

# MỤC TIÊU

**Hiểu được khái niệm và ý nghĩa của đề qui**

1

**Hiểu và biết cách xây dựng hàm đề qui**

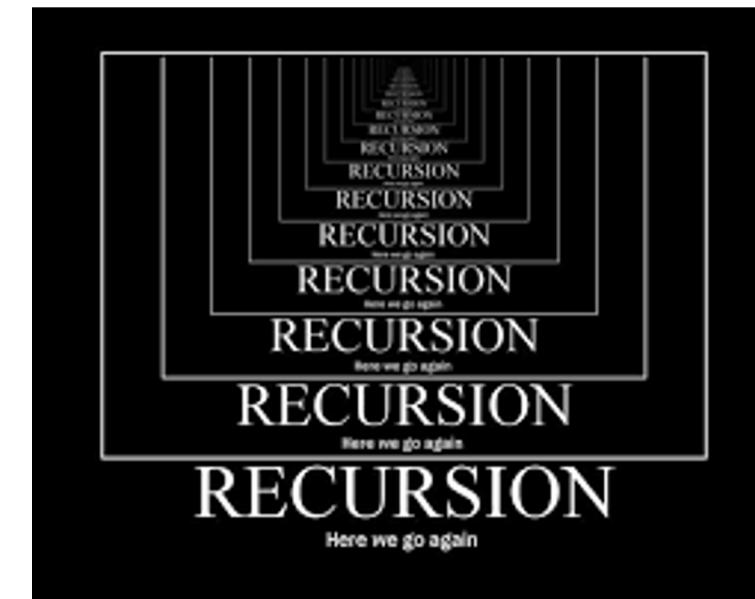
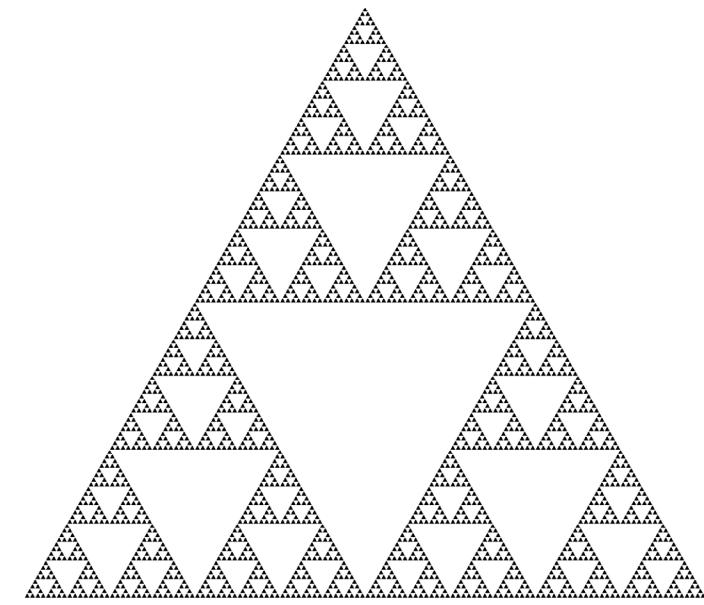
2

- **Giới thiệu, định nghĩa đệ quy**
- **Hàm đệ quy**
- **Một số loại đệ quy**
- **So sánh đệ quy với lặp**



## CHƯƠNG 2: ĐỆ QUY

# GIỚI THIỆU, ĐỊNH NGHĨA ĐỆ QUY



*Hình ảnh minh họa ví dụ đệ quy*

*Nguồn: hình ảnh internet*

# GIỚI THIỆU, ĐỊNH NGHĨA ĐỆ QUY

- ② Thuật giải đệ quy (recursive algorithm) được cài đặt gọi là **HÀM ĐỆ QUY** (recursive function)
- ③ **Định nghĩa đệ quy:** là một vấn đề mà trong định nghĩa của nó có sử dụng chính khái niệm đó.
- ④ **Định nghĩa đệ quy:** là các đối tượng được định nghĩa dưới dạng qui nạp từ những khái niệm đơn giản nhất cùng dạng với nó.

# GIỚI THIỆU, ĐỊNH NGHĨA ĐỆ QUY

Thuật giải đệ quy (**recursive algorithm**): là kỹ thuật dùng để tinh giảm vấn đề hiện tại.

