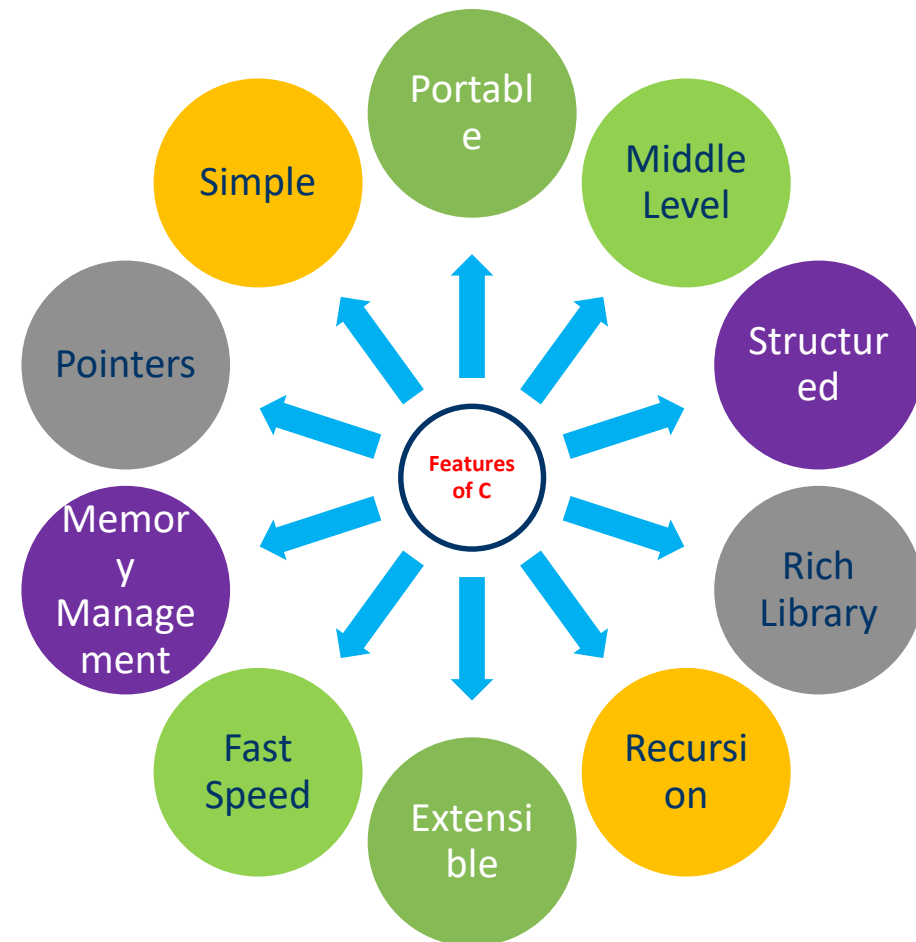


PHƯƠNG PHÁP LẬP TRÌNH

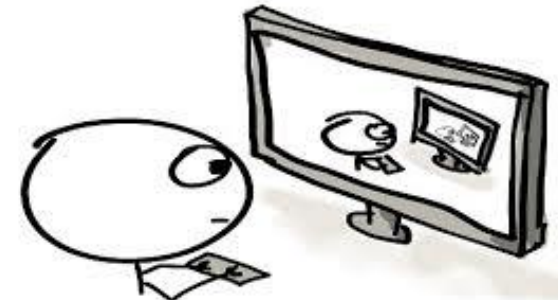


NỘI DUNG MÔN HỌC

1. Tổng quan về giải thuật
2. Ngôn ngữ lập trình C
3. Cấu trúc điều khiển (*Control structures and statements*)
4. Hàm (*Function*)
5. Mảng 1 chiều (*One array dimension*)
6. Chuỗi ký tự (*String*)
7. Mảng hai chiều (*Two array dimension*)
8. Kiểu dữ liệu cấu trúc (*Struct data type*)
9. Kiểu dữ liệu con trỏ (*Pointer data type*)
10. Tập tin (*File*)
11. Đề quy

Chương 11

ĐỆ QUY (*Recursion*)



MỤC TIÊU

- i. Giải thích được đệ quy là gì
- ii. Phân loại được các loại đệ quy
- iii. Biết cách giải quyết 1 tác vụ hướng đệ quy
- iv. Biết cách hiện thực hàm đệ quy
- v. Giải thích được cách thực hiện của một hàm đệ quy

NỘI DUNG

1. Tổng quan
2. Phân loại đệ quy
3. Đệ quy nhị phân
4. Đệ quy phi tuyến
5. Đệ quy hồi tưởng
6. Nhận xét
7. Thực hành
8. Bài tập

1. TỔNG QUAN

- Vấn đề đệ quy là vấn đề được định nghĩa bằng chính nó.

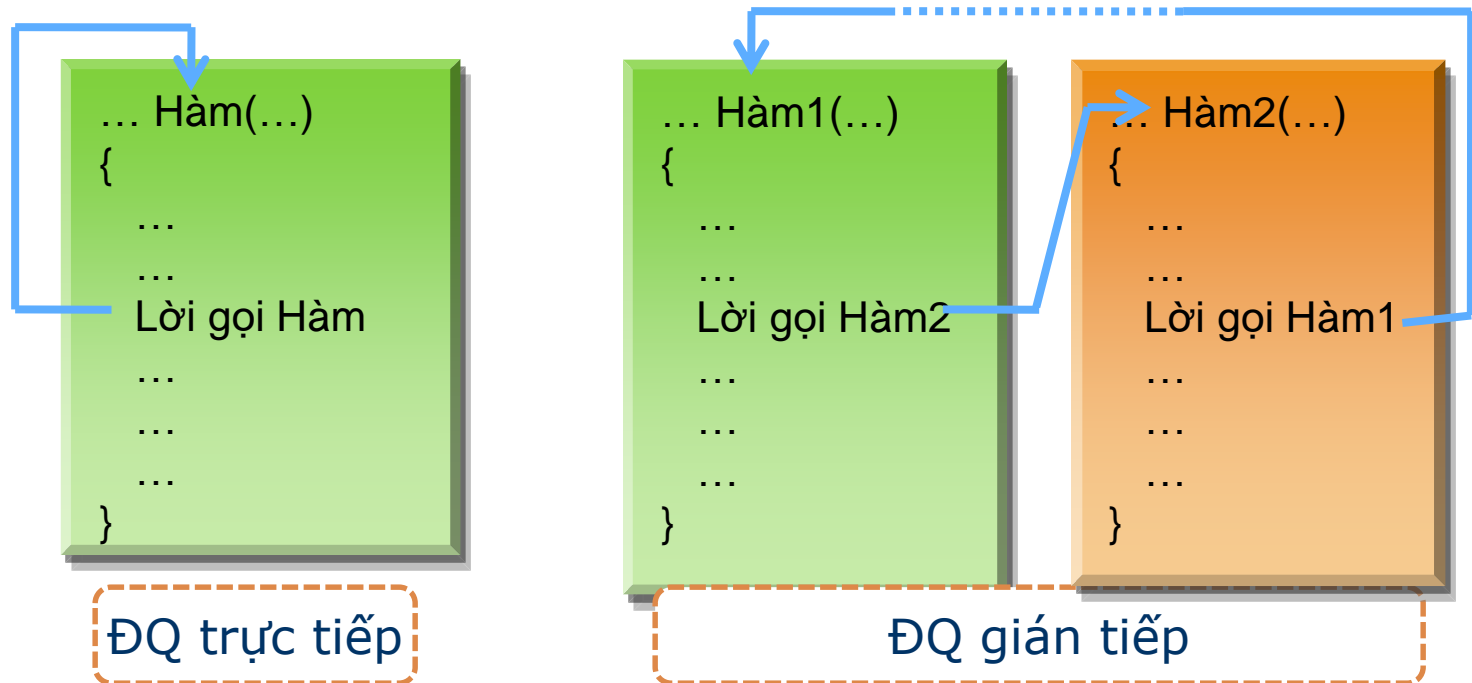
Ví dụ: Tổng $S(n)$ được tính thông qua tổng của $S(n-1)$.

- **Khái niệm**: Một hàm được gọi có tính đệ quy nếu trong thân của hàm đó có lệnh gọi lại chính nó một cách tường minh hay tiềm ẩn.
- **Điều kiện để giải theo phương pháp đệ quy**: cần hai điều kiện quan trọng sau:
 - Phải tồn tại bước đệ quy.
 - Phải có điều kiện dừng.



1. Tổng quan

- **Hàm đệ quy trong NNLT C**: 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.

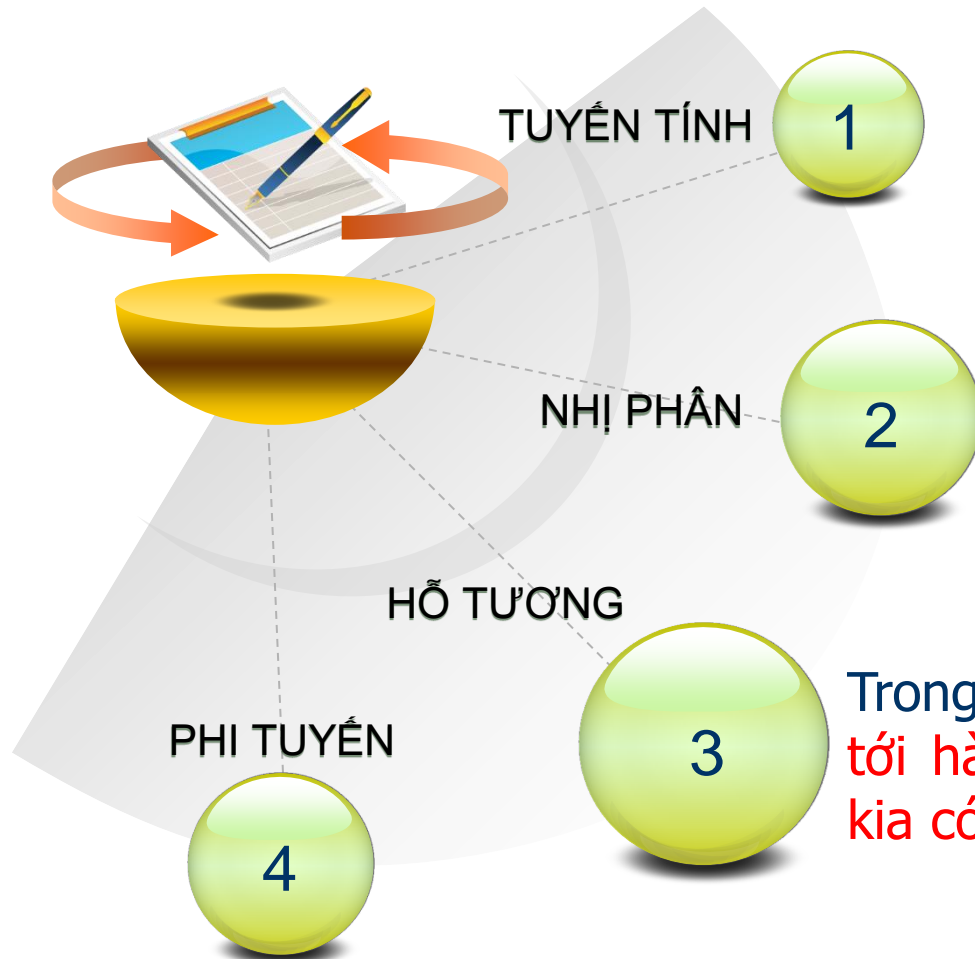


1. TỔNG QUAN

— *Một số khái niệm*

- ***Recursion*** – Đệ quy
- ***Recursive*** – Tính đệ quy.
- ***Recursive problem*** – vấn đề đệ quy
- ***Base case*** – Trường hợp cơ bản – trường hợp đầu vào mà ta có thể giải quyết vấn đề không cần dùng đến sự đệ quy.
- ***Single recursion*** – đệ quy tuyến tính, trường hợp đệ quy chỉ đề cập lại đến nó một lần
- ***Multiple recursion*** – đệ quy phi tuyến, trường hợp đệ quy đề cập lại chính nó nhiều lần.

