PHẦN I: PHẦN MỞ ĐẦU

----------📖----------

I. LÝ DO CHỌN ĐỀ TÀI :

Ngôn ngữ lập trình là một trong những nội dung được đưa vào dạy chính thức ở bộ môn Tin học ở nhà trường phổ thông. Đặc biệt, các lớp ở khối chuyên Tin thì yêu cầu học sinh phải nắm vững các kiến thức về tư duy thuật toán, sử dụng ngôn ngữ lập trình thành thạo để có thể giải được các bài toán trong Tin học.

Như ta đã biết, các phép lặp là một trong những kỹ thuật được dùng để giải các bài toán trong Tin học và đã được giới thiệu ở phần tin học cơ sở. Với các phép lặp, ta giải bài toán bằng cách thực hiện liên tiếp một số các câu lệnh trong vòng lặp cho tới khi một điều kiện nào đó được thỏa mãn.

Một kỹ thuật lập trình được sử dụng để thay thế cho các phép lặp đó là kỹ thuật đệ quy. Mặt khác, trong thực tế có rất nhiều bài toán đòi hỏi sự lặp đi lặp lại một cách phức tạp. Và đệ quy cung cấp cho ta cơ chế giải quyết bài toán phức tạp một cách đơn giản. Hơn nữa, đệ quy còn thích hợp để giải quyết các bài toán có bản chất đệ quy và rất nhiều bài toán cho đến nay vẫn chưa có lời giải phi đệ quy.

Để có thể hiểu rõ hơn về kỹ thuật đệ quy, ứng dụng của đệ quy để giải các bài toán trong Tin học chúng em chọn đề tài: "**Tìm hiểu về đệ quy**" làm đề tài nghiên cứu khoa học của mình.

II. MỤC ĐÍCH NGHIÊN CỨU:

Đề tài : "**Tìm hiểu về đệ quy**" nhằm đạt mục đích sau: Giúp cho bản thân chúng em có thể hiểu rõ hơn, sâu hơn về kỹ thuật đệ quy, ứng dụng đệ quy để giải các bài toán trong Tin học, từ đó nâng cao khả năng tự học, khả năng lập trình của bản thân.

III. PHƯƠNG PHÁP NGHIÊN CỨU:

Khi nghiên cứu đề tài này chúng em sử dụng các phương pháp sau:

- ✓ Phương pháp nghiên cứu lý thuyết
- ✓ Sử dụng phương pháp thu thập thông tin và tổng hợp, phân tích, so sánh dựa trên lý thuyết về *đệ quy*.

IV. ĐỐI TƯỢNG NGHIÊN CỨU:

Đối tượng nghiên cứu của đề tài này đó chính là *lý thuyết về đệ quy và ứng dụng của đệ quy để giải các bài toán trong Tin học* và các tài liệu, các thông tin liên quan.

PHẦN II: PHẦN NỘI DUNG

1. Khái niệm về đệ quy

1.1. Khái niệm về hình thức đệ quy

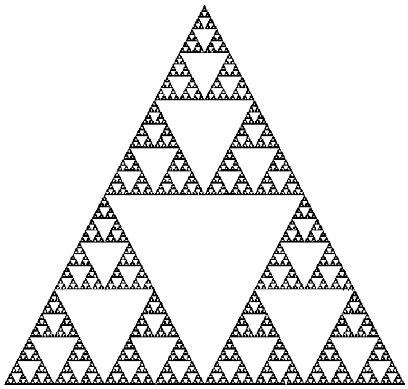
Trong toán học và khoa học máy tính, các tính chất (hoặc cấu trúc) được gọi là đệ quy nếu trong đó một lớp các đối tượng hoặc phương pháp được xác định bằng việc xác định một số rất ít các trường hợp hoặc phương pháp đơn giản (thông thường chỉ một) và sau đó xác định quy tắc đưa các trường hợp phức tạp về các trường hợp đơn giản.

◈ Chẳng hạn, định nghĩa sau là định nghĩa đệ quy của tổ tiên:

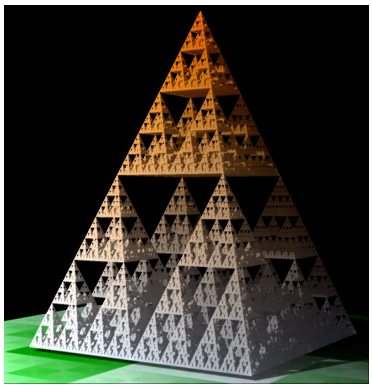
{ Bố mẹ của một người là tổ tiên của người ấy (trường hợp cơ bản).
{ Bố mẹ của tổ tiên một người bất kỳ là tổ tiên của người ấy (bước đệ quy).

Các định nghĩa kiểu như vậy cũng thường thấy trong toán học (chính là quy nạp toán học)

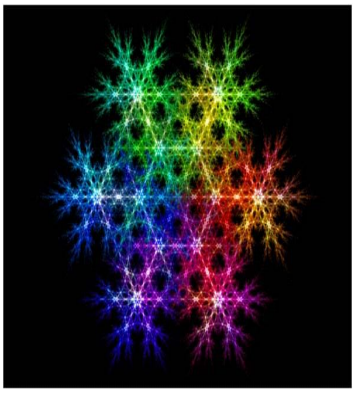
◈ Một số hình ảnh ứng dụng của đệ quy trong tạo hình:



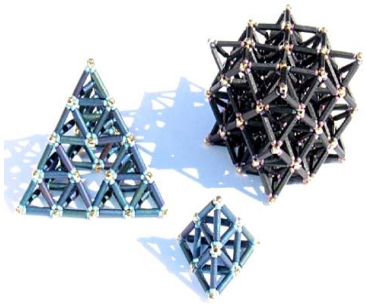
Tam giác sierpinski

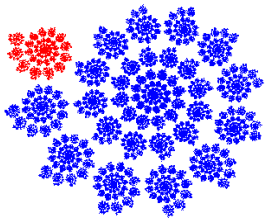


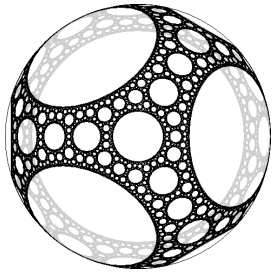
Kim tự tháp sierpinski



Bông tuyết Koch







Sierpinski cookies *Sierpinski 2 chiều* *Sierpinski tròn (carpet)*

1.2. Khái niệm về đệ quy

Từ những định nghĩa trên, ta có thể rút ra khái niệm cơ bản theo 2 cách:

1. Đệ quy (trong tiếng Anh là *recursion*) là phương pháp dùng trong các chương trình máy tính trong đó có một hàm tự gọi chính nó.

2. Một khái niệm X được định nghĩa theo đệ quy nếu trong định nghĩa X có sử dụng chính khái niệm X.

◈ ***Ví dụ:*** Sau đây là một số ví dụ điển hình về đệ quy:

1. Định nghĩa về số tự nhiên:

$$\begin{cases} 0 \text{ là một số tự nhiên} \\ n \text{ là một số tự mhiên khi n-1 là 1 số tự nhiên} \end{cases}$$

2. Định nghĩa về giai thừa n!:

$$\begin{cases} 0! = 1 \\ \text{Nếu } n>0 \text{ thì } n!=n*(n-1)! \end{cases}$$

1.3. Các bước để giải bài toán đệ quy

- Thông số hóa bài toán (hiểu bài toán)
- Tìm các điều kiện biên (chặn), tìm giải thuật cho các tình huống này
- Tìm giải thuật tổng quát theo hướng đệ quy lui dần về tình huống bị chặn.

