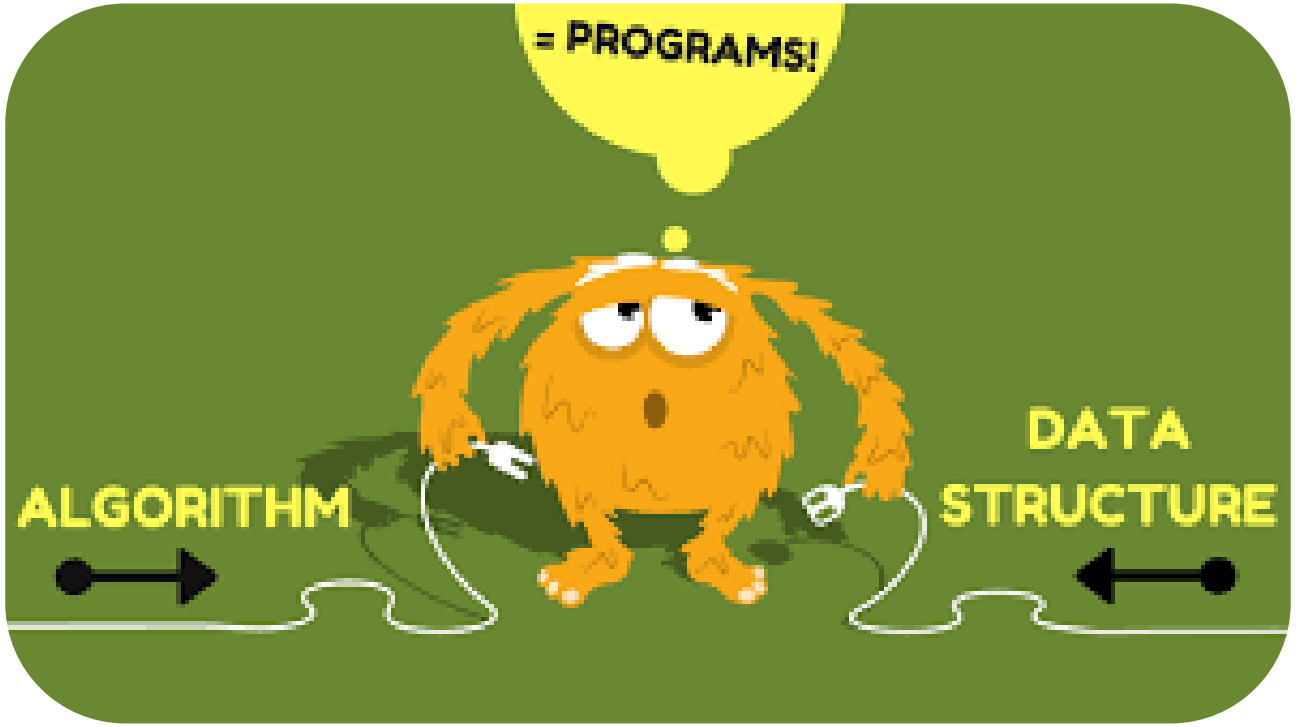
GIẢI THUẬT ĐỆ QUY VÀ ỨNG DỤNG



MỤC TIÊU BÀI HỌC

- Tìm hiểu khái niệm đệ qui.
- Xây dựng thủ tục đệ qui.
- Các ví dụ minh họa.

