CHƯƠNG 9 ĐỆ QUI

Giới thiệu phương pháp lập trình theo kỹ thuật đệ quy, phân loại, cách hoạt động và cách cài đặt các hàm đệ quy.

I. TÓM TẮT LÝ THUYẾT

I.1. Khái niệm

Một hàm được gọi có tính đệ qui nếu trong thân của hàm đó có lệnh gọi lại chính nó một cách tường minh hay tiềm ẩn.

I.2. Phân loại đệ qui

- Đệ qui tuyến tính.
- Đệ qui nhị phân.
- Đệ qui phi tuyến.
- Đệ qui hỗ tương.

a. Đệ qui 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>)
{
        if (điều kiện dừng)
        {
                . . .
                //Trả về giá trị hay kết thúc công việc
        }
        //Thực hiện một số công việc (nếu có)
        . . . TenHam (<danh sách tham số>);
        //Thực hiện một số công việc (nếu có)
}
```

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

Trước khi cài đặt hàm đệ qui ta xác định:

- Điều kiện dừng: S(0) = 0.

- Qui tắc (công thức) tính: S(n) = S(n-1) + n.

Ta cài đặt hàm đệ qui như sau:

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

Ví dụ 2: Tính P(n) = n!

Trước khi cài đặt hàm đệ qui ta xác định:

- Điều kiện dừng: P(0) = 0! = 1.

*- Qui tắc (công thức) tính: P(n) = P(n-1) * n.*

Ta cài đặt hàm đệ qui như sau:

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

b. Đệ qui 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>)
{
        if (điều kiện dừng)
        {
                . . .
                //Trả về giá trị hay kết thúc công việc
        }
        //Thực hiện một số công việc (nếu có)
        . . .TenHam (<danh sách tham số>); //Giải quyết vấn đề nhỏ hơn
        //Thực hiện một số công việc (nếu có)
        . . . TenHam (<danh sách tham số>); //Giải quyết vấn đề còn lại
        //Thực hiện một số công việc (nếu có)
}
```

Ví dụ 1: Tính số hạng thứ n của dãy Fibonaci được định nghĩa như sau:

$f_1 = f_0 = 1$;

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

Trước khi cài đặt hàm đệ qui ta xác định:

- Điều kiện dừng: f(0) = f(1) = 1.

Ta cài đặt hàm đệ qui như sau:

```
long Fibonaci (int n)
{
        if(n==0 || n==1)
                return 1;
        return  Fibonaci(n-1) + Fibonaci(n-2);
}
```

Ví dụ 2: Cho dãy số nguyên a gồm n phần tử có thứ tự tăng dần. Tìm phần tử có giá trị x có xuất hiện trong mảng không?

Trước khi cài đặt hàm đệ qui ta xác định:

- Điều kiện dừng: Tìm thấy x hoặc xét hết các phần tử.

- Giải thuật:

Do dãy số đã có thứ tự tăng nên ta có thể áp dụng cách tìm kiếm theo phương pháp nhị phân. Ý tưởng của phương pháp này là tại mỗi bước ta tiến hành so sánh x với phần tử nằm ở vị trí giữa của dãy để thu hẹp phạm vi tìm.

Gọi: l: biên trái của dãy (ban đầu l=0).

r: biên phải của dãy (ban đầu r = n-1).

m: vị trí ở giữa (m = (l+r)/2).

l ↓			*m* ↓			*R* ↓
a[0]	*a[1]*	*...*	*a[(l+r)/2]*	*...*	*a[n-2]*	*a[n-1]*

Thu hẹp dựa vào giá trị của phần tử ở giữa, có hai trường hợp:

i. *Nếu x lớn hơn phần tử ở giữa thì x chỉ có thể xuất hiện ở bên phải vị trí này. (từ m+1 đến r).*
ii. *Ngược lại nếu x nhỏ hơn phần tử ở giữa thì x chỉ có thể xuất hiện ở bên trái vị trí này. (từ l đến m-1).*

Quá trình này thực hiện cho đến khi gặp phần tử có giá trị x, hoặc đã xét hết các phần tử.

Ta cài đặt hàm đệ qui như sau:

```
int TimNhiPhan(int a[], int l, int r, int x)
{
   int m = (l+r)/2;
   if(l>r)
      return -1;// Không có phần tử x
   if(a[m]>x)  return TimNhiPhan(a, l, m-1, x);
   if(a[m]<x)  return TimNhiPhan(a, m+1, r, x);
   return m;//Trả về vị trí tìm thấy
```

```
}
```

Ví dụ 3: Bài toán tháp Hà Nội:

Bước 1: Di chuyển n -1 đĩa nhỏ hơn từ cọc A sang cọc B.

Bước 2: Di chuyển đĩa còn lại từ cọc A sang cọc C.

Bước 3: Di chuyển n -1 đĩa nhỏ hơn từ cọc B sang cọc C.

