# Cấu trúc dữ liệu và Giải thuật
# Danh Sách và Danh Sách Liên Kết

Bùi Ngọc Thăng

September 30, 2015

# Outline

# Mục tiêu bài học

1. Tìm hiểu các khái niệm danh sách cài đặt bởi mảng và danh sách liên kết
2. Các phép toán trên danh sách và danh sách liên kết
3. Cài đặt các phép toán trên các loại danh sách

# Danh sách được cài đặt bởi mảng

1. Khái niệm danh sách
2. Các phép toán trên danh sách
3. Cài đặt danh sách và các phép toán (C/C++)

# Danh sách: Khái niệm

## Definition

**Danh sách:** Là một cấu trúc dữ liệu lưu trữ tập các phần tử cùng kiểu dữ liệu.

**Tính chất của danh sách:**

1. Các phần tử trong danh sách sắp xếp được hoặc không
2. Cấu trúc có thể cài đặt không phụ thuộc vào ngôn ngữ lập trình
3. Các phần tử trong danh sách phải cùng kiểu dữ liệu
4. Mỗi phần tử trong danh sách có thể xác định được bởi chỉ số/ các chỉ số của danh sách.

# Danh sách: Ví dụ minh hoạ

Table: Mảng lưu trữ 04 số nguyên

| 12 | 5 | 7 | 20 |
|---|---|---|---|

Table: Bảng lưu trữ 16 số thực

| | | | |
|---|---|---|---|
| 12.5 | 5.3 | 7.2 | 20.9 |
| 12.5 | 5.23 | 8.2 | 11.8 |
| 17.8 | 7.13 | 6.2 | 12.6 |
| 17.5 | 4.3 | 8.2 | 6.9 |

# Danh sách: Ví dụ minh hoạ

**DANH SÁCH SINH VIÊN THEO LỚP MÔN HỌC**

Môn học: **Lập trình nâng cao** Mã lớp môn học: **INT 2027** Số tín chỉ: 4
Thứ - Tiết: **6, 2 - 4** Giảng đường: 303-G2

| STT | Mã SV | Họ và tên | Ngày sinh | Lớp khóa học | Điểm giữa kỳ lần 1 | Điểm giữa kỳ lần 2 | Ghi chu |
|---|---|---|---|---|---|---|---|
| 1 | 11020004 | Đinh Trung Anh | 1/8/94 | QH-2011-I/CQ-C-A | 9 | 10 | |
| 2 | 11020531 | Phan Tuấn Anh | 8/27/93 | QH-2011-I/CQ-C-A | 10 | 10 | |
| 3 | 11020033 | Đặng Minh Công | 3/19/93 | QH-2011-I/CQ-C-A | 8 | 9.5 | |
| 4 | 11020055 | Đặng Minh Dũng | 6/10/93 | QH-2011-I/CQ-C-A | 10 | 9.5 | |
| 5 | 11020530 | Nguyễn Đình Dũng | 4/17/93 | QH-2011-I/CQ-C-A | 8.5 | 9.5 | |
| 6 | 11020412 | Nguyễn Văn Đại | 9/8/93 | QH-2011-I/CQ-C-A | 9.5 | 8.5 | |
| 7 | 11020067 | Mai Công Đạt | 1/21/93 | QH-2011-I/CQ-C-A | 5 | 8.5 | |
| 8 | 11020070 | Nguyễn Thành Đạt | 5/25/93 | QH-2011-I/CQ-C-A | 0 | | |
| 9 | 11020072 | Phạm Tất Đạt | 1/17/93 | QH-2011-I/CQ-C-A | 7 | 7 | |
| 13 | 11020095 | Phạm Trần Hương Giang | 10/30/93 | QH-2011-I/CQ-C-A | 4 | 8.5 | |
| 14 | 11020107 | Vũ Minh Hải | 1/15/93 | QH-2011-I/CQ-C-A | 8.5 | 9 | |
| 15 | 11020417 | Võ Anh Hưng | 3/13/93 | QH-2011-I/CQ-C-A | 7 | 9 | |
| 16 | 11020157 | Trịnh Xuân Hướng | 5/25/93 | QH-2011-I/CQ-C-A | 4 | 7 | |
| 21 | 11020179 | Nguyễn Thanh Lịch | 3/4/93 | QH-2011-I/CQ-C-A | 0 | | |
| 22 | 11020186 | Ngô Lê Bảo Lộc | 3/14/93 | QH-2011-I/CQ-C-A | 5 | 9 | |
| 24 | 11020194 | Trần Minh Luyện | 3/29/93 | QH-2011-I/CQ-C-A | 7 | 10 | |
| 25 | 11020201 | Nguyễn Văn Minh | 9/10/87 | QH-2011-I/CQ-C-A | 0 | | |
| 26 | 11020204 | Trần Đức Mười | 7/26/93 | QH-2011-I/CQ-C-A | 10 | 10 | |
| 27 | 11020206 | Dương Đức Nam | 2/27/93 | QH-2011-I/CQ-C-A | 4 | 8 | |
| 28 | | Vũ Tiến Đạt | | | 4 | 7 | |

