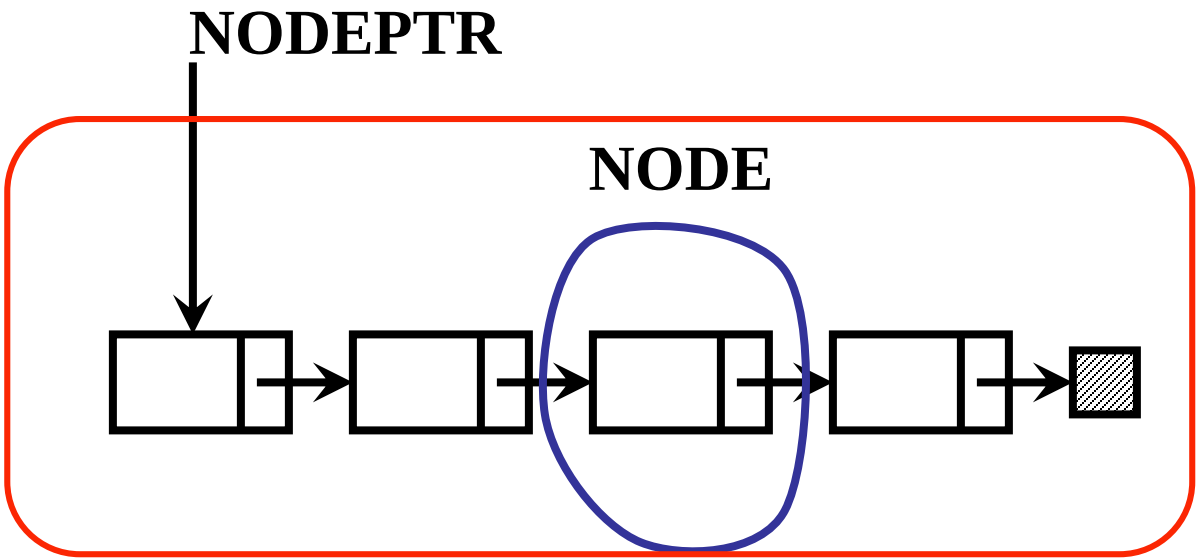
NỘI DUNG

DANH SÁCH LIÊN KẾT ĐƠN (LIST)

ThS.Nguyễn Thúy Loan

Hình ảnh xâu đơn một trỏ



typedef NODE * NODEPTR

ThS.Nguyễn Thúy Loan

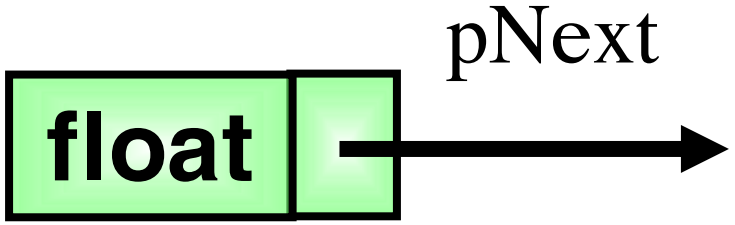
Bài toán: dùng DS liên kết đơn 1 trỏ, viết chương trình

1. Nhập vào một dãy số thực , nhập cho đến khi gặp số 0 thì dừng.
2. Xuất ra dãy số vừa nhập.
3. Tính tổng các phần tử có trong dãy số
4. Tìm phần tử lớn nhất có trong dãy.
5. Đếm số phần tử lớn nhất có trong dãy
6. Thêm phần tử có giá trị là x vào sau phần tử có giá trị là y đầu tiên ở trong dãy số. Nếu không tồn tại y thì thông báo
7. Xóa phần tử âm đầu tiên có trong dãy.

VÍ DỤ 1 CTDL DANH SÁCH LIÊN KẾT ĐƠN

Ví dụ 1: Hãy khai báo CTDL cho dslk đơn các số NGUYÊN.

```
1.struct node
2.{
3.      float info;
4.      struct node*pNext;
5.};
6.typedef struct node NODE;
7.typedef NODE *NODEPTR;
```



3. Các thao tác trên DSLK đơn

1. Khởi tạo DSLK
2. Nhập dữ liệu
3. Tạo nút
4. Thêm lần lượt từng nút vào DSLK
5. Xuất Danh sách ra màn hình

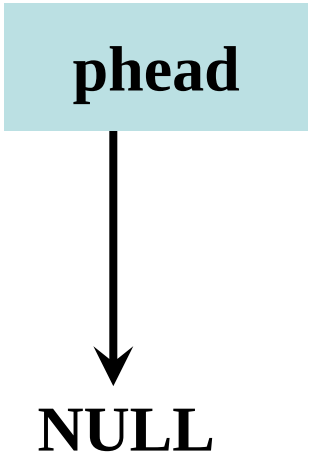
Các bước 2,3 và 4 lặp lại nhiều lần

1.KHỞI TẠO DANH SÁCH LIÊN KẾT ĐƠN

- Khái niệm: Khởi tạo danh sách liên kết đơn là tạo ra danh sách rỗng không chứa node nào hết.

