# Chương 11
# CON TRỎ – PHẦN 04 – DSLK ĐƠN

1. TS. Nguyễn Tấn Trần Minh Khang
2. ThS. Võ Duy Nguyên
3. ThS. Nguyễn Hoàng Ngân

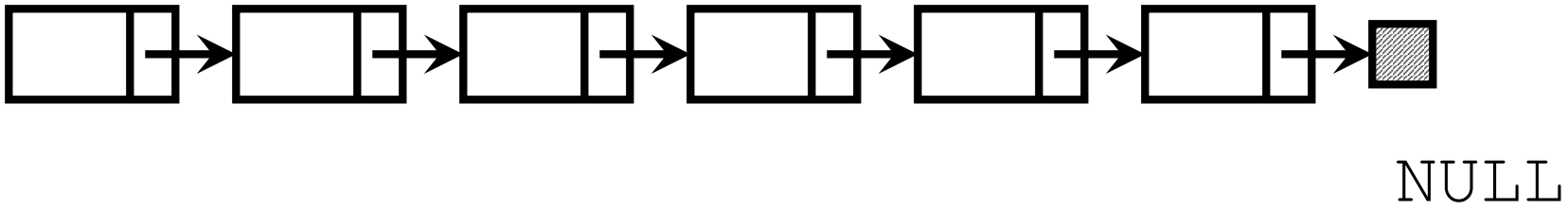
# GHI NHỚ

# Ghi nhớ

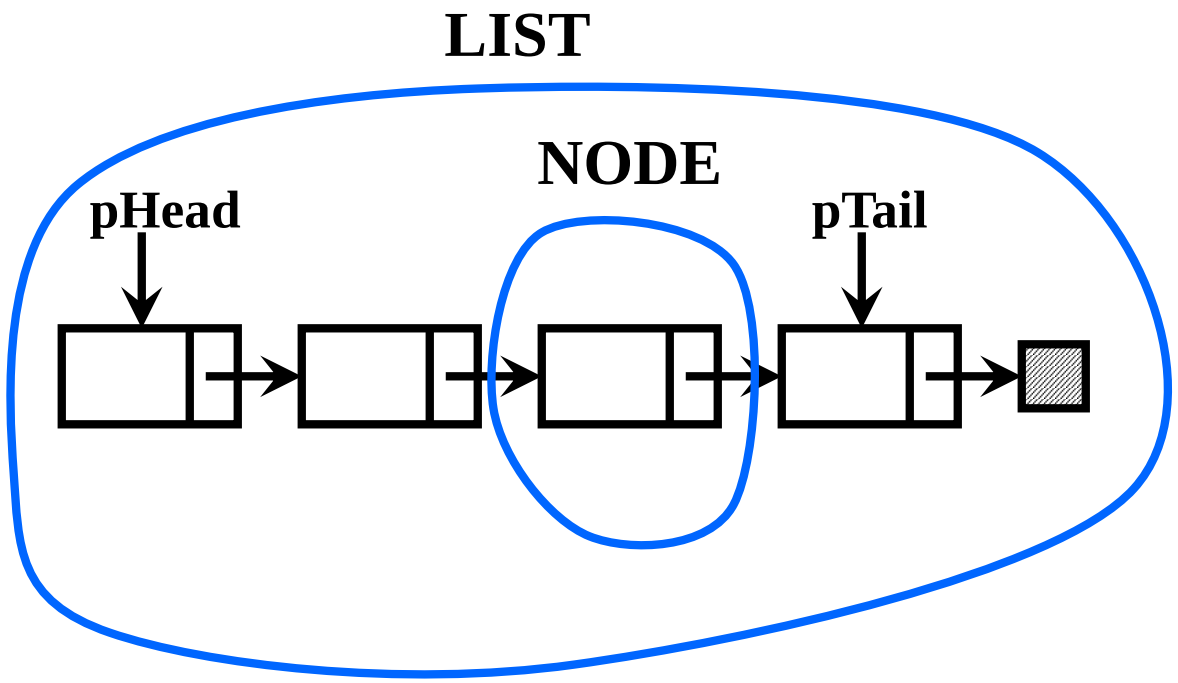
Miền giá trị
của một biến con trỏ là
địa chỉ ô nhớ.

# HÌNH ẢNH DANH SÁCH LIÊN KẾT ĐƠN

# Hình ảnh danh sách liên kết đơn

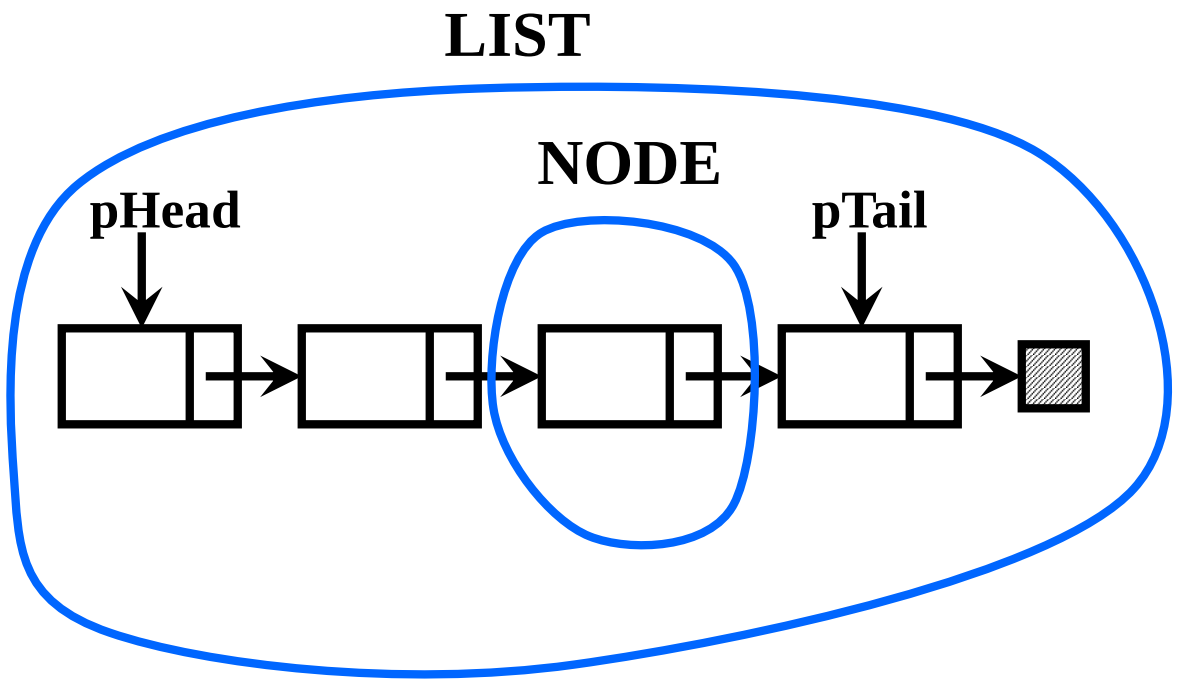


# CẤU TRÚC DỮ LIỆU DSLK ĐƠN



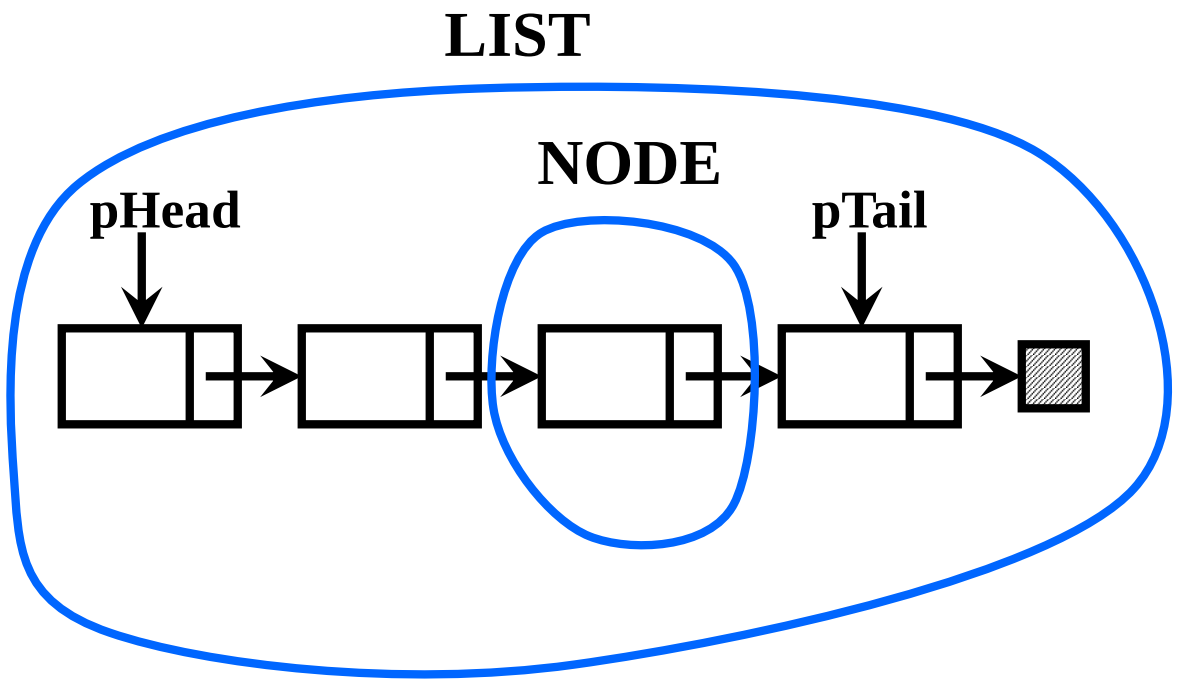

– Khai báo cấu trúc dữ liệu trừu tượng cho danh sách liên kết đơn (hai trỏ)

```
11.struct node
12.{
13.|    KDL info;
14.|    struct node *pNext;
15.};
16.typedef struct node NODE;
17.struct list
18.{
19.|    NODE *pHead;
20.|    NODE *pTail;
21.};
22.typedef struct list LIST;
```



