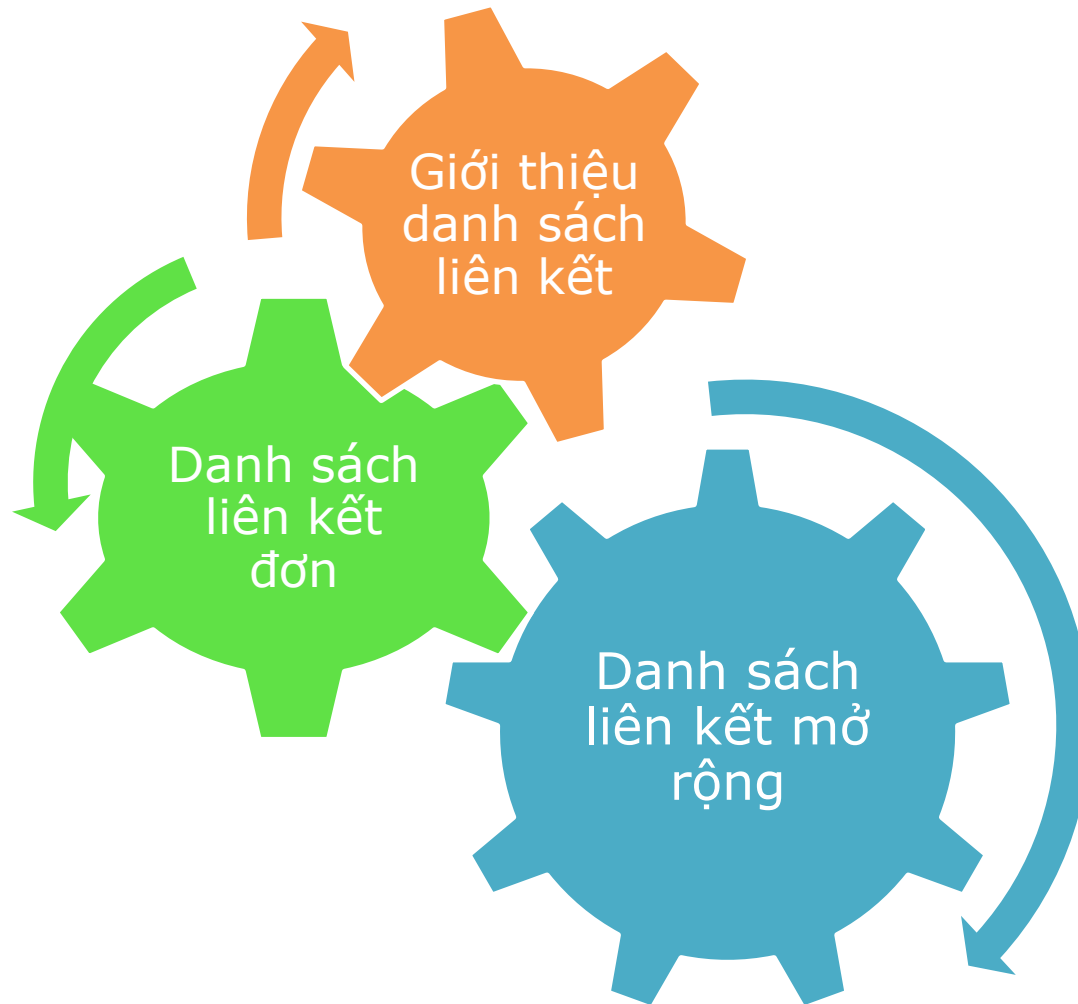

Chương 4

Danh sách liên kết



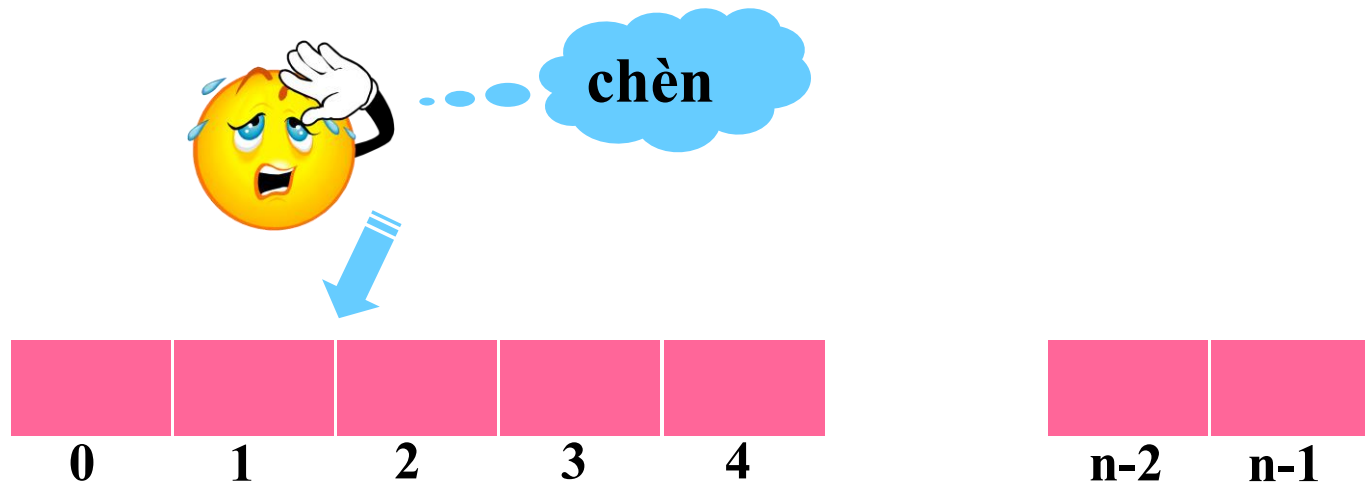
Nội dung



4.1 Giới thiệu danh sách liên kết

□ Mảng 1 chiều

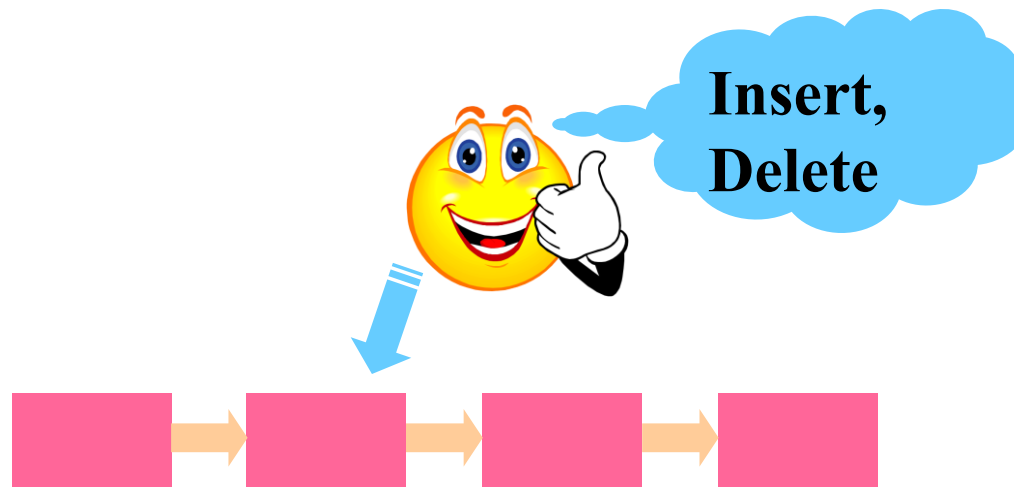
- Kích thước cố định (fixed size)
- Chèn 1 phần tử vào mảng rất khó
- Các phần tử tuần tự theo chỉ số $0 \text{ } \textcircled{R} \text{ } n-1$
