# 有关函数的编程思想

**递归思想——递归函数**

**递归函数，就是：在一个函数内部调用它直接的函数！**

**先考察一个最简单的函数**

**Function f1（$n）{**

**Echo $n**

**$n++**

**F1($n)**

**}**

**F1(1);**

从这个简单的函数可以看出，该函数调用的是“永无止境”的，最终会将内存消耗完毕

显然，这不是一个正常的做法！

实用的递归函数是:能够控制这个调用过程中，会在某个时刻（条件）停下来。

实例 求 5 的阶乘

数字上，有这样两个有关阶乘的基本规则：

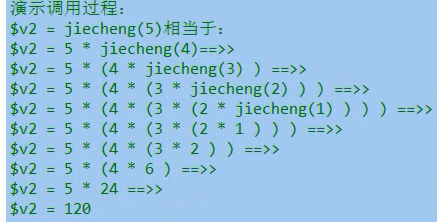
1. n的阶乘，是n-1的阶乘，乘以n的结果
2. 1的阶乘是1；

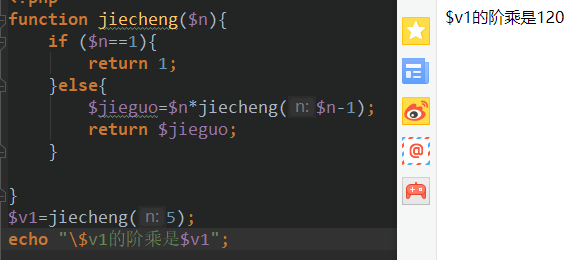
现在假设，有一个函数，该函数能够计算n的阶乘

Function jiecheng（$n）{

}

$v1=jiecheng(8);//结果应是8的阶乘





递归思想总结：

当面对一个“大问题”，该大问题可以经由该问题的同类问题的“小一级问题”，经过简单计算获得。而且可以获知（已知）这类问题的“最小一级问题”的答案。则，此时可以使用1递归方式来解决该问题。

则此时该函数的基本模式是：

Function digui（$n）{

If(最小一级){

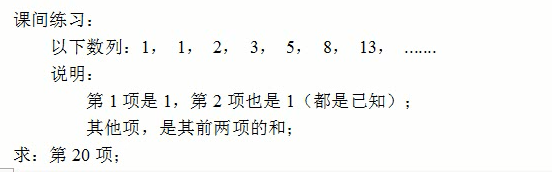
Return 已知答案

}

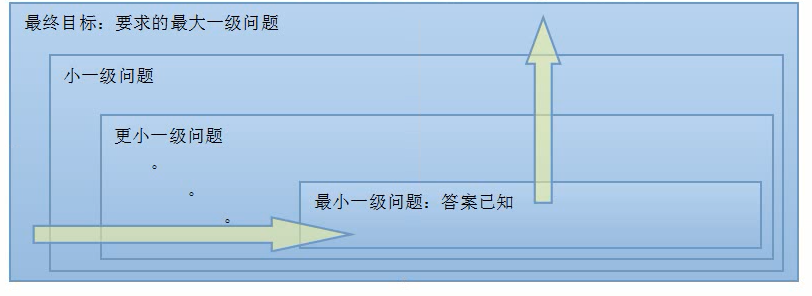
$jieguo=对digui（$n-1）进行简单运算

Return $jieiguo;

}



递归思想图示：

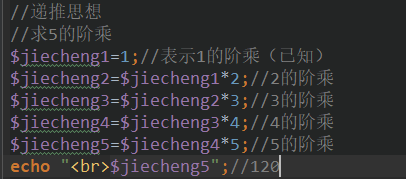


**递推（迭die代）思想**

也同样思考这个问题：

求5的阶乘：

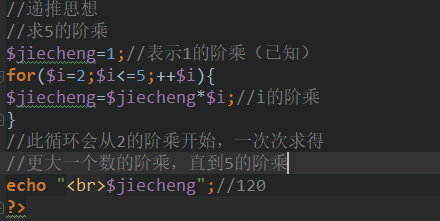
最初做法



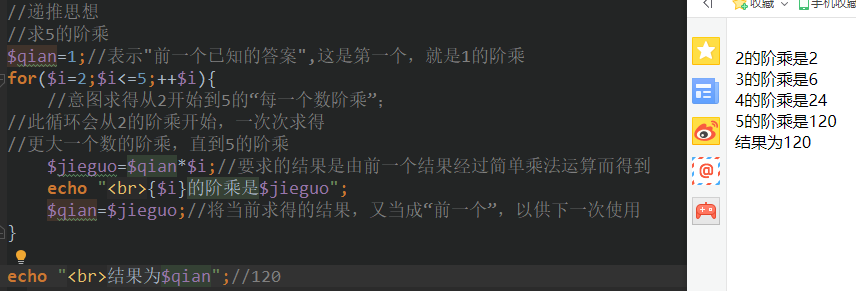
将上述代码，使用一个变量，也同样能完成



将上述代码的规律性体现出来—for



**将该语句再次进转换，使用递推思想中的2个观念：前一个答案，后一个答案；**



**递推总结：**

如果要求一个“大问题”，且该问题有如下2个特点

1. 已知该问题的同类问题的最小答案的答案
2. 如果知道这种问题的小一级的答案，就可以轻松求得其“大一级”问题的答案，并且此问题的级次有一定规律

则此时就可以使用递推思想来解决该问题，代码模式为：

$qian=已知的最小一级问题的答案

For（$i=最小一级的下一级，$i<=最大一级的级次；$i++）{

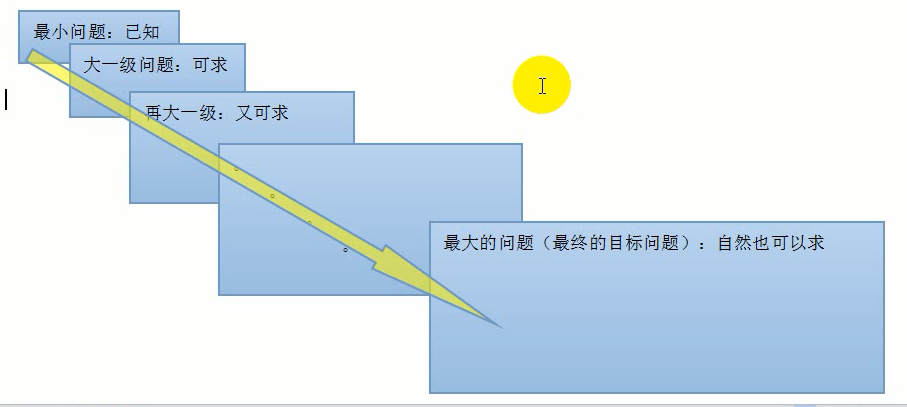
$jieguo=对qian进行一定的计算，通常需要使用到$i

$qian=$jieguo;

}

Echo “结果为:”.$jieguo

**递推思想图示：**



通常，如果一个问题，既能使用递归计算解决，又能使用递推解决，则应该使用递推算法；