2. PHÂN LOẠI ĐỆ QUY



Trong thân hàm có **duy nhất** một lời gọi hàm gọi lại chính nó một cách tường minh.

Trong thân hàm có **hai** lời gọi hàm gọi lại chính nó một cách tường minh.

Trong thân hàm này có lời gọi hàm tới hàm kia và bên trong thân hàm kia có lời gọi hàm tới hàm này.

Trong thân hàm có lời gọi hàm lại chính nó được đặt bên trong thân vòng lặp.

2. PHÂN LOẠI ĐỆ QUY

2.1. ĐỆ QUY TUYẾN TÍNH

- Trong thân hàm có duy nhất một lời gọi hàm gọi lại chính nó một cách tường minh.

```
<Kiểu dữ~liệu hàm> TenHam (<danh sách tham số>)  
{  
    //B1: Kiểm tra điều kiện dừng  
    if (điều kiện dừng)  
    {  
        //Tra`vê`gia`trị hay kết thúc công việc  
    }  
    /*B2: Thực hiện một số công việc (thường là  
gọi đệ quy)*/  
    . . . TenHam (<danh sách tham số>);  
}
```

2. PHÂN LOẠI ĐỀ QUY

2.1. Đề quy tuyến tính

– Ví dụ 1: Tính $S(n) = 1 + 2 + 3 + \dots + n$

- Quy tắc (công thức) tính:

Ta có: $S(n) = 1 + 2 + 3 + \dots + (n-3) + (n-2) + (n-1) + n$

Mà $S(n-1) = 1 + 2 + 3 + \dots + (n-3) + (n-2) + (n-1)$

và: $S(n-2) = 1 + 2 + 3 + \dots + (n-3) + (n-2)$

.....

Suy ra quy tắc (công thức) tính: $S(n) = S(n-1) + n;$

- Điều kiện dừng: $S(1) = 1$

Giả sử với $n=4$:

$$S(4) = S(3) + 4$$

$$S(3) = S(2) + 3$$

$$S(2) = S(1) + 2$$

$$S(1) = 1$$

2. PHÂN LOẠI ĐỆ QUY

2. Đệ quy tuyến tính

– Ví dụ 1 : Tính $S(n) = 1 + 2 + 3 + \dots + n$, $n > 0$

- Cài đặt: $s = s + 1$

```
long TongS (int n)
{
    //B1: Kiểm tra điều kiện dừng
    if (n==0)           // hay if (n==1)
        return 0; //          return 1;
    // B2: gọi đệ quy
    return ( TongS (n-1) + n );
}
```

2. PHÂN LOẠI ĐỀ QUY

2.1. Đề quy tuyến tính

– Ví dụ 2: Tính $P(n) = n!$

- Quy tắc (công thức) tính:

Ta có : $P(n) = 1 \times 2 \times 3 \times \dots \times (n-2) \times (n-1) \times n$

Do đó : $P(n-1) = 1 \times 2 \times 3 \times \dots \times (n-2) \times (n-1)$

Tương tự: $P(n-2) = 1 \times 2 \times 3 \times \dots \times (n-2)$

...

Suy ra quy tắc (công thức) tính: $P(n) = P(n-1) \times n$;

- Điều kiện dừng: $S(1) = 1$

- Cài đặt

```
long GiaiThua (int n)
{
    //B1: Kiểm tra điều kiện dừng
    if(n==1)
        return 1;
    // B2: gọi đệ quy
    return (GiaiThua (n-1) * n );
}
```

2. PHÂN LOẠI ĐỆ QUY

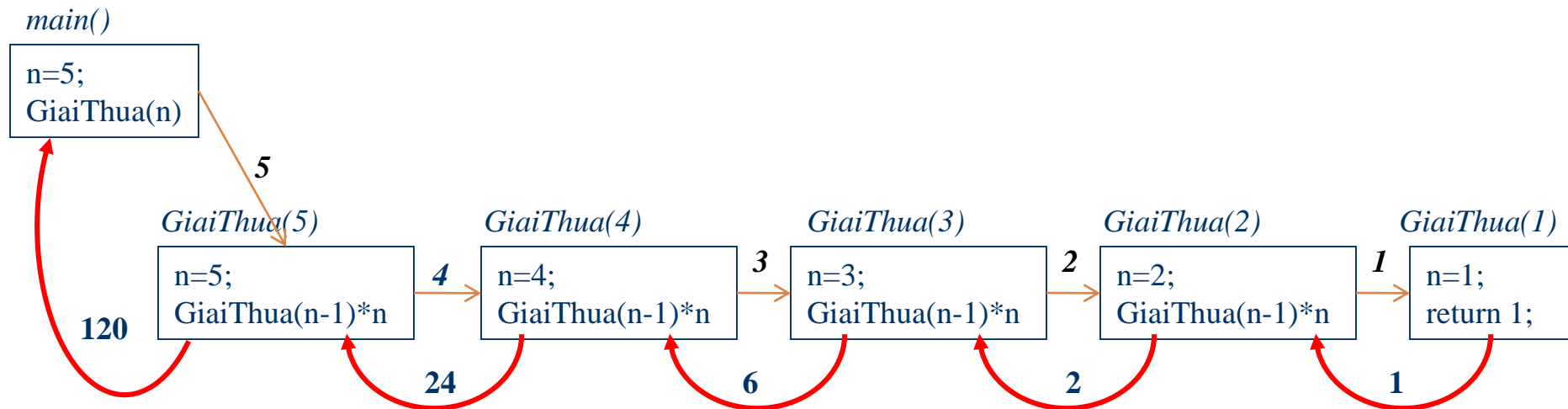
2.1. Đệ quy tuyến tính

Cách hoạt động hàm đệ quy

– Ví dụ 2: tính $P(n)$? với $n=5$

```
void main()  
{   int n=5;  
    printf("%ld", GiaiThua(n));  
}
```

```
long GiaiThua (int n)  
{   //B1:Kiểm tra điều kiện dừng  
    if (n==1)  
        return 1;  
    // B2: gọi đệ quy  
    return (GiaiThua(n-1)*n);  
}
```



2. PHÂN LOẠI ĐỀ QUY

2.1. Đề quy tuyến tính

- Ví dụ 3: Cho mảng một chiều các số nguyên. Viết hàm tính tổng cho các số chẵn trong mảng bằng phương pháp đệ quy. Ta có hình ảnh mảng a có n phần tử như sau:

0	1		n-2	n-1
12	43	...	23	44

- Quy tắc (công thức tính):

Ta có: $tong = a[1] + a[2] + \dots + a[n-2] + a[n-1]$

Suy ra quy tắc (công thức) tính: $tong = tong(n-1) + n$;

- Điều kiện dừng: $n=0$ (phần tử cuối của của mảng có chỉ số là 0)
- Cài đặt:

```
long Tong(int a[], int n)
{ //B1: Kiểm tra điều kiện dừng
    if (n==0)
        return 0;
    // B2: gọi đệ quy
    if (a[n-1]%2==0)
        return Tong(a,n-1)+a[n-1];
    return Tong(a,n-1);
}
```

2. PHÂN LOẠI ĐỆ QUY

2.1. Đệ quy tuyến tính

- Một số lưu ý đối với đệ quy tuyến tính:
 - Đệ quy tuyến tính dễ dàng chuyển sang cấu trúc lặp
 - Hầu hết trường hợp cấu trúc lặp sẽ chạy nhanh hơn, dùng ít bộ nhớ hơn đệ quy

2.2. Đệ quy nhị phân

Trong thân của hàm có hai lời gọi hàm gọi lại chính nó một cách tường minh.

```
<Kiểu dữ liệu hàm> TenHam (<danh sách tham số>)  
{  
    //B1: Kiểm tra điều kiện dừng  
    if (điều kiện dừng)  
    { ...  
        //Trả về giá trị hay kết thúc công việc  
    }  
    // B2: gọi đệ quy  
    //Thực hiện một số công việc (nếu có)  
    //Giải quyết vấn đề nhỏ hơn  
    ... TenHam (<danh sách tham số>);  
    //Thực hiện một số công việc (nếu có)  
    //Giải quyết vấn đề còn lại  
    ... TenHam (<danh sách tham số>);  
    //Thực hiện một số công việc (nếu có)  
}
```

2. PHÂN LOẠI ĐỀ QUY

2.2. Đề quy nhị phân

Ví dụ: Tính số hạng thứ ***n*** của dãy ***Fibonacci*** được định nghĩa như sau (*Số tiếp theo trong dãy sẽ bằng tổng hai số liền trước nó*):

$$f_1 = f_0 = 1 ;$$

$$f_n = f_{n-1} + f_{n-2} ; \quad (n > 1)$$

- Điều kiện dừng: $f(0) = f(1) = 1$.
- Cài đặt

```
long Fibonacci (int n)
{
    //B1: Kiểm tra điều kiện dừng
    if (n==0 || n==1)
        return 1;
    // B2: gọi đệ quy
    return  Fibonacci(n-1) + Fibonacci(n-2) ;
}
```

2.2. Độ quy nhị phân

Minh họa Call stack trong đệ quy nhị phân của bài toán Fibonacci

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

int main()
{
    printf("%d",f(4));
}
```

- Tại một thời điểm, stack chỉ có thể chứa số lượng phần tử có hạn.
- Khi chiều cao của stack quá lớn, chương trình có thể sẽ gặp lỗi ***stack overflow***

[illegible]

2.3. Đề quy phi tuyến

Trong thân của hàm có lời gọi hàm gọi lại chính nó được đặt bên trong vòng lặp.