◈ ***Ví dụ:*** Bài toán tính giai thừa của một số tự nhiên n (tính *n*!):

- Thông số hóa bài toán: Cách tính giai thừa n!:

$$\begin{cases} 0! = 1 \\ \text{Nếu } n>0 \text{ thì } n!=n*(n-1)! \end{cases}$$

- Tìm các điều kiện biên (chặn), tìm giải thuật cho các tình huống này:

```
If n = 0 then  gt : =1
```

- Tìm giải thuật tổng quát theo hướng đệ quy lui dần về tình huống bị chặn.

```
If n > 0 then  gt := n*gt(n - 1);
```

2. Giải thuật đệ quy

2.1. Khái niệm giải thuật đệ quy

Nếu lời giải của một bài toán T được thực hiện bằng lời giải của một bài toán T có dạng giống như T, thì đó là một lời giải đệ quy Giải thuật tương ứng với lời giải như vậy gọi là giải thuật đệ quy.

Thoạt nghe thì các bạn thấy có vẻ hơi lạ nhưng điểm mấu chốt cần lưu ý là: T' tuy có dạng giống như T, nhưng theo một nghĩa nào đó, nó phải "nhỏ" hơn T.

Trong khoa học máy tính có một phương pháp chung để giải các bài toán trong Tin học là chia bài toán thành các bài toán con đơn giản hơn cùng loại. Phương pháp này được gọi là kỹ thuật lập trình chia để trị. Chính nó là chìa khóa để thiết kế nhiều giải thuật quan trọng, là cơ sở của quy hoạch động.

2.2. Ví dụ

◈ ***Ví dụ 1:*** Tính giai thừa của một số tự nhiên n (tính *n*!):

```
If n = 0 then  gt : =1
    Else  gt := n*gt(n - 1);
```

Bài toán tính n! được chia nhỏ như sau:

Ta nhận thấy rằng, để tính n! ta cần phải tính được (n-1)!, để tính được (n-1)! Thì ta phải tính được (n-2)!, … cho đến khi chúng ta sẽ gặp côngviệc tính 1!, mà 1!=1, vì thế cứ thực hiện tính ngược trở lên chúng ta sẽ tính được n!.

◈ ***Ví dụ 2:*** Hãy xét bài toán tìm một từ trong một quyển từ điển.

Có thể nêu giải thuật như sau:

if Từ điển là một trang **then** Tìm từ trong trang này

else begin

Mở từ điển vào trang giữa

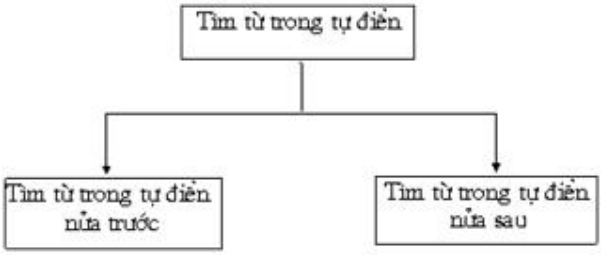
Xác định xem nửa nào của từ điển chứa từ cần tìm

if Từ đó nằm ở nửa trước của từ điển ***then*** Tìm từ đó trong nửa trước

else Tìm từ đó trong nửa sau

end;

Có thể hình dung chiến thuật tìm kiếm này một cách khái quát như sau:



Ta thấy có hai điểm chính cần lưu ý:

- Sau mỗi lần từ điển được tách đôi thì một nửa thích hợp sẽ lại được tìm kiếm bằng một "chiến thuật" như đã dùng trước đó.

- Có một trường hợp đặc biệt, khác với mọi trường hợp trước, sẽ đạt được sau nhiều lần tách đôi, đó là trường hợp tự điển chỉ còn duy nhất một trang. Lúc đó việc

tách đôi ngừng lại và bài toán trở thành đủ nhỏ để ta có thể giải quyết trực tiếp bằng cách tìm từ mong muốn trên trang đó chẳng hạn, bằng cách tìm tuần tự. Trường hợp đặc biệt này được gọi là trường hợp suy biến.

Có thể coi đây là một "chiến thuật " kiểu "chia để trị". Bài toán được tách thành bài toán nhỏ hơn và bài toán nhỏ hơn lại được giải quyết với thuật chia để trị như trước, cho tới khi xuất hiện trường hợp suy biến.

Ta thể hiện giải thuật tìm kiếm này dưới dạng một thủ tục:

Procedure TIMKIEM (TD,Tu)
{TD được coi là đầu mối để truy nhập được vào tự điển đang xét, Tu chỉ từ cần tìm}
Begin
if Tự điển chỉ còn là một trang
then Tìm từ Tu trong trang này
else begin
Mở tự điển vào trang giữa
Xác định xem nửa nào của tự điển chứa từ Tu
if Tu nằm ở nửa trước của tự điển
then TIMKIEM (TD 1,Tu)
else TIMKIEM (TD 2,Tu)
end;
{TD 1 và TD 2 là đầu mối để truy nhập được vào nửa trước và nửa sau của từ điển}
End;

3. Chương trình con đệ quy

3.1. Khái niệm chương trình con đệ quy

Là một chương trình con (hàm, thủ tục) mà trong thân của nó có lời gọi đến chính nó (hay còn gọi là lời gọi đệ quy) với kích thước nhỏ hơn của tham số.

◈ ***Ví dụ***: Hàm tính giai thừa của một số tự nhiên n (tính *n*!):

```
Function gt(n: Word): Longint;
 Begin
   If n = 0 then  gt : =1
    Else  gt := n*gt(n - 1);
  End;
```

3.2. Cấu trúc chính của chương trình con đệ quy

Một chương trình con đệ qui căn bản gồm hai phần.

- *Phần neo (phần suy biến, cơ sở):* chứa các tác động của hàm hoặc thủ tục với một số giá trị cụ thể ban đầu của tham số.

  ◈ *Ví dụ:* `If n:=0 then GT:= 1;`

- Phần đệ quy (phần tổng quát, hạ bậc) : định nghĩa tác động cần được thực hiện cho giá trị hiện thời của các tham số bằng các tác động đã được định nghĩa trước đây với kích thước tham số nhỏ hơn.

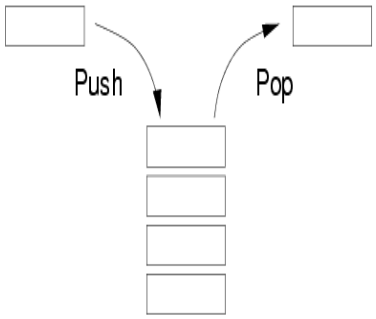
  ◈ *Ví dụ:* `GT:=n*GT(n-1);`

4. Nguyên tắc hoạt động của giải thuật đệ quy

4.1. Khái niệm stack:

Stack là một cấu trúc lưu trữ, hoạt động theo nguyên tắc sau:

- Mỗi lần nộp vào hoặc lấy ra chỉ thực hiện với một phần tử.
- Phần tử nộp vào sau sẽ được lấy ra trước.



