

(Phạm_Minh_Tú_DHKL17A2HN_23174600120(ca chiều))

Bài 1: Cho hàm đệ quy để tính tổng các số từ 1 đến n. Hãy giải thích từng bước thực hiện của hàm đệ quy này khi $n = 7$

```
def sum_of_numbers(n):  
    if n == 1:  
        return 1  
    else:  
        return n + sum_of_numbers(n-1)  
print(sum_of_numbers(7))
```

Bước 1: sum_of_numbers(7)

- n không bằng 1, nên đi vào nhánh else.
- Tính $7 + \text{sum_of_numbers}(6)$.

Bước 2: sum_of_numbers(6)

- n không bằng 1, nên đi vào nhánh else.
- Tính $6 + \text{sum_of_numbers}(5)$.

Bước 3: sum_of_numbers(5)

- n không bằng 1, nên đi vào nhánh else.
- Tính $5 + \text{sum_of_numbers}(4)$.

Bước 4: sum_of_numbers(4)

- n không bằng 1, nên đi vào nhánh else.
- Tính $4 + \text{sum_of_numbers}(3)$.

Bước 5: sum_of_numbers(3)

- n không bằng 1, nên đi vào nhánh else.
- Tính $3 + \text{sum_of_numbers}(2)$.

(Phạm_Minh_Tú_DHKL17A2HN_23174600120(ca chiều))

Bước 6: `sum_of_numbers(2)`

- `n` không bằng 1, nên đi vào nhánh `else`.
- Tính $2 + \text{sum_of_numbers}(1)$.

Bước 7: `sum_of_numbers(1)`

- `n` bằng 1, nên trả về 1.

Bây giờ, chúng ta quay lại các lời gọi hàm đệ quy và cộng dần kết quả trả về từ dưới lên:

`sum_of_numbers(1)` trả về 1.

`sum_of_numbers(2)` là $2 + \text{sum_of_numbers}(1) = 2 + 1 = 3$.

`sum_of_numbers(3)` là $3 + \text{sum_of_numbers}(2) = 3 + 3 = 6$.

`sum_of_numbers(4)` là $4 + \text{sum_of_numbers}(3) = 4 + 6 = 10$.

`sum_of_numbers(5)` là $5 + \text{sum_of_numbers}(4) = 5 + 10 = 15$.

`sum_of_numbers(6)` là $6 + \text{sum_of_numbers}(5) = 6 + 15 = 21$.

`sum_of_numbers(7)` là $7 + \text{sum_of_numbers}(6) = 7 + 21 = 28$.

=> **Kết quả**

- Do đó, khi chạy `print(sum_of_numbers(7))`, kết quả in ra sẽ là 28.

Bài 2: Cho hàm đệ quy để tính số Fibonacci thứ `n`. Hãy giải thích từng bước

thực hiện của hàm đệ quy này khi `n = 8`.

```
def fibonacci(n):  
    if n <= 1:  
        return n  
    else:  
        return fibonacci(n-1) + fibonacci(n-2)  
print(fibonacci(8))
```

Bước 1: Gọi `fibonacci(8)`:

(Phạm_Minh_Tú_DHKL17A2HN_23174600120(ca chiều))

- Gọi fibonacci(7)

- Gọi fibonacci(6)

Bước 2: Gọi fibonacci(7):

- Gọi fibonacci(6)

- Gọi fibonacci(5)

Bước 3: Gọi fibonacci(6):

- Gọi fibonacci(5)

- Gọi fibonacci(4)

Bước 4: Gọi fibonacci(5):

- Gọi fibonacci(4)

- Gọi fibonacci(3)

Bước 5: Gọi fibonacci(4):

- Gọi fibonacci(3)

- Gọi fibonacci(2)

Bước 6: Gọi fibonacci(3):

- Gọi fibonacci(2)

- Gọi fibonacci(1)

Bước 7: Gọi fibonacci(2):

- Gọi fibonacci(1) -> Trả về 1 (điều kiện cơ sở)

- Gọi fibonacci(0) -> Trả về 0 (điều kiện cơ sở)

Kết quả: fibonacci(2) = 1 + 0 = 1

(Phạm_Minh_Tú_DHKL17A2HN_23174600120(ca chiều))

Bước 8: Trở lại fibonacci(3) (tiếp tục từ bước 6):

- Đã có fibonacci(2) = 1 (từ bước 7)
- Gọi fibonacci(1) -> Trả về 1 (điều kiện cơ sở)

Kết quả: fibonacci(3) = 1 + 1 = 2

Bước 9: Trở lại fibonacci(4) (tiếp tục từ bước 5):

- Đã có fibonacci(3) = 2 (từ bước 8)
- Gọi fibonacci(2) -> Trả về 1 (từ bước 7)

Kết quả: fibonacci(4) = 2 + 1 = 3

Bước 10: Trở lại fibonacci(5) (tiếp tục từ bước 4):

- Đã có fibonacci(4) = 3 (từ bước 9)
- Gọi fibonacci(3) -> Trả về 2 (từ bước 8)