- Có 1 hoặc nhiều trường hợp cơ sở (**basic case**): giải trực tiếp
- Trường hợp tổng quát (**general case**): tinh giảm dần để quay về basic case.
- Lưu ý: trường hợp cơ sở sẽ đóng vai trò là điểm dừng của các giải thuật đệ quy

# GIỚI THIỆU, ĐỊNH NGHĨA ĐỀ QUY

- ?] Ví dụ: **định nghĩa** giải thừa của 1 số nguyên không âm?
- ?] **Định nghĩa** đệ quy:

$$n! = \begin{cases} 1 & \text{nếu } n = 0 \\ n \times (n - 1)! & \text{nếu } n > 0 \end{cases}$$

- ?] Trong đó:
  - $n! = 1$  nếu  $n = 0$  gọi là trường hợp cơ sở (basic case)
  - $n! = n \times (n - 1)!$  Nếu  $n > 0$  gọi là trường hợp tổng quát (general case)

# GIỚI THIỆU, ĐỊNH NGHĨA ĐỀ QUY

$$n! = \begin{cases} 1 & \text{nếu } n = 0 \\ n \times (n - 1)! & \text{nếu } n > 0 \end{cases}$$

- **Tính  $2! = ???$** 
  - $2!: 2 > 0$ : general case nên kết quả =  $2 \times (2 - 1)! = 2 \times 1!$
  - $1!: 1 > 0$ : general case nên kết quả =  $1 \times (1 - 1)! = 1 \times 0!$
  - $0!: 0 = 0$ : basic case nên kết quả = 1
  - Khi đó kết quả tính toán cuối cùng =  $2 \times 1 \times 1 = 2$

# GIỚI THIỆU, ĐỊNH NGHĨA ĐỆ QUY

- ② Hàm đệ quy (recursive function) là hàm gọi chính bản thân nó, được định nghĩa và khai báo như các hàm khác.
- ② Thân hàm đệ quy bao gồm:
  - Phần giải quyết trường hợp cơ sở (basic case)
  - Phần giải quyết trường hợp tổng quát (general case)

- **Giới thiệu, định nghĩa đệ quy**
- **Hàm đệ quy**
- **Một số loại đệ quy**
- **So sánh đệ quy với lặp**



## CHƯƠNG 2: ĐỆ QUY

# HÀM ĐỆ QUY

Thân hàm đệ quy có dạng tổng quát như sau:

**if** (biểu thức điều kiện đúng trong trường hợp cơ sở)

Biểu thức tính toán hay trả về kết quả (không gọi đệ quy chính nó)

**else {**

Chia thành vấn đề con đồng dạng

Các vấn đề con gọi chính bản thân hàm đệ qui đang xây dựng

Tính toán, tổng hợp và trả về kết quả cuối

}



# HÀM ĐỆ QUY

✉ **Ví dụ:** Viết hàm đệ quy tính giai thừa của một số nguyên không âm

**Hàm đệ quy: KHÔNG sử dụng vòng lặp khi tính**

```
int giaithua( int n)
{
    //assert ( n >= 0);
    if ( n == 0)
        return 1;
    else
        return n * giaithua( n - 1);
}
```

Basic case

General case

# HÀM ĐỆ QUY

?) **Ví dụ:** Viết hàm đệ quy tính giai thừa của một số nguyên không âm

**Hàm đệ quy: KHÔNG sử dụng vòng lặp khi tính**

```
int gaiithua( int n )
{
    //assert ( n >= 0);
    if ( n == 0)
        return 1;
    else
        return n * gaiithua( n - 1);
}
```

Tính 3!

• gaiithua(3):

return  $3 * \text{gaiithua}(2)$

$2 * \text{gaiithua}(1)$

$1 * \text{gaiithua}(0)$

1

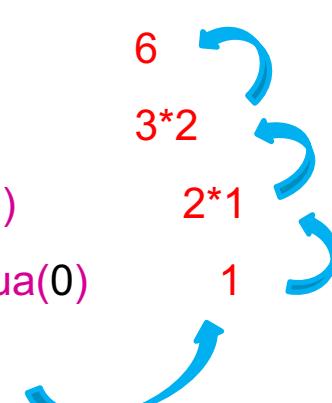
Trả kết quả

6

$3 * 2$

$2 * 1$

1



# HÀM ĐỆ QUY

?) **Ví dụ:** Viết hàm đệ quy tính tổng các số từ 1 đến n (n là số nguyên dương)

```
int tinhTong (int n)
{
    if (n == 1)
        return 1;
    else
        return n + tinhTong(n - 1);
}
```

