

CHƯƠNG 4

KIỂU CẤU TRÚC VÀ HỢP

NGUYỄN QUỲNH DIỆP – KHOA CNTT - ĐH THỦY LỢI



1. Kiểu cấu trúc
2. Cấu trúc tự trở và danh sách liên kết
3. Con trỏ tới thành phần
4. Kiểu hợp
5. Kiểu liệt kê



A 3D rendered yellow figure, resembling a stylized person, is holding a large, rectangular, light-colored sign. The figure is positioned behind the sign, with its hands visible at the top and bottom edges. The sign has a thin yellow border and contains the text "KIỂU CẦU TRÚC (STRUCT)" in a bold, black, sans-serif font. The background is a plain, light gray, and the floor at the bottom is made of wooden planks.

KIỂU CẦU TRÚC (STRUCT)

- Khái niệm
- Cách khai báo
- Cách sử dụng



- Mảng là một tập hợp các phần tử cùng kiểu dữ liệu. Số phần tử được xác định trước do vậy mảng được xem như là một **cấu trúc dữ liệu tĩnh**.
- Việc xác định số phần tử trước khi sử dụng → có trường hợp thừa, thiếu bộ nhớ.
- Một số thao tác như thêm, xóa bỏ một phần tử trong mảng sẽ dẫn tới nhiều phí tổn khi phải di dời một số lượng lớn các phần tử.
- Khi cần lưu trữ thông tin về các đối tượng với các kiểu dữ liệu khác nhau, ta sử dụng kiểu cấu trúc.

KHÁI NIỆM KIỂU CẤU TRÚC

6

- Điểm khác biệt so với mảng:
 - Mảng là tập các giá trị có cùng kiểu
 - Cấu trúc là tập các giá trị có kiểu khác nhau
- Định nghĩa cấu trúc
 - Định nghĩa trước khi khai báo biến
 - Ở phạm vi toàn cục
 - Việc định nghĩa sẽ không cấp phát bộ nhớ

```
struct tencautruc {  
    kieudulieu1  thuoctinh1;  
    .....  
    kieudulieun  thuoctinhn;  
};    //Phải có dấu ; ở đây
```

- VD1: định nghĩa một cấu trúc có tên là **date**, trong đó có 3 biến thành phần kiểu **int** là **ngay**, **thang**, **nam**

```
struct date
{
    int ngay;
    int thang ;
    int nam;
};
```


- VD2: định nghĩa một cấu trúc có tên là **Sinhvien**, trong đó có 3 biến thành phần kiểu **char[]** là **tensv**, **masv**, **lop**, và 1 biến kiểu **date** là **ngaysinh** có giá trị khởi tạo là 2/3/2012.

```
struct Sinhvien
{
    char tensv[20];
    char masv[10];
    char lop[10];
    date ngaysinh = {2,3,2012};
};
```

KHỞI TẠO GIÁ TRỊ BAN ĐẦU CHO CẤU TRÚC

10

- Khởi tạo ngay trong định nghĩa cấu trúc

```
struct date
{
    int ngay = 1;
    int thang = 1;
    int nam = 2001;
};
```

KHỞI TẠO GIÁ TRỊ BAN ĐẦU CHO BIẾN

- Khởi tạo khi khai báo biến

```
struct date
{
    int ngay;
    int thang;
    int nam;
};
int main()
{
    date thu2 = {12, 3, 2018}; //khởi tạo giá trị
    date thu3 = {13, 3, 2018}; //khởi tạo giá trị
```

TRUY CẬP TỚI CÁC THÀNH PHẦN

12

- Các thành phần được gọi là các **thuộc tính** hoặc **trường**
 - Là thành phần của biến cấu trúc
 - Các biến cấu trúc khác nhau có thể có các trường trùng tên
- Sử dụng toán tử **.** để truy cập vào từng trường của cấu trúc.
- Đối với biến con trỏ kiểu cấu trúc thì dùng toán tử **→** để truy cập tới từng thành phần.
- Kiểu cấu trúc có thể lồng nhau, tức là thành viên có thể là một kiểu cấu trúc khác.
- Ví dụ:

```
tencautruc tenbien;  
tenbien.tenbienthanhphan_1;  
tenbien.tenbienthanhphan_2;
```



```
struct date
{
    int ngay;
    int thang ;
    int nam;
};
```

```
int main()
{
    date d[3];
    for(int i = 0; i<3; i++)
    {
        cout<< "Nhap ngay/thang/nam thu "<<i + 1<<": "<<endl;
        cout<<"Ngay: ";      cin>>d[i].ngay;
        cout<<"thang: ";      cin>>d[i].thang;
        cout<<"nam: ";        cin>>d[i].nam;
    }
    cout<<"Ban vua nhap vao 3 ngay la: "<<endl;
    for(int i = 0; i<3; i++)
    {
        cout<< "Ngay "<<i +1 <<": "
        <<d[i].ngay<<" - "<<d[i].thang<<" - "<<d[i].nam<<endl;
    }
}
```

Khai báo một mảng **d** gồm 3 phần tử kiểu **date**

Truy cập vào từng thành phần của kiểu **date** bằng dấu **.**

- Cấu trúc sau khi định nghĩa sẽ được dùng như một kiểu cơ sở.
 - ⇒ Có thể gán trực tiếp 2 biến cùng kiểu cấu trúc
 - ⇒ Có thể dùng làm đối số của hàm
 - ⇒ Có thể làm kiểu trả về của một hàm
 - ⇒ Có thể dùng để khai báo biến trong 1 cấu trúc khác



- Cho 2 biến p và q có kiểu cấu trúc A
- Phép gán $p=q$ có tác dụng sao chép giá trị của tất cả các biến thành phần trong q vào các biến thành phần tương ứng trong p

- Có thể được truyền giống như các kiểu dữ liệu đơn giản
 - Truyền tham trị
 - Truyền tham chiếu
 - Truyền kết hợp
- Có thể được trả về bởi hàm
 - Kiểu trả về là kiểu cấu trúc
 - Lệnh trả về trong định nghĩa hàm (return) gửi biến cấu trúc trở lại lời gọi hàm.

- **Khái niệm:** Con trỏ cấu trúc là con trỏ chứa địa chỉ của một biến cấu trúc hoặc một vùng nhớ có kiểu cấu trúc nào đó.
- **Cách khởi tạo một con trỏ cấu trúc:**
 - Con trỏ được khởi tạo bằng việc sử dụng toán tử **new** để cấp phát bộ nhớ
 - Đối với con trỏ **p** trỏ đến mảng **a**, có thể truy cập đến các thành phần của phần tử của mảng



- Khai báo cấu trúc:

```
struct Tencautruc {  
    Kieudulieu1  thuoctinh1;  
    .....  
    Kieudulieun  thuoctinhn;  
};
```

- Khai báo con trỏ cấu trúc:

```
tencautruc *contro= new tencautruc [kichthuoc];
```

- Truy cập các thuộc tính: 3 cách


```
contro[chiso].thuoctinh  
(*(contro+chiso)).thuoctinh  
(contro+chiso)-> thuoctinh
```



```
1  #include<iostream>
2  #include<string>
3
4  using namespace std;
5
6  struct Sinhvien{
7      string ht;
8      float diem;
9  };
10
11  int main(){
12      int n;
13
14      Sinhvien *sv=new Sinhvien[100];
15      cout<<"N=";cin>>n;
16      for(int i=0;i<n;i++)
17      {
18          cin.ignore();
19          cout<<"Ho ten:";getline(cin,sv[i].ht);
20          cout<<"Diem:";cin>>(*(sv+i)).diem;
21      }
22      return 0;
23  }
```

Bài 1: Dùng mảng động và cấu trúc để thực hiện yêu cầu sau:

1. Tạo một cấu trúc **Sach** có các thành phần là: Tên sách, Nhà xuất bản, Giá.
2. Viết chương trình khai báo 1 danh sách gồm n quyển sách.
3. Nhập và hiển thị danh sách n quyển sách vừa nhập
4. Đếm xem trong danh sách có bao nhiêu cuốn sách của Nhà xuất bản GIÁO DỤC.
5. Thông báo ra màn hình Tên, Nhà xuất bản của cuốn sách có giá tiền cao nhất

A 3D rendered yellow figure, resembling a stylized person, is holding a large, rectangular, light-yellow sign. The figure is positioned behind the sign, with its hands visible at the top and sides. The sign has a thin black border and contains text in Vietnamese. The background is a plain, light yellow gradient. At the bottom of the image, there is a horizontal band showing a wooden floor with vertical planks.