# Danh sách: Cài đặt cấu trúc dữ liệu

**I. Các phần tử được lưu trữ trong mảng một chiều**

```
const int max =...// Số lượng tối đa các phần tử mà danh sách có thể lưu trữ
struct List{
        item element[max];// mảng lưu các phần tử có kiểu item
        int count;// số lượng các phần tử trong danh sách
} ;
```

**II. Các phần tử được lưu trữ trong mảng 2 chiều**

```
const int max1 =...
const int max2 =...
struct List{
        item element[max1][max2];
        int count;
};
```

# Danh sách: Các phép toán trên danh sách

1. Khởi tạo danh sách
2. Kiểm tra danh sách rỗng hoặc đầy
3. Lấy thông tin phần tử trong danh sách tại vị trí $p$
4. Thay đổi thông tin của phần tử tại vị trí $p$
5. Loại một phần tử tại vị trí $p$
6. Chèn một phần tử ở vị trí $p$
7. Duyệt danh sách

# Khởi tạo danh sách

**Số lượng phần tử của danh sách được gán bằng 0 (count=0)**

**Mã nguồn:**

```
void Initialization(List &L){
        L.count=0;
}
void Initialization(List *L){
        L→count=0;
}
```

# Kiểm tra danh sách rỗng hoặc đầy

**I. Kiểm tra danh sách rỗng:**
**Input:** Một danh sách $L$
**Output:** Nhận 02 giá trị: true– Danh sách rỗng; false– danh sách không rỗng.
**Mã nguồn:**

```
bool isEmpty(List L){
        if(L.count==0) return true;
        else return false;
}
```

# Kiểm tra danh sách rỗng hoặc đầy

**II. Kiểm tra danh sách đầy:**

**Input:** Một danh sách $L$

**Output:** Nhận 02 giá trị: true– Danh sách đầy; false– danh sách không đầy.

**Mã nguồn:**

```
bool isFull(List L){
        if(L.count==max) return true;
        else return false;
}
```

# Danh sách: Các phép toán trên danh sách

1. Khởi tạo danh sách
2. Kiểm tra danh sách rỗng hoặc đầy
3. Lấy thông tin phần tử trong danh sách tại vị trí $p$
4. Thay đổi thông tin của phần tử tại vị trí $p$
5. Loại một phần tử tại vị trí $p$
6. Chèn một phần tử ở vị trí $p$
7. Duyệt danh sách

