NGÔN NGỮ LẬP TRÌNH

Bài 10: ĐỆ QUY (Chương 13)

Giảng viên: Nguyễn Xuân Hùng

Mobile: 0908 386 366

Email: nguyenxuanhung@wru.vn

Nội dung

- Tìm hiểu về đề quy
- Các loại đệ quy
- Bài tập

1. 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:

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

1. Đệ quy (Recursion)

- Phương pháp thiết kế một giải thuật đệ quy:
  - Tham số hoá 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

1. 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

1. Đệ quy (Recursion)

- *<u>Ví dụ 1</u> : Lập hàm tính n! bằng đệ quy*

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

```
int GT(int n)
{
        if (n==0)     // điểm dừng
                return 1;
        else
                return n*GT(n-1);
}
```

1. Đệ quy (Recursion)

Minh họa

Gọi hàm **`answer <- GT(5)`**

CT chính: Chưa xong: answer <- GT(5)

1. Đệ quy (Recursion)

Minh họa

GT. 1st: N=5,	Chưa xong: 5*GT(4)

CT chính:	Chưa xong: answer <- GT(5)

1. Đệ quy (Recursion)

Minh họa

GT. 2nd: N=4, Chưa xong: 4*GT(3)
GT. 1st: N=5, Chưa xong: 5*GT(4)
CT chính: Chưa xong: answer <- GT(5)

1. Đệ quy (Recursion)

Minh họa

GT. 3rd: N=3,	**Chưa xong: 3*GT(2)**
GT. 2nd: N=4,	**Chưa xong: 4*GT(3)**
GT. 1st: N=5,	**Chưa xong: 5*GT(4)**
CT chính:	**Chưa xong: answer <- GT(5)**

1. Đệ quy (Recursion)

Minh họa

GT. 4th: N=2,	**Chưa xong: 2*GT(1)**
GT. 3rd: N=3,	**Chưa xong: 3*GT(2)**
GT. 2nd: N=4,	**Chưa xong: 4*GT(3)**
GT. 1st: N=5,	**Chưa xong: 5*GT(4)**
CT chính:	**Chưa xong: answer <- GT(5)**

1. Đệ quy (Recursion)

Minh họa

GT. 5th: N=1,	**Chưa xong: 1*GT(0)**
GT. 4th: N=2,	**Chưa xong: 2*GT(1)**
GT. 3rd: N=3,	**Chưa xong: 3*GT(2)**
GT. 2nd: N=4,	**Chưa xong: 4*GT(3)**
GT. 1st: N=5,	**Chưa xong: 5*GT(4)**
CT chính:	**Chưa xong: answer <- GT (5)**

1. Đệ quy (Recursion)

Minh họa

GT. 6th: N=0,	**xong: returns 1**
GT. 5th: N=1,	**Chưa xong: 1*GT(0)**
GT. 4th: N=2,	**Chưa xong: 2*GT(1)**
GT. 3rd: N=3,	**Chưa xong: 3*GT(2)**
GT. 2nd: N=4,	**Chưa xong: 4*GT(3)**
GT. 1st: N=5,	**Chưa xong: 5*GT(4)**
CT chính:	**Chưa xong: answer <- GT(5)**

1. Đệ quy (Recursion)

Minh họa

GT. 5th: N=1,	**xong: returns 1*1**
GT. 4th: N=2,	**Chưa xong: 2*GT(1)**
GT. 3rd: N=3,	**Chưa xong: 3*GT(2)**
GT. 2nd: N=4,	**Chưa xong: 4*GT(3)**
GT. 1st: N=5,	**Chưa xong: 5*GT(4)**
CT chính:	**Chưa xong: answer <- GT(5)**

1. Đệ quy (Recursion)

Minh họa

GT. 4th: N=2,	**xong: returns 2*1**
GT. 3rd: N=3,	**Chưa xong: 3*GT(2)**
GT. 2nd: N=4,	**Chưa xong: 4*GT(3)**
GT. 1st: N=5,	**Chưa xong: 5*GT(4)**
CT chính:	**Chưa xong: answer <- GT(5)**

1. Đệ quy (Recursion)

Minh họa

GT. 3rd: N=3,	**xong: returns 3*2**
GT. 2nd: N=4,	**Chưa xong: 4*GT(3)**
GT. 1st: N=5,	**Chưa xong: 5*GT(4)**
CT chính:	**Chưa xong: answer <- GT(5)**

1. Đệ quy (Recursion)

Minh họa

GT. 2nd: N=4,	**xong: returns 4*6**
GT. 1st: N=5,	**Chưa xong: 5*GT(4)**
CT chính:	**Chưa xong: answer <- GT(5)**

1. Đệ quy (Recursion)

Minh họa

GT. 1st: N=5,	**xong: returns 5*24**

CT chính:	**Chưa xong: answer <- GT(5)**

1. Đệ quy (Recursion)

Minh họa

CT chính: xong: answer <- 120

1. Đệ quy (Recursion)

- Ví dụ 2: Tính bằng đệ quy

Dãy số Fibonaci: $\begin{cases} F_1 = F_2 = 1; \\ F_n = F_{n-1} + F_{n-2}. \end{cases}$ $(n \geq 3)$

```
int  Fibo(int  n)
{
        if (n≤ 2)      // điểm dừng
                return 1;
        else
                return Fibo(n-1)+Fibo(n-2);
}
```

1. 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.