# HÀM ĐỆ QUI

- ☒ **Ví dụ:** Viết hàm đệ quy để tính dãy số Fibonacci với số nguyên  $n \geq 0$

$$f_n = \begin{cases} 0 & \text{nếu } n = 0 \\ 1 & \text{nếu } n = 1 \\ f_{n-1} + f_{n-2} & \text{nếu } n > 1 \end{cases}$$

- 
- ☒ Basic case của hàm trên là  $n = 0$  hoặc  $1$ , khi đó giá trị của hàm đệ quy sẽ là  $0$  hoặc  $1$ .
  - ☒ General case của hàm trên sẽ được tổng quát hoá là  $F_{n-1} + F_{n-2}$  khi  $n > 1$ .

- Giới thiệu, định nghĩa đệ quy
- Hàm đệ quy
- Một số loại đệ quy
- So sánh đệ quy với lặp



## CHƯƠNG 2: ĐỆ QUY

# MỘT SỐ LOẠI ĐỆ QUY

⇒ Đệ qui tuyến tính (linear recursion): trong thân hàm đệ qui chỉ 1 lần gọi lại chính bản thân nó.

**Ví dụ:** hàm đệ quy tính giai thừa của số nguyên không âm.

⇒ Đệ quy nhị phân (binary recursion) – đệ qui nhánh: trong thân hàm đệ qui có 2 lần gọi đến chính bản thân nó.

**Ví dụ:** hàm đệ quy tính Fibonacci thứ n.

# MỘT SỐ LOẠI ĐỆ QUY

- ② **Đệ quy đuôi (tail recursion):** là 1 dạng của đệ qui tuyến tính. Trong câu lệnh gọi đệ quy chỉ có gọi lại chính bản thân nó mà không có kết hợp với giá trị hay phép toán nào khác.
- ② **Đệ quy phi tuyến:** trong thân hàm đệ quy có vòng lặp gọi chính bản thân nó 1 số lần (nhiều lần). Thường xuất hiện trong hàm đệ quy tính theo công thức truy hồi.
- ② **Đệ quy hỗ tương:** các hàm đệ quy gọi lẫn nhau.

# MỘT SỐ LOẠI ĐỆ QUY

## ?) Lưu ý:

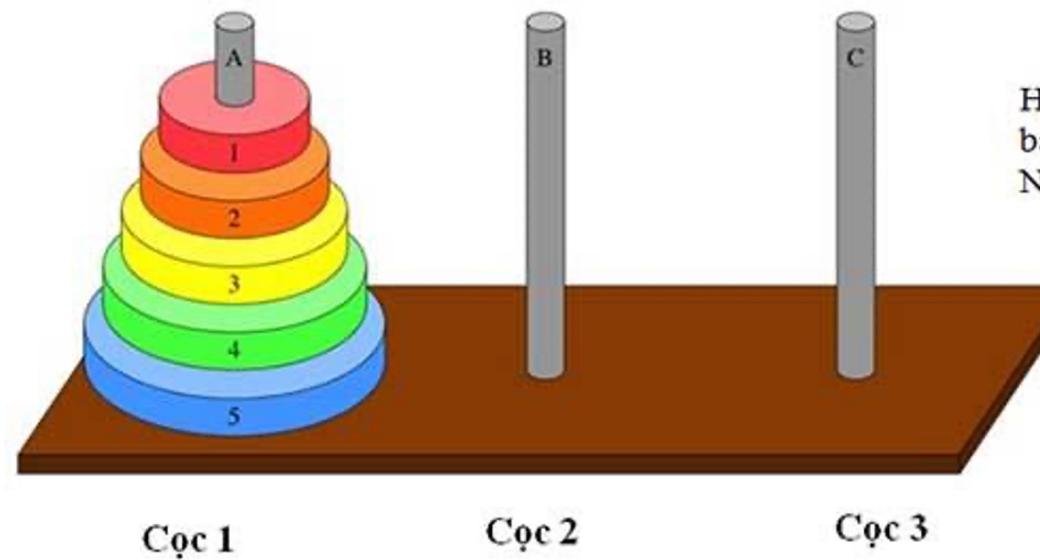
- Các hàm đệ quy khi thực thi sẽ thực hiện gọi lại chính nó
  - **Mỗi lần gọi sẽ tiêu tốn một vùng nhớ nhất định**
- Nếu **trường hợp cơ sở** không thể dừng được hàm đệ quy
  - **Hàm đệ quy** sẽ thực hiện **gọi vô tận**
    - Các **vùng nhớ** đều được **cấp hết**
      - **Gây ra tràn bộ nhớ** (buffer overflow)
- Nếu **trường hợp tổng quát** không để mô tả đầy đủ bài toán
  - **Kết quả** của bài toán sẽ **không tối ưu**
  - **Không đưa vấn đề** về trường hợp cơ sở được

