

Đệ quy

GV. Nguyễn Minh Huy

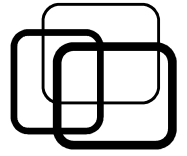


- Tổng quan về đệ quy.
- Phân loại đệ quy.
- Các vấn đề đệ quy thông dụng.



- **Tổng quan về đệ quy.**
- Phân loại đệ quy.
- Các vấn đề đệ quy thông dụng.

Tổng quan về đệ quy



■ Khái niệm đệ quy:

■ Đệ quy là gì?

➔ Xem phần..."Đệ quy là gì?"!!

■ Đệ quy là...

➔ Định nghĩa một vấn đề bằng chính vấn đề đó!!

■ Một vài định nghĩa đệ quy:

➤ $n! = n * (n - 1)!$.

➤ $f(n) = f(n - 1) + f(n - 2)$.

➤ n là số tự nhiên nếu $n - 1$ cũng là số tự nhiên N .

➤ Tổ tiên của A là những người sinh ra...tổ tiên của A .

Tổng quan về đệ quy



■ Khái niệm đệ quy:

■ Cấu trúc một định nghĩa đệ quy:

- Phần dừng: trường hợp cơ bản.
- Phần đệ quy: suy biến vấn đề về trường hợp đơn giản hơn.
 - $0! = 1$
 - $n! = n * (n - 1)!$.

 - $f(0) = 0$
 - $f(1) = 1$
 - $f(n) = f(n - 1) + f(n - 2)$.

 - 0 là số tự nhiên nhỏ nhất.
 - n là số tự nhiên nếu $n - 1$ là số tự nhiên.

 - Tổ tiên gần nhất của A là ba mẹ A.
 - Tổ tiên của ba mẹ A cũng là A tổ tiên của A.



■ Đệ quy trong lập trình:

■ Hàm đệ quy:

- Thân hàm có lời gọi hàm lại chính nó.
- Lời gọi hàm trực tiếp hay gián tiếp.

// Đệ quy trực tiếp.

```
void func()
{
    // ...
    func();
    // ...
}
```

// Đệ quy gián tiếp.

```
void func1()
{
    // ...
    func2();
    // ...
}
void func2()
{
    // ...
    func1();
    // ...
}
```

Tổng quan về đệ quy



■ Đệ quy trong lập trình:

■ Cấu trúc hàm đệ quy:

```
<Kiểu trả về> <Tên hàm>( [Danh sách tham số] )  
{  
    if (<Trường hợp cơ bản>)  
        // Xử lý trường hợp cơ bản.  
    else  
        // Gọi lại hàm đệ quy.  
}
```

```
int tinghGT(int n)  
{  
    if (n == 0)  
        return 1;  
    return n * tinghGT(n - 1);  
}
```

```
int fibo(int n)  
{  
    if (n == 0)  
        return 0;  
    if (n == 1)  
        return 1;  
    return fibo(n - 1) + fibo(n - 2);  
}
```

Tổng quan về đệ quy



■ Đệ quy trong lập trình:

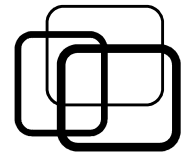
■ Kiểu dữ liệu đệ quy:

```
struct <Tên cấu trúc>
{
    // ...
    <Tên cấu trúc> <Tên thành phần>;
    // ...
};
```

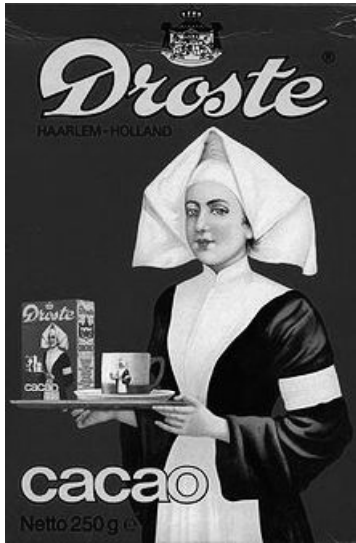
```
struct Nguoi
{
    char    *hoten;
    int     tuoi;
    Nguoi *cha;
    Nguoi *me;
};
```

```
struct NhanVien
{
    char    *hoten;
    char    *diachi;
    double  luong;
    NhanVien *nguoiquanly;
};
```


Tổng quan về đệ quy



■ Đệ quy trong cuộc sống:



Quảng cáo

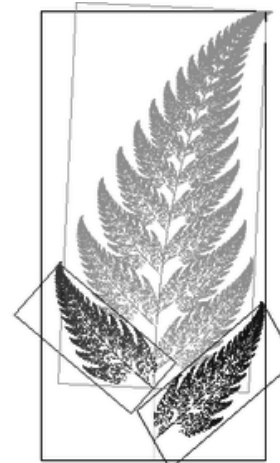


Búp bê Nga

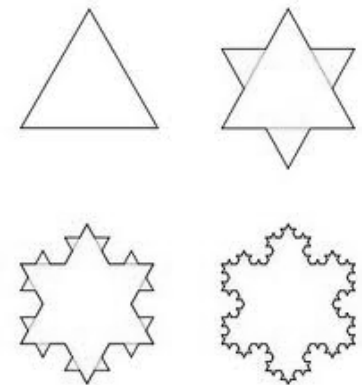


Thực phẩm

Thực vật



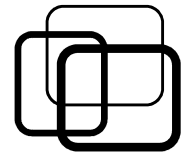
Bông tuyết





- Tổng quan về đệ quy.
- **Phân loại đệ quy.**
- Các vấn đề đệ quy thông dụng.

Phân loại đệ quy



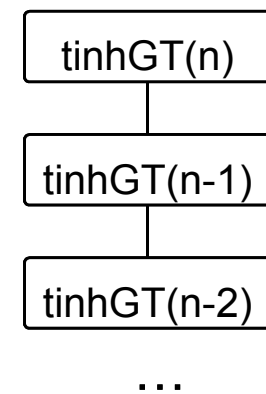
■ Các loại đệ quy:

■ Đệ quy tuyến tính:

- Thân hàm có duy nhất một lời gọi đệ quy.

```
int tinghGT(int n)
{
    if (n == 0)
        return 1;
    return n * tinghGT(n - 1);
}
```

- Độ phức tạp: tuyến tính $O(n)$.



■ Đệ quy đuôi:

- Một dạng của đệ quy tuyến tính.
- Lời gọi đệ quy là duy nhất và là lệnh cuối cùng.
- ➔ Ít tốn bộ nhớ (không lưu các trường hợp trước đó).

Phân loại đệ quy



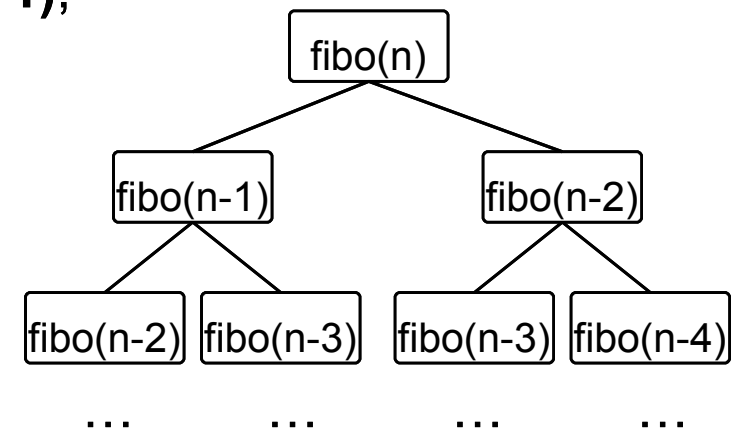
■ Các loại đệ quy:

■ Đệ quy nhị phân:

- Thân hàm có 2 lời gọi đệ quy.

```
int fibonacci(int n)
{
    if (n == 0)
        return 0;
    if (n == 1)
        return 1;
    return fibonacci(n - 1) + fibonacci(n - 2);
}
```

- Độ phức tạp: $O(2^n)$.





■ Các loại đệ quy:

■ Đệ quy hỗ tương:

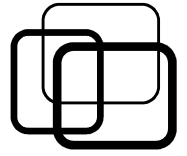
- Hàm f1 có lời gọi đến hàm f2.
- Hàm f2 có lời gọi đến hàm f1.

```
bool soChan(int n)
{
    if (n == 0)
        return true;
    return soLe(n - 1);
}
```

```
bool soLe(int n)
{
    if (n == 0)
        return false;
    return soChan(n - 1);
}
```

- Độ phức tạp: tùy thuộc độ phức tạp từng hàm đệ quy.