Kết quả: fibonacci(5) = 3 + 2 = 5

Bước 11: Trở lại fibonacci(6) (tiếp tục từ bước 3):

- Đã có fibonacci(5) = 5 (từ bước 10)
- Gọi fibonacci(4) -> Trả về 3 (từ bước 9)

Kết quả: fibonacci(6) = 5 + 3 = 8

Bước 12: Trở lại fibonacci(7) (tiếp tục từ bước 2):

- Đã có fibonacci(6) = 8 (từ bước 11)
- Gọi fibonacci(5) -> Trả về 5 (từ bước 10)

Kết quả: fibonacci(7) = 8 + 5 = 13

Bước 13: Trở lại fibonacci(8) (tiếp tục từ bước 1)

- Đã có fibonacci(7) = 13 (từ bước 12)

(Phạm_Minh_Tú_DHKL17A2HN_23174600120(ca chiều))

- Gọi fibonacci(6) -> Trả về 8 (từ bước 11)

Kết quả: $\text{fibonacci}(8) = 13 + 8 = 21$

=> **Kết Quả**

- Khi gọi `print(fibonacci(8))`, chương trình sẽ in ra kết quả là 21.

Bài 3: Cho hàm đệ quy để tính x mũ n. Hãy giải thích từng bước thực hiện của hàm đệ quy này khi $x = 2$ và $n = 6$.

```
def power(x, n):  
    if n == 0:  
        return 1  
    else:  
        return x * power(x, n-1)  
print(power(2, 6))
```

Bước 1: Gọi `power(2, 6)`:

- n không bằng 0, do đó trả về $2 * \text{power}(2, 5)$

Bước 2: Gọi `power(2, 5)`:

- n không bằng 0, do đó trả về $2 * \text{power}(2, 4)$

Bước 3: Gọi `power(2, 4)`:

- n không bằng 0, do đó trả về $2 * \text{power}(2, 3)$

Bước 4: Gọi `power(2, 3)`:

- n không bằng 0, do đó trả về $2 * \text{power}(2, 2)$

Bước 5: Gọi `power(2, 2)`:

- n không bằng 0, do đó trả về $2 * \text{power}(2, 1)$

(Phạm_Minh_Tú_DHKL17A2HN_23174600120(ca chiều))

Bước 6: Gọi $\text{power}(2, 1)$:

- n không bằng 0, do đó trả về $2 * \text{power}(2, 0)$

Bước 7: Gọi $\text{power}(2, 0)$:

- $n=0$, do đó trả về 1 (điều kiện cơ sở)

Bước 8: Quay lại $\text{power}(2, 1)$:

- Đã có $\text{power}(2, 0)=1$

- Do đó $\text{power}(2, 1)=2 * 1=2$

Bước 9: Quay lại $\text{power}(2, 2)$:

- Đã có $\text{power}(2, 1)=2$

- Do đó $\text{power}(2, 2)=2 * 2=4$

Bước 10: Quay lại $\text{power}(2, 3)$:

- Đã có $\text{power}(2, 2)=4$

- Do đó $\text{power}(2, 3)=2 * 4=8$

Bước 11: Quay lại $\text{power}(2, 4)$:

- Đã có $\text{power}(2, 3)=8$

- Do đó $\text{power}(2, 4)=2 * 8=16$

Bước 12: Quay lại $\text{power}(2, 5)$:

- Đã có $\text{power}(2, 4)=16$

- Do đó $\text{power}(2, 5)=2 * 16=32$

(Phạm_Minh_Tú_DHKL17A2HN_23174600120(ca chiều))

Bước 13: Quay lại $\text{power}(2, 6)$:

- Đã có $\text{power}(2, 5) = 32$

- Do đó $\text{power}(2, 6) = 2 * 32 = 64$

=> **Kết Quả**

Khi gọi $\text{print}(\text{power}(2, 6))$, chương trình sẽ in ra kết quả là 64.

Bài 4: Cho hàm đệ quy giải bài toán Tháp Hà Nội. Hãy giải thích từng bước thực hiện của hàm đệ quy này chuyển 4 đĩa từ cọc A sang cọc B, với trung gian là cọc C.

```
def thap_ha_noi(n, A, C, B):  
    if n == 1:  
        print(f"Chuyển đĩa 1 từ cột {A} sang cột {B}")  
    else:  
        thap_ha_noi(n-1, A, B, C)  
        print(f"Chuyển đĩa {n} từ cột {A} sang cột {B}")  
        thap_ha_noi(n-1, C, A, B)  
  
#Chuyển 4 đĩa từ cọc A sang cọc B, với trung gian là cọc C  
thap_ha_noi(4, "A", "C", "B")
```

- Hàm $\text{thap_ha_noi}(n, A, C, B)$

Nếu $n == 1$, in ra thông báo chuyển đĩa từ cọc A sang cọc B.

