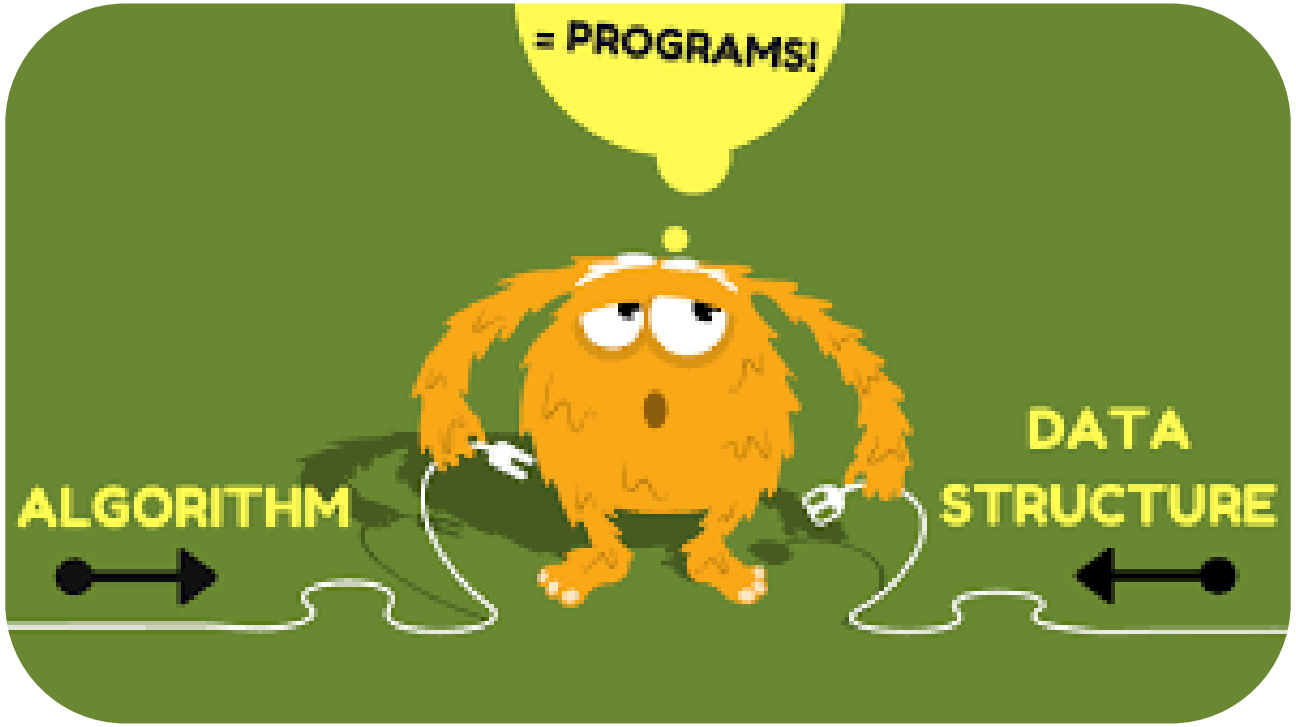
DANH SÁCH LIÊN KẾT ĐƠN



MỤC TIÊU BÀI HỌC

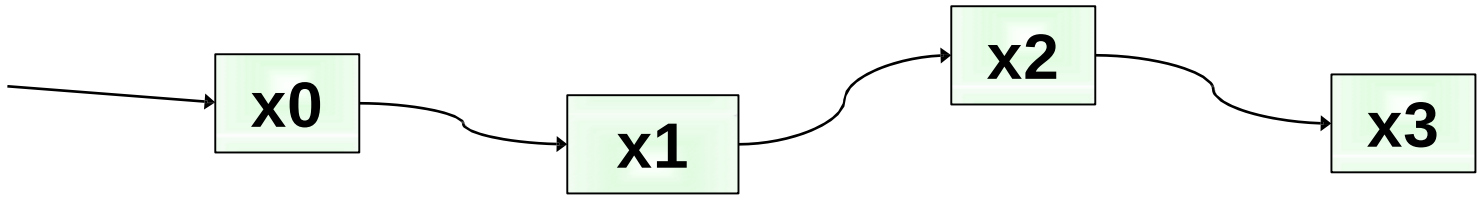
- Khái niệm danh sách liên kết đơn
- Các thao tác trên danh sách liên kết đơn