# Lấy thông tin phần tử ở vị trí *p*

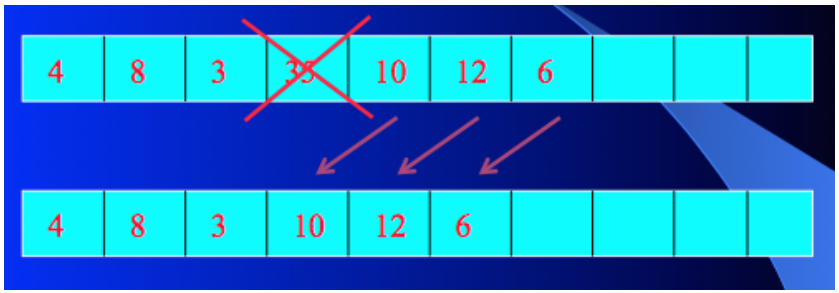
**Input:** Một danh sách *L*, ví trí *p* của phần tử cần lấy thông tin
**Output:** trả về phần tử tại vị trí *p* nếu $0 < p \leq count$; trả về giá trị -1 nếu $p \leq 0$ hoặc $p > count$
**Mã nguồn:**

```
bool Get_Element(List L, item &x, int p){
        if((isEmpty(L)==true)) return false;
        if((p<=0)||(p>L.count)) return false;
        x=L.element[p]; return true;
}
```

*tượng tự cho phép toán thay đổi thông tin tin của phần từ x tại vị trí p*

```
bool ChangeElement(List L, item &x, int p);
```

# Loại phần tử ở vị trí $p$



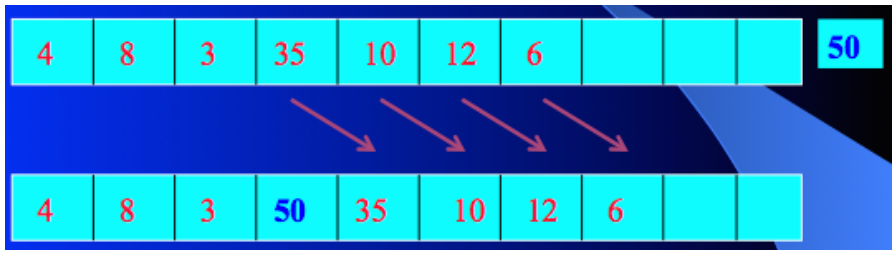

**Input:** Một danh sách $L$, ví trí $p$ của phần tử cần loại bỏ
**Output:** Một danh sách $L$ sau khi loại bỏ phần tử tại vị trí $p$
**Phương pháp:**
**Bước 1:** Kiểm tra giá trị $p$ có thuộc $[1, \ldots, count]$ hay không, nếu có thực hiện **Bước 2**.
**Bước 2:** Các phần tử được lưu tại vị trí $p+1$ đến vị trí $count$ trong danh sách được lưu lại tại các vị trí $p$ đến vị trí $count-1$. Giảm số lượng phần tử của danh sách xuống 1 đơn vị.

# Loại phần tử ở vị trí $p$

**Input:** Một danh sách $L$, ví trí $p$ của phần tử cần loại bỏ
**Output:** Một danh sách $L$ sau khi loại bỏ phần tử tại vị trí $p$
**Mã nguồn:**

```
bool Deletion(List &L, int p){
        if((isEmpty(L)==true)) return false;
        if((p<=0)||(p>L.count)) return false;
        for(int i=p; i<L.count; i++)
                L.element[i]=L.element[i+1];
        L.count=L.count-1; return true;
}
```

# Chèn một phần tử vào vị trí $p$




**Input:** Một danh sách $L$, một phần tử $x$, và ví trí $p$ trong danh sách để chèn $x$

**Output:** Một danh sách $L$ sau khi chèn $x$ tại vị trí $p$

# Chèn một phần tử vào vị trí $p$

**Input:** Một danh sách $L$, một phần tử $x$, và ví trí $p$ trong danh sách để chèn $x$
**Output:** Một danh sách $L$ sau khi chèn $x$ tại vị trí $p$
**Phương pháp:**
**Bước 1:** Kiểm tra giá trị $p$ có thuộc $[1, \ldots, count]$ hay không, nếu có thực hiện **Bước 2**.
**Bước 2:** Các phần tử được lưu tại vị trí $p$ đến vị trí $count$ trong danh sách được lưu lại tại các vị trí $p+1$ đến vị trí $count+1$.
**Bước 3:** Gán thông tin của x cho phần tử tại vị trí $p$
**Bước 4:** Tăng số lượng phần tử của danh sách lên 1 đơn vị.