Phân loại đệ quy



■ Các loại đệ quy:

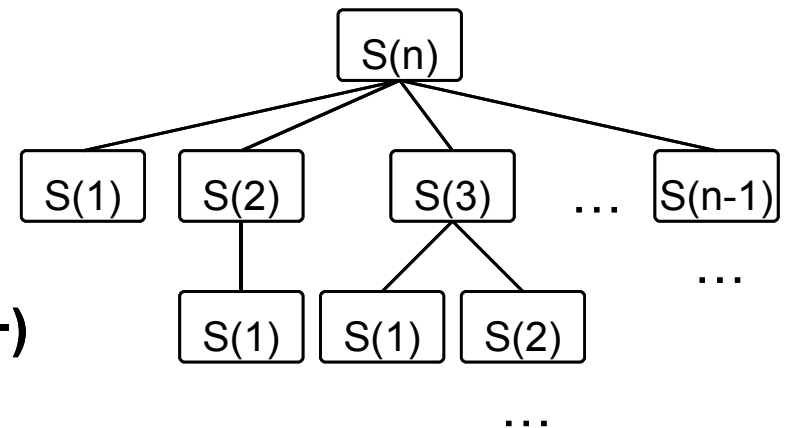
■ Đệ quy phi tuyến:

- Thân hàm có lời gọi đệ quy đặt trong vòng lặp.

Tính: $S(1) = 1$

$$S(n) = S(1) + S(2) + \dots + S(n-1).$$

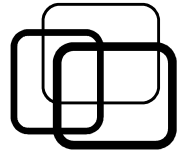
```
int tinh(int n)
{
    if (n == 1)
        return 1;
    int S = 0;
    for (int i = 1; i <= n - 1; i++)
        S += tinh(i);
    return S;
}
```



- Độ phức tạp: $O(n!)$.



- Tổng quan về đệ quy.
- Phân loại đệ quy.
- **Các vấn đề đệ quy thông dụng.**



■ Giải bài toán bằng đệ quy:

■ Cách giải trực tiếp:

- Tìm từng bước đi đến kết quả.
- Lời giải không đệ quy.

■ Cách giải đệ quy:

- Lời giải gián tiếp.
- Định nghĩa lại bài toán bằng đệ quy.
- Bước 1: định nghĩa trường hợp cơ bản.
- Bước 2: suy biến bài toán về trường hợp đơn giản hơn.
 - Công thức truy hồi (suy biến bằng công thức).
 - Chia để trị (suy biến bằng chia đôi).
 - Lăn ngược (tìm tất cả lời giải).

Các vấn đề đệ quy thông dụng



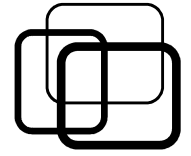
■ Công thức truy hồi:

- Tính phần tử A_n của dãy số $\{ A \}$:

- Công thức truy hồi:

- Tính trực tiếp A_0 .
- Tìm công liên hệ A_n với A_{n-1} .

Các vấn đề đệ quy thông dụng



■ Công thức truy hồi:

■ Ví dụ 1:

- Vi khuẩn cứ 1 phút nhân đôi.
- Ban đầu có 1 vi khuẩn.
- Sau 20 phút bao nhiêu con?

$$V(0) = 1$$

$$V(n) = 2 * V(n - 1).$$

```
int tinghVK(int phut)
{
    if (phut == 0)
        return 1;
    return 2 * tinghVK(phut - 1);
}
```

■ Ví dụ 2:

- Lãi suất tiết kiệm 7%/năm.
- Ban đầu gửi 1 triệu.
- Sau 20 năm bao nhiêu tiền?

$$T(0) = 1$$

$$\begin{aligned} T(n) &= T(n - 1) + 0.07 * T(n - 1) \\ &= 1.07 T(n - 1). \end{aligned}$$

```
int tinghTK(int nam)
{
    if (nam == 0)
        return 1;
    return 1.07 * tinghTK(nam - 1);
}
```

Các vấn đề đệ quy thông dụng



■ Kỹ thuật chia để trị (Divide & Conquer):

■ Làm sao để ăn hết một con bò?

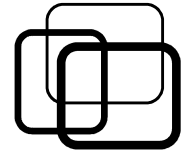
→ Chia thành từng phần nhỏ.

→ Thế nào là đủ nhỏ?

■ Giải toán bằng kỹ thuật chia để trị:

```
Trị ( P )  
{  
    if (P đủ nhỏ)  
        Xử lý trực tiếp P;  
    else  
        Chia P → P1, P2;  
        Trị ( P1 );  
        Trị ( P2 );  
        Tổng hợp kết quả;  
}
```

Các vấn đề đệ quy thông dụng



■ Kỹ thuật chia để trị (Divide & Conquer):

■ Ví dụ:

- Cho mảng nguyên 1 chiều.
- Đếm số âm trong mảng.

