



ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH  
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

# SLIDE BÀI GIẢNG

## MÔN

# CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT



TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN, KHU PHỐ 6, PHƯỜNG LINH TRUNG, QUẬN THỦ ĐỨC, TP. HỒ CHÍ MINH

[T] 08 3725 2002 101 | [F] 08 3725 2148 | [W] [www.uit.edu.vn](http://www.uit.edu.vn) | [E] [info@uit.edu.vn](mailto:info@uit.edu.vn)



ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH  
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

# CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

## CHƯƠNG III

# CẤU TRÚC DỮ LIỆU ĐỘNG



Nguyễn Trọng Chính  
chinhnt@uit.edu.vn

[www.uit.edu.vn](http://www.uit.edu.vn)



# MỤC TIÊU CHƯƠNG III

- ❖ Hiểu các khái niệm về quản lý bộ nhớ trên C++
- ❖ Biết các cấu trúc danh sách liên kết
- ❖ Hiểu các thao tác trên danh sách liên kết đơn, liên kết kép và vận dụng vào các danh sách liên kết khác
- ❖ Áp dụng danh sách liên kết để giải quyết bài toán trong chương trình C++.



# CẤU TRÚC DỮ LIỆU ĐỘNG

- ❖ ĐẶT VẤN ĐỀ
- ❖ KIỂU DỮ LIỆU CON TRỎ
- ❖ DANH SÁCH LIÊN KẾT
- ❖ DANH SÁCH ĐƠN
- ❖ MỘT SỐ DẠNG DANH SÁCH LIÊN KẾT KHÁC



# DANH SÁCH ĐƠN

## ❖ TỔ CHỨC



Mỗi phần tử là một cấu trúc gồm:

- Dữ liệu: thông tin cần quản lý
- Liên kết: con trỏ có giá trị
  - Khác NULL: địa chỉ của phần tử liền sau
  - NULL nếu là phần tử cuối danh sách

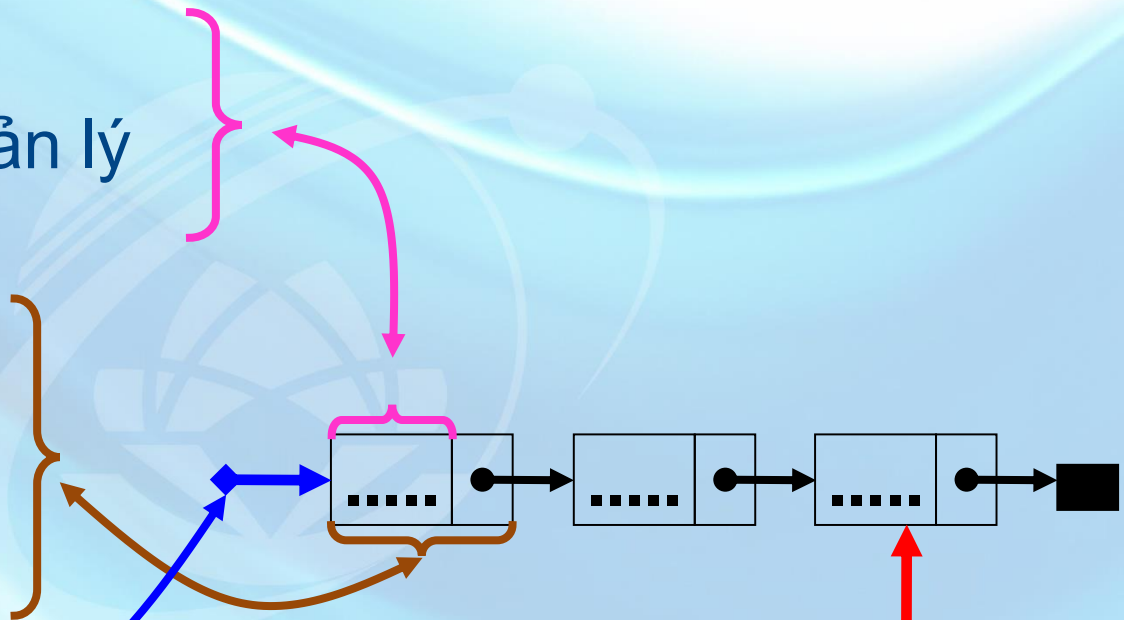




# DANH SÁCH ĐƠN

## ❖ TỔ CHỨC

```
struct TenDulieu {  
    ... // Thông tin quản lý  
};  
  
struct Node {  
    TenDulieu info;  
    Node * pNext;  
};  
  
struct TenDS {  
    Node *pHead, *pTail,  
};
```





# DANH SÁCH ĐƠN

## ❖ TỔ CHỨC

Ví dụ: Tổ chức dữ liệu cho một danh sách các hình tròn.