- Truy cập ngẫu nhiên (random access)



4.1 Giới thiệu danh sách liên kết

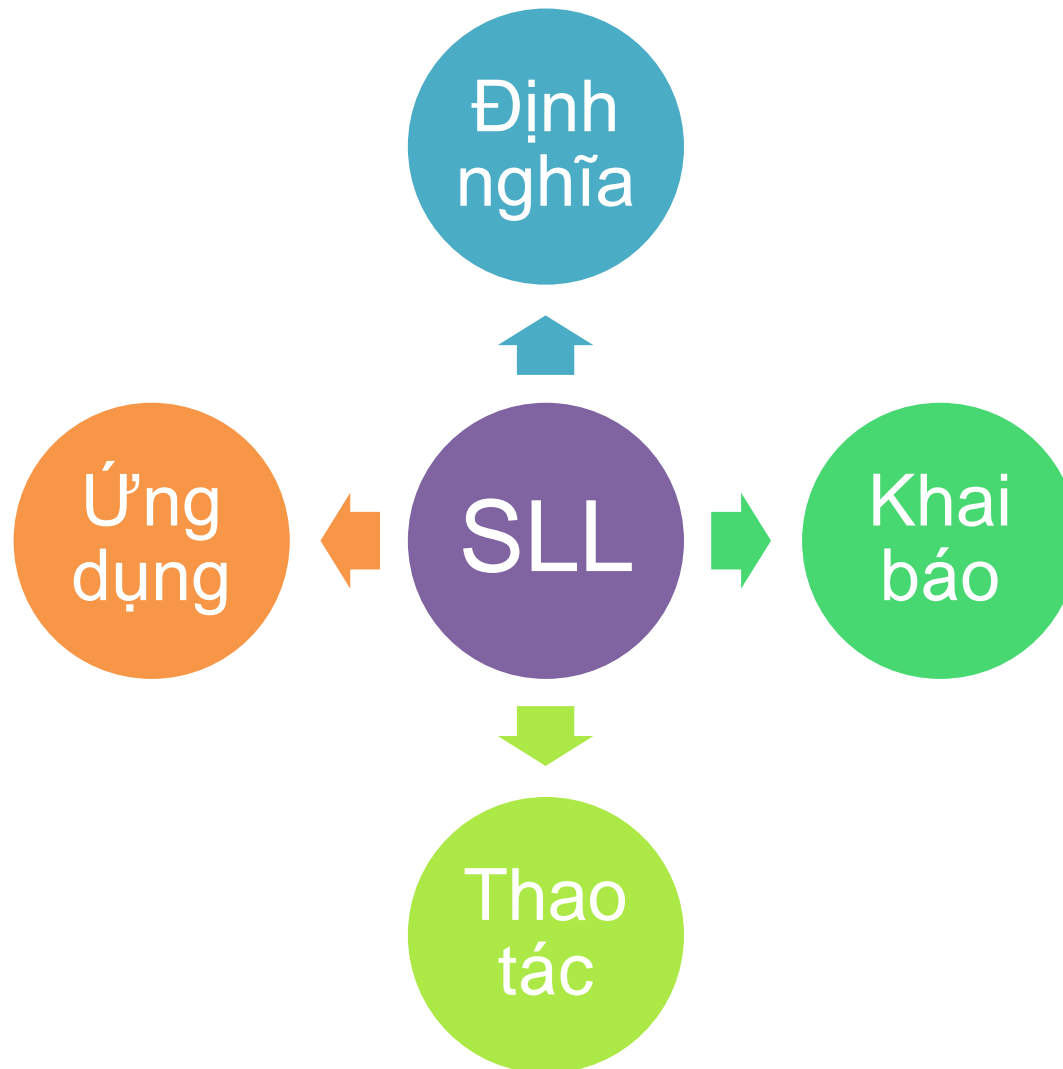
□ Danh sách liên kết

- 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 thêm xóa 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 => info
 - Phần link hay con trỏ trỏ đến node kế tiếp => next



