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 | [E] 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 Chỉnh
chinhnt@uit.edu.vn
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 KÉP

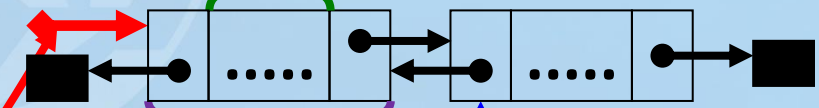
TỔ CHỨC

- Mỗi phần tử chứa liên kết đến phần tử đứng liền trước và sau nó
- Mỗi phần tử là một cấu trúc gồm 3 thành phần:
  - Thành phần dữ liệu: chứa thông tin cần quản lý
  - Hai thành phần liên kết: chứa địa chỉ của phần tử liền trước và sau nó, hoặc chứa giá trị NULL.

DANH SÁCH KÉP

❖TỔ CHỨC

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



DANH SÁCH KÉP

❖TỔ CHỨC

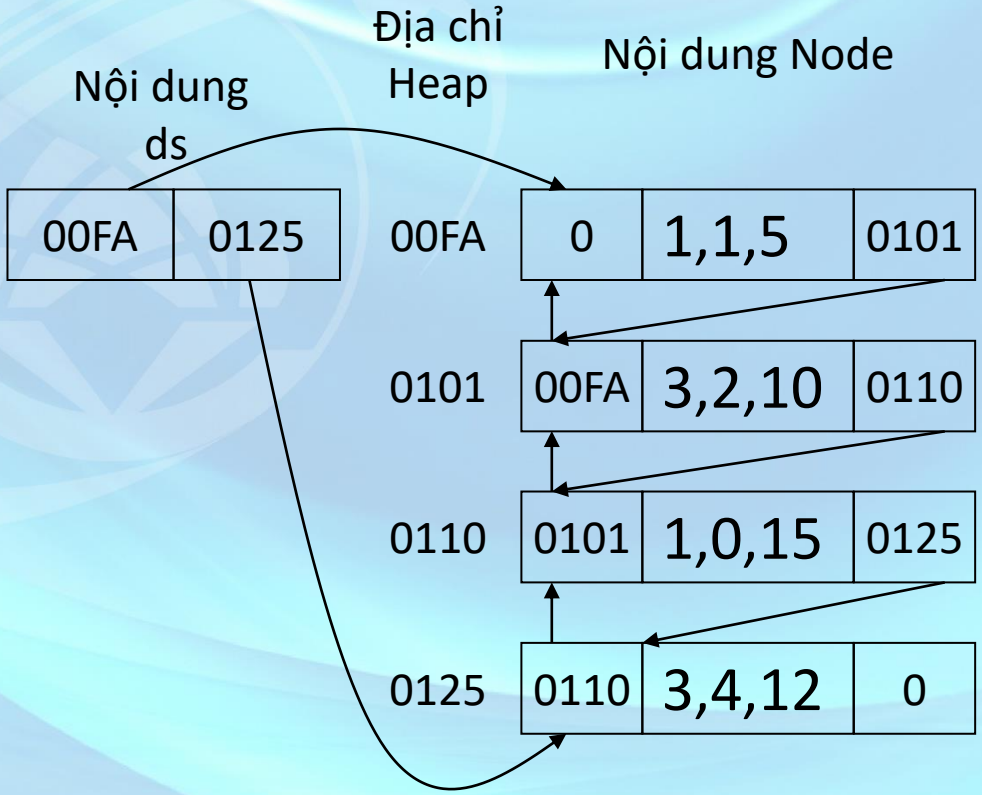
Ví dụ: Tổ chức dữ liệu cho một danh sách các hình tròn.

```
struct  HinhTron{
    double x, y, r;
};
struct NodeHinhTron {
    HinhTron info;
    NodeHinhTron *pNext, *pPrev;
};
```

DANH SÁCH KÉP

```
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 KÉP

❖CÁC THAO TÁC CƠ BẢN

- Tạo danh sách 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

Lưu ý: Các thao tác được thực hiện như danh sách đơn, cần duy trì liên kết với phần tử trước

DANH SÁCH KÉP

❖CÁC THAO TÁC CƠ BẢN

- **Tạo danh sách đơn rỗng**

  Danh sách rỗng có pHead và pTail trỏ đến NULL

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

Ví dụ

