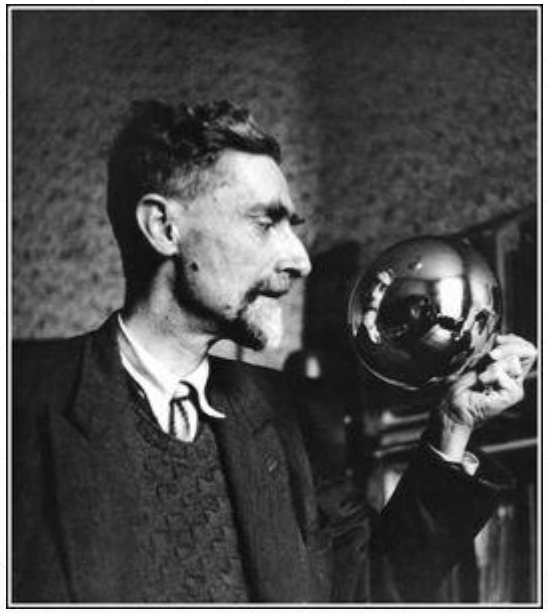
“递归”是只拦路虎，你是武松吗?

重复创造了美

递归与递推

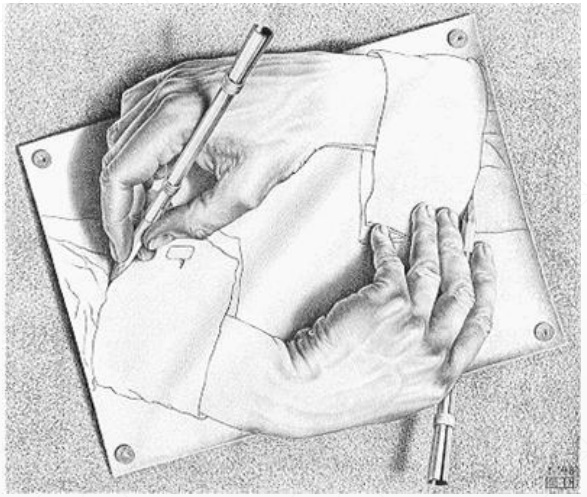
北京理工大学计算机学院
金旭亮

