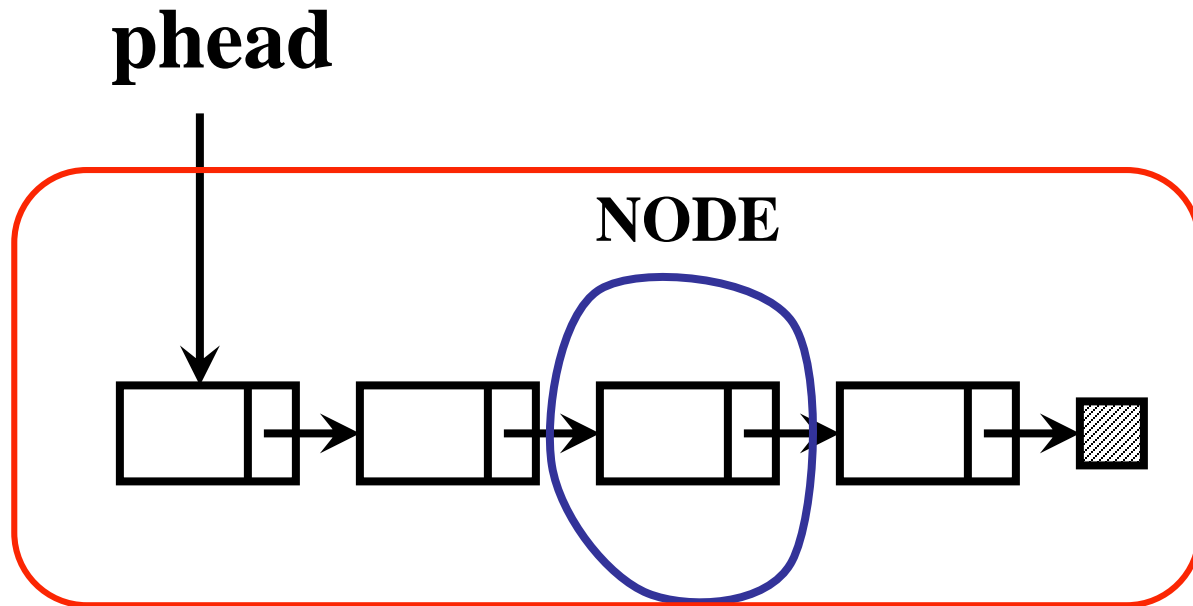


DANH SÁCH LIÊN KẾT ĐƠN (LIST)

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

1. Nhập vào một dãy số nguyên , 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. Thêm phần tử có giá trị là 3 vào sau phần tử có giá trị là 2 đầu tiên ở trong dãy số. Nếu không tồn tại phần tử có giá trị là 2 thì thêm vào đầu dãy
6. Xóa phần tử âm đầu tiên có trong dãy.
7. Xóa phần tử có giá trị là 5 cuối cùng có trong dãy số

Hình ảnh xâu đơn một trỏ



NODE * phead

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

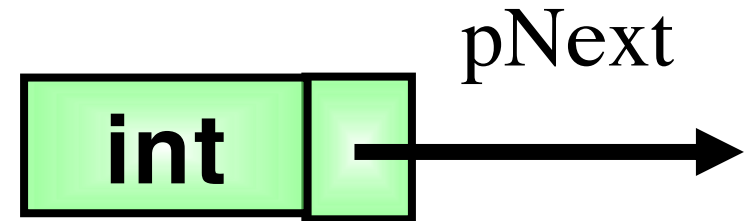
2. {

3. info;

4. pNext;

5. };

6. typedef struct node NODE;



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

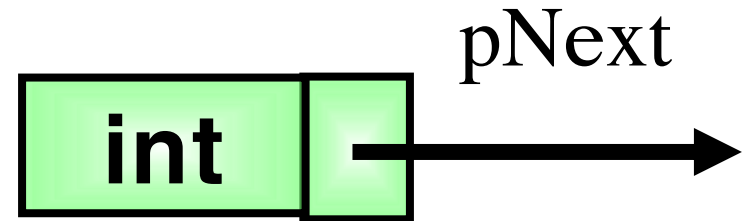
2. {

3. int info;

4. pNext;

5. };

6. typedef struct node NODE;



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

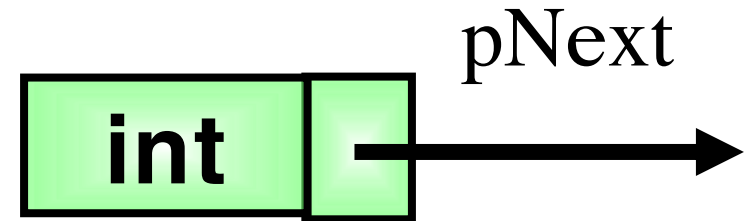
2. {

3. int info;