CẤU TRÚC TỰ TRỞ VÀ DANH SÁCH LIÊN KẾT

- **Khái niệm:** là kiểu cấu trúc mà trong các thành phần của nó có 1 trường con trỏ, trỏ đến chính kiểu cấu trúc đó.
- **Ví dụ:**

```
6 □ struct Sinhvien{  
7     string ht;  
8     float diem;  
9     Sinhvien *tiep;  
10  };
```


- Cách 1

```
typedef struct Tên cấu trúc tên biến cấu trúc;  
struct Tên cấu trúc {  
    Khai báo các thuộc tính;  
    tên biến cấu trúc *control;  
};
```

- Cách 2

```
struct Tên cấu trúc {  
    Khai báo các thuộc tính;  
    Tên cấu trúc *control;  
};  
typedef Tên cấu trúc tên biến cấu trúc;
```

Cách 3

```
typedef struct tên biến cấu trúc {  
    Khai báo các thuộc tính;  
    tên biến cấu trúc *control;  
};
```

Cách 4

```
struct tên biến cấu trúc {  
    Khai báo các thuộc tính;  
    tên biến cấu trúc *control;  
};
```


CẤU TRÚC TỰ TRỞ

24

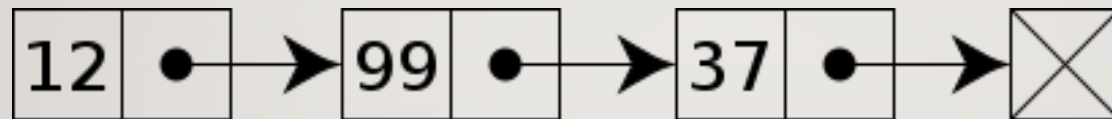
```
13 typedef struct Sinhvien sv
14 struct Sinhvien{
15     string ht;
16     float diem;
17     sv *tiep;
18 };
```

```
28 typedef struct sv
29 {
30     string ht;
31     float diem;
32     sv *tiep;
33 };
```

```
21 struct Sinhvien{
22     string ht;
23     float diem;
24     Sinhvien *tiep;
25 };
26 typedef struct Sinhvien sv
```

```
6 struct Sinhvien{
7     string ht;
8     float diem;
9     Sinhvien *tiep;
10 };
```

1. Danh sách liên kết đơn (Single linked list): Chỉ có sự kết nối từ phần tử phía trước tới phần tử ngay phía sau.



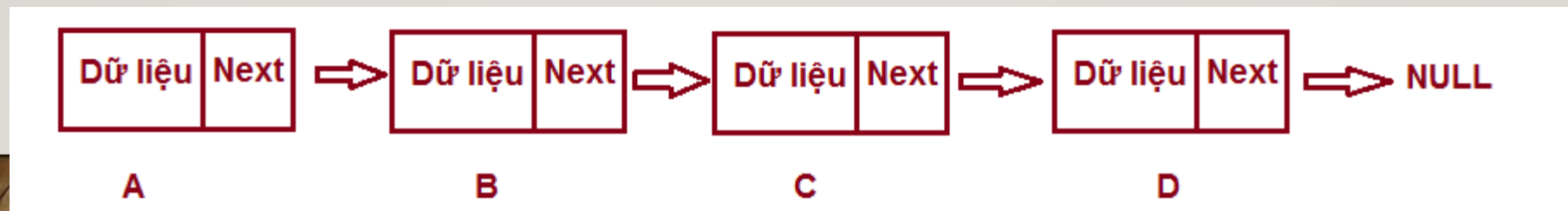
2. Danh sách liên kết đôi (Double linked list): Có sự kết nối 2 chiều giữa phần tử ngay trước với phần tử ngay sau



3. Danh sách liên kết vòng(Circular Linked List): Có thêm sự kết nối giữa 2 phần tử đầu tiên và phần tử cuối cùng để tạo thành vòng khép kín.



- **Danh sách liên kết đơn** (Single Linked List) là một cấu trúc dữ liệu động, nó là một danh sách mà mỗi phần tử đều liên kết với phần tử đứng ngay sau nó trong danh sách.
- Mỗi phần tử trong danh sách liên kết đơn (gọi là một node hay nút) là một cấu trúc gồm 2 thành phần:
 - Thành phần dữ liệu: lưu thông tin về bản thân phần tử đó.
 - Thành phần liên kết: lưu địa chỉ phần tử đứng ngay sau trong danh sách, nếu phần tử đó là phần tử cuối cùng thì thành phần này bằng NULL.



Nội dung	Mảng	Danh sách liên kết
Kích thước	Kích thước cố định Cần chỉ rõ kích thước khi khai báo	Kích thước thay đổi trong quá trình thêm/xóa phần tử
Cấp phát bộ nhớ	Tĩnh. Bộ nhớ được cấp phát trong quá trình biên dịch	Động. Bộ nhớ được cấp phát trong quá trình chạy CT
Thứ tự và sắp xếp	Lưu trữ trên 1 dãy ô nhớ liên tiếp	Lưu trữ trên các ô nhớ ngẫu nhiên
Truy cập	Truy cập được tới phần tử ngẫu nhiên thông qua chỉ số	Truy cập tới phần tử phải duyệt từ đầu đến cuối
Tìm kiếm	Tìm kiếm tuyến tính hoặc tìm kiếm nhị phân	Tìm kiếm tuyến tính

- Do tính liên kết của phần tử đầu và phần tử đứng ngay sau nó trong DSLK đơn nên nó có các đặc điểm sau:
 - Chỉ cần nắm được phần tử đầu là có thể quản lý được danh sách
 - Muốn truy cập tới phần tử ngẫu nhiên phải duyệt từ đầu đến vị trí đó
 - Chỉ có thể tìm kiếm tuyến tính một phần tử



- Tạo nút
- Tạo DSLK
- Thêm phần tử vào DSLK
- Xóa phần tử trong DSLK
- Tìm kiếm phần tử trong DSLK
- Duyệt DSLK



- Định nghĩa nút

```
struct Nut
{
    Kieudulieu  Dulieu;
    Nut *tiep ;
};
```

- Khai báo trên sẽ được sử dụng cho mọi nút trong danh sách liên kết.
- Trường **Dulieu** sẽ chứa giá trị
- Trường **tiep** sẽ là con trỏ để trỏ đến nút kế tiếp

Tạo nút mới:

- Thực hiện cấp phát động nút mới.
- Khởi tạo giá trị ban đầu cho nút
- Nút vừa tạo chưa có liên kết với phần tử nào cả. Do đó, phần liên kết của nó bằng NULL
- Trả về địa chỉ con trỏ, trỏ đến nút mới.

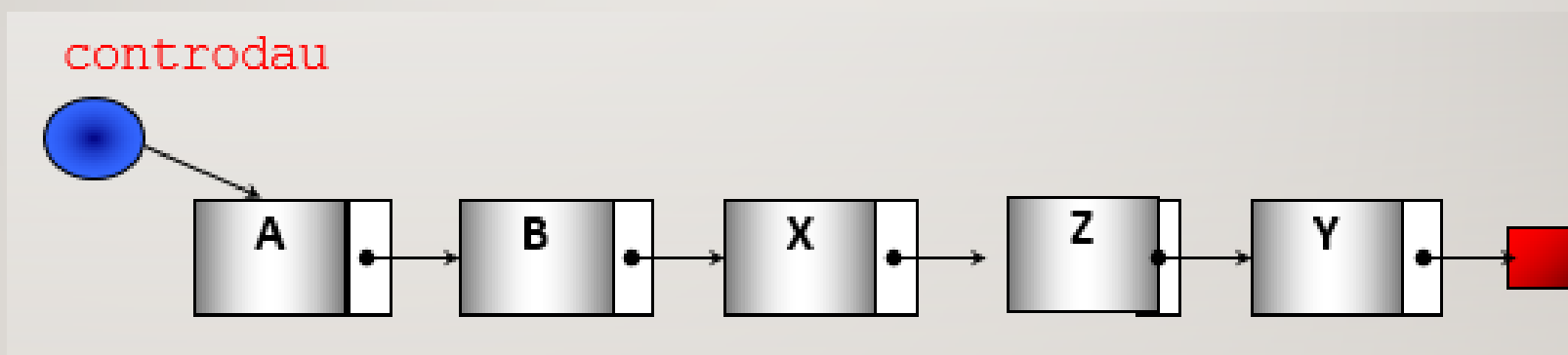
```
struct Nut
{
    Kieudulieu  Dulieu;
    Nut *tiep;
};
```

```
Nut *Taonut(Kieudulieu  Giatri)
{
    Nut *nutmoi = new Nut;
    nutmoi -> dulieu = Giatri;
    nutmoi-> tiep = NULL;
    return nutmoi;
}
```

