递归的基本概念:程序调用自身的编程技巧称为递归,是函数自己调用自己.

一个函数在其定义中直接或间接调用自身的一种方法,它通常把一个大型的复杂的问题转化为一个与原问题相似的规模较小的问题来解决,可以极大的减少代码量.递归的能力在于用有限的语句来定义对象的无限集合.

使用递归要注意的有两点:

1)递归就是在过程或函数里面调用自身;

2)在使用递归时,必须有一个明确的递归结束条件,称为递归出口.

递归分为两个阶段:

1)递推:把复杂的问题的求解推到比原问题简单一些的问题的求解;

2)回归:当获得最简单的情况后,逐步返回,依次得到复杂的解.

利用递归可以解决很多问题:如背包问题,汉诺塔问题,...等.

斐波那契数列为:0,1,1,2,3,5...

fib(0)=0;

fib(1)=1;

fib(n)=fib(n-1)+fib(n-2);

**[cpp]** [view plain](http://blog.csdn.net/swliao/article/details/5337896) [copy](http://blog.csdn.net/swliao/article/details/5337896)

1. **int** fib(**int** n)
2. {
3. **if**(0 == n)
4. **return** 0;
5. **if**(1 == n)
6. **return** 1;
7. **if**(n > 1)
8. **return** fib(n-1)+fib(n-2);
9. }

上面就是一个简单的递归调用了.由于递归引起一系列的函数调用,并且有可能会有一系列的重复计算,递归算法的执行效率相对较低.

迭代:利用变量的原值推算出变量的一个新值.如果递归是自己调用自己的话,迭代就是A不停的调用B.

递归中一定有迭代,但是迭代中不一定有递归,大部分可以相互转换.能用迭代的不用递归,递归调用函数,浪费空间,并且递归太深容易造成堆栈的溢出.

**[cpp]** [view plain](http://blog.csdn.net/swliao/article/details/5337896) [copy](http://blog.csdn.net/swliao/article/details/5337896)

1. //这是递归
2. **int** funcA(**int** n)
3. {
4. **if**(n > 1)
5. **return** n+funcA(n-1);
6. **else**
7. **return** 1;
8. }
9. //这是迭代
10. **int** funcB(**int** n)
11. {
12. **int** i,s=0;
13. **for**(i=1;i<n;i++)
14. s+=i;
15. **return** s;
16. }