不可思议的递归……

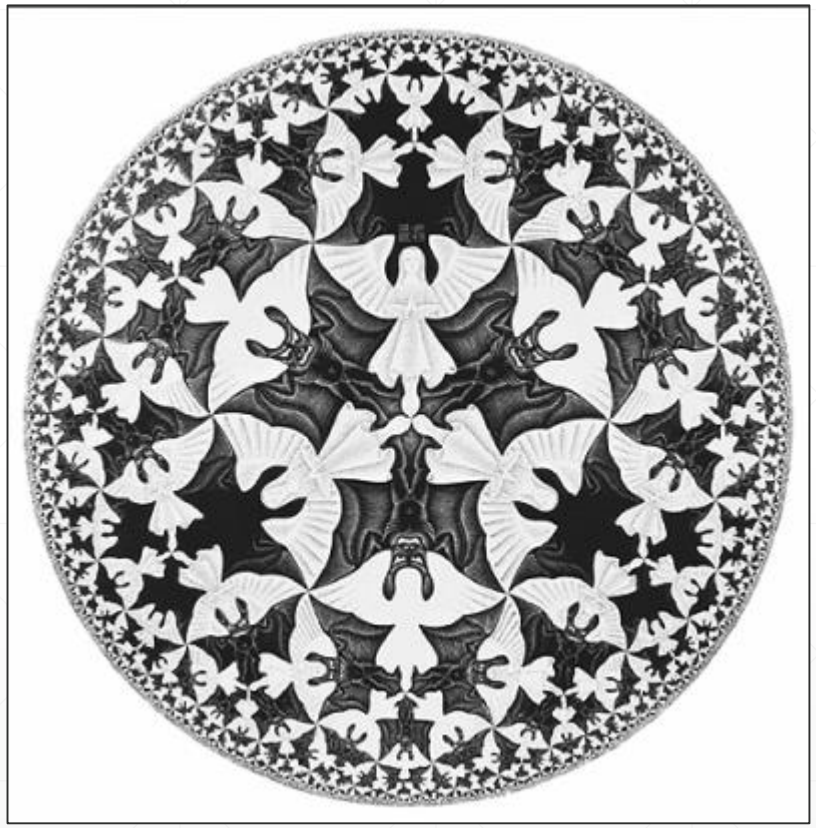


M.C.埃舍尔（M. C. Escher，1898~1972）

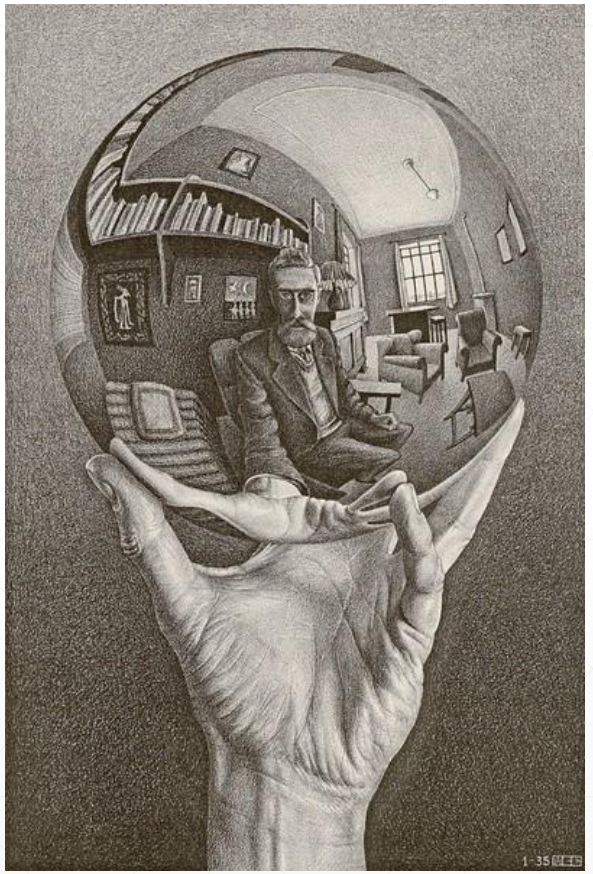
以“鱼”为笔绘制“鱼”