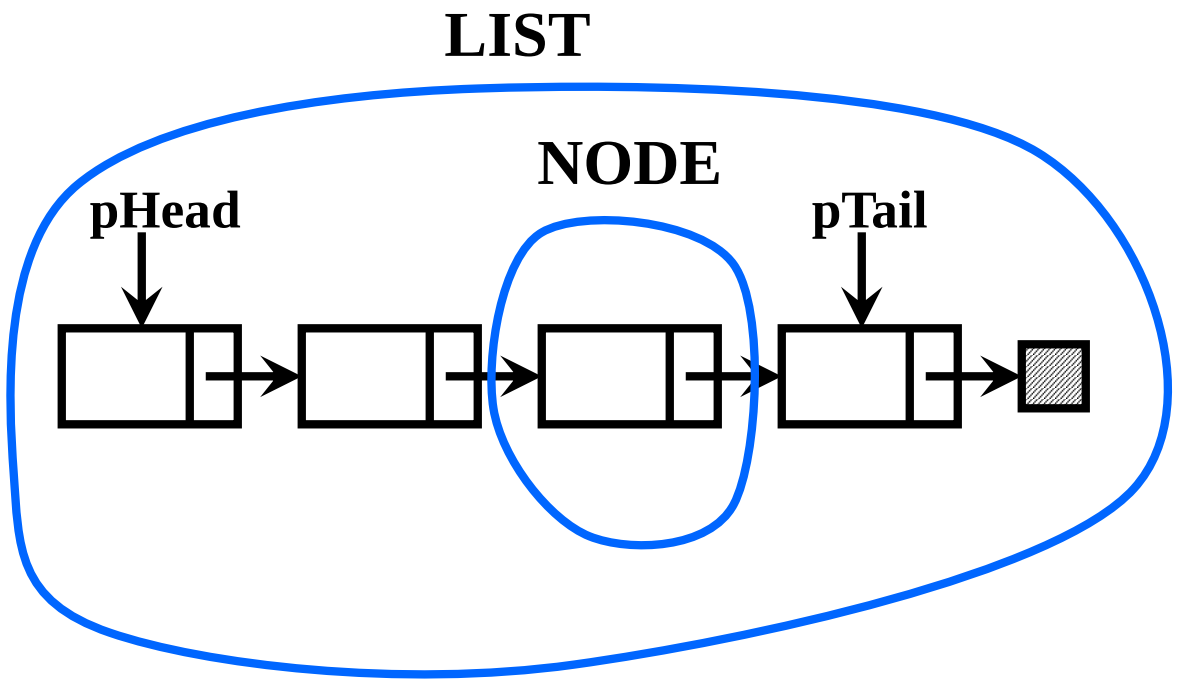

– Ví dụ 1: Khai báo cấu trúc dữ liệu cho danh sách liên kết đơn các số nguyên.

```
11.struct node
12.{
13.|    int info;
14.|    struct node *pNext;
15.};
16.typedef struct node NODE;
17.struct list
18.{
19.|    NODE *pHead;
20.|    NODE *pTail;
21.};
22.typedef struct list LIST;
```



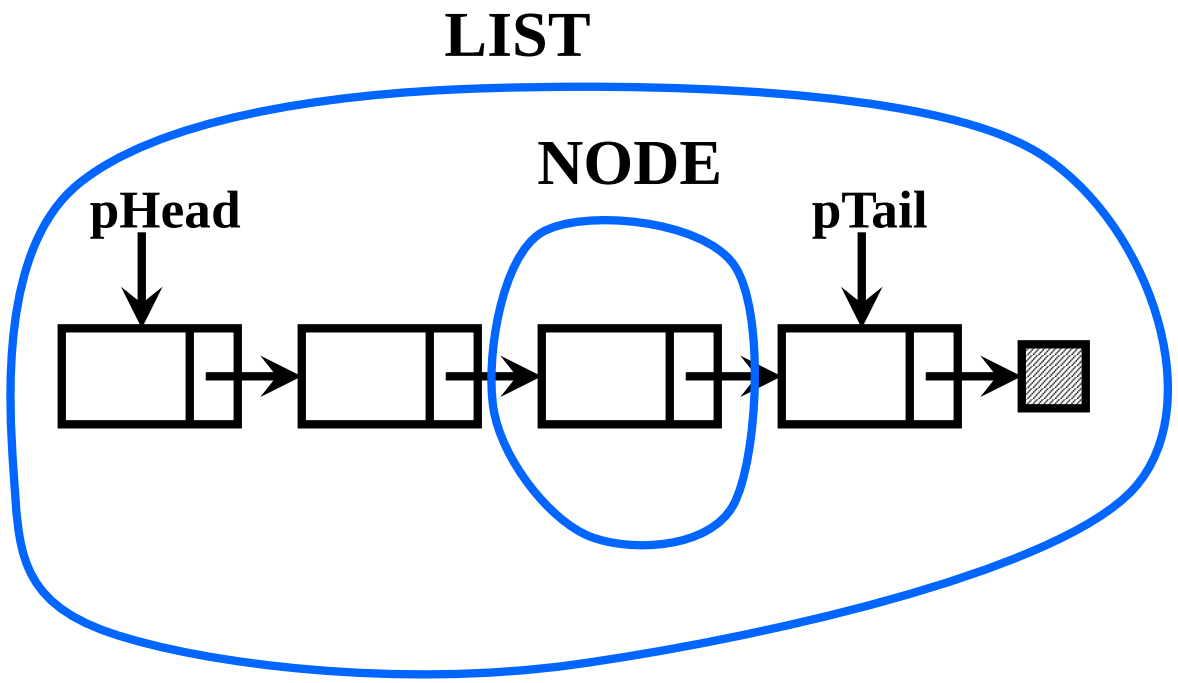

– Ví dụ 2: Khai báo cấu trúc dữ liệu cho danh sách liên kết đơn các số thực.

```
11.struct node
12.{
13.|    float info;
14.|    struct node *pNext;
15.};
16.typedef struct node NODE;
17.struct list
18.{
19.|    NODE *pHead;
20.|    NODE *pTail;
21.};
22.typedef struct list LIST;
```



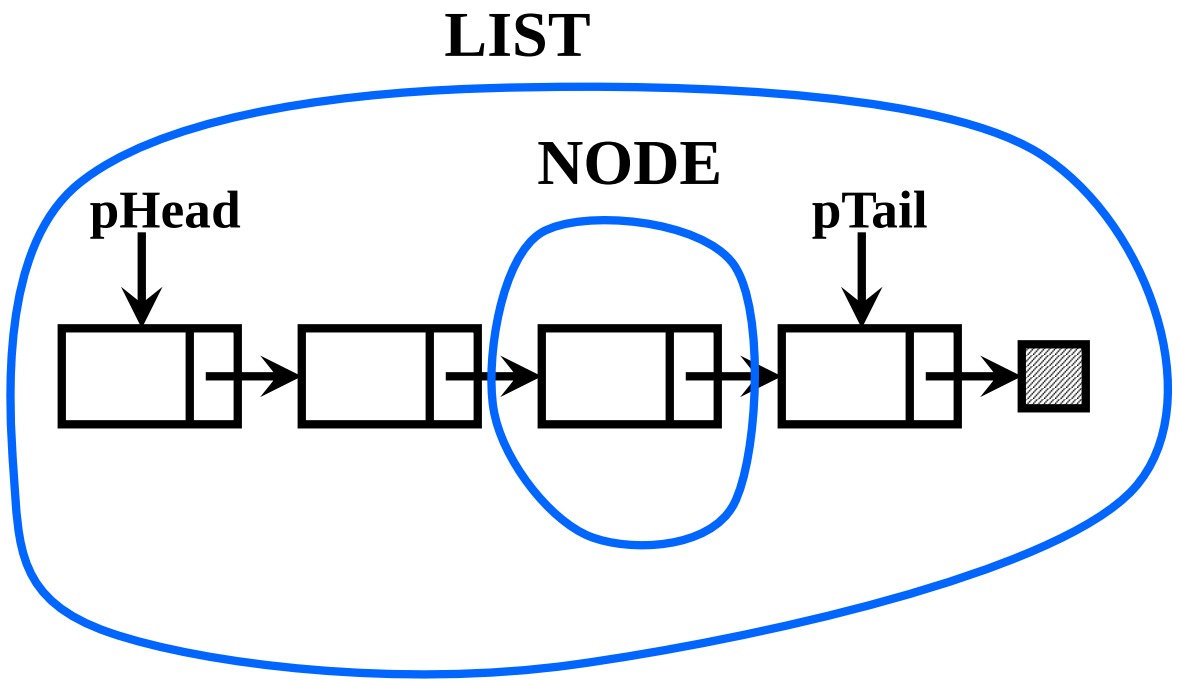

– Ví dụ 3: Khai báo cấu trúc dữ liệu cho danh sách liên kết đơn các phân số.

```
11.struct phanso
12.{
13.|    int tu;
14.|    int mau;
15.};
16.typedef struct phanso PHANSO;
17.struct node
18.{
19.|    PHANSO info;
20.|    struct node *pNext;
21.};
22.typedef struct node NODE;
23.struct list
24.{
25.|    NODE *pHead;
26.|    NODE *pTail;
27.};
28.typedef struct list LIST;
```