- Định nghĩa hàm

```
1.void Init(NODEPTR &phead)
2.{
3.     phead = NULL;
4.}
```



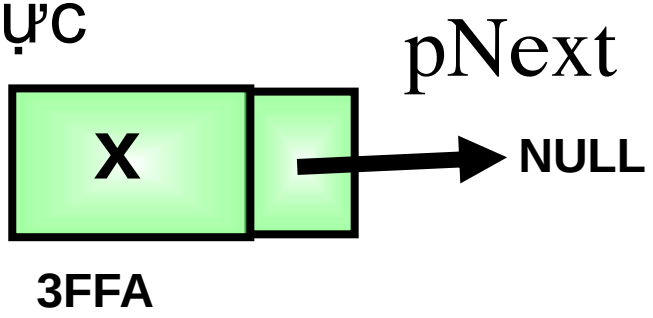
2. TẠO NODE CHO DANH SÁCH LIÊN KẾT ĐƠN

Khái niệm:

- ❖ **Tạo node cho danh sách liên kết đơn là :**
- ➢ **xin cấp phát bộ nhớ**
  - **Có kích thước bằng với kích thước của kiểu dữ liệu NODE**
  - **Để chứa thông tin đã được biết trước.**

2. TẠO NODE CHO DANH SÁCH LIÊN KẾT ĐƠN

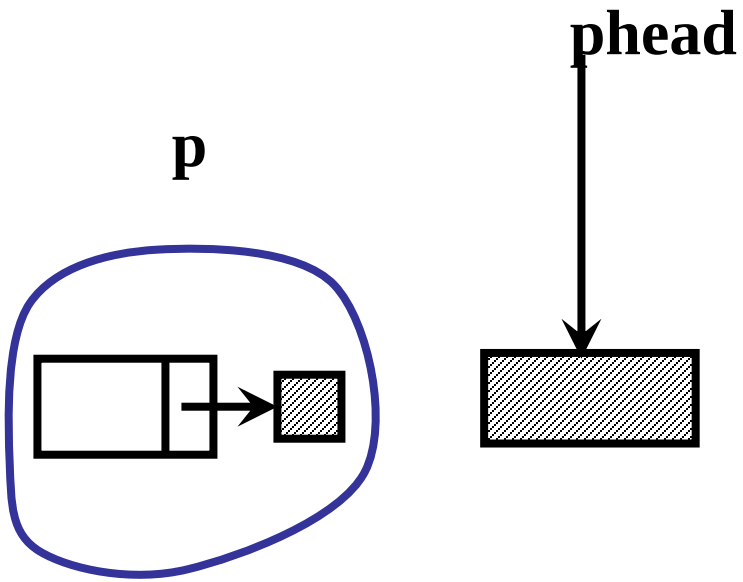
- Định nghĩa hàm tạo nút cho số thực




```
1.NODE* GetNode( float x)
2.{    NODE *p=new NODE;
3.     if (p==NULL)    return NULL;
4.     p->info = x;
5.     p->pNext = NULL;
6.     return p;
7.}
```

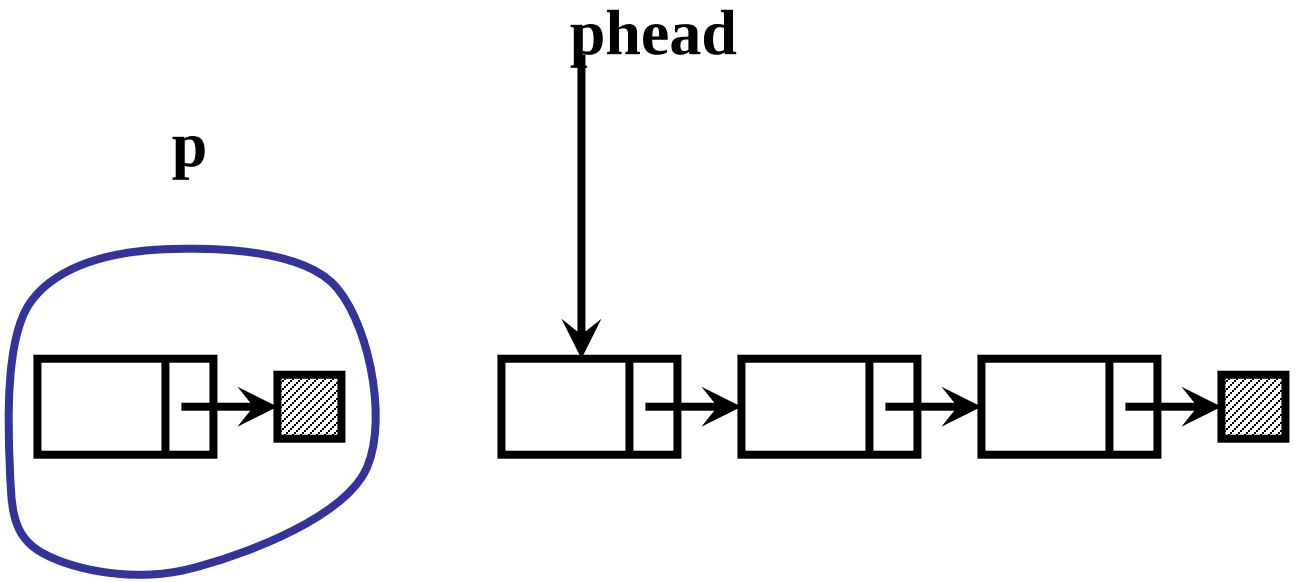
6. THÊM MỘT NODE VÀO ĐẦU DS LIÊN KẾT ĐƠN