1. Đệ quy (Recursion)

- Tính giai thừa dùng vòng lặp:

```
int GT(int n)
{
    int s = 1;
    for(int i= 2; i<= n; i++)
            s = s* i;

    return s;
}
```

2. Các loại đệ quy

- Đệ quy tuyến tính (Linear Recursion)
- Đệ quy đuôi (Tail Recursion)
- Đệ quy nhị phân (Binary Recursion)
- Đệ quy lồng (Nested Recursion)
- Đệ quy tương hỗ (Mutual Recursion)

2. Các loại đệ quy

- Đệ quy tuyến tính (Linear Recursion)
  - mỗi lần hàm thực thi chỉ gọi đệ quy 1 lần

(only makes a single call to itself each time the function runs)

Ví dụ: tính giai thừa bằng đệ quy:

```
int GT(int n)
{
        if (n==0)        // điểm dừng
                return 1;
        else
                return n*GT(n-1);
}
```

2. Các loại đệ quy

- Đệ quy đuôi (Tail Recursion)
  - là một dạng đệ quy tuyến tính
  - lệnh cuối cùng của hàm là một lời gọi đệ quy (the last operation of the function is a recursive call)
  - dễ chuyển thành vòng lặp

Ví dụ: tìm Ước số chung lớn nhất của m, n bằng đệ quy (Greatest Common Denominator)

```
int gcd(int m, int n)
{
      int r;
      if (m < n)  return gcd(n,m);
      r = m%n;
      if (r == 0) return n;
      else  return gcd(n,r);
}
```

2. Các loại đệ quy

- Đệ quy nhị phân (Binary Recursion)
  - mỗi lần hàm thực thi có thể gọi đệ quy 2 lần

    (A recursive function which calls itself twice during the course of its execution)

Ví dụ: tính số các tổ hợp chập k của n phần tử (C(n,k)) bằng đệ quy:

$$C(n, k) = \begin{cases} 1 \text{ nếu } k = 0 \text{ or } k=n \\ C(n-1, k) + C(n-1, k-1) \text{ nếu } 0 < k < n \end{cases}$$

```
int choose(int n, int k)
{
        if (k == 0 || k == n)
                return 1;
        else
                return (choose(n-1, k) + choose(n-1, k-1));
}
```

2. Các loại đệ quy

- Đệ quy lồng (Nested Recursion)
  - trong đệ quy lồng, tham số trong lời gọi đệ quy là một lời gọi đệ quy
  - Đệ quy lồng phát triển rất nhanh

  Ví dụ: viết hàm Ackermann's:

$$Acker(m,n) = \begin{cases} n+1 & \text{if } m=0 \\ Acker(m-1,1) & \text{if } n=0 \\ Acker(m-1, Acker(m,n-1)) & \text{(other cases)} \end{cases}$$

```
int ackerman (int m, int n)
{
      if (m == 0) return (n+1);
      else if (n == 0)
            return ackerman(m-1,1);
      else
            return ackerman(m-1, ackerman(m,n-1));
}
```

2. Các loại đệ quy

- Đệ quy lồng (Nested Recursion)

m\n	0	1	2	3	4	n
0	0+1	1+1	2+1	3+1	4+1	$n+1$
1	A(0,1)	A(0,A(1,0))	A(0,A(1,1))	A(0,A(1,2))	A(0,A(1,3))	$n+2 = 2+(n+3)-3$
2	A(1,1)	A(1,A(2,0))	A(1,A(2,1))	A(1,A(2,2))	A(1,A(2,3))	$2n+3 = 2*(n+3)-3$
3	A(2,1)	A(2,A(3,0))	A(2,A(3,1))	A(2,A(3,2))	A(2,A(3,3))	$2^{(n+3)}-3$
4	A(3,1)	A(3,A(4,0))	A(3,A(4,1))	A(3,A(4,2))	A(3,A(4,3))	$\underbrace{2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}}_{n+3 \text{ twos}} - 3$
5	A(4,1)	A(4,A(5,0))	A(4,A(5,1))	A(4,A(5,2))	A(4,A(5,3))	A(4, A(5, n-1))
6	A(5,1)	A(5,A(6,0))	A(5,A(6,1))	A(5,A(6,2))	A(5,A(6,3))	A(5, A(6, n-1))

2. Các loại đệ quy

- Đệ quy hỗ tương (Mutual Recursion)
  - hàm đệ quy không cần thiết phải gọi chính nó
  - một số hàm đệ quy gọi lẫn nhau
  - ví dụ: hàm A gọi hàm B, hàm B gọi hàm C, hàm C lại gọi hàm A

Ví dụ: viết hàm kiểm tra chẵn, lẻ bằng đệ quy:

```
int is_even(unsigned int n)
{
        if (n==0) return 1;
        else return (is_odd(n-1));
}
int is_odd(unsigned int n)
{
        return (!is_even(n));
}
```

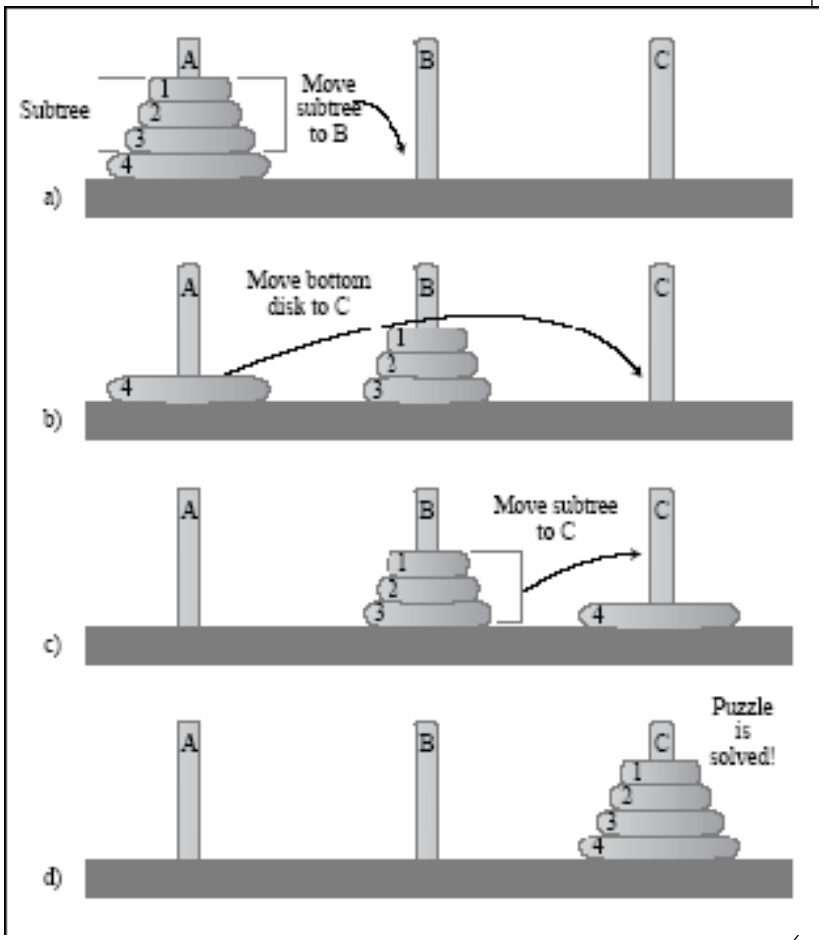
3. Giải một số bài tập đệ quy

- Ví dụ 1: Bài toán tháp Hà Nội
  - Chuyển một chồng đĩa gồm n đĩa với kích thước khác nhau từ cột A sang cột C theo cách:

+ Mỗi lần chỉ chuyển 1 đĩa

+ Không có trường hợp đĩa lớn được đặt trên đĩa nhỏ

+ Khi chuyển có thể dùng cột trung gian B



3. Giải một số bài tập đệ quy

- Ví dụ 1: Bài toán tháp Hà Nội

Tham số hoá bài toán: HaNoi (n, A, B, C)

Trong đó: n: Số đĩa.

A: Cọc nguồn cần chuyển đĩa đi

B: Cọc trung gian

C: Cọc đích để chuyển đĩa đến

(A, B, C có kiểu ký tự)

3. Giải một số bài tập đệ quy

Giải thuật đệ quy bài toán Tháp Hà Nội:

- Trường hợp suy biến (điểm dừng):

  Nếu n = 1 thì chuyển đĩa từ A qua C

- Trường hợp chung (n ≥ 2):

  Thử với n=2: + Chuyển đĩa thứ nhất từ A sang B

  + Chuyển đĩa thứ hai từ A sang C

  + Chuyển đĩa thứ nhất từ B sang C

→ Tổng quát:

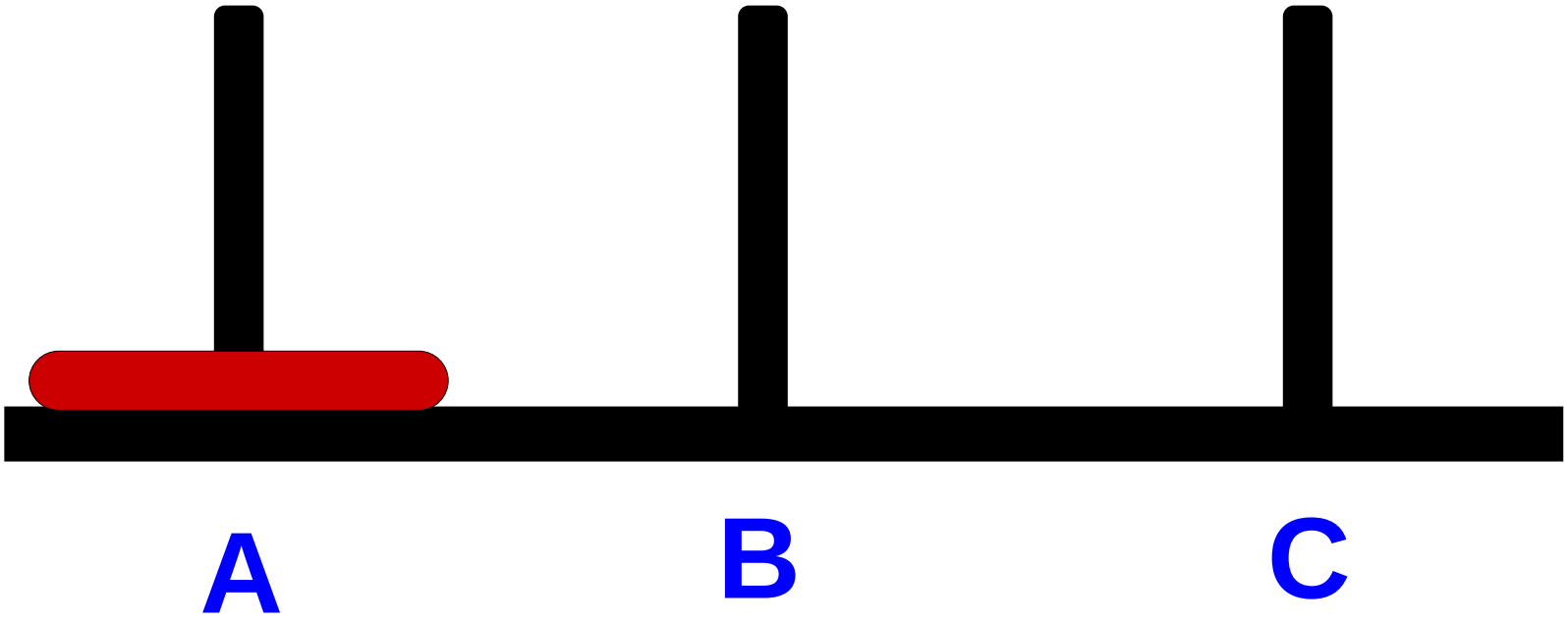
+ Chuyển (n -1) đĩa từ A sang B (C làm trung gian)

+ Chuyển 1 đĩa từ A sang C (B làm trung gian)

+ Chuyển (n -1) đĩa từ B sang C (A làm trung gian)