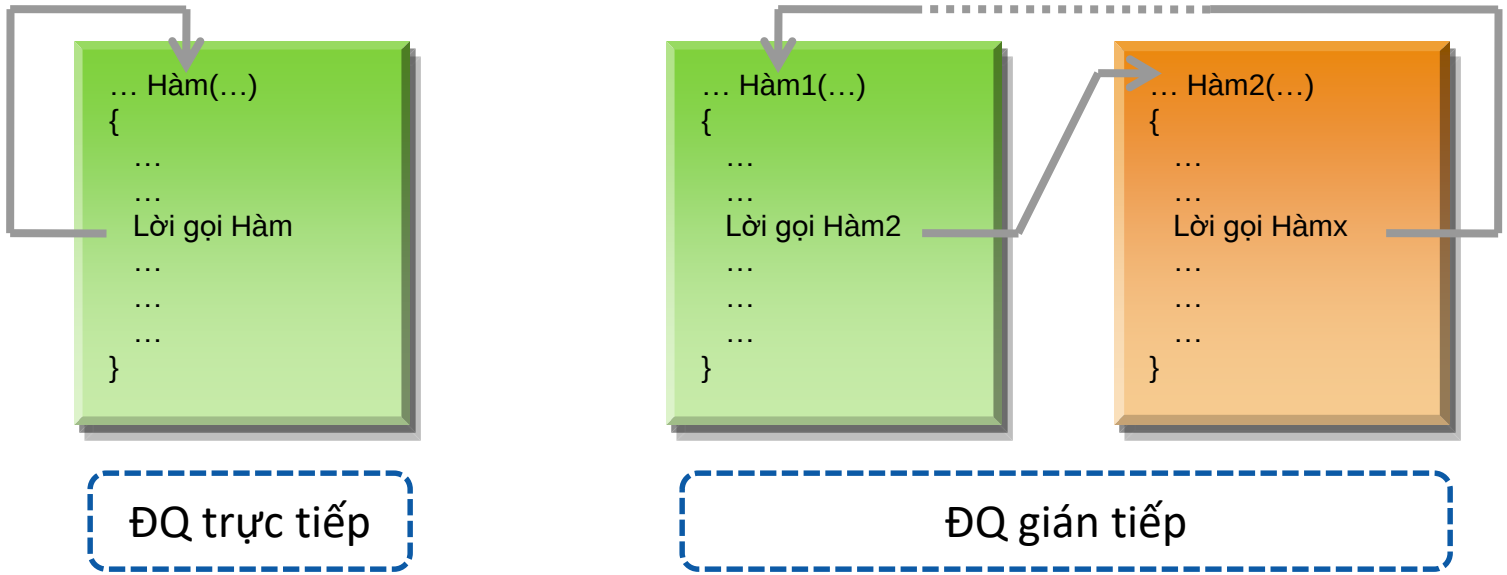
NỘI DUNG BÀI HỌC

1. Sự đệ quy
2. Giải thuật đệ quy
3. Xây dựng thủ tục (hàm) đệ quy
4. Các ví dụ minh họa đệ quy

1 GIỚI THIỆU

➢ Khái niệm đệ quy:

✓ Một hàm được gọi là đệ quy nếu bên trong thân của hàm đó có lời gọi hàm lại chính nó một cách trực tiếp hay gián tiếp.



KHÁI NIỆM ĐỆ QUY

➢Các qui tắc cơ bản:

✓String:

- Quy tắc 1: 1 char = String

✓Số tự nhiên:

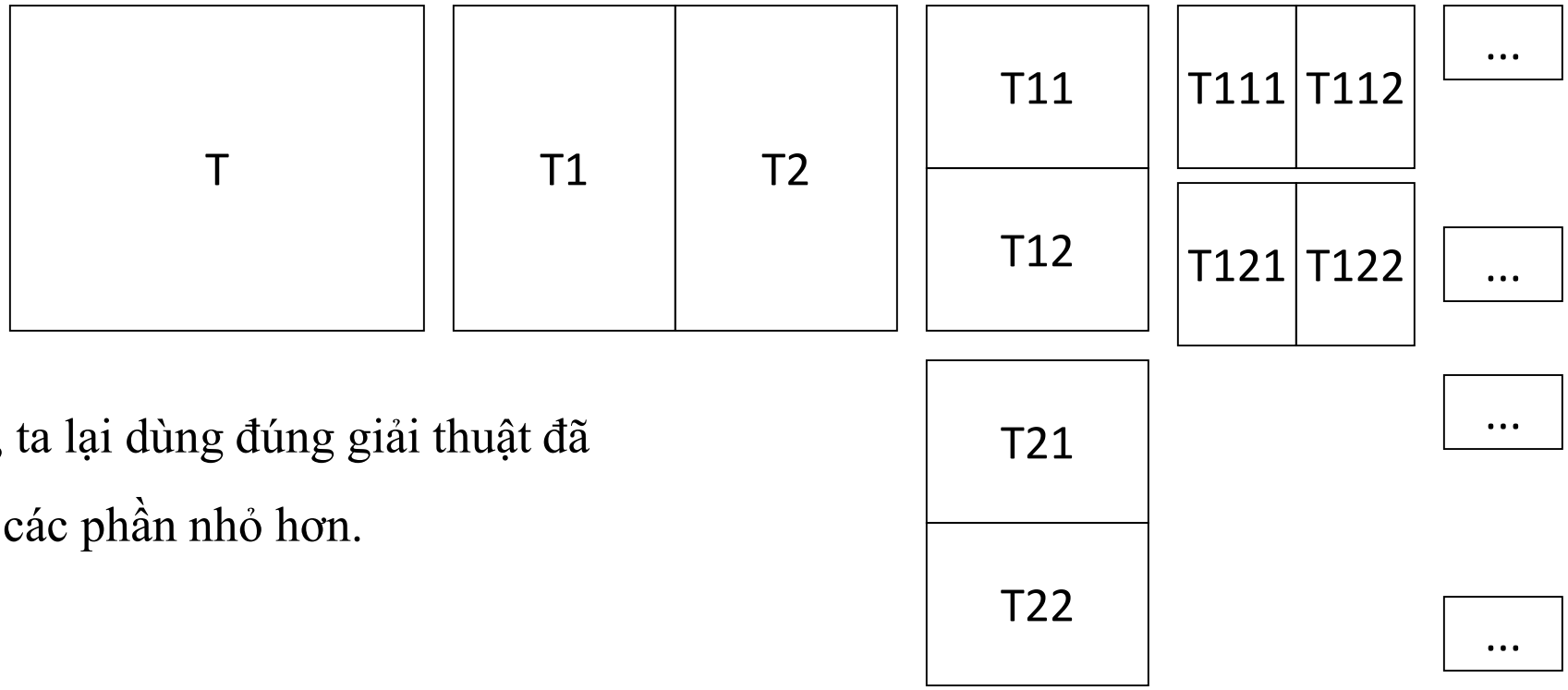
- Quy tắc 1: 1 ∈ N
- Quy tắc 2: x ∈ N if (x-1) ∈ N

✓N!