4. node *pNext;

5. };

6. typedef struct node NODE;



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

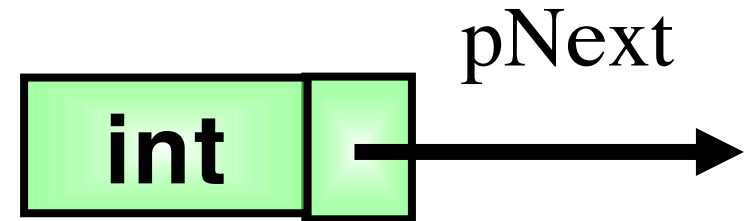
2. {

3. int info;

4. struct node *pNext;

5. };

6. typedef struct node NODE;



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. {

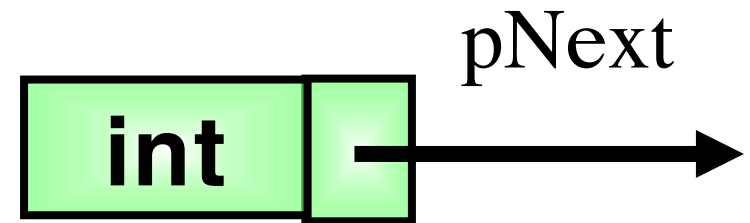
3. int info;

4. struct node *pNext;

5. };

6. typedef struct node NODE;