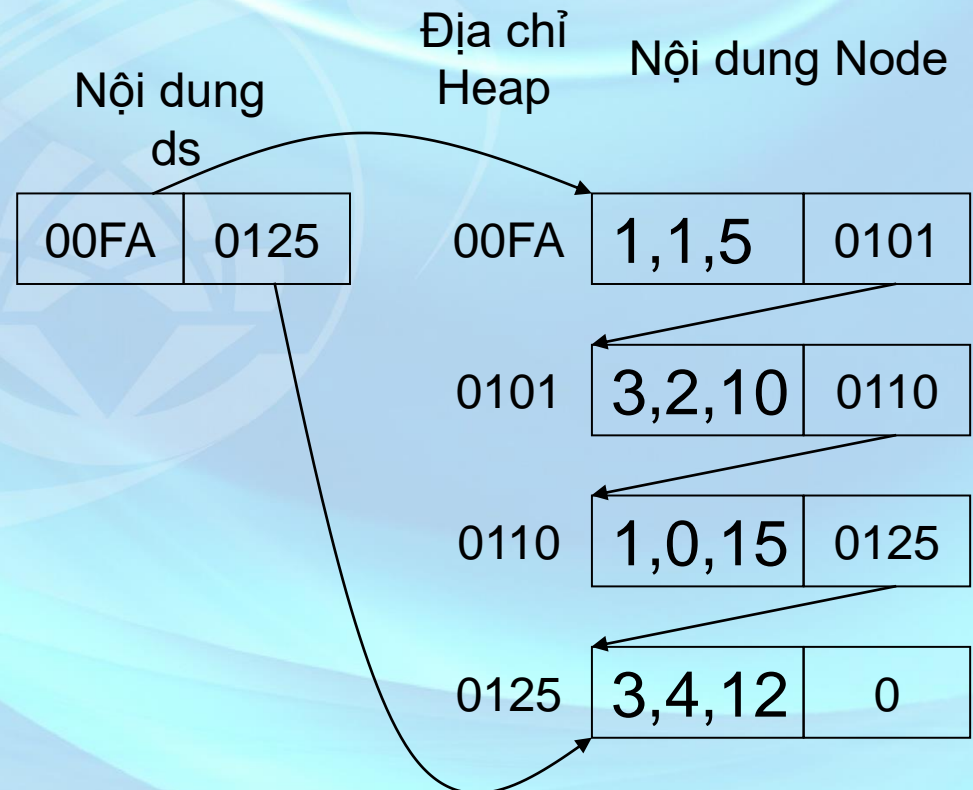
```
struct HìnhTron{  
    double x, y, r;  
};  
struct NodeHìnhTron {  
    HìnhTron info;  
    NodeHìnhTron *pNext;  
};
```



# DANH SÁCH ĐƠN

```
struct DSHinhTron{  
    NodeHinhTron  
    *pHead, *pTail;  
};
```

Giả sử có biến cấp phát tĩnh **ds** có kiểu **DSHinhTron** lưu trữ danh sách 4 hình tròn. Hình ảnh của ds như sau:







# DANH SÁCH ĐƠN

## ❖ CÁC THAO TÁC CƠ BẢN

- Tạo danh sách đơn rỗng
- Tạo một nút có trường info bằng x
- Thêm phần tử vào danh sách
- Duyệt danh sách
- Hủy phần tử trong danh sách
- Hủy danh sách
- Sắp xếp danh sách



# DANH SÁCH ĐƠN

## ❖ CÁC THAO TÁC CƠ BẢN

### - Tạo danh sách đơn rỗng

**pHead** và **pTail** trở đến **NULL**

```
void createList(TenDS &p) {  
    p.pHead = NULL; p.pTail = NULL;  
}
```

### Ví dụ

```
void createDSHinhTron(DSHinhTron &ds) {  
    ds.pHead = NULL; ds.pTail = NULL;  
}
```



# DANH SÁCH ĐƠN

## ❖ CÁC THAO TÁC CƠ BẢN

- Tạo một nút có trường info bằng x
  - B1: Cấp phát động một biến có kiểu **Node**
  - B2: Gán giá trị **x** cho trường **info**.
  - B3: Gán **pNext**  $\leftarrow$  **NULL**.



# DANH SÁCH ĐƠN

## ❖ CÁC THAO TÁC CƠ BẢN

- Tạo một nút có trường info bằng x

```
Node* createNode(TenDuLieu x) {  
    Node *p = new Node; // cấp phát vùng nhớ  
    if (p != NULL) { // kiểm tra kết quả cấp phát  
        p->info = x;  
        p->pNext = NULL;  
    }  
    return p;  
}
```



# DANH SÁCH ĐƠN

## ❖ CÁC THAO TÁC CƠ BẢN

- Tạo một nút có trường info bằng x

Ví dụ

```
NodeHinhTron* createDSHinhTron(HinhTron x) {  
    NodeHinhTron *p = new NodeHinhTron;  
    if (p != NULL) {  
        p->info = x;    p->pNext = NULL;  
    }  
    return p;  
}
```





# DANH SÁCH ĐƠN

## ❖ CÁC THAO TÁC CƠ BẢN

### - Thêm phần tử vào danh sách

Có các trường hợp:

- Thêm phần tử vào đầu danh sách
- Thêm phần tử vào cuối danh sách
- Thêm phần tử vào ngay sau phần tử  $q$  trong danh sách.



# DANH SÁCH ĐƠN

## ❖ CÁC THAO TÁC CƠ BẢN

- Thêm phần tử vào danh sách
- Thêm vào đầu danh sách
- Nếu ds rỗng:
  - $l.pHead \leftarrow p$
  - $l.pTail \leftarrow p$
- Ngược lại:
  - $p \rightarrow pNext \leftarrow l.pHead$
  - $l.pHead \leftarrow p$



# DANH SÁCH ĐƠN

## ❖ CÁC THAO TÁC CƠ BẢN

- Thêm phần tử vào danh sách
  - Thêm vào đầu danh sách

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



# DANH SÁCH ĐƠN

Địa chỉ      Nội dung

|

FFFE

**00FA**

0125

00FA

1,1,5

0101

0101

3,2,10

0110

0110

1,0,15

0125

0125

3,4,12

0

Địa chỉ

Nội dung

|

FFFE

**01FB**

0125

00FA

1,1,5

0101

0101

3,2,10

0110

0110

1,0,15

0125

0125

3,4,12

0

01FB

6,1,2

**00FA**



# DANH SÁCH ĐƠN

## ❖ CÁC THAO TÁC CƠ BẢN

- Thêm phần tử vào danh sách

- Thêm vào cuối danh sách

- Nếu ds rỗng:

  - $l.pHead \leftarrow p$

  - $l.pTail \leftarrow p$

- Ngược lại:

  - $l.pTail \rightarrow pNext \leftarrow p$

  - $l.Tail \leftarrow p$





# DANH SÁCH ĐƠN

## ❖ CÁC THAO TÁC CƠ BẢN

- Thêm phần tử vào danh sách
  - Thêm vào cuối danh sách

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



# DANH SÁCH ĐƠN

Địa chỉ      Nội dung

|

FFFE	00FA	0125
00FA	1,1,5	0101
0101	3,2,10	0110
0110	1,0,15	0125
0125	3,4,12	0

Địa chỉ      Nội dung

|

FFFE	00FA	01FB
00FA	1,1,5	0101
0101	3,2,10	0110
0110	1,0,15	0125
0125	3,4,12	01FB
01FB	6,1,2	0



# DANH SÁCH ĐƠN

## ❖ CÁC THAO TÁC CƠ BẢN

- Thêm phần tử vào danh sách
  - Thêm vào sau phần tử q trong danh sách
  - B1. Nếu  $q \neq \text{NULL}$   
$$p \rightarrow \text{pNext} \leftarrow q \rightarrow \text{pNext}$$
$$q \rightarrow \text{pNext} \leftarrow p,$$
ngược lại qua B3.
  - B2. Nếu  $l.\text{pTail} = q$  thì  $l.\text{pTail} \leftarrow p$ , qua B4.
  - B3. Thêm p vào đầu danh sách l.
  - B4. Kết thúc.



# DANH SÁCH ĐƠN

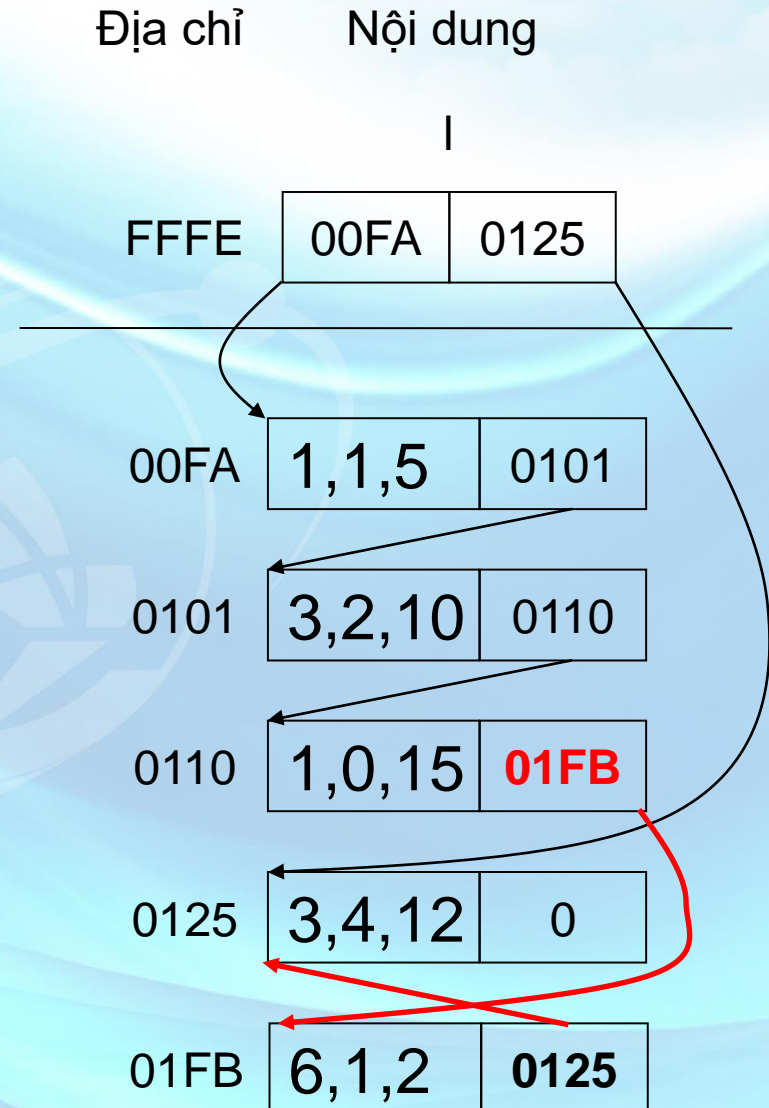
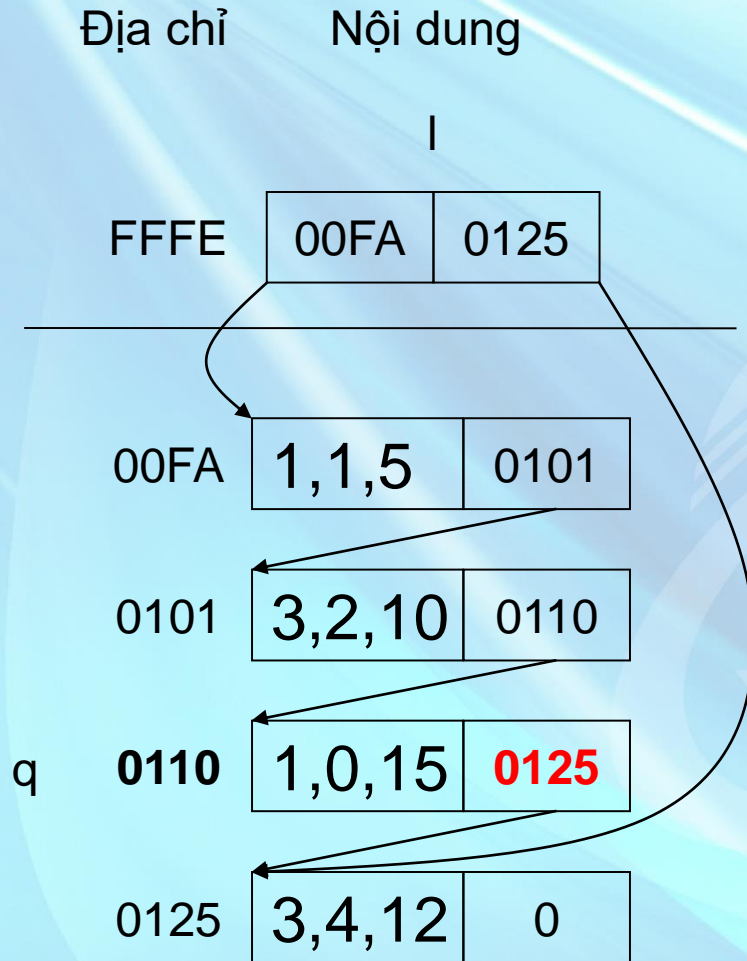
## ❖ CÁC THAO TÁC CƠ BẢN