NỘI DUNG BÀI HỌC

1. Tổ chức của danh sách liên kết đơn
2. Tạo mới một phần tử trong danh sách
3. Thêm phần tử có trong danh sách
4. Hủy một phần tử trong danh sách
5. Tìm một phần tử trong danh sách

1 TỔ CHỨC CỦA DANH SÁCH LIÊN KẾT ĐƠN

TỔ CHỨC LIÊN KẾT CỦA DANH SÁCH LIÊN KẾT ĐƠN



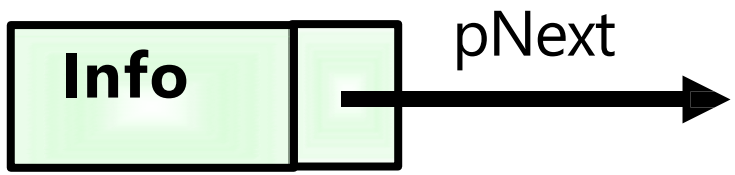
- Mỗi phần tử liên kết với phần tử đứng liền sau trong danh sách.
- Mỗi phần tử trong danh sách liên kết đơn là một cấu trúc có hai thành phần.
  - ***Thành phần dữ liệu***: Lưu trữ thông tin về bản thân phần tử.
  - ***Thành phần liên kết***: Lưu địa chỉ phần tử đứng sau trong danh sách hoặc bằng NULL nếu là phần tử cuối danh sách.

TỔ CHỨC LIÊN KẾT CỦA DANH SÁCH LIÊN KẾT ĐƠN

- Cấu trúc dữ liệu của 1 nút trong List đơn

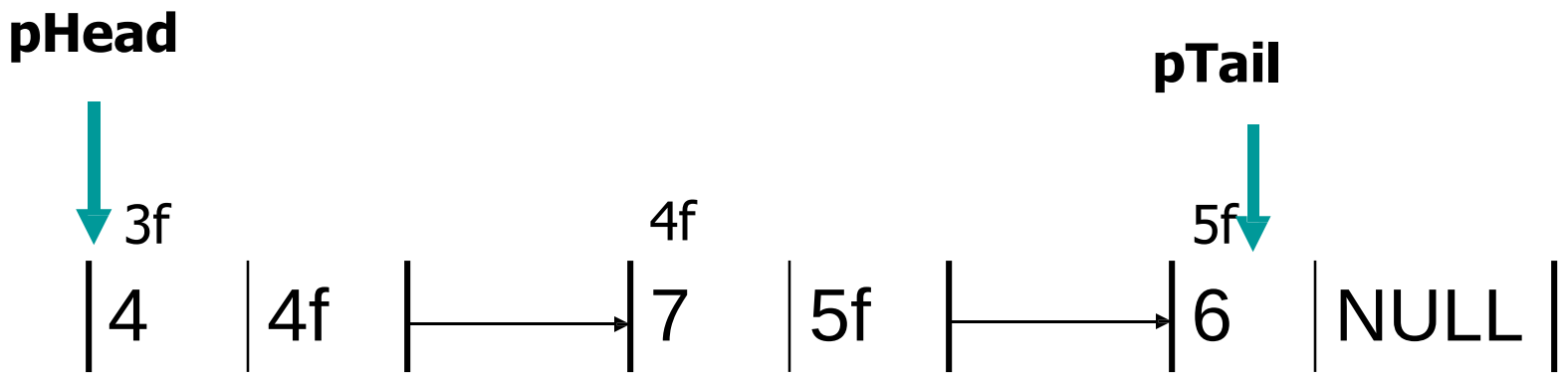
```
typedef     struct    tagNode
{  Data     Info;    // Lưu thông tin bản thân
   struct   tagNode    *pNext; //Lưu địa chỉ của Node đứng sau
}Node;
```