- Quy tắc 1: 0! = 1
- Quy tắc 2: n! = n (n-1)!

➢ Các ví dụ:

✓ **Ví dụ 1:** Tìm phần tử trong một tập hợp (tìm từ trong từ điển)



✓ **Nhận xét:**

– Sau mỗi lần tách đôi, ta lại dùng đúng giải thuật đã dùng để áp dụng với các phần nhỏ hơn.

✓ **Ví dụ 2:** Tính giá trị dãy số Fibonacci

- F (1) = 1 với n <= 2;
- F (n) = F (n-1) + F (n-2) với n > 1;

✓ **Nhận xét:**

- Thay vì tính F (n) ta quy về tính các giá trị hàm F với biến n nhỏ hơn là F(n-1) và F(n-2).
- Cách tính các hàm F(n-1), F(n-2) này cũng giống như cách tính F(n), có nghĩa là cũng quy về việc tính các hàm F với biến n nhỏ hơn nữa
- Trường hợp đặc biệt: khi n đủ nhỏ (n <= 2) ta tính được trực tiếp giá trị của hàm F (1) = 1, F (2) = 1

✓**Ví dụ 3:** Tìm số nhỏ nhất trong một dãy N số a_1, a_2, …, a_N

- 1) Nếu N=1 thì min=a_1;
- 2) Chia dãy ban đầu thành hai dãy con:

  $L_1 = a_1, a_2, \ldots, a_m$ và $L_2 = a_{m+1}, a_{m+2}, \ldots, a_N$

  với m=(1+N) DIV 2.

  Tìm min_1 trong dãy L_1 và min_2 trong dãy L_2.

  So sánh min_1 và min_2 để tìm ra min của dãy ban đầu.

✓Lưu ý:

- Trong trường hợp khái quát hơn, từ bài toán ban đầu ta có thể phải đưa về nhiều bài toán con tương tự.
- Trong trường hợp không có trường hợp đặc biệt, giải thuật thường dễ rơi vào vòng lặp vô hạn (không có tính dừng) và bài toán không thể giải được. Ví dụ vòng lặp vô hạn.

2

GIẢI THUẬT ĐỆ QUY

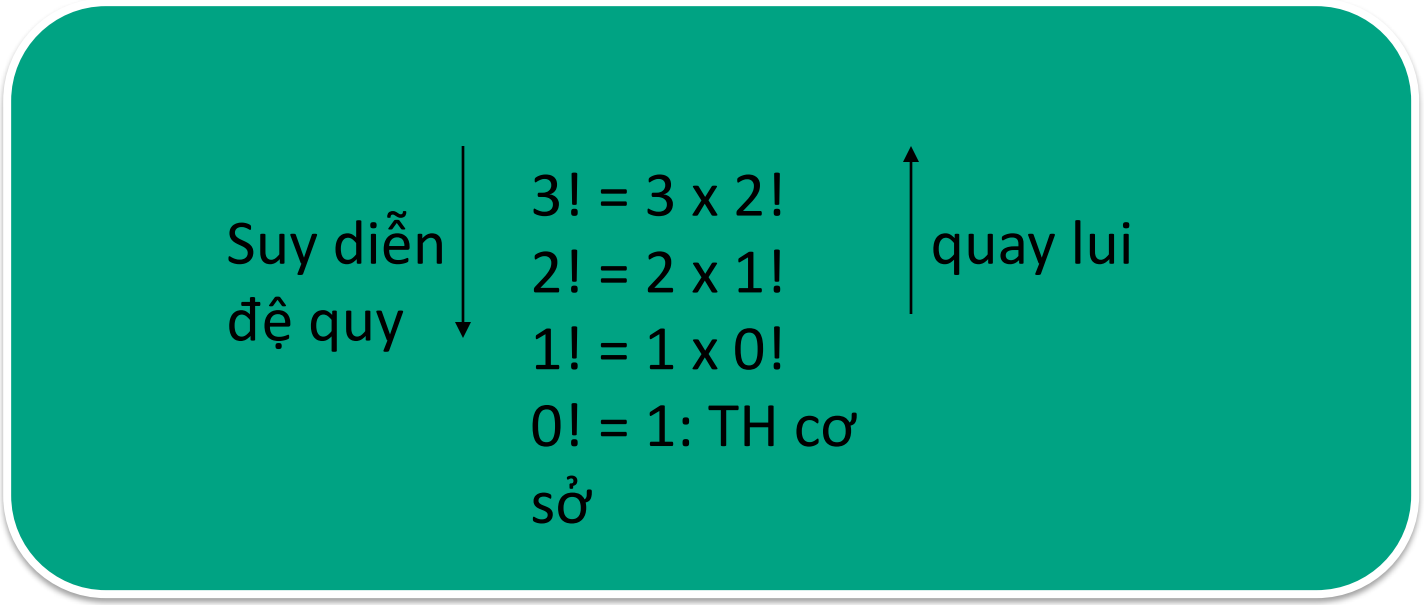
➢Cấu tạo giải thuật đệ quy: gồm hai thành phần sau:

- *Trường hợp cơ sở:* là trường hợp bài toán Pn, có quy mô đủ nhỏ để ta có thể giải trực tiếp. Nó đóng vai trò điểm dừng trong quá trình suy diễn đệ quy, và cũng quyết định tính dừng của giải thuật đệ quy.
- *Trường hợp đệ quy:* là trường hợp khái quát chứa cơ chế đệ quy, là cơ chế đưa bài toán cần giải về một hay nhiều bài toán tương tự nhưng có quy mô nhỏ hơn.
- Hai trường hợp trên có mối quan hệ khăng khít để tạo thành giải thuật đệ quy và đảm bảo giải thuật đệ quy có thể giải được và có điểm dừng.

➢Hoạt động của giải thuật bao gồm hai quá trình:

- *Quá trình suy diễn đệ quy:* là quá trình thu nhỏ bài toán ban đầu về các bài toán trung gian tương tự nhưng có quy mô giảm dần bằng cách áp dụng cơ chế đệ quy, cho đến khi gặp trường hợp cơ sở (điểm dừng của quá trình suy diễn đệ quy).
- *Quá trình quay lui (hay suy diễn ngược):* là quá trình từ kết quả thu được trong trường hợp cơ sở, thực hiện giải các bài toán trung gian ở quá trình suy diễn đệ quy *theo trật tự ngược lại*, cho đến khi giải quyết được bài toán ban đầu.

- **Ví dụ 5**: tính 3! theo giải thuật đệ quy



3

XÂY DỰNG THỦ TỤC ĐỆ QUY

XÂY DỰNG THỦ TỤC ĐỆ QUY

➢Thủ tục đệ quy

✓*Khái niệm:* để cài đặt các giải thuật đệ quy một cách đơn giản và hiệu quả, đa số các ngôn ngữ lập trình hiện nay đều có cơ chế, công cụ để hỗ trợ công việc này, đó là thủ tục đệ quy.

✓Thủ tục đệ quy là ***một chương trình con* trực tiếp** cài đặt cho giải thuật đệ quy. Tuỳ theo ngôn ngữ lập trình, nó có thể có các tên gọi khác nhau như: trong Pascal ta có thể có thủ tục đệ quy hay hàm con đệ quy, trong C/C++ chỉ có hàm con đệ quy, …

XÂY DỰNG THỦ TỤC ĐỆ QUY

✓Ví dụ 6: Tính giá trị n!

- 0! = 1
- n! = n (n-1)!

```
int Fact (int n){
        if  (n <= 1) return 1;
        else return n * Fact (n-1);
}
```

✓**Quay lại ví dụ 2:** Tính giá trị dãy Fibonacci

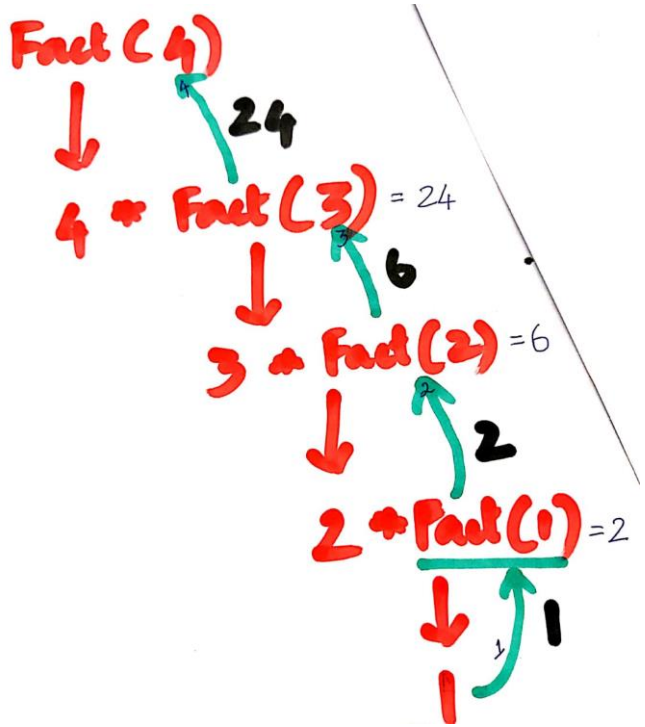
- F (1) = 1 với n <= 2;
- F (n) = F (n-1) + F (n-2) với n > 1;

```
int Fibo1 (int n) {
        if (n <= 2) return 1;
        else return (Fibo1 (n-1) + Fibo1 (n-2));
}
```

➢ **Nhận xét:** qua các ví dụ trên ta có thể rút ra các đặc điểm chung cho các thủ tục (hàm) đệ quy như sau:

+ Trong thủ tục đệ quy có lời gọi đến chính thủ tục đó.