Ta cài đặt hàm đệ qui như sau:

```
void ThapHaNoi (int n, char A, char B, char C)
{
        if (n = = 1)
                printf("Di chuyen dia tren cung tu %d den %d\n", A, C);
         else
         {
                ThapHaNoi(n-1, A, C, B);
                ThapHaNoi(1, A, B, C);
                ThapHaNoi(n-1, B, A, C);
         }
}
```

c. *Đệ qui 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>);
                }
        }
}
```

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

$X_0 = 1$;

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

Trước khi cài đặt hàm đệ qui ta xác định:

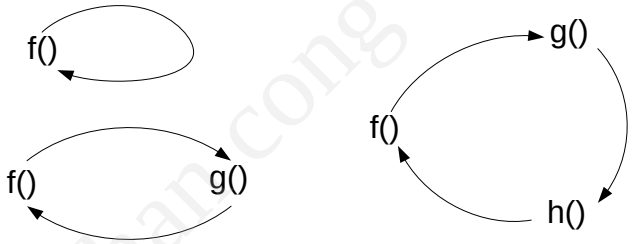
- Điều kiện dừng:X(0) = 1.

Ta cài đặt hàm đệ qui như sau:

```
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;
}
```

d. Đệ qui hỗ 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.




```
<Kiểu dữ liệu hàm> TenHam2 (<danh sách tham số>);
<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)
}
```

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

$X_0 = Y_0 = 1$;

$X_n = X_{n-1} + Y_{n-1}$; (n>0)

$Y_n = n^2 X_{n-1} + Y_{n-1}$; (n>0)

Trước khi cài đặt hàm đệ qui ta xác định:

- Điều kiện dừng:X(0) = Y(0) = 1.

Ta cài đặt hàm đệ qui như sau:

```
long TinhYn(int n);
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);
}
```

I.3. Tìm hiểu cách hoạt động của hàm đệ qui

Phục vụ cho công việc kiểm chứng kết quả thực thi của chương trình bằng tay.

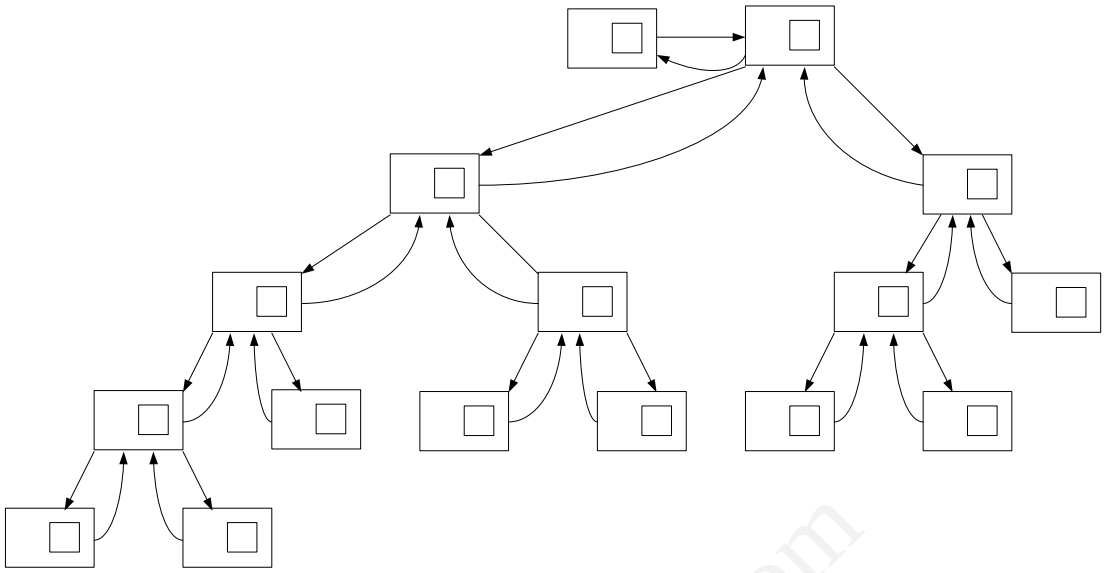
Ví dụ 1: Lấy lại ví dụ tính **P(n) = n!** bằng phương pháp đệ qui như đã mô tả cài đặt ở trên với n = 5

Lệnh gọi khởi đầu trong hàm main(), truyền đến hàm GiaiThua(). Ở đó, giá trị của tham số n là 5, do đó nói gọi GiaiThua(4), truyền 4 đến hàm GiaiThua(). Ở đó giá trị của tham số n là 4, do đó nó gọi GiaiThua(3), truyền 3 đến hàm GiaiThua(). Tiến trình này tiếp tục (đệ quy) đến khi gọi GiaiThua(1) được thực hiện từ bên trong lệnh gọi GiaiThua(2). Ở đó, giá trị của tham số n là 1, do đó nó trả về giá trị 1, mà không thục hiện thêm bất kì lệnh gọi nào. Sau đó lần ngược về lệnh gọi GiaiThua(2) trả 2*1=2 trở về lệnh gọi GiaiThua(3). Sau đó lệnh gọi GiaiThua(3) trả 3*2=6 trở về lệnh gọi GiaiThua(4). Sau đó lệnh gọi GiaiThua(4) trả 4*6=24 trở về lệnh gọi GiaiThua(5). Sau cùng, lệnh gọi GiaiThua(5) trả về giá trị 120 cho hàm main().

Ví dụ 2: Lấy lại ví dụ tính số hạng thứ n của dãy Fibonaci như đã mô tả cài đặt ở trên với n = 5, quá trình thực hiện tương tự như trong ví dụ trước, ta có sơ đồ sau:

main() GiaiThua(5) GiaiThua(4)

5 4



I.4. Ví dụ

Viết chương trình nhập vào mảng một chiều số nguyên a, xuất ra màn hình và tính tổng các phần tử có giá trị chẵn bằng phương pháp đệ qui.