TẠO DANH SÁCH LIÊN KẾT ĐƠN

34

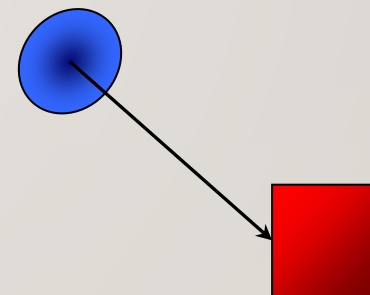
- Vì thành phần tạo nên DSLK đơn là **nút** nên cần quản lý chúng bằng cách biết được phần tử đầu DSLK.
- Vì mỗi phần tử đều liên kết với phần tử kế tiếp, vì vậy cần dùng 1 con trỏ lưu trữ địa chỉ phần tử đầu (head)



- Khi mới tạo danh sách, danh sách sẽ không có phần tử nào. Do đó, phần tử đầu không trở vào đâu cả nên chúng được gán bằng NULL.
- **Hàm tạo DS rỗng (chưa có nút nào)**

```
void Taods(Nut * &contro dau) {  
    contro dau = NULL;  
}
```

contro dau



- **Thêm một nút vào danh sách:** Có 3 vị trí thêm
 - Thêm vào đầu danh sách
 - Thêm vào cuối danh sách
 - Thêm vào sau nút q trong danh sách
- Lưu ý trường hợp danh sách ban đầu rỗng

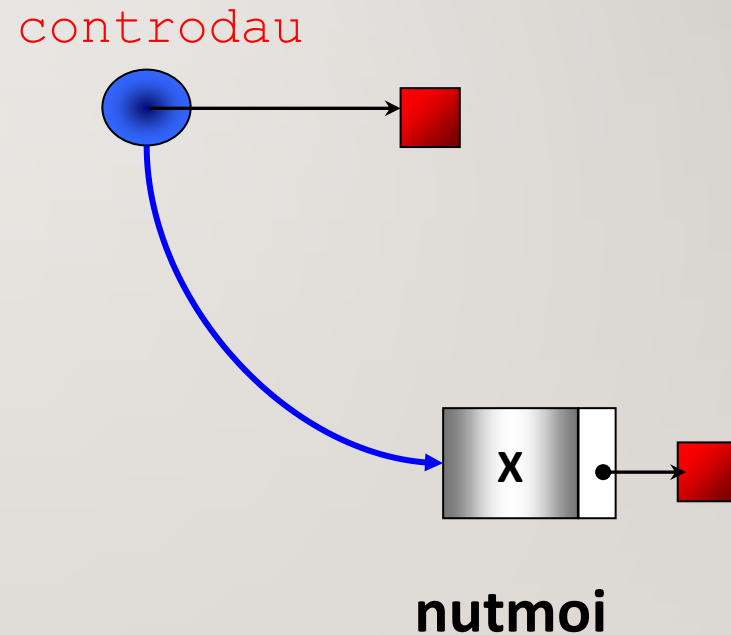


- Thêm vào đầu DSLK đơn

a. Nếu DS ban đầu rỗng:

- Tạo 1 nút mới
- Con trỏ đầu trỏ vào phần tử này

`Controdau = nutmoi;`



- Thêm vào đầu DSLK đơn

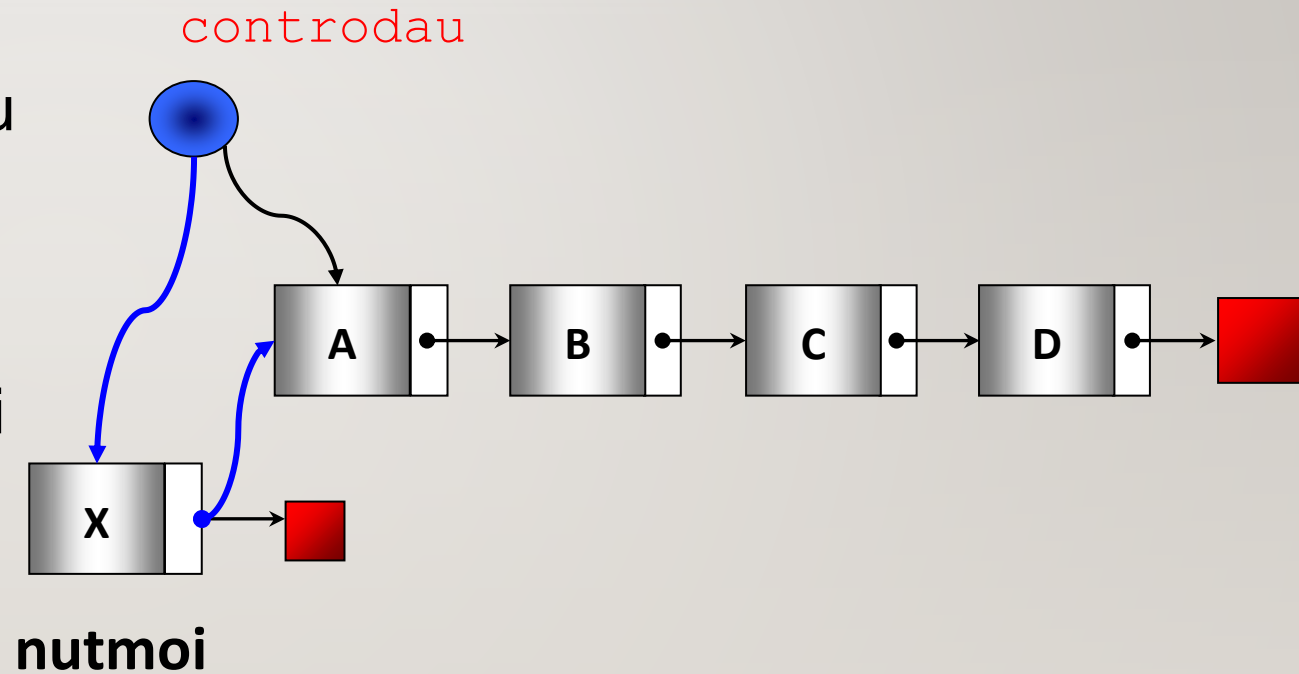
b. Nếu DS ban đầu khác rỗng:

- Cho liên kết của nút mới trở tới nút đầu DS cũ

nutmoi -> **tiếp** = **controldau**;

- Cập nhật lại con trỏ đầu trở tới nút mới

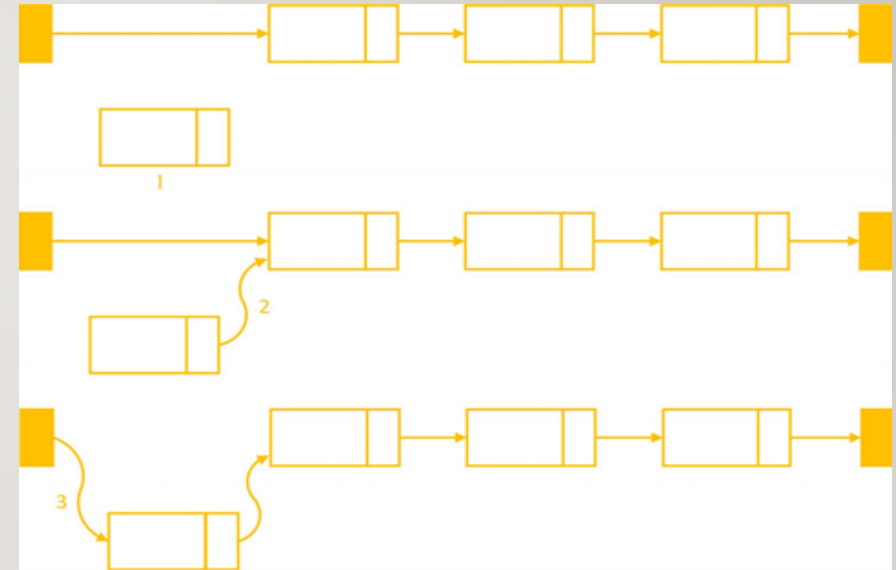
Controldau = **nutmoi** ;



THÊM NÚT VÀO DANH SÁCH

39

```
void Themvaodau(Nut *&controldau, Nut *p)
{
    if (controldau == NULL)
    {
        controldau = p;
    }
    else
    {
        p ->tiếp = controldau;
        controldau = p;
    }
}
```