SLL – Ôn pointer

□ Nhắc lại pointer

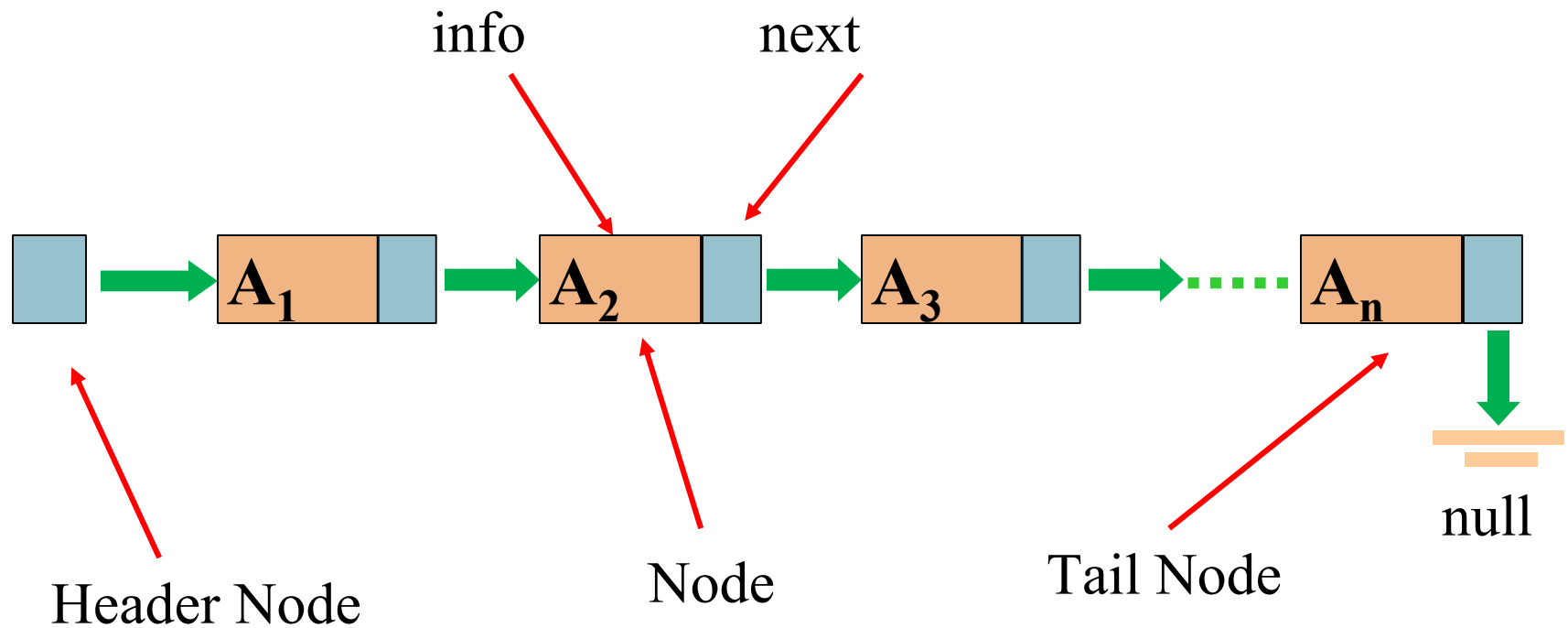
`int i, *pi;` **i** **1FF0** ? **pi** **1FF4** ?

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

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

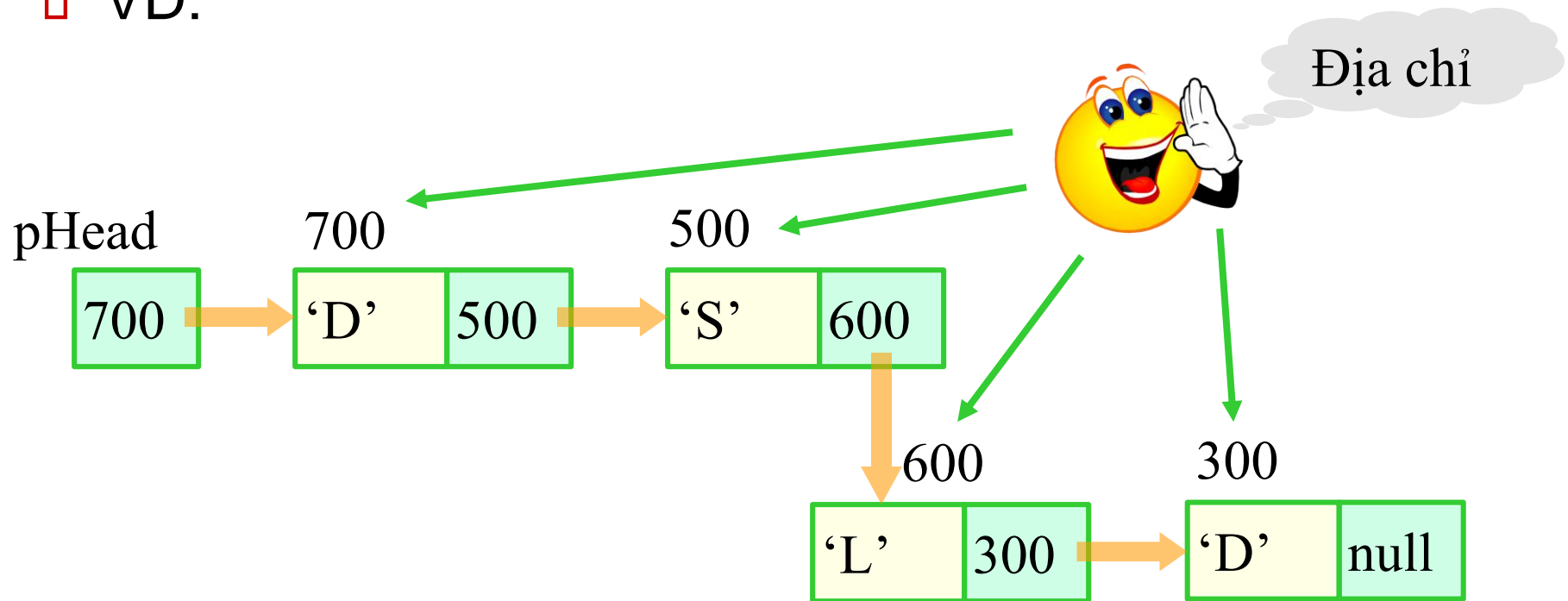
SLL – Minh hoạ

□ 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
{
    int    ngay;
    int    thang;
    int    nam;
} DateTime;
```

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

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 node
{
    DataType    info;
    struct node * next;
}NODE;
```