- Khái niệm: Thêm một node vào đầu danh sách liên kết đơn là gắn node đó vào đầu danh sách.

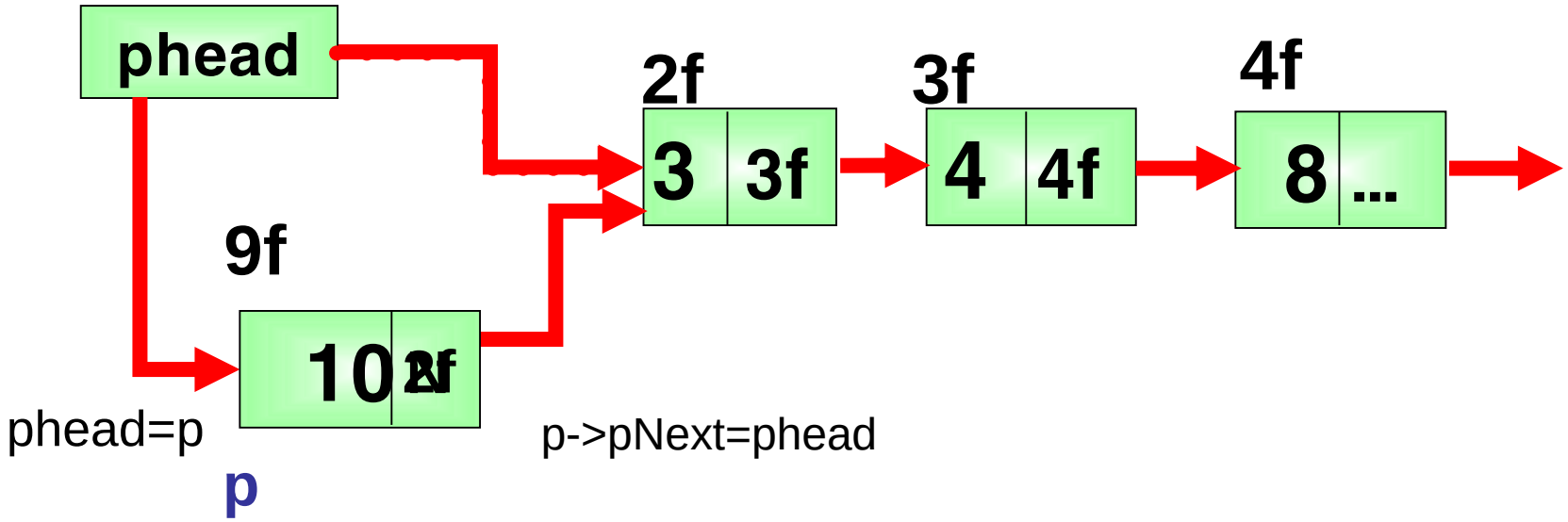


6. THÊM MỘT NODE VÀO ĐẦU DS LIÊN KẾT ĐƠN

- Khái niệm: Thêm một node vào đầu danh sách liên kết đơn là gắn node đó vào đầu danh sách.



