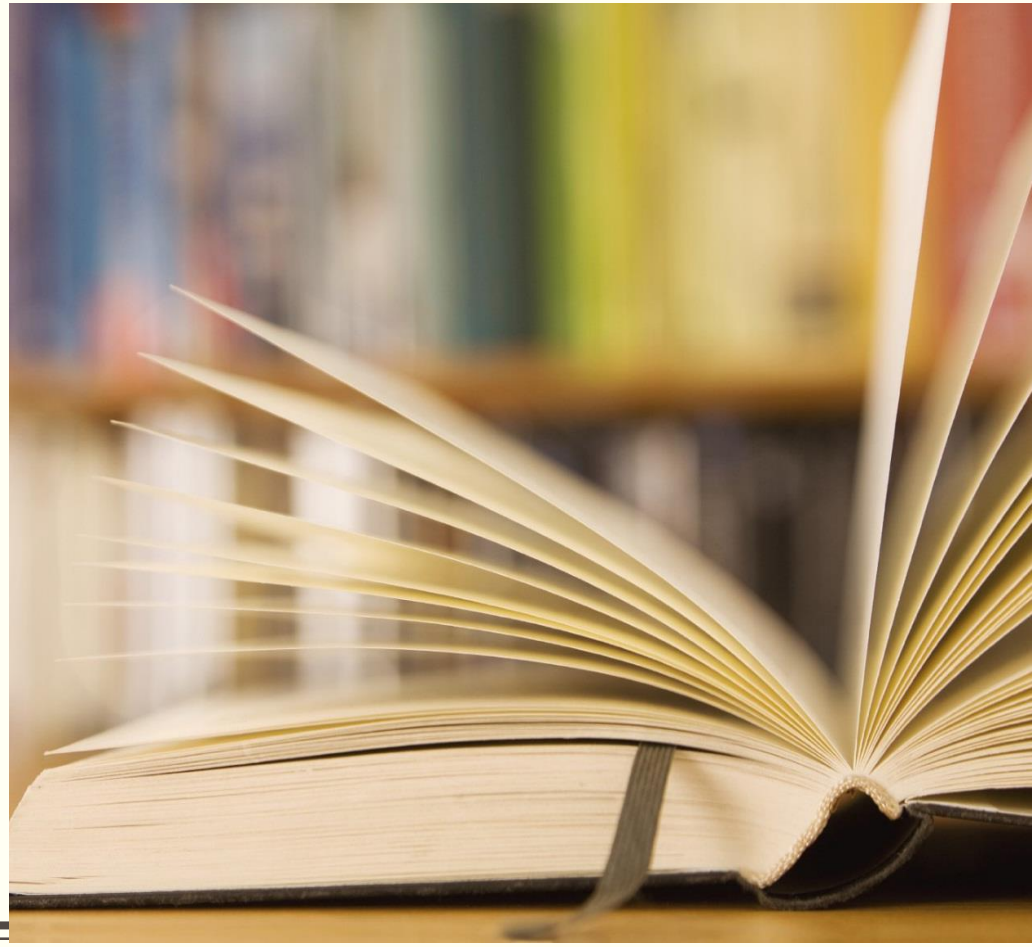
KỸ THUẬT ĐỆ QUY



Mục tiêu

- Hiểu được khái niệm đệ qui
- Hiểu được cấu trúc và nguyên lý hoạt động của chương trình đệ qui
- Biết cách vận dụng cài đặt chương trình theo đệ qui

ĐỆ QUI

1. Khái niệm
2. Cấu trúc chương trình
3. Nguyên lý hoạt động
4. Ưu và nhược điểm
5. Bài toán minh hoạ
    1. Tính N!
    2. Tìm số hạng thứ n của dãy Fibonacci
    3. Bài toán tháp Hà Nội
6. Khử đệ qui

Khái niệm

- Một khái niệm X gọi là định nghĩa theo đệ qui nếu trong chính định nghĩa X có sử dụng khái niệm X

Ví dụ:

1. Định nghĩa Số tự nhiên:
   - 0 là một số tự nhiên
   - n > 0 là số tự nhiên nếu n – 1 là số tự nhiên
2. Định nghĩa n giai thừa
   - 0! = 1
   - Nếu n >0 thì n! = n* (n-1)!

Cấu trúc chương trình

Chương trình đệ qui gồm hai phần chính:

1. Điều kiện thoát khỏi đệ qui
2. 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.

Cấu trúc chương trình(tiếp…)

```
Chươngtrình A(x) {
        If (điều kiện thoát đệ qui)
         Kết thúc đệ qui;
        ….
        ChươngtrìnhA(x’);
        ….
}
```

Với x’ < x

Bài toán tính n!

- Input:
  N (là số tự nhiên)
- Output:
  N!

Bài toán tính N!(tiếp…)

- Theo định nghĩa đệ qui ta có:
  1. 0!=1
  2. N >0, N! = N*(N-1)!
- Ví dụ: Tính 5!

