Đệ quy (Recursion)

- Là một phương pháp lập trình cho phép một hàm có thể gọi lại chính nó trực tiếp hoặc gián tiếp.
- Ví dụ:

```
void Test()
{
        Test();
}
```

- Một chương trình đệ quy hoặc một định nghĩa đệ quy thì không thể gọi đến chính nó mãi mãi mà phải có một điểm dừng đến một trường hợp đặc biệt nào đó, mà ta gọi là trường hợp suy biến (degenerate case).
- Ví dụ: Ta định nghĩa n! như sau:

$$n! = \begin{cases} n * (n-1)! \\ 0! = 1 \end{cases}$$

Đệ quy (Recursion)

- **Phương pháp thiết kế một giải thuật đệ quy:**
  - Tham số hoá bài toán (Hiểu bài toán)
  - Phân tích trường hợp chung : đưa bài toán dưới dạng bài toán cùng loại nhưng có phạm vi giải quyết nhỏ hơn theo nghiã dần dần sẽ tiến đến trường hợp suy biến
  - Tìm trường hợp suy biến —> **Tìm các điều kiện biên (chặn), tìm giải thuật cho các tình huống này (Tìm điểm dừng)**

Đệ quy (Recursion)

- Chương trình đệ quy gồm hai phần chính:
  1. Phần cơ sở: Điều kiện thoát khỏi đệ quy (điểm dừng)
  2. Phần đệ quy: Trong phần thân chương trình có lời gọi đến chính bản thân chương trình với giá trị mới của tham số nhỏ hơn giá trị ban đầu

Đệ quy (Recursion)

- Nhận xét:
  - Thông thường thay vì sử dụng lời giải đệ quy cho một bài toán, ta có thể thay thế bằng lời giải không đệ quy (khử đệ quy) bằng phương pháp lặp.
  - Việc sử dụng giải thuật đệ quy có:

Ưu điểm	Khuyết điểm
• Thuận lợi cho việc biểu diễn bài toán • Gọn (đối với chương trình)	• Có khi không được tối ưu về thời gian • Có thể gây tốn bộ nhớ

  - Chính vì vậy, trong lập trình người ta cố tránh sử dụng thủ tục đệ quy nếu thấy không cần thiết.

Bài tập Đệ quy (Recursion)

Bài 1: Tính tổng S(n) = 1 + 2 + 3 + ... + (n-1) + n với n ≥ 1.

- Giả sử ta cần tính S(4).

S(4) = (1 + 2 + 3) + 4 = S(3) + 4

- Để tính được S(4), ta cần tính S(3)
- Giả sử ta cần tính S(3)

S(3) = (1 + 2) + 3 = S(2) + 3

- Để tính được S(3), ta cần tính S(2)
- Giả sử ta cần tính S(2)

S(2) = 1 + 2 = S(1) + 2

- Để tính được S(2), ta cần tính S(1)

Vì n ≥ 1 nên ta cho S(1) = 1 ban đầu.

Công thức tổng quát:

* S(1) = 1

* S(n) = n + S(n – 1)

```
float TinhTong(n){
  if(n==1){
    return 1;
  }
  return TinhTong(n-1) + n;
}
```

Bài tập Đệ quy (Recursion)

Bài 2: Tính S(n) = 1^2 + 2^2 + 3^2 + ... + (n-1)^2 + n^2 .

* Giả sử ta cần tính S(4).

S(4) = **(1^2 + 2^2 + 3^2)** + 4^2 = S(3) + 4*4

* Để tính được S(4), ta cần tính S(3)

* Giả sử ta cần tính S(3)

S(3) = **(1^2 + 2^2)** + 3^2 = S(2) + 3*3

* Để tính được S(3), ta cần tính S(2)

* Giả sử ta cần tính S(2)

S(2) = **1^2** + 2^2 = S(1) + 2*2

* Để tính được S(2), ta cần tính S(1)

S(1) = 1*1 = 1 (điểm dừng).

Công thức tổng quát:

* S(1) = 1

* S(n) = n*n + S(n – 1)

```
int Tinh(int n) {
   if (n==1)
         return1;
   return Tinh(n-1) + n*n;
}
```

Bài tập Đệ quy (Recursion)

Bài 3 :Tính S(n) = 1 + 1.2 + 1.2.3 + ... + 1.2.3....n

```
long GiaiThua(int n)
 {
      if(n==1)
      {
           return 1;
      }
      return GiaiThua(n-1)*n;
 }

 long Tong(int n)
 {
      if(n == 1)
      {
           return 1;
      }
      return Tong(n-1) + GiaiThua(n-1)*n;
 }
```