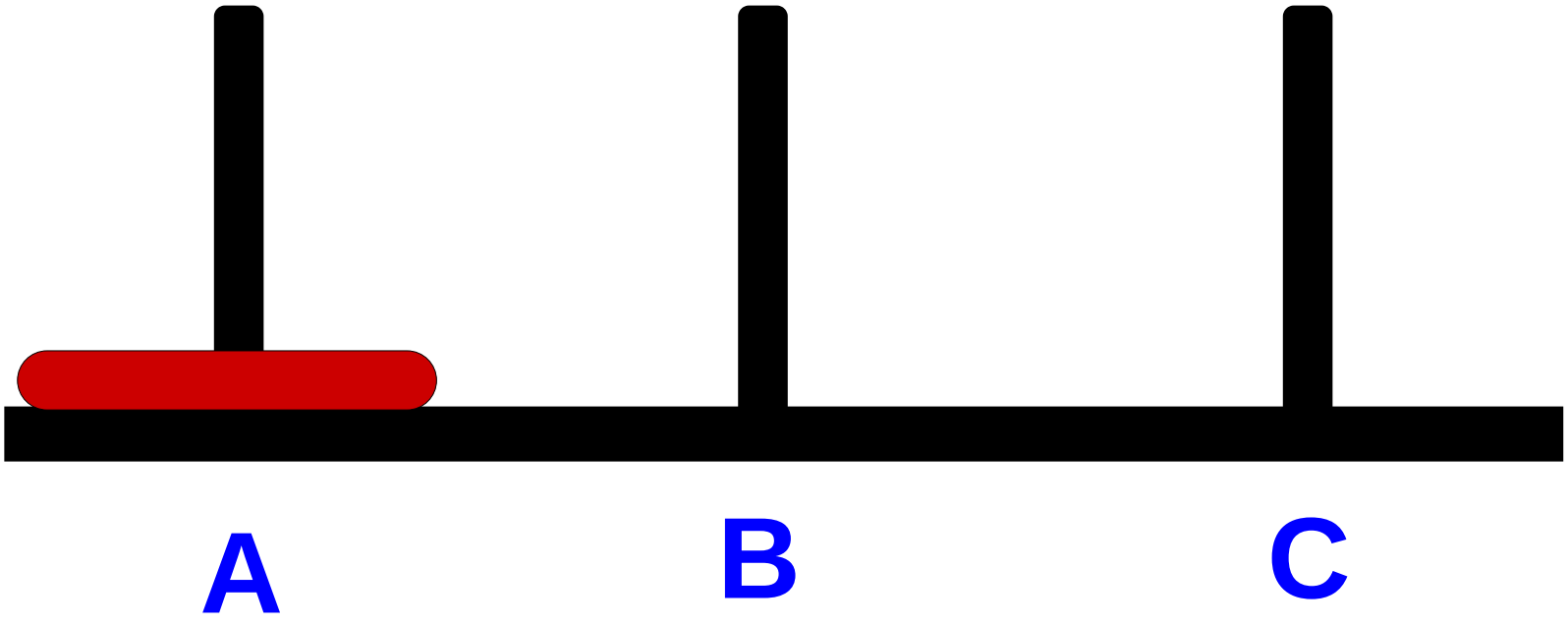
3. Giải một số bài tập đệ quy

1 đĩa



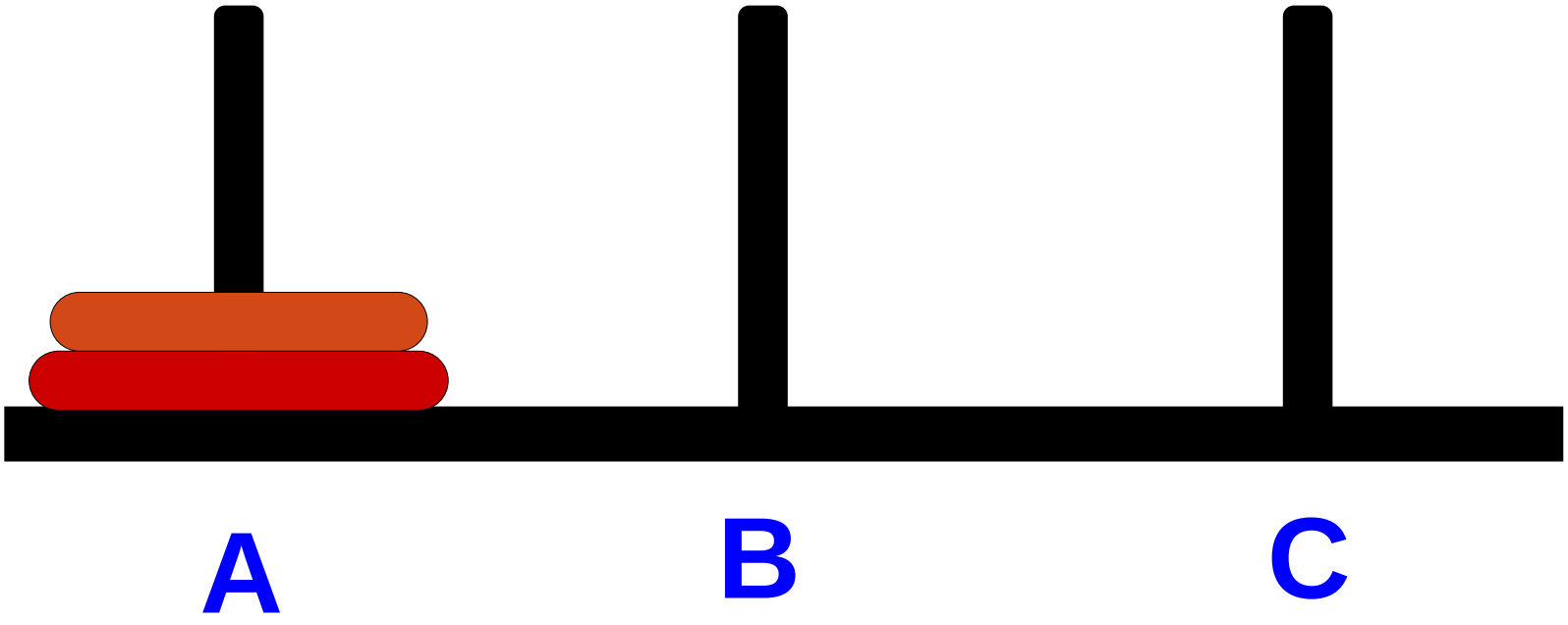
3. Giải một số bài tập đệ quy

1 đĩa



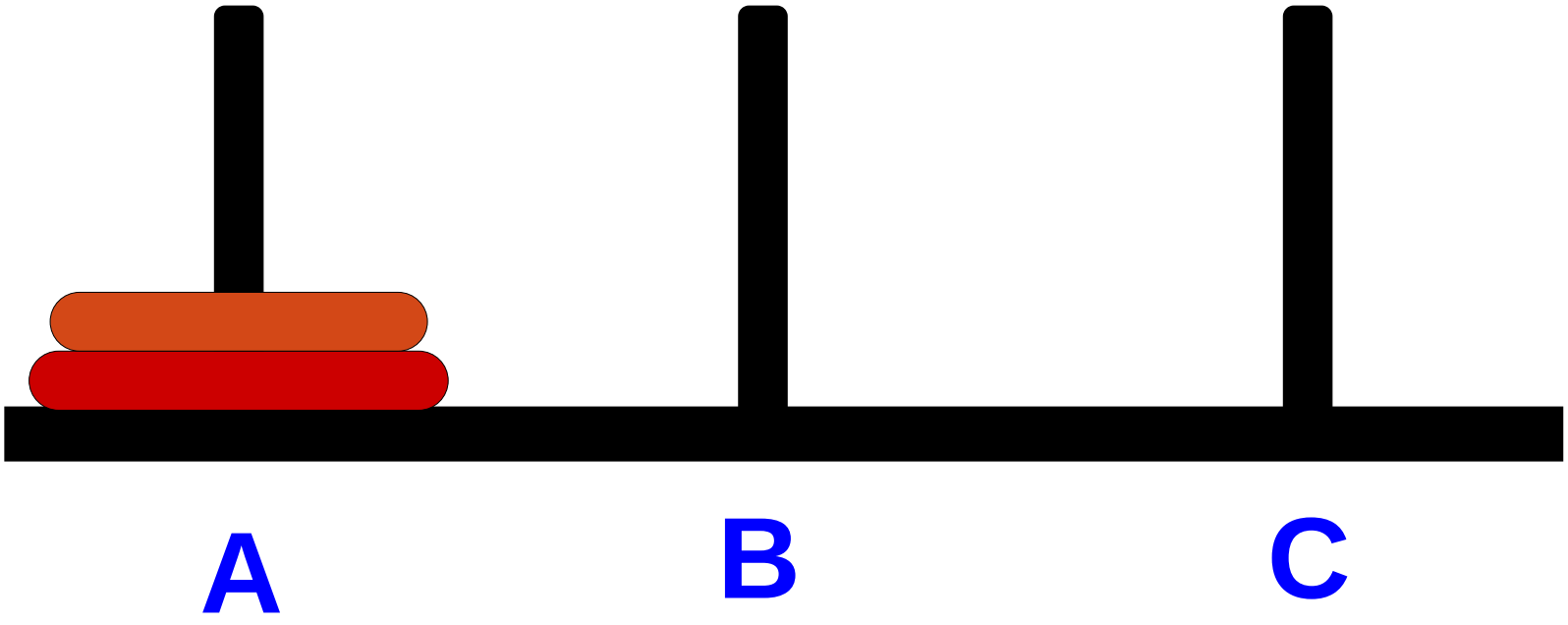
3. Giải một số bài tập đệ quy

2 đĩa



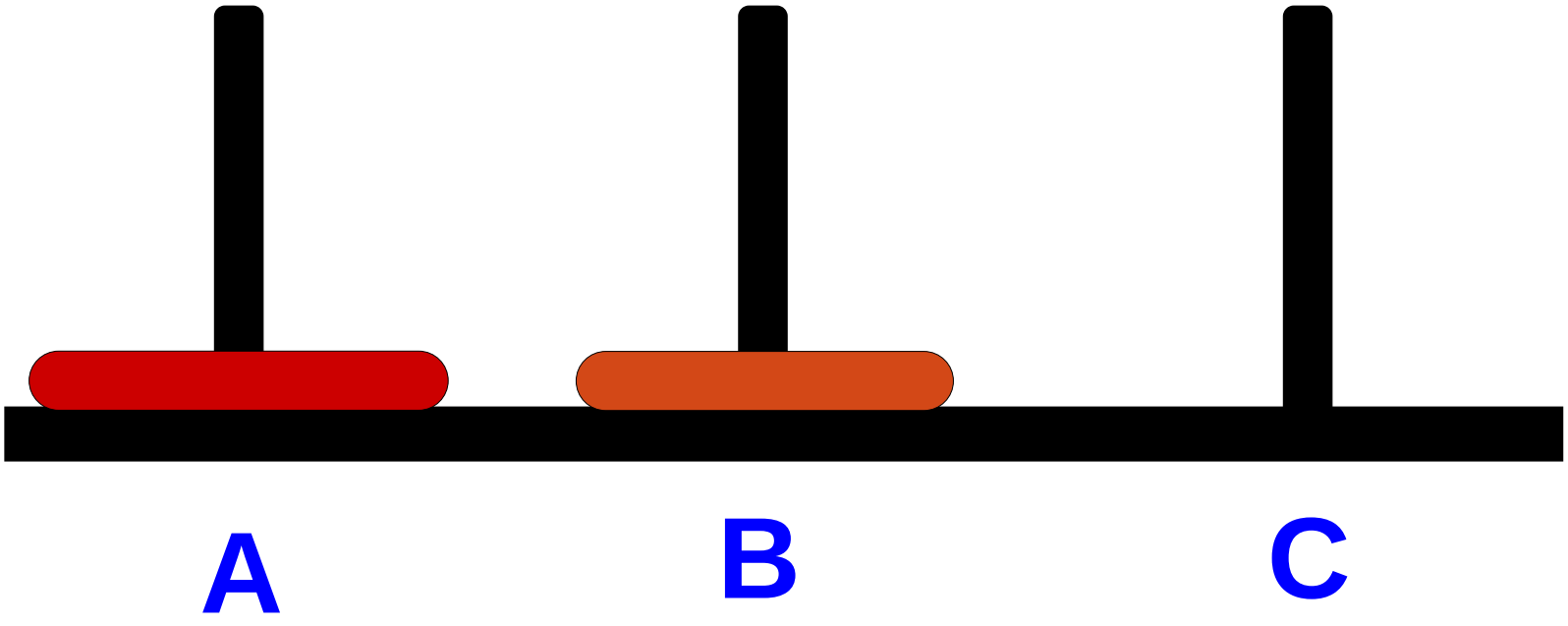
3. Giải một số bài tập đệ quy

2 đĩa



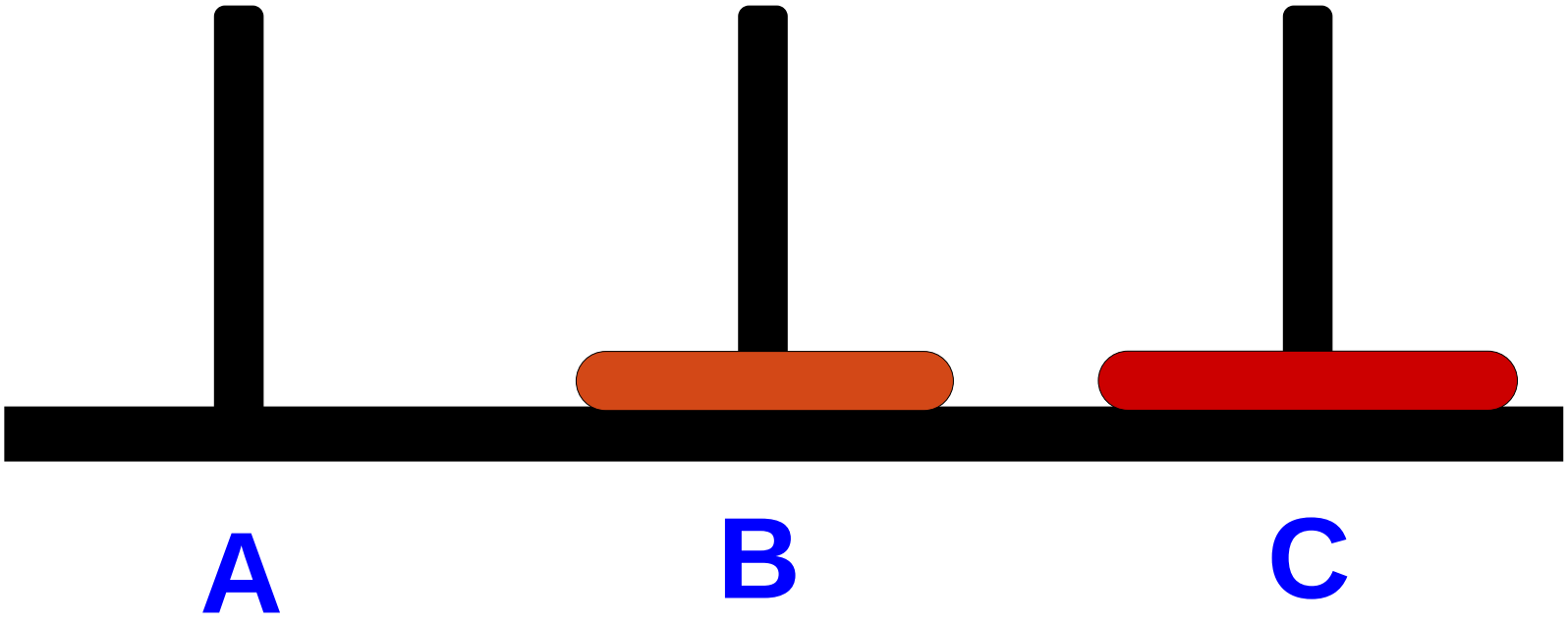
3. Giải một số bài tập đệ quy

2 đĩa



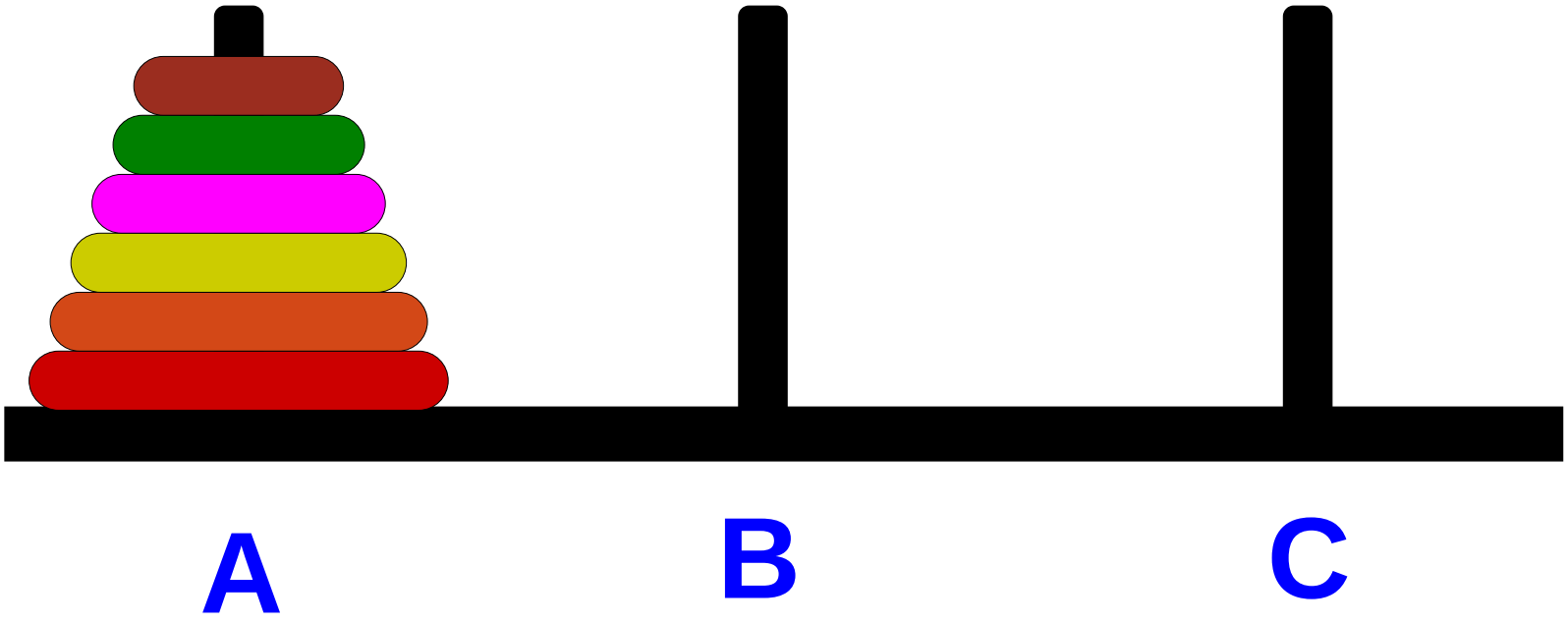
3. Giải một số bài tập đệ quy

2 đĩa



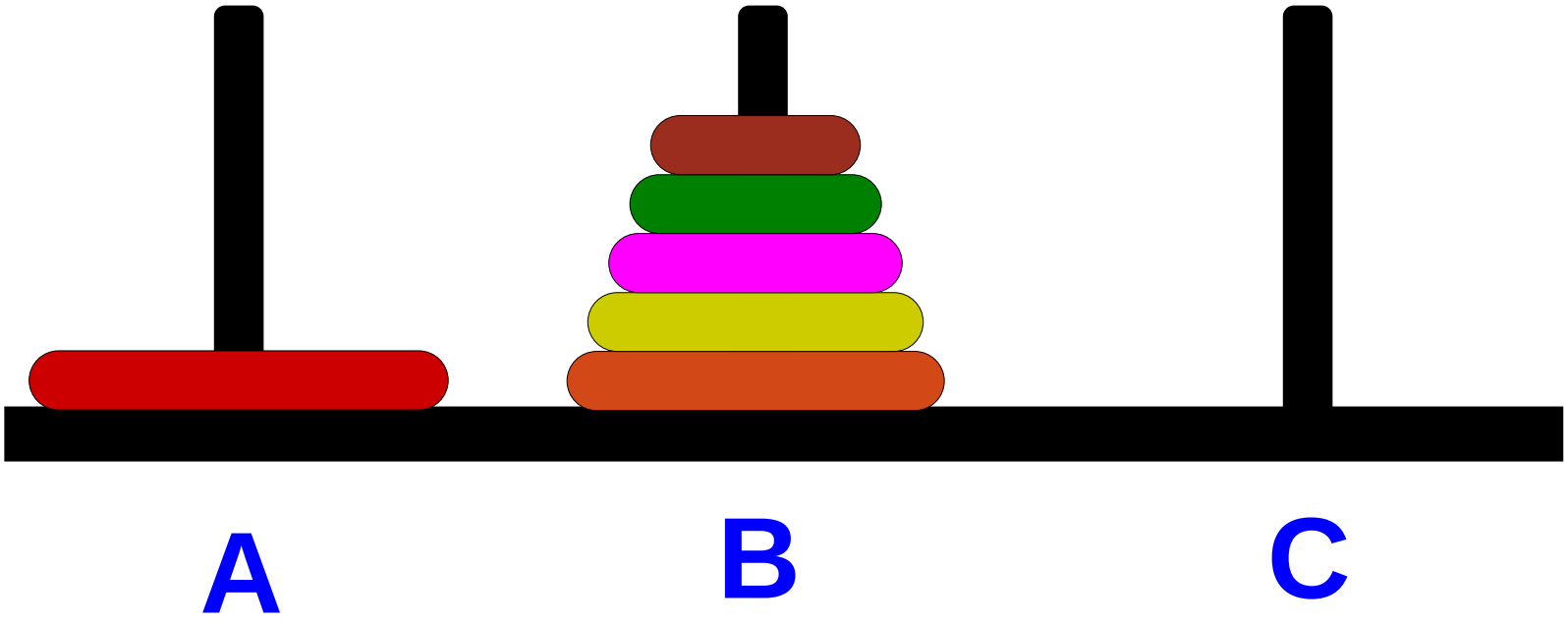
3. Giải một số bài tập đệ quy

N đĩa



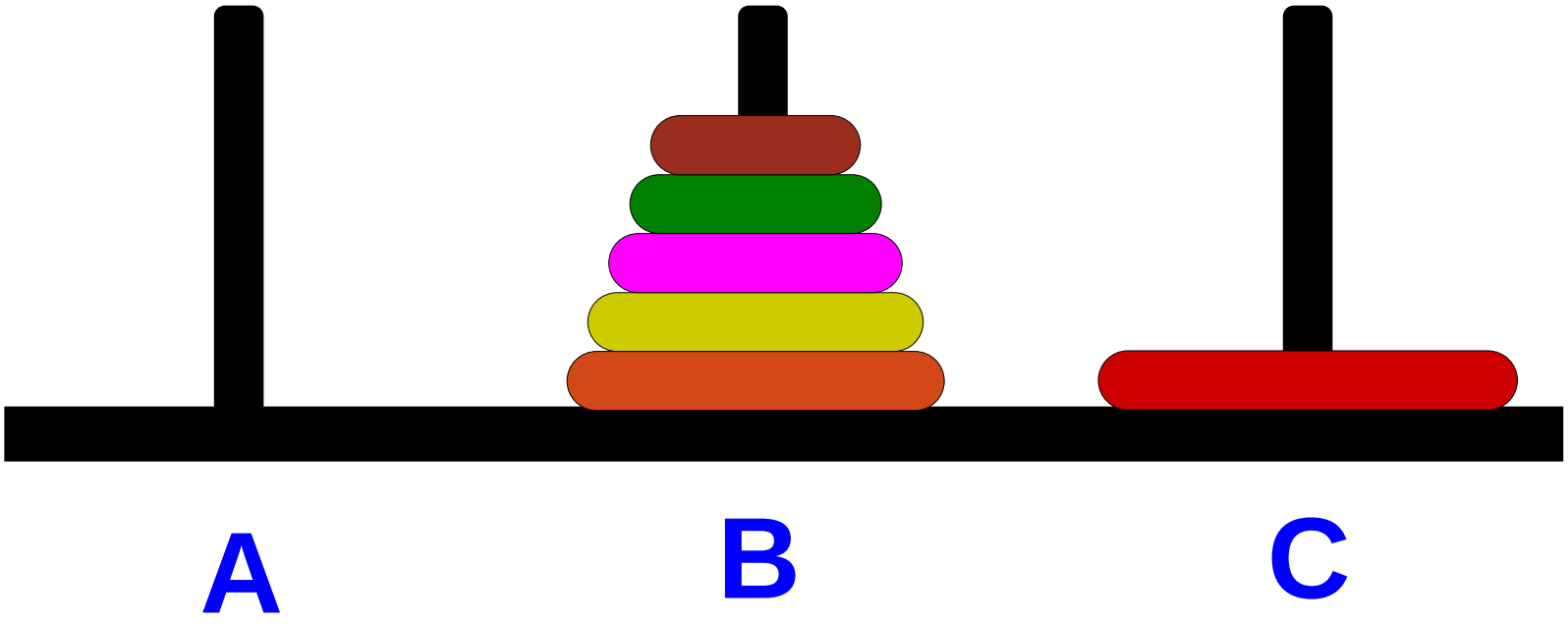
3. Giải một số bài tập đệ quy

N đĩa



3. Giải một số bài tập đệ quy

N đĩa



3. Giải một số bài tập đệ quy

Giải thuật đệ quy bài toán Tháp Hà Nội:

```
void HaNoi (int n, char A, char B, char C){
        if (n==1)
                cout<<A<<"→"<< C;