- Cấu trúc dữ liệu của DSLK đơn




```
typedef     struct    tagList
{  Node    *pHead;//Lưu địa chỉ Node đầu tiên trong List
   Node    *pTail; //Lưu địa chỉ của Node cuối cùng trong List
}LIST;                  // kiểu danh sách liên kết đơn
```

TỔ CHỨC LIÊN KẾT CỦA DANH SÁCH LIÊN KẾT ĐƠN



- Địa chỉ của nút đầu tiên, địa chỉ của nút cuối cùng đều không có

```
void CreateList(List &l)
{
        l.pHead=NULL;  l.pTail=NULL;
}
```

2

TẠO MỘT PHẦN TỬ MỚI

TẠO MỘT PHẦN TỬ MỚI

- Hàm trả về địa chỉ phần tử mới tạo

```
Node* CreateNode(Data x) // trong bài học là int
{       Node *p;
        p = new Node;//Cấp phát vùng nhớ cho phần tử
        if ( p==NULL)          exit(1);
        p ->Info = x; //gán dữa liệu cho nút
        p->pNext = NULL;
        return p;
}
```

3 THÊM MỘT PHẦN TỬ VÀO DANH SÁCH LIÊN KẾT

THÊM MỘT PHẦN TỬ VÀO DANH SÁCH LIÊN KẾT

- **Nguyên tắc thêm:** Khi thêm 1 phần tử vào List thì có làm cho pHead, pTail thay đổi?
- **Các vị trí cần thêm 1 phần tử vào List:**
  - ✓ Thêm vào đầu List đơn
  - ✓ Thêm vào cuối List
  - ✓ Thêm vào sau 1 phần tử q trong list

THÊM MỘT PHẦN TỬ VÀO DANH SÁCH LIÊN KẾT

- Thuật toán thêm nút p vào đầu danh sách liên kết đơn

Bắt đầu:

Nếu List rỗng thì

+ pHead = p;

+ pTail = pHead; Ngược lại

+ p->pNext = pHead;