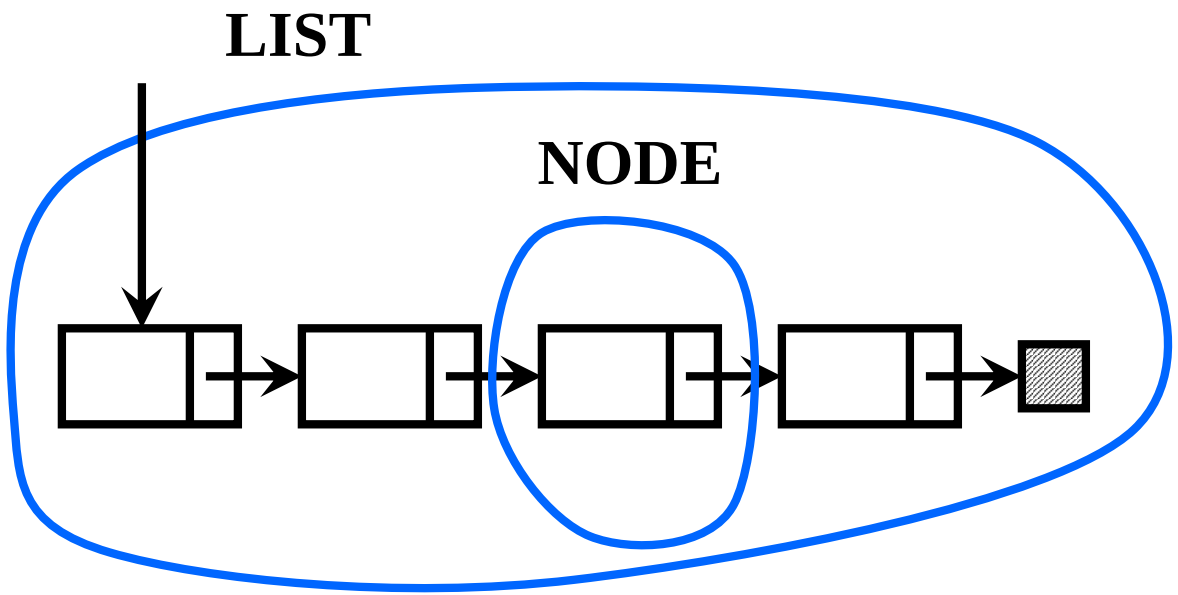

– Ví dụ 4: Khai báo cấu trúc dữ liệu cho danh sách liên kết đơn các số phức.

```
11.struct sophuc
12.{
13.|    float thuc;
14.|    float ao;
15.};
16.typedef struct sophuc SOPHUC;
17.struct node
18.{
19.|    SOPHUC info;
20.|    struct node *pNext;
21.};
22.typedef struct node NODE;
23.struct list
24.{
25.|    NODE *pHead;
26.|    NODE *pTail;
27.};
28.typedef struct list LIST;
```



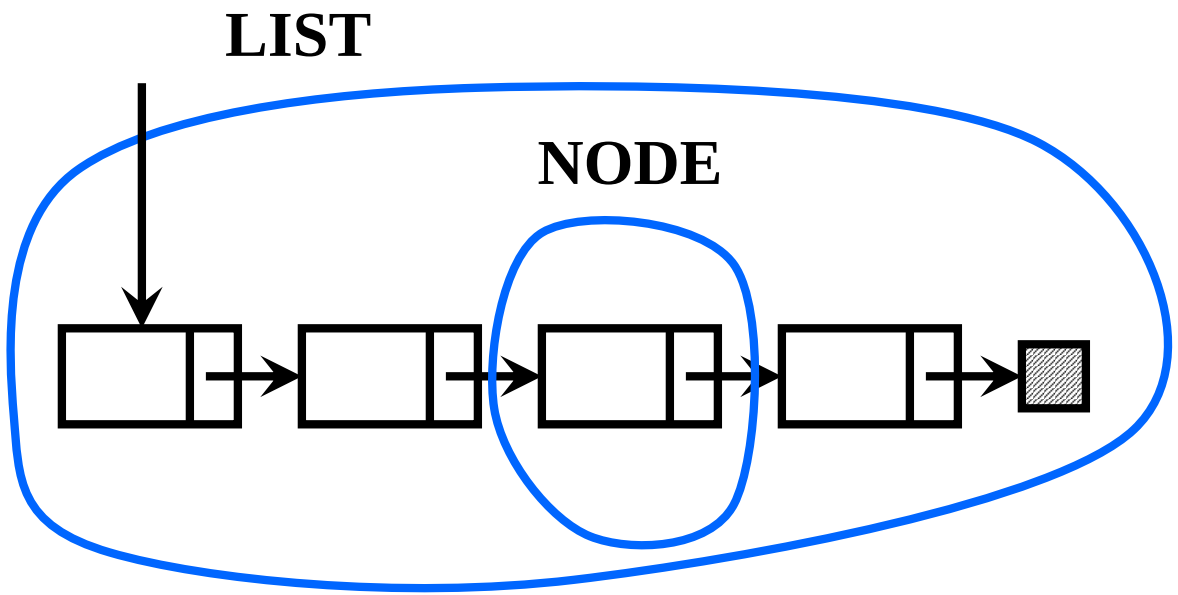

– Ví dụ 5: Khai báo cấu trúc dữ liệu cho dslk đơn tọa độ điểm trong mp Oxy.

```
11.struct diem
12.{
13.|    float x;
14.|    float y;
15.};
16.typedef struct diem DIEM;
17.struct node
18.{
19.|    DIEM info;
20.|    struct node *pNext;
21.};
22.typedef struct node NODE;
23.struct list
24.{
25.|    NODE *pHead;
26.|    NODE *pTail;
27.};
28.typedef struct list LIST;
```



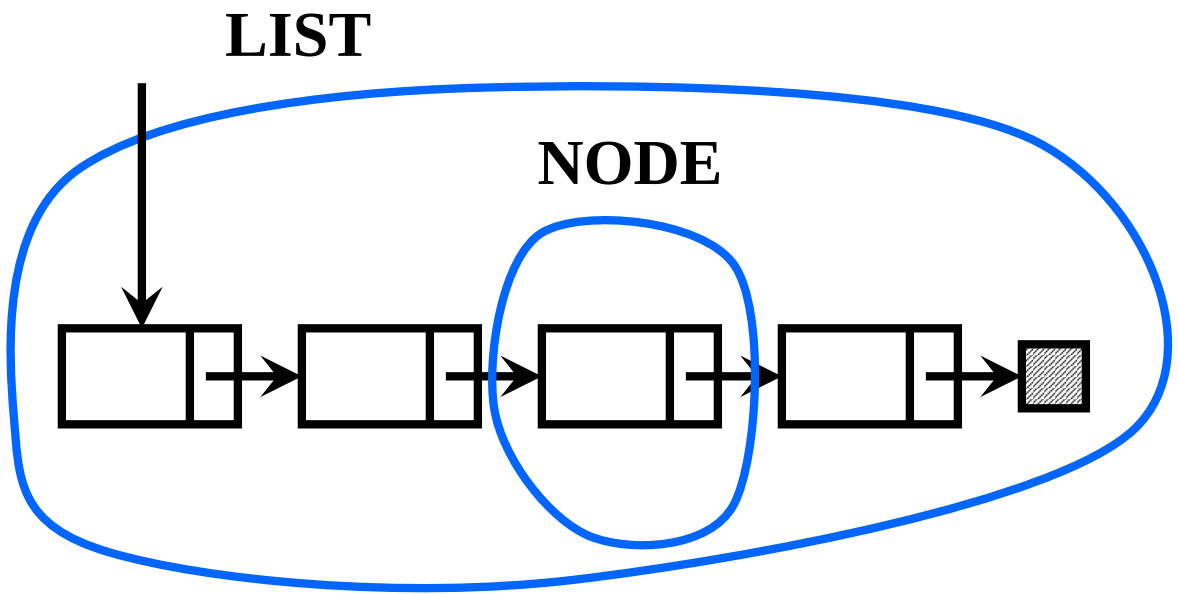

– Khai báo cấu trúc dữ liệu trừu tượng cho danh sách liên kết đơn (một trỏ).

```
11.struct node
12.{
13.|    KDL info;
14.|    struct node *pNext;
15.};
16.typedef struct node NODE;
17.typedef NODE* LIST;
```



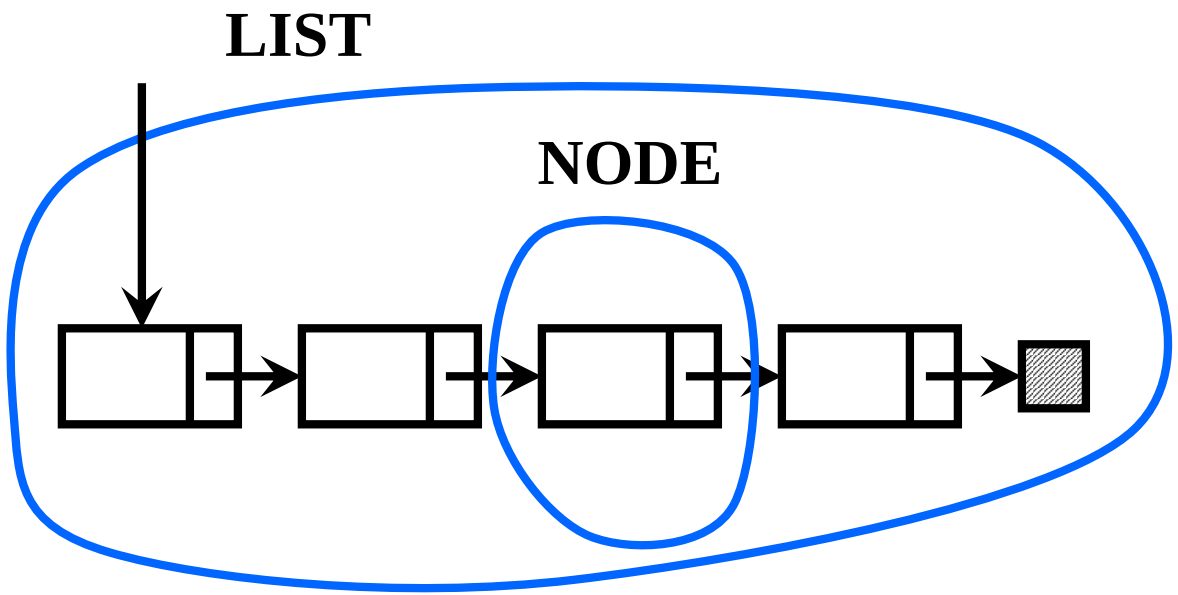

– Ví dụ 1: Khai báo cấu trúc dữ liệu cho danh sách liên kết đơn các số nguyên.

```
11.struct node
12.{
13.|    int info;
14.|    struct node *pNext;
15.};
16.typedef struct node NODE;
17.typedef NODE* LIST;
```