# Danh sách: Các phép toán trên danh sách

1. Khởi tạo danh sách
2. Kiểm tra danh sách rỗng hoặc đầy
3. Lấy thông tin phần tử trong danh sách tại vị trí $p$
4. Thay đổi thông tin của phần tử tại vị trí $p$
5. Loại một phần tử tại vị trí $p$
6. Chèn một phần tử ở vị trí $p$
7. Duyệt danh sách

# Độ phức tạp của các phép toán trên danh sách

Table: Độ phức tạp của các phép toán

| **Phép toán** | **Độ phức tạp** |
|---|---|
| Khởi tạo danh sách | $\mathcal{O}(1)$ |
| Kiểm tra danh sách rỗng hoặc đầy | $\mathcal{O}(1)$ |
| Lấy thông tin phần tử ở vị trí $p$ | $\mathcal{O}(1)$ |
| Chèn phần tử vào vị trí $p$ | $\mathcal{O}(n)$ |
| Loại phần tử ở vị trí $p$ | $\mathcal{O}(n)$ |
| Duyệt danh sách | $\mathcal{O}(n)$ |

# Danh sách liên kết

1. Định nghĩa danh sách liên kết
2. Các phép toán trên danh sách liên kết
3. Cài đặt danh sách liên kết và các phép toán

# Danh sách liên kết

## Definition

Danh sách liên kết: Là một cấu trúc dữ liệu danh sách (list) chứa các phần tử (node) có 02 thành phần chính: Phần chứa dữ liệu (data), phần chứa địa chỉ của phần tử tiếp theo (next)

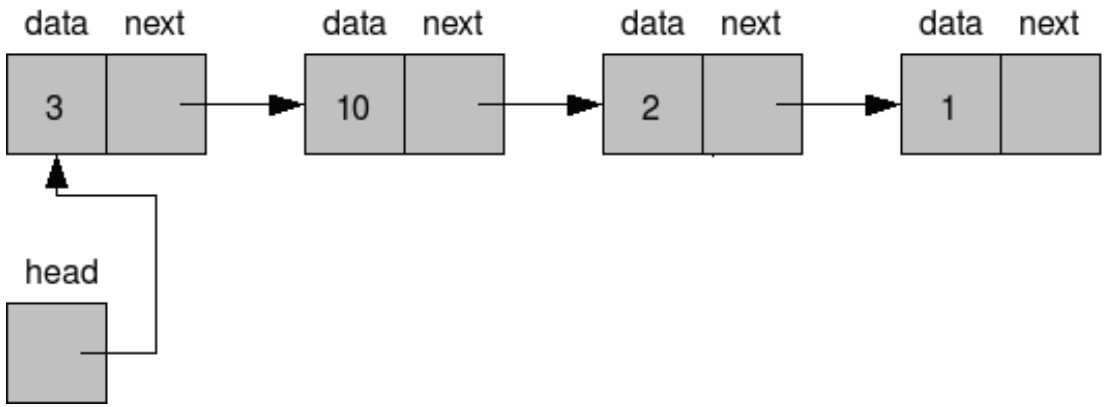



Figure: Danh sách liên kết

# Cài đặt danh sách liên kết

```
struct LinkedList{
        item data;//thành phần lưu dữ liệu
        LinkedList *next;//Lưu địa chỉ của phần tử tiếp theo
} head;
```