```
#define MAX 100
void Nhap(int a[], int n)
{
   if(n==0)
      return;
   Nhap(a, n-1);
   printf("\nNhap phan tu thu %d: ", n);
   scanf("%d", &a[n-1]);
}

void Xuat(int a[], int n)
{
   if(n==0)
      return;
   Xuat(a, n-1);
   printf("%d\t", a[n-1]);
}

long TongChan(int a[], int n)
{
   if(n==0)
      return 0;
   int s = TongChan(a, n-1);
   if(a[n-1]%2==0)
      s+=a[n-1];
   return s;
}
```

```
void main()
{
   int a[MAX], n;
   long s;

   printf("\nNhap so phan tu cua mang: ");
   scanf("%d", &n);
   Nhap(a, n);
   Xuat(a, n);
   s=TongChan(a, n);
   printf("\nTong cac so chan trong mang la: %ld", s);
   getch();
}
```

II. BÀI TẬP

Viết hàm đệ qui thực hiện các yêu cầu sau:

II.1. Bài tập cơ bản

1. Cài đặt lại những bài tập ở chương mảng một chiều.
2. Tìm chữ số có giá trị lớn nhất của số nguyên dương n.
3. Hãy xây dựng một dãy gồm N số có giá trị từ 1 đến K cho trước, sau cho không có hai dãy con liên tiếp đứng kề nhau.

   Ví dụ: *N = 6*

   K = 3

   Kết quả: 121312
4. Tìm ước số chung lớn nhất của hai số nguyên dương a và b.
5. Tìm chữ số đầu tiên của số nguyên dương n.
6. Tìm dãy nhị phân dài nhất sao cho trên dãy này không có hai bộ k bất kỳ trùng nhau. Bộ k là dãy con có k số liên tiếp nhau trên dãy tìm được.

   Ví dụ: k = 3

   Kết quả: 000 101 110 0
7. Tính $P(n) = 1.3.5\ldots(2n+1)$, với $n \geq 0$
8. Tính $S(n) = 1+3+5+\cdots+(2\times n+1)$, với $n \geq 0$
9. Tính $S(n) = 1-2+3-4+\cdots+(-1)^{n+1}n$, với $n > 0$
10. Tính $S(n) = 1+1.2+1.2.3+\cdots+1.2.3\ldots n$, với $n > 0$
11. Tính $S(n) = 1^2+2^2+3^2+\cdots+n^2$, với $n > 0$

12. Tính $S(n) = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$, với $n > 0$

13. Tính $S(n) = 1 + \frac{1}{1+2} + \frac{1}{1+2+3} + \cdots + \frac{1}{1+2+3+\cdots+n}$, với $n > 0$

14. Tính $P(x, y) = x^y$.

15. Tính $S(n) = 1 + (1+2) + (1+2+3) + \cdots + (1+2+3+\cdots+n)$, với $n > 0$

II.2. Bài tập luyện tập và nâng cao

16. Cho số nguyên dương n. In ra biểu diễn nhị phân của n.

17. **(*)** Cài đặt và minh hoạ bài toán tháp Hà Nội.

18. **(**)** Cài đặt bài toán mã đi tuần.

19. **(**)** Cài đặt bài toán tám hậu.

20. **(*)** Tính $S(n) = \sqrt{n + \sqrt{(n-1) + \sqrt{(n-2) + \ldots + \sqrt{1}}}}$, với $n > 0$

21. **(*)** Tính $S(n) = \sqrt{1 + \sqrt{2 + \sqrt{3 + \ldots + \sqrt{n}}}}$, với $n > 0$

22. **(*)** Tính $S(n) = \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{\ddots \cfrac{1}{1 + \cfrac{1}{1+1}}}}}}$ có n dấu phân số.

III. KẾT LUẬN

- Đệ qui cung cấp cho ta cơ chế giải quyết các bài toán phức tạp một cách đơn giản hơn.
- Xây dựng hàm đệ qui thông qua việc **xác định điều kiện dừng** và **bước thực hiện tiếp theo**.
- Chỉ nên cài đặt bằng phương pháp đệ qui khi không còn cách giải quyết bằng cách lặp thông thường.