```
<Kiểu dữ liệu hàm> TenHam (<danh sách tham số>)  
{  
    for (int i = 1; i<=n; i++)  
    { //Thực hiện một số công việc (nếu có)  
        if (điều kiện dừng)  
        {  
            ...  
            //Trả về giá trị hay kết thúc công việc  
        }  
        else  
        {  
            //Thực hiện một số công việc (nếu có)  
            TenHam (<danh sách tham số>);  
        }  
    }  
}
```

2. PHÂN LOẠI ĐỀ QUY

2.3. Đề quy phi tuyến

- Ví dụ: Tính số hạng thứ n của dãy $\{X_n\}$ được định nghĩa như sau:

$$X_0 = 1 ;$$

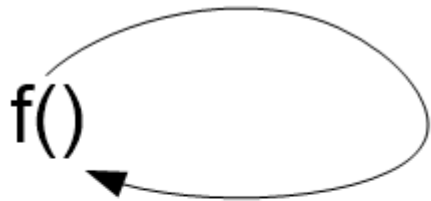
$$X_n = n^2 X_0 + (n-1)^2 X_1 + \dots + 1^2 X_{n-1} ; (n \geq 1)$$

- Điều kiện dừng: $X(0) = 1$
- Cài đặt

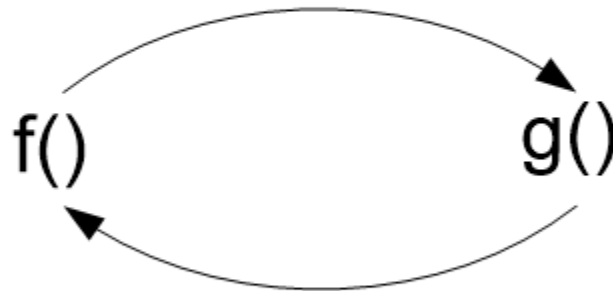
```
long TinhXn (int n)
{
    if (n==0)
        return 1;
    long s = 0;
    for (int i=1; i<=n; i++)
        s = s + i * i * TinhXn(n-i);
    return s;
}
```

2.4. ĐỆ QUY HỒ TƯƠNG

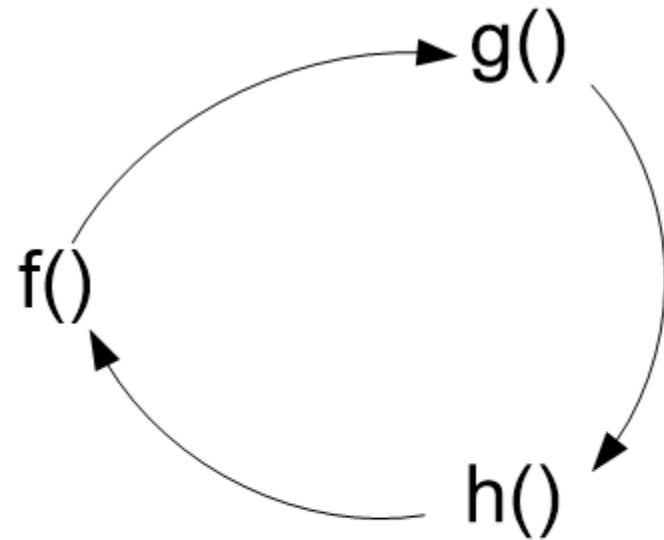
Trong thân của hàm này có lời gọi hàm đến hàm kia và trong thân của hàm kia có lời gọi hàm tới hàm này.



Đệ quy
tuyến tính



(a)



(b)

Đệ quy hồ tương

2. PHÂN LOẠI ĐỀ QUY

2.4. Đề quy hỗ tương

```
<Kiểu dữ liệu hàm> TenHam1 (<danh sách tham số>)  
{  
    //Thực hiện một số công việc (nếu có)  
    ... TenHam2 (<danh sách tham số>);  
    //Thực hiện một số công việc (nếu có)  
}
```

```
<Kiểu dữ liệu hàm> TenHam2 (<danh sách tham số>)  
{  
    //Thực hiện một số công việc (nếu có)  
    ... TenHam1 (<danh sách tham số>);  
    //Thực hiện một số công việc (nếu có)  
}
```

2. PHÂN LOẠI ĐỀ QUY

2.4. Đề quy hồi tương

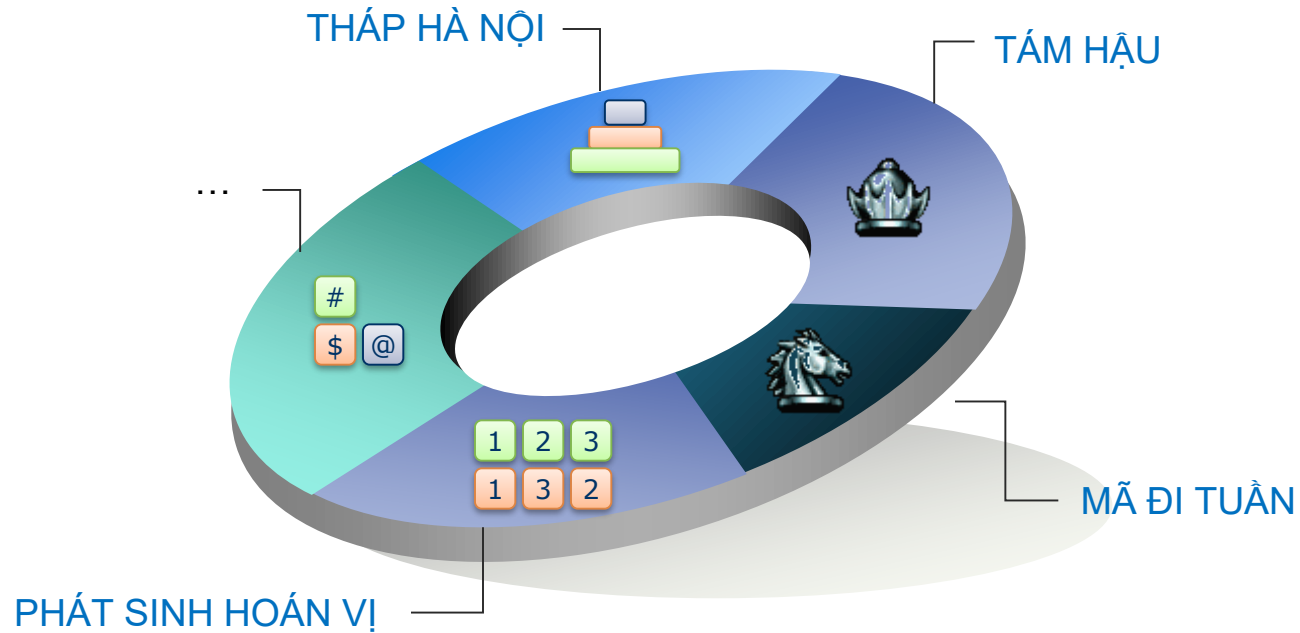
- Ví dụ: Tính số hạng thứ n của hai dãy $\{X_n\}$, $\{Y_n\}$ được định nghĩa như sau:

$$\begin{aligned}X_0 &= Y_0 = 1 ; \\X_n &= X_{n-1} + Y_{n-1}; \quad (n > 0) \\Y_n &= n^2 X_{n-1} + Y_{n-1}; \quad (n > 0)\end{aligned}$$

- Điều kiện dừng: $X(0) = Y(0) = 1$
- Cài đặt:

```
long TinhXn (int n)
{
    if (n==0)
        return 1;
    return TinhXn(n-1) + TinhYn(n-1);
}
long TinhYn (int n)
{
    if (n==0)
        return 1;
    return n*n*TinhXn(n-1) + TinhYn(n-1);
}
```

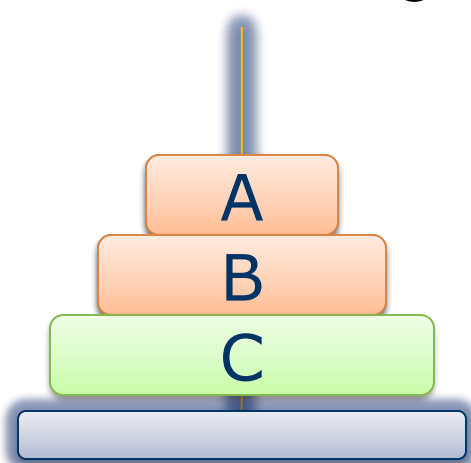

3. MỘT SỐ BÀI TOÁN KINH ĐIỂN SỬ DỤNG ĐỆ QUY



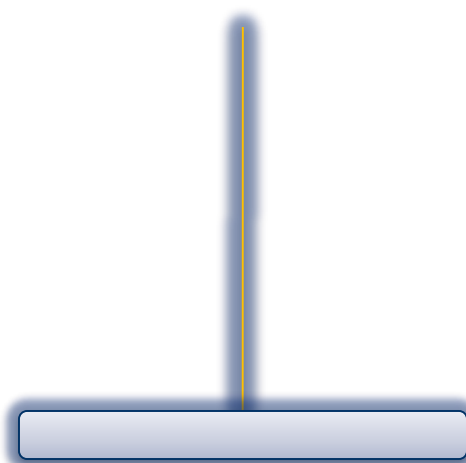
3.1. Tháp Hà Nội

— Mô tả bài toán

- Có 3 cột A, B và C và cột A hiện có N đĩa.
- Tìm cách chuyển N đĩa từ cột A sang cột C sao cho:
 - Một lần chuyển 1 đĩa
 - Đĩa lớn hơn phải nằm dưới.
 - Có thể sử dụng các cột A, B, C làm cột trung gian.



Cột nguồn 1



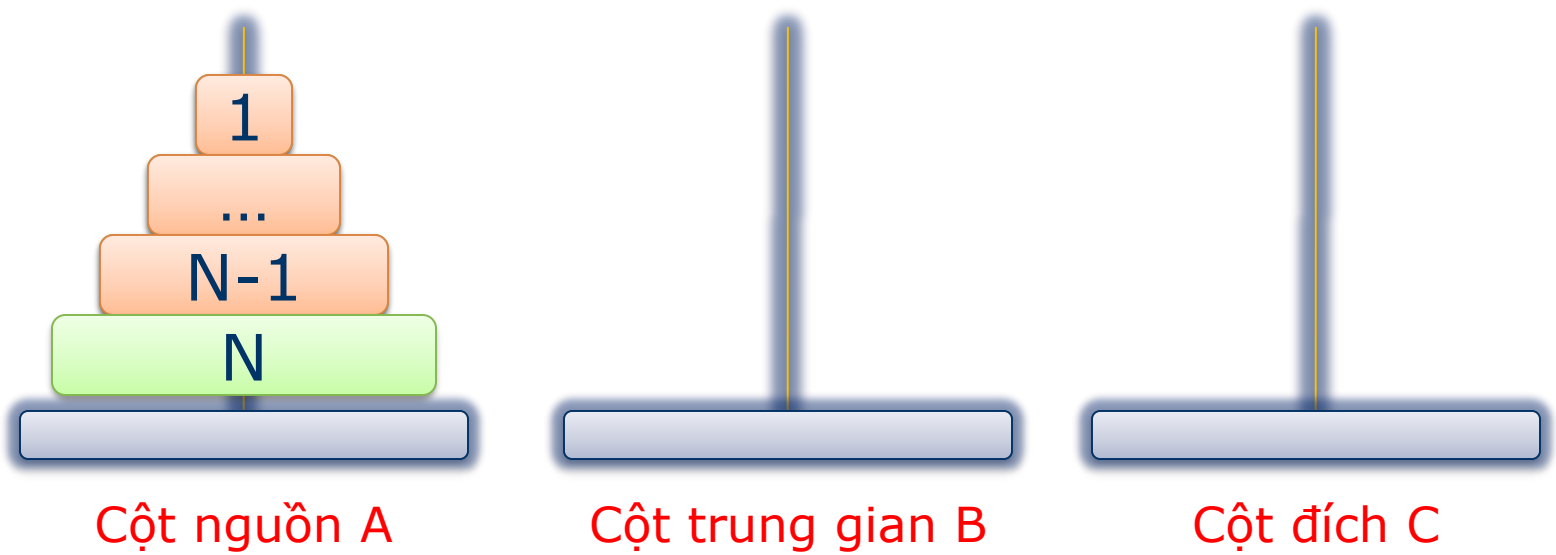
Cột trung gian 2



Cột đích 3

3.1.Tháp Hà Nội

$$N \text{ đĩa } A \rightarrow C = ? \text{ đĩa } A \rightarrow B + \text{Đĩa } N \text{ } A \rightarrow C + N-1 \text{ đĩa } B \rightarrow C$$



3.2. Tám hậu

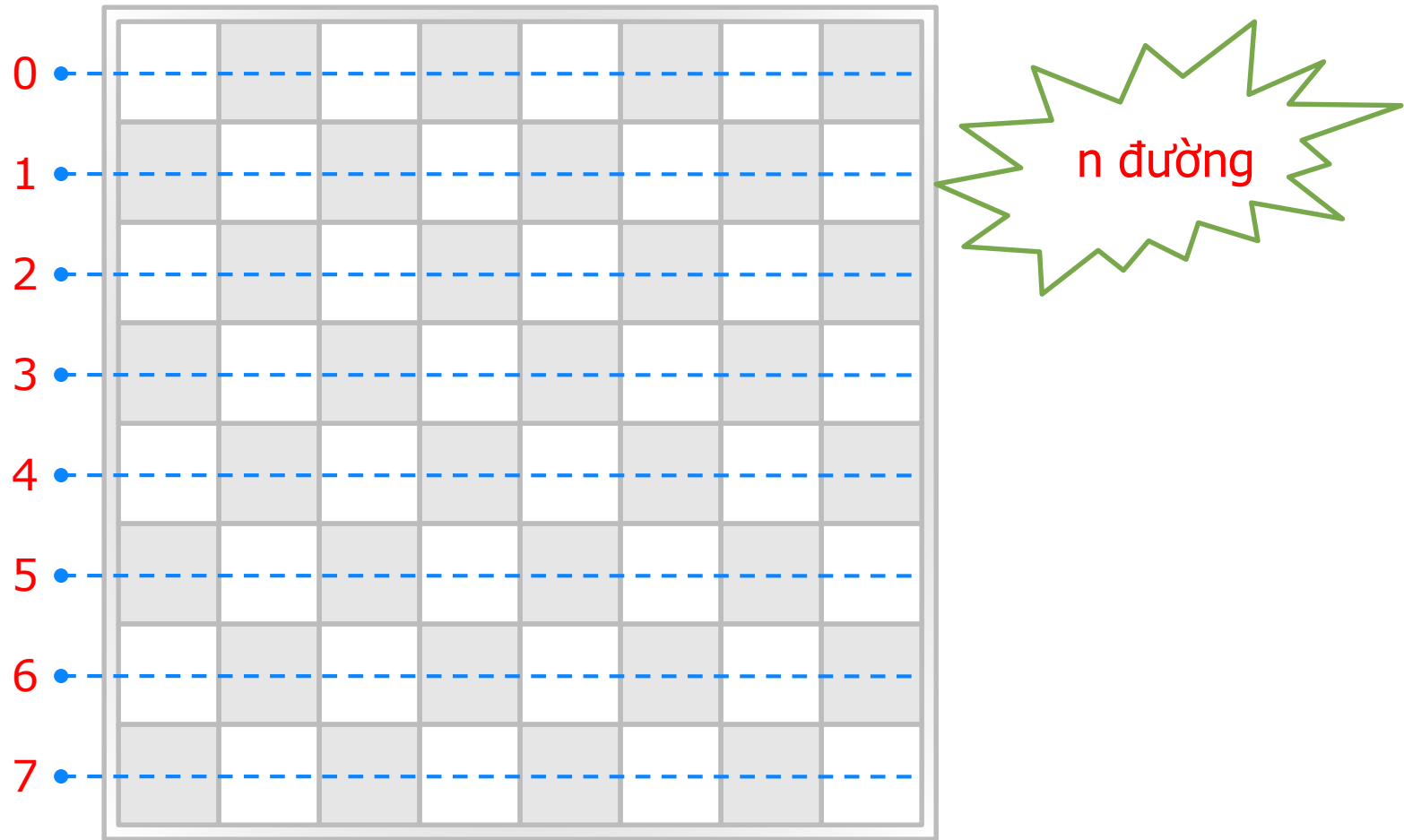
— Mô tả bài toán

- Cho bàn cờ vua kích thước 8×8
- Hãy đặt 8 hoàng hậu lên bàn cờ này sao cho không có hoàng hậu nào “ăn” nhau:
 - Không nằm trên cùng dòng, cùng cột
 - Không nằm trên cùng đường chéo xuôi, ngược.

3. Một số bài toán kinh điển sử dụng đệ quy

3.2. Tám hậu

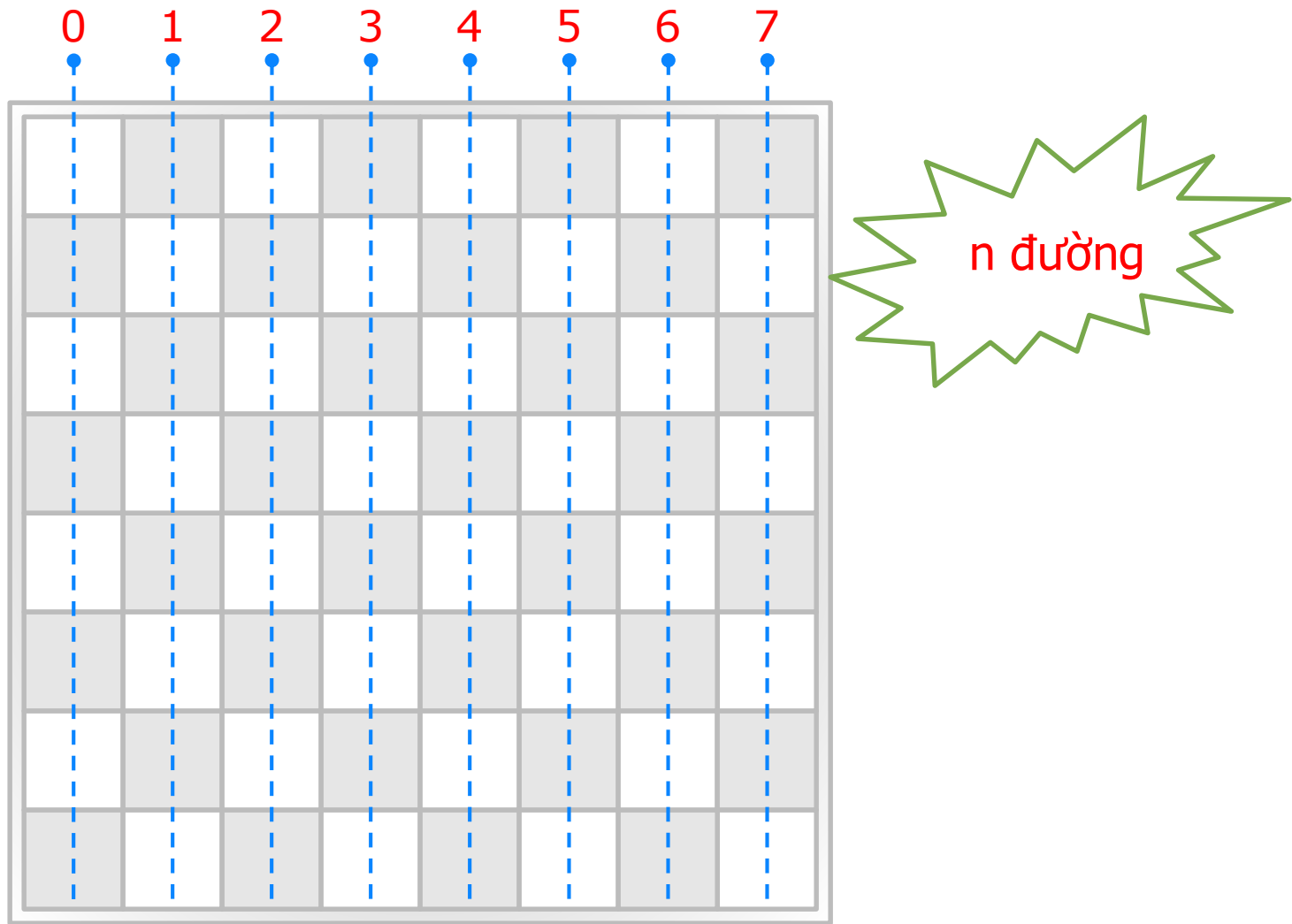
— Các dòng



3. Một số bài toán kinh điển sử dụng đệ quy

3.2. Tám hậu

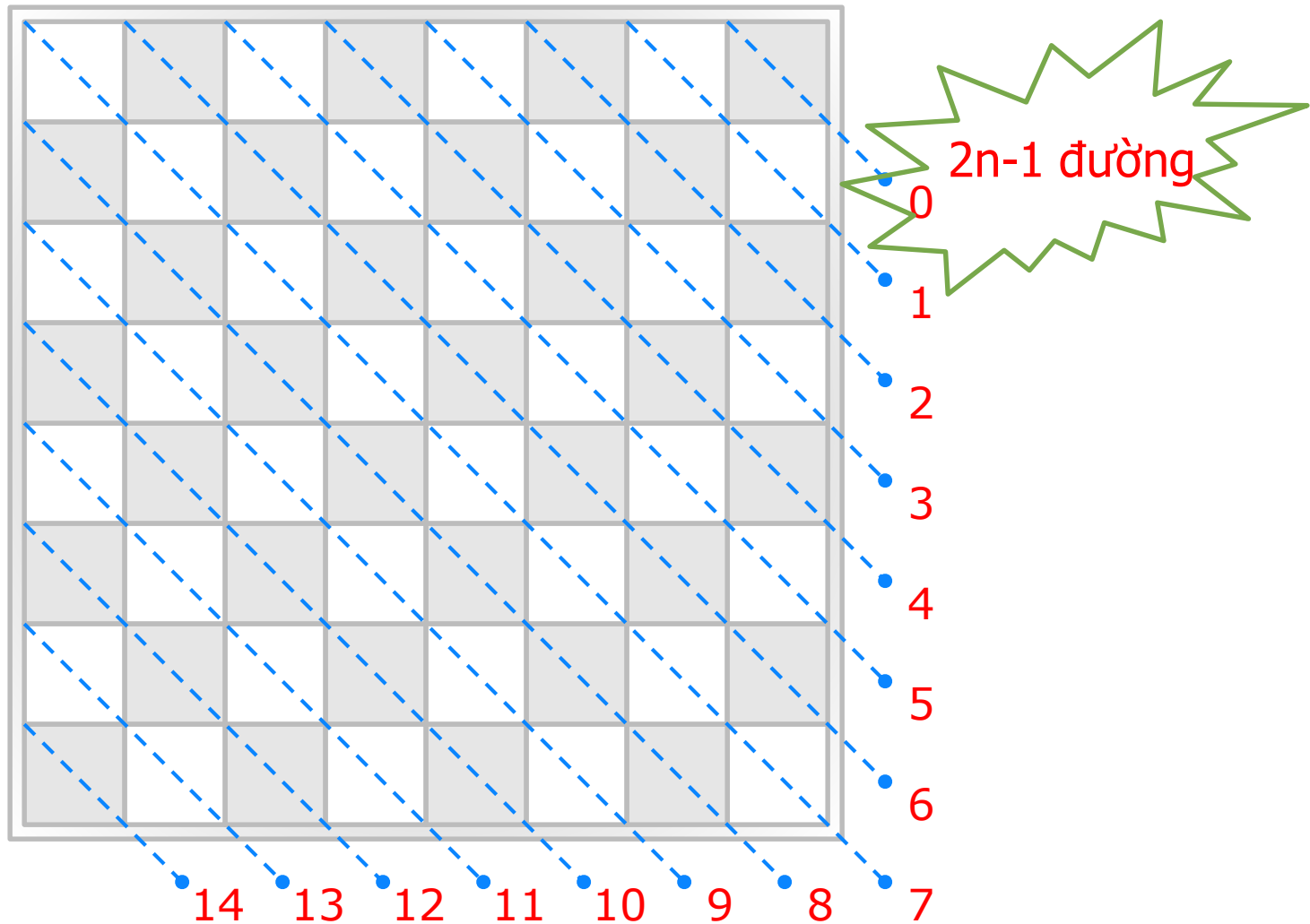
— Các cột



3. Một số bài toán kinh điển sử dụng đệ quy

3.2. Tám hậu

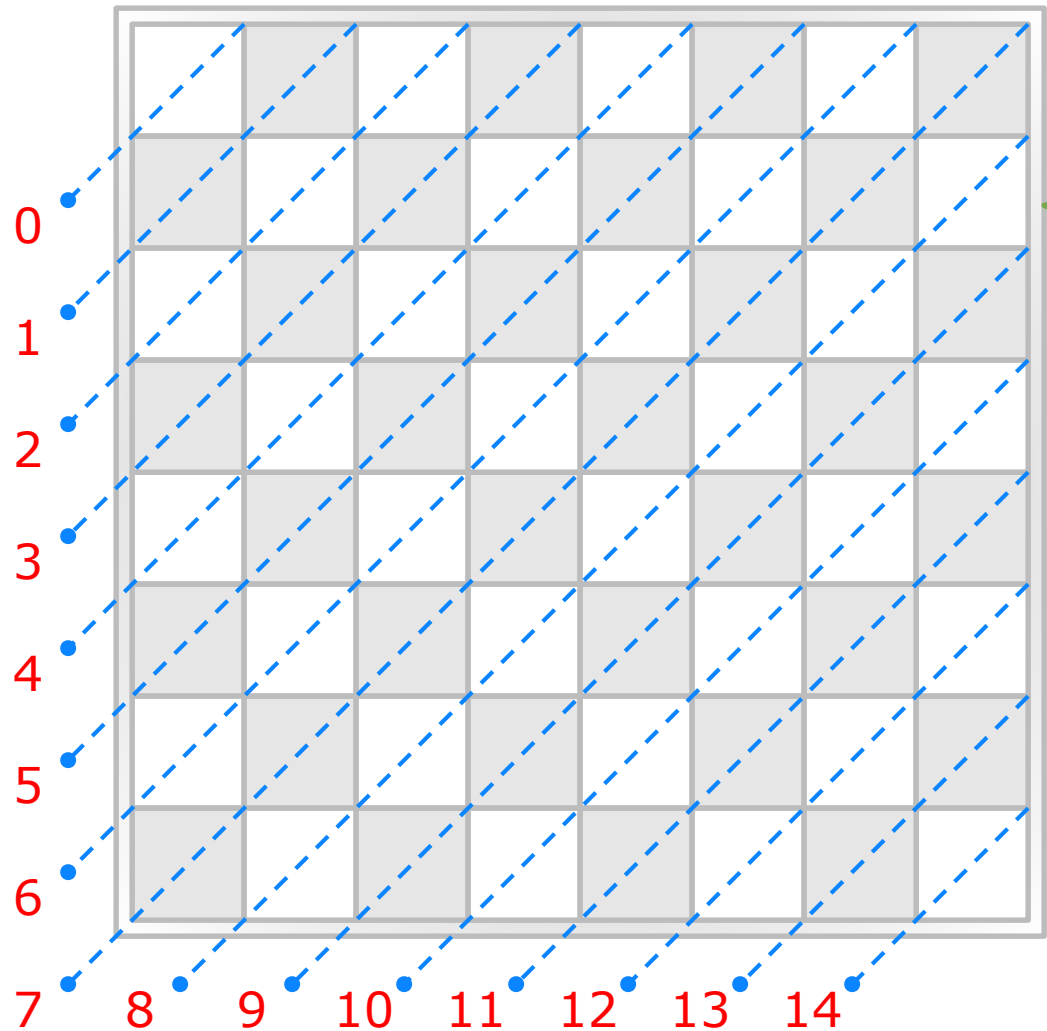
– Các đường chéo chính



3. Một số bài toán kinh điển sử dụng đệ quy

3.2. Tám hậu

— Các đường chéo phụ

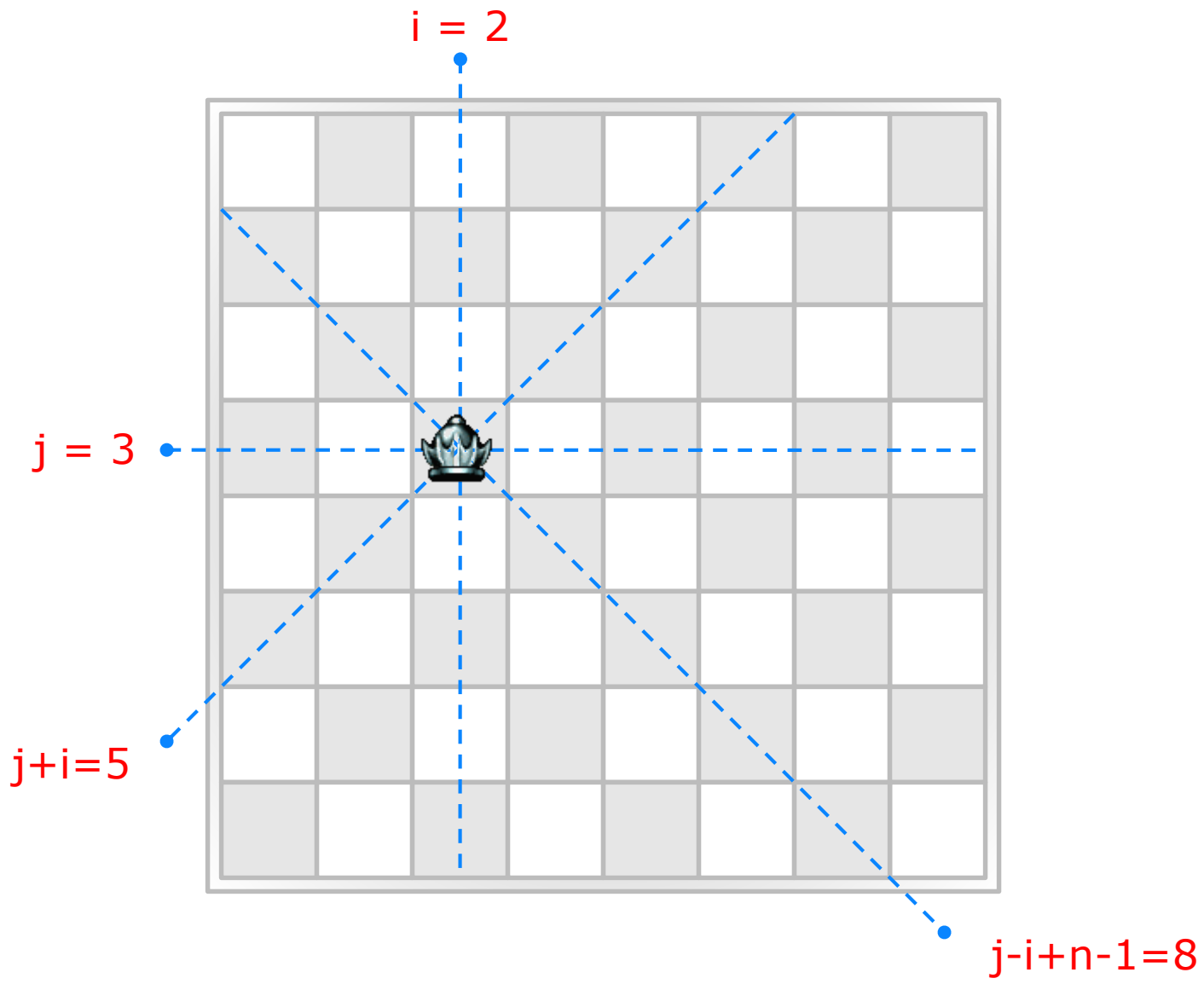


2n-1 đường

3. Một số bài toán kinh điển sử dụng đệ quy

3.2. Tám hậu

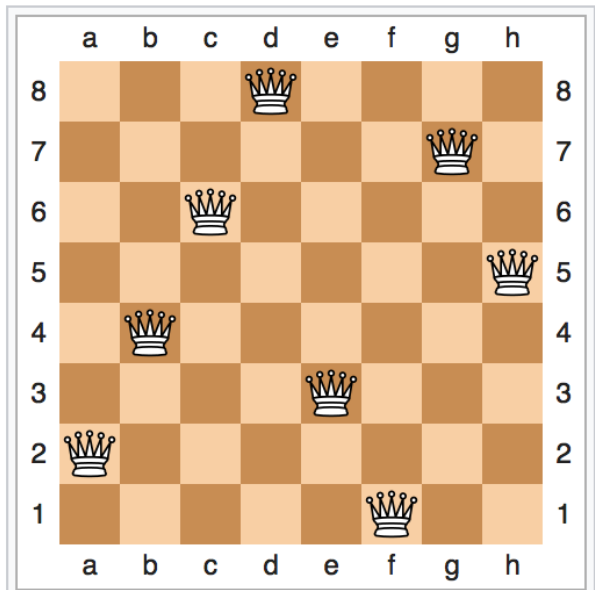
– Đặt 8 hậu ?



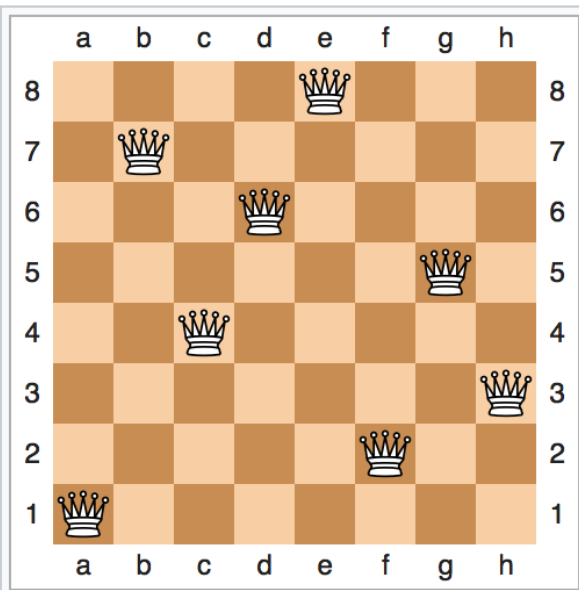
3. Một số bài toán kinh điển sử dụng đệ quy

3.2. Tám hậu

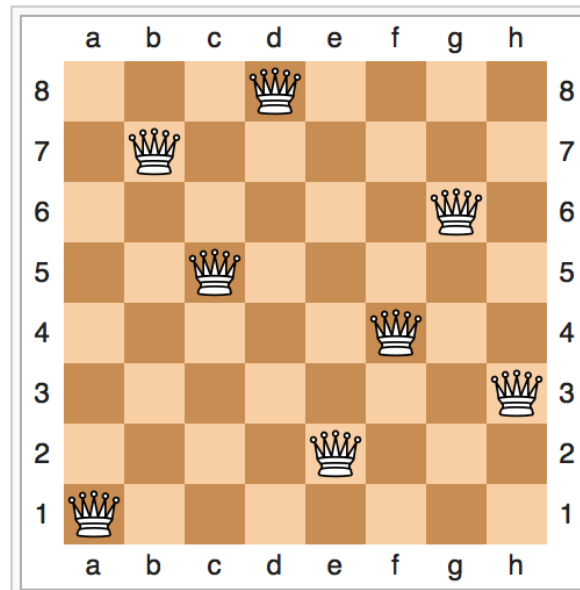
— Đặt 8 hậu ?



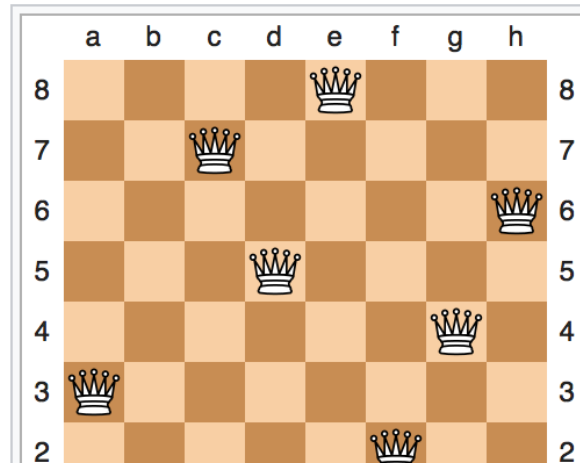