# MỘT SỐ LOẠI ĐỆ QUY

?) **Bài toán Tháp Hà Nội:** Có 3 cột (giả sử A, B, C), trên cột có 1 số lượng đĩa nhất định (có kích thước khác nhau) được đặt theo quy luật đĩa nhỏ đặt trên đĩa lớn hơn. Tiến hành dời toàn bộ số đĩa ở cột A sang cột khác (giả sử cột C) sao cho số bước là nhỏ nhất theo yêu cầu sau:

- Di chuyển đĩa phải đặt vào 1 cột.
- Mỗi lần chỉ di chuyển 1 đĩa.
- Khi di chuyển phải đảm bảo đĩa nhỏ phải luôn nằm trên đĩa lớn.
- Có thể dùng cột B làm trung gian.

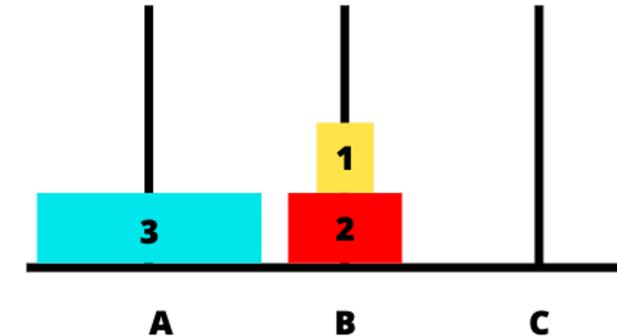
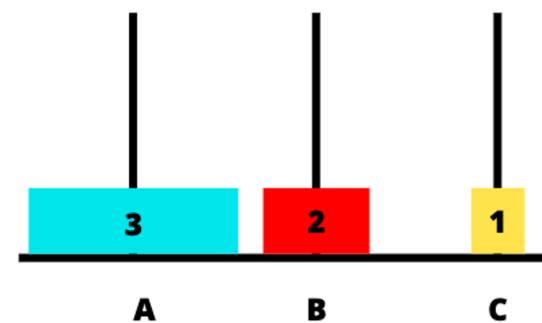
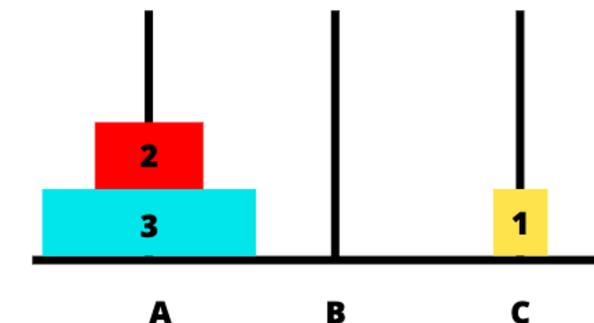
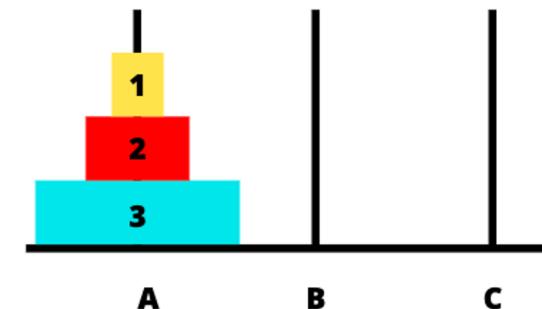
# MỘT SỐ LOẠI ĐỆ QUY



Hình ảnh mô tả bài toán Tháp Hà Nội.

