# Huấn luyện tư duy thuật toán

## Bài toán: Print from 1 to N (Recursion)

Slide Learning C++

Ngày 16 tháng 1 năm 2026

# Tiếp nhận & Phẫu thuật (Briefing)

## Tóm tắt đề bài

Máy tính cho bạn một số nguyên dương $N$. Nhiệm vụ của bạn là in ra các số từ 1 đến $N$, mỗi số nằm trên một dòng riêng biệt.

## Yêu cầu bắt buộc

Phải sử dụng **Đệ quy (Recursion)** để giải quyết.

**Lộ trình tư duy:**

1. **Chunk 1:** Hiểu về đệ quy qua hình ảnh thực tế.
2. **Chunk 2:** Xác định điểm dừng (Base case).
3. **Chunk 3:** Quy luật "Gửi thông điệp"và thứ tự thực hiện.

# Chunk 1: Đệ quy là gì? (Ẩn dụ "Búp bê Nga")

- Tưởng tượng Đệ quy giống như bộ **Búp bê Nga (Matryoshka)**.
- Khi mở một con búp bê lớn, bên trong lại có một con búp bê nhỏ hơn y hệt nó.
- Quá trình tiếp diễn cho đến khi chạm đến con búp bê nhỏ nhất.

## Định nghĩa trong lập trình

Đệ quy là một hàm **tự gọi lại chính nó** nhưng với một phiên bản bài toán nhỏ hơn (giá trị truyền vào giảm dần).

**Ví dụ:** `in_so(5)` nhờ `in_so(4)` làm giúp phần việc phía trước, sau đó nó mới làm phần của mình.

# Thử thách tư duy: Điểm dừng

## Câu hỏi

Nếu chúng ta cứ "nhờ"mãi như vậy (`in_so(4)` nhờ `in_so(3)`,...), đến số mấy thì chúng ta phải dừng lại để tránh việc máy tính chạy mãi không nghỉ?

# Thử thách tư duy: Điểm dừng

## Câu hỏi

Nếu chúng ta cứ "nhờ"mãi như vậy (`in_so(4)` nhờ `in_so(3)`,...), đến số mấy thì chúng ta phải dừng lại để tránh việc máy tính chạy mãi không nghỉ?

## Giải đáp: Điểm dừng (Base case)

Điểm dừng quan trọng nhất chính là **số 1** (hoặc số 0 tùy cách cài đặt).

- Vì đề bài yêu cầu in từ 1 đến $N$.
- Nếu tiếp tục lùi xuống các số âm, chúng ta sẽ làm sai yêu cầu đề bài.

# Chunk 2: Quy luật "Gửi thông điệp"

Có hai cách để chúng ta sắp xếp công việc trong đệ quy:

### 1. Làm xong rồi mới gọi

In số hiện tại ra, sau đó mới nhờ "đệ tử"tiếp theo làm phần còn lại.

### 2. Gọi xong mới làm

Nhờ "đệ tử"làm hết mọi việc phía trước, khi nào xong mới quay lại in số của mình.

**Trắc nghiệm:** Để in theo thứ tự tăng dần ($1 \to N$), ta chọn cách nào?

- **A.** In số $N$ trước, sau đó mới gọi `print(N-1)`.
- **B.** Gọi `print(N-1)` trước, sau đó mới in số $N$.

# Chunk 2: Quy luật "Gửi thông điệp"

Có hai cách để chúng ta sắp xếp công việc trong đệ quy:

### 1. Làm xong rồi mới gọi

In số hiện tại ra, sau đó mới nhờ "đệ tử"tiếp theo làm phần còn lại.

### 2. Gọi xong mới làm

Nhờ "đệ tử"làm hết mọi việc phía trước, khi nào xong mới quay lại in số của mình.

**Trắc nghiệm:** Để in theo thứ tự tăng dần ($1 \to N$), ta chọn cách nào?

- **A.** In số $N$ trước, sau đó mới gọi `print(N-1)`.
- **B.** Gọi `print(N-1)` trước, sau đó mới in số $N$.

**Đáp án: B** - Vì ta muốn các số nhỏ hơn hiện ra trước.

# Phân tích luồng thực thi

Cấu trúc của hàm đệ quy:

1. **Kiểm tra điểm dừng:** Nếu $N < 1$, thoát ra.
2. **Gọi đệ quy:** Nhờ hàm giải quyết bài toán với $N - 1$.
3. **Thực hiện công việc:** In số $N$ ra màn hình.

## Diễn biến khi $N = 3$

- `print(3)` gọi `print(2)`
- `print(2)` gọi `print(1)`
- `print(1)` gọi `print(0)` $\rightarrow$ **Dừng!**
- Quay lại `print(1)`: In ra **1**
- Quay lại `print(2)`: In ra **2**
- Quay lại `print(3)`: In ra **3**

# Cơ chế Ngăn xếp (Stack)

- **Giai đoạn "Vào"(Calling phase):** Máy tính "tạm dừng"các hàm và xếp chúng vào một chiếc hộp (Stack).
- **Giai đoạn "Ra"(Returning phase):** Khi chạm điểm dừng, máy tính lấy các hàm ra theo thứ tự ngược lại (từ trên xuống).

## Ghi nhớ

Lệnh in đặt **sau** lời gọi đệ quy sẽ được thực hiện trong giai đoạn "Ra", tạo nên thứ tự tăng dần.

# Chunk 3: Chốt thuật toán

Dưới đây là cấu trúc logic (Mã giả) chúng ta đã xây dựng:

```
H  m  Print_Numbers(N):
    // 1. Diem dung (Base case)
    Neu N == 0:
        Ket thuc

    // 2. Goi de quy truoc de in cac so nho hon
    Print_Numbers(N - 1)

    // 3. Thuc hien in so hien tai (Giai doan "Ra")
    In N ra man hinh
```

Listing 1: Mã giả cho bài toán Print 1 to N

**Kết quả với** $N = 5$**:** 1 2 3 4 5 (mỗi số một dòng).

# Thử thách mở rộng

## Câu hỏi mở rộng

Nếu bài toán yêu cầu in **ngược lại** từ $N$ về 1 $(5, 4, 3, 2, 1)$, bạn sẽ thay đổi vị trí của lệnh **In N** như thế nào?

- **A.** Giữ nguyên (In $N$ sau khi gọi đệ quy).
- **B.** Đảo lên trên (In $N$ trước khi gọi đệ quy).

# Thử thách mở rộng

## Câu hỏi mở rộng

Nếu bài toán yêu cầu in **ngược lại** từ $N$ về 1 $(5, 4, 3, 2, 1)$, bạn sẽ thay đổi vị trí của lệnh **In N** như thế nào?

- **A.** Giữ nguyên (In $N$ sau khi gọi đệ quy).
- **B.** Đảo lên trên (In $N$ trước khi gọi đệ quy).

## Giải đáp

**Đáp án B:** In trước khi gọi đệ quy sẽ thực hiện lệnh in ngay trong giai đoạn "Vào", giúp in từ số lớn đến số bé.