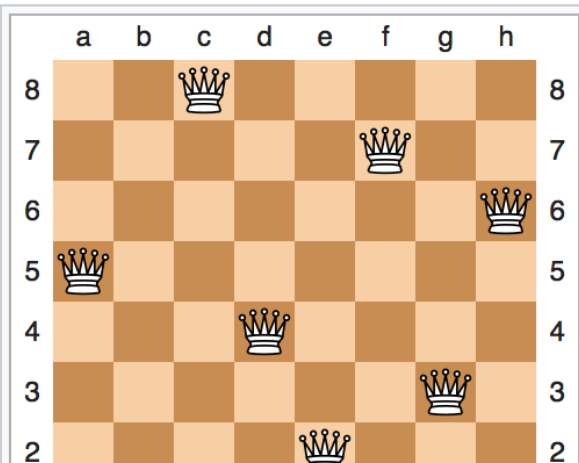
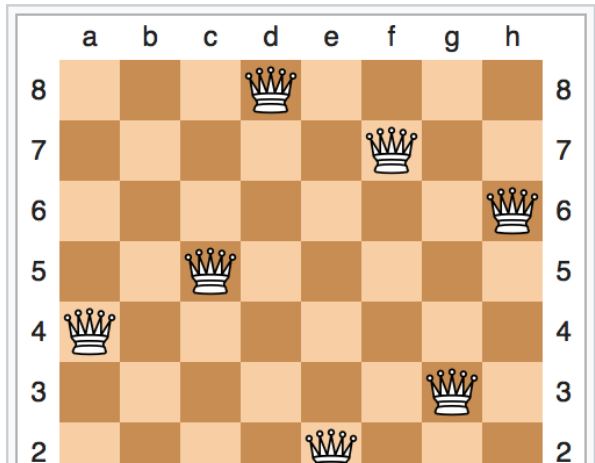
Lời giải 1



Lời giải 2

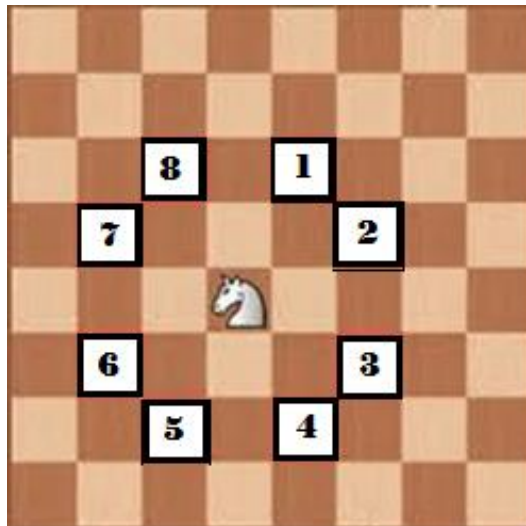


Lời giải 3



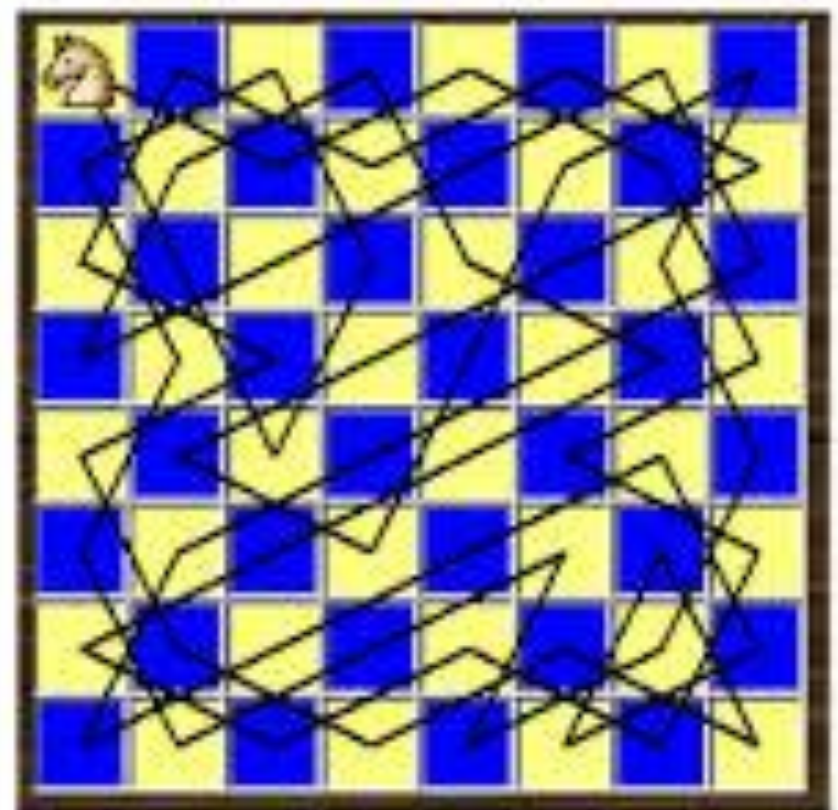
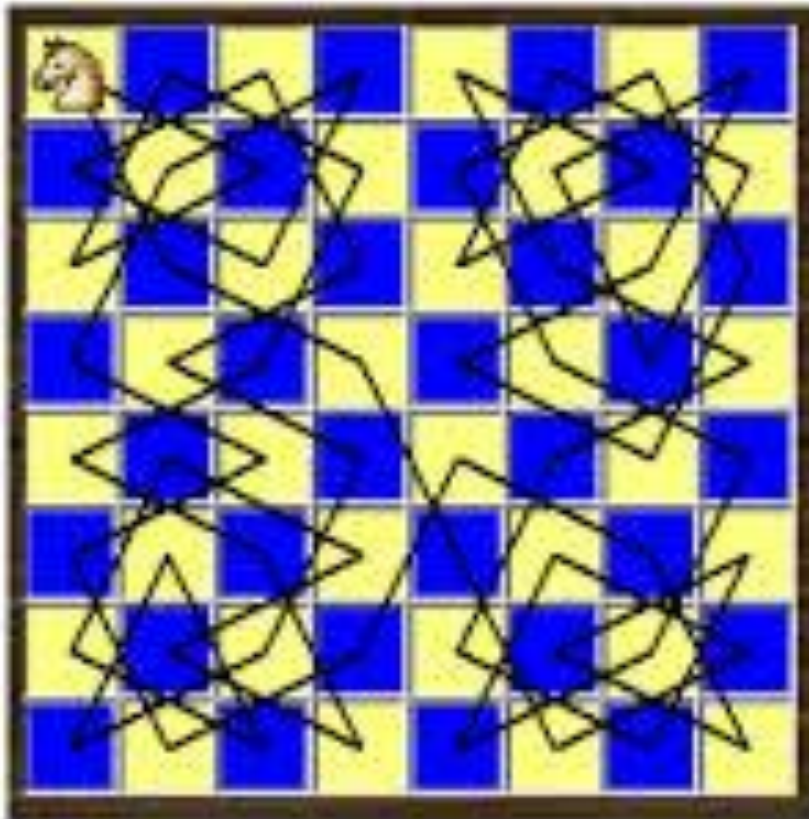
3.3. Mã đi tuần

- Mô tả bài toán
 - Cho bàn cờ vua kích thước 8×8 (64 ô)
 - Hãy đi con mã 64 nước sao cho mỗi ô chỉ đi qua 1 lần (xuất phát từ ô bất kỳ)
 - Trên bàn cờ, con mã chỉ có thể đi theo đường chéo của hình chữ nhật 2×3 hoặc 3×2 ô vuông. Giả sử bàn cờ có 8×8 ô vuông. Hãy tìm đường đi của con mã qua được tất cả các ô của bàn cờ, mỗi ô chỉ một lần rồi trở lại ô xuất phát.



3. Một số bài toán kinh điển sử dụng đệ quy

3.3. Mã đi tuần



3. Một số bài toán kinh điển sử dụng đệ quy

3.3. Mã đi tuần

NguaDiTuan

1	4	35	20	47	6	43	22
34	19	2	5	42	21	46	7
3	36	41	48	39	54	23	44
18	33	38	53	50	45	8	55
37	14	49	40	61	56	51	24
32	17	60	57	52	27	62	9
13	58	15	30	11	64	25	28
16	31	12	59	26	29	10	63

NguaDiTuan

4	1	18	27	44	53	16	57
19	26	3	52	17	56	43	54
2	5	28	45	48	51	58	15
25	20	49	32	59	46	55	42
6	29	24	47	50	41	14	63
21	36	31	60	33	64	11	40
30	7	34	23	38	9	62	13
35	22	37	8	61	12	39	10

NguaDiTuan

36	21	18	3	34	49	16	1
19	4	35	50	17	2	33	48
22	37	20	43	54	47	60	15
5	42	51	46	59	56	53	32
38	23	44	55	52	61	14	57
9	6	41	62	45	58	31	28
24	39	8	11	26	29	64	13
7	10	25	40	63	12	27	30

4. Các bước xây dựng hàm đệ quy

Ví dụ 1: yêu cầu viết hàm đệ quy tính giai thừa cho số nguyên n

B1. Tham số cần truyền cho hàm là gì? Số lượng tham số là bao nhiêu?

B2. Hàm có trả về giá trị hay không? (*void, int, float, ...*)

B3. Điều kiện dừng là gì? \Rightarrow Khi đó giá trị trả về là bao nhiêu?

B4. Phần không đệ quy là gì? (*$n-1$, $n/10$, $n\%10$, $A[n-1]$, ...*)

B5. Phần đệ quy là gì? \Rightarrow Kích thước nhỏ hơn của n là gì (*$n-1$, $n/10$, $n\%10$, ...*)?

B6. Tinh chỉnh lại nội dung hàm (nếu có thể)