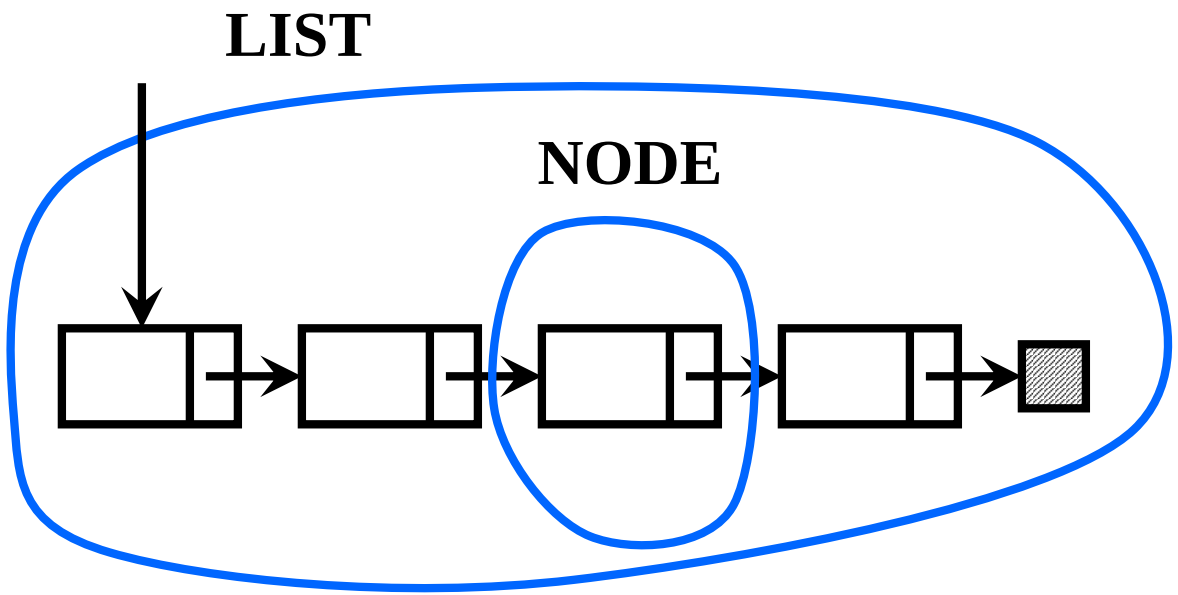

– Ví dụ 2: Khai báo cấu trúc dữ liệu cho danh sách liên kết đơn các số thực.

```
11.struct node
12.{
13.|    float info;
14.|    struct node *pNext;
15.};
16.typedef struct node NODE;
17.typedef NODE* LIST;
```



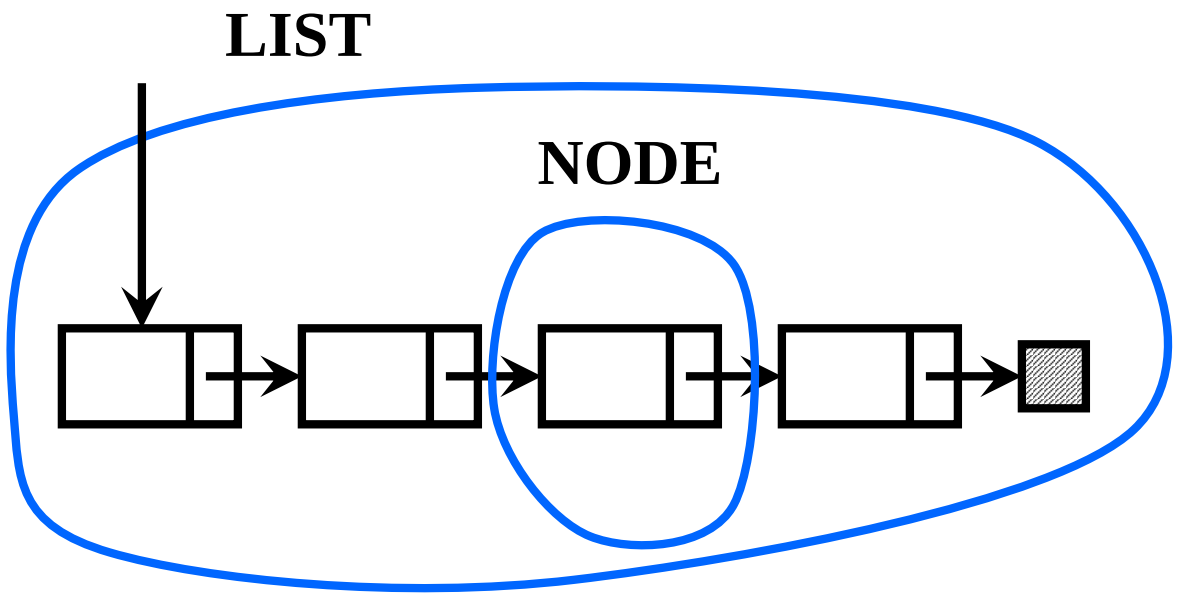

– Ví dụ 3: Khai báo cấu trúc dữ liệu cho danh sách liên kết đơn các phân số.

```
11.struct phanso
12.{
13.|    int tu;
14.|    int mau;
15.};
16.typedef struct phanso PHANSO;
17.struct node
18.{
19.|    PHANSO info;
20.|    struct node *pNext;
21.};
22.typedef struct node NODE;
23.typedef NODE* LIST;
```




– Ví dụ 4: Khai báo cấu trúc dữ liệu cho danh sách liên kết đơn các số phức.

```
11.struct sophuc
12.{
13.|    float thuc;
14.|    float ao;
15.};
16.typedef struct sophuc SOPHUC;
17.struct node
18.{
19.|    SOPHUC info;
20.|    struct node *pNext;
21.};
22.typedef struct node NODE;
23.typedef NODE* LIST;
```




– Ví dụ 5: Khai báo cấu trúc dữ liệu cho dslk đơn tọa độ điểm trong mp Oxy.

```
11.struct diem
12.{
13.|    float x;
14.|    float y;
15.};
16.typedef struct diem DIEM;
17.struct node
18.{
19.|    DIEM info;
20.|    struct node *pNext;
21.};
22.typedef struct node NODE;
23.typedef NODE* LIST;
```

# KHỞI TẠO DANH SÁCH LIÊN KẾT ĐƠN

# 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 (hai trỏ)

```
11.void Init(LIST &ℓ)
12.{
13.    ℓ.pHead = NULL;
14.    ℓ.pTail = NULL;
15.}
```

# 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 (một trỏ)

```
11.void Init(LIST &ℓ)
12.{
13.    ℓ = NULL;
14.}
```

# KIỂM TRA DSLK ĐƠN RỖNG

# Kiểm tra dslk đơn rỗng

- Khái niệm: Kiểm tra danh sách liên kết đơn rỗng là hàm trả về giá trị 1 khi danh sách rỗng. Trong tình huống danh sách không rỗng thì hàm sẽ trả về giá trị 0.
- Định nghĩa hàm (hai trỏ)

```
11.int IsEmpty(LIST ℓ)
12.{
13.    if(ℓ.pHead==NULL)
14.        return 1;
15.    return 0;
16.}
```

# Kiểm tra dslk đơn rỗng

- Khái niệm: Kiểm tra danh sách liên kết đơn rỗng là hàm trả về giá trị 1 khi danh sách rỗng. Trong tình huống danh sách không rỗng thì hàm sẽ trả về giá trị 0.
- Định nghĩa hàm (một trỏ)

```
11.int IsEmpty(LIST ℓ)
12.{
13.    if(ℓ==NULL)
14.        return 1;
15.    return 0;
16.}
```

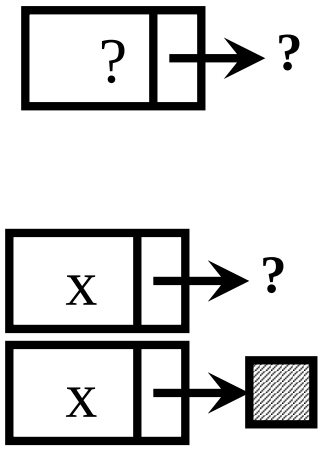
# TẠO NODE CHO DSLK ĐƠN

# Tạo node cho dslk 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.

- Định nghĩa hàm trừu tượng

```
11.NODE* GetNode(KDL x)
12.{
13.    NODE *p=new NODE;
14.    if(p==NULL)
15.        return NULL;
16.    p->info = x;
17.    p->pNext = NULL;
18.    return p;
19.}
```



# Tạo node cho dslk đơn

- Phân tích câu lệnh dòng 11

```
18.NODE* GetNode(KDL x)
```

- Tên hàm tạo node cho dslk đơn là `GetNode`.
- Có một tham số đầu vào, tên tham số là `x`, tham số là tham trị.
- Kiểu dữ liệu trả về của hàm `GetNode` là con trỏ kiểu cấu trúc `NODE`.
- Về mặt bản chất hàm `GetNode` sẽ trả về một địa chỉ ô nhớ.

- Phân tích câu lệnh dòng 18

```
18.     return p;
```

- Kết thúc lời gọi hàm và trả về địa chỉ ô nhớ đang được lưu trong biến con trỏ `p`.
- Ý nghĩa của địa chỉ ô nhớ đang lưu biến con trỏ `p` xem ở slide ngay sau.

# Tạo node cho dslk đơn

Phân tích dòng lệnh 13.

```
13.NODE *p=new NODE;
```

- `p` là một biến con trỏ kiểu cấu trúc `NODE`.
- Miền giá trị của biến con trỏ `p` là địa chỉ ô nhớ.
- `new NODE` là xin cấp phát động một vùng nhớ có kích thước bằng kích thước của kiểu dữ liệu `NODE`.
- Nếu việc cấp phát thành công compiler sẽ trả về địa chỉ ô nhớ đầu tiên của vùng nhớ được cấp phát, địa chỉ ô nhớ này được gán cho biến con trỏ `p`.
- Nếu việc cấp phát thất bại, compiler sẽ trả về một địa chỉ đặc biệt là địa chỉ `NULL`, địa chỉ `NULL` này sẽ được gán cho biến con trỏ `p`.
- Như vậy, thông thường sau câu lệnh thứ 13, biến con trỏ `p` sẽ giữ địa chỉ ô nhớ đầu tiên của vùng nhớ có kích thước bằng kích thước của KDL `NODE`.

# Tạo node cho dslk đơn

- Ví dụ 1: Định nghĩa hàm tạo một NODE cho danh sách liên kết đơn các số nguyên để chứa thông tin đã được biết trước.

- Định nghĩa hàm

```
11.NODE* GetNode(int x)
12.{
13.    NODE *p=new NODE;
14.    if(p==NULL)
15.        return NULL;
16.    p->info = x;
17.    p->pNext = NULL;
18.    return p;
19.}
```

# Tạo node cho dslk đơn

- Ví dụ 2: Định nghĩa hàm tạo một NODE cho danh sách liên kết đơn các số thực để chứa thông tin đã được biết trước.