        else{
                HaNoi(n -1, A, C, B);
                HaNoi(1, A, B, C);
                HaNoi(n -1, B, A, C);
        }
}
```

3. Giải một số bài tập đệ quy

- Bài tập: Viết hàm đệ quy cho phép in chuỗi đảo ngược

  - Trường hợp chung: + In ký tự cuối của chuỗi X
    + Lấy phần chuỗi còn lại
  - Trường hợp suy biến: Nếu chuỗi rỗng thì không làm gì

```
void InNguoc(char *X)
{
        static int len=strlen(X);
        if (len>0) {
                cout<<X[len-1];
                len--;
                InNguoc(X);
        }
}
```

3. Giải một số bài tập đệ quy

- Bài tập: Viết hàm đệ quy cho phép xuất biểu diễn nhị phân của 1 số nguyên n, ví dụ: n=13 → 1101

Xuất dạng nhị phân của n:

Nếu (n/2>0) Xuất dạng nhị phân của n/2;

Xuất (n%2);

```
void XuatNhiPhan(int n)
{
        if (n/2>0)
                XuatNhiPhan (n/2);
        cout<<n%2;
}
```

3. Giải một số bài tập đệ quy

- Bài tập: Viết hàm đệ quy cho phép nhập số giây và chuyển thành giờ, phút, giây. Ví dụ: nhập 3665 -> 1 giờ 1 phút 5 giây

```
void DoiGio(int n, int &g, int &p, int &gi)
{
        if (n<60)
                gi=n;
        else if (n/3600>0) {
                        g=n/3600;
                        return DoiGio(n%3600, g, p, gi);
                }
            else{
                        p=n/60;
                        return DoiGio(n%60, g, p, gi);
                }
}
```

3. Giải một số bài tập đệ quy

- Bài tập: Viết hàm đệ quy cho phép kiểm tra xem một số có phải số nguyên tố không

```
isPrime(N)  = prime(N, N-1)
prime(N, 1) = true
prime(N, D) = if D divides N, false
              else prime(N, D-1)
```

```
int isPrime (int N)
{
        if (N==1) return 1;
        int static M=N-1;
        if (M==1) return 1;