+ pHead = p

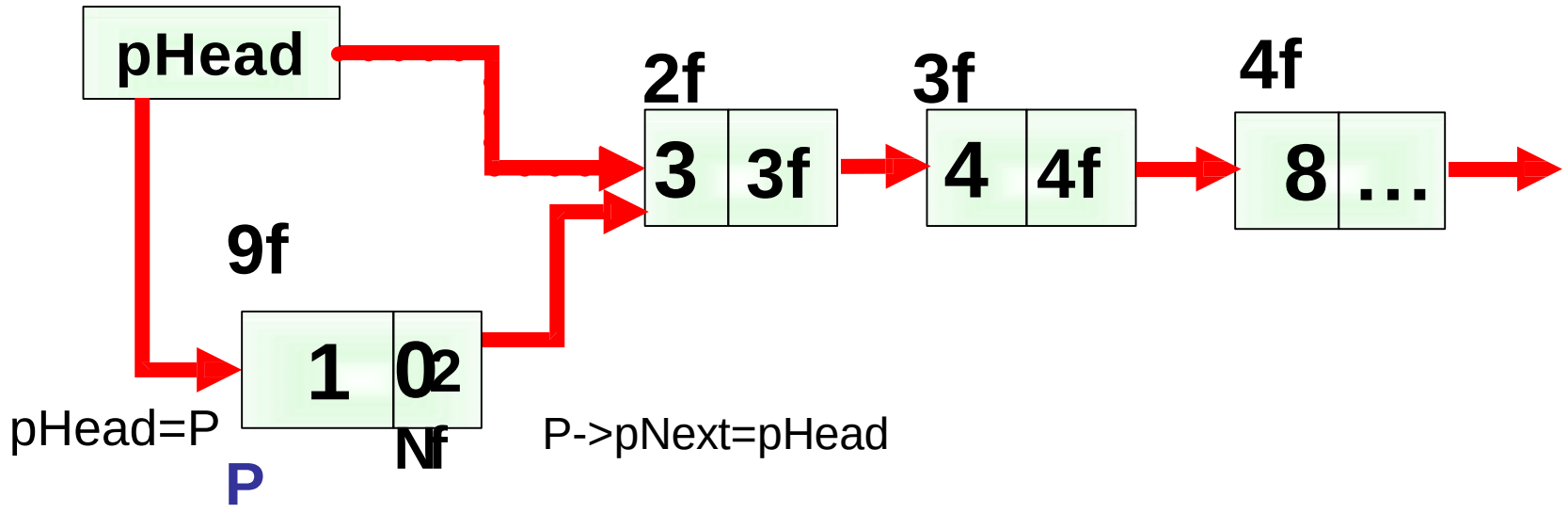
THÊM MỘT PHẦN TỬ VÀO DANH SÁCH LIÊN KẾT

- Hàm thêm nút p vào đầu danh sách liên kết đơn

```
void AddHead(LIST &l, Node* p)
{ if (l.pHead==NULL)
        {
                l.pHead = p;  l.pTail = l.pHead;
        }
        else{
                p->pNext = l.pHead;  l.pHead = p;
        }
}
```

THÊM MỘT PHẦN TỬ VÀO DANH SÁCH LIÊN KẾT

- Minh họa thêm nút p vào đầu danh sách liên kết đơn



THUẬT TOÁN THÊM PHẦN TỬ VÀO CUỐI DSLK

- Ta cần thêm nút p vào cuối list đơn

Bắt đầu:

Nếu List rỗng thì

+ pHead = p;

+ pTail = pHead; Ngược lại

+ pTail->pNext=p;