- Định nghĩa hàm

```
11.NODE* GetNode(float x)
12.{
13.    NODE *p=new NODE;
14.    if(p==NULL)
15.        return NULL;
16.    p->info = x;
17.    p->pNext = NULL;
18.    return p;
19.}
```

# Tạo node cho dslk đơn

- Ví dụ 3: Định nghĩa hàm tạo một NODE cho danh sách liên kết đơn các phân số để chứa thông tin đã được biết trước.

- Định nghĩa hàm

```
11.NODE* GetNode(PHANSO x)
12.{
13.    NODE *p=new NODE;
14.    if(p==NULL)
15.        return NULL;
16.    p->info = x;
17.    p->pNext = NULL;
18.    return p;
19.}
```

# Tạo node cho dslk đơn

- Ví dụ 4: Định nghĩa hàm tạo một NODE cho danh sách liên kết đơn các số phức để chứa thông tin đã được biết trước.

- Định nghĩa hàm

```
11.NODE* GetNode(SOPHUC x)
12.{
13.    NODE *p=new NODE;
14.    if(p==NULL)
15.        return NULL;
16.    p->info = x;
17.    p->pNext = NULL;
18.    return p;
19.}
```

# Tạo node cho dslk đơn

- Ví dụ 5: Định nghĩa hàm tạo một NODE cho danh sách liên kết đơn tọa độ các điểm trong mặt phẳng Oxy để chứa thông tin đã được biết trước.

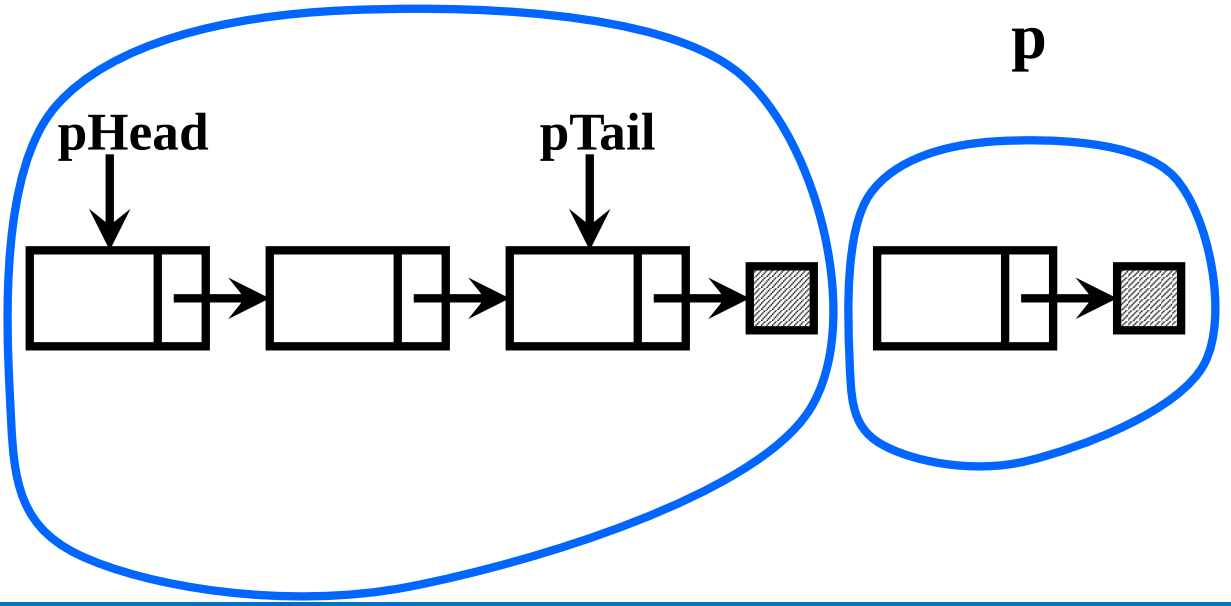
- Định nghĩa hàm

```
11.NODE* GetNode(DIEM PP)
12.{
13.    NODE *p=new NODE;
14.    if(p==NULL)
15.        return NULL;
16.    p->info = PP;
17.    p->pNext = NULL;
18.    return p;
19.}
```

# THÊM MỘT NODE VÀO CUỐI DSLK ĐƠN

# Thêm một node vào cuối dslk đơn

- Khái niệm: Thêm một node vào cuối danh sách liên kết đơn là gắn node đó vào cuối danh sách.
- Hình vẽ $\ell$

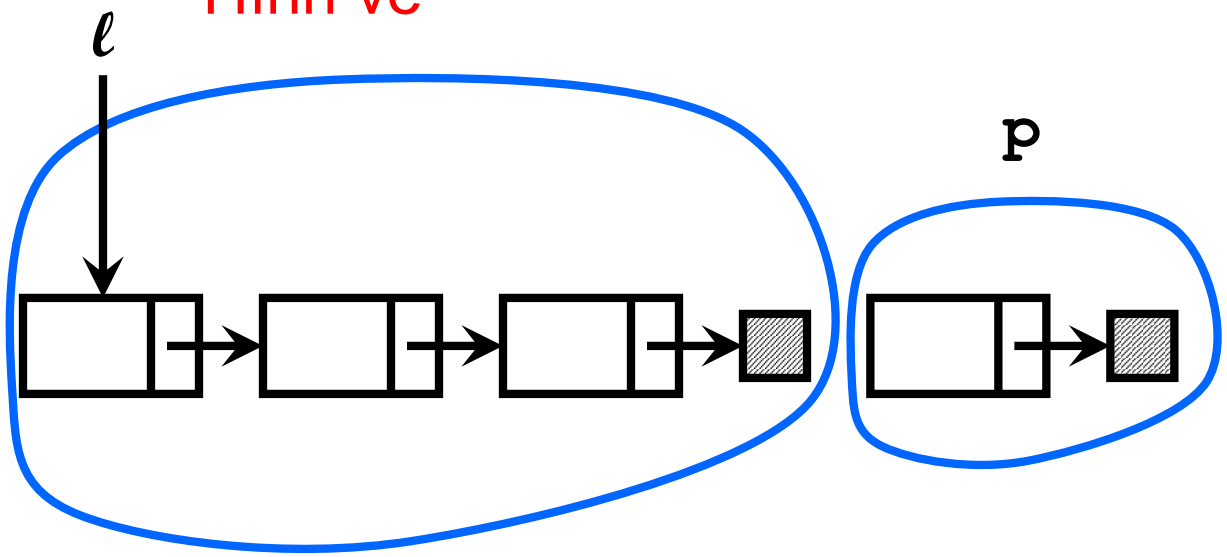



- Định nghĩa hàm (hai trỏ)

```
11.void AddTail(LIST&ℓ, NODE*p)
12.{
13.    if(ℓ.pHead==NULL)
14.        ℓ.pHead=ℓ.pTail=p;
15.    else
16.    {
17.        ℓ.pTail->pNext = p;
18.        ℓ.pTail = p;
19.    }
20.}
```

# Thêm một node vào cuối dslk đơn

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




- Định nghĩa hàm (một trỏ)

```
11.void AddTail(LIST &l,NODE*p)
12.{
13.    if(l==NULL)
14.        l = p;
15.    else
16.    {
17.        NODE *q=l;
18.        while(q->pNext)
19.            q = q->pNext;
20.        q->pNext = p;
21.    }
22.}
```

# NHẬP TỪ BÀN PHÍM DSLK ĐƠN

# Nhập từ bàn phím dslk đơn

- Khái niệm: Nhập từ bàn phím danh sách liên kết đơn là lần lượt nhập các thông tin của từng node trong danh sách liên kết đơn.

- Định nghĩa hàm trừu tượng

```
11.void Input(LIST&ℓ)
12.{
13.|    int n;
14.|    cout<<"Nhap n: ";
15.|    cin>>n;
16.|    Init(ℓ);
17.|    for(int i=1;i<=n;i++)
18.|    {
19.|    |    KDL x;
20.|    |    Nhap(x);
21.|    |    NODE*p = GetNode(x);
22.|    |    if(p!=NULL)
23.|    |        AddTail(ℓ,p);
24.|    }
25.}
```

# Nhập từ bàn phím dslk đơn

- Ví dụ 1: Nhập danh sách liên kết đơn các số nguyên.

- Định nghĩa hàm

```
11.void Input(LIST&ℓ)
12.{
13.     int n;
14.     cout<<"Nhap n: ";
15.     cin>>n;
16.     Init(ℓ);
17.     for(int i=1;i<=n;i++)
18.     {
19.         int x;
20.         cin>>x;
21.         NODE*p = GetNode(x);
22.         if(p!=NULL)
23.             AddTail(ℓ,p);
24.     }
25.}
```

# Nhập từ bàn phím dslk đơn

- Ví dụ 2: Nhập danh sách liên kết đơn các số thực.

