递归和迭代，是算法中经常出现的两个词，他们的具体含义是什么呢，递归和迭代是计算机处理问题的两种基本方法。在算法的分析与设计中，递归和迭代都是特别有力的工具，是解决循环问题常用的两种方法。

通过简单的代码对比这两种算法

1. //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. //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. *}*

从代码中我们分析递归和迭代的含义和不同

递归

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,它通常把一个大型复杂的问题层层转化为一个与原问题类似的规模较小的问题来求解，递归策略只需少量的程序就可描述出解题过程所需要的多次重复计算，大大地减少了程序的代码量。递归的能力在于用有限的语句来定义对象的无限集合。用递归思想写出的程序往往十分简洁易懂。

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.

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

迭代的基本概念：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，是一种循环形式。

递归和迭代的转化

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.

例：1+2+3+4+..+100的值。

递归的做法：我要求1到100的累加值，如果我已经得到1到99的累加值，将这个值加上100就是1到100的累加值；要得到1到99的累加值，如果已经得到1到98的累加值，将这个值加上99，就是1到99的累加值……最后我要得到1到2的累加值，我如果得到1自身累加值，再加上2即可，1自身的累加值显然就是1了。于是现在我们得到了1到2的累加值，将这个值加3就得到了1到3的累加值，……最后直到得到1到100的累加值。

程序表示，其中函数会调用自身，这就是递归方法的典型特征

int GetSum(int n)

{

if(n<=0) return 0;

else return n+GetSum(n-1);

}

迭代的做法：从1到100，顺着往下累加。1+2=3,3+3=6,6+4=10,10+5=15……

程序表示，

int i=1,sum=0;

while(i<=100){

sum = sum +i;

}

上述例子中，其实递归最后得到结果也是用迭代方法完成的，只是在程序的处理上直观看不出来。两者都能很好的完成计算任务，不同之处在于思维方式上，从而导致不同的计算方法：迭代是正向思维，从头到尾思考问题；递归是逆向思维，他假设我们已经得到了部分结果(假设我已经知道了1到99的累加值，把这个值加上100我们就得到了1到100的累加值了)，从尾部追溯到头部，从而让问题简化