// Chia đôi xứng...

```
int demAm( int *a, int l, int r )  
{  
    if ( l == r )  
        return a[ r ] < 0 ? 1 : 0;  
  
    int mid = ( l + r ) / 2;  
    int dem1 = demAm( a, l, mid );  
    int dem2 = demAm( a, mid + 1, r );  
  
    return dem1 + dem2;  
}
```

// Chia bất đối xứng...

```
int demAm( int *a, int n )  
{  
    int dem1 = a[ n - 1 ] < 0 ? 1 : 0;  
  
    if ( n == 1 )  
        return dem1;  
  
    int dem2 = demAm( a, n - 1 );  
  
    return dem1 + dem2;  
}
```



- Kỹ thuật lần ngược (Back-tracking):
 - Còn gọi là kỹ thuật “thử và sai”.
 - Áp dụng tìm nhiều lời giải khác nhau.
 - Phù hợp khi có thể lần từng bước đến lời giải.

```
Thử ( S )
{
    if (S là lời giải)
        Ghi nhận S;
    else
        while bước tiếp được
        {
            Bước tiếp S;
            Thử ( S );
            Quay lui S;
        }
}
```



■ Kỹ thuật lần ngược (Back-tracking):

■ Ví dụ:

- Cho mảng nguyên dương 1 chiều.
- Tìm tất cả các bộ phần tử có tổng bằng K.

```
void tim( int *a, int n, int K, int start, int T, bool *flag ) {  
    if ( T == K )  
        xuat( a, n, flag );  
    else  
        for ( int i = start; i < n; i++ )  
        {  
            T += a[ i ]; flag[ i ] = true;  
            tim( a, n, K, i + 1, T, flag );  
            T -= a[ i ]; flag[ i ] = false;  
        }  
}
```



■ Tổng quan về đệ quy:

- Vấn đề được định nghĩa bằng chính nó.
- Cấu trúc:
 - Phần dừng: trường hợp cơ bản.
 - Phần đệ quy: vấn đề suy biến về trường hợp đơn giản hơn.
- Hàm đệ quy: hàm gọi lại chính nó.
- Kiểu dữ liệu đệ quy: kiểu dữ liệu có thành phần con là chính nó.





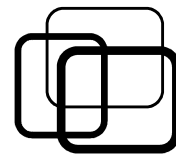
■ Phân loại đệ quy:

- Đệ quy tuyến tính: 1 lời gọi đệ quy.
- Đệ quy nhị phân: 2 lời gọi đệ quy.
- Đệ quy hồi tưởng: 2 hàm đệ quy gọi lẫn nhau.
- Đệ quy phi tuyến: lời gọi đệ quy trong vòng lặp.

■ Các vấn đề đệ quy thông dụng:

- Công thức truy hồi.
- Kỹ thuật chia để trị.
- Kỹ thuật lán ngược.





■ Bài tập 6.1: (công thức truy hồi)

Tìm công thức truy hồi và viết hàm đệ quy để:

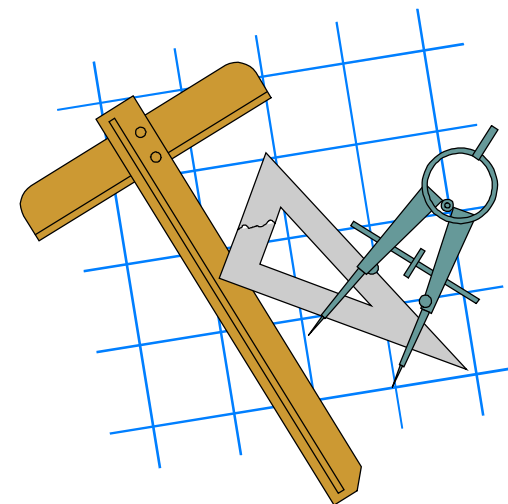
a) Tính $A = 1 + 2 + \dots + n$.

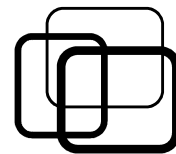
b) Tính $B = x * x * \dots * x$ (n lần).

c) Tính $C = 1 - 1/2 + 1/3 - \dots (+/-) 1/n$.

d) In ra màn hình tam giác Pascal có chiều cao N.