- Thêm phần tử vào danh sách
  - Thêm vào sau phần tử q trong danh sách

```
void addAfter(TenDS &l, Node *p, Node *q) {  
    if (q != NULL) {  
        p->pNext = q->pNext; q->pNext = p;  
        if (l.pTail == q) l.pTail = p;  
    }  
    else addHead(l, p);  
}
```



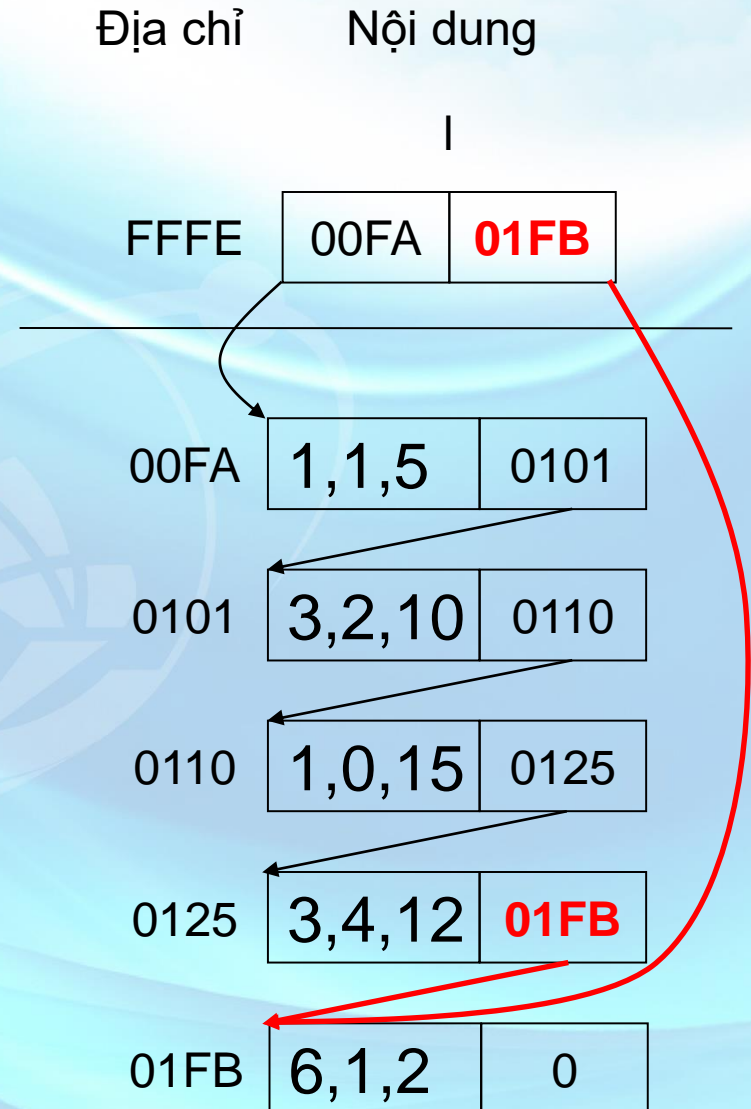
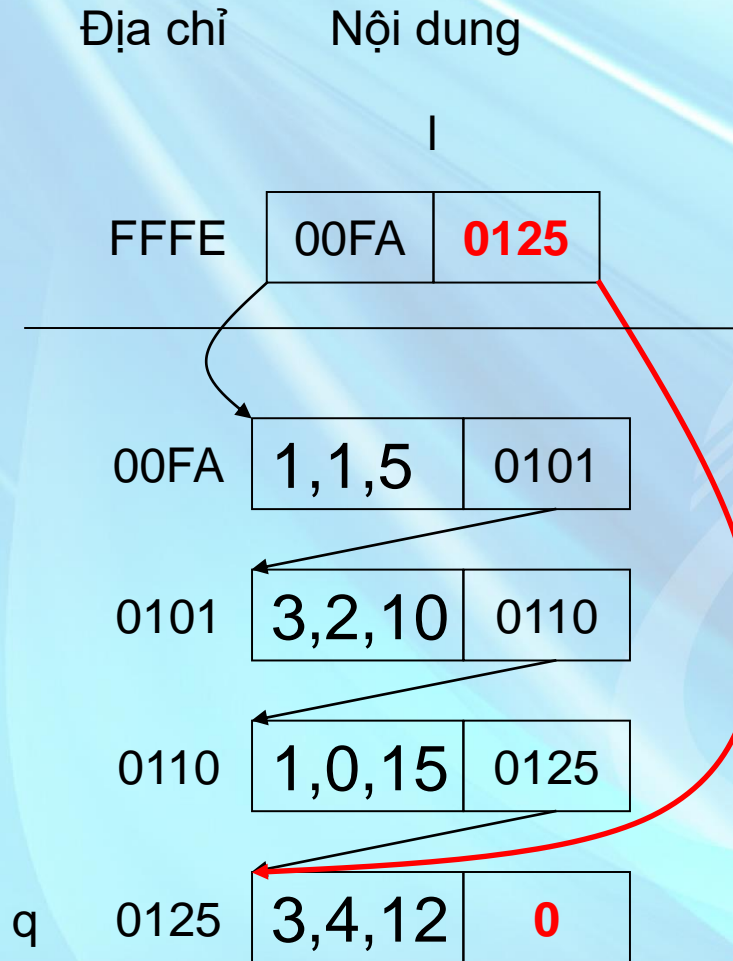
# DANH SÁCH ĐƠN