  5! = 5*4! =5*4*3! = 5*4*3*2! = 5*4*3*2*1! = 5*4*3*2*1*0! =5*4*3*2*1*1 = 120

Bài toán tính N!(tiếp…)

- GiaiThừa(n)

  Bước 1: Nếu n = 0 thì GiaiThừa =1, ngược lại

  GiaiThừa = n*GiaiThừa(n-1)

Bài toán tính N!(tiếp…)

```
1   // Chuong trinh nhap vao mot so nguyen duong n, tinh n!
2   #include<iostream>
3   using namespace std;
4   // Ham de quy tinh n!
5   long int giaithua(int n){
6         if (n<0) cout<<"Nhap so <0"<<endl;
7         else
8               if (n==0||n==1) return 1;
9               else
10               return n*giaithua(n-1);
11
12  }
```

Nguyên lý hoạt động

Khi ta thực hiện gọi hàm Giai thừa: GiaiThừa(5)

Ưu điểm

- Thích hợp với các bài toán có bản chất đệ quy
- Sáng sủa
- Dễ hiểu
- Thể hiện được bản chất của bài toán
- Chương trình ngắn gọn

Nhược điểm

- Tốc độ thực hiện chương trình chậm
- Lỗi "Tràn stack"

Bài toán tìm số hạng thứ n của dãy Fibonacci

- Số hạng thứ n của dãy fibonacci được định nghĩa:
  - $F_0 = 1$
  - $F_1 = 1$
  - $F_n = F_{n-1} + F_{n-2}$
- Ví dụ:
  - 1,1,2,3,5,8,13,21,34,55,…. là dãy fibonacci

Bài toán tìm số hạng thứ n của dãy Fibonacci (tiếp…)

- Input
  Cho n là số tự nhiên
- Output
  Tính số hạng thứ n của dãy fibonacci

Bài toán tìm số hạng thứ n của dãy Fibonacci (tiếp…)

- Điều kiện thoát khỏi đệ qui

  $N<=1$

- Đệ qui

  $F_n = F_{n-1} + F_{n-2}$

Bài toán tìm số hạng thứ n của dãy Fibonacci (tiếp…)

Fibonacci(n)

1. Nếu n<=1 thì Fibonacci =1 ngược lại
2. F(n) = F(n-1) +F(n-2)

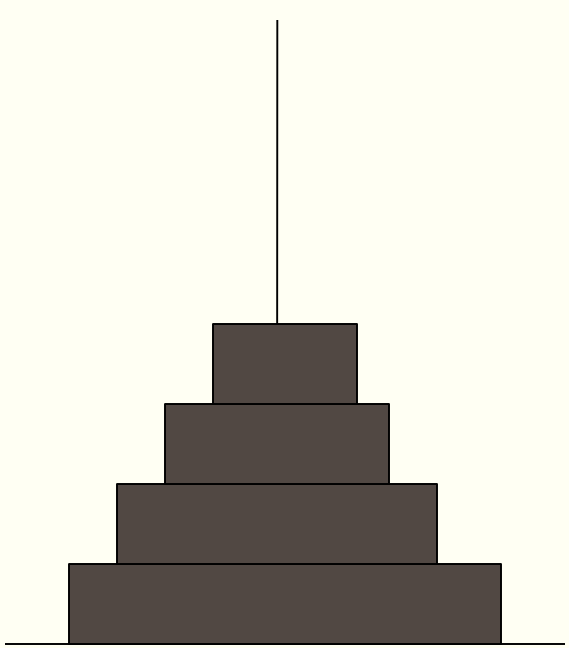
Bài toán tìm số hạng thứ n của dãy Fibonacci (tiếp…)

```
1   // Chuong trinh tim so fibonacii thu n
2   #include<iostream>
3   using namespace std;
4   // Ham de quy tinh so Fibonacii thu n
5   long int Fibo(int n)
6   {
7       if (n<0)
8           cout<<"Nhap so <0"<<endl;
9       else if (n==0||n==1)
10          return 1;
11      else
12          return Fibo(n-1)+Fibo(n-2);
13
14  }
15  // Ham main su dung ham Fibo tinh so fibonacii
    thu n voi n nhap tu ban phim
16  int main()
17  {
18
19      int n;
20      // Nhap n
21      cout<<"Moi nhap n=";
22      cin>>n;
23      // Hien thi n va so fibonacii thu n
24      cout<<"So Fibo thu "<<n<<" la
    "<<Fibo(n)<<endl;
25      return 0;
26  }
```

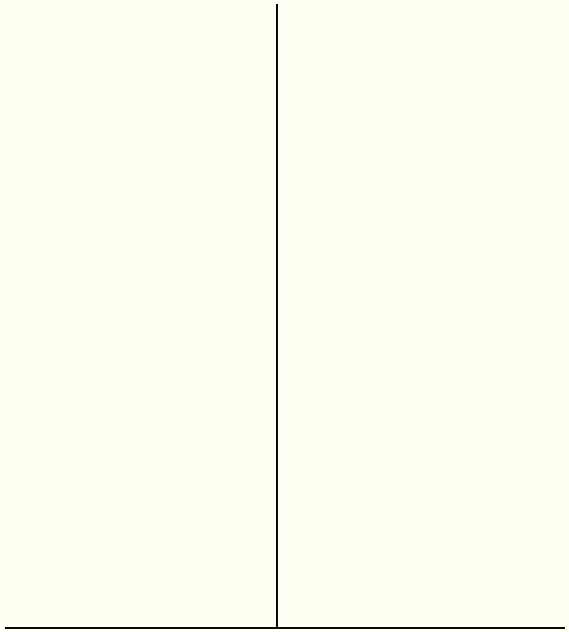
Bài toán tháp Hà nội

- Có 2 cột A,B
- Tại cột A người ta xếp n đĩa kích thước khác nhau, đĩa lớn ở dưới, đĩa nhỏ ở trên.
- Yêu cầu hãy chuyển các đĩa từ cột A sang B sao cho:
  - Mỗi lần chỉ được chuyển 1 đĩa
  - Ở các cột, đĩa lớn luôn ở dưới, đĩa nhỏ ở trên

