**前言:**

《编程之美》这本书，爱不释手！

**问题描述:**

1. 给定一个整数N,那么N的阶乘N！末尾有多少个0呢?例如:N=10，N!=362800，N!的末尾有两个0;
2. 求N!的二进制表示中最低位1的位置。

**问题1的求解：**

**分析：**

**解法一：**

       首先，最直接的算法当然是直接求出来N！然后看末尾有几个0就行了。但这里存在两个问题：

     （1）不管使用long或者double一定会产生溢出。

     （2）效率低下。

      对于问题（1），我们可以采用字符串存储的办法解决，但问题（2）是由本身算法决定的，所以只能采用其他的算法。

那 到底有没有更好的算法呢？我们来分析，N！能产生0的质数组合只能是2 \* 5，也就是说当对N!进行质数分解之后，N!末尾0的个位M取决于2的个数X和5的个数Y的最小值，即M = min（X,Y）。又因为能被2整除的数出现的频率比能被5整除的数高得多，且出现一个5的时,最少会同时出现一个2，所以M = Y。即得出Y的值就可以得到N!末尾0的个数。

计算Y，最直接的方法，就是计算机1…N的因式分解中5的个数，然后求和。

 代码如下：

[复制代码](javascript:void(0);)

static long GetZeroNum(long n)

{

long num = 0;

int i,j;

for(i=1; i<=n; i++)

{

j=i;

while(j % 5 == 0)

{

num++;

j/=5;

}

}

return num;

}

[复制代码](javascript:void(0);)

**解法二：**

      那 么还有没有更简单点的方法呢？我们想，Y还能怎么样得到？举个例子 25的阶乘中，总共有6个五，其中5,10,15,20，各贡献一个，25贡献两个，也可以说成，5,10,15,20，25各贡献一个，25又额外贡献 一个，即5的倍数各贡献一个5,25的倍数各贡献一个5,即Y=[25/5] + [25/25]。同理，125中，5的倍数各贡献一个5,25的倍数各贡献一个5,125的倍数也各贡献一个5，所以Y=[125/5] + [125/25] + [125/125],所以可得公式：

Y = [N/5] + [N/52] + [N/53] + …

代码如下：

[复制代码](javascript:void(0);)

static long GetZeroNum(long n)

{

long num = 0;

while(n != 0)

{

num=num+n/5;

n=n/5;

}

return num;

}

[复制代码](javascript:void(0);)

**问题2的求解：**

**分析：**

     首先我们来分析一个二进制数乘以2和除以2的过程和结果是怎么样。

     一个二进制数乘以2就是把将此二进制数向左移一位，末位补零。除以2时，则要判断末位是否为0，若为0，向右移一位，若不能为0，则不能被2整除。

     所以，其实本问题其实是求N!含有多少个2，最低位1的位置等于N！中含有2的个数加1。

代码如下：

[复制代码](javascript:void(0);)

//计算n的阶乘的二进制中最低位1的位置，

//返回值表示倒数第几位；

static long LowestOnew(long n)

{

long num=0;

while(n!=0)

{

num=num+n/2;

n=n/2;

}

return num+1;

}

例如N=9

9,8,7,6,5,4,3,2,1中

8,6,4,2,共4个数可以贡献2（也就是9/2=4）

之后变成了4,3,2,1

4,2 共2个数可以贡献2（也就是4/2=2）

之后变成了2,1

共2 一个数可以贡献2（也就是2/2=1）

之后变成1

之后就没有可以贡献2的了

所以4+2+1