# DANH SÁCH ĐƠN





# DANH SÁCH ĐƠN

## ❖ CÁC THAO TÁC CƠ BẢN

- Duyệt danh sách
- Thực hiện tuần tự từ phần tử đầu danh sách đến phần tử cuối danh sách.
- Duyệt danh sách nhằm mục đích đếm số phần tử, tìm phần tử thỏa điều kiện.



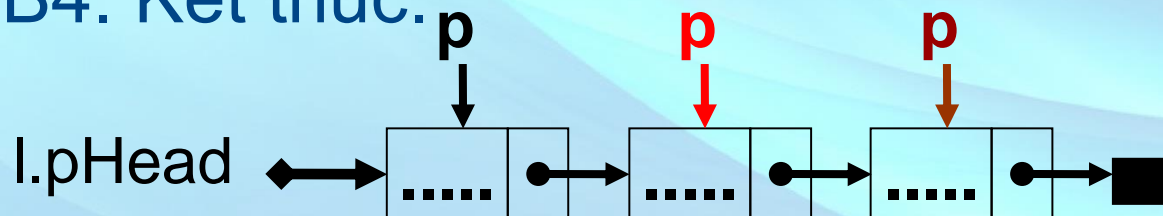
# DANH SÁCH ĐƠN

## ❖ CÁC THAO TÁC CƠ BẢN

### - Duyệt danh sách

Nguyên tắc: Để duyệt danh sách l

- B1.  $p \leftarrow l.pHead$
- B2. Nếu  $p = \text{NULL}$  qua B4
- B3. Xử lý cho phần tử  $p$ ,
- $p \leftarrow p \rightarrow pNext$ , qua B2.
- B4. Kết thúc.





# DANH SÁCH ĐƠN

## ❖ CÁC THAO TÁC CƠ BẢN

- Duyệt danh sách: *Tìm phần tử có trường info bằng x*
- Đầu vào: Danh sách l, giá trị x
- Đầu ra: phần tử p có giá trị x
  - B1.  $p \leftarrow l.pHead$
  - B2. Nếu  $p = NULL$  qua B4
  - B3. Nếu  $p \rightarrow info = x$ , qua B4,
    - Ngược lại:  $p \leftarrow p \rightarrow pNext$  qua B2.
  - B4. Kết quả tìm là p, kết thúc.



# DANH SÁCH ĐƠN

## ❖ CÁC THAO TÁC CƠ BẢN

- Duyệt danh sách: *Tìm phần tử có trường info bằng x*

`int equal(TenDuLieu x, TenDuLieu y);` // hàm so sánh

`Node * search(TenDS l, TenDuLieu x) {`

`Node *p = l.pHead;`

`while ((p != NULL) && (!equal(p->info, x))`

`p = p->pNext;`

`return p;`

`}`





# DANH SÁCH ĐƠN

## ❖ CÁC THAO TÁC CƠ BẢN

### Bài tập 1:

Viết chương trình cho phép:

- Nhập một danh sách hình tròn trong không gian 2 chiều cho tới khi nhập bán kính bằng 0.
- In ra màn hình các hình tròn có diện tích bằng s nhập từ bàn phím.



