NodePtr p;
p = pHead;
while (p!=NULL) {
if (p->info %5 !=0) dem++;
p = p->next; }
return dem;
```

D.

```
int dem=1;
NodePtr p;
p = pHead;
while (p!=NULL) {
if (p->info %5 != 0) dem++;
p = p->next; }
return dem;
```

Câu hỏi củng cố bài

3. Nếu có một con trỏ `p` thuộc kiểu `NodePtr` trỏ vào một nút hợp lệ trong danh sách, các lệnh nào sau đây sẽ thực hiện loại bỏ nút sau `p` trong danh sách?

A. `p->next = p->next->next;`

`delete p -> next;`

B. `tmp = p -> next;`

`p -> next = tmp -> next;`

`delete tmp;`

C. `tmp = p->next->next;`

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

`delete tmp;`

D. `tmp = p -> next;`

`tmp -> next = p -> next;`

`delete tmp;`

Câu hỏi củng cố bài

4. Cho danh sách nối đơn, p là con trỏ trỏ vào đầu danh sách có nhiều hơn một phần tử. Đoạn lệnh dưới đây thực hiện chèn nút q vào cuối danh sách nói trên.

```
NodePtr temp;
```

```
temp = p;
```

```
while (**A**)
```

```
    **B**;
```

```
temp->next = q;
```

Câu lệnh nào sẽ điền vào vị trí A?

- A. temp ->next != NULL
- B. temp ->next != p
- C. temp ->next != q
- D. temp != NULL

Câu hỏi củng cố bài

5. Cho danh sách nối đơn, p là con trỏ trỏ vào đầu danh sách có nhiều hơn một phần tử. Đoạn lệnh dưới đây thực hiện chèn nút q vào cuối danh sách nói trên.

```
NodePtr temp;
```

```
temp = p;
```

```
while (**A**)
```

```
    **B**;
```

```
temp->next = q;
```

Câu lệnh nào sẽ điền vào vị trí B?

- A. `p = p ->next;`
- B. `temp++;`
- C. `temp = temp.next;`
- D. `temp = temp -> next;`

Câu hỏi củng cố bài

6. Cho danh sách nối đơn, p là con trỏ trỏ vào đầu danh sách có nhiều hơn một phần tử. Lệnh nào dưới đây có tác dụng di chuyển nút đầu danh sách?

A. `p++;`

B. `p = p->next;`

C. `p->next = p->next->next`

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

Tổng kết

Mô tả

Định nghĩa

Là chuỗi các nút được tổ chức theo thứ tự tuyến tính

Cấu trúc 1 nút

info next
Node

Khai báo

```
typedef struct node
{
    int info;
    struct node * next;
} NODE;
typedef NODE* NodePtr;
NodePtr pHead;
```

Các thao tác

Cơ bản

Init, IsEmpty, ShowList

Bổ sung

InsertFirst, InsertLast, InsertAfter, InsertBefore

Loại bỏ

DeleteFirst, DeleteLast, DeleteAfter,
DeleteBefore, DeleteNode, DeleteAll

Khác

Search, Sort

Danh sách liên kết đơn

SLL– Bài tập

1. Cho một danh sách nối đơn có nút đầu được trỏ bởi pHead. Trường info của các nút chứa giá trị nguyên. Viết giải thuật thực hiện các công việc sau”
 - a. Đếm số nút của danh sách
 - b. Bổ sung một nút mới với thông tin x vào làm nút thứ k trong danh sách
 - c. Loại bỏ nút có giá trị y trong danh sách
 - d. Tìm và in ra các nút chia 5 dư 2 trong danh sách

SLL– Bài tập

2. Cho một danh sách nối đơn có nút đầu được trỏ bởi pHead. Trường info của các nút chứa giá trị nguyên. Viết giải thuật thực hiện các công việc sau:

- a. Bổ sung một nút mới với thông tin x vào làm cuối danh sách
- b. Loại bỏ nút trước nút p bất kỳ trong danh sách
- c. Tìm và in ra các nút chia hết cho 7 trong danh sách

SLL– Bài tập

3. Cho một danh sách nối đơn có nút đầu được trỏ bởi pHead. Trường info của các nút chứa giá trị nguyên. Viết giải thuật thực hiện các công việc sau:

- a. Bổ sung một nút mới với thông tin x vào trước nút p bất kỳ trong danh sách
- b. Loại bỏ nút cuối danh sách
- c. Cho một số nguyên y. Tìm xem trong danh sách có nút nào mà trường info = y hay không? Nếu tìm thấy trả về địa chỉ của nút đó, nếu không tìm thấy trả về NULL.

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