Cấu trúc
node


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


```
typedef struct node
{
    SinhVien    info;
    struct node * next;
}NODE;
```

4.2.2 SLL – Khai báo

□ Khai báo và khởi tạo danh sách liên kết đơn

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

NodePtr pHead;  pHead quản lý ds

pHead = NULL;  Khởi tạo dslk

4.2.3. SLL – Các thao tác

□ Các thao tác cơ bản

- Init
- IsEmpty
- ShowList



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

- Search
- Sort

4.2.3 SLL – Các thao tác

□ **Init**: khởi tạo danh sách, ban đầu chưa có phần tử

```
1. void      Init (NodePtr      &pHead)
2. {
3.     *pHead = NULL;
4. }
```


4.2.3 SLL - Các thao tác

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

```
1. int      IsEmpty (NodePtr  pHead)
2. {
3.     if (pHead == NULL)
4.         return      1;
5.     else
6.         return      0;
7. }
```

4.2.3 SLL – Các thao tác

□ **ShowList**: duyệt toàn bộ danh sách

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

```
1. void      ShowList (NodePtr      pHead)
2. {  NodePtr  p;      //p là con trỏ để duyệt
3.    p = pHead;        //duyet từ đầu danh sách
4.    while (p!=NULL) //khi chưa hết ds
5.    {
6.        ShowNode (p);    //tác động lên nút
7.        p = p->next;      //chuyển nút sau
8.    }
9. }
```

4.2.3 SLL - Các thao tác

□ ShowNode:

```
1. void      ShowNode (NodePtr q)
2. {
3.     cout<< q -> info;
4. }
```

4.2.3 SLL – Các thao tác

□ Bổ sung:

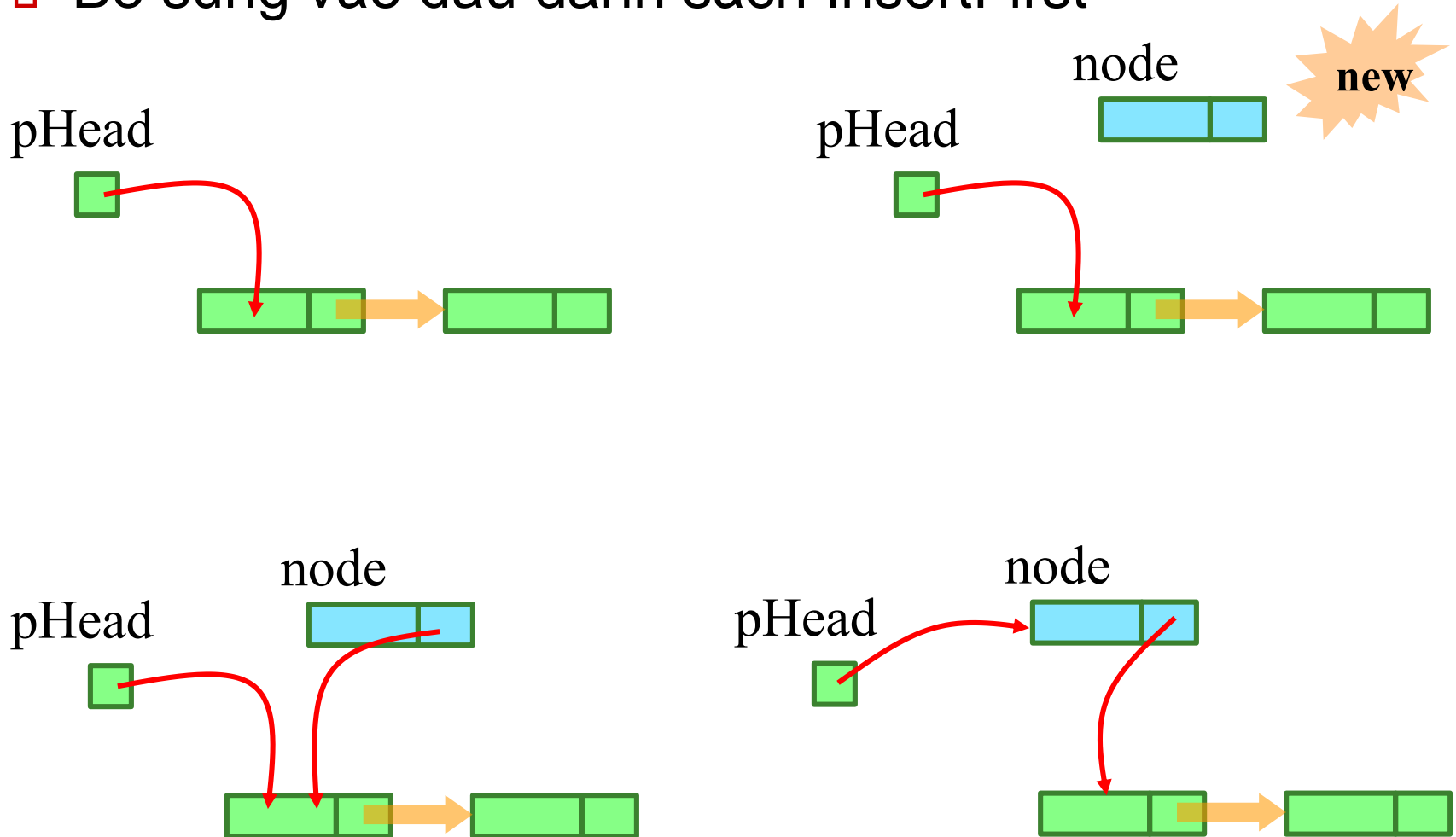
- Tạo nút mới
- Nối vào danh sách

Tạo nút mới

```
NodePtr    node;    //node là con trỏ trỏ nút mới  
node = new        NODE; //xin cấp phát địa chỉ  
node -> info = x;
```