+ Mỗi lần thực hiện gọi lại thủ tục thì kích thước của bài toán (tham số n) lại thu nhỏ hơn trước: bài toán được chia nhỏ



+ Trường hợp cơ sở - Trường hợp suy biến: các trường hợp mà ta có thể giải quyết bài toán một cách khác, một cách dễ dàng và hiển nhiên (ở 2 ví dụ trên, các hàm được tính một cách dễ dàng) và bài toán kết thúc.

+ Việc chia nhỏ dần bài toán đảm bảo dẫn tới tính trạng suy biến nói trên.

➢Thủ tục đệ quy – Phương pháp xây dựng

✓**Cấu trúc của thủ tục đệ quy**: Viết dưới dạng tựa C cấu trúc chung của một thủ tục đệ quy như sau:

```
void  P (A) {
   if A==A0 then
       Trường hợp cơ sở
   else {              //Trường hợp đệ quy
      Q1();
      P(A1);           //Lời gọi đệ quy
      Q2();
      P(A2);           //Lời gọi đệ quy
   }
}
```

Xây dựng thủ tục đệ quy

Nhận xét cấu trúc của thủ tục đệ quy:

✓ Phần đầu thủ tục: Thủ tục đệ quy luôn luôn phải có tham số.

Ngoài vai trò biểu diễn các thông tin vào/ra như các thủ tục thông thường không đệ quy, tham số của thủ tục đệ quy còn biểu diễn quy mô bài toán cần giải.

Tham số này phải có khả năng thu nhỏ khi gọi đệ quy và có quy mô nhỏ nhất khi đến trường hợp cơ sở.

XÂY DỰNG THỦ TỤC ĐỆ QUY

✓ Phần thân thủ tục: bao gồm hai nhánh của một cấu trúc rẽ nhánh tương ứng với hai trường hợp của giải thuật đệ quy:

- **Trường hợp cơ sở:** khi quy mô bài toán suy biến đủ nhỏ, bài toán có thể được giải trực tiếp, nên trong trường hợp này sẽ chứa các lệnh thi hành việc giải trực tiếp này.
- **Trường hợp đệ quy:** nó chứa một hay nhiều lời gọi đệ quy, là lời gọi đến chính thủ tục đang xây dựng, nhưng có các tham số khác với (mà thường là nhỏ hơn) tham số ban đầu.

XÂY DỰNG THỦ TỤC ĐỆ QUY

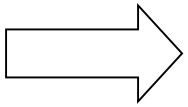
✓ Thủ tục đệ quy: gọi đến chính nó

✓ Trong một số trường hợp, việc gọi đến chính nó có thể được thực hiện qua hai cách:

- Gọi trực tiếp (2 ví dụ trên): ta có Đệ quy trực tiếp ***Ví dụ: xem ví dụ 6, ví dụ 2***
- Gọi gián tiếp: gọi tới nó thông qua việc ta gọi đến một thủ tục (hàm) khác và tại đó, ta mới gọi đến thủ tục (hàm) đệ quy. Ta có Đệ quy gián tiếp (***indirectly recursive***)

Ví dụ:

```
void RecursiveA () {
        ...
        ProcedureB();
        ...
}
```



```
void ProcedureB (){
        ...
        RecursiveA();
        ...
}
```

➢Thủ tục đệ quy - Hoạt động của thủ tục đệ quy

Quá trình hoạt động của thủ tục đệ quy gồm hai giai đoạn:

✓Giai đoạn gọi đệ quy:

- Bắt đầu từ lời gọi thủ tục đầu tiên, CT sẽ đi theo nhánh gọi đệ quy trong thân thủ tục để liên tục gọi các lời gọi đệ quy trung gian cho đến khi gặp trường hợp cơ sở.
- Khi đến điểm dừng này, hệ thống sẽ tự động kích hoạt giai đoạn thứ hai là giai đoạn quay lui.

XÂY DỰNG THỦ TỤC ĐỆ QUY

✓Giai đoạn quay lui:

- Hệ thống sẽ thi hành các thủ tục đệ quy trung gian trong giai đoạn đầu theo thứ tự ngược lại, cho đến khi thi hành xong thủ tục được gọi đầu tiên thì kết thúc, đồng thời kết thúc hoạt động của thủ tục đệ quy.

✓Vấn đề cài đặt: làm thế nào hệ thống có thể lưu giữ các lời gọi đệ quy trung gian và các kết quả xử lý trung gian trong giai đoạn gọi đệ quy để ta có thể lấy chúng ra để xử lý trong giai đoạn quay lui.

➢Thủ tục đệ quy - Nguyên tắc cài đặt thủ tục đệ quy

Nhận xét: việc lưu trữ, xử lý các lời gọi đệ quy có đặc điểm sau:

✓Tính vào sau ra trước (LIFO):

- Việc lưu trữ các lời gọi theo đúng thứ tự của quá trình gọi đệ quy, lời gọi trước lưu trữ trước và ngược lại.
- Trong quá trình quay lui, các lời gọi được lấy ra theo thứ ngược lại để xử lý. Vậy cần sử dụng cấu trúc ngăn xếp để lưu trữ các lời gọi đệ quy.