- Định nghĩa hàm

```
11.void Input(LIST&ℓ)
12.{
13.     int n;
14.     cout<<"Nhap n: ";
15.     cin>>n;
16.     Init(ℓ);
17.     for(int i=1;i<=n;i++)
18.     {
19.         float x;
20.         cin>>x;
21.         NODE*p = GetNode(x);
22.         if(p!=NULL)
23.             AddTail(ℓ,p);
24.     }
25.}
```

# Nhập từ bàn phím dslk đơn

- Ví dụ 3: Nhập danh sách liên kết đơn các phân số.
- Định nghĩa hàm

```
11.void Nhap(PHANSO &x)
12.{
13.|    cout<<"Nhap tu:";
14.|    cin>>x.tu;
15.|    cout<<"Nhap mau:";
16.|    cin>>x.mau;
17.}
```

- Định nghĩa hàm

```
11.void Input(LIST&ℓ)
12.{
13.|    int n;
14.|    cout<<"Nhap n: ";
15.|    cin>>n;
16.|    Init(ℓ);
17.|    for(int i=1;i<=n;i++)
18.|    {
19.|        PHANSO x;
20.|        Nhap(x);
21.|        NODE*p = GetNode(x);
22.|        if(p!=NULL)
23.|            AddTail(ℓ,p);
24.|    }
25.}
```

# Nhập từ bàn phím dslk đơn

- Ví dụ 4: Nhập danh sách liên kết đơn các số phức.
- Định nghĩa hàm

```
11.void Nhap(SOPHUC &x)
12.{
13.|    cout<<"Nhap thuc:";
14.|    cin>>x.thuc;
15.|    cout<<"Nhap ao:";
16.|    cin>>x.ao;
17.}
```

- Định nghĩa hàm

```
11.void Input(LIST&ℓ)
12.{
13.|    int n;
14.|    cout<<"Nhap n: ";
15.|    cin>>n;
16.|    Init(ℓ);
17.|    for(int i=1;i<=n;i++)
18.|    {
19.|        |   SOPHUC x;
20.|        |   Nhap(x);
21.|        |   NODE*p = GetNode(x);
22.|        |   if(p!=NULL)
23.|        |       AddTail(ℓ,p);
24.|    }
25.}
```

# Nhập từ bàn phím dslk đơn

- Ví dụ 5: Nhập danh sách liên kết đơn tọa độ các điểm trong mặt phẳng Oxy.
- Định nghĩa hàm

```
11.void Nhap(DIEM &P)
12.{
13.|    cout<<"Nhap x:";
14.|    cin>>P.x;
15.|    cout<<"Nhap y:";
16.|    cin>>P.y;
17.}
```

- Định nghĩa hàm

```
11.void Input(LIST &ℓ)
12.{
13.|    int n;
14.|    cout<<"Nhap n: ";
15.|    cin>>n;
16.|    Init(ℓ);
17.|    for(int i=1;i<=n;i++)
18.|    {
19.|        |   DIEM P;
20.|        |   Nhap(P);
21.|        |   NODE*p = GetNode(P);
22.|        |   if(p!=NULL)
23.|        |       AddTail(ℓ,p);
24.|    }
25.}
```

# DUYỆT TUẦN TỰ DSLK ĐƠN

# Duyệt tuần tự dslk đơn

- Khái niệm: duyệt danh sách liên kết đơn là thăm qua tất cả các node mỗi node một lần.
- Ở đây ta sử dụng một con trỏ có kiểu dữ liệu NODE lần lượt giữ địa chỉ các NODE trong danh sách liên kết đơn.

- Định nghĩa hàm trừu tượng (hai trỏ)

```
11.KDL <Tên Hàm>(LIST ℓ)
12.{
13.|    ...
14.|    NODE *p = ℓ.pHead;
15.|    while(p!=NULL)
16.|    {
17.|    |   ...
18.|    |   p = p->pNext;
19.|    }
20.|    ...
21.}
```

# Duyệt tuần tự dslk đơn

- Khái niệm: duyệt danh sách liên kết đơn là thăm qua tất cả các node mỗi node một lần.
- Ở đây ta sử dụng một con trỏ có kiểu dữ liệu NODE lần lượt giữ địa chỉ các NODE trong danh sách liên kết đơn.

- Định nghĩa hàm trừu tượng (một trỏ)

```
11.KDL <Tên Hàm>(LIST ℓ)
12.{
13.|    ...
14.|    NODE *p = ℓ;
15.|    while(p!=NULL)
16.|    {
17.|    |   ...
18.|    |   p = p->pNext;
19.|    }
20.|    ...
21.}
```

# Duyệt tuần tự dslk đơn

- Ví dụ 1: Định nghĩa hàm tính tổng các số lẻ trong dslk đơn các số nguyên.

```
11.struct node
12.{
13.|    int info;
14.|    struct node *pNext;
15.};
16.typedef struct node NODE;
17.struct list
18.{
19.|    NODE *pHead;
20.|    NODE *pTail;
21.};
22.typedef struct list LIST;
```

- Định nghĩa hàm (hai trỏ)

```
11.int TongLe(LIST ℓ)
12.{
13.|    int s = 0;
14.|    NODE *p = ℓ.pHead;
15.|    while(p!=NULL)
16.|    {
17.|    |   if(p->info%2!=0)
18.|    |       s = s + p->info;
19.|    |   p = p->pNext;
20.|    }
21.|    return s;
22.}
```

# Duyệt tuần tự dslk đơn

– Ví dụ 1: Định nghĩa hàm tính tổng các số lẻ trong dslk đơn các số nguyên.

```
11.struct node
12.{
13.|    int info;
14.|    struct node *pNext;
15.};
16.typedef struct node NODE;
17.typedef NODE* LIST;
```

– Định nghĩa hàm (một trỏ)

```
11.int TongLe(LIST ℓ)
12.{
13.|    int s = 0;
14.|    NODE *p = ℓ;
15.|    while(p!=NULL)
16.|    {
17.|    |   if(p->info%2!=0)
18.|    |       s = s + p->info;
19.|    |   p = p->pNext;
20.|    }
21.|    return s;
22.}
```

# Duyệt tuần tự dslk đơn

– Ví dụ 2: Định nghĩa hàm xuất dslk đơn các số nguyên.

```
11.struct node
12.{
13.|    int info;
14.|    struct node *pNext;
15.};
16.typedef struct node NODE;
17.struct list
18.{
19.|    NODE *pHead;
20.|    NODE *pTail;
21.};
22.typedef struct list LIST;
```

– Định nghĩa hàm (hai trỏ)

```
11.void Xuat(LIST ℓ)
12.{
13.|    NODE *p = ℓ.pHead;
14.|    while(p!=NULL)
15.|    {
16.|    |   cout<<p->info;
17.|    |   p = p->pNext;
18.|    }
19.}
```

# Duyệt tuần tự dslk đơn

– Ví dụ 2: Định nghĩa hàm xuất dslk đơn các số nguyên.

```
11.struct node
12.{
13.|    int info;
14.|    struct node *pNext;
15.};
16.typedef struct node NODE;
17.typedef NODE* LIST;
```

– Định nghĩa hàm (một trỏ)

```
11.void Xuat(LIST ℓ)
12.{
13.|    NODE *p = ℓ;
14.|    while(p!=NULL)
15.|    {
16.|    |   cout<<p->info;
17.|    |   p = p->pNext;
18.|    }
19.}
```

# CHƯƠNG TRÌNH ĐẦU TIÊN DSLK ĐƠN

# Chương trình đầu tiên dslk đơn

- Bài toán: Viết chương trình thực hiện các yêu cầu sau:
  + Nhập dslk đơn các số nguyên.
  + Xuất dslk đơn.
  + Đếm số lượng giá trị lẻ có trong dslk đơn.
  + Tính tổng các giá trị trong dslk đơn.

# Chương trình đầu tiên dslk đơn hai trỏ

```
#include <iostream>
#include <iomanip>
using namespace std;
struct node
{
    int info;
    struct node *pNext;
};
typedef struct node NODE;
struct list
{
    NODE *pHead;
    NODE *pTail;
};
```

```
typedef struct list LIST;
//Khai báo hàm
void Init(LIST&);
int IsEmpty(LIST);
NODE* GetNode(int);
void AddTail(LIST&,NODE*);

void Input(LIST &);
void Output(LIST);

int DemLe(LIST);
int Tong(LIST);
```

# Chương trình đầu tiên dslk đơn hai trỏ

```
11.int main()
12.{
13.    LIST lst;
14.    Input(lst);
15.    cout<<"Danh sach lien ket don ban dau";
16.    Output(lst);
17.    int kq = DemLe(lst);
18.    cout<<"\nSo luong gia tri le trong danh sach: "<<kq;
19.    kq = Tong(lst);
20.    cout<<"\nTong cac gia tri trong danh sach: "<<kq;
21.    return 1;
22.}
```

# Chương trình đầu tiên dslk đơn hai trỏ

```
11.void Init(LIST &ℓ)
12.{
13.|    ℓ.pHead = NULL;
14.|    ℓ.pTail = NULL;
15.}
16.int IsEmpty(LIST ℓ)
17.{
18.|    if (ℓ.pHead==NULL)
19.|        return 1;
20.|    return 0;
21.}
```

```
11.NODE* GetNode(int x)
12.{
13.|    NODE *p=new NODE;
14.|    if(p==NULL)
15.|        return NULL;
16.|    p->info = x;
17.|    p->pNext = NULL;
18.|    return p;
19.}
```

# Chương trình đầu tiên dslk đơn hai trỏ

```
11.void AddTail(LIST&ℓ, NODE*p)
12.{
13.|    if(ℓ.pHead==NULL)
14.|        ℓ.pHead=ℓ.pTail=p;
15.|    else
16.|    {
17.|    |    ℓ.pTail->pNext = p;
18.|    |    ℓ.pTail = p;
19.|    }
20.}
```

```
11.void Input(LIST &ℓ)
12.{
13.|    int n;
14.|    cout<<"Nhap n: ";
15.|    cin>>n;
16.|    Init(ℓ);
17.|    for(int i=1;i<=n;i++)
18.|    {
19.|    |    int x;
20.|    |    cout<<"Nhap x:";
21.|    |    cin>>x;
22.|    |    NODE*p = GetNode(x);
23.|    |    if(p!=NULL)
24.|    |        AddTail(ℓ,p);
25.|    }
26.}
```

# Chương trình đầu tiên dslk đơn hai trỏ

```
11.void Output(LIST ℓ)
12.{
13.|    NODE *p = ℓ.pHead;
14.|    while(p!=NULL)
15.|    {
16.|    |   cout<<p->info;
17.|    |   p = p->pNext;
18.|    }
19.}
```

```
11.int DemLe(LIST ℓ)
12.{
13.|    int dem=0;
14.|    NODE *p = ℓ.pHead;
15.|    while(p!=NULL)
16.|    {
17.|    |   if(p->info%2==0)
18.|    |       dem++;
19.|    |   p = p->pNext;
20.|    }
21.|    return dem;
22.}
```

# Chương trình đầu tiên dslk đơn hai trỏ

```
11.int Tong(LIST ℓ)
12.{
13.    int s=0;
14.    NODE *p = ℓ.pHead;
15.    while(p!=NULL)
16.    {
17.       s = s + p->info;
18.       p = p->pNext;
19.    }
20.    return s;
21.}
```

# Chương trình đầu tiên dslk đơn một trỏ

```
#include <iostream>
#include <iomanip>
using namespace std;
struct node
{
    int info;
    struct node *pNext;
};
typedef struct node NODE;
typedef NODE* LIST;
```

```
//Khai báo hàm
void Init(LIST&);
int IsEmpty(LIST);
NODE* GetNode(int);
void AddTail(LIST&,NODE*);

void Input(LIST &);
void Output(LIST);

int DemLe(LIST);
int Tong(LIST);
```