7. NODE *phead;



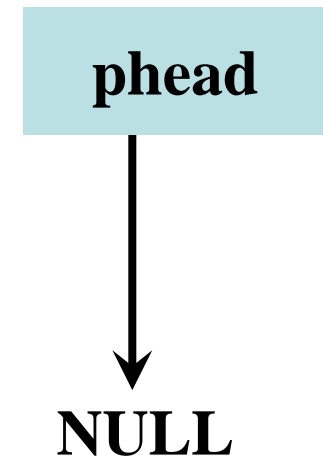
3. Các thao tác trên DSLK đơn

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 (NODE * &phead)  
2. {  
3.     phead = NULL;  
4. }
```



2. TẠO NODE CHO DANH SÁCH LIÊN KẾT ĐƠN

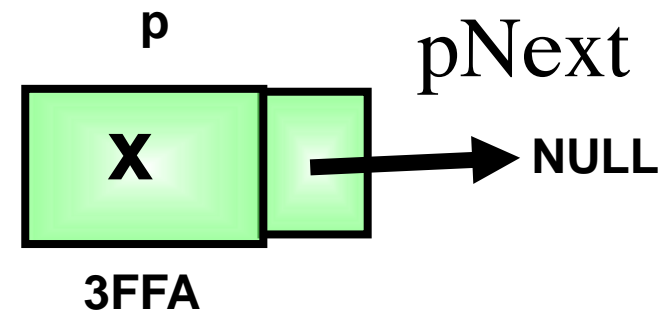
Khái niệm:

- ❖ 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

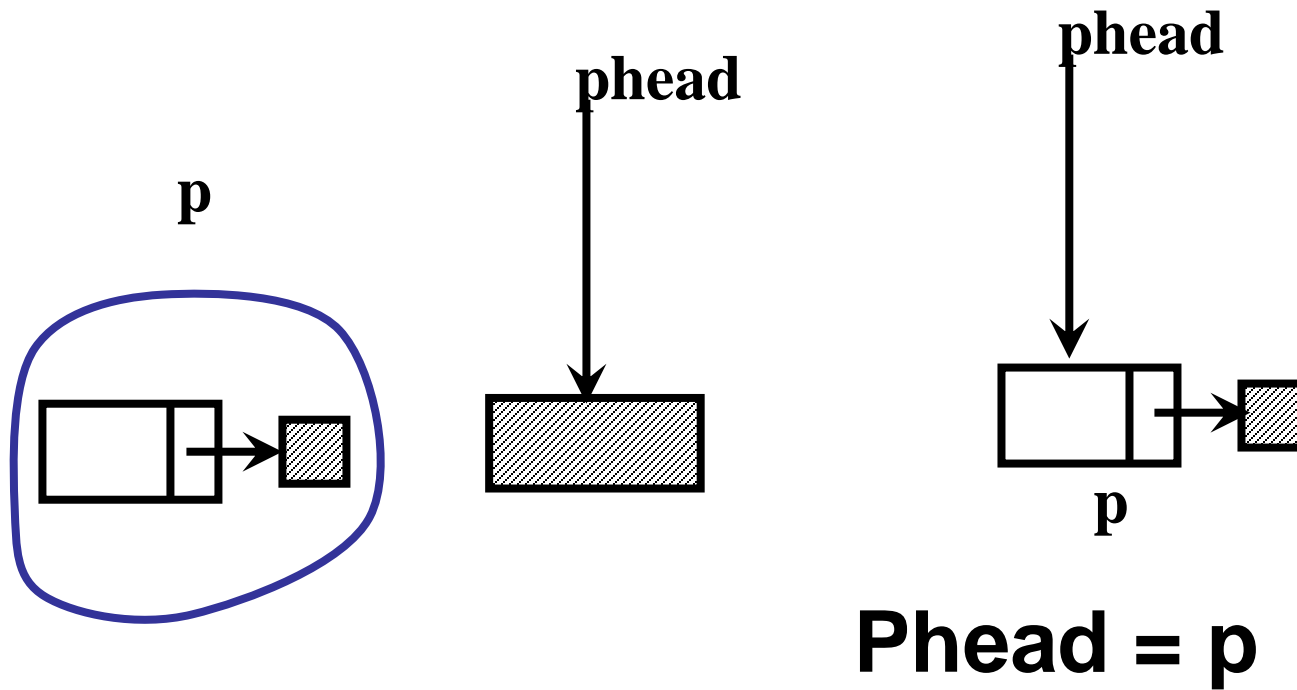
➤ Định nghĩa hàm tạo nút cho số nguyên

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



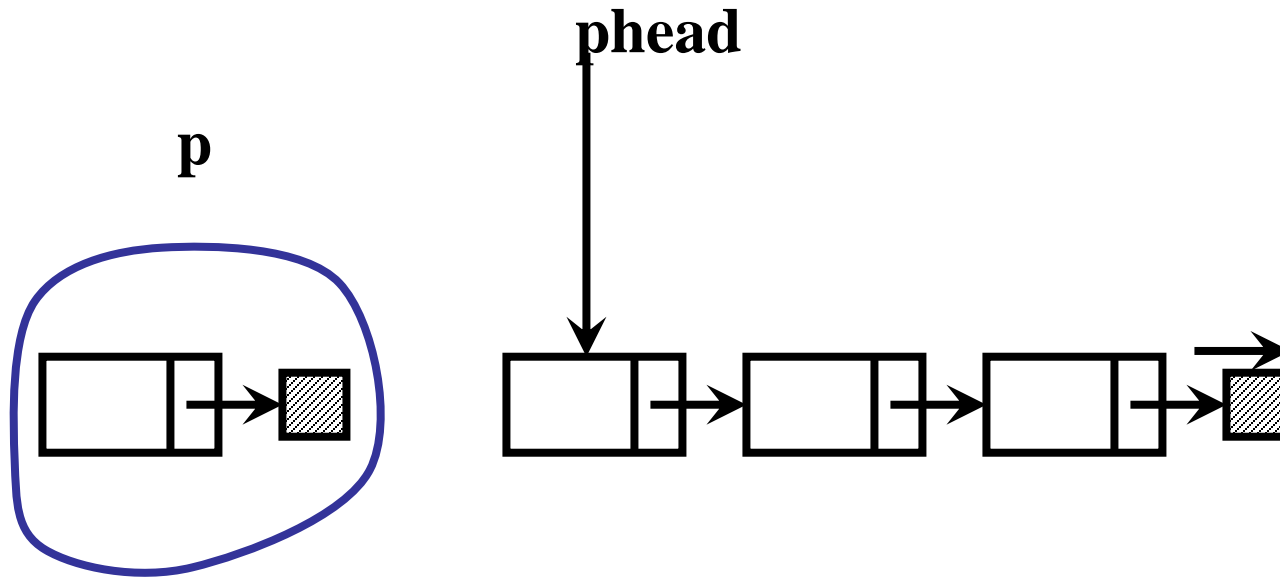
6. THÊM MỘT NODE VÀO ĐẦU DS LIÊN KẾT ĐƠN