```
void CreateDSHinhTron(DSHinhTron &p) {
   p.pHead = NULL; p.pTail = NULL;
}
```

DANH SÁCH KÉP

❖CÁC THAO TÁC CƠ BẢN

- **Tạo một nút có trường info bằng x**

- Cấp phát động một biến có kiểu Node,
- Gán giá trị x cho trường info.
- Gán pNext và pPrev bằng giá trị NULL.

DANH SÁCH KÉP

❖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->pPrev = NULL; p->pNext = NULL;
    }
    return p;
}
```

DANH SÁCH KÉP

❖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->pPrev = NULL; p->pNext = NULL; }
    return p;
}
```

DANH SÁCH KÉP

❖CÁC THAO TÁC CƠ BẢN

- **Thêm phần tử vào danh sách**

  Xét việc thêm phần tử vào danh sách theo các trường hợp sau:

  - 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.
  - Thêm phần tử vào ngay trước phần tử q trong danh sách.

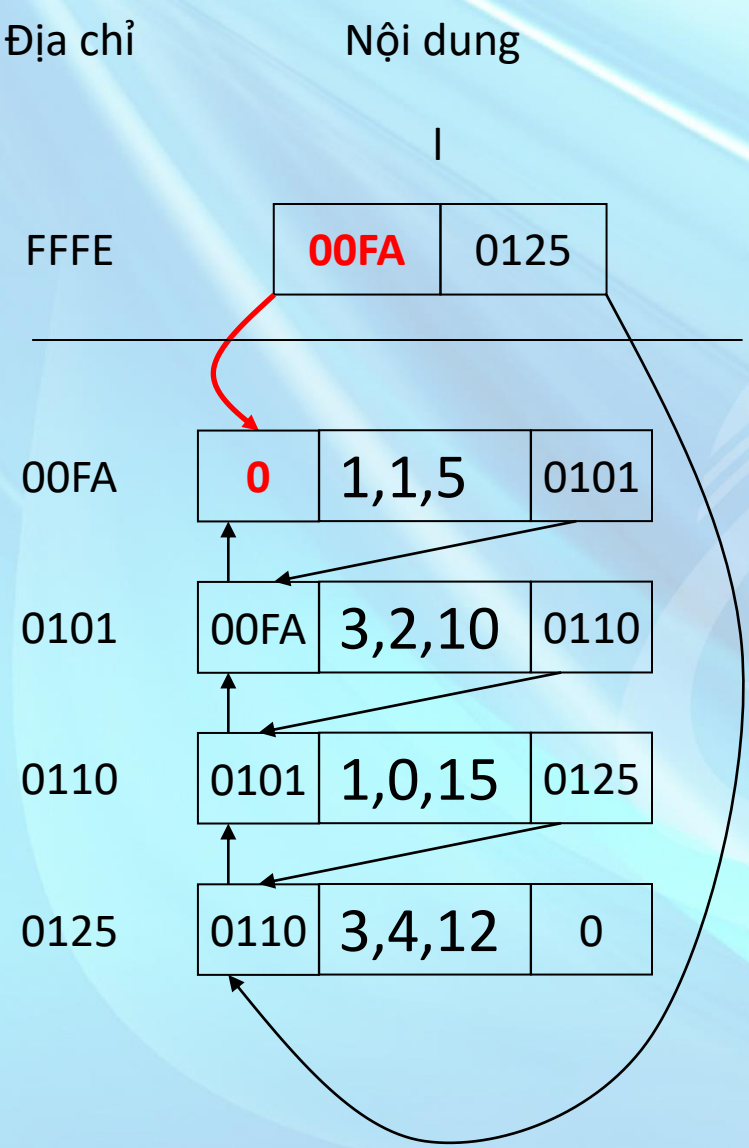
DANH SÁCH KÉP

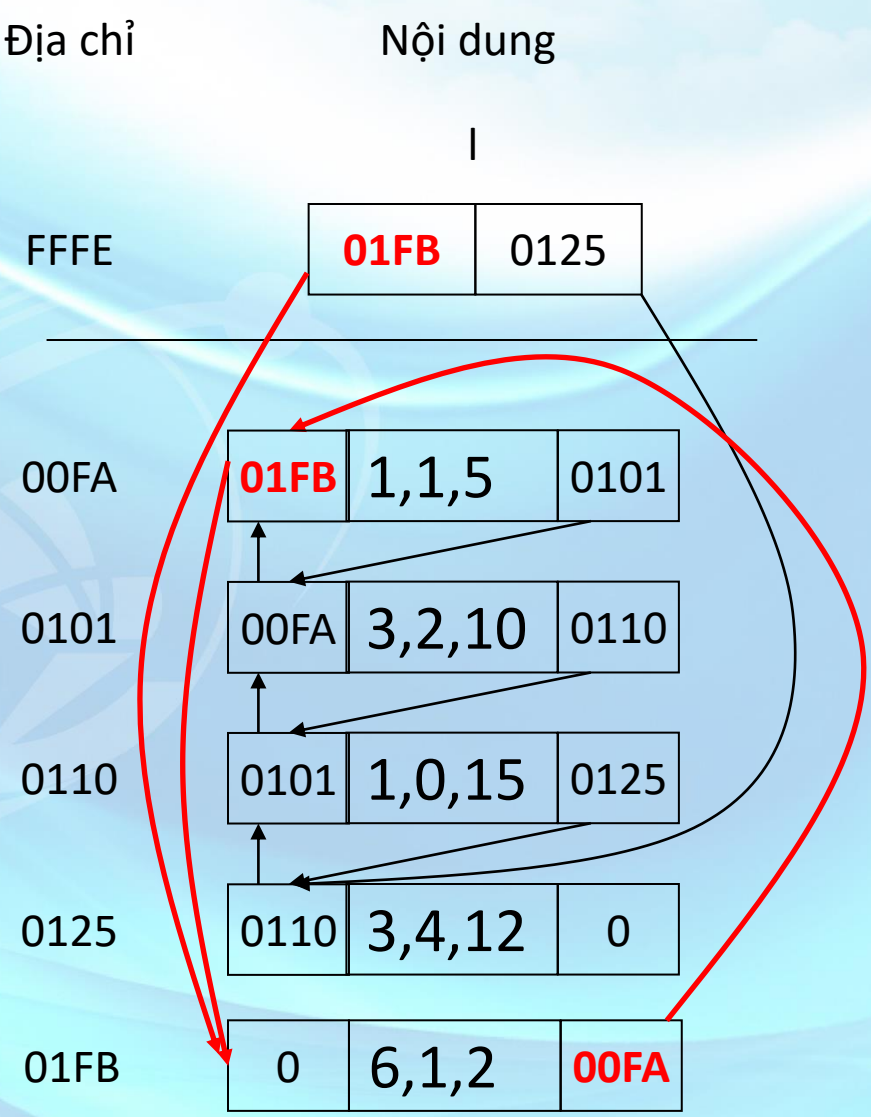
❖CÁC THAO TÁC CƠ BẢN

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

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

DANH SÁCH KÉP





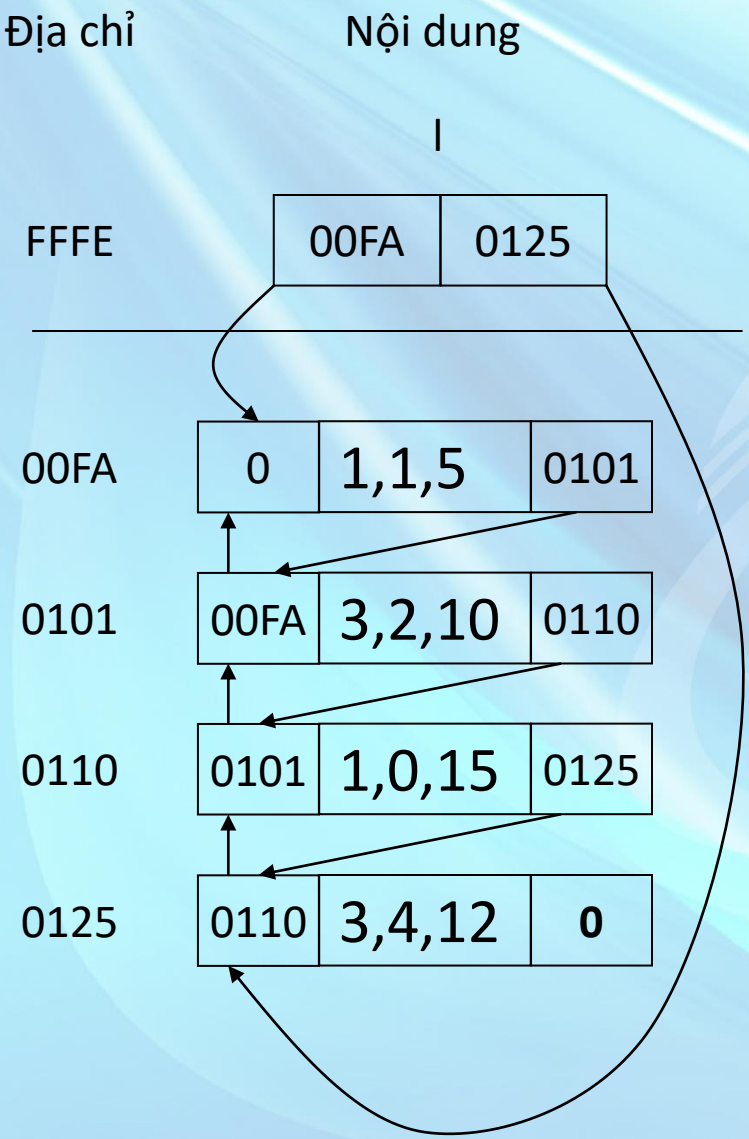
DANH SÁCH KÉP

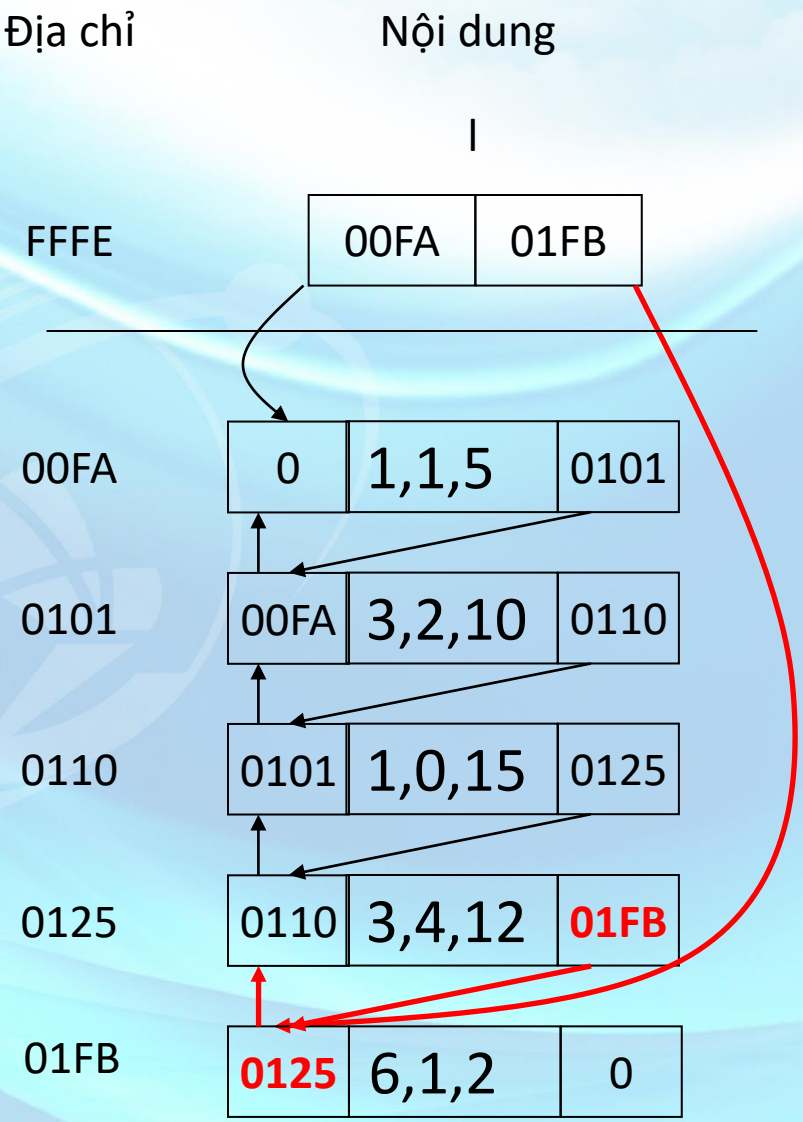
❖CÁC THAO TÁC CƠ BẢN

- *Thêm vào cuối danh sách*

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

DANH SÁCH KÉP





DANH SÁCH KÉP

❖CÁC THAO TÁC CƠ BẢN

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

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

DANH SÁCH KÉP

❖CÁC THAO TÁC CƠ BẢN

- *Thêm vào trước phần tử q trong danh sách*

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

DANH SÁCH KÉP

❖CÁC THAO TÁC CƠ BẢN