```
long GiaiThua (int n)
{
    //Kiểm tra điều kiện dừng
    if (n==1)
        return 1;
    else //gọi đệ quy
        return GiaiThua(n-1) * n;
}
```

The diagram illustrates the mapping of the recursive function code to the requirements B1 through B6:

- B1** points to the parameter `(int n)`.
- B2** points to the return type `long`.
- B3** points to the base case `if (n==1) return 1;`.
- B4** points to the recursive call `GiaiThua(n-1)`.
- B5** points to the recursive call `GiaiThua(n-1)`.
- B6** points to the recursive call `GiaiThua(n-1)`.

4. Các bước xây dựng hàm đệ quy

Ví dụ 1: yêu cầu viết hàm đệ quy tính giai thừa cho số nguyên n

B1. Tham số cần truyền cho hàm là gì? Số lượng tham số là bao nhiêu?

B2. Hàm có trả về giá trị hay không? (void, int, float, ...)

B3. Điều kiện dừng là gì? \Rightarrow Khi đó giá trị trả về là bao nhiêu?

B4. Phần không đệ quy là gì? ($n-1$, $n/10$, $n\%10$, $A[n-1]$, ...)

B5. Phần đệ quy là gì? \Rightarrow Kích thước nhỏ hơn của n là gì ($n-1$, $n/10$, $n\%10$, ...)?

B6. Tinh chỉnh lại nội dung hàm (nếu có thể)

