KỸ THUẬT LẬP TRÌNH C

Chương 2:
Đệ Qui

1. Vấn đề đệ qui
2. Hàm đệ qui
3. Hoạt động của chương trình đệ qui
4. Biểu thức
5. Ưu khuyết điểm của đệ qui

1. Bài toán đệ qui

- Một bài toán được gọi là đệ qui nếu bài toán đó được định nghĩa lại bằng chính nó.

  Ví dụ 1: Xét bài toán tính S(n) = 1 + … + n với n là số nguyên dương. Đây là bài toán đệ qui có thể được định nghĩa như sau:

  S(1) = 1, S(n) = S(n – 1) + n nếu n > 1

  Ta thấy S(n) được định nghĩa thông qua S(n – 1).

Ví dụ 2: Xét bài toán tính $P(n) = x^n$ với x là số thực và n là số nguyên không âm. Đây là bài toán đệ qui có thể được định nghĩa như sau:

$P(0) = 1$

$P(n) = P(n – 1) * x$ nếu $n > 0$

Ta thấy P(n) được định nghĩa thông qua P(n – 1).

Ví dụ 3: Xét bài toán tính Fac(n) = n! với n là số nguyên không âm. Đây là bài toán đệ qui có thể được định nghĩa như sau:

Fac(0) = 1

Fac(n) = Fac(n – 1) * n nếu n > 0

Ta thấy Fac(n) được định nghĩa thông qua Fac(n – 1).

Ví dụ 4: Xét bài toán tính UCLN(a, b) với a và b là hai số nguyên dương. Đây là bài toán đệ qui có thể được định nghĩa như sau:

UCLN(a, b) = a, nếu a = b

UCLN(a, b) = UCLN(a – b, b), nếu a > b

UCLN(a, b) = UCLN(a, b – a), nếu a < b

Ta thấy UCLN(a, b) được định nghĩa thông qua UCLN(a - b, b) và USCLN(a, b - a).

Ví dụ 5: Xét bài toán tính số hạng thứ n của dãy số Fibonaci {F(n)}. Dãy số {F(n)} được định nghĩa đệ qui như sau:

$F(0) = F(1) = 1$

$F(n) = F(n – 1) + F(n – 2)$ nếu $n \geq 2$

Ta thấy F(n) được định nghĩa thông qua F(n – 1) và F(n – 2).

- Bài toán đệ qui thường được giải qua hai bước: phân tích và thế ngược.

Ví dụ 6: Để tính tổng S(n) = 1 + 2 + . . .+ n, chúng ta có thể thực hiện thông qua hai bước như sau:

Bước phân tích:

Để tính S(n) trước tiên ta phải tính S(n-1) sau đó tính S(n) = S(n-1) +n.

Để tính được S(n-1) trước tiên ta phải tính S(n-2) sau đó tính S(n-1) = S(n-2) + n-1.

. . . . . .

Để tính toán được S(2) trước tiên ta phải tính S(1) sau đó tính S(2) = S(1) + 2.

Và cuối cùng S(1) có ngay kết quả là 1.

Bước thế ngược:

Có kết quả S(1) thay thế ngược lại chúng ta xác định S(n):

S(1) = 1

S(2) = S(1) + 2

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

. . . . . .

$$S(n) = S(n - 1) + n$$

Ví dụ 7: Để tính n!, chúng ta có thể thực hiện thông qua hai bước như sau:

Bước phân tích:

Để tính n! trước tiên ta phải tính (n - 1)! sau đó tính n! = n* (n - 1)!

Để tính (n - 1)! trước tiên ta phải tính (n-2)! sau đó tính (n-1)! = (n – 1) * (n-2) !.

. . . . . .

Để tính 1!, trước tiên ta phải tính 0! sau đó tính 1! = 0! * 1

Và cuối cùng 0! cho kết quả là 1.

Bước thế ngược:

Xuất phát từ 0! thay thế ngược lại chúng ta xác định n!:

0! = 1

1! = 1 * 0!

2! = 2 * 1!

. . . . . .

n! = n* (n - 1)!

- Bài toán giải theo phương pháp đệ qui cần hai điều kiện sau đây:
  - Phải tồn tại **bước đệ qui**.
  - Phải có **điều kiện dừng**.

Bài toán tính S(n) có

Bước đệ qui: $S(n) = S(n - 1) + n$

Điều kiện dừng: $S(1) = 1$

Bài toán tính $P(n) = x^n$ có

Bước đệ qui: $P(n) = P(n - 1) * x$

Điều kiện dừng: $P(0) = 1$

2. Hàm đệ qui

Một hàm được gọi là có tính đệ qui nếu trong thân hàm đó có lệnh gọi lại chính nó một cách tường minh (trực tiếp) hay tiềm ẩn (gián tiếp)

a. Đệ qui tuyến tính

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

Ví dụ 1: Hàm đệ qui tính S(n) = 1 + 2 + … + n

```
int Tong(int n) {
    if(n == 1)      //Điều kiện dừng
        return 1;
    else    //Bước đệ qui
        return (Tong(n - 1) + n);
}
```

Ví dụ 2: Hàm đệ qui tính x luỹ thừa n

```
float LuyThua(float x, int n) {
        if(n == 0)
                return 1;
        else
                return (x * LuyThua(x, n-1));
}
```

Ví dụ 3: Hàm đệ qui tính n giai thừa

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

b. Đệ qui nhị phân

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

Ví dụ 1: Hàm đệ qui tính USCLN của hai số nguyên dương a và b

```
int USCLN(int a, int b) {
        if(a == b)
                return a;
        else if(a > b)
                return USCLN(a - b, b);
        else  return USCLN(a, b - a);
}
```

Ví dụ 2: Hàm đệ qui tính số hạng thứ n của dãy số Fibonaci

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

c. Đệ qui phi tuyến

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

Ví dụ: Dãy {Xn} được định nghĩa như sau:

X0 =1 ;

$Xn = n^2X0+(n-1)^2X1+ \ldots +1^2Xn-1 , (n\geq 1)$

Tính số hạng thứ n của dãy

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

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.

Ví dụ: Tính số hạng thứ n của hai dãy {Xn}, {Yn} được định nghĩa như sau:

$X0 = Y0 = 1$;

$Xn = Xn-1 + Yn-1; (n>0)$

$Yn = n^2Xn-1 + Yn-1; (n>0)$

Tính số hạng thứ n của hai dãy trên

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

3. Cách hoạt động CT đệ qui

- Mỗi chương trình đệ qui sẽ được biểu diễn bằng một cây đệ qui: Bước phân tích thể hiện qua cây đệ qui đi từ trên xuống dưới và bước thế ngược thể hiện qua cây đệ qui đi từ trái qua phải, từ dưới lên trên.

- Tùy theo loại đê qui mà cây đệ qui có cấu trúc đặc trưng

  Đệ qui nhị phân -> cây đệ qui được biểu diễn bằng một cây nhị phân

  Đệ qui tuyến tính -> cây đệ qui được biểu diễn bằng một cây suy thoái (một danh sách đơn).

4. Ưu khuyết điểm của đệ qui

- **Ưu điểm**

  Dùng phương pháp đệ qui giúp đơn giản quá trình thiết kế và mã hoá chương trình, chương trình thường ngắn gọn, dễ hiểu và gần gũi với người lập trình.

- **Khuyết điểm**

  Phương pháp đệ qui thường không tối ưu về mặt thời gian. Sự chiếm dụng bộ nhớ của phương pháp đệ quy cũng khá lớn.

Hết