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

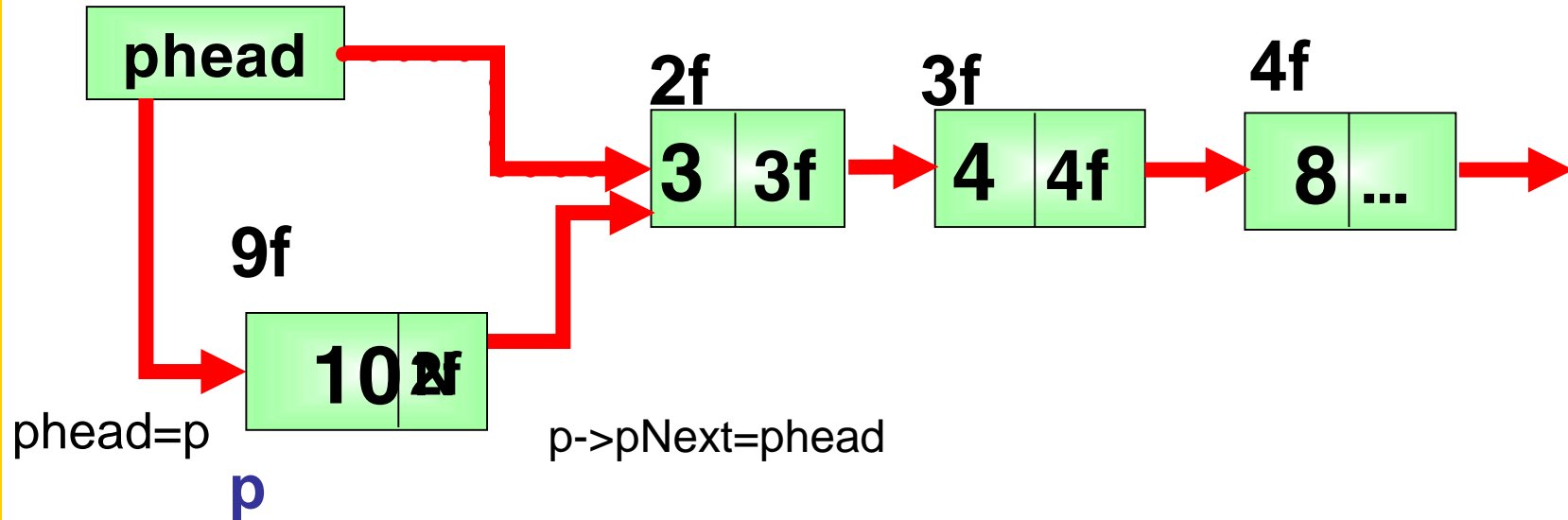


6. THÊM MỘT NODE VÀO ĐẦU DS LIÊN KẾT ĐƠN

- ◆ 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 (NODE *&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.

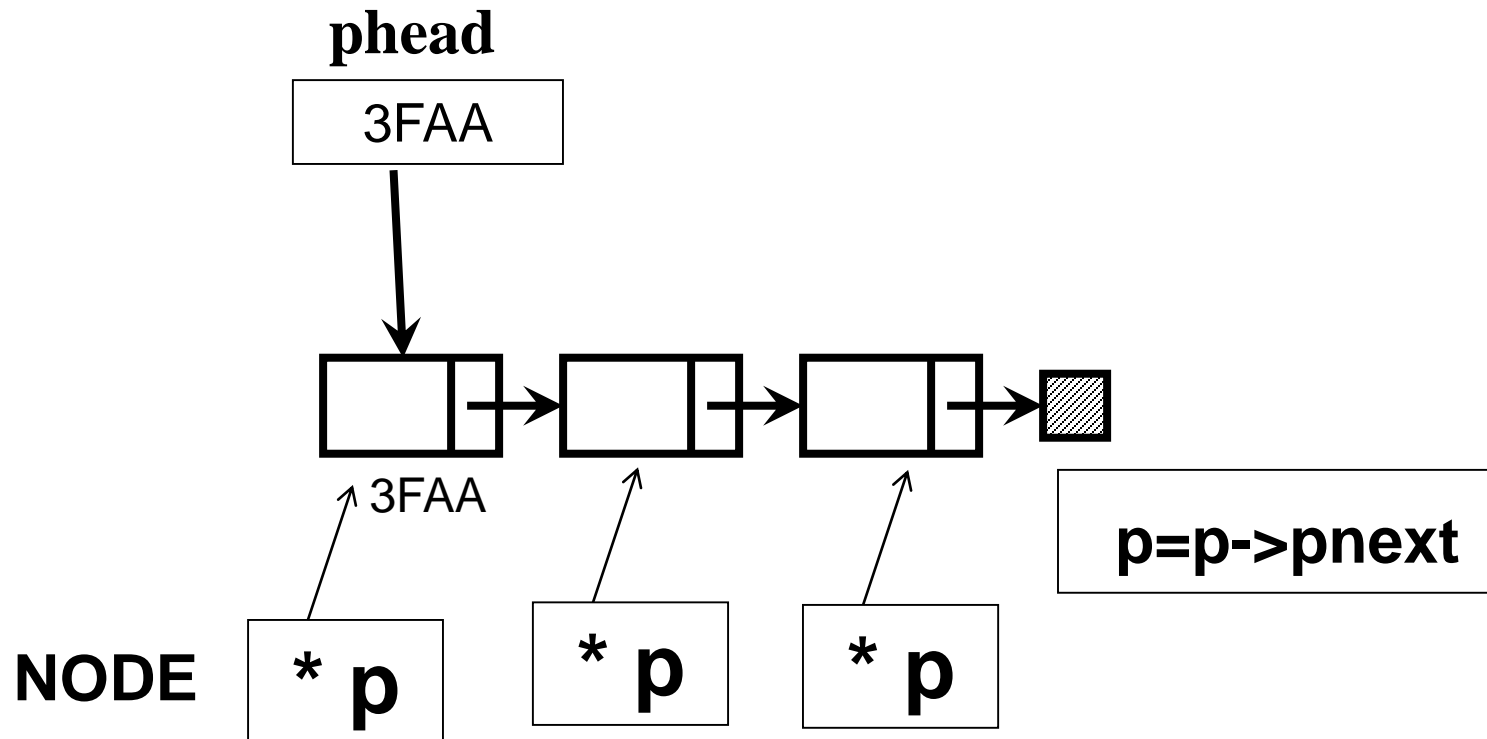
```
1. void Input (NODE * &phead)
2. {
3.     Init (phead) ;
4.     do {
5.         printf ("Nhap gia tri x: ");
6.         scanf ("%d", &x) ;
7.         NODE*p=GetNode (x) ;
8.         AddHead (phead,p) ;
9.         ///
10.    } while (x!=0)
```

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