XÂY DỰNG THỦ TỤC ĐỆ QUY

✓Tính đồng nhất:

- Do cấu trúc các lời gọi đệ quy và các kết quả trung gian giống nhau, nên cấu trúc ngăn xếp được sử dụng cũng có tính đồng nhất.

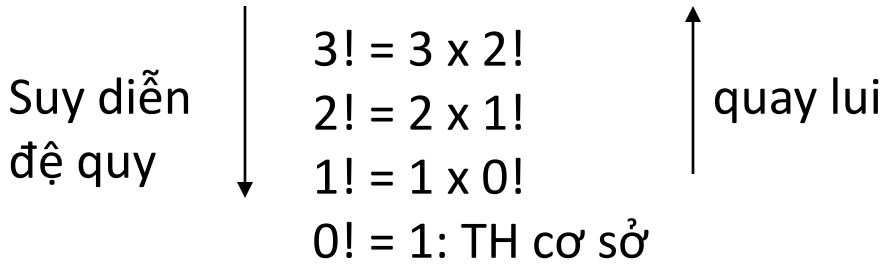
✓Trong thực tế

- Các ngôn ngữ lập trình có hỗ trợ cài đặt thủ tục đệ quy đều đã tự động cài đặt các cấu trúc ngăn xếp thích hợp để phục vụ cho quá trình cài đặt và hoạt động của thủ tục đệ quy.

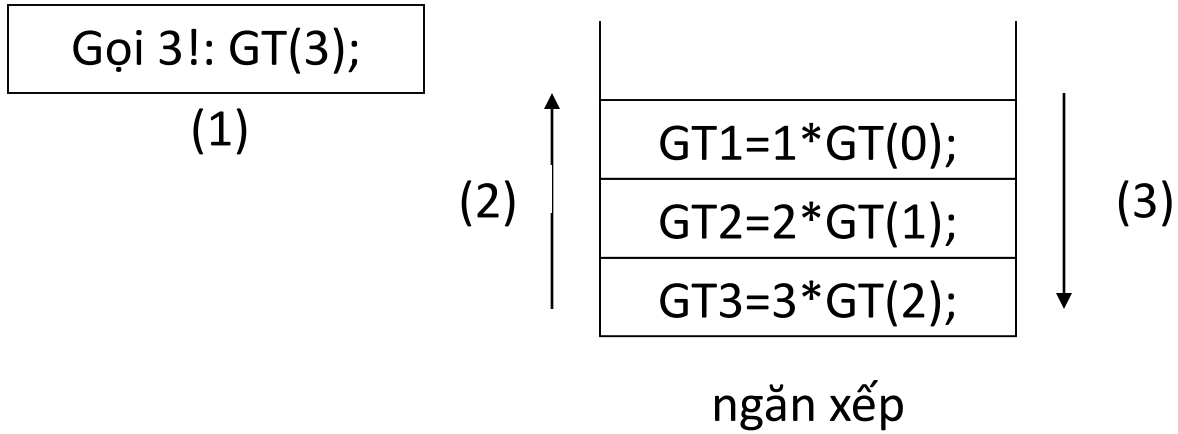
Xây dựng thủ tục đệ quy

✓ Ví dụ: minh hoạ hoạt động của thủ tục đệ quy tính 3!.

Giải thuật đệ quy



Hoạt động của thủ tục đệ quy



➢Thủ tục đệ quy – Ưu nhược điểm

✓**Ưu điểm**: viết chương trình dễ dàng, dễ hiểu, ngắn gọn

✓**Nhược điểm**: theo nguyên tắc cài đặt của thủ tục đệ quy, có thể thấy các nhược điểm sau:

- Thời gian thực hiện: tốn thời gian
- Bộ nhớ: tốn bộ nhớ

XÂY DỰNG THỦ TỤC ĐỆ QUY

✓**Các vấn đề khác:** không phải lúc nào cũng có thể xây dựng bài toán theo giải thuật và thủ tục đệ quy một cách dễ dàng, các vấn đề có thể là:

- Có thể định nghĩa bài toán dưới dạng bài toán cùng loại nhưng nhỏ hơn như thế nào?
- Làm thế nào để đảm bảo kích thước bài toán giảm đi sau mỗi lần gọi?
- Xem xét và định nghĩa các trường hợp đặc biệt (trường hợp suy biến) như thế nào?