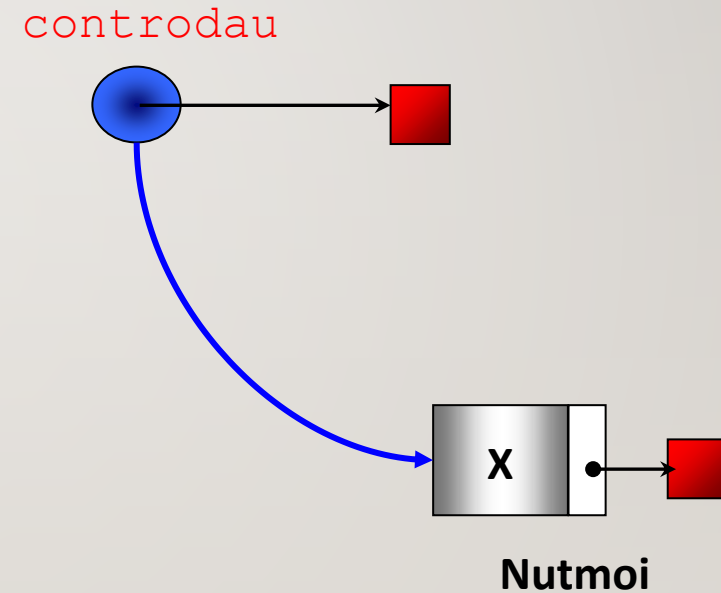


- Thêm vào cuối DSLK đơn

a. Nếu DS ban đầu rỗng:

- Tạo 1 nút mới
- Con trỏ đầu trỏ vào phần tử này

`Controdau = nutmoi;`



THÊM NÚT VÀO DANH SÁCH

- Thêm vào cuối DSLK đơn

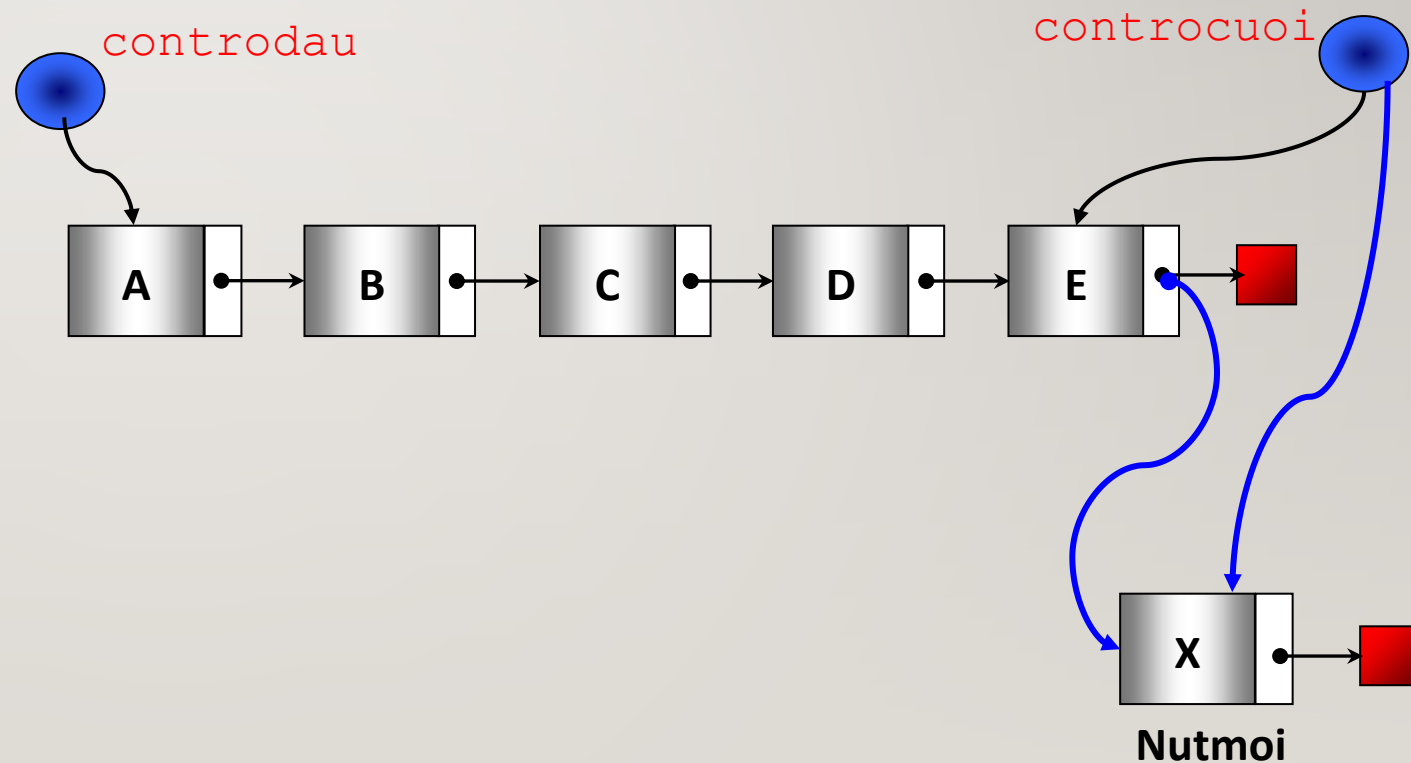
b. Nếu DS ban đầu khác rỗng:

- Dùng **controcuoi** trở đến nút cuối của DSLK đơn.
- Cho liên kết của con trỏ cuối trở tới nút mới

controcuoi -> **tiep** = **nutmoi**;

- Cập nhật lại con trỏ cuối

controcuoi = **nutmoi** ;



THÊM PHẦN TỬ VÀO DANH SÁCH

42

//chạy **controcuoi** đến nút cuối của DS

Nut *controcuoi;

controcuoi=contro dau;

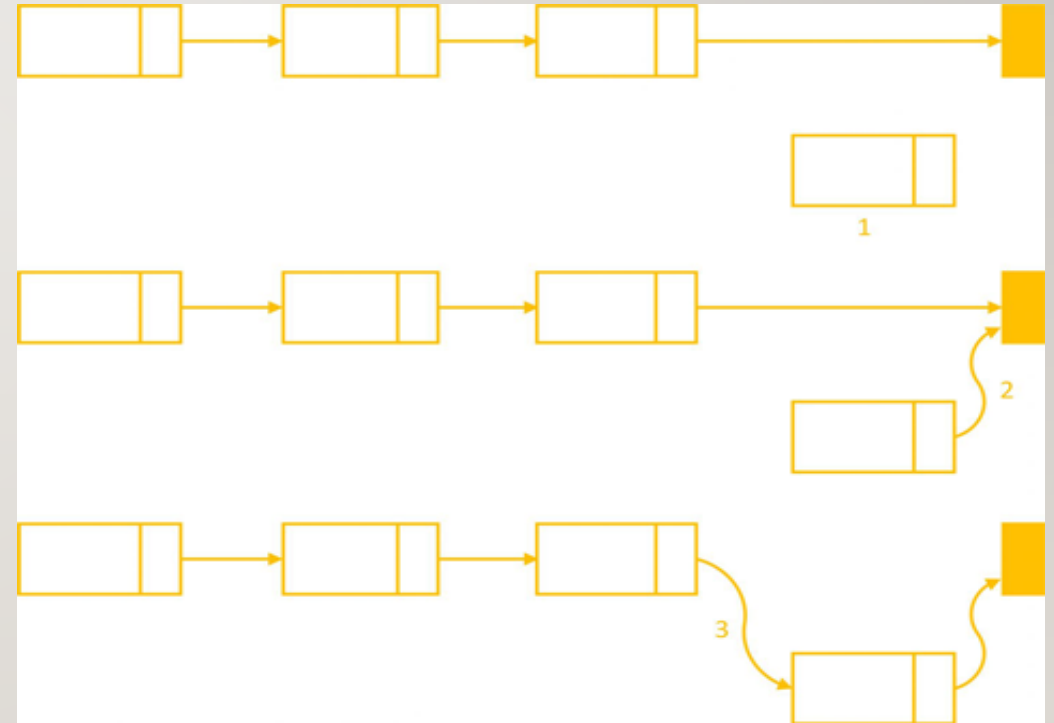
while (controcuoi-> tiep!=NULL)

controcuoi=controcuoi->tiep;

THÊM PHẦN TỬ VÀO DANH SÁCH

43

```
void Themvaocuoi (Nut *contro dau, Nut *p)
{
    Nut *contro cuoi; //la con tro tro den nut cuoi của DS
    if (contro dau == NULL) //DS rỗng
    {
        contro dau = p;
    }
    else
    {
        contro cuoi->tiếp = p;
        contro cuoi=p;
    }
}
```