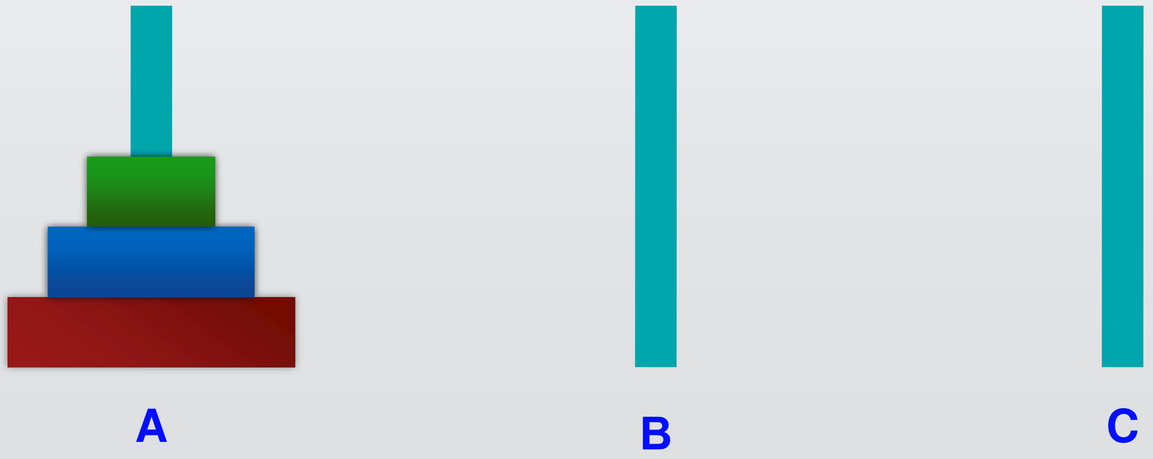
Bài tập Đệ quy (Recursion)

Bài 4: Bài toán Tháp Hà Nội (Hanoi Tower)

Cho 3 cọc A, B, C và bộ n đĩa. Các đĩa có lỗ thủng ở giữa và có kích thước khác nhau. Ban đầu các đĩa đều nằm trong một cọc. Đĩa nhỏ nằm trên đĩa lớn.

Yêu cầu:

- Chuyển n đĩa từ cọc đầu A sang cọc đích C mà vẫn giữ nguyên thứ tự sắp xếp của các đĩa.
- Mỗi lần chuyển chỉ chuyển 1 đĩa
- Đĩa lớn phải nằm dưới đĩa nhỏ.



Bài tập Đệ quy (Recursion)

Bài 4: Bài toán Tháp Hà Nội (Hanoi Tower)

Phân tích: Dùng Giải thuật Đệ Quy

1. **Nếu cọc A chỉ có 1 đĩa**
2. **Nếu cọc A có 2 đĩa**
3. **Nếu cọc A có >=3 đĩa**

Bài tập Đệ quy (Recursion)

Bài 4: Bài toán Tháp Hà Nội (Hanoi Tower)

Tóm lại:

- Nếu cọc A có **1 đĩa** : chuyển luôn từ **A —> C** (điều kiện dừng)
- Nếu cọc có từ 2 đĩa trở lên:

- Chuyển **n-1** đĩa từ cọc **A —> B**
- Chuyển đĩa thứ **n** từ **A —> C**
- Chuyển **n-1** đĩa từ cọc **B —> C**

```
void chuyen (int n, char X, char Y){ // Chuyển đĩa từ cọc X sang cọc Y
    System.out.println("Chuyển đĩa thứ " + n + " từ cọc " + X + "sang cọc"  + Y);
}

void thapHaNoi( int n, char A, char B, char C){
// điểm dừng
 if (n == 1){
     chuyen(1, A, C);
else{
    thapHaNoi(n-1, A, C, B);
    chuyen(n, A, C);
    thapHaNoi(n-1, B, A, C);
}
```

Bài tập Đệ quy - Tự luyện

Bài 1: Tính tổng $T(n) = \frac{1}{2} + \frac{2}{3} + \frac{3}{4} + \cdots + \frac{n}{n+1}$

Bài 2: Tính $R(n) = \sqrt{n + \sqrt{(n-1) + \sqrt{(n-2) + \sqrt{\ldots + \sqrt{1}}}}}$ n dấu căn

Bài tập 3 : Tính S(n) = 1 + 1/2 + 1/3 + ... + 1/n

Bài tập 4 : Tính S(n) = 1/2 + 1/4 + ... + 1/2n

Bài tập 5: Tính S(n) = 1/(1*2) + 1/(2*3) + 1/(n(*n-1))

Bài tập 6 :Tính S(x,n) = x + x^2 + x^3 + ... + x^n

Bài tập 7 :Tính S(n) = 1 + 1/(1+2) + 1/(1+2+3) + ... + 1/(1+2+3+...+n)

Bài tập 8 :Hãy đếm số lượng chữ số của số nguyên dương n

Bài tập Đệ quy - Tự luyện

Bài tập 9 :Hãy tính tổng các chữ số của số nguyên dương n

Bài tập 10 :Hãy tính tích các chữ số của số nguyên dương n

Bài tập 11 :Hãy đếm số lượng chữ số lẻ của số nguyên dương n

Bài tập 12 :Cho số nguyên dương n . Hãy tìm chữ số đầu tiên của n

Bài tập 13 :Tìm chữ số lớn nhất của số nguyên dương n

Bài tập 14 :Hãy tính tổng các chữ số chẵn của số nguyên dương n

Bài tập 15: Tính S(n) = 1 + 1/3 + 1/5 + ... + 1/(2n+1)