Minh họa thuật toán thêm vào đầu



6. THÊM MỘT NODE VÀO ĐẦU DS LIÊN KẾT ĐƠN

➢ Định nghĩa hàm:

```
1.void AddHead(NODEPTR &phead, NODE*p)
2.{    if(phead==NULL) phead=p;
     1.else
     2.{    p->pNext = phead;
3.          phead = p;
4.      }
5.}
```

7. NHẬP DỮ LiỆU VÀO DS LIÊN KẾT ĐƠN

Các thao tác phải thực hiện :

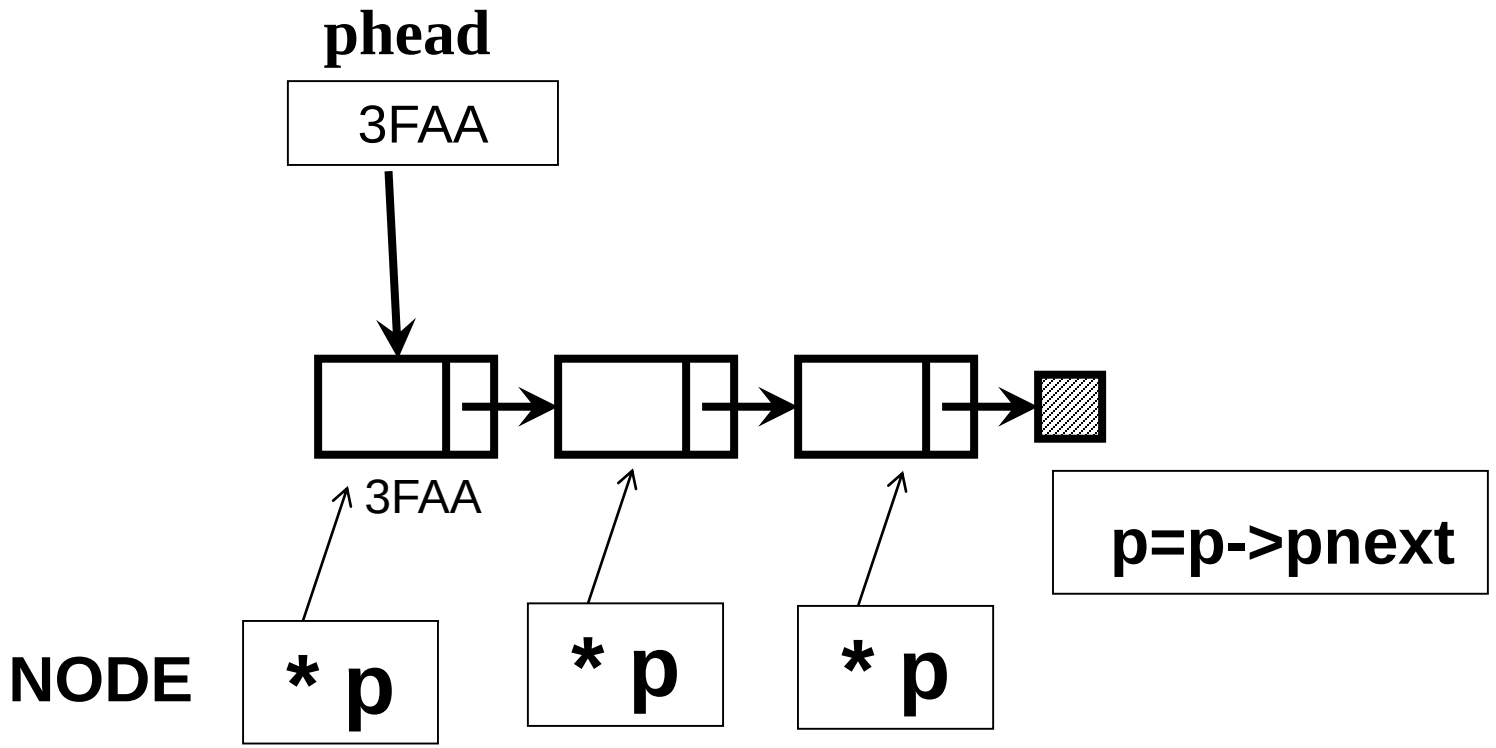
1. **Khởi tạo danh sách**
2. **Nhập dữ liệu**
3. **Tạo Node cho dữ liệu**
4. **Gắn Node vào danh sách**
5. **Lặp lại bước 2 ,3 và 4 cho đến khi hết dữ liệu cần nhập**

Ví dụ: Nhập danh sách liên kết đơn các số nguyên.

```
1.void Input(NODEPTR &phead)
2.{    float x;
3.     Init(phead);
4.     do{
5.          printf("Nhap gia tri x: ");
6.          scanf("%f",&x);
7.          if( x!=0)
8.          {
9.             NODE*p=GetNode(x);
10.            if (p!=NULL)
11.               AddHead(phead,p);
12.         }
13.       }while(x!=0)
14.}
```

8. DUYỆT TUẦN TỰ DANH SÁCH LIÊN KẾT ĐƠN

- **Khái niệm: duyệt danh sách liên kết đơn là thăm qua tất cả các node mỗi node một lần.**



8. DUYỆT TUẦN TỰ DANH SÁCH LIÊN KẾT ĐƠN

- **Khái niệm: duyệt danh sách liên kết đơn là thăm qua tất cả các node mỗi node một lần.**
- Định nghĩa hàm trừu tượng

```
1.KDL <Tên Hàm>(NODEPTR phead)
2.{    ...
3.   for(NODE*p=phead;p!=NULL;p=p->pnext )
4.      {   ...
5.        //
6.      }
7.      ...
8.}
```