+ pTail=p

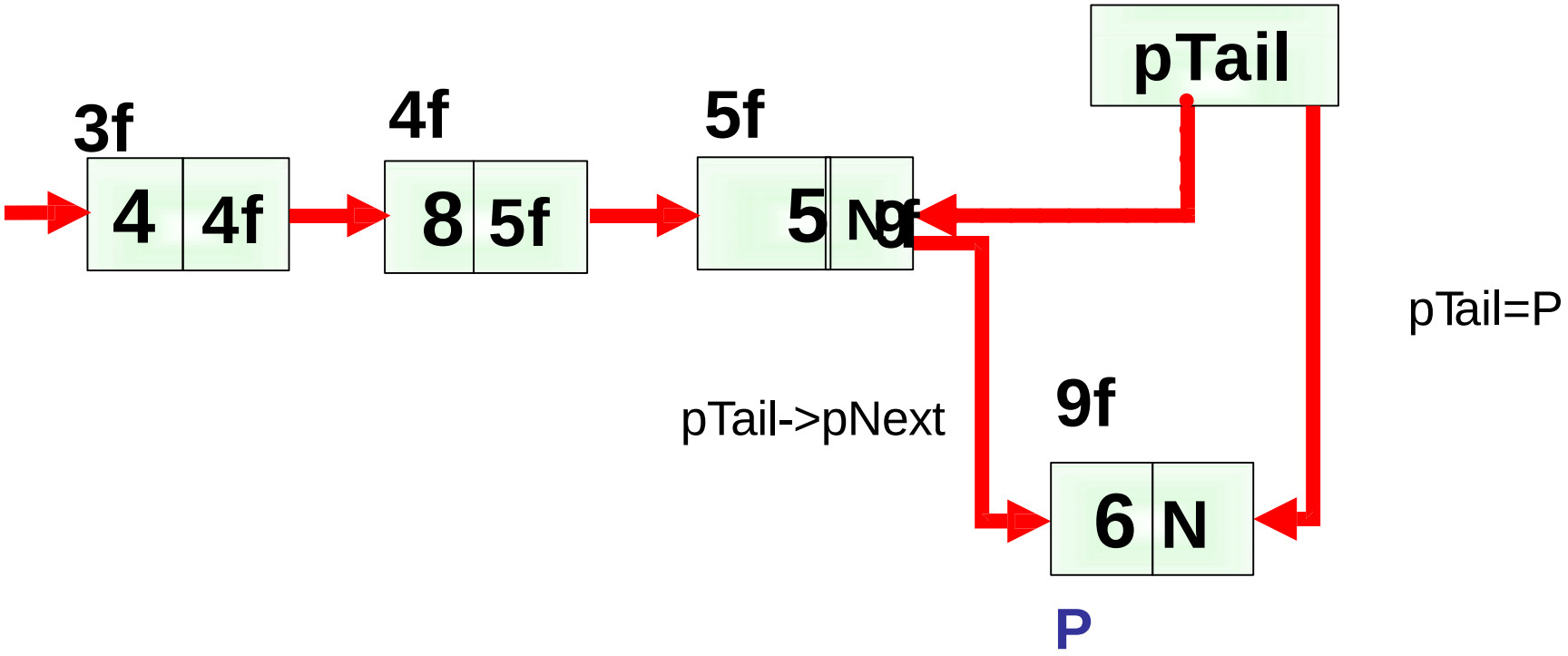
THUẬT TOÁN THÊM PHẦN TỬ VÀO CUỐI DSLK

➢Hàm thêm:

```
void AddTail(LIST &l, Node *p)
{
    if (l.pHead==NULL)
    {
            l.pHead = p;  l.pTail = l.pHead;
    }
    else
    {
            l.pTail->Next = p;  l.pTail = p;
    }
}
```

THUẬT TOÁN THÊM PHẦN TỬ VÀO CUỐI DSLK

- Minh họa thuật toán:



THUẬT TOÁN THÊM PHẦN TỬ VÀO PHẦN TỬ P

- Ta cần thêm nút p vào sau nút q trong list đơn

Bắt đầu:

Nếu (q!=NULL) thì

B1: p->pNext = q->pNext

B2: + q->pNext = p

+ nếu q = pTail thì

pTail=p

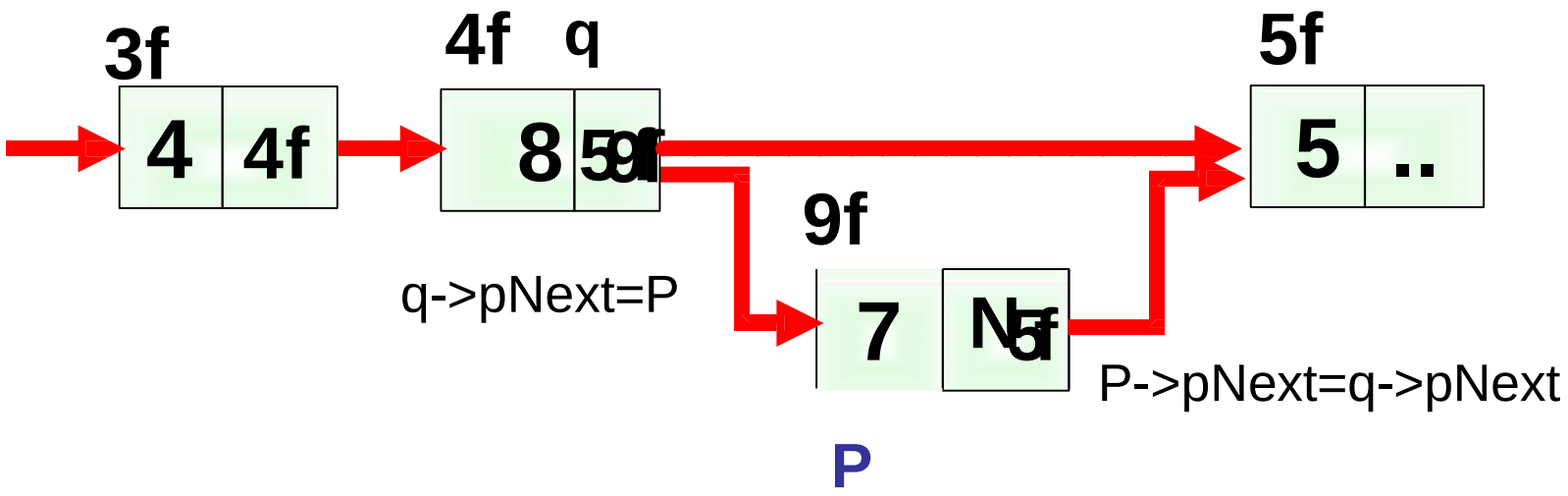
THUẬT TOÁN THÊM PHẦN TỬ VÀO PHẦN TỬ P

- Cài đặt thuật toán:

```
void InsertAfterQ(List &l, Node *p, Node *q)
{
    if(q!=NULL)
    {
        p->pNext=Q->Next;
        q->pNext=p;
        if(l.pTail==q)
                l.Tail=q;
   }
   else
        AddHead(l,q);// thêm q vào đầu list
}
```