        else if (N%M==0) return 0;
                else {
                        M--;
                        isPrime (N);
                }
}
```

3. Giải một số bài tập đệ quy

- Bài tập: Viết hàm đệ quy cho phép tính tổng các chữ số của một số nguyên n, ví dụ n=1980 =>Sum=1+9+8+0=18

Tổng các chữ số của n:

+ Nếu (n<10) thì Tổng bằng n;

+ Nếu (n<10) thì Tổng bằng n%10 + Tổng các chữ số của n/10

```
int tong(int n)
{
        if (n<10)
                return n;
        else
                return n%10+tong(n/10);
}
```

3. Giải một số bài tập đệ quy

- Bài tập: Viết hàm đệ quy cho phép xuất ngược một số nguyên n, ví dụ n=1980 → xuất 0891

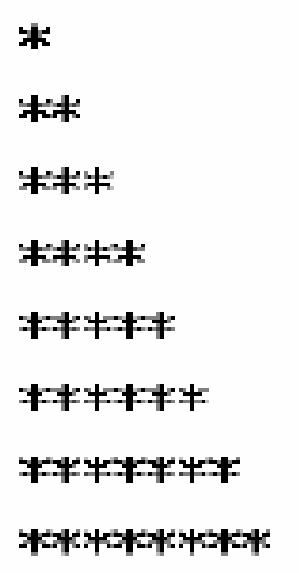
Xuất ngược n:

+ Nếu n<10 thì Xuất n
+ Nếu n>=10 thì Xuất n%10 và Xuất ngược n/10

```
void XuatSoNguoc(int n)
{
        if (n<10)
                cout<<n;
        else {
                cout<<n%10;
                XuatSoNguoc(n/10);
        }
}
```

3. Giải một số bài tập đệ quy

- Bài tập: In hình tam giác sau bằng cách đệ quy



```
void InSao(int n)
{
        if (n>1)
                InSao(n-1);
        for (int i=0; i<n; i++)
                cout<<"*";
        cout<<endl;
}
```

3. Giải một số bài tập đệ quy

- Bài tập: Cho mảng a có n phần tử, tính tổng các phần tử trong mảng bằng đệ quy

Điều kiện biên: Mảng 0 phần tử thì tổng bằng 0

Giải thuật chung:

$$Sum(a,n) = \begin{cases} 0 , & n=0 \\ a[n-1] + Sum(a, n-1), & n>0 \end{cases}$$

3. Giải một số bài tập đệ quy

- Bài tập: Cho mảng a có n phần tử, tìm giá trị lớn nhất trong mảng bằng đệ quy

Điều kiện biên: Mảng 1 phần tử thì trị lớn nhất là a[0]

Giải thuật chung:

$$Max(a,n) = \begin{cases} a[0], & n=1 \\ a[n-1] > Max(a, n-1)\ ?\ a[n-1] : Max(a,n-1), & n>1 \end{cases}$$

EOF!