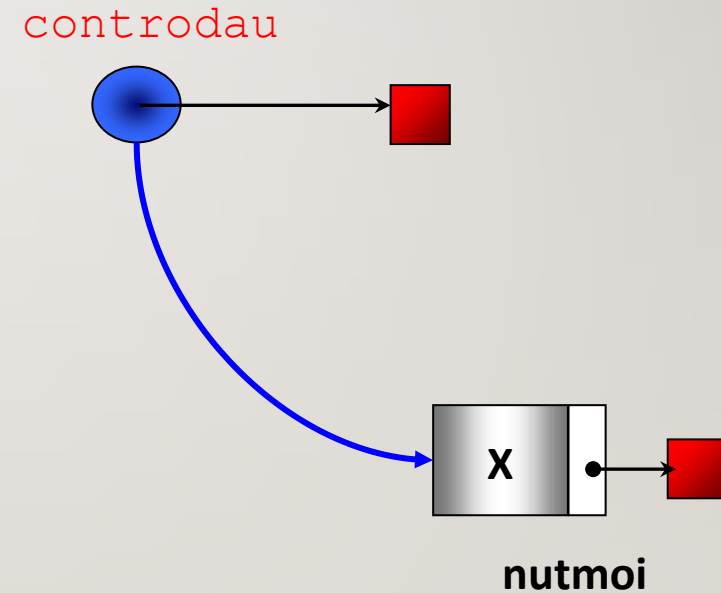


- Thêm vào sau nút q trong DSLK đơn

a. Nếu DS ban đầu rỗng:

- Tạo 1 nút mới
- Con trỏ đầu trỏ vào phần tử này

`controldau = nutmoi;`



THÊM NÚT VÀO DANH SÁCH

- Thêm vào sau nút q trong DSLK đơn

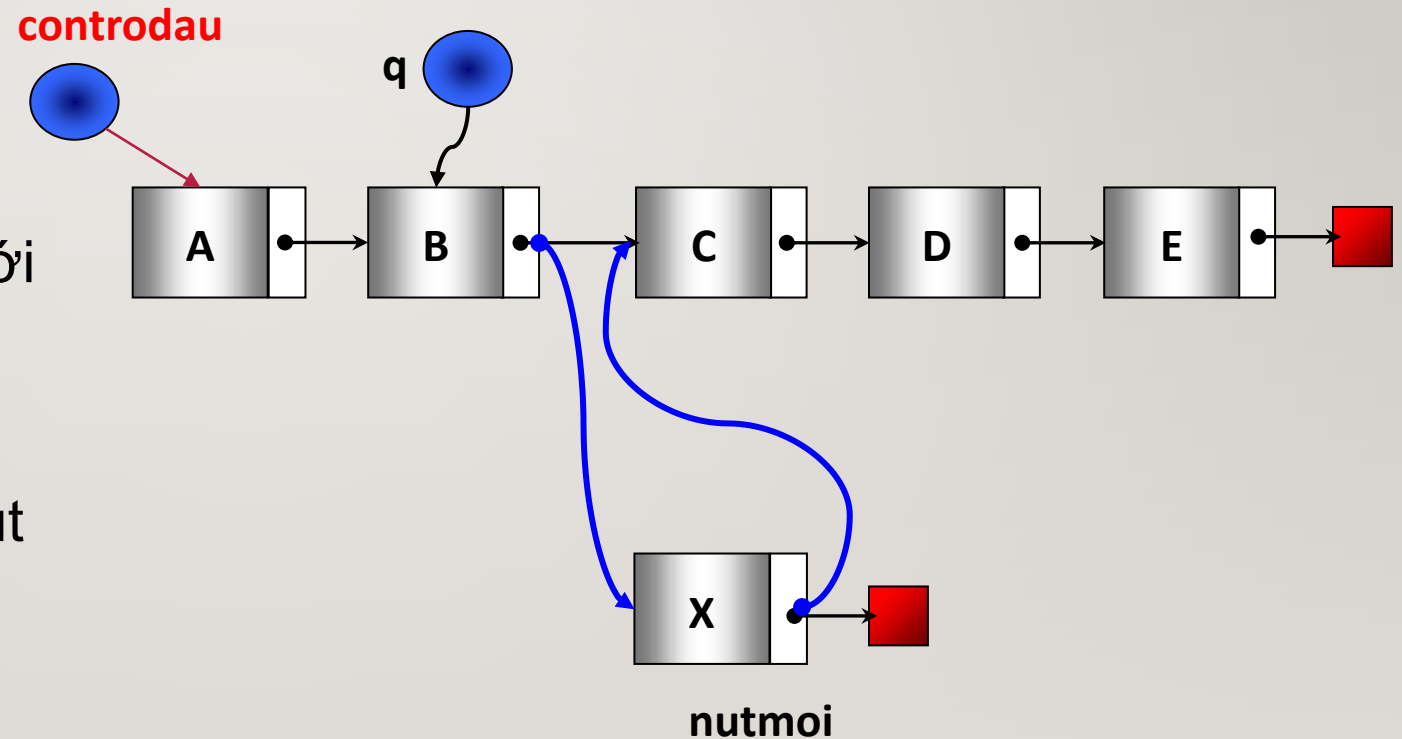
b. Nếu DS ban đầu khác rỗng:

- Tạo 1 nút mới
- Chèn nút mới vào sau q
 - Cho liên kết ở nút mới trở tới liên kết của q

$\text{Nutmoi} \rightarrow \text{tiếp} = q \rightarrow \text{tiếp}$

- Cho liên kết của q trở tới nút mới

$q \rightarrow \text{tiếp} = \text{nutmoi}$



- Thêm vào sau nút q trong DSLK đơn 1 nút p

```
void Themvaosau(Nut *contro dau, Nut *q, Nut * p)
{
    if (q!=NULL)
    {
        p->tiiep = q->tiiep;
        q->tiiep = p;
    }
}
```

Bài 2:

- Viết chương trình tạo 1 danh sách liên kết đơn dùng để lưu trữ n số nguyên x , với n và x nhập từ bàn phím.
- Danh sách liên kết trên được tạo nên từ các nút bằng cách thêm vào **đầu** danh sách.
- Viết hàm tính và thông báo ra màn hình **tổng** các số trong danh sách trên.
- *Chú ý: Trong chương trình sử dụng hàm Tạo nút, tạo danh sách rỗng, thêm 1 nút vào đầu danh sách.*

Bài 3:

- Viết chương trình tạo 1 danh sách liên kết đơn dùng để lưu trữ n số nguyên x , với n và x nhập từ bàn phím.
- Danh sách liên kết trên được tạo nên từ các nút bằng cách thêm vào **cuối** danh sách.
- Đếm số lượng các số **chẵn** trong danh sách trên.
- *Chú ý: Trong chương trình sử dụng hàm Tạo nút, Tạo danh sách rỗng, Thêm 1 nút vào cuối danh sách.*

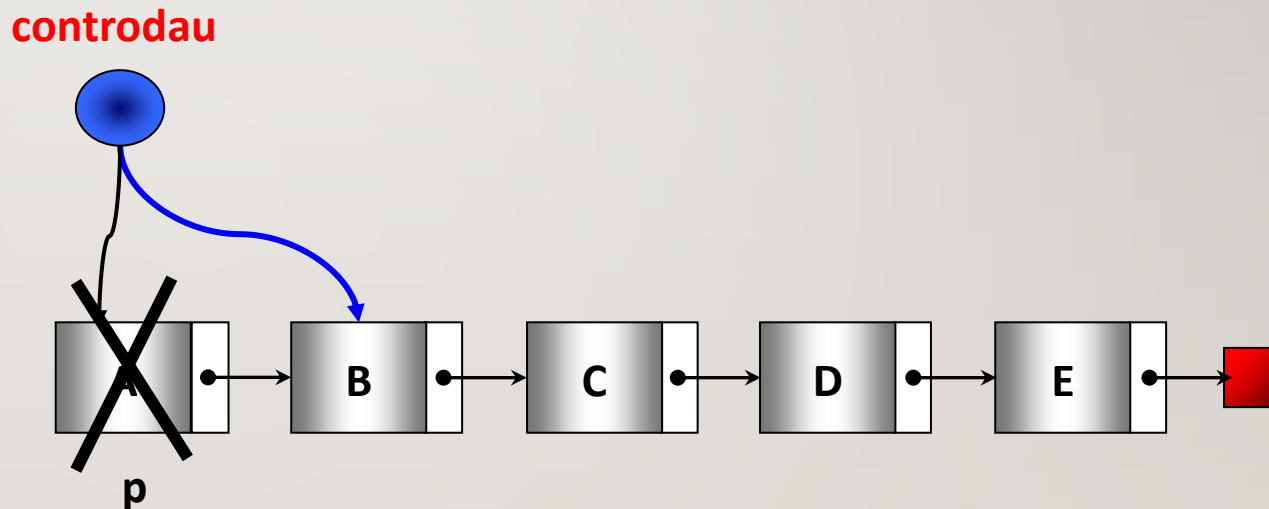
- Xóa một nút của danh sách liên kết:
 - Xóa nút đầu của danh sách
 - Xóa nút sau nút q trong danh sách
 - Xóa nút có giá trị k (hoặc xóa 1 nút trở bởi con trở p)