- **Duyệt danh sách**

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

DANH SÁCH KÉP

❖CÁC THAO TÁC CƠ BẢN

- **Duyệt danh sách**

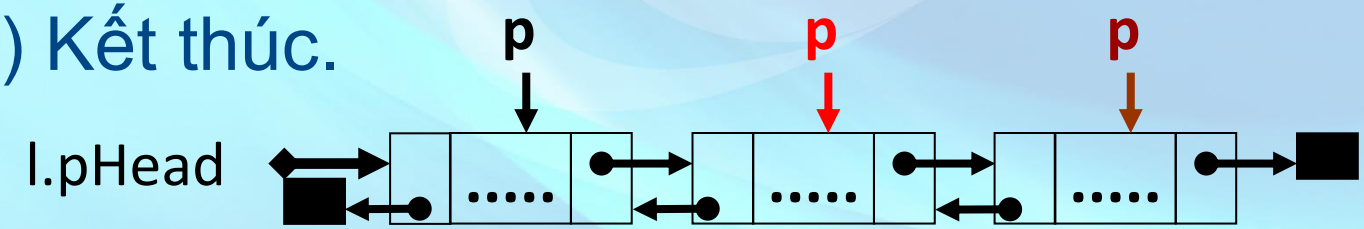
Nguyên tắc: Để duyệt danh sách l

B1) p ← l.pHead

B2) Nếu p = NULL qua B4

B3) Xử lý cho phần tử p, p ← p->pNext, qua B2.

B4) Kết thúc.



Lưu ý: Có thể duyệt từ phần tử cuối theo pPrev

DANH SÁCH KÉP

❖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 KÉP

❖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 cuối danh sách
  - Hủy phần tử ngay sau phần tử q trong danh sách
  - Hủy phần tử ngay trước phần tử q trong danh sách
  - Hủy phần tử có khóa x

DANH SÁCH KÉP

❖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;
       else l.pHead->pPrev = NULL;    }
   return r;      }
```