Stack

4.2. Nguyên tắc hoạt động:

- Khi thực hiện một giải thuật đệ quy thì các bước của giải thuật đệ quy sẽ lần lượt được thực hiện tuần tự.
- Khi gặp lời gọi đệ quy thì trước khi thực hiện lời gọi đệ quy, đoạn mã lệnh chưa được thực hiện xong cùng với các đối tượng dữ liệu liên quan tại thời điểm này sẽ được lưu vào stack.
- Đến lúc nào đó không thể thực hiện lời gọi đệ quy nữa thì các đối tượng được lưu trong stack sẽ lần lượt được lấy ra để xử lý.

◈ ***Ví dụ:*** Trong ví dụ trên, qui trình thực hiện như sau:

- Khi có lệnh gọi hàm, chẳng hạn: `x := gt(3);` thì máy sẽ ghi nhớ là: `gt(3) := 3 * gt(2);` và đi tính `gt(2)`
- Kế tiếp máy lại ghi nhớ: `gt(2):= 2*gt(1);` và đi tính `gt(1)`
- Theo định nghĩa của hàm thì khi `gt(1):= 1;` máy sẽ quay ngược lại: `gt(2):= 2 * 1;` và cho kết quả là 2
- Tiếp tục: `gt(3) := 3 * 2;` cho kết quả là 6
- Như vậy kết quả cuối cùng trả về là 6. Ta có: 3! = 6.

5. Ưu điểm và hạn chế của đệ quy

5.1. Ưu điểm:

- Mô tả được một số thao tác tính toán thông qua một đoạn lệnh ngắn, làm cho chương trình ngắn gọn, dễ hiểu, lộ rõ bản chất đệ quy.
- Rất thuận tiện để giải quyết các bài toán có bản chất đệ quy.
- Hiện nay vẫn có nhiều thuật toán chưa có lời giải đệ quy.
- Nhiều giải thuật rất dễ mô tả dạng đệ quy nhưng lại rất khó mô tả với giải thuật không-đệ-quy.
- Một chương trình viết theo giải thuật có tính đệ qui sẽ mang tính "người" hơn, do đó sẽ sáng sủa, dễ hiểu, nêu bật được bản chất của vấn đề.

5.2. Hạn chế:

- Vừa tốn bộ nhớ, chương trình chạy chậm
- Do đệ quy lưu trữ các dữ liệu trung gian vào Stack nên nếu Lưu nhiều bộ dữ liệu lớn trên stack nên có thể gây ra hiện tượng tràn Stack

6. Một số bài toán về đệ quy

Bài 1: Tính lũythừa: a^n (với n là số nguyên dương)

```
Program Giai_thua;
Const fi='Giaithua.inp';
         fo='Giaithua.out';
Var a:longint;n: Integer;
       f:text;
Procedure Doc;
Begin
     Assign(f,fi);
     Reset(f);
     Readln(f,a,n);
     Close(f);
End;
Function lt(a,n: Word): Longint;
Begin
    If n = 0 then lt := 1
            else lt := a * lt(a,n - 1);
End;
Procedure Ghi;
Begin
     Assign(f,fo);
```

```
     Rewrite(f);
     Writeln(f,lt(a,n));
     Close(f);
End;
Begin
      Doc;Ghi;
End.
```

Bài 2: **Tìm số fibonaxi thứ n**

```
Program Fibonaxi;
Const fi='Fibonxi.inp';
         fo='Fibonaxi.out';
Var a:longint;
       f:text;
Procedure Doc;
Begin
     Assign(f,fi);
     Reset(f);
     Readln(f,a);
     Close(f);
End;
Function Fibo(n:longint):longint;
Begin
        If ((n=0)or(n=1)) then Fibo:=1
              else Fibo:=Fibo(n-2)+Fibo(n-1);
End;
Procedure Ghi;
Begin
     Assign(f,fo);
     Rewrite(f);
     Writeln(f,Fibo(a));
     Close(f);
End;
Begin
```

```
     Doc;Ghi;
End.
```

<u>Bài 3:</u> Tính tổng S=1+2+3+...+n ?

```
Program TongS;
Const fi='Tong.inp';
         fo='Tong.out';
Var n:longint;
       f:text;
Procedure Doc;
Begin
     Assign(f,fi);
     Reset(f);
     Readln(f,n);
     Close(f);
End;
Function S(k:longint):longint;
Begin
     If k=1 then S:=1
              else If k>1 then S:=S(k-1) + k;
        End;
Begin
Procedure Ghi;
Begin
     Assign(f,fo);
     Rewrite(f);
     Writeln(f,S(n));
     Close(f);
End;
Begin
     Doc;Ghi;
End.
```

<u>Bài 4</u>: Tính S=xsinx+$x^2\sin^2 x$+…+$x^n\sin^n x$?

```
Program Tong_sinx;
Const fi='tongsin.inp';
        fo='tongsin.out';
Var  f:text;
       a,b:longint;
Procedure Doc;
Begin
     Assign(f,fi);
     Reset(f);
     Readln(f,a,b);
     Close(f);
End;
  Var      a,b:longint;
      f:text;
  Function  tinh(n,x:longint):real;
  Begin
      If n=1 then tinh:=x*sin(x)
           else tinh:=x*sin(x)*(1+tinh(n-1,x));
  End;
Procedure Ghi;
Begin
     Assign(f,fo);
     Rewrite(f);
     Writeln(f,Tongsin(a,b));
     Close(f);
End;
  Begin
   Doc;Ghi;
  End.
```

Bài 5: **Tìm số đảo ngược**

```
Program So_dao_nguoc;
```

```
Const fi='Daonguoc.inp';
         fo='Daonguoc.out';
Var      a:longint;
            f:text;
Procedure Doc;
Begin
     Assign(f,fi);
     Reset(f);
     Readln(f,a);
     Close(f);
End;
Procedure Daoso(n: longint);
Begin
     Assign(f,fo);
     Rewrite(f);
     If n > 0 then
     Begin
           Write(f,n mod 10);
           Daoso(n div 10);
     End;
     Close(f);
End;
Begin
     Doc;Daoso(a);
End.
```

<u>Bài 6:</u> Tìm ước chung lớn nhất của 2 số nguyên dương.

```
Program Tim_UCLN;
Const fi='UCLN.inp';
           fo='UCLN.out';
Var x,y:longint;
       f:text;
Procedure Doc;
Begin
```

```
     Assign(f,fi);
     Reset(f);
     Readln(f,x,y);
     Close(f);
End;
Function UCLN(m,n:longint):longint;
Begin
  If(m=n) then UCLN:= m
  else
     If (m>n) then UCLN:=UCLN(m-n,n)
      else UCLN:=UCLN(m,n-m);
End;
Procedure Ghi;
Begin
     Assign(f,fo);
     Rewrite(f);
     Writeln(f,UCLN(x,y));
     Close(f);
End;
Begin
     Doc;Ghi;
End.
```

<u>Bài 7:</u> Tính tổng các chữ số của 1 số nguyên dương.

```
Program Tong_chu_so;
Const fi='sum.inp';
           fo='sum.out';
Var n:longint;
        f:text;
Procedure Doc;
Begin
     Assign(f,fi);
     Reset(f);
     Readln(f,n);
```

```
     Close(f);
End;
Function tong(m:longint):longint;
Var t:longint;
Begin
     If m div 10=0 then tong:=m mod 10
        else tong:=m mod 10 + tong(m div 10);
End;
Procedure Ghi;
Begin
     Assign(f,fo);
     Rewrite(f);
     Writeln(f,tong(n));
     Close(f);
End;
Begin
     Doc;Ghi;
End.
```

<u>Bài 8:</u> Bài toán Tháp Hà Nội:

◈ ***Đề bài:*** Có 3 cái cọc, đánh dấu A, B, C, và N cái đĩa. Mỗi đĩa đều có một lỗ chính giữa để đặt xuyên qua cọc, các đĩa đều có kích thước khác nhau. Ban đầu tất cả đĩa đều được đặt ở cọc thứ nhất theo thứ tự đĩa nhỏ hơn ở trên.