8. DUYỆT TUẦN TỰ DANH SÁCH LIÊN KẾT ĐƠN

- **Khái niệm: duyệt danh sách liên kết đơn là thăm qua tất cả các node mỗi node một lần.**
- Định nghĩa hàm trừu tượng

```
1.KDL <Tên Hàm>(NODEPTR phead)
2.{    ...
3.     NODE*p=phead;
4.     while(p!=NULL)
5.     {   ...
6.        p=p->pNext;
7.     }
8.     ...
9.}
```

8. DUYỆT TUẦN TỰ DANH SÁCH LIÊN KẾT ĐƠN

```
1.Void  xuat( NODEPTR phead )
2.{
3.   for(NODE*p=phead;p!=NULL;p=p->pnext )
4.   {
5.          printf("%.1f", p->info );
6.   }
7.
8.}
```

8. DUYỆT TUẦN TỰ DANH SÁCH LIÊN KẾT ĐƠN

```
1.void xuat (NODEPTR phead)
2.{
3.     NODE*p=phead;
4.     while(p!=NULL)
5.     {
          printf("%f", p->info );
6.         p=p->pNext;
7.     }
8.
9.}
```

VD:Tính tổng các số dslk đơn các số thực.

```
1.float Tong(NODEPTR phead)
2.{
3.     float s=0;
4.     NODE * p = phead;
5.     while(p!=NULL)
6.     {
7.        s=s+p->info;
8.        p=p->pNext;
9.     }
10.     return s
11.}
```

VD:Tính tổng các số lẻ trong dslk đơn các số nguyên.

```
1.float TongLe(NODEPTR phead)
2.{
3.     float s=0;
4.     NODE * p = phead;
5.     while(p!=NULL)
6.     {
7.        if(p->info%2!=0)
8.           s=s+p->info;
9.        p=p->pNext;
10.    }
11.    return s
12.}
```

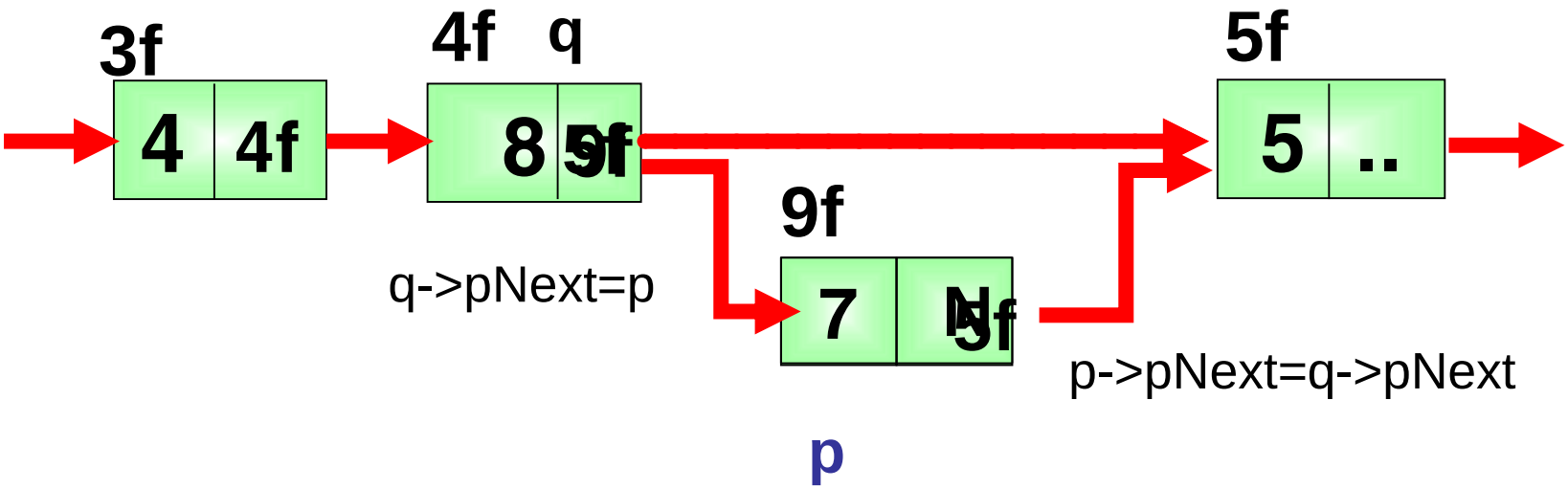
VD:Tim max trong dslk đơn các số nguyên.

```
1.float Timmax(NODEPTR phead)
2.{
3.     float max = phead->info ;
4.     NODE * p = phead;
5.     while(p!=NULL)
6.     {
7.        if( p->info > max )
8.              max= p->info;
9.        p=p->pNext;
10.     }
11.     return s
12.}
```

