A. Print Recursion

Slide Learning C++

2026

Bước 1: Tiếp nhận & Phẫu thuật (Briefing)

Yêu cầu cốt lõi

- **Đầu vào (Input):** Một số nguyên N ($1 \leq N \leq 100$).
- **Nhiệm vụ:** In ra dòng chữ `"I love Recursion"` đúng N lần.
- **Điều kiện bắt buộc:** Phải sử dụng **Đệ quy (Recursion)**, không dùng vòng lặp.

- **Chunk 1:** Hiểu về Đệ quy qua hình ảnh đời thường.
- **Chunk 2:** Xác định Điểm dừng (Base case).
- **Chunk 3:** Xác định Bước gọi lại (Recursive step).

Chunk 1: Đệ quy là gì? (Metaphor)

Tưởng tượng

Bạn có xấp N tờ giấy trắng. Quy trình:

- Nếu còn giấy: In dòng chữ, rồi **đưa xấp còn lại cho chính mình** làm tiếp.
- Nếu hết giấy: Dừng lại.

Định nghĩa

Đệ quy là việc một hàm **tự gọi lại chính nó** với một phiên bản "nhỏ hơn"của vấn đề ban đầu.

Thử thách tư duy 1

Nếu bắt đầu với $N = 3$:

1. Lần 1: In 1 dòng, còn lại $N = 2$. Gọi lại chính mình.
2. Lần 2: In thêm 1 dòng, còn lại $N = 1$. Gọi lại chính mình.
3. Lần 3: In thêm 1 dòng, còn lại $N = 0$.

Câu hỏi: Tại $N = 0$, bạn có in thêm dòng nào không?

Thử thách tư duy 1

Nếu bắt đầu với $N = 3$:

1. Lần 1: In 1 dòng, còn lại $N = 2$. Gọi lại chính mình.
2. Lần 2: In thêm 1 dòng, còn lại $N = 1$. Gọi lại chính mình.
3. Lần 3: In thêm 1 dòng, còn lại $N = 0$.

Câu hỏi: Tại $N = 0$, bạn có in thêm dòng nào không?

Kết quả

Chính xác! Khi $N = 0$, chúng ta kết thúc. Đây là **Điểm dừng (Base case)** để ngăn máy tính chạy vô tận.

Chunk 2: Thiết kế hàm đệ quy

Một hàm đệ quy thường có 2 phần chính:

1. **Phần dừng (Base Case):** Nếu $N = 0$, thoát khỏi hàm.
2. **Phần thực thi & Gọi lại (Recursive Step):** In dòng chữ và gọi lại hàm với giá trị $N - 1$.

```
void solve(int n) {
    if (n == 0) return; // Diem dung

    // BUOC A: In dong chu "I love Recursion"
    // BUOC B: solve(n - 1);
}

```

Thử thách tư duy 2

Nếu đổi thứ tự: Đưa **BƯỚC B** lên trước **BƯỚC A** thì kết quả có thay đổi không?

- `solve(3)` gọi `solve(2)`...
- `solve(2)` gọi `solve(1)`...
- `solve(1)` gọi `solve(0)`...

Thử thách tư duy 2

Nếu đổi thứ tự: Đưa **BƯỚC B** lên trước **BƯỚC A** thì kết quả có thay đổi không?

- `solve(3)` gọi `solve(2)`...
- `solve(2)` gọi `solve(1)`...
- `solve(1)` gọi `solve(0)`...

Lưu ý

Thứ tự thực thi bị đảo lộn! Các lệnh in chỉ bắt đầu khi hàm "quay về"(backtracking). Tuy nhiên, vì các dòng in giống hệt nhau, bạn vẫn thấy đủ N dòng nhưng bản chất thực hiện đã khác.

Chunk 3: Tổng kết thuật toán

Mã giả (Pseudocode)

```
Hàm InDeQuy(số_lần):
    Nếu (số_lần == 0):
        Thoát (Base Case)

    In dòng chữ "I love Recursion"

    InDeQuy(số_lần - 1)

```

Bẫy logic

Luôn phải tiến về phía **Điểm dừng**. Nếu gọi `InDeQuy(N)` mà không giảm N, máy tính sẽ bị treo (Stack Overflow).

Trắc nghiệm cuối Chunk

Nếu truyền vào $N = 3$, hàm sẽ được **gọi tổng cộng bao nhiêu lần** (tính cả lần gọi đầu tiên và lần gọi tại điểm dừng)?

- A. 3 lần
- B. 4 lần
- C. 2 lần

Trắc nghiệm cuối Chunk

Nếu truyền vào $N = 3$, hàm sẽ được **gọi tổng cộng bao nhiêu lần** (tính cả lần gọi đầu tiên và lần gọi tại điểm dừng)?

- A. 3 lần
- B. 4 lần
- C. 2 lần

Đáp án: B. 4 lần

1. `solve(3)` $\rightarrow$ In dòng 1, gọi `solve(2)`.
2. `solve(2)` $\rightarrow$ In dòng 2, gọi `solve(1)`.
3. `solve(1)` $\rightarrow$ In dòng 3, gọi `solve(0)`.
4. `solve(0)` $\rightarrow$ Kiểm tra $N = 0$ nên `return`.

Kiến trúc chương trình hoàn chỉnh

```cpp
void printLove(int n) {
    if (n == 0) return; // 1. Base Case

    // 2. Thuc hien hanh dong
    cout << "I love Recursion" << endl;

    // 3. Goi de quy
    printLove(n - 1);
}

int main() {
    int n;
    cin >> n;
    printLove(n);
    return 0;
}

```

Thử thách cuối cùng

Trong C++, lệnh nào dùng để **thoát ngang** khỏi một hàm `void` ngay khi gặp điều kiện dừng?

- A. `break;`
- B. `return;`
- C. `exit(0);`

Thử thách cuối cùng

Trong C++, lệnh nào dùng để **thoát ngang** khỏi một hàm `void` ngay khi gặp điều kiện dừng?

- A. `break;`
- B. `return;`
- C. `exit(0);`

Đáp án: B. `return;`

Dùng để kết thúc thực thi hàm và quay lại nơi nó được gọi.

Tổng kết & Bước tiếp theo

Tóm tắt luồng chạy

1. **Main**: Gọi "nhân viên"đệ quy đầu tiên.
2. **Đệ quy**: Nếu $N = 0$ thì nghỉ. Nếu $N > 0$ thì làm việc rồi chuyển phần còn lại cho phiên bản sau.

- **Lựa chọn 1:** Tự tay viết Code dựa trên bản thiết kế.
- **Lựa chọn 2:** Thử thách in **ngược** (Dòng N trước, dòng 1 sau) bằng cách đổi vị trí 1 dòng code.