➢Khái niệm

✓Đưa các bài toán đệ quy về các bài toán không sử dụng đệ quy,đó là ***sự khử đệ quy.***

✓Sự khử đệ quy thông qua các vòng lặp (for, while) và phân nhánh (if...then...else)

✓**Ví dụ 1:**

```
int Fact (int n){
        if  (n <= 1) return 1;
        else return n * Fact (n-1);
}
```

```
int Fact (int n){
   if  (n <= 1) return 1;
   else {
      int x=1;
      for (int i=2;i<=n;i++) x*=i;
      return x;
   }
}
```

✓Ví dụ 2:

```
int Fibo1 (int n){
    if (n <= 2) return 1;
    else
      return (Fibo1 (n-1) + Fibo1 (n-2));
}
```

```
int  Fibo2 (int n) {
    int i, f, f1, f2;
    f1 = 1 ;
    f2 = 1 ;
    if  (n <= 2)  f = 1;
    else for (i = 3;i<=n;i++){
              f = f1 + f2;
              f1 = f2;
              f2 = f;
    }
    return f;
}
```

4

CÁC VÍ DỤ MINH HỌA

CÁC VÍ DỤ MINH HỌA

➢Tìm kiếm trong danh sách liên kết:

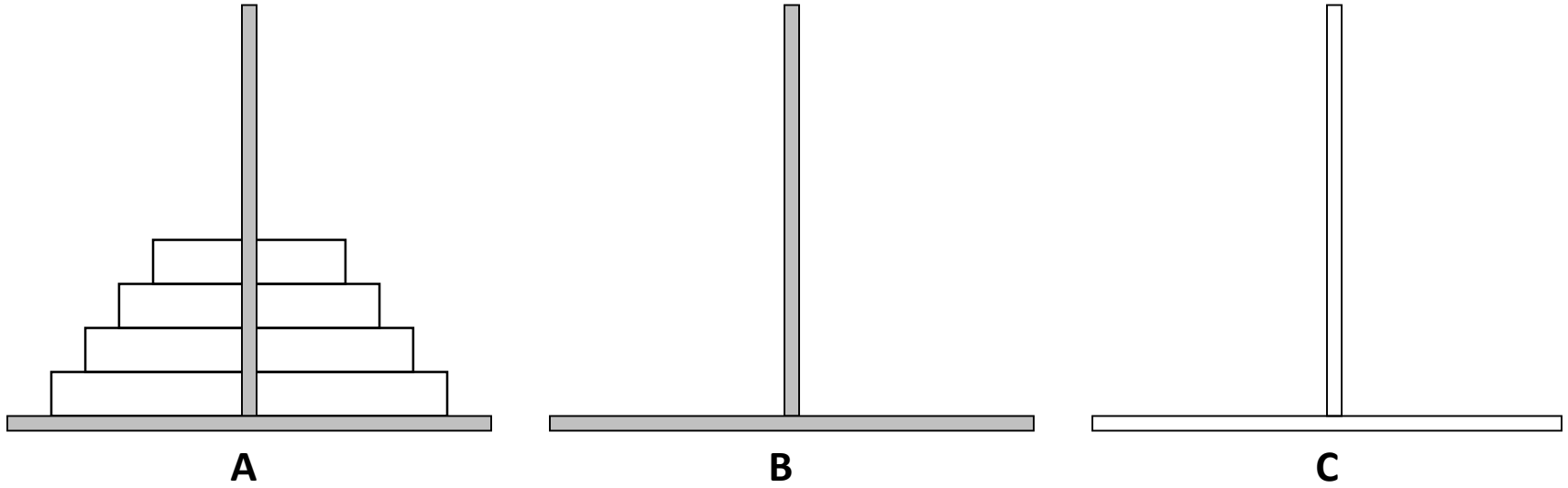
✓Tìm kiếm một phần tử x trong danh sách có H là con trỏ hiện đang trỏ đến nút đầu tiên. Hàm trả về con trỏ trỏ vào nút tìm thấy; Trái lại nếu không tìm thấy thì trả về NULL.

```
PNode Search (Item x, PNode H) {
    if (H == NULL) return NULL;
    else
        if  (H->info == x) return H;
        else  return Search (x; H->next) ;
}
```

➢Bài toán Tháp Hà Nội

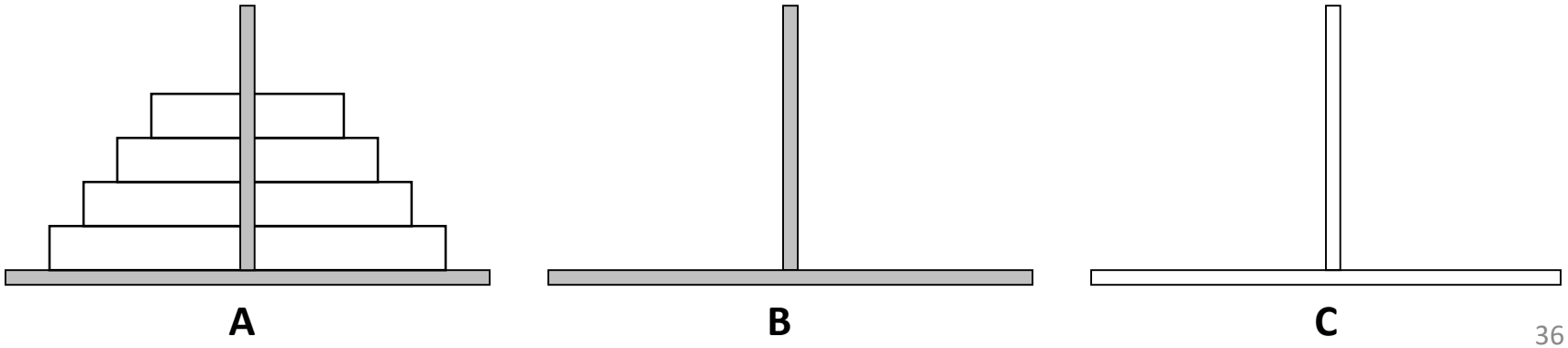
✓Bài toán: chuyển cột đĩa từ A sang B

- Mỗi lần chỉ được chuyển 1đĩa
- Không được đĩa to trên đĩa nhỏ, dù chỉ tạm thời
- Được phép chuyển qua một cột trung gian C