XÓA KHỎI DSLK 1 NÚT

- **Xóa nút đầu của DSLK**

- Gán $p = \text{contro dau}$
- Cho contro dau trở vào nút sau nút p: $\text{contro dau} = p \rightarrow \text{tiếp}$
(hoặc $\text{contro dau} = \text{contro dau} \rightarrow \text{tiếp}$)
- Giải phóng vùng nhớ mà p trở tới: `delete p`



XÓA KHỎI DSLK 1 NÚT

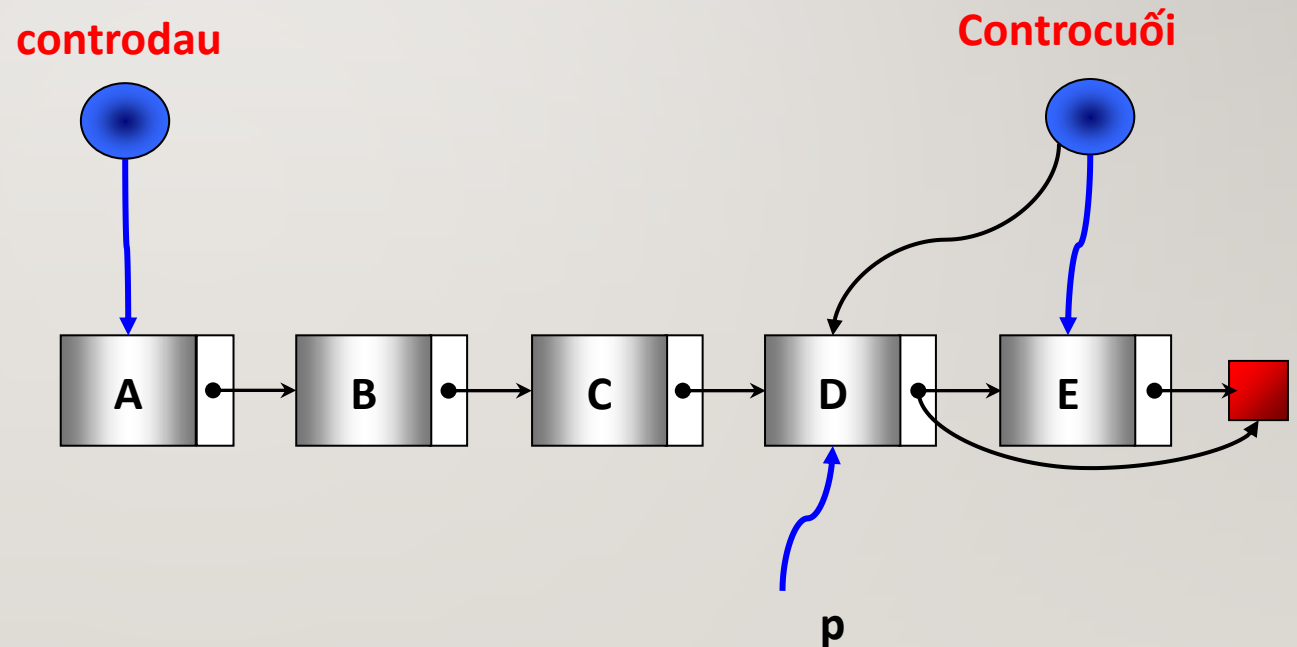
51

```
void Xoadauds(Nut *contro dau)
{ if contro dau = NULL
    cout<<"Danh sach rong";
    else
    {Nut* p= contro dau;
      contro dau = p->tiep;      //hoặc contro dau = contro dau->tiep;
      delete p;
    }
}
```

XÓA KHỎI DSLK 1 NÚT

- **Xóa nút cuối của DSLK**

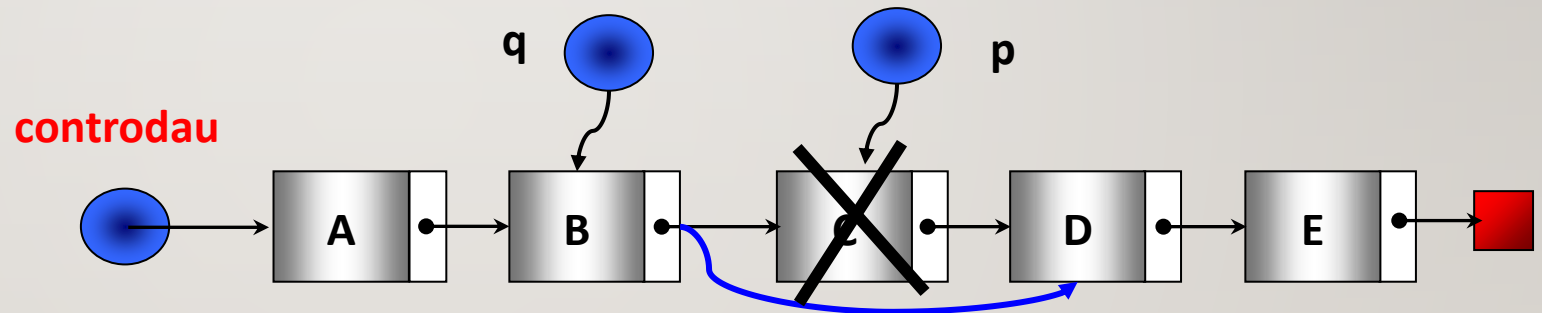
- Giả sử **controcuoi** trở vào nút cuối của DSLK
- Gán **p = contro dau** và chạy p đến trước nút cuối cùng.
- Làm cho p trở thành nút cuối của DS: **p->tiếp=NULL**
- Giải phóng vùng nhớ mà controcuoi trở tới:
delete controcuoi
- Cập nhật lại cotrocuoi: **controcuoi=p**



- **Xóa nút sau nút q trong DSLK đơn**
 - Điều kiện để có thể xóa được nút sau q là:
 - q phải khác NULL ($q \neq \text{NULL}$)
 - nút sau q phải khác NULL ($q \rightarrow \text{tiep} \neq \text{NULL}$)
 - Có các thao tác:
 - Gọi p là nút sau q
 - Cho vùng tiep của q trở vào nút đứng sau p
 - Nếu p là phần tử cuối thì q sẽ là phần tử cuối.
 - Giải phóng vùng nhớ mà p trở tới

XÓA KHỎI DSLK 1 NÚT

- $q \rightarrow \text{tiếp} = p \rightarrow \text{tiếp}$
- Giải phóng vùng nhớ mà p đang trỏ tới: **delete p**;



XÓA KHỎI DSLK 1 NÚT

55

```
void Xoasauq(Nut *contro dau, Nut *q )
{
    if (q !=NULL && q->tiep !=NULL)
    {
        Nut *p = q->tiep;
        q->tiep = p->tiep;
        delete p;
    }
    else cout <<“ Khong co nut sau q”;
}
```

- **Xóa nút trở bởi con trở p trong DSLK đơn**
 - Tìm được nút q đứng trước p
 - Đưa về bài toán xóa nút đứng sau q

- Là thao tác thường được thực hiện khi có nhu cầu xử lý các phần tử của DSLK hoặc khi cần lấy thông tin từ các phần tử của DSLK như:
 - Đếm các phần tử của danh sách
 - Tìm tất cả các phần tử thoả điều kiện nào đó.
 - ...