# Chương trình đầu tiên dslk đơn một trỏ

```
11.int main()
12.{
13.    LIST lst;
14.    Input(lst);
15.    cout<<"Danh sach lien ket don ban dau";
16.    Output(lst);
17.    int kq = DemLe(lst);
18.    cout<<"\nSo luong gia tri le trong danh sach: "<<kq;
19.    kq = Tong(lst);
20.    cout<<"\nTong cac gia tri trong danh sach: "<<kq;
21.    return 1;
22.}
```

# Chương trình đầu tiên dslk đơn một trỏ

```
11.void Init(LIST &ℓ)
12.{
13.     ℓ = NULL;
14.}
15.int IsEmpty(LIST ℓ)
16.{
17.     if (ℓ==NULL)
18.          return 1;
19.     return 0;
20.}
```

```
11.NODE* GetNode(int x)
12.{
13.     NODE *p=new NODE;
14.     if(p==NULL)
15.          return NULL;
16.     p->info = x;
17.     p->pNext = NULL;
18.     return p;
19.}
```

# Chương trình đầu tiên dslk đơn một trỏ

```
11.void AddTail(LIST&ℓ, NODE*p)
12.{
13.|    if(ℓ==NULL)
14.|        ℓ = p;
15.|    else
16.|    {
17.|    |    NODE *q=ℓ;
18.|    |    while(q->pNext)
19.|    |        q = q->pNext;
20.|    |    q->pNext = p;
21.|    }
22.}
```

```
11.void Input(LIST &ℓ)
12.{
13.|    int n;
14.|    cout<<"Nhap n: ";
15.|    cin>>n;
16.|    Init(ℓ);
17.|    for(int i=1;i<=n;i++)
18.|    {
19.|    |    int x;
20.|    |    cout<<"Nhap x:";
21.|    |    cin>>x;
22.|    |    NODE*p = GetNode(x);
23.|    |    if(p!=NULL)
24.|    |        AddTail(ℓ,p);
25.|    }
26.}
```

# Chương trình đầu tiên dslk đơn một trỏ

```
11.void Output(LIST ℓ)
12.{
13.|    NODE *p = ℓ;
14.|    while(p!=NULL)
15.|    {
16.|    |   cout<<p->info;
17.|    |   p = p->pNext;
18.|    }
19.}
```

```
11.int DemLe(LIST ℓ)
12.{
13.|    int dem=0;
14.|    NODE *p = ℓ;
15.|    while(p!=NULL)
16.|    {
17.|    |   if(p->info%2==0)
18.|    |       dem++;
19.|    |   p = p->pNext;
20.|    }
21.|    return dem;
22.}
```

# Chương trình đầu tiên dslk đơn một trỏ

```
11.int Tong(LIST ℓ)
12.{
13.    int s=0;
14.    NODE *p = ℓ;
15.    while(p!=NULL)
16.    {
17.       s = s + p->info;
18.       p = p->pNext;
19.    }
20.    return s;
21.}
```

# CHƯƠNG TRÌNH THỨ HAI DSLK ĐƠN

# Chương trình thứ hai dslk đơn

- Bài toán: Viết chương trình thực hiện các yêu cầu sau:
  - + Nhập dslk đơn các số thực.
  - + Xuất dslk đơn.
  - + Tìm địa chỉ node lớn nhất trong dslk đơn.
  - + Sắp xếp dslk tăng dần.

# Chương trình thứ hai dslk đơn hai trỏ

```
11.include <iostream>
12.include <iomanip>
13.using namespace std;
14.struct node
15.{
16.    float info;
17.    struct node *pNext;
18.};
19.typedef struct node NODE;
20.struct list
21.{
22.    NODE *pHead;
23.    NODE *pTail;
24.};
```

```
11.typedef struct list LIST;
12.//Khai báo hàm
13.void Init(LIST&);
14.int IsEmpty(LIST);
15.NODE* GetNode(float);
16.void AddTail(LIST&,NODE*);

17.void Input(LIST &);
18.void Output(LIST);

19.NODE* LonNhat(LIST);
20.void SapTang(LIST);
```

# Chương trình thứ hai dslk đơn hai trỏ

```
11.int main()
12.{
13.    LIST lst;
14.    Input(lst);
15.    cout<<"Danh sach lien ket don ban dau: \n";
16.    Output(lst);
17.    NODE *kq = LonNhat(lst);
18.    cout<<"\nDia chi node lon nhat: "<<kq<<endl;
19.    SapTang(lst);
20.    cout<<"\nDanh sach sau khi sap tang: "<<endl;
21.    Output(lst);
22.    return 1;
23.}
```

# Chương trình thứ hai dslk đơn hai trỏ

```
11.void Init(LIST &ℓ)
12.{
13.|    ℓ.pHead = NULL;
14.|    ℓ.pTail = NULL;
15.}
16.int IsEmpty(LIST ℓ)
17.{
18.|    if (ℓ.pHead==NULL)
19.|        return 1;
20.|    return 0;
21.}
```

```
11.NODE* GetNode(float x)
12.{
13.|    NODE *p=new NODE;
14.|    if(p==NULL)
15.|        return NULL;
16.|    p->info = x;
17.|    p->pNext = NULL;
18.|    return p;
19.}
```

# Chương trình thứ hai dslk đơn hai trỏ

```
11.void AddTail(LIST&ℓ, NODE*p)
12.{
13.|    if(ℓ.pHead==NULL)
14.|        ℓ.pHead=ℓ.pTail=p;
15.|    else
16.|    {
17.|    |    ℓ.pTail->pNext = p;
18.|    |    ℓ.pTail = p;
19.|    }
20.}
```

```
11.void Input(LIST &ℓ)
12.{
13.|    int n;
14.|    cout<<"Nhap n: ";
15.|    cin>>n;
16.|    Init(ℓ);
17.|    for(int i=1;i<=n;i++)
18.|    {
19.|    |    float x;
20.|    |    cout<<"Nhap x:";
21.|    |    cin>>x;
22.|    |    NODE*p = GetNode(x);
23.|    |    if(p!=NULL)
24.|    |        AddTail(ℓ,p);
25.|    }
26.}
```

# Chương trình thứ hai dslk đơn hai trỏ

```
11.NODE* LonNhat(LIST ℓ)
12.{
13.    NODE *lc = ℓ.pHead;
14.    NODE *p = ℓ.pHead;
15.    while(p!=NULL)
16.    {
17.        if(p->info>lc->info)
18.            lc = p;
19.        p = p->pNext;
20.    }
21.    return lc;
22.}
```

# Chương trình thứ hai dslk đơn hai trỏ

```
11.void Output(LIST l)
12.{
13.    NODE *p = l.pHead;
14.    while(p!=NULL)
15.    {
16.        cout<<" Address: "<<p;
17.        cout<<setw(6);
18.        cout<<setprecision(3);
19.        cout<<" Value:"<<p->info;
20.        cout<<endl;
21.        p = p->pNext;
22.    }
23.}
```

# Chương trình thứ hai dslk đơn hai trỏ

```
11.void SapTang(LIST ℓ)
12.{
13.    for(NODE *p = ℓ.pHead;p->pNext!=NULL;p=p->pNext)
14.        for(NODE *q = p->pNext;q!=NULL;q=q->pNext)
15.            if(p->info>q->info)
16.            {
17.                int temp = p->info;
18.                p->info = q->info;
19.                q->info = temp;
20.            }
21.}
```

# Chương trình thứ hai dslk đơn một trỏ

```
11.include <iostream>
12.include <iomanip>
13.using namespace std;
14.struct node
15.{
16.    float info;
17.    struct node *pNext;
18.};
19.typedef struct node NODE;
20.typedef NODE* LIST;
```

```
11.//Khai báo hàm
12.void Init(LIST&);
13.int IsEmpty(LIST);
14.NODE* GetNode(float);
15.void AddTail(LIST&,NODE*);

16.void Input(LIST &);
17.void Output(LIST);

18.NODE* LonNhat(LIST);
19.void SapTang(LIST);
```

# Chương trình thứ hai dslk đơn một trỏ

```
11.int main()
12.{
13.    LIST lst;
14.    Input(lst);
15.    cout<<"Danh sach lien ket don ban dau: \n";
16.    Output(lst);
17.    NODE *kq = LonNhat(lst);
18.    cout<<"\nDia chi node lon nhat: "<<kq<<endl;
19.    SapTang(lst);
20.    cout<<"\nDanh sach sau khi sap tang: "<<endl;
21.    Output(lst);
22.    return 1;
23.}
```

# Chương trình thứ hai dslk đơn một trỏ

```
11.void Init(LIST &ℓ)
12.{
13.     ℓ = NULL;
14.}
15.int IsEmpty(LIST ℓ)
16.{
17.     if (ℓ==NULL)
18.          return 1;
19.     return 0;
20.}
```

```
11.NODE* GetNode(float x)
12.{
13.     NODE *p=new NODE;
14.     if(p==NULL)
15.          return NULL;
16.     p->info = x;
17.     p->pNext = NULL;
18.     return p;
19.}
```

# Chương trình thứ hai dslk đơn một trỏ

```
11.void AddTail(LIST&ℓ, NODE*p)
12.{
13.|    if(ℓ==NULL)
14.|        ℓ=p;
15.|    else
16.|    {
17.|    |    NODE *q=ℓ;
18.|    |    while(q->pNext)
19.|    |        q = q->pNext;
20.|    |    q->pNext = p;
21.|    }
22.}
```

```
11.void Input(LIST &ℓ)
12.{
13.|    int n;
14.|    cout<<"Nhap n: ";
15.|    cin>>n;
16.|    Init(ℓ);
17.|    for(int i=1;i<=n;i++)
18.|    {
19.|    |    float x;
20.|    |    cout<<"Nhap x:";
21.|    |    cin>>x;
22.|    |    NODE*p = GetNode(x);
23.|    |    if(p!=NULL)
24.|    |        AddTail(ℓ,p);
25.|    }
26.}
```