9. CHƯƠNG TRÌNH ĐẦU TIÊN DSLK ĐƠN

```
// Khai báo hàm
void Init(NODEPTR&);
NODE* GetNode(int);
void AddHead(NODEPTR&, NODE*);
void Input(NODEPTR&);
void Output(NODEPTR);
int Tong(NODEPTR);
// Hàm main
void main()
{    NODEPTR lst;
     Input(lst);
     Output(lst);
     float kq = Tong(lst);
     printf("Tong la: %f",kq);
}
```

Thêm phần tử p vào sau phần tử q



ThS.Nguyễn Thúy Loan

Thêm phần tử p vào sau phần tử q

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

Bắt đầu:

Nếu (q!=NULL) thì

B1: p->pNext = q->pNext

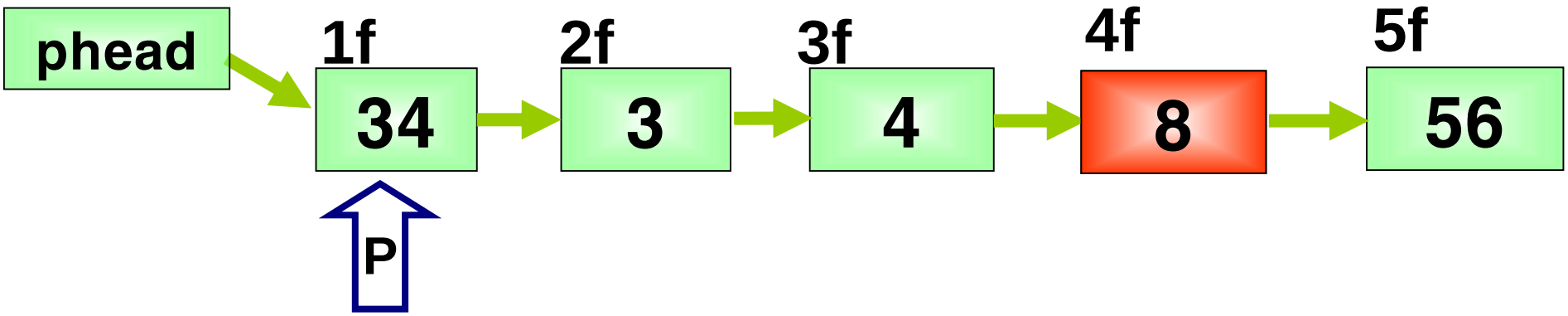
B2: q->pNext = p

Ngược lại: Thêm vào đầu danh sách

Cài đặt thuật toán

```
void addAfterq(NODEPTR &phead, NODE *p, NODE *q)
  {
      if(q!=NULL)
      {
        p->pNext=q->pNext;
        q->pNext=p;
      }
     else
        AddHead(phead,p);// thêm pvào đầu list
}
```

Minh họa thuật toán tìm phần tử trong DSLK



X = 8

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

ThS.Nguyễn Thúy Loan

Tìm 1 phần tử trong DSLK đơn

- Tìm tuần tự (hàm trả về), 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 ds 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 1 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

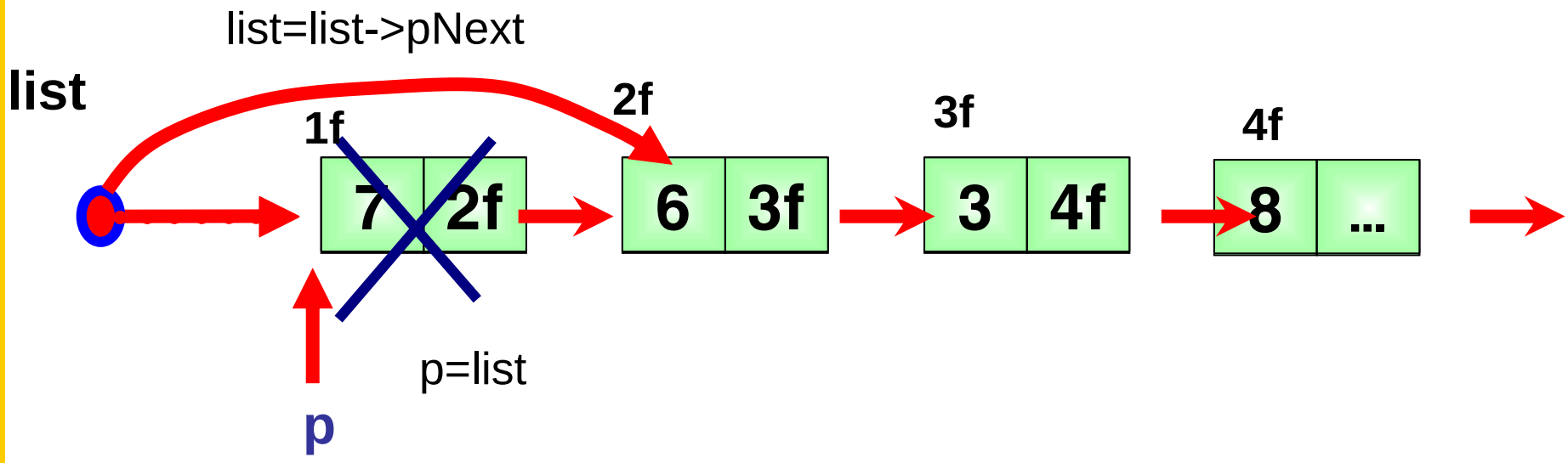
```
NODE * search( NODEPTR phead, int x)
{
    NODE *p = phead;
    while(p!=NULL && p->info!=x)
        p=p->pnext;
    return p;

}
```