Bài toán Tháp Hà Nội

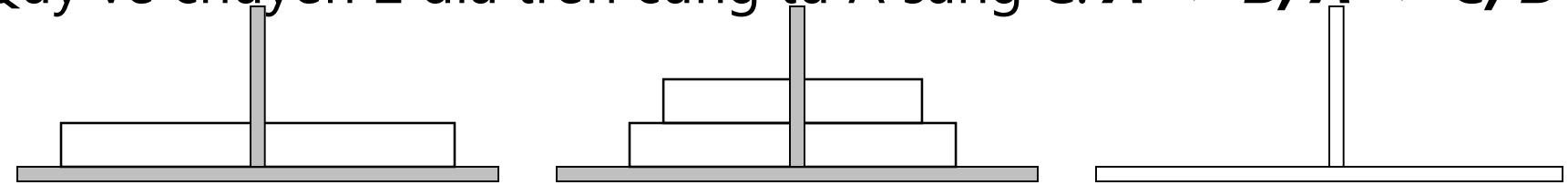
- ✓Xác định trường hợp đơn giản (suy biến)
  - 1 đĩa: A -> B
  - 2 đĩa: A -> C, A -> B, C -> B
- ✓Tổng quát: quy về trường hợp 1 hoặc 2 đĩa:
  - 1, Quy về bài toán chuyển (n-1) đĩa A sang C
  - 2, Quy về bài toán chuyển 1 đĩa A sang B
  - 3, Quy về bài toán chuyển (n-1) đĩa C sang B



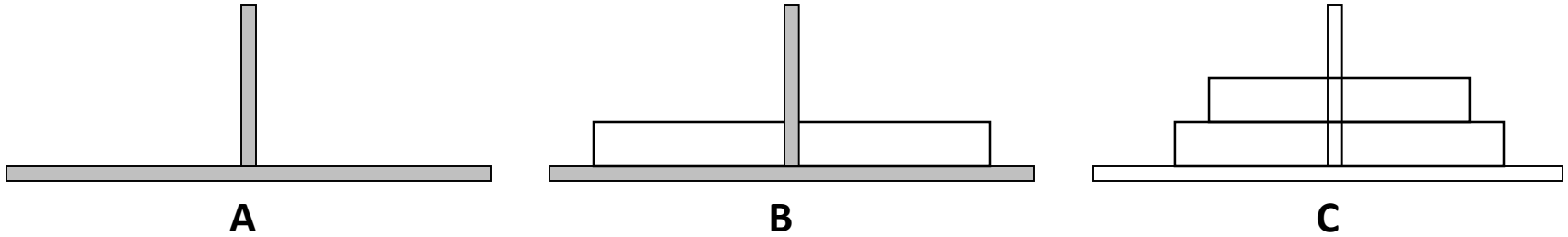
➢Bài toán Tháp Hà Nội

✓Ví dụ 3 đĩa

- 1, Quy về chuyển 2 đĩa trên cùng từ A sang C: ***A -> B, A -> C, B -> C***



- 2, Quy về chuyển 1 đĩa từ A sang B: ***A -> B***



- 3, Quy về chuyển 2 đĩa từ C sang B: ***C -> A, C -> B, A -> B***

➢Bài toán Tháp Hà Nội

✓Trường hợp đơn giản (suy biến): 1 đĩa: A -> B

✓Tổng quát: quy về trường hợp 1 hoặc 2 đĩa:

- 1, Quy về bài toán chuyển (n-1) đĩa A sang C
- 2, Quy về bài toán chuyển 1 đĩa A sang B
- 3, Quy về bài toán chuyển (n-1) đĩa C sang B

```
void  TowerHN (int n, Tower A, B, C){
    if (n == 1) Transfer (A, B);
    else {
                    TowerHN (n-1, A, C, B);
                    TowerHN (1, A, B, C);
                    TowerHN (n-1, C, B, A);}
}
```

TỔNG KẾT

- Hàm đệ quy là hàm mà trong thân hàm lại gọi chính nó.
- Hàm đệ quy kém hiệu quả vì:
  - ✓ Tốn bộ nhớ.
  - ✓ Gọi hàm nhiều lần.
- Tuy nhiên viết hàm đệ quy rất ngắn gọn.


感謝你
Danke
Gracias
Grazie
Merci
Thanks!
THANK
YOU!


VAIO