Nếu $n > 1$, thực hiện ba bước sau:

- Gọi đệ quy để chuyển $n-1$ đĩa từ cọc A sang cọc C, sử dụng cọc B làm trung gian.
- Chuyển đĩa thứ n từ cọc A sang cọc B.
- Gọi đệ quy để chuyển $n-1$ đĩa từ cọc C sang cọc B, sử dụng cọc A làm trung gian.

Bây giờ ta giải thích từng bước cụ thể khi chuyển 4 đĩa từ cọc A sang cọc B với trung gian là cọc C bằng cách gọi hàm $\text{thap_ha_noi}(4, "A", "C", "B")$.

Gọi $\text{thap_ha_noi}(4, "A", "C", "B")$

Bước 1: Gọi $\text{thap_ha_noi}(3, "A", "B", "C")$

1.1: Gọi $\text{thap_ha_noi}(2, "A", "C", "B")$

(Phạm_Minh_Tú_DHKL17A2HN_23174600120(ca chiều))

a, Gọi `thap_ha_noi(1, "A", "B", "C")`

- In: "Chuyển đĩa 1 từ cột A sang cột B"

b, In: "Chuyển đĩa 2 từ cột A sang cột C"

c, Gọi `thap_ha_noi(1, "B", "A", "C")`

- In: "Chuyển đĩa 1 từ cột B sang cột C"

1.2: In: "Chuyển đĩa 3 từ cột A sang cột B"

1.3: Gọi `thap_ha_noi(2, "C", "A", "B")`

a, Gọi `thap_ha_noi(1, "C", "B", "A")`

- In: "Chuyển đĩa 1 từ cột C sang cột A"

b, In: "Chuyển đĩa 2 từ cột C sang cột B"

c, Gọi `thap_ha_noi(1, "A", "C", "B")`

- In: "Chuyển đĩa 1 từ cột A sang cột B"

Bước 2: In: "Chuyển đĩa 4 từ cột A sang cột B"

Bước 3: Gọi `thap_ha_noi(3, "C", "A", "B")`

3.1: Gọi `thap_ha_noi(2, "C", "B", "A")`

a, Gọi `thap_ha_noi(1, "C", "A", "B")`

- In: "Chuyển đĩa 1 từ cột C sang cột A"

b, In: "Chuyển đĩa 2 từ cột C sang cột B"

c, Gọi `thap_ha_noi(1, "A", "C", "B")`

- In: "Chuyển đĩa 1 từ cột A sang cột B"

3.2: In: "Chuyển đĩa 3 từ cột C sang cột B"

3.3: Gọi `thap_ha_noi(2, "A", "C", "B")`

a, Gọi `thap_ha_noi(1, "A", "B", "C")`

- In: "Chuyển đĩa 1 từ cột A sang cột C"

b, In: "Chuyển đĩa 2 từ cột A sang cột B"

(Phạm_Minh_Tú_DHKL17A2HN_23174600120(ca chiều))

c, Gọi `thap_ha_noi(1, "C", "A", "B")`

- In: "Chuyển đĩa 1 từ cột C sang cột B"

Bài 5: Cho hàm đệ quy giải bài toán cổ vừa gà vừa chó. Hãy giải thích từng bước thực hiện của hàm đệ quy của bài toán này.

```
def cho_ga(tong_so_con, tong_so_chan):
    if tong_so_con == 0 and tong_so_chan == 0:
        return 0,0
    if tong_so_chan % 2 != 0:
        return -1, -1
    for cho in range(tong_so_con + 1):
        ga = tong_so_con - cho
        if ga * 2 + cho * 4 == tong_so_chan:
            return cho, ga
    cho, ga = cho_ga(tong_so_con - 1, tong_so_chan - 4)
    if ga != -1:
        return cho + 1, ga
    else:
        return -1, -1

tong_so_chan = 100
tong_so_con = 36
so_cho, so_ga = cho_ga(tong_so_con, tong_so_chan)
print("Số gà là:", so_ga)
print("Số chó là:", so_cho)
```

Bước 1: Gọi hàm `cho_ga(36, 100)`

1.1: Kiểm tra điều kiện cơ bản

- `tong_so_con = 36` và `tong_so_chan = 100`, không thỏa mãn điều kiện `tong_so_con == 0 and tong_so_chan == 0`.

- `100 % 2 == 0`, không thỏa mãn điều kiện `tong_so_chan % 2 != 0`.

1.2: Thử các khả năng về số lượng chó

- Vòng lặp từ `cho = 0` đến `cho = 36`.

- Nếu `cho = 0`, `ga = 36`. Kiểm tra `36 * 2 + 0 * 4 != 100`.

- Nếu `cho = 1`, `ga = 35`. Kiểm tra `35 * 2 + 1 * 4 != 100`.

- ...

**(Phạm_Minh_Tú_DHKL17A2HN_23174600120(ca
chiều))**

- Nếu cho = 14, ga = 22. Kiểm tra $22 * 2 + 14 * 4 == 100$. Đúng! => trả về (14, 22).

=> Kết quả

- Tìm thấy giải pháp trong vòng lặp, không cần gọi đệ quy.
- Trả về (14, 22).