Bài toán Tháp Hà Nội

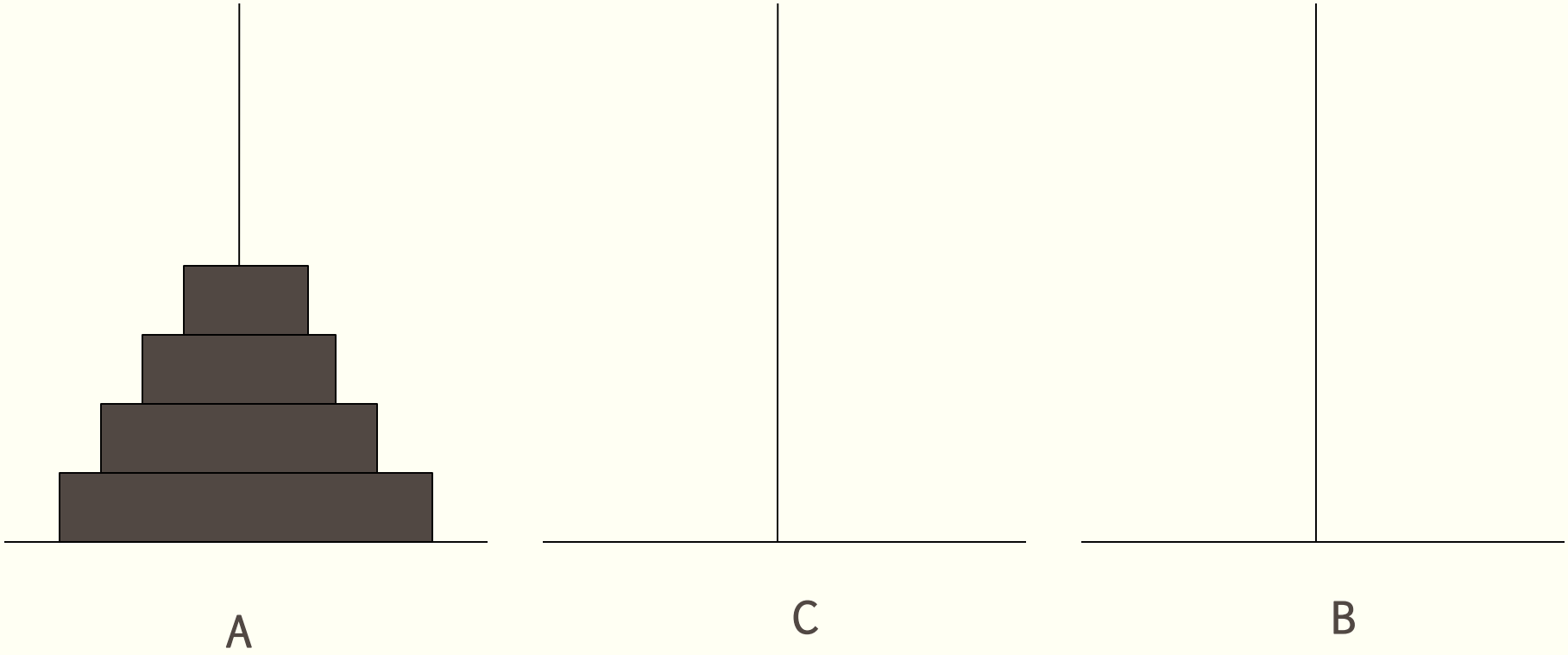


A

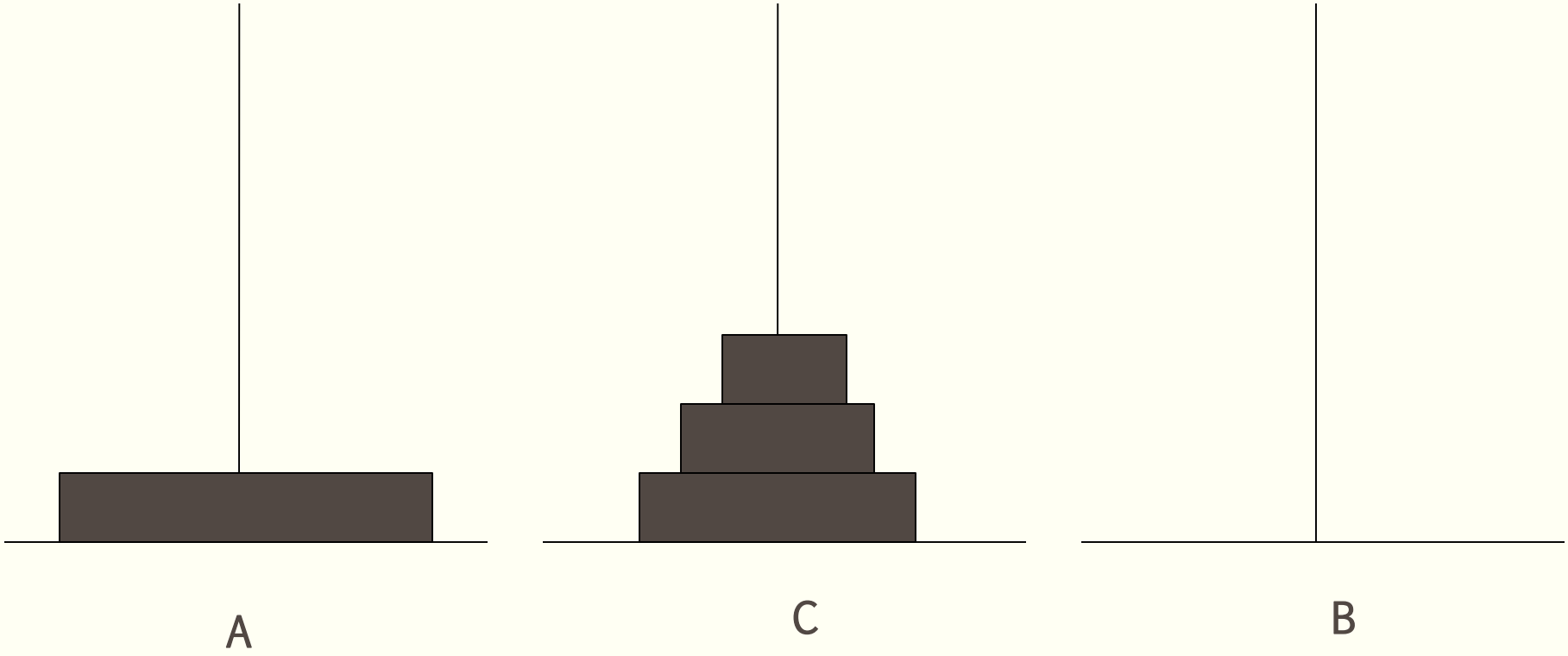


B

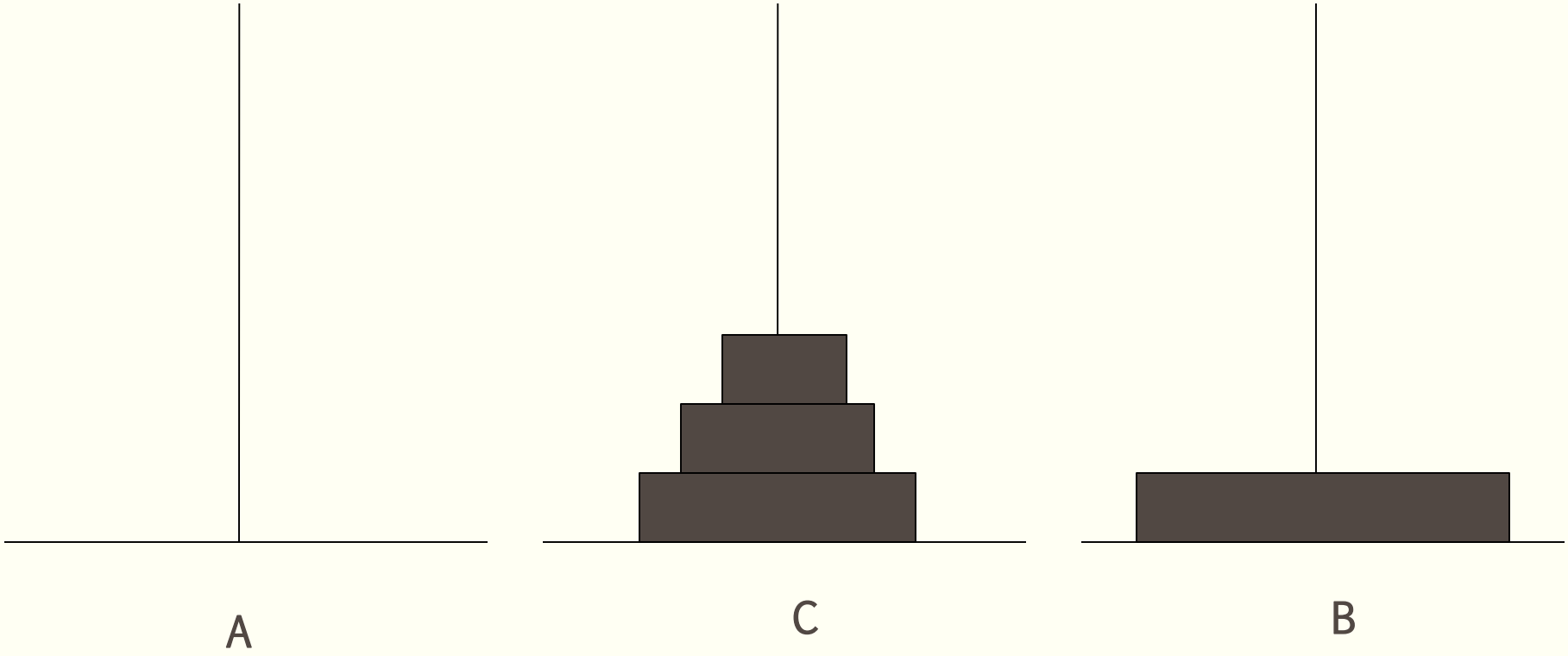
Bài toán Tháp Hà Nội



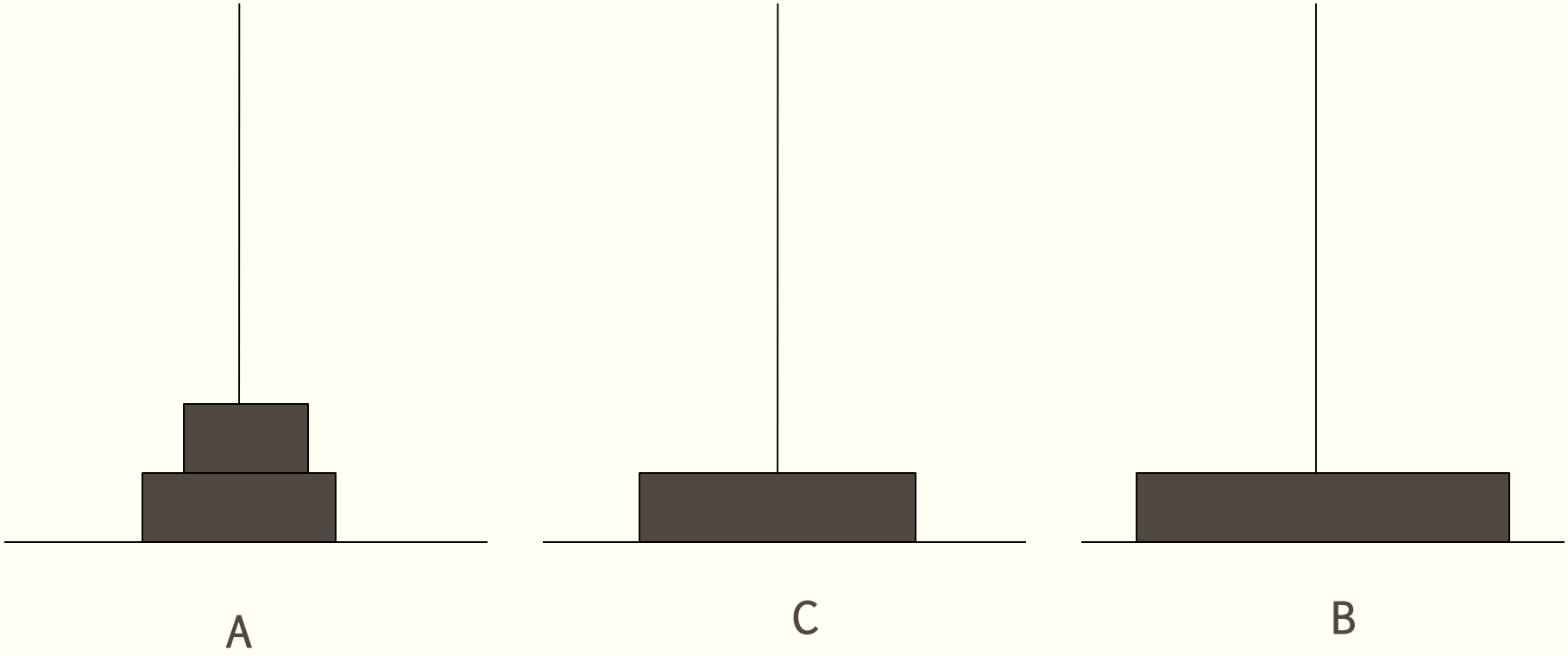
Bài toán Tháp Hà Nội



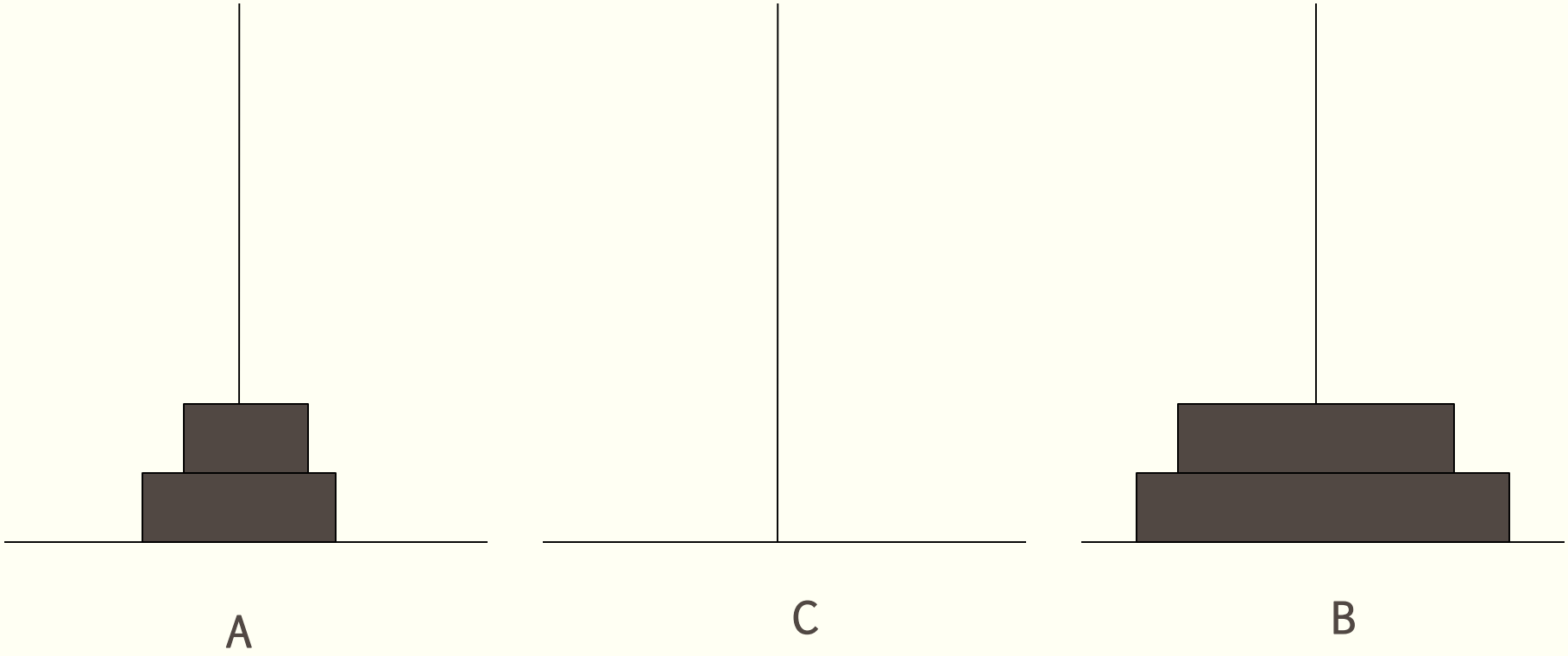
Bài toán Tháp Hà Nội



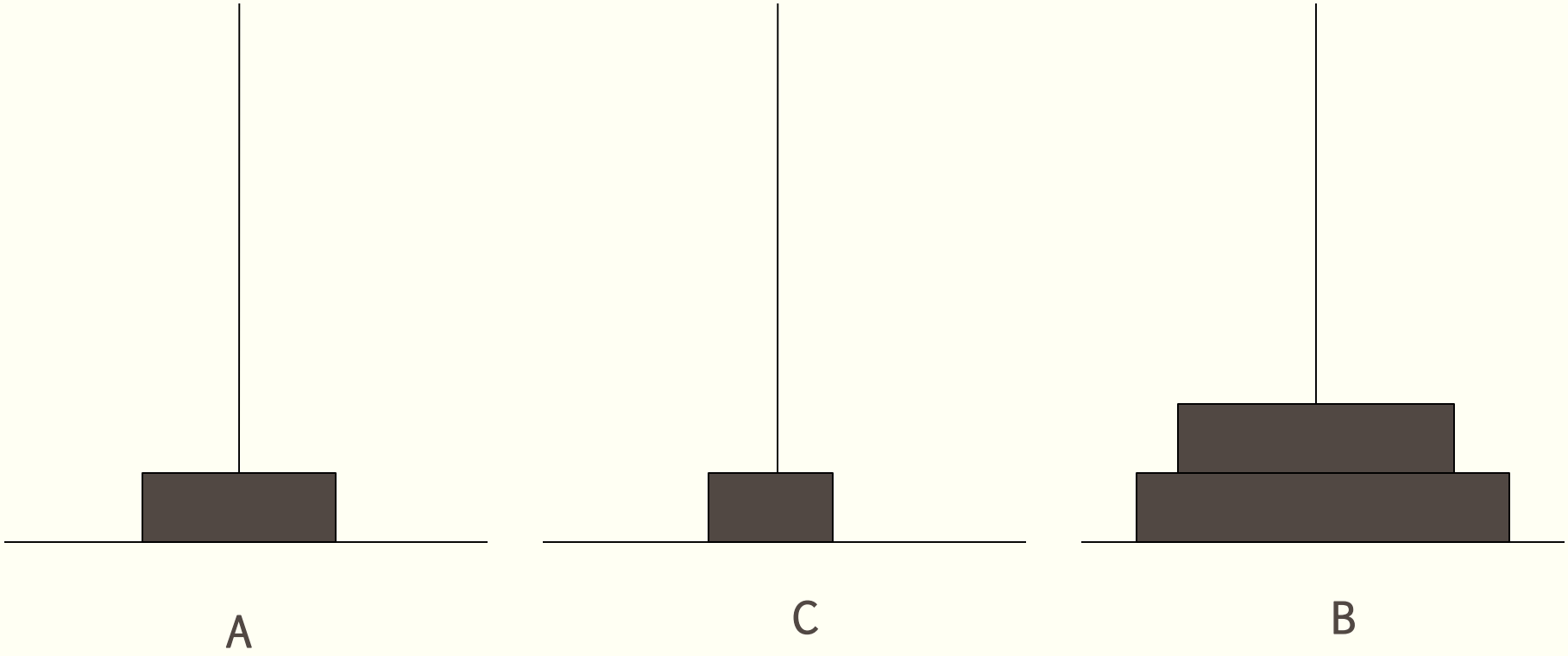
Bài toán Tháp Hà Nội



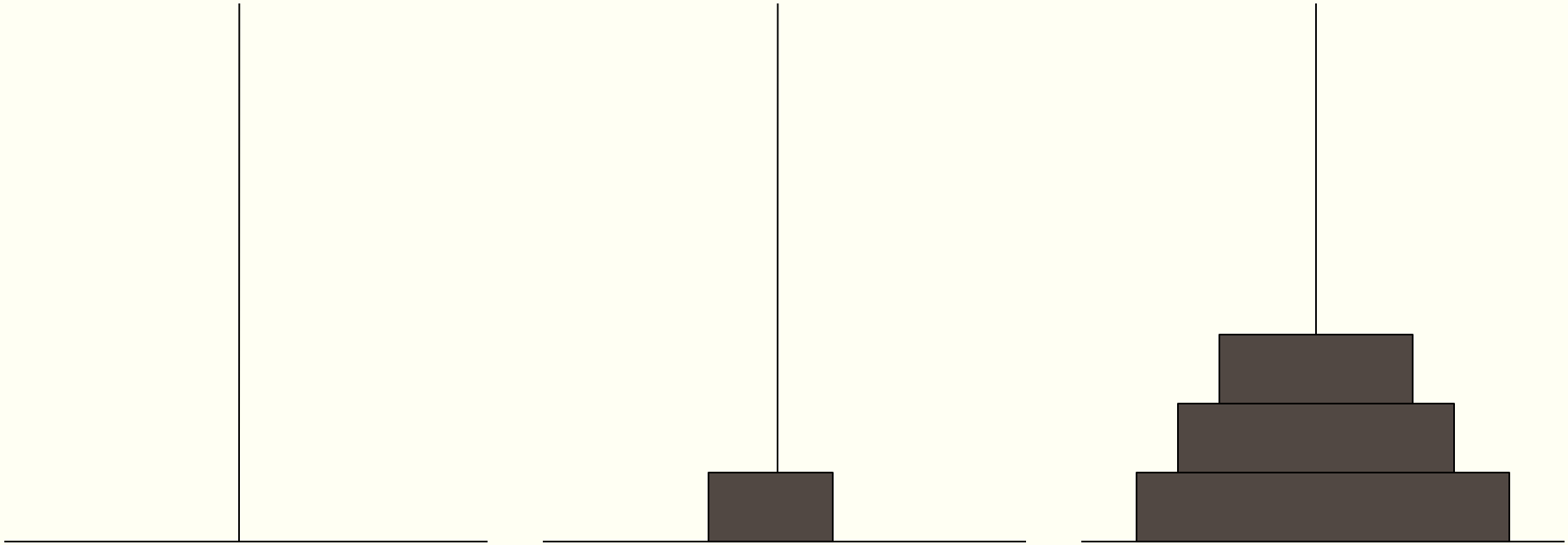
Bài toán Tháp Hà Nội



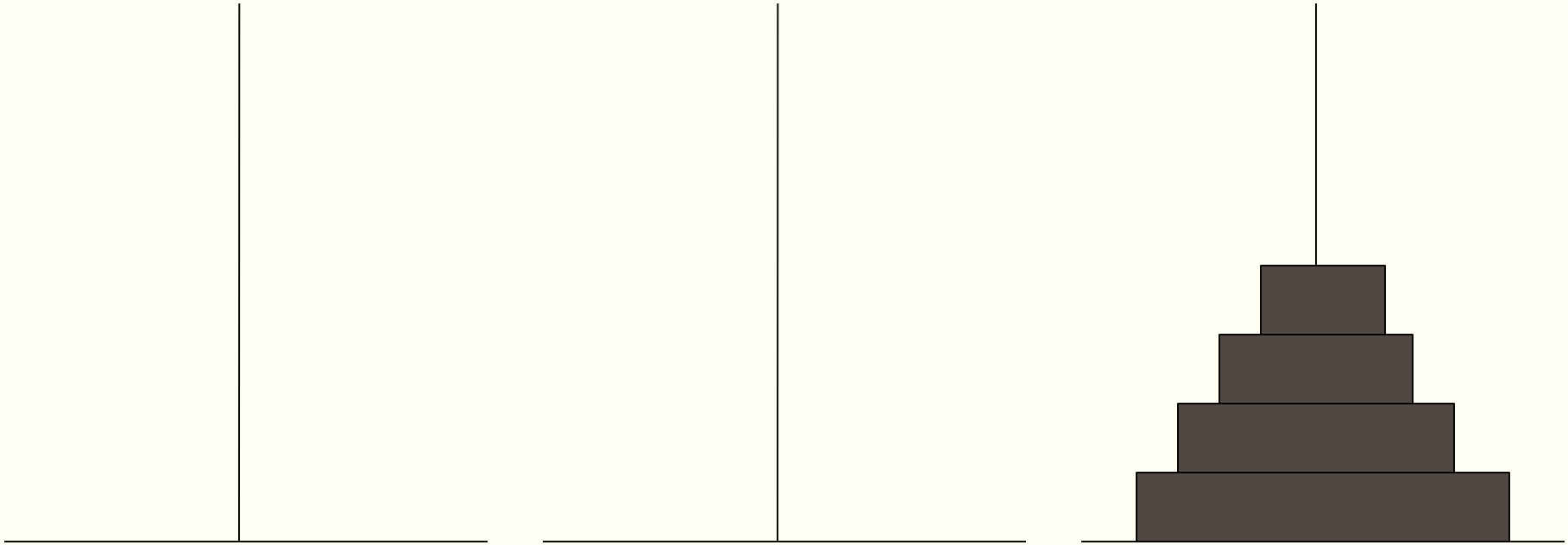
Bài toán Tháp Hà Nội



Bài toán Tháp Hà Nội



Bài toán Tháp Hà Nội



Bài toán tháp Hà nội - Thuật toán