4.2.3 SLL – Các thao tác

□ Bổ sung vào đầu danh sách InsertFirst



4.2.3 SLL – Các thao tác

□ InsertFirst: bổ sung nút có nội dung x vào đầu ds

```
1. void InsertFirst(NodePtr &pHead, int x)
2. {   NodePtr      node;
3.     node = new      NODE;
4.     node->info = x;
5.     if (pHead == NULL)
6.     {       pHead = node;
7.             pHead ->next = NULL;           }
8.     else
9.     {       node->next = pHead;
10.           pHead = node;                     }
11. }
```

4.2.3 SLL – Các thao tác

□ InsertLast: thêm node có nội dung x vào cuối ds

```
1. void InsertLast (NodePtr &pHead, int x)
2. { //tao nut moi
3.     NodePtr    node;
4.     node = new    NODE;
5.     node->info = x;
6.     node -> next =null;
7.     //noi vao danh sach
8.     if (pHead == NULL)
9.         pHead = node;
```

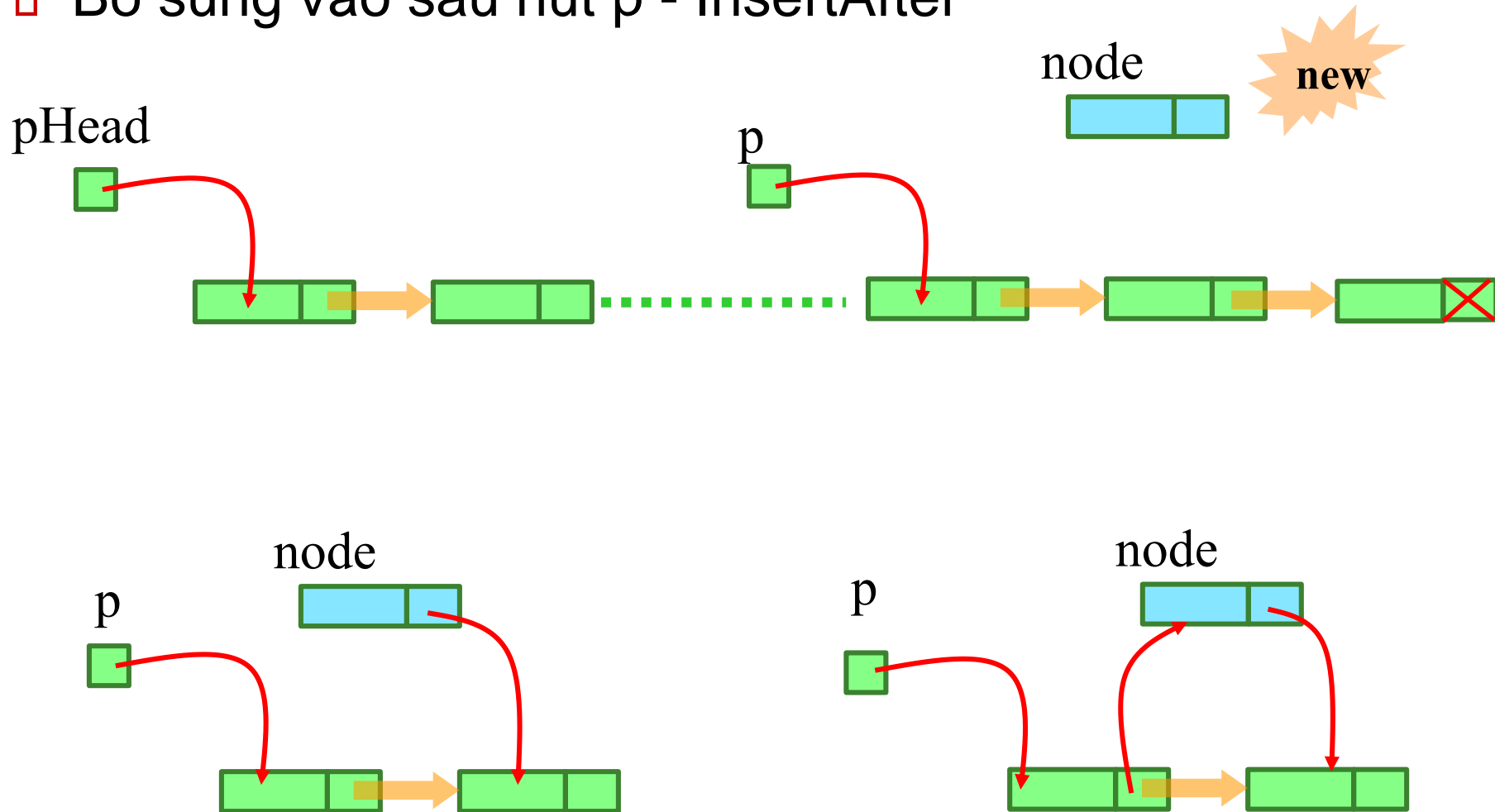
4.2.3 SLL – Các thao tác

□ InsertLast: thêm node có nội dung x vào cuối ds

```
10.  else
11.  {      // tìm đến nút cuối
12.      NodePtr    p;
13.      p = pHead;
14.      while (p -> next != NULL)
15.          p = p->next;
16.      // nối vào cuối danh sách
17.      p ->next = node;
18.  }
19. }
```