# DANH SÁCH ĐƠN

```
struct Circle {  
    double x, y, r;  
};  
struct CircleNode {  
    Circle info;  
    CircleNode *pNext;  
};  
struct CircleList {  
    CircleNode *pHead, *pTail;  
};
```



# DANH SÁCH ĐƠN

```
void createList(CircleList &l) {  
    l.pHead = NULL; l.pTail = NULL;  
}  
  
CircleNode* createNode(Circle x) {  
    CircleNode *p = new CircleNode;  
    if (p != NULL) {  
        p->info = x;    p->pNext = NULL;  
    }  
    return p;  
}
```



# DANH SÁCH ĐƠN

```
void addTail(CircleList &l, CircleNode *p) {  
    if (l.pHead == NULL) {  
        l.pHead = p; l.pTail = p;  
    } else {  
        l.pTail->pNext = p; l.pTail = p;  
    }  
}
```



# DANH SÁCH ĐƠN

```
int compare(Circle x, double s) {  
    double stmp = x.r * x.r * 3.14;  
    if (stmp == s) return 0;  
    if (stmp < s) return -1;  
    return 1;  
}  
  
void print(Circle x) {  
    cout << '(' << x.x << ", " << x.y << "), " << x.r << '  
    ';  
}
```





# DANH SÁCH ĐƠN

```
void browse(CircleList &l, double s) {  
    CircleNode *p = l.pHead;  
    while (p != NULL) {  
        if (compare(p->info, s) == 0)  
            print(p->info);  
        p = p->pNext;  
    }  
}
```



# DANH SÁCH ĐƠN

```
void inputList(CircleList &l) {  
    CircleNode *p;  
    Circle x;  
    cin >> x.x >> x.y >> x.r;  
    while (x.r > 0) {  
        p = createNode(x);  
        if (p == NULL) return;  
        addTail(l, p);  
        cin >> x.x >> x.y >> x.r;  
    }  
}
```



# DANH SÁCH ĐƠN

```
int main(int argc, char **argv) {  
    CircleList list;  
    double s;  
    createList(list);  
    inputList(list);  
    cin >> s;  
    browse(list, s);  
    return EXIT_SUCCESS;  
}
```



# DANH SÁCH ĐƠN

## ❖ CÁC THAO TÁC CƠ BẢN

### Bài tập 2:

Với đề bài như Bài tập 1,  
Viết hàm filter dùng để lọc các hình tròn có tọa độ tâm nằm trong góc phần tư thứ nhất của mặt phẳng tọa độ. Cho nguyên mẫu của hàm filter như sau:

```
void filter(CircleList in, CircleList &out);
```



# DANH SÁCH ĐƠN

```
int check(Circle x) {  
    if ((x.x >= 0) && (x.y >= 0))  
        return 1;  
    return 0;  
}
```





# DANH SÁCH ĐƠN

```
void filter(CircleList in, CircleList &out) {  
    CircleNode *p = in.pHead, *pTmp;  
    createList(out);  
    while (p != NULL) {  
        if (check(p->info)) {  
            pTmp = createNode(p->info);  
            if (pTmp == NULL) return;  
            addTail(out, pTmp);  
        }  
        p = p->pNext;  
    }  
}
```



# DANH SÁCH ĐƠN

## ❖ CÁC THAO TÁC CƠ BẢN

- Hủy một phần tử trong danh sách: Xét các trường hợp sau:
  - Hủy phần tử đầu danh sách
  - Hủy phần tử ngay sau phần tử  $q$  trong danh sách
  - Hủy phần tử có khóa  $x$



# DANH SÁCH ĐƠN

## ❖ CÁC THAO TÁC CƠ BẢN

- Hủy một phần tử trong danh sách:
- Hủy phần tử đầu danh sách
- Đầu vào: Danh sách l
- Đầu ra: Danh sách l,  
Dữ liệu x của node bị hủy,  
Kết quả thực hiện r có các giá trị:
  - + 0  $\rightarrow$  không xóa được
  - + 1  $\rightarrow$  xóa thành công



# DANH SÁCH ĐƠN

## ❖ CÁC THAO TÁC CƠ BẢN

- Hủy một phần tử trong danh sách:
  - Hủy phần tử đầu danh sách
    - B1) Nếu  $l.pHead = NULL$  thì  $r \leftarrow 0$  qua B4
    - B2)  $p \leftarrow l.pHead$ ,  $l.pHead \leftarrow p \rightarrow pNext$ ,  
 $x \leftarrow p \rightarrow info$ ,  
 $r \leftarrow 1$ , giải phóng  $p$ .
    - B3) Nếu  $l.pHead = NULL$  thì  $l.pTail \leftarrow NULL$
    - B4) Trả về  $r$ . Kết thúc



# DANH SÁCH ĐƠN

## ❖ CÁC THAO TÁC CƠ BẢN

- Hủy một phần tử trong danh sách:
- Hủy phần tử đầu danh sách

```
int removeHead(TenDS &l, TenDulieu &x) {  
    Node *p = l.pHead; int r = 0;  
    if (l.pHead != NULL) {  
        x = p->info; l.pHead = p->pNext;  
        delete p; r = 1;  
        if (l.pHead == NULL) l.pTail = NULL;  
    }  
    return r;  
}
```





# DANH SÁCH ĐƠN

Địa chỉ      Nội dung

|

FFFE

**00FA**

0125

00FA

5

0101

0101

2

0110

0110

7

0125

0125

1

0

Địa chỉ

Nội dung

|

FFFE

**0101**

0125

00FA

5

0101

0101

2

0110

0110

7

0125

0125

1

0



# DANH SÁCH ĐƠN

## ❖ CÁC THAO TÁC CƠ BẢN

- Hủy một phần tử trong danh sách:
  - Hủy phần tử ngay sau phần tử  $q$  trong danh sách
- Đầu vào: Danh sách  $l$ , phần tử  $q$
- Đầu ra: Danh sách  $l$ ,  
Dữ liệu  $x$  của phần tử bị hủy  
Kết quả thực hiện  $r$  với các giá trị:
  - + 0  $\rightarrow$  không thực hiện được
  - + 1  $\rightarrow$  thực hiện thành công