- Sử dụng thêm một cột C làm trung gian
- Ta chuyển n-1 đĩa bên trên tại cột A sang cột C
- Chuyển 1 đĩa ở dưới cùng tại cột A sang cột B
- Sau đó ta chuyển n-1 bên trên tại cột C sang cột A
- Chuyển 1 đĩa ở dưới cùng từ cột C sang B
- …

Bài toán tháp Hà nội - Thuật toán (tiếp…)

- ChuyểnĐĩa(n,A,B,C) {Chuyển n đĩa từ cột A sang cột B, sử dụng cột trung gian C - chuyển n-1 đĩa từ A sang C}
  - ChuyểnĐĩa(n-1,A,B,C)
  - Chuyển1Đĩa(A,B)
  - ChuyểnĐĩa(n-1,C,B,A)

Bài toán tháp Hà nội - Thuật toán (tiếp…)

```
1  // Chuong trinh hien thi cach chuyen n dia duoc sap xep theo thu tu o cot 1 sang cot2
2  // su dung cot3 lam cot trung gian.
3  #include<iostream>
4  using namespace std;
5  // Ham de quy chuyen n dia tu cot1 sang cot 2 su dung cot3 lam cot trung gian
6  void HanoiTower(int cot1,int cot2,int cot3, int n) {
7      // neu co mot dia thi chuyet dia tu cot1 sang cot2
8      if (n==1)
9          cout<<"Chuyen dia tu "<<cot1<<" sang "<<cot2<<endl;
10     else {
11         // chuyen n-1 dia tu cot1 sang cot 3
12         HanoiTower(cot1,cot3,cot2,n-1);
13         // chuyen dia con lai tu cot1 sang cot2
14         cout<<"Chuyen dia tu "<<cot1<<" sang "<<cot2<<endl;