4.2.3 SLL – Các thao tác

□ Bổ sung vào sau nút p - InsertAfter



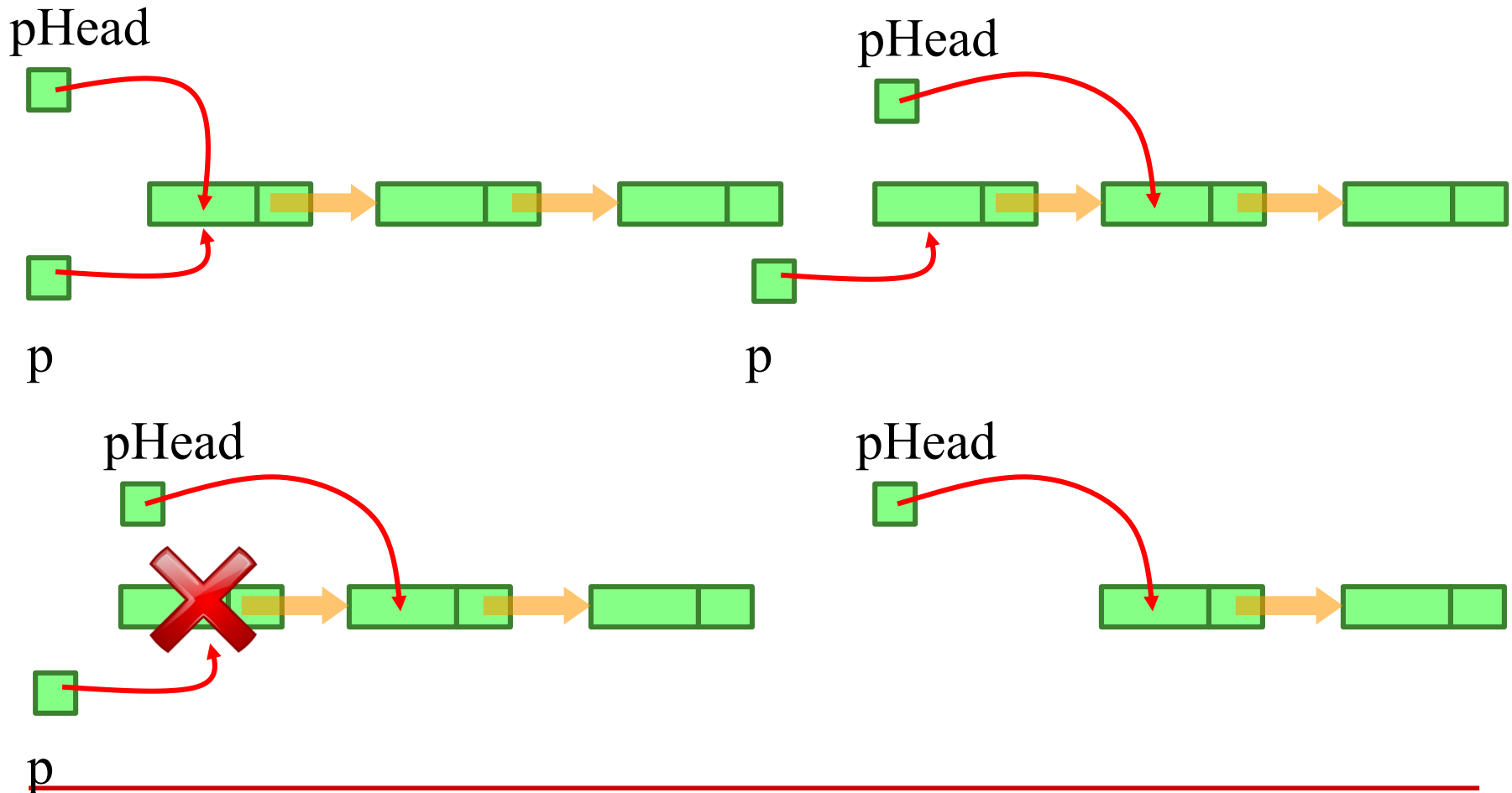
4.2.3 SLL – Các thao tác

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

```
1. void InsertAfter(NodePtr &pHead, NodePtr &p, int x)
2. {   if (p == NULL)
3.     cout<<"Cannot insert new node!";
4.     else
5.     {   NodePtr      node;
6.         node = new      NODE;
7.         node->info = x;
8.         node->next = p->next;
9.         p->next = node;
10.    }
11. }
```