```
1. void Input (NODE * &phead)
2. {   int x;
3.   Init (phead) ;
4.   do {
5.       printf ("Nhap gia tri x: ");
6.       scanf ("%d", &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> (NODE* 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> (NODE *phead)
2. {
3.     ...
4.     NODE *p=phead;
5.     while (p!=NULL)
6.     {
7.         ...
8.         p=p->pNext;
9.     }
```

8. DUYỆT TUẦN TỰ DANH SÁCH LIÊN KẾT ĐƠN

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

8. DUYỆT TUẦN TỰ DANH SÁCH LIÊN KẾT ĐƠN

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


8. DUYỆT TUẦN TỰ DANH SÁCH LIÊN KẾT ĐƠN

```
1. long  tinhhtong( NODE *phead )
2. {    long s=0;
3.     for (NODE*p=phead; p!=NULL; p=p->pnext )
4.
5.         s=s+p->info;
6.     return s;
7.
8. }
```

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

```
1.int Tong(NODE *phead)
2.{
3.    int 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.}
```

8. DUYỆT TUẦN TỰ DANH SÁCH LIÊN KẾT ĐƠN

```
1. long  tinhhtong( NODE *phead )
2. {    long s=0;
3.     for (NODE*p=phead; p!=NULL; p=p->pnext )
4.
5.         1. if ( p->info % 2 == 0 )
6.             s=s+p->info;
7.     return s;
8. }
```

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

```
1.int  TongLe (NODE *phead)
2.{
3.    int 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.

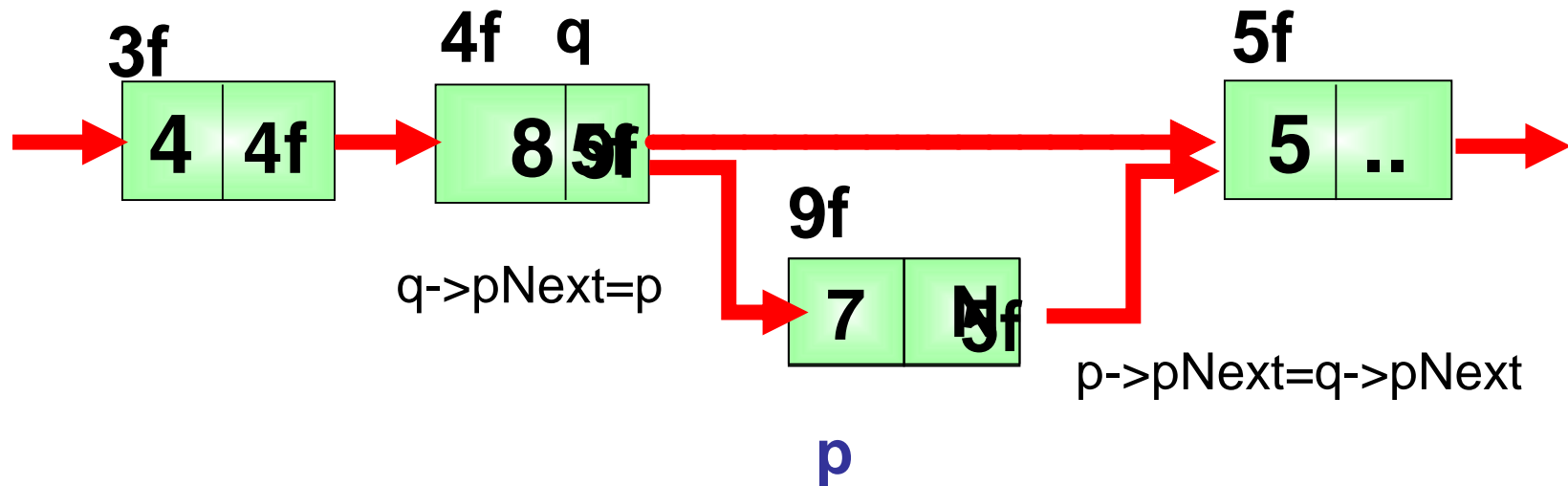
```
1.int Timmax(NODE *phead)
2.{
3.    int max = phead->info ;
4.    NODE * p = phead;
5.    while (p!=NULL)
6.    {
7.        if( p->info > max )
8.            max= p->info;
9.        p=p->pNext;
10.    }
11.    return max;
12.}
```