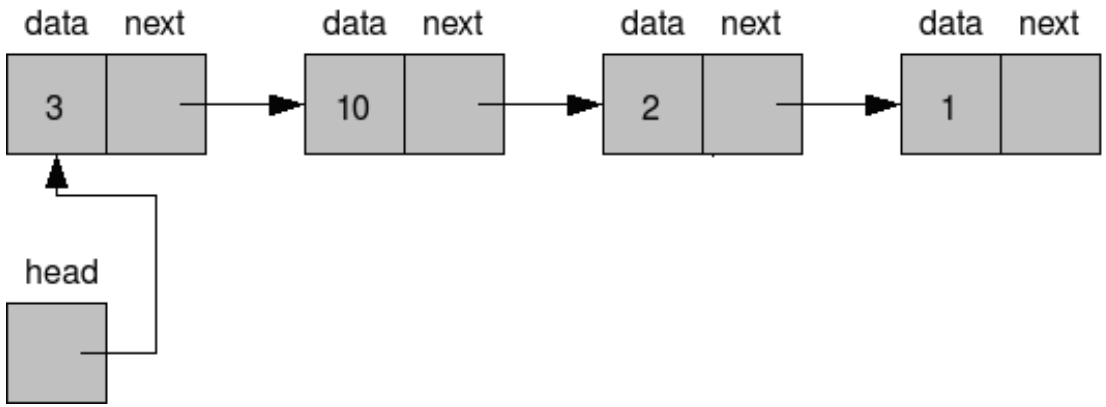



Figure: Danh sách liên kết

# Danh sách liên kết: Các phép toán

1. Khởi tạo danh sách
2. Kiểm tra danh sách rỗng
3. Lấy thông tin phần tử trong danh sách tại vị trí $p$
4. Thay đổi thông tin của phần tử tại vị trí $p$
5. Loại một phần tử tại vị trí $p$
6. Chèn một phần tử ở vị trí $p$
7. Duyệt danh sách

# Khởi tạo danh sách

Danh sách sau khi khởi tạo là danh sách rỗng: không tồn tại phần tử trong danh sách. Do đó, con trỏ head quản lý danh sách được gán bằng giá trị NULL.

**Mã nguồn:**

```
void Initialization(List *head ){
        head=NULL;
}
```

# Kiểm tra danh sách rỗng

**Input:** Một danh sách $L$
**Output:** Nhận 02 giá trị: true– Danh sách rỗng; false– danh sách không rỗng.
**Mã nguồn:**

```
bool isEmpty(List *head){
        if(head==NULL) return true;
        else return false;
}
```

# Danh sách liên kết: Các phép toán trên danh sách

1. Khởi tạo danh sách
2. Kiểm tra danh sách rỗng hoặc đầy
3. Lấy thông tin phần tử trong danh sách tại vị trí $k$
4. Loại một phần tử tại vị trí $k$
5. Chèn một phần tử ở vị trí $k$
6. Duyệt danh sách

# Lấy thông tin phần tử ở vị trí $k$

**Input:** Một danh sách liên kết $L$ (được quản lý bởi head), ví trí $k$ của phần tử cần lấy thông tin
**Output:** trả về phần tử tại vị trí $k$ ; trả về giá trị -1 nếu không tồn tại

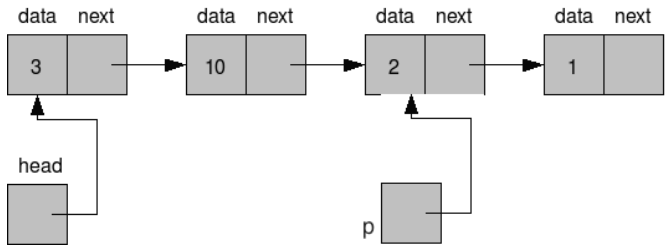



Figure: Lấy thông tin tại vị trí được quản lý bởi con trỏ $p$

# Lấy thông tin phần tử ở vị trí *k*

**Mã nguồn:**

```
bool Get_Element(List *head, item &x, int k){
List *P=head;
        if((isEmpty(head)==true)) return false;
        for(int i=1; i<k; i++){
                P=P→next;
                if(P==NULL)return false;
        }
        x= P→data;
        return true;
}
```

# Loại phần tử ở vị trí $k$

**Input:** Một danh sách $L$(được quản lý bởi con trỏ head), ví trí $k$ của phần tử cần loại bỏ
**Output:** Một danh sách $L$ sau khi loại bỏ phần tử tại vị trí $k$