4.2.3 SLL – Các thao tác

□ Loại bỏ phần tử đầu danh sách DeleteFirst



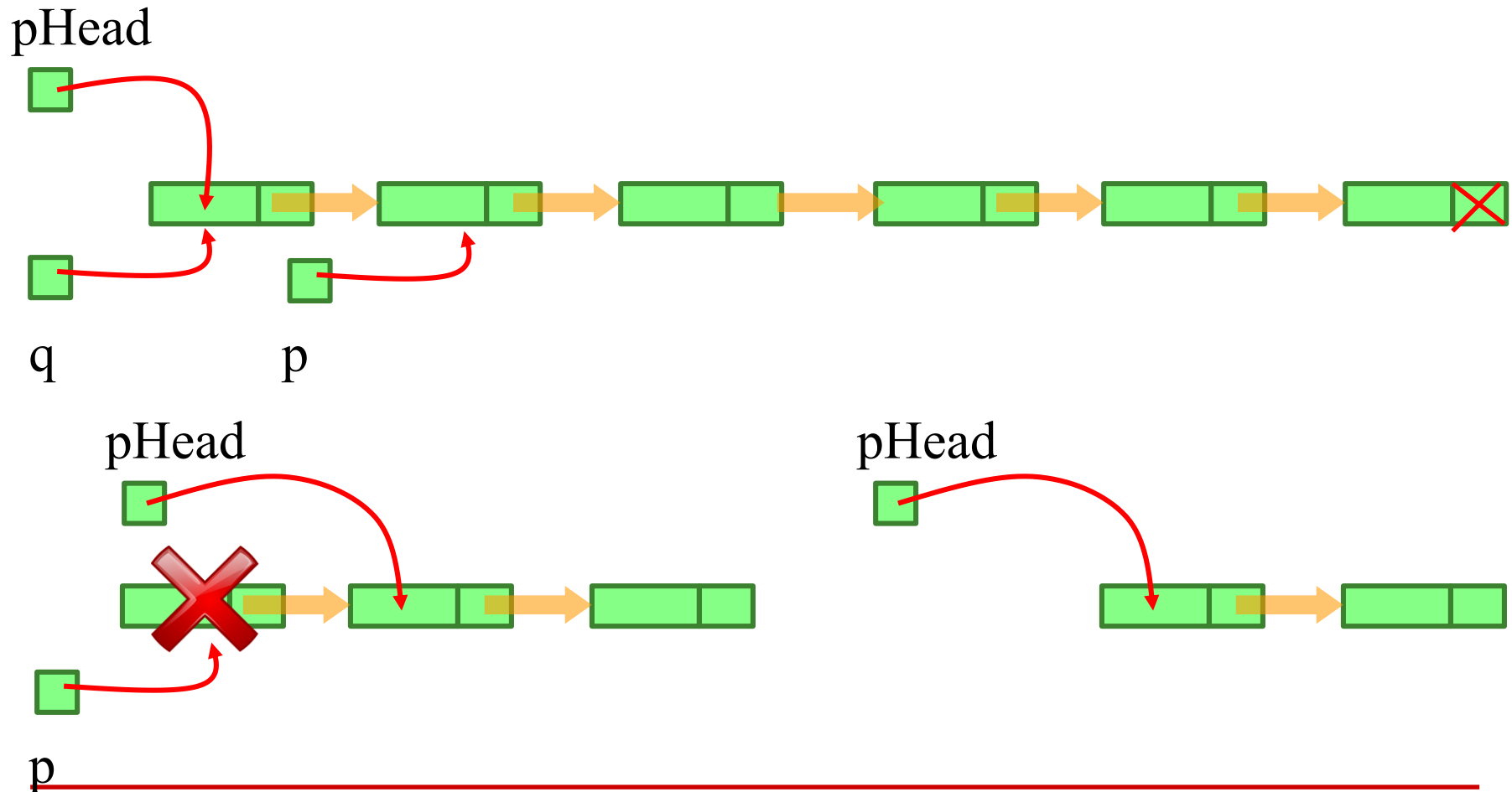
4.2.3 SLL – Các thao tác

□ DeleteFirst: loại bỏ node đầu tiên của danh sách

```
1. void DeleteFirst (NodePtr &pHead)
2. {   NodePtr      p;
3.     if (IsEmpty (pHead) )
4.         cout<<"List is empty!";
5.     else
6.     {   p = pHead;
7.         pHead = pHead->next;
8.         delete    p;
9.     }
10. }
```

4.2.3 SLL – Các thao tác

□ Loại bỏ phần tử cuối danh sách DeleteLast



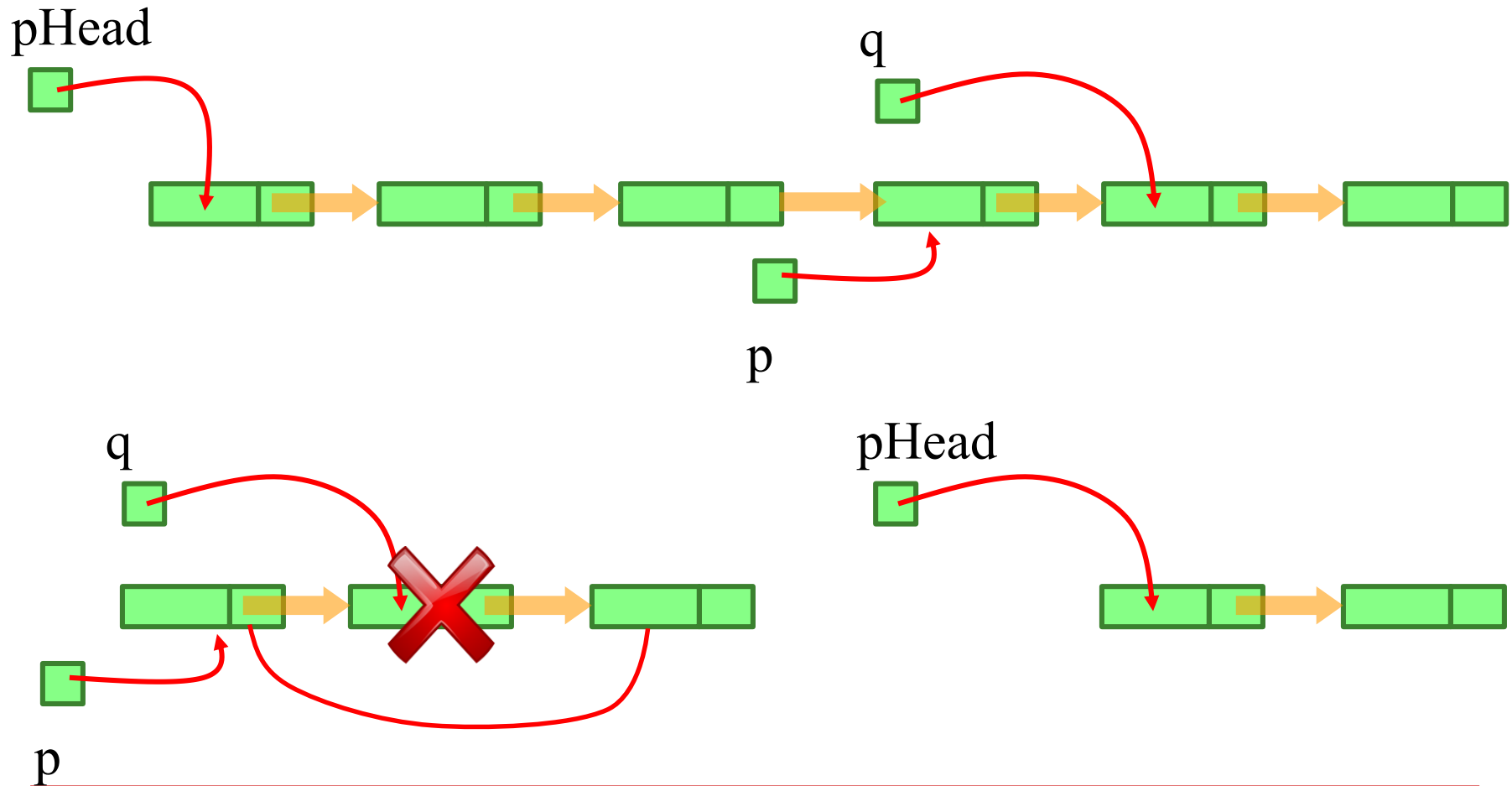
4.2.3 SLL – Các thao tác

□ DeleteLast: xoá node cuối trong danh sách

```
1. void DeleteLast (NodePtr &pHead)
2. {   if (pHead == NULL)
3.     cout<<"List is empty!";
4.     else
5.         { NodePtr  p, q; //p tìm đến nút cuối, q trước nút p
6.           p = pHead -> next; q = pHead;
7.           while (p -> next != null)
8.               {       p = p->next;
9.                       q = q->next;
10.                  }
11.           delete    p;
12.           q -> next = null;           } //end if
13. }
```