15         // chuyen n-1 tu cot3 sang cot2
16         HanoiTower(cot3,cot2,cot1,n-1);
17     }
18 }
19 // Ham main su dung ham HanoiTower de chuyen 4 dia tu cot1 sang cot 2
20 int main() {
21
22     int n;
23     HanoiTower(1,2,3,4);
24
25
26     system("pause");
27     return 0;
28 }
```

Khử đệ qui

- Thay thế lời gọi đệ qui trong chương trình bằng các lệnh khác để giải quyết bài toán.
- VD: Khử đệ qui bài tính n!

```
function GiaiThua(n: integer): integer;
Var T,i:integer;
Begin
 T=1;
 if (n>0) then
   for i=0 to n do T = T*i;
 GiaiThua =T;
End;
```

Khử đệ qui (tiếp…)

```
1  //Chương trình khử đệ qui tính số hạng fibo thứ n
2  #include <iostream>
3  using namespace std;
4  int fibonaci(int n){
5      int i,a,b,kq;
6      if((n==0)||(n==1))
7          kq=1;
8      else {
9          i=1;a=0;b=1;
0          while(i<=n){
1              i++;
2              a=a+b;
3              b=a-b;
4          }
5          kq=a;
6      }
7      return kq;
8  }
9  int main(){
0      int n;
1      cout << "Moi ban nhap so n" <<endl;
2      cin >> n;
3      cout <<"So fibonaci thu n la " <<fibonaci(n) <<endl;
4      return 0;
5  }
```

Bài tập

1. Viết chương trình sử dụng đệ qui để tìm USCLN của hai số tự nhiên a,b.
2. Viết chương trình sử dụng đệ qui để tính
   C(n,k) = n! / (n-k)!k!
   Với:
   C(n,n)=1
   C(n,0)=1
   C(n,k) = C(n-1,k-1) +C(n-1,k) 0<k<n
3. Viết thủ tục đệ qui thực hiện in ngược một dòng ký tự cho trước. (vd: "PASCAL" thì in ra "LACSAP")

Tổng kết

- Kỹ thuật đệ qui giúp cài đặt chương trình ngắn gọn, dễ hiểu phù hợp với các bài toán có tính chất đệ qui
- Kỹ thuật đệ qui được sử dụng để cài đặt cho nhiều thuật toán
- Để triển khai kỹ thuật đệ qui cần:
  - Điều kiện thoát đệ qui
  - Công thức đệ qui

Tiếp theo…

- Cấu trúc dữ liệu