# DANH SÁCH ĐƠN

## ❖ CÁC THAO TÁC CƠ BẢN

- Hủy một phần tử trong danh sách:
  - Hủy phần tử ngay sau phần tử q trong danh sách
    - B1. Nếu  $q = \text{NULL}$  thì  $r \leftarrow 0$ , qua B5
    - B2.  $p \leftarrow q \rightarrow \text{pNext}$ ,  
nếu  $p = \text{NULL}$  thì  $r \leftarrow 0$ , qua B5
    - B3. Nếu  $p = l.\text{pTail}$  thì  $l.\text{pTail} \leftarrow q$
    - B4.  $x \leftarrow p \rightarrow \text{info}$ ,  $q \rightarrow \text{pNext} \leftarrow p \rightarrow \text{pNext}$ ,  
 $r \leftarrow 1$ , giải phóng  $p$
    - B5. Trả về  $r$ . Kết thúc



# DANH SÁCH ĐƠN

```
int removeAfter(TenDS &l, Node *q, TenDulieu &x) {  
    Node *p;    int r = 0;  
    if (q != NULL) {    p = q->pNext;  
        if (p != NULL) {  
            if (l.pTail == p) l.pTail = q;  
            x = p->info; q->pNext = p->pNext;  
            delete p; r = 1;  
        }  
    }  
    return r;  
}
```



# DANH SÁCH ĐƠN

## ❖ CÁC THAO TÁC CƠ BẢN

- Hủy một phần tử trong danh sách:

- Hủy phần tử có khóa x

- Đầu vào: Danh sách l, khóa x

- Đầu ra: Danh sách l

Kết quả thực hiện r với các giá trị

+ 0 → không thực hiện được

+ 1 → thực hiện thành công





# DANH SÁCH ĐƠN

## ❖ CÁC THAO TÁC CƠ BẢN

- Hủy một phần tử trong danh sách:
  - Hủy phần tử có khóa x
    - B1. Tìm p có khóa x và q là phần tử trước p.
    - B2. Nếu p = NULL thì  $r \leftarrow 0$ , qua B4
    - B3. Nếu q = NULL thì  $r \leftarrow \text{removeHead}(l, x)$  ;
      - ngược lại thì  $r \leftarrow \text{removeAfter}(l, q, x)$ .
    - B4. Trả về r. Kết thúc



# DANH SÁCH ĐƠN

```
int remove(TenDS &l, TenDulieu x) {  
    Node *p = l.pHead, *q = NULL; int r = 0;  
    while ((p != NULL) && (!Equal(p->info, x))) {  
        q = p; p = p->pNext;  
    }  
    if (p != NULL)  
        if (q == NULL) r = removeHead(l,x);  
        else r = removeAfter(l, q, x);  
    return r;  
}
```



# DANH SÁCH ĐƠN

## ❖ CÁC THAO TÁC CƠ BẢN

### - Hủy danh sách:

- Đầu vào: Danh sách l.
- Đầu ra: Danh sách l rỗng.
- B1. Nếu l.pHead = NULL qua B3
- B2.  $p \leftarrow l.pHead$ ,  $l.pHead \leftarrow p \rightarrow pNext$ ,  
giải phóng p, qua B1.
- B3.  $l.pTail = NULL$ .
- B4. Kết thúc.



# DANH SÁCH ĐƠN

## ❖ CÁC THAO TÁC CƠ BẢN

### - Hủy danh sách:

```
void removeList(TenDS &l) {  
    Node *p;  
    while (l.pHead != NULL) {  
        p = l.pHead; l.pHead = p->pNext;  
        delete p;  
    }  
    l.pTail = NULL;  
}
```



# DANH SÁCH ĐƠN

## ❖ CÁC THAO TÁC CƠ BẢN

### - Sắp xếp danh sách:

Danh sách có thể được sắp xếp theo hai cách

- Hoán đổi thành phần info của các phần tử trong danh sách
- Thiết lập lại liên kết giữa các phần tử trong danh sách





# DANH SÁCH ĐƠN

## ❖ CÁC THAO TÁC CƠ BẢN

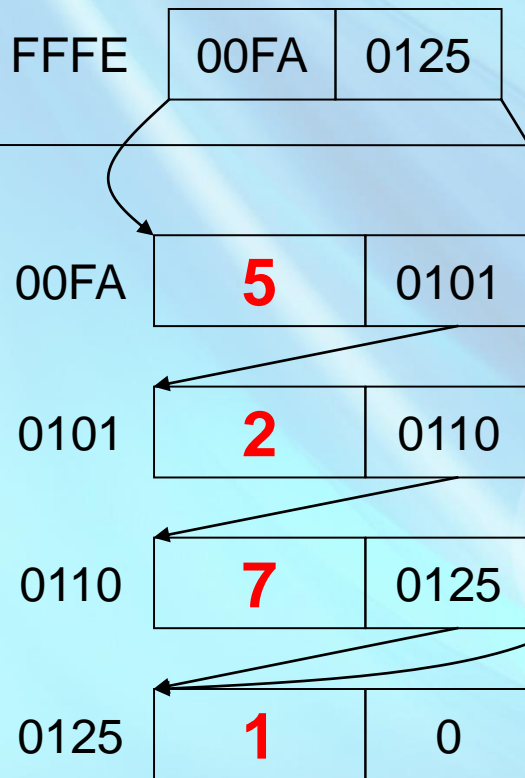
- Sắp xếp danh sách
- Hoán đổi thành phần info của các phần tử trong danh sách:
  - Cài đặt đơn giản, tương tự sắp xếp mảng
  - Khi kích thước của info lớn, chi phí cho việc hoán đổi rất lớn dẫn đến thời gian sắp xếp chậm.