8. DUYỆT TUẦN TỰ DANH SÁCH LIÊN KẾT ĐƠN

```
1.int timmax( NODE *phead )
2.{   int max= phead->info ;
3.   for (NODE*p=phead->pnext;p!=NULL;p=p-
      >pnext )
4.       if ( p->info > max )
5.           max= p->info;
6.   return max ;
7.
8. }
```

9. CHƯƠNG TRÌNH ĐẦU TIÊN DSLK ĐƠN

Thêm phần tử p vào sau phần tử q



Thêm phần tử p vào sau phần tử q

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

Bắt đầu:

Nếu ($q \neq \text{NULL}$) thì

B1: $p \rightarrow \text{pNext} = q \rightarrow \text{pNext}$

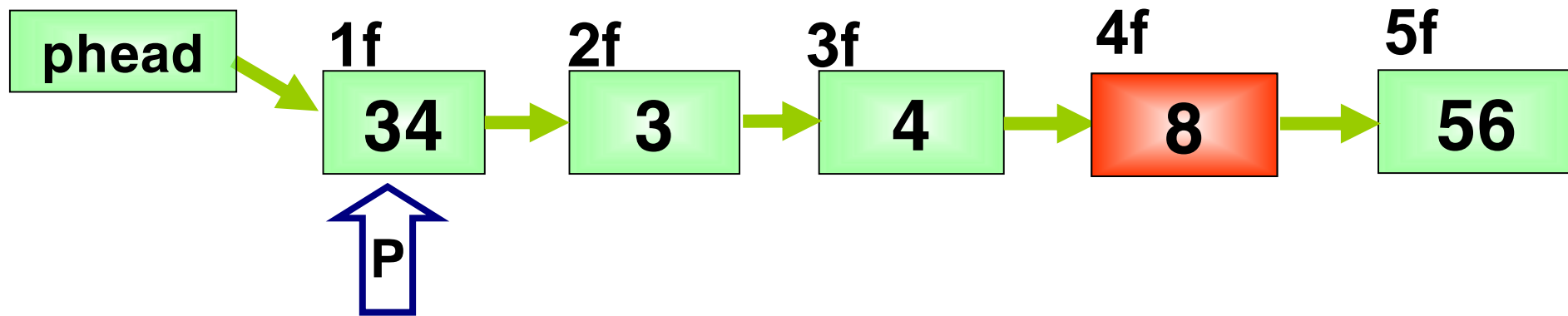
B2: $q \rightarrow \text{pNext} = p$

Ngược lại: Thêm vào đầu danh sách

Cài đặt thuật toán

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

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 $p \neq NULL$ và $p \rightarrow info \neq x$
 - $p = p \rightarrow pNext$;// xét phần tử kế
- Bước 3:
 - + Nếu $p \neq NULL$ 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