```
long GiaiThua (int n)
{
    //Kiểm tra điều kiện dừng
    if (n==1)
        return 1;
    else //gọi đệ quy
        return GiaiThua(n-1) * n ;
}
```

B6

```
long GiaiThua (int n)
{
    //Kiểm tra điều kiện dừng
    if (n==1)
        return 1;
    //gọi đệ quy
    return GiaiThua(n-1) * n ;
}
```

4. Các bước xây dựng hàm đệ quy

Ví dụ 2: viết hàm đệ quy tính tổng các số lẻ trong mảng 1 chiều các số nguyên

B1. Tham số cần truyền cho hàm là gì? Số lượng tham số là bao nhiêu?

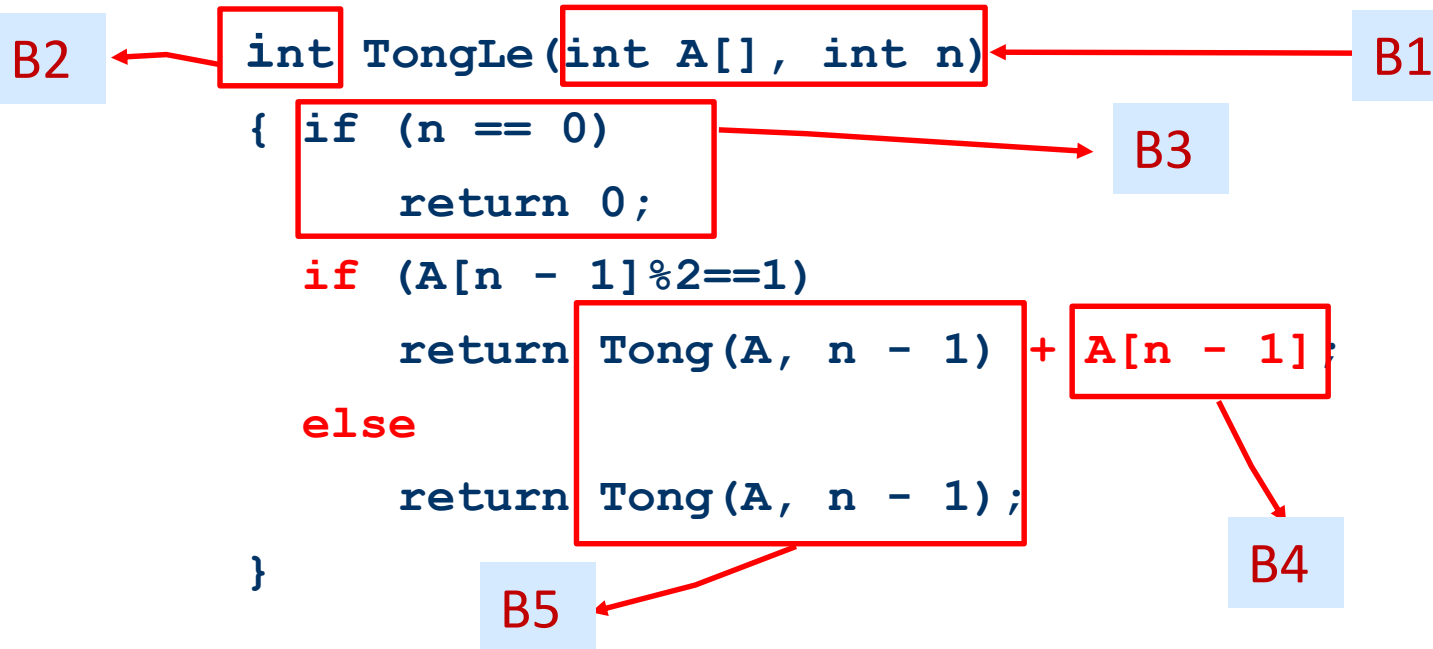
B2. Hàm có trả về giá trị hay không? (*void, int, float, ...*)

B3. Điều kiện dừng là gì? \Rightarrow Khi đó giá trị trả về là bao nhiêu?

B4. Phần không đệ quy là gì? (*$n-1$, $n/10$, $n\%10$, $A[n-1]$, ...*)

B5. Phần đệ quy là gì? \Rightarrow Kích thước nhỏ hơn của n là gì (*$n-1$, $n/10$, $n\%10$, ...*)?

B6. Tinh chỉnh lại nội dung hàm (nếu có thể)



4. Các bước xây dựng hàm đệ quy

Ví dụ 2: viết hàm đệ quy tính tổng các số lẻ trong mảng 1 chiều các số nguyên

B1. Tham số cần truyền cho hàm là gì? Số lượng tham số là bao nhiêu?

B2. Hàm có trả về giá trị hay không? (*void, int, float, ...*)

B3. Điều kiện dừng là gì? \Rightarrow Khi đó giá trị trả về là bao nhiêu?

B4. Phần không đệ quy là gì? (*$n-1$, $n/10$, $n\%10$, $A[n-1]$, ...*)

B5. Phần đệ quy là gì? \Rightarrow Kích thước nhỏ hơn của n là gì (*$n-1$, $n/10$, $n\%10$, ...*)?

B6. Tinh chỉnh lại nội dung hàm (nếu có thể)

```
int TongLe(int A[], int n)
{
    if (n == 0)
        return 0;
    if (A[n - 1] % 2 == 1)
        return TongLe(A, n - 1) + A[n - 1];
    else
        return TongLe(A, n - 1);
}
```

```
int TongLe(int A[], int n)
{
    if (n == 0)
        return 0;
    return Tong(A, n - 1) + (A[n - 1] % 2 == 1 ? A[n - 1] : 0);
}
```

5. MỘT SỐ LỖI THƯỜNG GẶP

- Không (hoặc chưa) xác định đúng điều kiện dừng.
- Không (hoặc chưa) xác định đúng phần đệ quy (bài toán đồng dạng đơn giản hơn).
- Thông điệp thường gặp là **StackOverflow** do:
 - Thuật giải đệ quy đúng nhưng số lần gọi đệ quy quá lớn làm tràn STACK.
 - Thuật giải đệ quy sai do không hội tụ hoặc không có điều kiện dừng.

6. NHẬN XÉT

– Ưu điểm

- Sáng sủa, dễ hiểu, nêu rõ bản chất vấn đề.
- Tiết kiệm thời gian thực hiện mã nguồn.
- Một số bài toán rất khó giải nếu không dùng đệ quy.

– Khuyết điểm

- Tốn nhiều bộ nhớ, thời gian thực thi lâu.
- Một số tính toán có thể bị lặp lại nhiều lần.
- Một số bài toán không có lời giải đệ quy.

6. Nhận xét

- Chỉ nên dùng phương pháp đệ quy để giải các bài toán kinh điển như giải các vấn đề “chia để trị”, “lần ngược”.
- Vấn đề đệ quy không nhất thiết phải giải bằng phương pháp đệ quy, có thể sử dụng phương pháp khác thay thế (*khử đệ quy*)
- Tiện cho người lập trình nhưng không tối ưu khi chạy trên máy.
- Bước đầu nên giải bằng đệ quy nhưng từng bước khử đệ quy để nâng cao hiệu quả.

7. THỰC HÀNH

Bài 1: Viết hàm đệ quy cho phép tính tổng các chữ số của một số nguyên n. Ví dụ $n=2019 \Rightarrow \text{Sum}=2+0+1+9=12$

```
int Tong (int n)
{
    int S=0;
    while(n>0)
    {
        S=S+ n%10;
        n=n/10;
    }
    return S;
}

int main()
{
    int n=2019;
    printf("Tong= %d",Tong(n));
    return 0;
}
```

```
int TongDQ(int n)
{
    if (n<10)
        return n;
    return TongDQ(n/10) + n%10;
}

void main()
{
    int n=2019;
    printf("Tong DQ=%d", TongDQ(n));
}
```

7. THỰC HÀNH

Bài 2: Đếm số lượng chữ số của số nguyên dương n

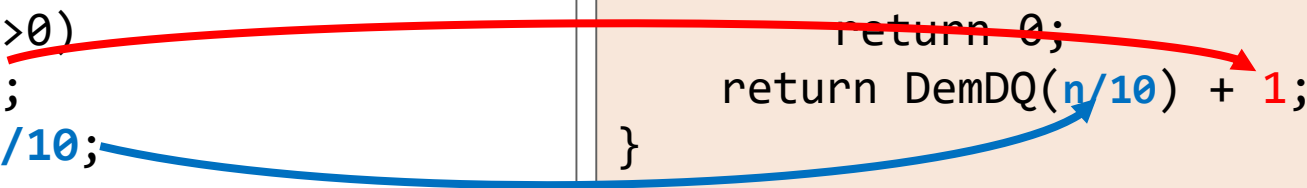
. Ví dụ: n=2019 => kết quả xuất ra: "Co 4 chu so"

```
int Dem (int n)
{
    int d=0;
    while(n>0)
    {
        d++;
        n=n/10;
    }
    return d;
}

int main()
{
    int n=2019;
    printf("Co %d chu so",Dem(n));
    return 0;
}
```

```
int DemDQ(int n)
{
    if (n<=0)
        return 0;
    return DemDQ(n/10) + 1;
}

void main()
{
    int n=2019;
    printf("Co %d chu so",DemDQ(n));
}
```



7. THỰC HÀNH

Bài 3: Viết hàm đệ quy tìm chữ số lớn nhất có trong số nguyên n.

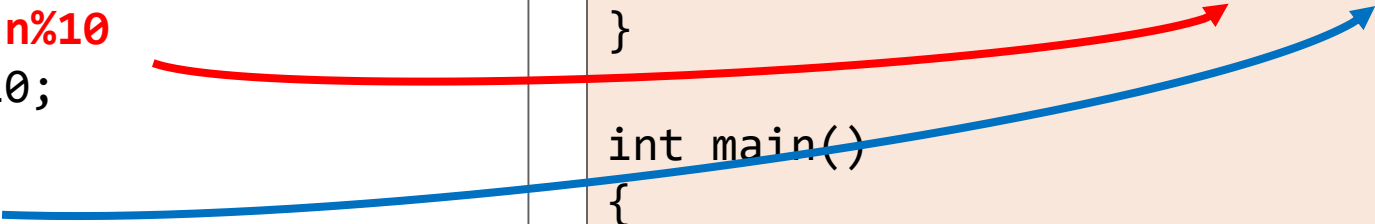
Ví dụ n=2019 → Lớn nhất là 9