THUẬT TOÁN THÊM PHẦN TỬ VÀO PHẦN TỬ P

- Minh họa thuật toán:



4

HỦY PHẦN TỬ TRONG DSLK

HỦY PHẦN TỬ TRONG DSLK

- **Nguyên tắc**: Phải cô lập phần tử cần hủy trước hủy.
- Các vị trị cần hủy
  - Hủy phần tử đứng đầu List
  - Hủy phần tử có khoá bằng x
  - Huỷ phần tử đứng sau q trong danh sách liên kết đơn
- Các phần tử trong DSLK đơn được cấp phát vùng nhớ động bằng hàm new, thì sẽ được giải phóng vùng nhớ bằng hàm delete.

HỦY PHẦN TỬ TRONG DSLK

- Thuật toán:

Bắt đầu: Nếu (pHead!=NULL) thì

B1: p=pHead

B2:

+ pHead = pHead->pNext

+ delete (p)

B3: Nếu pHead==NULL thì pTail=NULL

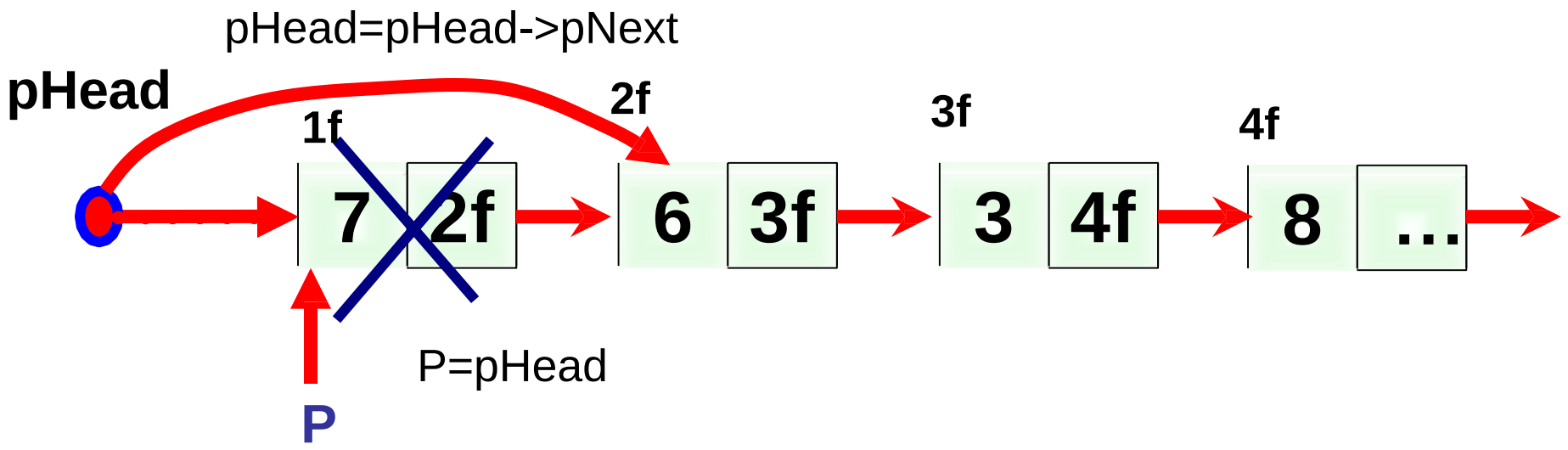
HỦY PHẦN TỬ TRONG DSLK

- Hủy được hàm trả về 1, ngược lại hàm trả về 0

```
int    RemoveHead(List &l, int &x)
{      Node *p;
       if(l.pHead!=NULL) {
                p=l.pHead;
                x=p->Info; //lưu Data của nút cần hủy
                l.pHead=l.pHead->pNext;
                delete p;  if(l.pHead==NULL)
                          l.pTail=NULL;
       }        return 1;
       return 0; }
```