Yêu cầu của bài là chuyển tất cả các đĩa từ cọc A qua cọc C với ba ràng buộc như sau:

1. Mỗi lần chỉ chuyển được một đĩa.
2. Trong quá trình chuyển đĩa có thể dùng cọc còn lại (B) để làm cọc trung gian.
3. Chỉ cho phép đặt đĩa có bán kính nhỏ hơn lên đĩa có bán kính lớn hơn.

◈ ***Phân tích bài toán:***

Trong bài toán trên hình dung một lời giải tổng quát cho trường hợp tổng quát N đĩa là không dễ dàng.

Hãy bắt đầu với các trường hợp đơn giản:

- Với N = 1: Chỉ cần chuyển đĩa này từ cọc A qua cọc C là xong.

- Với N = 2: Để đảm bảo ràng buộc thứ hai ta bắt buộc chuyển đĩa trên cùng từ cọc A qua cọc B. Chuyển tiếp đĩa còn lại từ cọc A qua cọc C. Chuyển tiếp đĩa đang ở cọc B sang cọc C.

- Với N = 3: Ta phải thực hiện 7 bước như sau:

	A	B	C
Trạng thái ban đầu			
Bước 1: Chuyển một đĩa từ A qua C.			
Bước 2: Chuyển một đĩa từ A qua B.			
Bước 3: Chuyển một đĩa từ C qua B.			
Bước 4: Chuyển một đĩa từ A qua C.			
Bước 5: Chuyển một đĩa từ B qua A.			
Bước 6: Chuyển một đĩa từ B qua C.			
Bước 7: Chuyển một đĩa từ A qua C.			

** <u>Nhận xét:</u>*

Ở kết quả của bước thứ ba. Đây là một kết quả quan trọng vì nó cho ta thấy từ trường hợp N=3 bài toán đã được phân chia thành hai bài toán với kích thước nhỏ

hơn: đó là bài toán chuyển 1 đĩa từ cọc A qua cọc C lấy cọc B làm trung gian và bài toán chuyển 2 đĩa (dời) từ cọc B sang cọc C lấy cọc A làm trung gian. Hai bài toán con này đã biết cách giải (trường hợp N=1 và trường hợp N=2).

Nhận xét đó cho ta gợi ý trong trường hợp tổng quát:

- Bước 1: Dời (N-1) đĩa trên cùng từ cọc A sang cọc B lấy cọc C làm trung gian.
- Bước 2: Chuyển 1 đĩa dưới cùng từ cọc A sang cọc C.
- Bước 3: Dời (N-1) đĩa đang ở cọc B sang cọc C lấy cọc A làm trung gian.

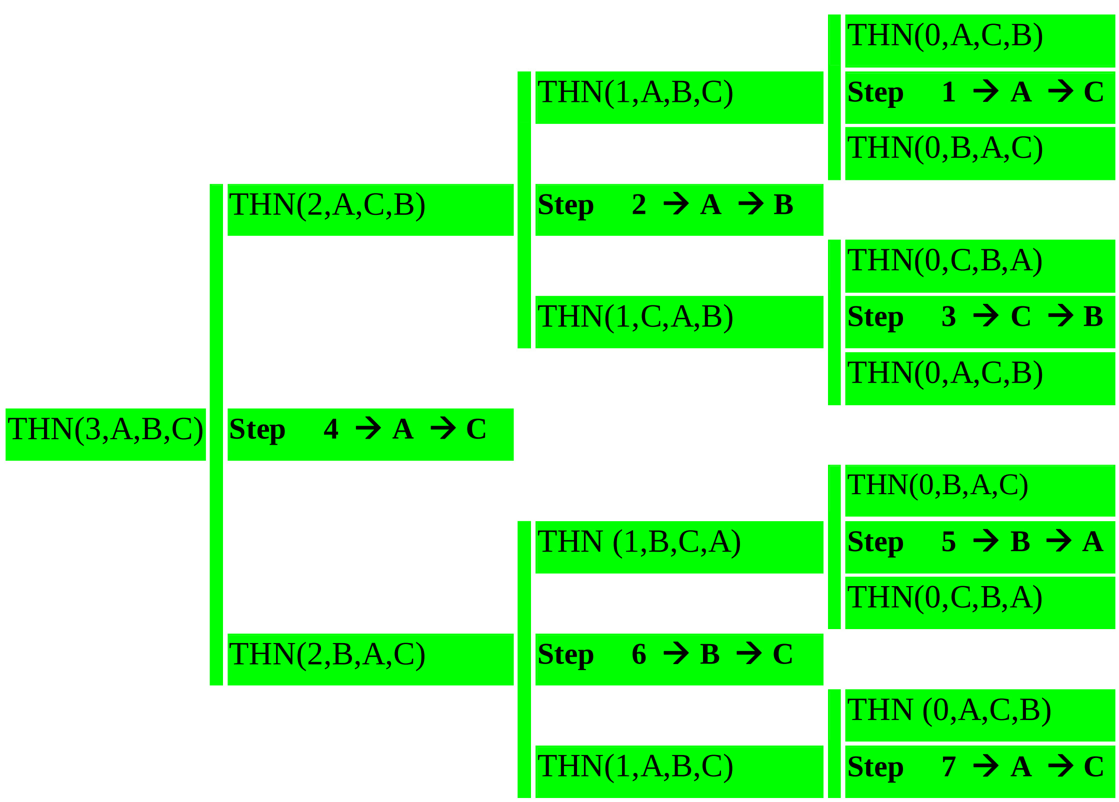
Như vây, bài toán đối với N đĩa ở trên được "đệ qui" về hai bài toán (N-1) đĩa và bài toán 1 đĩa. Quá trình đệ qui sẽ dừng lại khi N=0 (không còn đĩa để dời hoặc chuyển).