```
int Max(int n)
{
    int k=n%10;
    while (n > 0)
    {
        if (k < n%10)
            k = n%10;
        n = n/10;
    }
    return k;
}

int main()
{
    int n = 2019;
    printf("Lon nhat la: %d",Max(n));
    return 0;
}
```

```
int MaxDQ(int n)
{
    if (n == 0)
        return 0;
    int k=MaxDQ(n/10);
    return (k < n%10) ? n%10 :k;
}

int main()
{
    int n = 2019;
    printf("Lon nhat la: %d",MaxDQ(n));
    return 0;
}
```



7. THỰC HÀNH

Bài 4: Viết hàm đệ quy xuất ngược các chữ số có trong số nguyên n.
Ví dụ n=2019 → xuất ngược thành 9102

```
void XuatNguoc(int n)
```

```
{
    while (n>0)
    {
        printf("%d", n%10);
        n= n/10;
    }
}
```

```
void main()
```

```
{
    int n=2019;
    XuatNguoc(n);
}
```

```
void XuatNguocDQ(int n)
```

```
{
    if (n<=0)
        return;
    printf("%d", n%10);
    XuatNguocDQ(n/10);
}
```

```
void main()
```

```
{
    int n=2019;
    XuatNguocDQ(n);
}
```

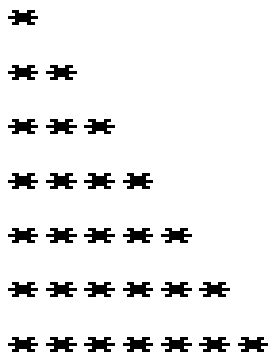
```
void XuatNguocDQ(int n)
```

```
{
    if ((n<10)
        printf("%d", n);
    else
    { printf("%d", n%10);
      XuatNguocDQ(n /10);
    }
}
```


Bài 5: In hình tam giác sau bằng cách đệ quy (VD: n=7)

```
#include <stdio.h>
#include <conio.h>
void InDong(int n)
{
    if (n < 1)
        return;
    InDong(n - 1);
    for (int i = 0; i<n; i++)
        printf("*");
    printf("\n");
}
void main()
{
    int n=7;
    InDong(n);
}
```

```
void VeHinh(long n,string s)
{
    if (n==0)
        return;
    cout << s <<'\n';
    VeHinh(n-1, s+"*");
}
```



7. THỰC HÀNH

Bài 6: Viết hàm đệ quy cho phép biểu diễn nhị phân của 1 số nguyên n, ví dụ: **n=13 → 1101**

```
#include <stdio.h>
#include <conio.h>

void XuatNhiPhan(int n)
{
    if (n/2>=1)
        XuatNhiPhan(n/2);
    printf("%d",n%2);
}

void main()
{
    int n=13;
    XuatNhiPhan(n);
}
```

7. THỰC HÀNH

Bài 7: Viết hàm đệ quy cho phép in chuỗi đảo ngược

- Trường hợp chung:
 - In ký tự cuối của chuỗi X
 - Lấy phần chuỗi còn lại
- Trường hợp suy biến: Nếu chuỗi rỗng thì không làm gì

```
#include <stdio.h>
#include <string.h>
#include <conio.h>
void InNguoc(char *X, int n)
{
    if (n>0)
    {
        printf("%c", X[n - 1]);
        InNguoc(X,n-1);
    }
}
void main()
{
    char *s = "Sai gon - TP Ho Chi Minh";
    int n= strlen(s);
    InNguoc(s, n);
}
```

Bài 8: Viết hàm đệ quy cho phép nhập số giây và chuyển thành giờ, phút, giây. Ví dụ: nhập 3665 -> 1 giờ 1 phút 5 giây

```
#include <stdio.h>
#include <conio.h>
void DoiGio(int n, int &h, int &m, int &s)
{
    if (n < 60)
        s = n;
    else
        if (n/3600>0)
        {
            h = n / 3600;
            return DoiGio(n%3600, h, m, s);
        }
        else
        {
            m = n/60;
            return DoiGio(n%60, h, m, s);
        }
}
void main()
{
    int n=3665, h=0, m=0, s=0;
    DoiGio(n, h, m, s);
    printf("%d giay = %d gio %d phut %d giay", n,h,m,s);
}
```

Bài 9: Viết hàm đệ quy cho phép kiểm tra xem một số có phải số nguyên tố hay không?

```
#include <stdio.h>
#include <conio.h>
int isPrime(int N, int &M)
{
    if (N == 1)    return 0;
    if (M == 1)    return 1;
    else
        if (N%M == 0)
            return 0;
        else
        {
            M--;
            isPrime(N,M);
        }
}

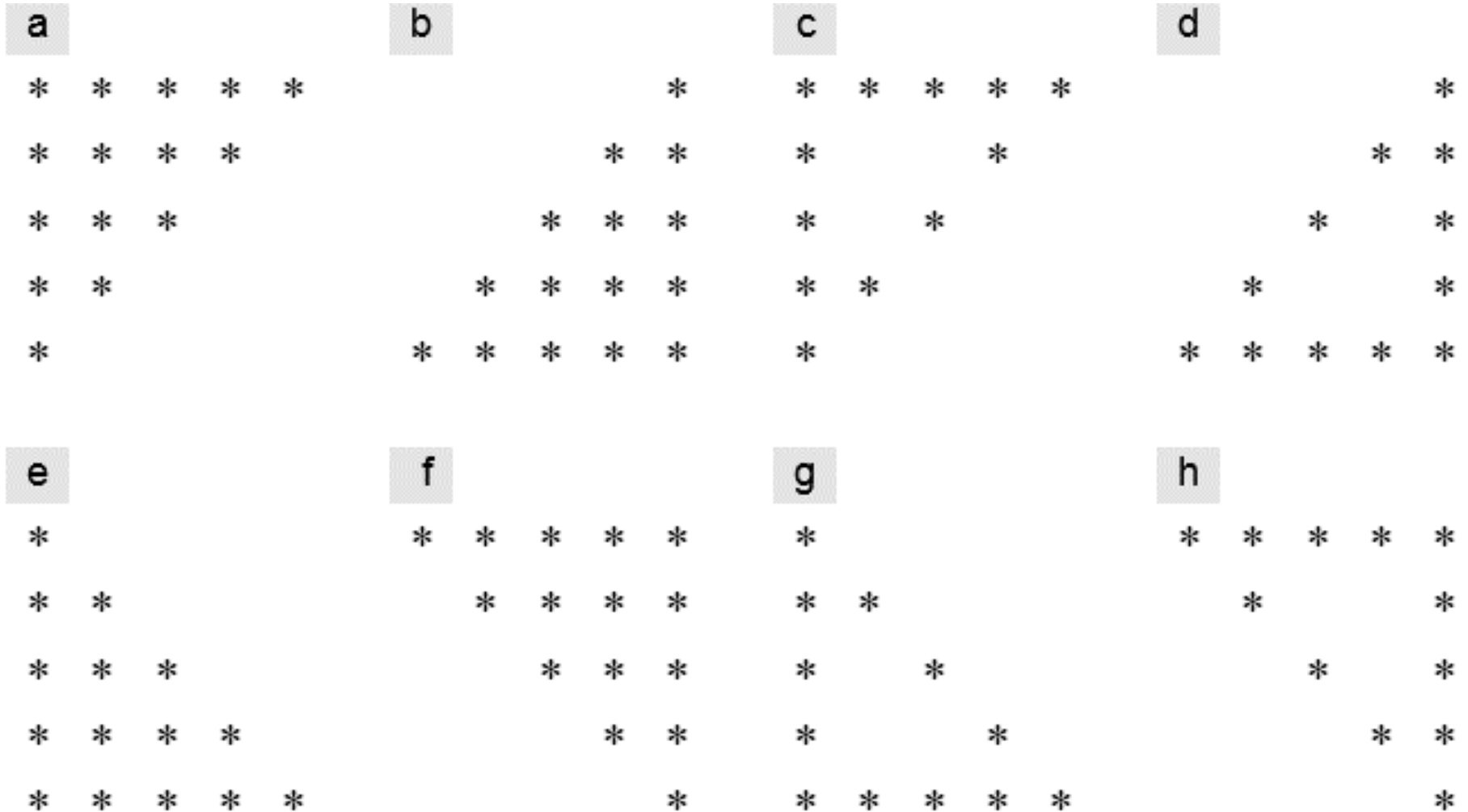
void main()
{
    int n=7, m = n - 1;
    if (isPrime(n,m)==1)
        printf("%d la SNT", n);
    else
        printf("%d KHONG la SNT", n);
}
```

8. BÀI TẬP

- i. Tính giá trị của x lũy thừa y
- ii. Tính giá trị của $n!$
- iii. Tính $P(n) = 1.3.5 \dots (2n+1)$, với $n \geq 0$
- iv. Tính $S(n) = 1+3+5+\dots+(2 \times n+1)$, với $n \geq 0$
- v. Tính $S(n) = 1-2+3-4+\dots+(-1)^{n+1}n$, với $n > 0$
- vi. Tính $S(n) = 1+1.2+1.2.3+\dots+1.2.3 \dots n$, với $n > 0$
- vii. Tính $S(n) = 1^2 + 2^2 + 3^2 + \dots + n^2$, với $n > 0$
- viii. Kiểm tra xem số nguyên n chỉ chứa toàn số lẻ hay không?

8. BÀI TẬP

viii. Viết chương trình cho người dùng nhập cạnh (n) của tam giác.
Giả sử với $n=5$, chương sẽ lần lượt in ra các hình sau:



8. Bài tập

Mảng 1 chiều: cho mảng 1 chiều các số nguyên **A** và **n** là số lượng phần tử có trong mảng. Viết các hàm không đệ quy và đệ quy cho các yêu cầu sau:

- ix. Tạo mảng với giá trị ngẫu nhiên. **void TaoMang(int A[], int n)**
- x. Xuất giá trị đang có trong mảng A theo thứ tự thông thường (từ trái sang phải) **void XuatLtoR(int A[], int n)**
- xi. Xuất giá trị đang có trong mảng A theo thứ tự từ phải sang trái **void XuatRtoL (int A[], int n)**
- xii. Xuất giá trị lẻ có trong mảng A **void XuatLe (int A[], int n)**
- xiii. Tính tổng các số có trong mảng A. **int Tong (int A[], int n)**
- xiv. Đếm các số lẻ có trong mảng A. **int DemLe (int A[], int n)**
- xv. Đếm các số có giá trị =X có trong mảng A. **int DemX (int A[], int n, int X)**
- xvi. Tìm số chẵn đầu tiên có trong mảng. Nếu trong mảng không có số chẵn, hàm trả về -1. **int TimChan (int A[], int n)** Tìm giá trị X có xuất hiện trong mảng hay không? **bool TimX (int A[], int n)**
- xvii. Tính trung bình các số có trong mảng. **float TBinh (int A[], int n)**

8. Bài tập

Ma trận vuông: cho ma trận vuông chứa các số nguyên ***A*** và ***n*** là cạnh của ma trận vuông. Viết các hàm không đệ quy và đệ quy cho các yêu cầu sau:

xvii. Tính tổng các phần tử trên đường chéo chính

xviii. Tính tổng các phần tử trên đường chéo phụ.

xix. Tính tổng các phần tử trong mảng 2 chiều các số nguyên