4.2.3 SLL – Các thao tác

❑ Loại bỏ phần tử sau nút p - DeleteAfter



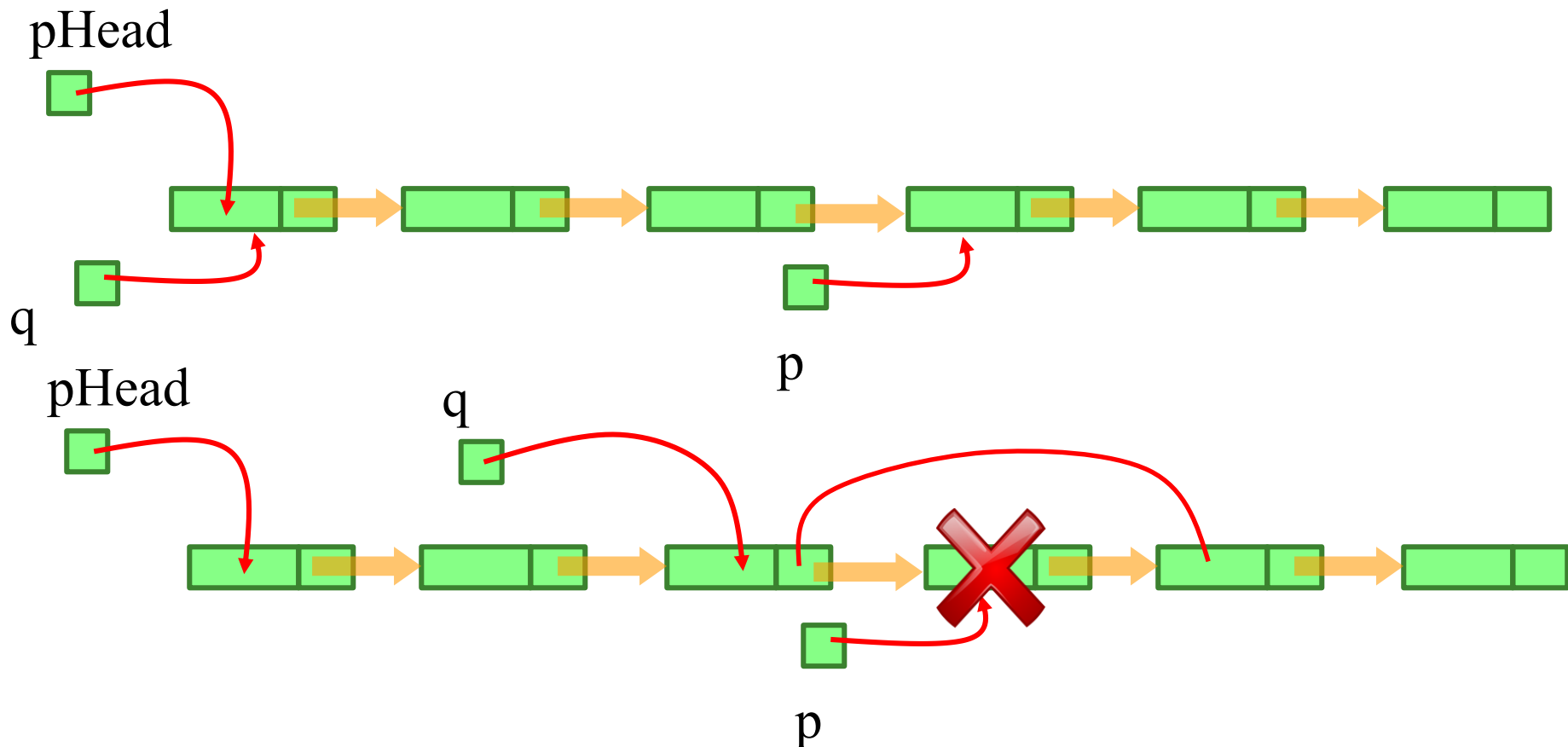
4.2.3 SLL – Các thao tác

□ DeleteAfter: xoá node sau node p trong danh sách

```
1. void DeleteAfter(NodePtr &pHead, NodePtr &p)
2. {   NodePtr      q;
3.     if (p->next == NULL)
4.         cout<<"Cannot delete node!";
5.     else
6.     {
7.         q = p->next;
8.         p->next = q->next;
9.         delete      q;
10.    }
11. }
```


4.2.3 SLL – Các thao tác

□ Loại bỏ phần tử được trỏ bởi p - DeleteNode



4.2.3 SLL – Các thao tác

□ DeleteNode: xoá node p trong danh sách

```
1. void DeleteNode (NodePtr &pHead, NodePtr &p)
2. {   if (p ==NULL)
3.         cout<<"Cannot delete node!";
4.     else
5.     {   NodePtr    q= pHead;
6.         while (q->next !=p)
7.             q=q->next;
8.         q->next = p->next;
9.         delete    p;
10.    }
11. }
```

4.2.3 SLL – Các thao tác

□ DeleteAll: xóa toàn bộ danh sách

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

4.2.3 SLL – Các thao tác

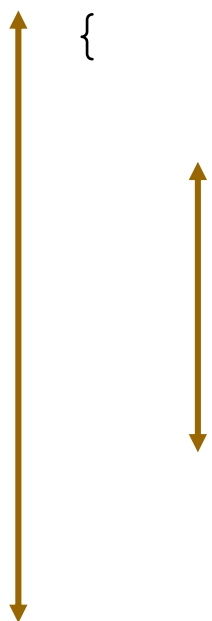
□ Search: Tìm kiếm phần tử x trong danh sách

```
1. NodePtr Search(NodePtr pHead, int x)
2. {
3.     NodePtr    p;        //p để duyệt và tìm
4.     p = pHead;           //tìm từ đầu ds
5.     while ( p != NULL    && p->info != 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 Sort(NodePtr &pHead)
2. {   NodePtr q, min, p = pHead;
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