绘画的手

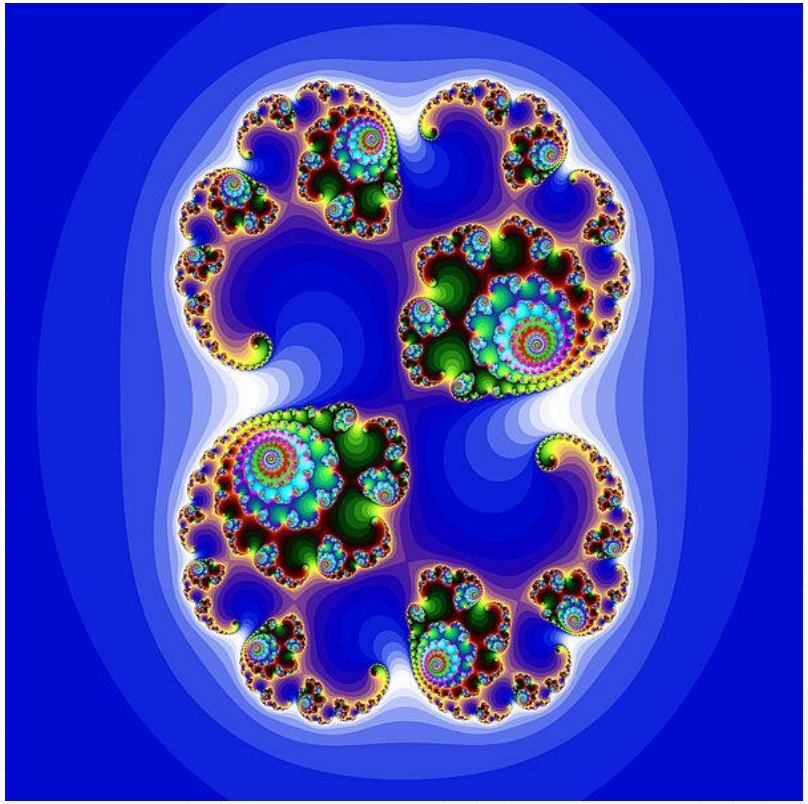


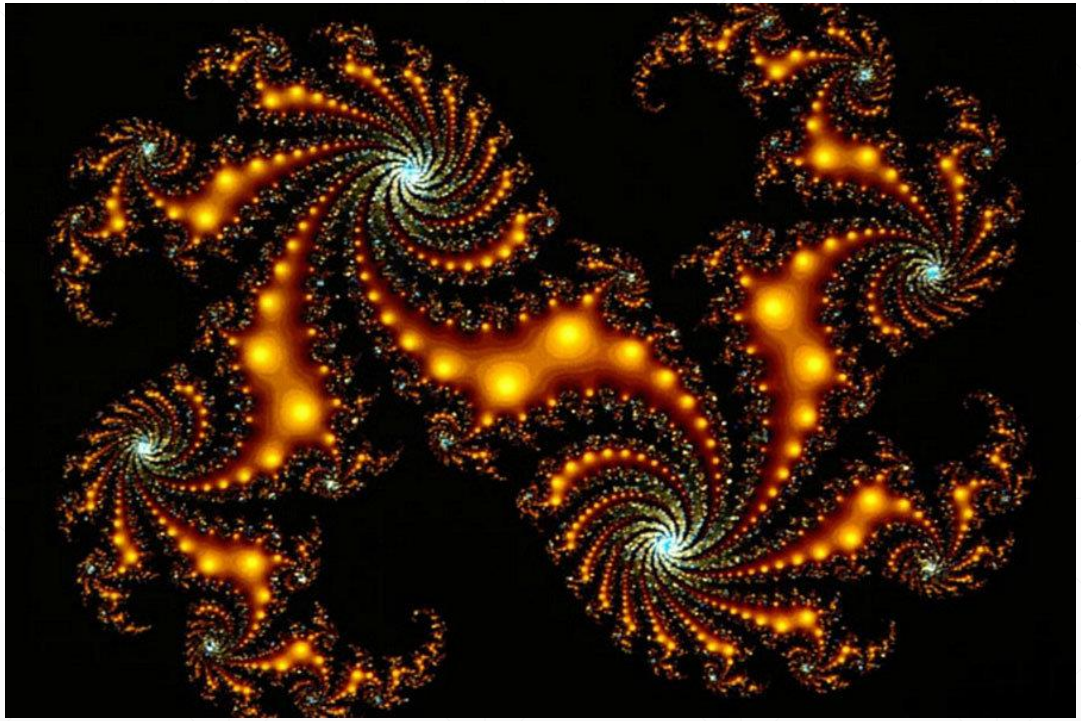
黑与白，密不可分
大与小，相辅相成



你在看我还是我在看你？

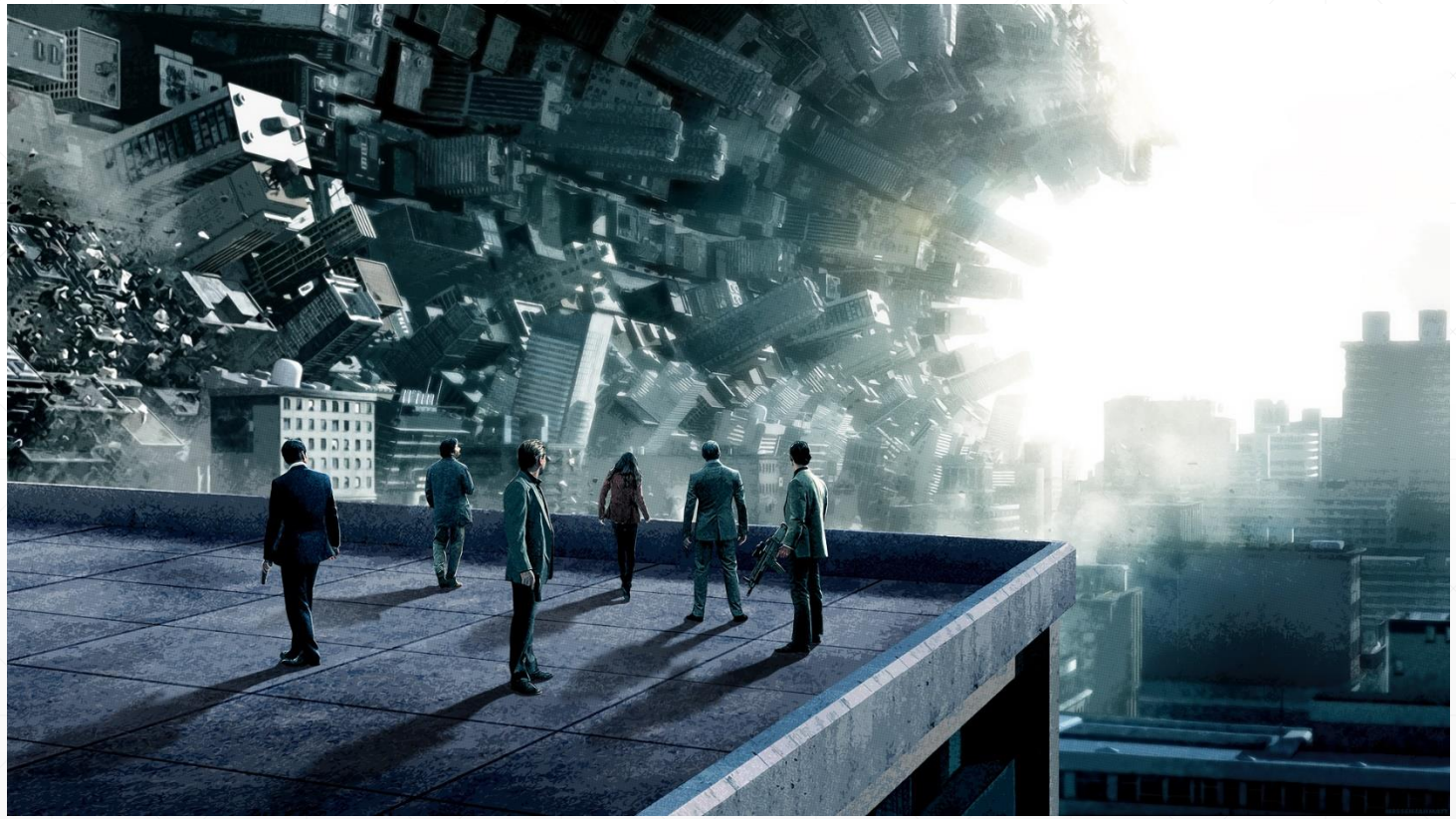
神奇的分形





想像着有一台可以放大无数倍的放大镜，
用它去观察分形图形，不管放大多少倍，
你总可以看到相似的精细结构……

递归与电影艺术



梦中人在梦中做梦……

——美国大片《盗梦空间》

欣赏科学技术与艺术结合带来的美

请使用搜索引擎搜索“**埃舍尔**”，观看这位艺术家所创作的其他作品。

观看美国大片《**盗梦空间**》，看看“梦中梦”的故事，注意一下梦中梦的时间流逝速度为什么不一样?

请使用搜索引擎搜索“**分形**”，了解一下分形是什么，欣赏一下分形的精美图形

软件开发中的递归

一个构成递归调用的函数

```
static void DonotRunMe()
{
    DonotRunMe();
}
```

递归就是“自己调用自己”。

引发“杯具”的递归

```
0 个引用
class Program
{
    0 个引用
    static void Main(string[] args)
    {
        Console.WriteLine("演示开始");
        DonotRunMe();
        Console.WriteLine("演示结束");
    }

    2 个引用
    static void DonotRunMe()
    {
        DonotRunMe();
    }
}
```

小知识：堆栈溢出（Stack Overflow）

程序代码其实是由“**线程（thread）**”负责执行的。

操作系统在创建线程（thread）时，会给每个线程配套一块内存区域，线程可以用这块区域存储一些数据。
这块内存区域被称为“**线程堆栈（thread stack）**”

线程堆栈有容量限制，当一个线程要保存的数据超过了这个容量时，就发生了“**堆栈溢出**”

“递归（recursive）”的算法

An algorithm is called **recursive** if it solves a problem by reducing it to an instance of the same problem with smaller input

意译：

一个递归的算法，会将一个难以处理的“大”问题的“规模”分多次地持续压缩，一直持续到压缩后的问题规模小到可以处理为止。其过程往往体现为代码要处理的数据量或计算量在递归前后不断“递减”。

实例：编个程序求n!