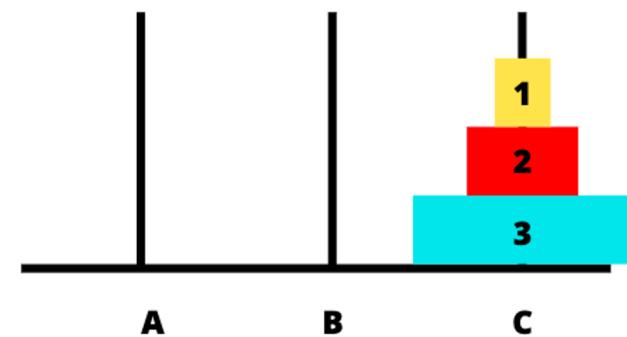
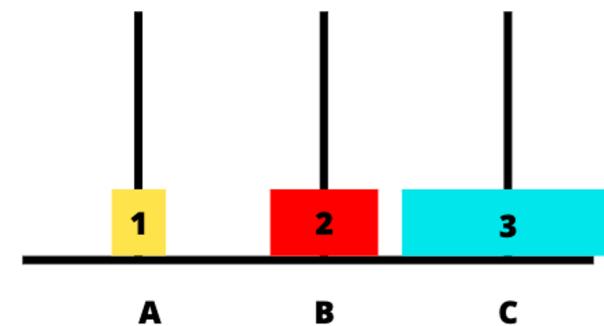
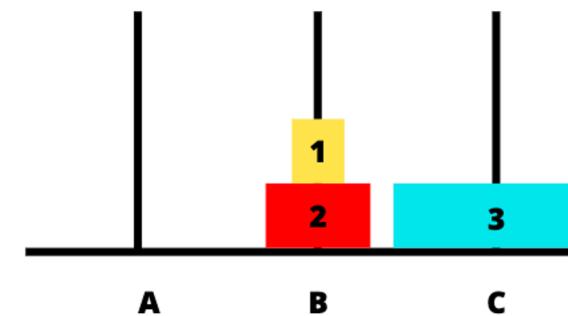
# MỘT SỐ LOẠI ĐỆ QUY

- Minh họa bài toán tháp Hà Nội



# MỘT SỐ LOẠI ĐỆ QUY

- Minh họa bài toán tháp Hà Nội



# MỘT SỐ LOẠI ĐỆ QUY

## ② Giải thuật đệ quy bài toán Tháp Hà Nội:

- Di chuyển (số đĩa – 1) từ cột A sang cột B (cột trung gian)
- Di chuyển đĩa cuối cùng từ cột A sang cột C (cột đích)
- Di chuyển (số đĩa – 1) từ cột B sang cột C dùng cột A làm trung gian.

```
void thapHaNoi(int n,char A,char B,char C)
{
    if(n==1){
        cout<<A<<" ==> "<<C<<"\n";// nếu n = 1 thì dịch chuyển từ A -> C
    }
    else {
        // Nếu n > 1 thì thực hiện lần lượt các bước
        thapHaNoi(n - 1, A, C, B); // 1. Dịch chuyển n-1 đĩa từ A -> B
        cout<<A<<" ==> "<<C<<"\n"; // 2. Dịch chuyển đĩa thứ n từ A -> C
        thapHaNoi(n - 1, B, A, C); // 3. Dịch chuyển n-1 đĩa từ B -> C
    }
}
```

- Giới thiệu, định nghĩa đệ quy
- Hàm đệ quy
- Một số loại đệ quy
- So sánh đệ quy với lặp



## CHƯƠNG 2: ĐỆ QUY

# SO SÁNH VÒNG LẶP VÀ ĐỆ QUY

	Đệ quy	Vòng lặp
Dùng cấu trúc điều khiển	Cấu trúc lựa chọn	Cấu trúc lặp
Lặp lại	Lặp lại qua lời gọi hàm	Dùng cấu trúc tường minh
Kiểm tra điều kiện dừng	Trường hợp cơ sở	Điều kiện lặp có giá trị sai
Liên tục tiếp cận điều kiện dừng	Phiên bản mới đơn giản	Cập nhật biến đếm
Lặp vô tận	Nếu bước đệ qui không giảm bớt vấn đề, không thể hội tụ về trường hợp cơ sở (basic case)	Nếu điều kiện lặp luôn thỏa



**THANK YOU**