Cài đặt thuật toán

```
void themxsauy(NODEPTR &phead)
  {    float x,y;
       printf (“ nhập gia tri can them:”);
       scanf(“%f ”,&x);
       NODE *p=getnode(x);
       printf (“ nhập gia tri muon them vao sau :”);
       scanf(“%f ”,&y);
      NODE *q= search (phead,y)
       addAfterq(phead, p,q)
 }
```

Thuật toán hủy phần tử đầu trong DSLK



Hủy phần tử trong DSLK đơn

- **Nguyên tắc**: Phải cô lập phần tử cần hủy trước khi hủy.
- Các vị trị cần hủy
  - Hủy phần tử đứng đầu danh sách
  - Hủy phần tử có khoá bằng x
  - Huỷ phần tử đứng sau q trong danh sách liên kết đơn
- Ở phần trên, 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.

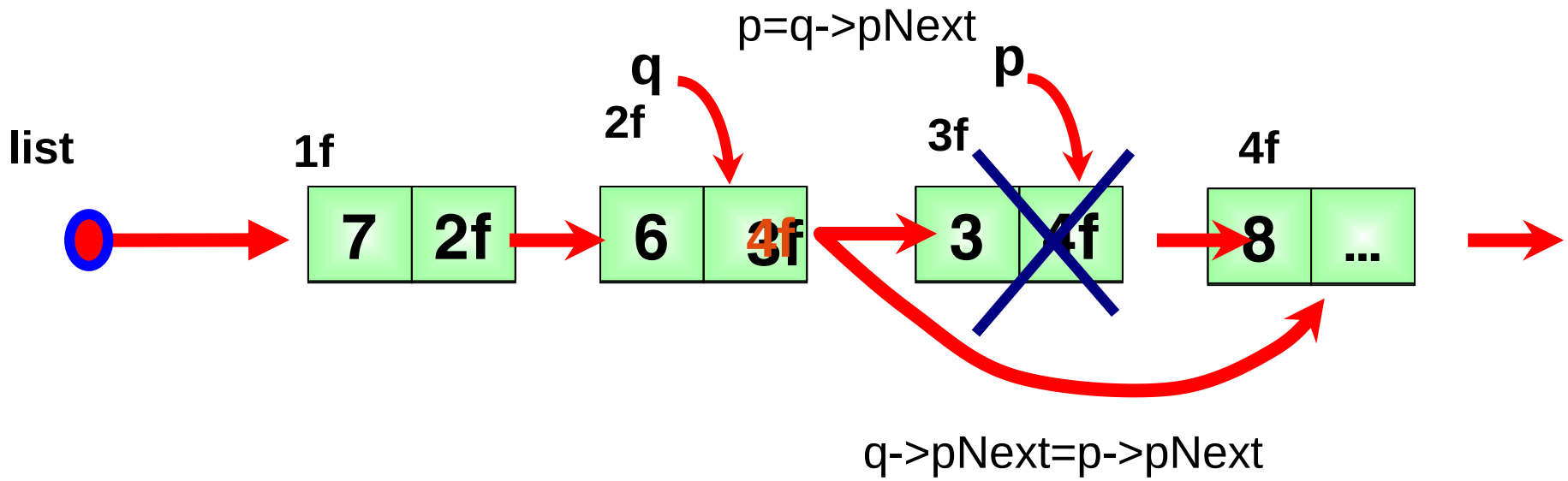
Thuật toán hủy phần tử đầu trong DSLK

- Bắt đầu:
  - Nếu (l !=NULL) thì
    - B1: p=pHead //cho p bằng phần tử cần xóa
    - B2: l = l->pNext
    - delete (p)

Thuật toán hủy phần tử đầu trong DSLK

```
void  deletehead ( list  &l )
{
        NODE *p;
        if ( l.phead !=NULL )
        {
                p=l.phead;
                l.phead = l.phead->pnext;
                delete(p);
        }
}
```