# DANH SÁCH ĐƠN

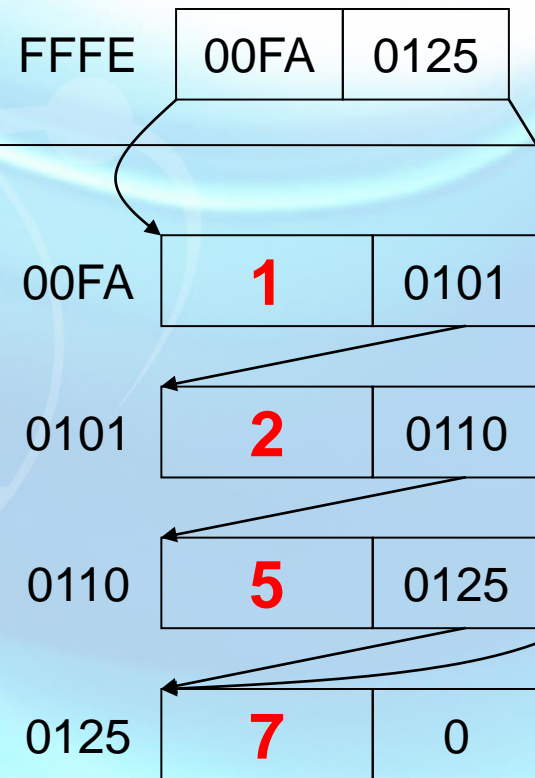
Địa chỉ      Nội dung

|



Địa chỉ      Nội dung

|





# DANH SÁCH ĐƠN

## ❖ CÁC THAO TÁC CƠ BẢN

- Sắp xếp danh sách
- Thiết lập lại liên kết giữa các phần tử trong danh sách:
  - Cài đặt phức tạp.
  - Chi phí hoán đổi liên kết cho một phần tử không chịu ảnh hưởng của trường info nên thời gian sắp xếp nhanh.

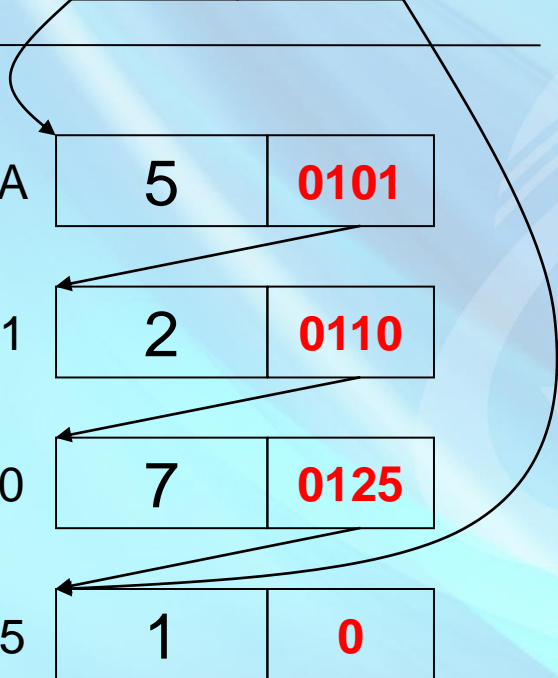


# DANH SÁCH ĐƠN

Địa chỉ      Nội dung

|

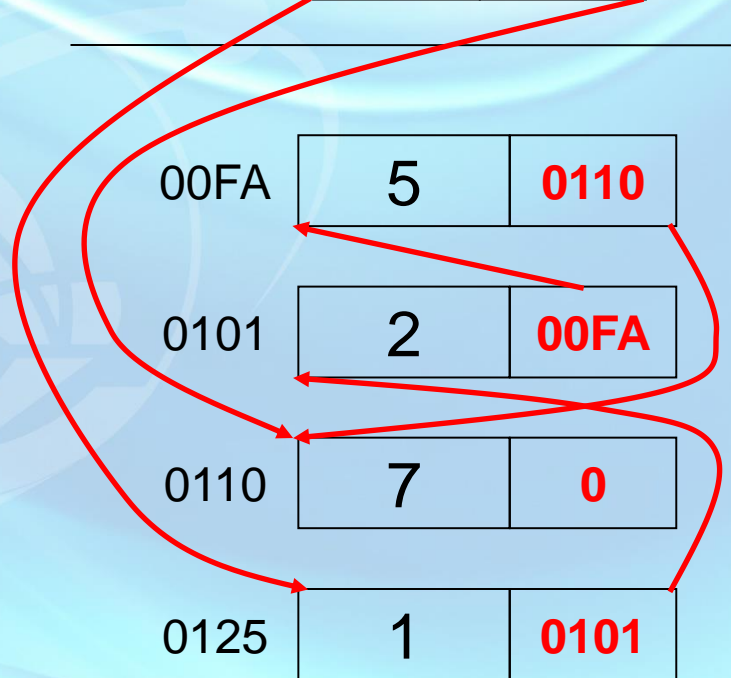
FFFE	00FA	0125
00FA	5	0101
0101	2	0110
0110	7	0125
0125	1	0



Địa chỉ      Nội dung

|

FFFE	0125	0110
00FA	5	0110
0101	2	00FA
0110	7	0
0125	1	0101





# DANH SÁCH ĐƠN

## ❖ CÁC THAO TÁC CƠ BẢN

- Sắp xếp danh sách
- Thiết lập lại liên kết giữa các phần tử trong danh sách: rất thích hợp với các giải thuật sắp xếp
  - Merge Sort
  - Radix Sort