- Bước 1: $p = \text{contro dau};$ *//Cho p trở đến phần tử đầu danh sách*
- Bước 2: Trong khi (Danh sách chưa hết) thực hiện:
 - B2.1 : Xử lý phần tử được trỏ bởi p
 - B2.2 : $p = p \rightarrow \text{tiep};$ *// Cho p trở tới phần tử kế*

DUYỆT DANH SÁCH LIÊN KẾT ĐƠN

```
void duyetDSLK(Nut *contro dau)
{
    Nut *p = contro dau;
    while (p!= NULL)
    {
        // xử lý cụ thể p tùy ứng dụng
        p = p->tiếp;
    }
}
```

DUYỆT DANH SÁCH LIÊN KẾT ĐƠN

Ví dụ: In các phần tử trong danh sách

```
void Hienthi(Nut *contro dau)
```

```
{
```

```
    Nut * p=contro dau;
```

```
    while (p!=NULL)
```

```
    {
```

```
        cout << p->data << "\t";
```

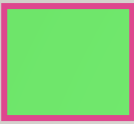
```
        p=p ->tiiep;
```

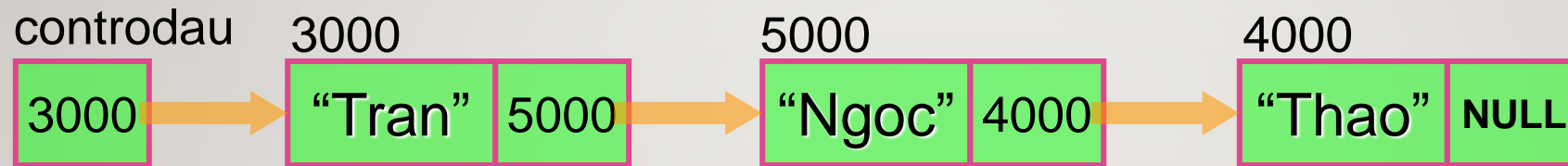
```
    }
```

```
    cout<<endl;
```

```
}
```

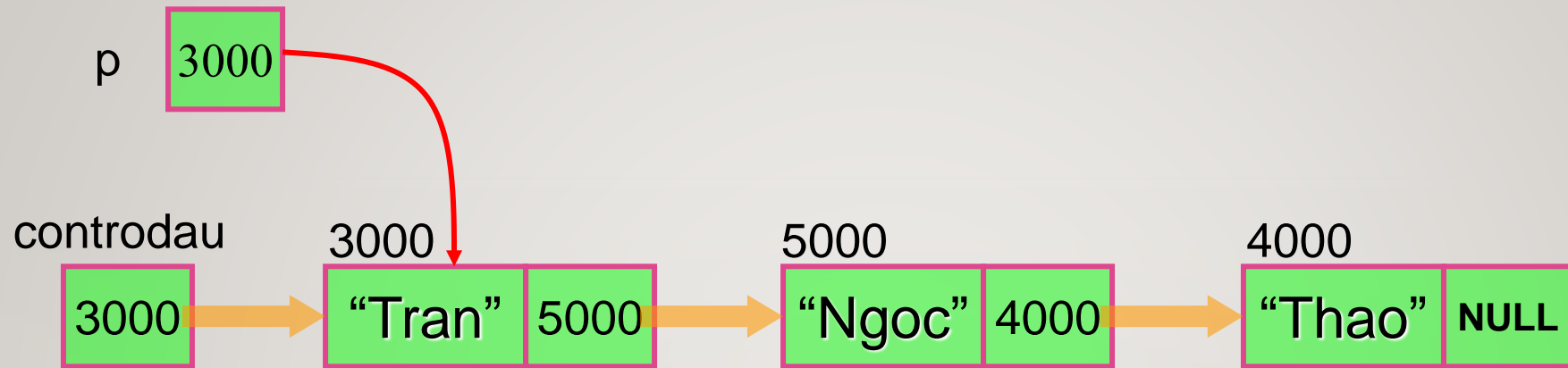

DUYỆT DANH SÁCH LIÊN KẾT ĐƠN

p 



```
p = contro dau;  
while (p!=NULL)  
{  
    cout<<p->data<<"\t";  
    p = p->link;  
}
```

DUYỆT DANH SÁCH LIÊN KẾT ĐƠN



```
p = controldau;  
while (p != NULL)  
{  
    cout << p->data << "\t";  
    p = p->link;  
}
```

TÌM KIẾM PHẦN TỬ TRONG DSLK ĐƠN

- Tìm kiếm một phần tử có giá trị = x

63

```
Nut *Tim (Nut *contro dau, int x)
```

```
{    if (contro dau == NULL) return NULL;
```

```
    else
```

```
    {    Nut* p = contro dau;
```

```
        while (p!=NULL)
```

```
        {    if (p->data == x)
```

```
            return p;
```

```
            else p=p->tiep;
```

```
        }
```


```
    }
```

```
}
```

Bài 4: Cho cấu trúc

```
struct Sinhvien{  
    string    ten;  
    int       diem;  
    Sinhvien  *tro;  
};
```

1. Viết chương trình tạo 1 danh sách liên kết đơn dùng để lưu trữ thông tin về **n** sinh viên bằng cách thêm vào **cuối** danh sách.
2. In ra danh sách **n** sinh viên. Thông tin của mỗi sinh viên gồm tên và điểm trên cùng 1 dòng.
3. Đưa ra số lượng sinh viên có điểm > 8
4. Nhập 1 tên cần tìm từ bàn phím. Tìm xem trong danh sách có sinh viên như vậy không? Nếu có thì xóa sinh viên đó khỏi danh sách.



KIỂU HỢP

Cách 1

1. Khai báo kiểu hợp

```
union Tenkieuhop{  
    Kieudulieu1  thuoctinh1;  
    .....  
    Kieudulieun  thuoctinhn;  
};
```

2. Khai báo biến hợp

```
Tenkieuhop tenbien;
```

Cách 2

1. Khai báo kiểu hợp và tạo biến

```
union Tenkieuhop{  
    Kieudulieu1  thuoctinh1;  
    .....  
    Kieudulieun  thuoctinhn;  
} tenbien_1, ...,tenbien_n;
```

TRUY CẬP CÁC THUỘC TÍNH

tenbien.thuoctinh

- Cách 1

```
5 union Sinhvien{  
6     char masv[10];  
7     char ht[50];  
8     char lop[20];  
9     float diem;  
10 };
```

```
17 Sinhvien sv;
```

Cách 2

```
5 union Sinhvien{  
6     char masv[10];  
7     char ht[50];  
8     char lop[20];  
9     float diem;  
10 }sv1, sv2;
```

- **Ưu điểm**

- Chiếm ít không gian bộ nhớ.
- Chỉ thuộc tính có kích thước **lớn nhất** mới có thể được truy cập trực tiếp trong khi sử dụng kiểu hợp.
- Phân bổ kích thước bộ nhớ cho tất cả các thuộc tính của nó với kích thước của thuộc tính có kích thước lớn nhất.

- **Nhược điểm:**

- Chỉ cho phép truy cập vào một thuộc tính tại một thời điểm.
- Union phân bổ một không gian bộ nhớ chung duy nhất cho tất cả các thuộc tính
- Không phải tất cả các thuộc tính được khởi tạo.



SO SÁNH KIỂU HỢP VÀ KIỂU CẤU TRÚC

69

Nội dung	KIỂU HỢP	KIỂU CẤU TRÚC
Kích thước	- Lấy kích thước của thuộc tính có kích thước lớn nhất	- Lấy tổng kích thước của tất các thuộc
Cấp phát bộ nhớ	- Bộ nhớ được chia sẻ cho mỗi thuộc tính	- Mỗi thuộc tính được gán duy nhất một vùng nhớ
Khởi tạo	- Chỉ một thuộc tính đầu tiên được khởi tạo	- Một số thuộc tính có thể khởi tạo
Truy cập	- Tại một thời điểm, chỉ một thuộc tính (kích thước lớn nhất) mới có thể được truy cập và tất cả các thuộc tính khác sẽ chứa các giá trị rác.	- Tất cả các thuộc tính của nó có thể được truy cập bất cứ lúc nào.

VÍ DỤ VỀ CẤP PHÁT BỘ NHỚ CHO UNION

70

```
1  #include <iostream>
2  #include <cstring>
3  using namespace std;
4
5  union Sinhvien
6  {
7      int masv; int sdt; char ht[30];
8  };
9
10 int main()
11 {
12     union Sinhvien p1;
13     p1.masv = 1;
14     p1.sdt = 1234567822;
15     strcpy(p1.ht, "Nguyen Thanh An");
16     cout << "masv : " << p1.masv << endl;
17     cout << "sdt : " << p1.sdt << endl;
18     cout << "ht : " << p1.ht << endl;
19     return 0;
20 }
```

```
masv : 2037737294
sdt : 2037737294
ht : Nguyen Thanh An
```

"masv" và "sdt" chứa giá trị rác

```
1  #include <iostream>
2  #include <cstring>
3  using namespace std;
4
5  union Sinhvien
6  {
7      int masv; int sdt; char ht[30];
8  };
9
10 int main()
11 {
12     union Sinhvien p1;
13     p1.masv = 1;
14     cout << "masv : " << p1.masv << endl;
15     strcpy(p1.ht, "Nguyen Thanh An");
16     cout << "ht : " << p1.ht << endl;
17     p1.sdt = 1234567822;
18     cout << "sdt : " << p1.sdt << endl;
19     return 0;
20 }
```

```
masv : 1
ht : Nguyen Thanh An
sdt : 1234567822
```