DANH SÁCH KÉP

❖CÁC THAO TÁC CƠ BẢN

- **Hủy một phần tử trong danh sách:**
  - *Hủy phần tử cuối danh sách*

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

DANH SÁCH KÉP

❖CÁC THAO TÁC CƠ BẢN

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

- *Hủy phần tử ngay sau phần tử q trong danh sách*

```
int removeAfter(TenDS &l, Node *q, TenDulieu &x) {
    Node *p;
    if (q != NULL)  {
        p = q->pNext;
        if (p != NULL) {
            q->pNext = p->pNext;
            if (p==l.pTail) l.pTail=q;
            else p->pNext->pPrev=q;
```

DANH SÁCH KÉP

```
            x = p->info; delete p;
        }
        return 1;
    }
    else return removeHead(l, x);
}
```

DANH SÁCH KÉP

❖CÁC THAO TÁC CƠ BẢN

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

- *Hủy phần tử ngay trước phần tử q trong danh sách*

```
int RemoveBefore(TenDS &l, Node *q, TenDulieu &x) {
    Node *p;
    if (q != NULL) {
        p = q->pPrev;
        if (p != NULL) {
            q->pPrev = p->pPrev;
            if(p==l.pHead) l.pHead=q;
            else p->pPrev->pNext=q;
```

DANH SÁCH KÉP

```
            x = p->info; delete p;
        }
        return 1;
    }
    return removeTail(l, x);
}
```

DANH SÁCH KÉP

❖CÁC THAO TÁC CƠ BẢN

- **Hủy một phần tử trong danh sách:**
- *Hủy phần tử có khóa x*

```
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;
    }
```

DANH SÁCH KÉP

```
    if (p != NULL)
        if (q == NULL) r = removeHead(l,x);
        else r = removeAfter(l, q, x);
    return r;
}
```

DANH SÁCH KÉP

❖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 KÉP

❖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 KÉP

❖CÁC THAO TÁC CƠ BẢN

- **Sắp xếp danh sách:** Một số thuật toán hiệu quả
  - *Quick Sort*
  - *Merge Sort*
  - *Radix Sort*