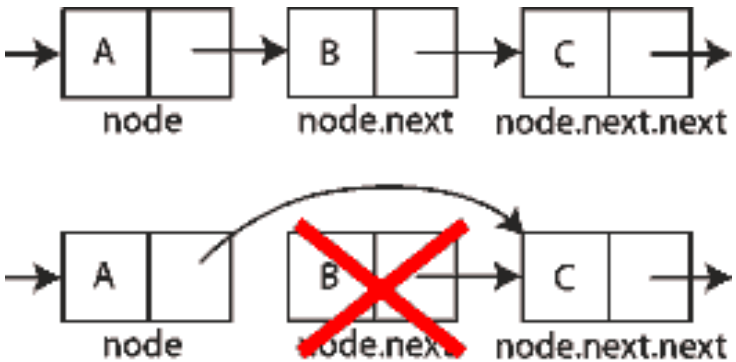



Figure: Loại bỏ một phần tử trong danh sách liên kết

# Loại phần tử ở vị trí $k$

**Mã nguồn:**

```
bool Remove(List *head, int k){
        List *P=head;
        List *preP;
        if((isEmpty(head)==true)) return false;
        if(k==1) head=head→next; return true;
         for(int i=1; i<k; i++){
                preP=p
                p=p→next;
                if(p==NULL)return false;
         }
         preP→next=p→next;
         return true;
}
```

# Chèn một phần tử vào vị trí $k$

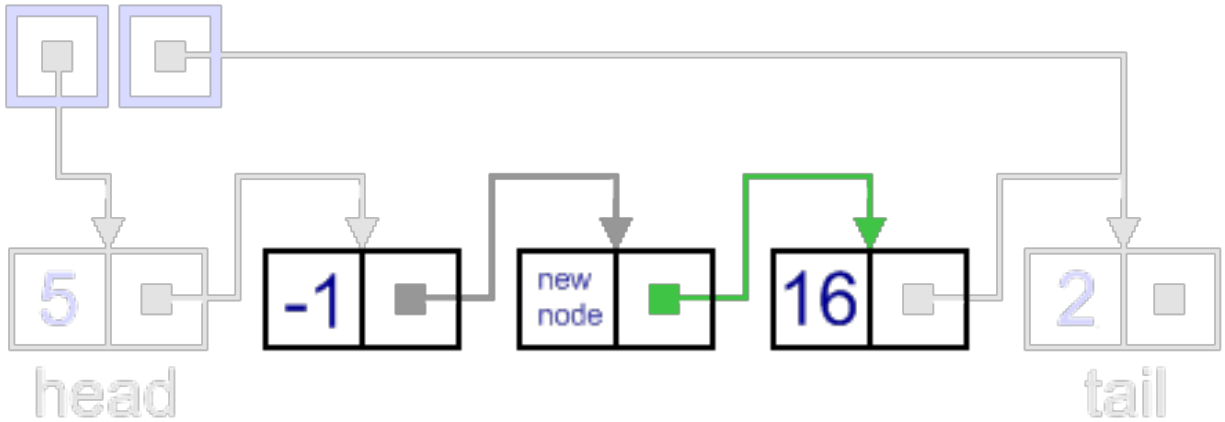



Figure: Chèn một phần tử vào danh sách liên kết

**Input:** Một danh sách $L$ (được quản lý bởi con trỏ head), một phần tử cần chèn $x$ (được quản lý bởi con trỏ $q$), và ví trí $k$ trong danh sách để chèn $x$
**Output:** Một danh sách $L$ sau khi chèn $x$ tại vị trí $k$

# Chèn một phần tử ở vị trí $k$

**Mã nguồn:**

```
bool Insertion(List *head, int k){
        List *P=head;
        List *preP;
        if((isEmpty(head)==true)) return false;
        if(k==1) head=head→next; return false;
         for(int i=1; i<k; i++){
                preP=p
                p=p→next;
                if(p==NULL) return false;
         }
         p→next=p→next;
         return true;
}
```

# Danh sách liên kết: Các vấn đề còn lại

- Duyệt danh sách ?
- Đánh giá độ phức tạp của các phép toán trên danh sách liên kết

# Thanks for your attention!