◈ ***Cài đặt chương trình:***

```
Program ThapHN;
Const   fi='THAP.INP';
         f0='THAP.OUT';
Var n,d:integer ;    f:text;
{— — — — — — — — — — — — — — — — — — — — — — — — — — — }
Procedure Readf;
Begin
    Assign(f,fi);
    Reset(f);
    Readln(f,n);
    d:=0;
    close(f);
end;
{— — — — — — — — — — — — — — — — — — — — — — — — — — — }
Procedure THN(n:integer;A,B,C:char);
  Begin
    If n>0 then Begin
         THN(n-1,A,C,B);
         inc(d);
         Writeln(f,'Step',d:5,' -×- ',A,' -¯> ',C);
         THN(n-1,B,A,C);
         end;
```

```
  end;
{————————————————————————————————————————— }
Procedure Writef;
Begin
    Assign(f,f0);
    Rewrite(f);
    THN(n,'A','B','C');
    Writeln(f,'Co   ',d,'  buoc');
    Close(f);
end;
{————————————————————————————————————————— }
BEGIN
        Readf;
        Writef;
END.
```

◈ Áp dụng chương trình này cho trường hợp N=3 ta có quá trình gọi đệ qui như sau:



THN(0,B,A,C)

7. Đệ quy quay lui (Back tracking)

7.1. Tổng quan về đệ quy quay lui

Trong lập trình, phương pháp giải một bài toán tổng quát rất được chú ý. Đó là việc xác định các giải thuật để tìm lời giải cho một số bài toán nào đó không theo một luật tính toán cố định bằng phương pháp "*Try and Error*" (Thử và sai).

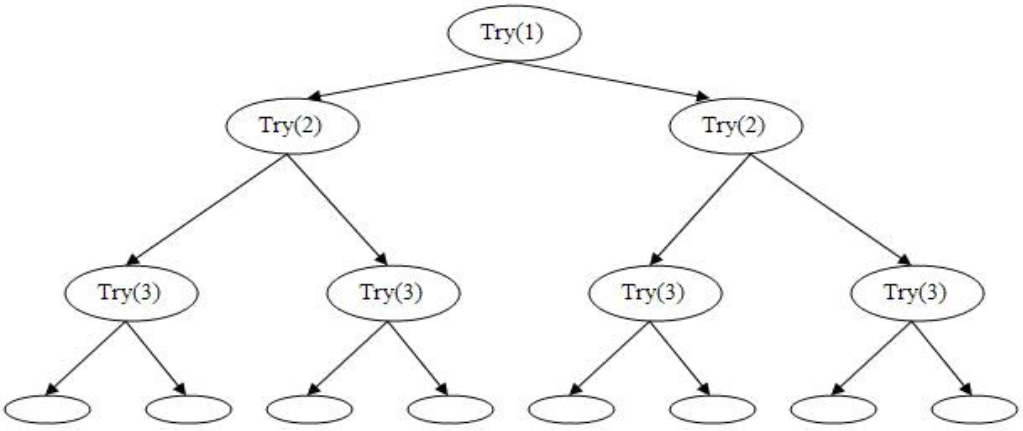
Nét đặc trưng của phương pháp này là ở chỗ các bước đi đến lời giải hoàn toàn bằng cách làm thử. Nếu có một lựa chọn được chấp nhận thì ghi nhớ các thông tin cần thiết các bước thử tiếp theo. Trái lại, nếu không có một lựa chọn nào thích hợp thì làm lại bước trước, xoá bớt các ghi nhớ và quay về chu trình thử với các lựa chọn còn lại. Hành động này được gọi là quay lui (Back tracking) và các giải thuật thể hiện phương pháp này gọi là các giải thuật quay lui.

7.2. Giải thuật tổng quát

```
Procedure Try(j); {Chọn thực hiện bước thứ j}
Begin
  For CÁC PHƯƠNG ÁN CHỌN do
   If CHỌN ĐƯỢC then
     Begin
     - THỰC HIỆN BƯỚC ĐI THỨ j;
     - IF THÀNH CÔNG then THÔNG BÁO KẾT QUẢ
       Else Try(j+1);
     - HỦY BƯỚC ĐI THỨ j; {Quay lui}
  End;
Endl;
```

- Thủ tục trên sẽ được khởi động bởi lệnh: `Try(1);`

Ta có thể trình bày quá trình tìm kiếm lời giải quả thuật toán quay lui bằng cây sau:



8. Một số bài toán về đệ quy quay lui

8.1. Bài toán Tìm hoán vị

◈ ***Đề bài:*** Nhập từ bàn phím một số tự nhiên N. Hãy liệt kê tất cả các hoán vị của tập số {1, 2, 3..., N}.

◈ ***Phân tích bài toán:***

- Với N=1 thì chỉ có 1 hoán vị duy nhất là: 1.
- Với N=2 thì có 2 hoán vị là:
  + Khi chọn 1 là số đứng đầu, ta có hoán vị: 1 2
  + Khi chọn 2 là số đứng đầu, ta có hoán vị: 2 1
- Với N=3:
  + Khi chọn 1 là số đứng đầu, ta có 2 hoán vị: 1 2 3
    1 3 2
  + Khi chọn 2 là số đứng đầu, ta có 2 hoán vị: 2 1 3
    2 3 1
  + Khi chọn 3 là số đứng đầu, ta có 2 hoán vị: 3 2 1
    3 1 2

.....

Với tập hợp N số, khi chọn một trong các số từ 1 đến N, ta sẽ có các hoán vị gồm: số đứng đầu đã chọn + một trong các hoán vị của tập số {1..N}.

Như vậy, ta nhận thấy có thể giải bài toán này bằng cách dùng thuật toán đệ quy quay lui.

Ta có:

- *Try(j):* thực hiện chọn số thứ j cho hoán vị.
- *Các phương án chọn*: Có N phương án chọn, i = 1, 2, 3..., N.
- *Chọn được*: khi giá trị i chưa được chọn.

Tổ chức lưu trữ: Dùng một mảng một chiều KT có N phần tử mang kiểu dữ liệu Boolean. Giá trị khởi tạo ban đầu của các phần tử đều là True

KT	True	True	True	...	True
i =	1	2	3	...	N

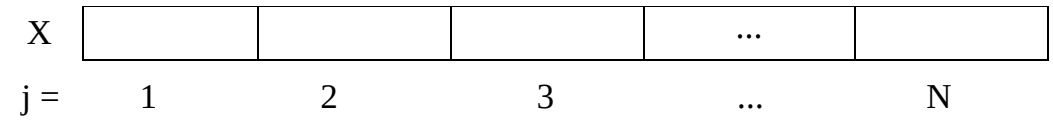
Giá trị i chưa được chọn khi KT[i]= True

- *Thực hiện bước đi thứ j*:

+ Đánh dấu giá trị được chọn: KT[i]:= False.

+ Gán giá trị của i cho số thứ j:

Tổ chức lưu trữ: Dùng một mảng một chiều X có N phần tử để lưu các giá trị đã được chọn:



Lưu vết: X[j]:=i;

- *Thành công*: khi đã chọn đủ N số (j=N).
- *Hủy bước đi thứ j*: gán giá trị True cho KT[i]: KT[i]: =True;

◈ ***Cài đặt chương trình:***

```
Program Hoanvi;
const maxn=100;
type mmc=array[1..maxn] of byte;
     mang=array[1..maxn] of boolean;
var x:mmc;kt:mang;n:byte;
 procedure try(j:byte);
  var i,k:byte;
  begin
   for i:=1 to n do
    if kt[i]=true then
     begin
      kt[i]:=false;
      x[j]:=i;
     if j=n then
      begin
       for k:=1 to n do write (x[k],' ');
       writeln;
      end
     else try(j+1);
      kt[i]:=true;
     end;
  end;
begin
```

```
    writeln('nhap so n');
    readln(n);
    writeln('cac hoan vi la:');
    try(1);
    readln;
  end.
```