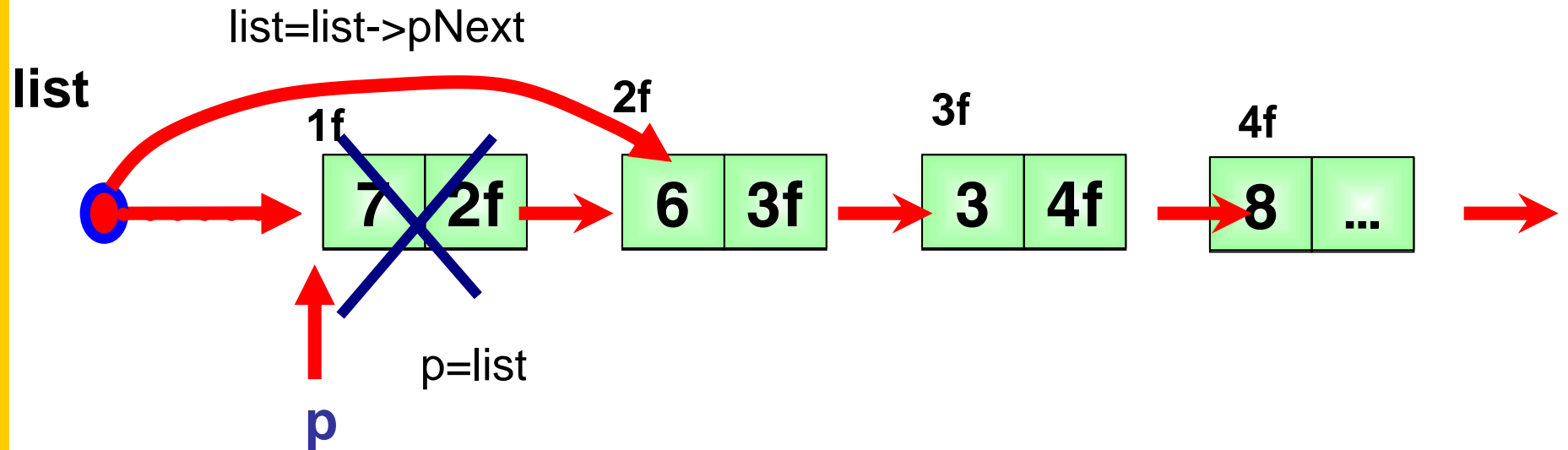
Hàm tìm phần tử có info = x, hàm trả về địa chỉ của nút có info = x, ngược lại hàm trả về NULL