# Chương trình thứ hai dslk đơn một trỏ

```
11.NODE* LonNhat(LIST ℓ)
12.{
13.    NODE *lc = ℓ;
14.    NODE *p = ℓ;
15.    while(p!=NULL)
16.    {
17.        if(p->info>lc->info)
18.            lc = p;
19.        p = p->pNext;
20.    }
21.    return lc;
22.}
```

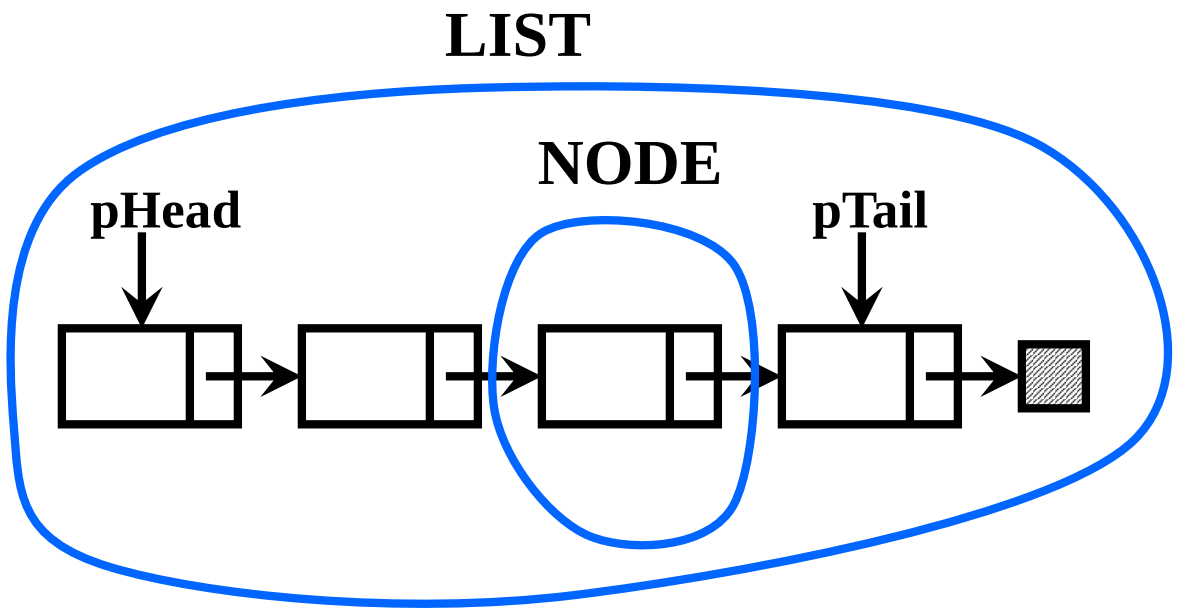
# Chương trình thứ hai dslk đơn một trỏ

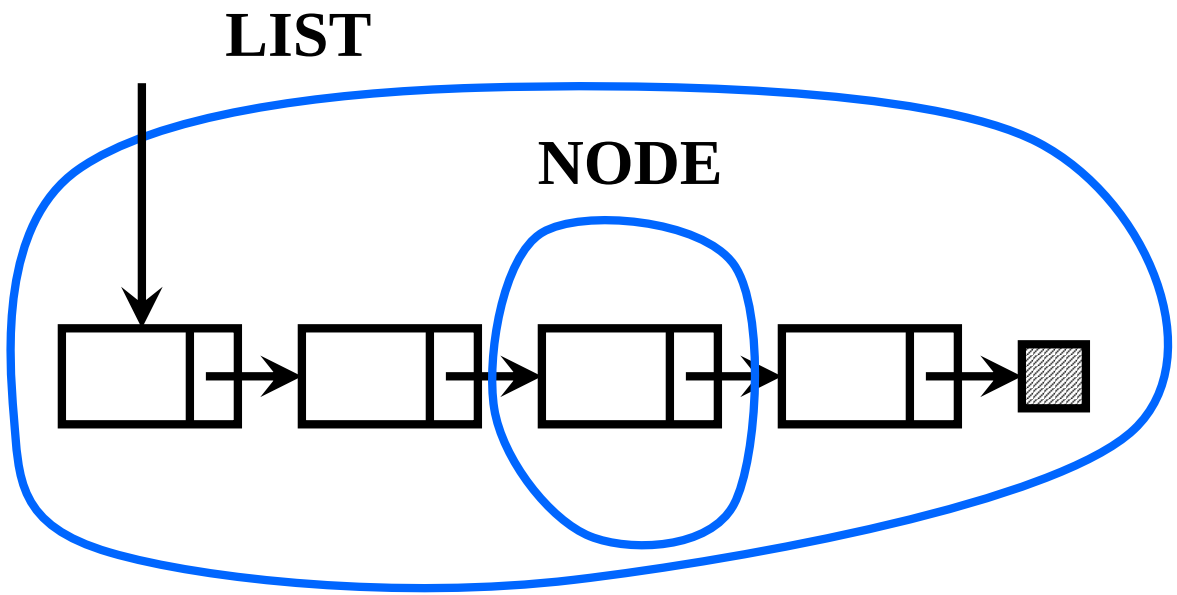
```
11.void Output(LIST ℓ)
12.{
13.    NODE *p = ℓ;
14.    while(p!=NULL)
15.    {
16.       cout<<" Address: "<<p;
17.       cout<<setw(6);
18.       cout<<setprecision(3);
19.       cout<<" Value:"<<p->info;
20.       cout<<endl;
21.       p = p->pNext;
22.    }
23.}
```

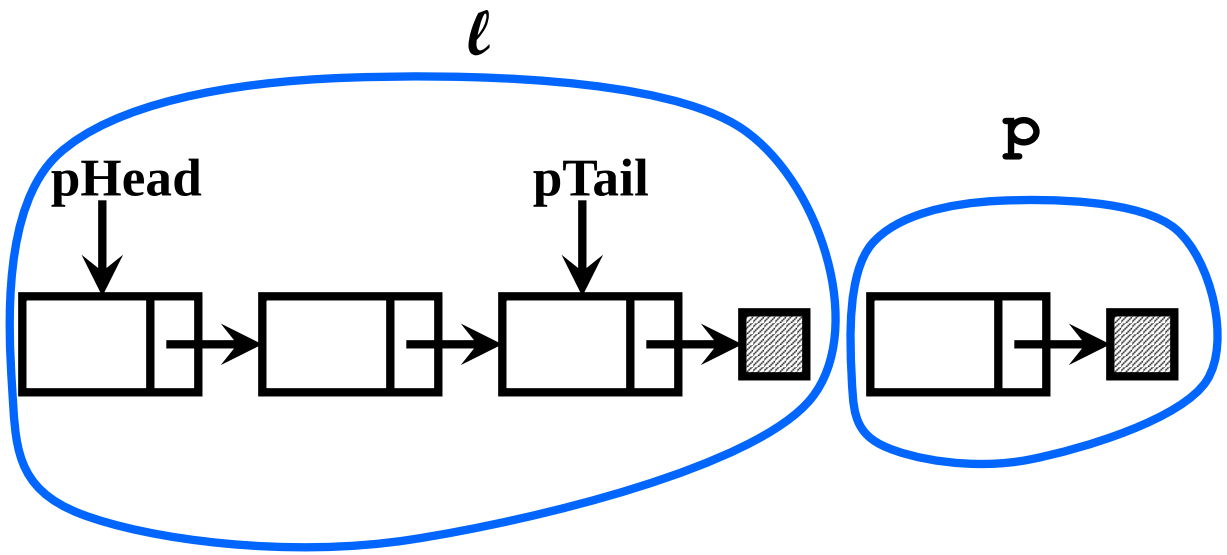
# Chương trình thứ hai dslk đơn một trỏ

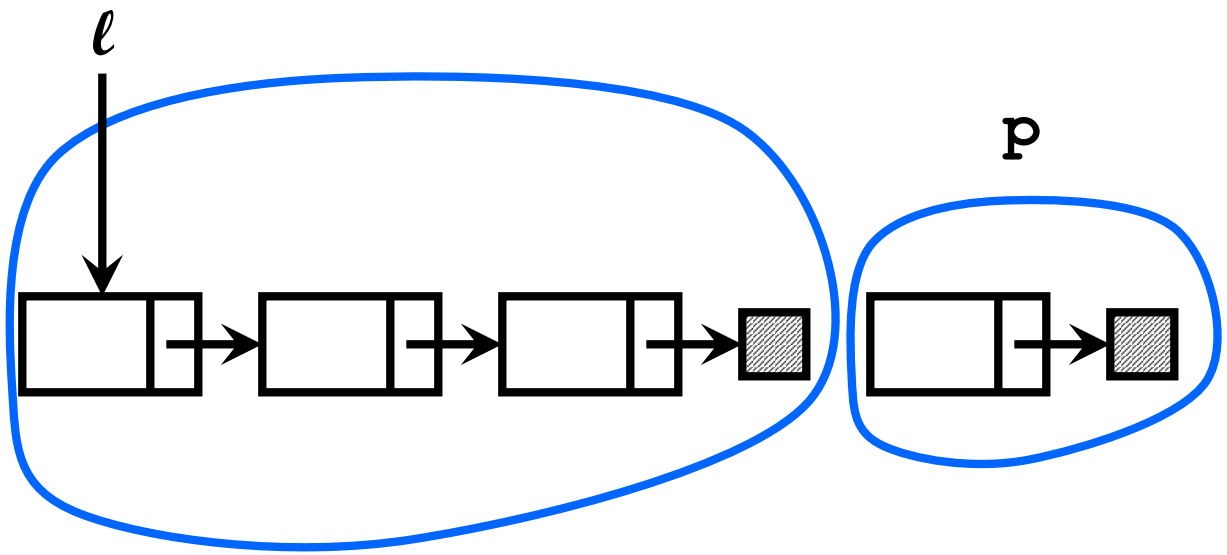
```
11.void SapTang(LIST ℓ)
12.{
13.    for(NODE *p = ℓ;p->pNext!=NULL;p=p->pNext)
14.        for(NODE *q = p->pNext;q!=NULL;q=q->pNext)
15.            if(p->info>q->info)
16.            {
17.                int temp = p->info;
18.                p->info = q->info;
19.                q->info = temp;
20.            }
21.}
```

# Chúc các bạn học tốt


LIST
NODE
pHead
pTail


LIST
NODE


ℓ
pHead
pTail
p


ℓ
p