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
...
```

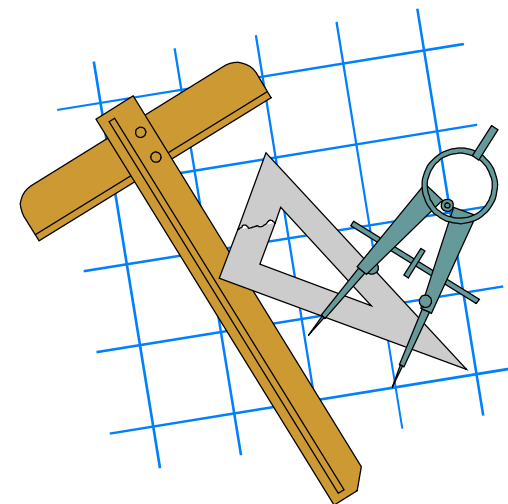


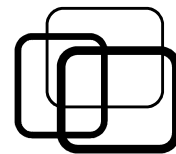


■ Bài tập 6.2: (chia để trị)

Viết hàm đệ quy thực hiện các việc sau:

- a) Tìm phần tử lớn nhất trong mảng nguyên 1 chiều.
- b) Tìm phần tử x trong mảng nguyên 1 chiều.
- c) Trả về chuỗi đảo của 1 chuỗi.
- d) In ra màn hình định dạng tiền tệ của 1 số nguyên dương.

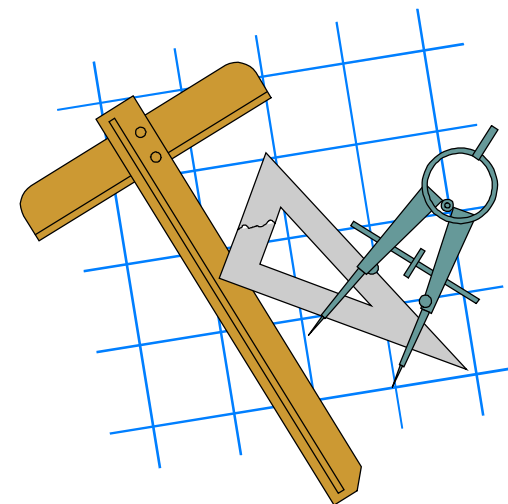


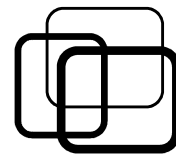


■ Bài tập 6.3: (lần ngược)

Viết hàm đệ quy thực hiện các thao tác trên chuỗi (ký tự phân biệt):

- a) In ra tất cả các hoán vị của một chuỗi.
- b) In ra tất cả các chỉnh hợp chập K của một chuỗi.
- c) In ra tất cả các tổ hợp chập K của một chuỗi.

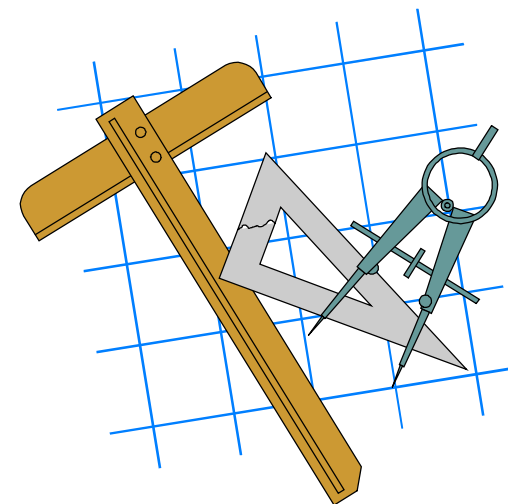


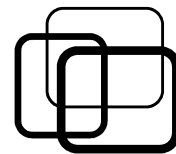


■ Bài tập 6.4:

Viết hàm đệ quy thực hiện các thao tác trên kiểu cấu trúc **Nguoi**:

- a) Đếm số tổ tiên của một người.
- b) Đếm số tổ tiên bên ngoại của một người.





■ Bài tập 6.5: (*)

Thuật toán Mid-Point vẽ đoạn thẳng AB:

- Nếu A, B liền kề nhau: vẽ điểm A và B.
- Nếu A, B xa nhau:
 - + Gọi M là trung điểm AB.
 - + Vẽ đoạn thẳng AM và MB.

Với hàm vẽ điểm SetPixel do giáo viên cung cấp, viết hàm đệ quy vẽ đoạn thẳng AB bằng thuật toán Mid-Point.