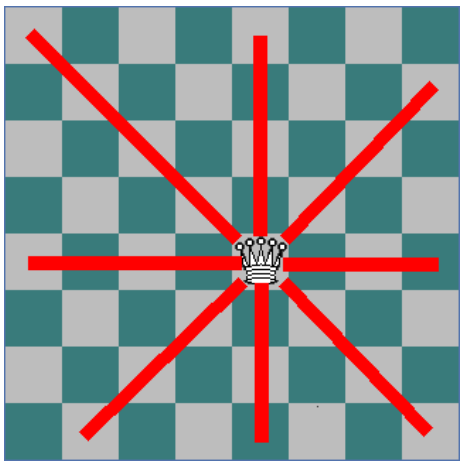
◈ ***Thử chương trình với N=4:***

Input	Output
N=4	1 2 3 4 1 2 4 3 1 3 2 4 1 3 4 2 1 4 2 3 1 4 3 2 2 1 3 4 2 1 4 3 2 3 1 4 2 3 4 1 2 4 1 3 2 4 3 1 3 1 2 4 3 1 4 2 3 2 1 4 3 2 4 1 3 4 1 2 3 4 2 1 4 1 2 3 4 1 3 2 4 2 1 3 4 2 3 1 4 3 1 2 4 3 2 1

8.2. Bài toán Tám quân hậu

◈ Đề bài:

Một bàn cờ quốc tế là một bảng hình vuông gồm 8 hàng, 8 cột. Quân hậu là một quân cờ có thể ăn được bất kì quân nào nằm trên cùng một hàng, cùng một cột, cùng một đường chéo. Bài toán đặt ra là: hãy sắp xếp 8 quân hậu trên cùng một bàn cờ sao cho không có quân hậu nào có thể ăn được quân hậu nào.



◈ Phân tích:

Dĩ nhiên ta không nên tìm lời giải bằng cách xét mọi trường hợp ứng với mọi vị trí của 8 quân hậu trên bàn cờ rồi lọc ra các trường hợp chấp nhân được. Khuynh hướng "thử từng bước" thoạt nghe có vẻ hơi lạ, nhưng lại thể hiện một giải pháp thiết thực: nó cho phép tìm ra tất cả các cách sắp xếp để không có quân hậu nào ăn được nhau.

Nét đặc trưng của phương pháp là ở chỗ các bước đi lời giải hoàn toàn được làm thử. Nếu có một lựa chọn được chấp nhận thì ghi nhớ các thông tin cần thiết và tiến hành bước thử tiếp theo. Nếu trái lại không có một lựa chọn nào thích hợp cả thì làm lại bước trước, xóa bớt các ghi nhớ và quay về chu trình thử với các lựa chọn còn lại. Hành động này được gọi là *quay lui*, và các giải thuật thể hiện phương pháp này gọi là *giải thuật quay lui.*

Đối với bài toán 8 quân hậu: do mỗi cột chỉ có thể có một quân hậu nên lựa chọn đối với quân hậu thứ j, ứng với cột j, là đặt nó vào hàng nào để đảm bảo "an toàn" nghĩa là không cùng hàng, không cùng ường chéo với j-1 quân hậu đã được sắp xếp trước đó. Rõ ràng để đi tới các lời giải ta phải thử tất cả các trường hợp sắp xếp quân hậu đầu tiên tại cột 1. Với một vị trí thử như vậy ta phải giải quyết bài toán 7 con hậu với phần còn lại của bàn cờ, nghĩa là ta đã "quay lại bài toán cũ"!

Sử dụng giải thuật đệ quy quay lui cho bài toán mã đi tuần, ta có:

- *Try(j):* Chọn dòng để đặt quân hậu thứ j (ở cột j).

- *Các phương án chọn*: Như trên đã nêu, đối với quân hậu thứ j, vị trí của nó chỉ chọn trong cột thứ j. Vậy tham biến j trở thành chỉ số cột và việc chọn lựa "các phương án chọn" được tiến hành trên 8 giá trị số hàng i (chọn dòng i: 1, 2,3 ,.., 8).

- *Chọn được:* Để lựa chọn i được chấp nhận, thì hàng i và hai đường chéo qua ô (i,j) phải không có quân hậu nào ở trên đó. Ta thấy trong đường chéo theo chiều ↗ (song song với chéo chính) có các ô (i,j) mà tổng i + j không đổi, còn đường chéo theo chiều ↘ (song song với chéo phụ) có các ô (i,j) mà i-j không đổi.

Do đó ta sẽ chọn các mảng một chiều Boolean để biểu diễn các tình trạnh này:

a

T	T	T	T	T	T	T	T
chỉ số m 1	2	3	4	5	6	7	8

b

T	T	T	T	T	…	T	T
chỉ số i 2	3	4	5	6		15	16

c

T	T	T	T	T	…	T	T
chỉ số k -7	-6	-5	-4	-3	….	6	7

a[i] = true có nghĩa là không có quân hậu nào chiếm hàng i.

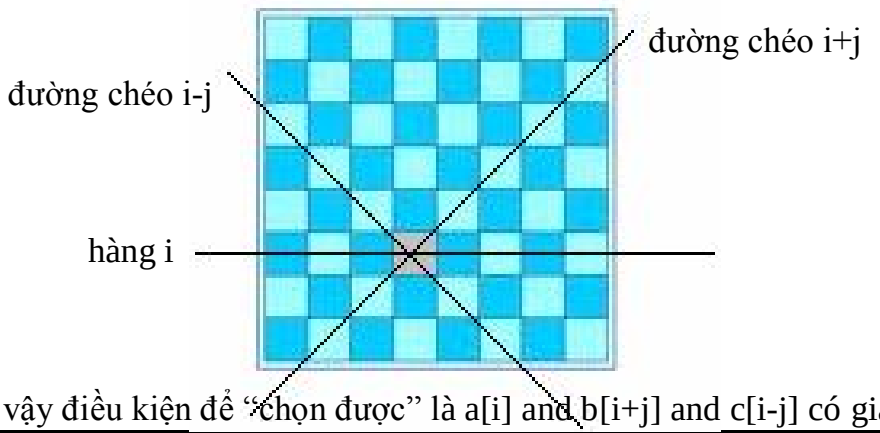
b[i + j] = true có nghĩa là không có quân hậu nào chiếm đường chéo i + j

c[i - j] = true có nghĩa là không có quân hậu nào chiếm đường chéo i - j.

Ta đã biết: 1<=i, j<=8

→ Nên suy ra 2<=i +j <=16

-7<=i - j<=7



Như vậy điều kiện để "chọn được" là a[i] and b[i+j] and c[i-j] có giá trị true.

Vị trí của mỗi con hậu được ghi vào mảng một chiều x gồm 8 phần tử là 8 vị trí dòng của các con hậu ở 8 cột tương ứng.

- *Thực hiện bước đi thứ j:*

+ *Tổ chức lưu trữ:*

x	i	i	…					
chỉ số k	1	2	3	4	5	6	7	8

+ *Lưu vết*:

x[j] := i;

a[i] := false;

b[i+j] := false;

c[i-j] := false;

- *Thành công:* khi j=8 tức là đặt xong vị trí của 8 con hậu, khi đó sẽ "thông báo kết quả" vị trí các con hậu đã đặt, ngược lại sẽ tiếp tục thử để đặt các con hậu tiếp theo.