HỦY PHẦN TỬ TRONG DSLK

- Minh họa:



HỦY PHẦN TỬ SAU PHẦN TỬ P TRONG DSLK

- Bắt đầu:

Nếu (q!=NULL) thì //q tồn tại trong List

B1: p=q->pNext;// p là phần tử cần hủy

B2: Nếu (p!=NULL) thì // q không phải là phần tử cuối

+ q->pNext=p->pNext;// tách p ra khỏi xâu

+ nếu (p== pTail) // nút cần hủy là nút cuối

pTail=q;

+ delete p;// hủy p

HỦY PHẦN TỬ SAU PHẦN TỬ P TRONG DSLK

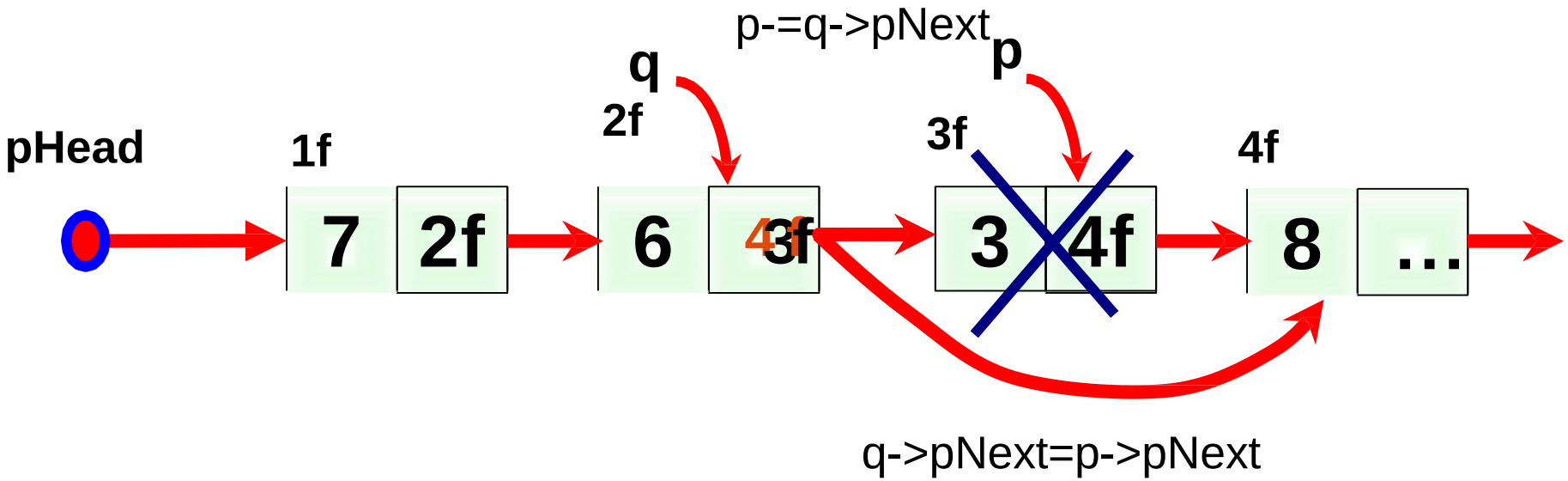
- Cài đặt thuật toán:

```
int RemoveAfterQ(List &l, Node *q, int &x)
{     Node *p;
if(q!=NULL)
      {       p=q->pNext; //p là nút cần xoá
              if(p!=NULL) // q không phải là nút cuối
              {    if(p==l.pTail) //nút cần xoá là nút cuối cùng
                                  l.pTail=q;// cập nhật lạ pTail
       q->pNext=p->pNext; x=p->Info;  delete p;

              }
              return 1;
      }
      Else return 0;}
```

HỦY PHẦN TỬ SAU PHẦN TỬ P TRONG DSLK

- Minh họa thuật toán:



THUẬT TOÁN HỦY PHẦN TỬ CÓ KHOÁ X

- Thuật toán:

Bước 1:

Tìm phần tử p có khoá bằng x, và q đứng trước p

Bước 2:

Nếu (p!=NULL) thì //tìm thấy phần tử có khoá bằng x

Hủy p ra khỏi List bằng cách hủy phần tử đứng sau q

Ngược lại

Báo không tìm thấy phần tử có khoá

THUẬT TOÁN HỦY PHẦN TỬ CÓ KHOÁ X

- Cài đặt thuật toán:

```
int RemoveX(List &l, int x)
{       Node *p,*q = NULL; p=l.Head;  while((p!=NULL)&&(p->Info!=x))//tìm
 { q=p;  p=p->Next; }
 if(p==NULL)                          //không tìm thấy phần tử có khoá bằng x
      return 0;
 if(q!=NULL)                          //tìm thấy phần tử có khoá bằng x
      DeleteAfterQ(l,q,x);
 else     RemoveHead(l,x);  return 1;        //phần tử cần xoá nằm đầu List
```

5 TÌM MỘT PHẦN TỬ TRONG DSLK ĐƠN

TÌM MỘT PHẦN TỬ TRONG DSLK ĐƠN

- Các bước của thuật toán tìm nút có Info bằng x trong list đơn:

Bước 1: p=pHead;// địa chỉ của phần tử đầu trong list đơn

Bước 2:

Trong khi p!=NULL và p->Info!=x p=p->pNext;//

xét phần tử kế

Bước 3:

+ Nếu p!=NULL thì p lưu địa chỉ của nút có Info = x

+ Ngược lại : Không có phần tử cần tìm

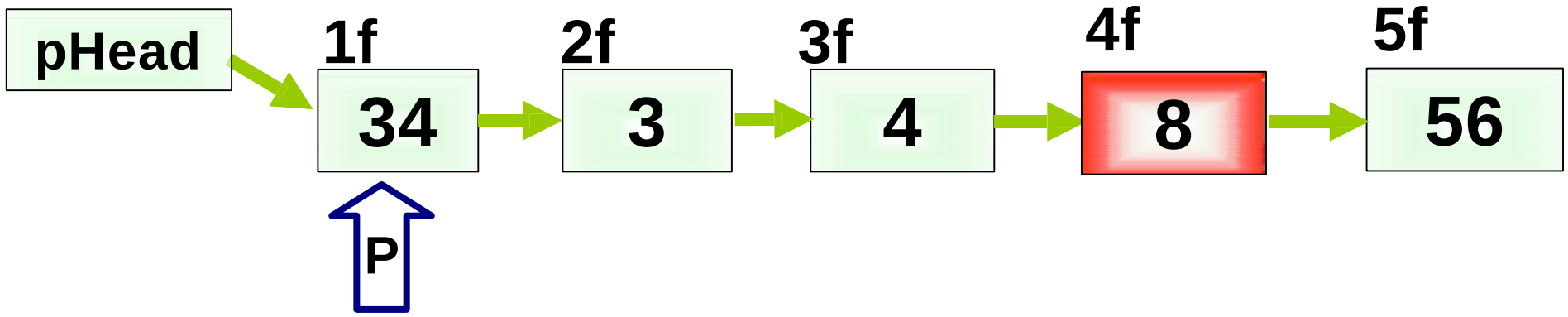
TÌM MỘT PHẦN TỬ TRONG DSLK ĐƠN

- Hàm tìm phần tử có Info = x, hàm trả về địa chỉ của nút có Info = x, ngược lại hàm trả về NULL

```
Node *Search(LIST l, Data x)
{
    Node        *p;
    p = l.pHead;
      while((p!= NULL)&&(p->Info != x))  p = p->pNext;
   return p;
}
```

TÌM MỘT PHẦN TỬ TRONG DSLK ĐƠN

- Minh họa thuật toán:



X = 8

Tìm thấy, hàm trả về địa chỉ của nút tìm thấy là 4f

TỔNG KẾT

- DSLK gồm tập hợp các nút liên kết với nhau thông qua vùng liên kết.
- Một số thao tác cơ bản trên danh sách liên kết: thêm phần tử, xóa phần tử, tìm kiếm, duyệt tất cả các phần tử…


感謝你
Danke
Gracias
Grazie
Merci
Thanks!
THANK
YOU!


VAIO