- **Khai báo**

Tenkieuhop Tenmang[Kichthuoc];

- **Truy cập**

Tenmang[chiso].thuoctinh

- Khai báo

Tenkieuhop ***contro= new** Tenkieuhop[Kichthuoc];

```
Sinhvien *sv=new Sinhvien[100];
```

- Truy cập: 3 cách

```
contro[chiso].thuoctinh  
(*(contro+chiso)).thuoctinh  
(contro+chiso)-> thuoctinh
```

```
(*(sv+i)).masv;
```




KIỂU LIỆT KÊ

- Kiểu liệt kê (enum) dùng để khai báo một tập hợp các hằng số có tên
- Làm tăng kiểm tra thời gian biên dịch và tránh các lỗi xảy ra bằng cách chuyển vào các hằng không hợp lệ.
- Đảm bảo enum không lấy các giá trị nằm ngoài các giá trị được khai báo trong dấu ngoặc nhọn



ĐỊNH NGHĨA KIỂU LIỆT KÊ

75

- Cách 1: gồm 1 danh sách các hằng

```
enum tenenum  
{  
    Giatri_1,  
    Giatri_2,  
    .....,  
    Giatri_n  
};
```

```
5 □ enum Xeploai{  
6     Xuatsac, Gioi, Kha,  
7     TrungBinh, Yeu, Kem  
8     };  
9
```

ĐỊNH NGHĨA KIỂU LIỆT KÊ

76

- Cách 2: gồm 1 danh sách các hằng có gán giá trị

```
enum tenenum
{
    Giatri_1 [=Giatri_1],
    Giatri_2 [=Giatri_2],
    .....
    Giatri_n [=Giatri_n]
};
```

```
5  enum Xeploai{
6      Xuatsac, Gioi, Kha=6,
7      TrungBinh, Yeu, Kem
8  };
```

Giá trị của **TrungBinh** sẽ là 7

- Giá trị mặc định các hằng số này là kiểu int và bắt đầu từ 0 trở đi trong khai báo kiểu liệt kê.

- **Ví dụ:** Xuatxac=0,....., Kem=5

```
5 enum Xeploai{  
6     Xuatsac, Gioi, Kha,  
7     TrungBinh, Yeu, Kem  
8 };  
9
```

- Có thể gán giá trị cho một số tên theo bất kỳ thứ tự nào. Tất cả các tên chưa được chỉ định nhận giá trị là giá trị của tên trước đó cộng với một.

- **Ví dụ:** Xuatxac=0, Gioi=1,Kha=6,TrungBinh=7,Yeu=8, Kem=9

```
5 enum Xeploai{  
6     Xuatsac, Gioi, Kha=6,  
7     TrungBinh, Yeu, Kem  
8 };
```

- Tất cả các hằng số enum phải là duy nhất trong phạm vi của chúng
 - **Ví dụ:** Lỗi xuất hiện do 2 enum cùng giá trị Xuatsac trong file

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  enum XeploaiHocLuc{
6      Xuatsac=, Gioi, Kha,
7      TrungBinh, Yeu, Kem
8  };
9
10 enum XeploaiHanhKiem{
11     Xuatsac, Tot, Kha,
12     TrungBinh, Yeu
13 };
14
15 int main()
16 {
```

- 1. Khai báo

tenenum **tenbien**;

- 2. Gán giá trị

tenbien= Giatri_i;

- Ví dụ:

17	Xeploai xl;
18	xl=Gioi;

Cách 1

```
switch (tenbien) {  
    case Giatri_1:  
        caulenh_1; break;  
    .....  
    case Giatri_n-1 :  
        caulenh_n-1; break;  
default:  
    caulenh_n; break;  
}
```

Học sinh xếp loại 4: Yeu

```
5 enum Xeploai{  
6     Xuatsac, Gioi, Kha,  
7     TrungBinh, Yeu, Kem  
8 };  
9
```

```
18 Xeploai x1;  
19 x1=Yeu;  
20  
21 switch (x1) {  
22     case Kem:  
23         cout<<"Học sinh xếp loại "<<x1<<" : Kem"; break;  
24     case Yeu:  
25         cout<<"Học sinh xếp loại "<<x1<<" : Yeu"; break;  
26     case TrungBinh:  
27         cout<<"Học sinh xếp loại "<<x1<<" : TrungBinh"; break;  
28     case Kha:  
29         cout<<"Học sinh xếp loại "<<x1<<" : Kha"; break;  
30     case 6:  
31         cout<<"Học sinh xếp loại "<<x1<<" : Gioi"; break;  
32     default:  
33         cout<<"Học sinh xếp loại "<<x1<<" :Xuất xac"; break;  
34 }
```


- Cách 2:

```
for(int i= Giatri_1;i<= Giatri_n;i++)  
    caulenh_i;
```

```
36 |   for(int i=Xuatxac;i<=Kem;i++)  
37 |       cout<<"\n Hoc sinh xep loai:"<<i;
```

```
Hoc sinh xep loai:0  
Hoc sinh xep loai:1  
Hoc sinh xep loai:2  
Hoc sinh xep loai:3  
Hoc sinh xep loai:4  
Hoc sinh xep loai:5
```

DUYỆT KIỂU LIỆT KÊ

82

```
5 enum Xeploai{
6     Xuatsac, Gioi, Kha=6,
7     TrungBinh, Yeu, Kem
8 };
9
10 int main()
11 {
12     Xeploai xl=Kha;
13
14     switch (xl) {
15         case Kem:
16             cout<<"Hoc sinh xep loai "<<xl<<" : Kem"; break;
17         case Yeu:
18             cout<<"Hoc sinh xep loai "<<xl<<" : Yeu"; break;
19         case TrungBinh:
20             cout<<"Hoc sinh xep loai "<<xl<<" : TrungBinh"; break;
21         case 6:
22             cout<<"Hoc sinh xep loai "<<xl<<" : Kha"; break;
23         case Gioi:
24             cout<<"Hoc sinh xep loai "<<xl<<" : Gioi"; break;
25         default:
26             cout<<"Hoc sinh xep loai "<<xl<<" :Xuat sac"; break;
27     }
```

Hoc sinh xep loai 6: Kha

```
5 enum Xeploai{
6     Xuatsac, Gioi, Kha=6,
7     TrungBinh, Yeu, Kem
8 };
9
10 int main()
11 {
12     Xeploai xl=Kha;
13
14     switch (xl) {
15         case Kem:
16             cout<<"Hoc sinh xep loai "<<xl<<" : Kem"; break;
17         case Yeu:
18             cout<<"Hoc sinh xep loai "<<xl<<" : Yeu"; break;
19         case TrungBinh:
20             cout<<"Hoc sinh xep loai "<<xl<<" : TrungBinh"; break;
21         case 5:
22             cout<<"Hoc sinh xep loai "<<xl<<" : Kha"; break;
23         case Gioi:
24             cout<<"Hoc sinh xep loai "<<xl<<" : Gioi"; break;
25         default:
26             cout<<"Hoc sinh xep loai "<<xl<<" :Xuat sac"; break;
27     }
```

Hoc sinh xep loai 6:Xuat xac

Một cửa hàng bán xe máy cần lưu trữ thông tin về xe máy trong cửa hàng mình. Hãy sử dụng con trỏ động để quản lý danh sách xe máy với các thông tin sau: ten, hangsx, giá.

1. Nhập và hiển thị danh sách n xe máy vừa nhập
2. Đưa ra thông tin về những xe của hãng **Honda**.
3. Tìm xe có giá nhỏ nhất và xóa khỏi danh sách. Đưa ra danh sách sau khi xóa.



Một cửa hàng bán quần áo cần lưu trữ thông tin về sản phẩm trong cửa hàng mình. Hãy sử dụng con trỏ động để quản lý sản phẩm với các thông tin sau: ten, nuocsx, mau.

1. Nhập và hiển thị danh sách n sản phẩm vừa nhập
2. Đưa ra thông tin về những sản phẩm có màu XANH.
3. Tìm xem cửa hàng có sản phẩm của TQ không? Nếu có thì xóa khỏi danh sách. Đưa ra danh sách sau khi xóa.



Một cửa hàng bán máy tính cần lưu trữ thông tin về máy tính trong cửa hàng mình. Hãy sử dụng con trỏ động để quản lý máy tính với các thông tin sau: ten, hangsx, gia.

1. Nhập và hiển thị danh sách n máy tính vừa nhập
2. Đưa ra thông tin về những máy tính của hãng **Acer**
3. Chèn thêm vào danh sách 1 máy tính vào sau máy tính có giá nhỏ nhất. Đưa ra danh sách sau khi chèn.