- *Hủy bước đi thứ j:*

a[i] := true;

b[i+j] := true;

c[i-j] := true;

◈ *Thủ tục đệ quy quay lui cho bài toán này được viết như sau:*

```
Procedure  TRY( j: byte );
 Begin
  For  i :=1  to  8  do
   If  (a[i])  and  (b[i+j])  and  (c[i-j])  then
    Begin
     x[j] := i;
     a[i] := false;
     b[i+j] := false;
     c[i-j]:=false;
   If j=8 then
    Begin
     For k:=1 to 8 do
      write(x[k],' ')
      writeln;
```

```
    End
   Else
    TRY(j+1);
    a[i]:=true;
    b[i+j]:=true;
    c[i-j]:=true;
   End;
 End;
```

◈ ***Cài đặt chương trình:***

```
Program Tam_quan_hau;
Var
  j, k: byte;
  i: integer;
  x: array[1..8] of byte;
  a: array[1..8]  of  boolean;
  b: array[2..16]  of  boolean
  c: array[-7..7]  of  boolean
{-----------------------------------------------------}
Procedure  TRY( j: byte );
 Begin
  For  i :=1  to  8  do
   If  (a[i])  and  (b[i+j])  and  (c[i-j])  then
    Begin
     x[j] := i;
     a[i] := false;
     b[i+j] := false;
     c[i-j]:=false;
   If j=8 then
    Begin
     For k:=1 to 8 do
      write(x[k],' ')
      writeln;
    End
```

```
   Else
    TRY(j+1);
    a[i]:=true;
    b[i+j]:=true;
    c[i-j]:=true;
   End;
 End;
{-----------------------------------------------------}
BEGIN
 For i:=1 to 8 do a[i]:=true;
 For i:=2 to 16 do b[i]:=true;
 For i:=-7 to 7 do c[i]:=true;
 TRY(1);
 Readln;
END.
```

◈ ***Chạy thử chương trình:***

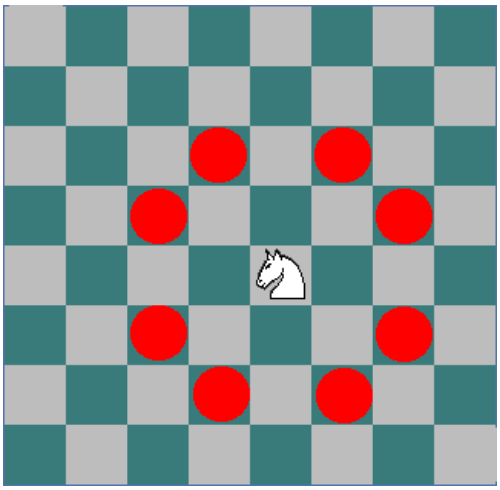
Sau đây là 12 trong số 92 lời giải của bài toán:

x1	x2	x3	x4	x5	x6	x7	x8
1	5	8	6	3	7	2	4
1	6	8	3	7	4	2	5
1	7	4	6	8	2	5	3
1	7	5	8	2	4	6	3
2	4	6	8	3	1	7	5
2	5	7	1	3	8	6	4
2	5	7	4	1	8	6	3
2	6	1	7	4	8	3	5
2	6	8	3	1	4	7	5
2	7	3	6	8	5	1	4
2	7	5	8	1	4	6	3
2	8	6	1	3	5	7	4

8.3. Bài toán Mã đi tuần

◈ ***Đề bài:*** **Mã đi tuần (hành trình của quân mã)** là bài toán về việc di chuyển một quân mã trên bàn cờ vua (8 x 8). Quân mã được đặt ở một ô trên một bàn cờ trống nó phải di chuyển theo quy tắc của cờ vua để đi qua mỗi ô trên bàn cờ đúng một lần. Hãy viết chương trình đặt quân mã vào một vị trí bất kỳ trên bàn cờ. Tìm chu trình của quân mã để ghé thăm một lần duy nhất tất cả các ô còn lại

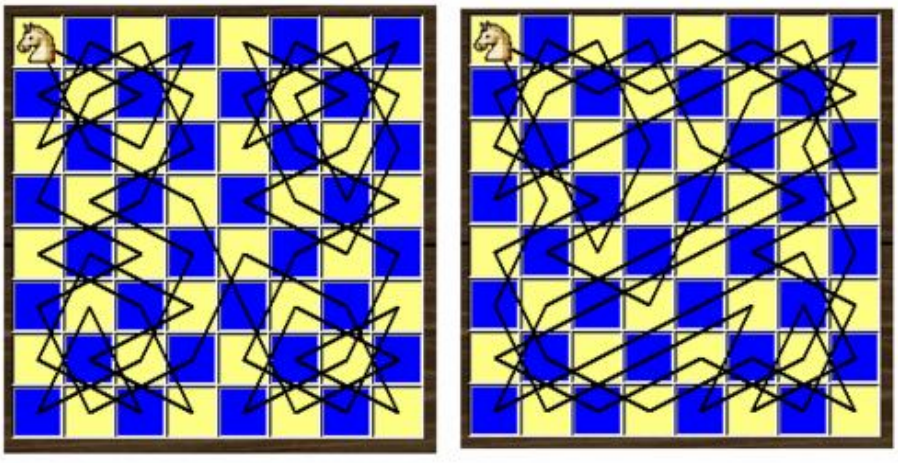
- Quy tắc di chuyển của quân mã trên bàn cờ vua: quân mã đi theo hình chữ L



◈ ***Phân tích bài toán:***

Có rất nhiều lời giải cho bài toán này, chính xác là 26.534.728.821.064 lời giải trong đó quân mã có thể kết thúc tại chính ô mà nó khởi đầu. Một hành trình như vật được gọi là hành trình đóng.

Có những hành trình, trong đó quân mã sau khi đi hết tất cả 64 ô của bàn cờ (kể cả ô xuất phát), thì từ ô cuối của hành trình không thể đi về ô xuất phát chỉ bằng một nước đi. Những hành trình như vậy được gọi là hành trình mở.



Hai trong số nhiều hành trình đóng trên bàn cờ 8 x 8.

Đối với bài toán mã đi tuần ta nhận thấy rằng, ban đầu tọa độ của quân mã là (i,j) bất kỳ trên bàn cờ. Tại vị trí này theo quy luật di chuyển của quân mã trên bàn cờ thì quân mã có tối đa 8 hướng đi tiếp theo. Rõ ràng để đi đến các lời giải ta phải thử tất cả các trường hợp quân mã có thể đi đến ở bước 1. Với mỗi vị trí thử như vậy ta lại thử tất cả các các trường hợp của quân mã có thể đi đến ở bước 2,… Cứ tiếp tục như vây cho đến khi quân mã đi qua tất cả các ô tên bàn cờ vua thì dừng lại. Như vậy, tính chất đệ quy của bài toán đó thể hiện trong các phép thử hướng đi cua quân mã ở trên.

Kết quả ta tìm được nhiều chu trình đi khác nhau nhưng số bước đi đều là 63.

Sử dụng giải thuật đệ quy quay lui cho bài toán mã đi tuần, ta có:

- *Try(k,i,j)*: Chọn thực hiện bước đi thứ k, với ô xuất phát là (i,j);

- *Các phương án chọn*: Có 8 phương án (tương ứng với 8 trường hợp có thể đi đến của quân mã):

+ Kí hiệu phương án: m = 1, 2, …, 8.