Hủy phần tử sau phần tử q trong List



ThS.Nguyễn Thúy Loan

Hủy phần tử sau phần tử q trong List

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*

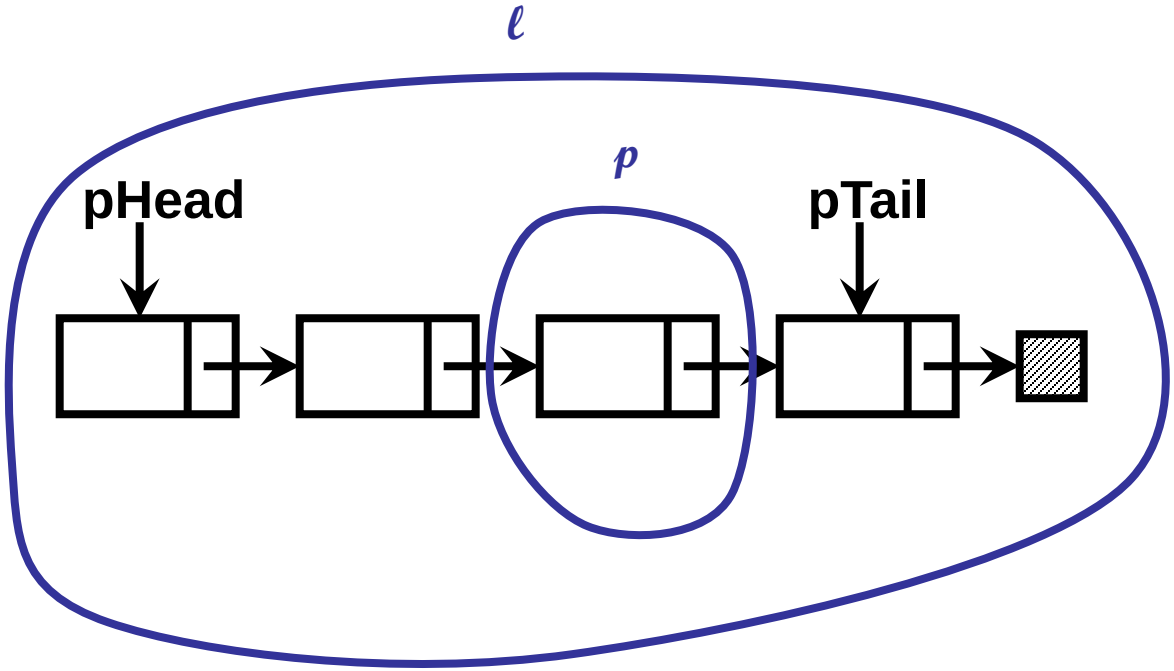
+ delete p;// *hủy p*

Hủy phần tử sau phần tử q trong List

```
void deleteafterq ( LIST &l, NODE *q)
{
    If (q!=NULL)  // q tồn tại trong List
    {
        p=q->pNext;  // p là phần tử cần hủy
        if (p!=NULL)  // q không phải là phần tử cuối
        {
            q->pNext=p->pNext;   // tách p ra khỏi xâu
             p-> pNext = NULL     // cô lập p
            delete p;    // hủy p
        }
    }
    }
```

Xóa 1 phần tử trong DSLK đơn

- Bài tập 012: Định nghĩa hàm tách node p trong danh sách liên kết đơn ra khỏi danh sách.
- Ý tưởng:



Thuật toán hủy phần tử có khoá x

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 danh sách bằng cách hủy phần tử đứng sau q

Ngược lại

Báo không tìm thấy phần tử có khoá

Xóa 1 phần tử trong DSLK đơn

```
void xoa1pt( list &l , float x)
{
    if(l.phead==NULL) return NULL;
    if(l.phead -> info == x)   xoahead(l);
    NODE * p= searchx ( l,x);
    NODE *q= Before(l,p);
    deleteafterq( l, q );
}
```

Tìm 1 node trong danh sách liên kết đơn.

```
NODE* Before( LIST l, NODE *p )
{
        if (phead==NULL) return NULL;
        if (phead==p) return NULL;
        NODE *lc=phead;
        while(lc->pnext !=p && lc!=NULL)
                lc=lc->pnext;
        return lc;
}
```

Hủy phần tử x trong List

```
void  delete x ( list &l , float x )
{
      node *q, *p ;
      p=l.phead ;
      if ( p->info == x )
     {     l.phead = l.phead->pnext ;
           delete p;
      }
  else
   {
```

Hủy phần tử x trong List

```
else
   {   q = p->pnext ;
       while( q != NULL )
      {     if ( q->info == x )
            {    p->pnext = q->pnext;
                 q->pnext = NULL;
                 delete q;
             }
            p = q;
            q = q->pnext;
      } } }
```