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

Cài đặt thuật toán

```
void themxsauby(NODE *&phead)
{
    int x,y;
    printf (" nhập gia tri can them:");
    scanf("%d",&x);
    NODE *p=getnode(x);
    printf (" nhập gia tri muon them vao sau :");
    scanf("%d",&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 trí cần hủy
 - ↪ Hủy phần tử đứng đầu danh sách
 - ↪ Hủy phần tử có khoá bằng x
 - ↪ Hủy 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:

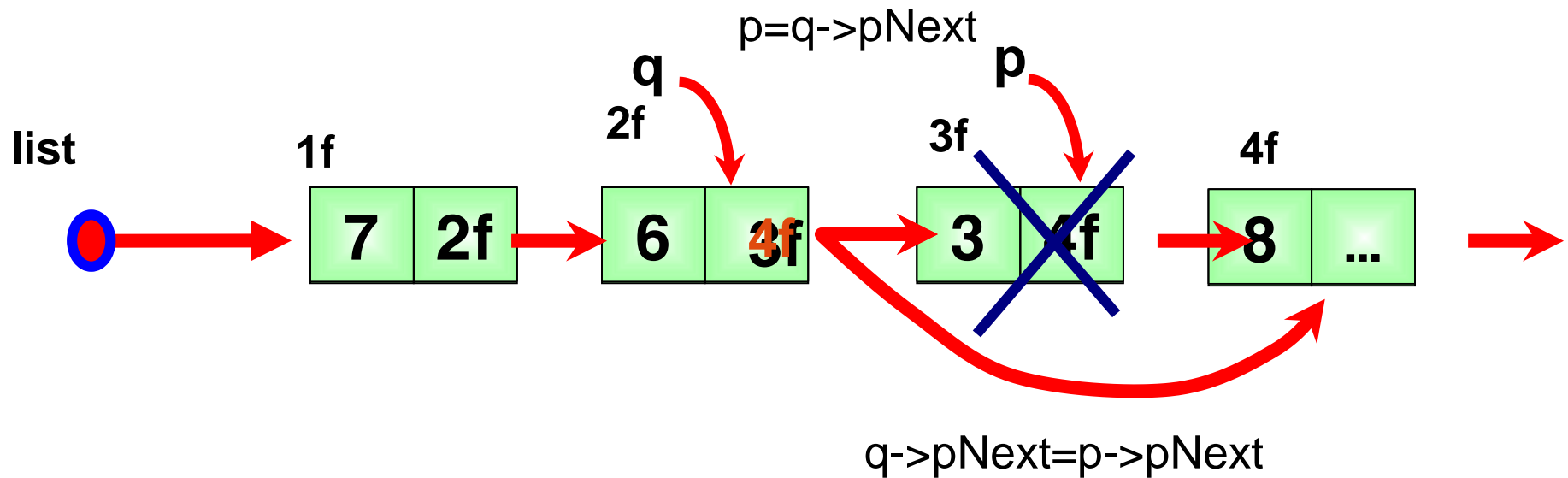
↪ Nếu ($l \neq \text{NULL}$) thì

- B1: $p = \text{pHead}$ //cho p bằng phần tử cần xóa
- B2: $l = l \rightarrow \text{pNext}$
- $\text{delete}(p)$

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



Hủy phần tử sau phần tử q trong List

Bắt đầu

Nếu ($q \neq \text{NULL}$) thì // *q tồn tại trong List*

B1: $p = q \rightarrow \text{pNext};$ // *p là phần tử cần hủy*

B2: Nếu ($p \neq \text{NULL}$) thì // *q không phải là phần tử cuối*

+ $q \rightarrow \text{pNext} = p \rightarrow \text{pNext};$ // *tách p ra khỏi xâu*

+ $\text{delete } p;$ // *hủy p*

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 $(p \neq \text{NULL})$ thì // tìm thấy phần tử có khoá bằng x

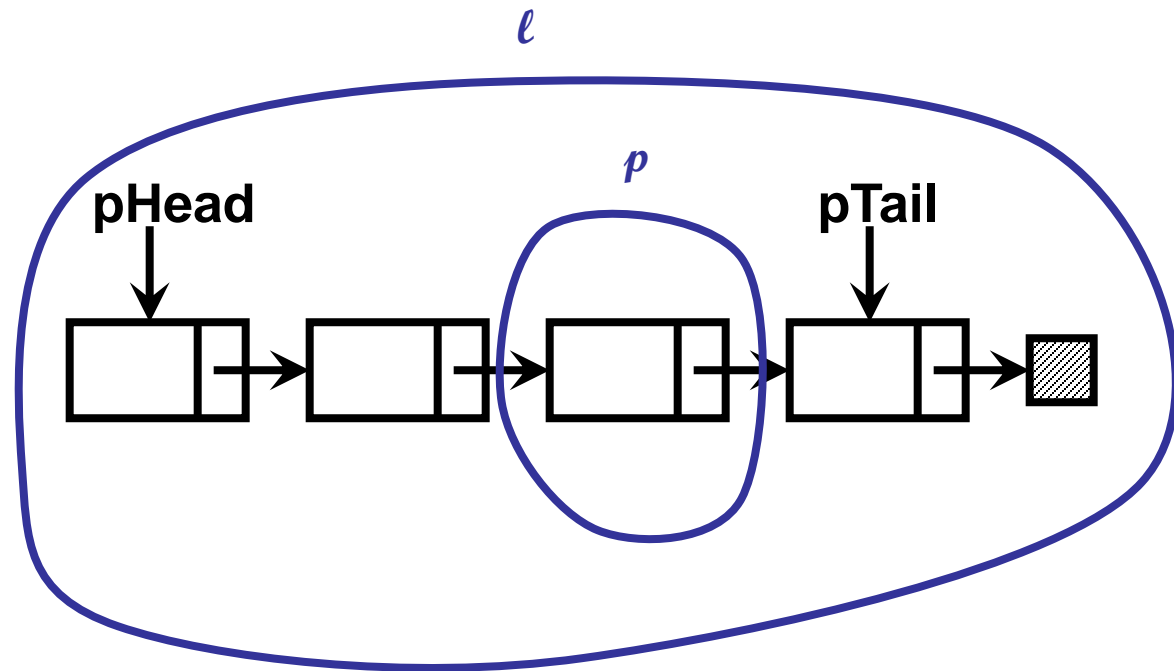
Hủy p ra khỏi danh sách bằng cách
hủy phần tử đứng sau q

Ngược lại

Báo không tìm thấy phần tử có khoá

Xóa 1 phần tử trong DSLK đơn

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



Xóa 1 phần tử trong DSLK đơn

```
int xoa1pt( list &l , NODE *p)
{
    if(l.phead==NULL) return NULL;
    if(l.phead==p)  xoahead(l);
    NODE *q= Before(l,p);
    q->pnext=p->pnext;
    p->pnext=NULL;
    delete p;
    return 0;
}
```

Tìm 1 node trong danh sách liên kết đơn.

```
NODE* Before( NODE *phead, 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 , int 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;
  } }
```