+ Thiết lập hai mảng một chiều d,c lưu trữ tọa độ tương ứng 8 khả năng di chuyển của quân mã:

d	-2	-2	-1	1	2	2	1	-1
chỉ số m	1	2	3	4	5	6	7	8

c	-1	1	2	2	1	-1	-2	-2
chỉ số m	1	2	3	4	5	6	7	8

Khi đó: phương án m là di chuyển mã đến ô (i+d[m], j+c[m]).

- *Chọn được*: nếu như ô (i+d[m], j+c[m]) quân mã chưa đi qua và 1<= i+d[m]<= 8 và 1<= j+c[m]<= 8.

- *Thực hiện bước đi thứ k*:

+ <u>*Tổ chức lưu trữ*</u>: Dùng mảng 2 chiều A có kích thước 8x8: A[1..8, 1..8]

A[i,j] = { 0 : nếu quân mã chưa đi qua ô (i,j)
k : nếu quân mã đã đi qua ô (i,j) ở bước thứ k

+ <u>*Lưu vết*</u>: A([i+d[m], j+c[m]):=k.

- *Thành công*: k=64 (hoặc k>63).

- *Thông báo kết quả*: In mảng A.

- *Lời gọi đệ quy tiếp theo*: Try(k+1, i+d[m], j+c[m]).

- *Hủy bước đi thứ k*: A([i+d[m], j+c[m]):=0.

** Thủ tục đệ quy quay lui cho nài toán này được viết như sau:*

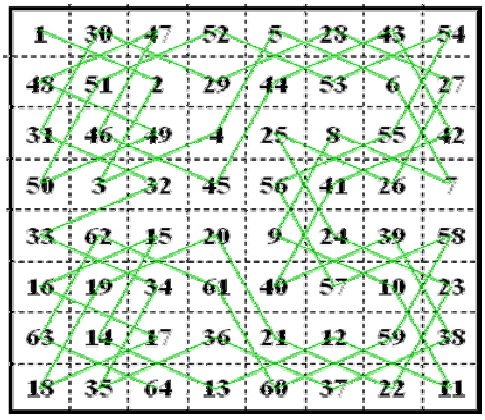
```
Procedure Try(k,i,j);
Begin
    for m:=1 to 8 do
    If (A(i+d[m], j+c[m]) = 0) and ((i+d[m]) in S) and
((j+c[m]) in S) Then
    Begin
        A(i+d[m], j+c[m]) = k;
        If k=64 Then Xuat Else Try(k+1, i+d[m], j+c[m]);
        A(i+d[m], j+c[m]) = 0;
    End;
End;
```

◈ Cài đặt chương trình:

```
Program Ma_di_tuan;
Const
    d:array[1..8] of integer=(-2,-2,-1,1,2,2,1,-1);
    c:array[1..8] of integer=(-1,1,2,2,1,-1,-2,-2);
Var
    A:array[1..8,1..8] of integer;
    S:set of 1..8;
    i, j: integer;
{-----------------------------------------------------}
Procedure Nhap;
Begin
    fillchar(A,sizeof(A),0);
    writeln('Cho biet toa do ban dau cua ma: ');
    readln(i,j);
    A[i,j]:=1;
    S:=[1,2,3,4,5,6,7,8];
End;
{-----------------------------------------------------}
Procedure xuat;
var x, y:integer;
Begin
```

```
    for x:=1 to 8 do
    begin
        for y:=1 to 8 do write(A[x,y]:5);
        writeln;
    end;
End;
{-------------------------------------------------------}
Procedure Try(k,i,j);
var m:byte;
Begin
    for m:=1 to 8 do
    If (A(i+d[m], j+c[m]) = 0) and ((i+d[m]) in S) and
((j+c[m]) in S) Then
    Begin
        A(i+d[m], j+c[m]) = k;
        If k=64 Then Xuat Else Try(k+1, i+d[m], j+c[m]);
        A(i+d[m], j+c[m]) = 0;
    End;
End;
{-------------------------------------------------------}
BEGIN
    Nhap;
    Try(2, i, j);
    Readln;
END.
```

◈ ***Chạy thử chương trình:*** Dưới đây là một đường đi của quân mã:



PHẦN III: PHẦN KẾT LUẬN

Có thể nói rằng đệ quy là quả tim trong các nghiên cứu lý thuyết, cũng như thực hành tính toán đã thể hiện rất nhiều sức mạnh và có ưu điểm trong nhiều bài toán. Căn cứ vào mục đích và nhiệm vụ của đề tài, chúng em đã đạt được những kết quả sau:

- Tìm hiểu và trình bày khái niệm đệ quy, giải thuật đệ quy, chương trình con đệ quy, đệ quy quay lui
- Tìm hiểu cấu trúc chung của chương trình con đệ quy
- Tìm hiểủ về nguyên tắc hoạt động của giải thuật đệ quy
- Xây dựng chương trình cài đặt 8 bài toán về đệ quy nói chung và 3 bài toán điển hình về đệ quy quay lui

Như vậy, đề tài cơ bản đã hoàn thành, đảm bảo các nhiệm vụ và mục đích nghiên cứu đã đề ra.

Mặc dù đã có nhiều thời gian tìm hiểu về đề tài, tuy nhiên do ảnh hưởng của công việc học tập, thi cử và nhiều công việc khác, bên cạnh đó do khả năng trình độ của bản thân chúng em có hạn nên việc thực hiện đề tài còn có một số hạn chế sau:

- Chưa trình bày kỹ lý thuyết về đệ quy quay lui
- Hệ thống các bài toán và lời giải chưa đa dạng và phong phú
- Chưa cài đặt các bài toán trên với lời giải không đệ quy để thấy rõ ưu điểm và hạn chế của đệ quy.

Chúng em hi vọng những hạn chế này sẽ là bài học kinh nghiệm, là những nội dung chúng em cần phấn đấu xây dựng trong thời gian tới.

Chúng em mong các thầy cô đóng góp ý kiến để đề tài được hoàn thiện hơn.

Xin chân thành cảm ơn!

t µ i l iÖu t h a m k h ¶ o

----

[1] Đỗ Xuân Lôi, *Cấu trúc dữ liệu và giải thuật*, Nhà xuất bản khoa học và kỹ thuật.

[2] Trần Đức Huyên, *Một số bài toán chọn lọc trong Tin học*, Nhà xuất bản giáo dục.

[3] Trần Đức Huyên, *Các vấn đề về lập trình Pascal*, Nhà xuất bản Thanh Niên.

[4] Nguyễn Xuân Huy, *Sáng tạo trong thuật toán và lập trình - Tập 1*, Nhà xuất bản giáo dục.

[5] Nguyễn Xuân My, *Một số vấn đề chọn lọc trong Tin học - Tập 1*, Nhà xuất bản giáo dục.

[6] Nguyễn Xuân My, *Một số vấn đề chọn lọc trong Tin học - Tập 2*, Nhà xuất bản giáo dục.

[7] Nguyễn Hải Lộc - Nguyễn Thanh Tiên, *Bài giảng ngôn ngữ lập trình Pascal,* Đại học sư phạm Huế.

[8] Các tài liệu về Đệ quy trên Internet.

mô c lô c