数学上阶乘的定义

1!=1

2!=1*2=1!*2

3!=1*2*3=2!*3

… …

n!=1*2*3*…*n=(n-1)!*n

从另一个角度分析阶乘的计算过程

从 n! 的数学公式可以看到，要计算 n! 可以先计算 (n-1)!，得到 (n-1)! 的结果之后，将其乘以 n 就得到 n!。

这个过程可以一直持续到 2!，这时，1! = 1已知，于是可以计算出 2! 。

由 2! 再倒推出 3!，4!，…，最后得到 n!，问题解决。

应用递归编写程序计算阶乘示例

示例：CalculateN

技术要点：

- 函数f(n)：返回n!
- 递归公式：f(n)=n*f(n-1)
- 结束递归的条件：n=1

注意掌握以下的技能……

使用Visual Studio设置断点，观察递归程序的执行过程
（1）设置断点
（2）查看变量
（3）即时窗口

小结：递归编程的“套路”

1. 每个递归函数的开头一定是判断递归结束条件是否满足的语句（一般是if语句）
2. 函数体一定至少有一句是“自己调用自己”的。
3. 每个递归函数一定有一个控制递归可以终结的变量（通常是作为函数的参数而存在）。每次自己调用自己时，此变量会变化（通常是变小），并传送给被调用的函数。

小结：递归的特点

1. 先从大到小，再从小到大。
2. 每个步骤要干的事情都是类似的，只不过其规模“小一号”。
3. 必须要注意保证递归调用的过程可以终结，否则，将导致“**堆栈溢出（stack overflow）**”。

下面我们再来看看另一种编程技巧——**递推**。

```
private long Factorial(int n)
{
    if (n == 1)
       return 1;
    long ret;
    ret = Factorial(n - 1) * n;
    return ret;

}
```

递归

VS.

```
private long Factorial2(int n)
{
    long result = 1;
    for(int i = 1; i <= n; i++)
    {
        result *= i;
    }
    return result;
}
```

递推

- “递归”是“由后至前再回来”，要求第n项，先求第n-1项，……，倒推到前面的可计算出的某项，然后再返回。
- “递推”是“从前到后”，先求第1项，然后，在此基础上求第2项，第3项，直至第n项。

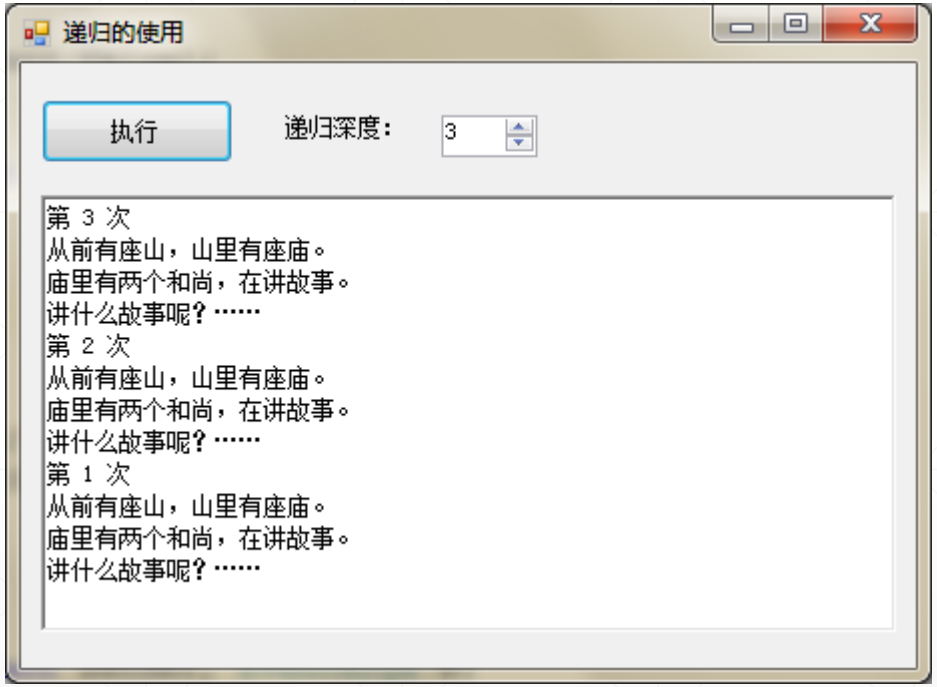
结论：递归与递推（iterative）是等价的。

小试身手：掌握最基本的递归编程技巧

编程计算“1+2+..+n = ？”

1. 使用递推的方法
2. 用递归的方法

小试身手：理解递归的执行顺序



Recursion示例

动手动脑

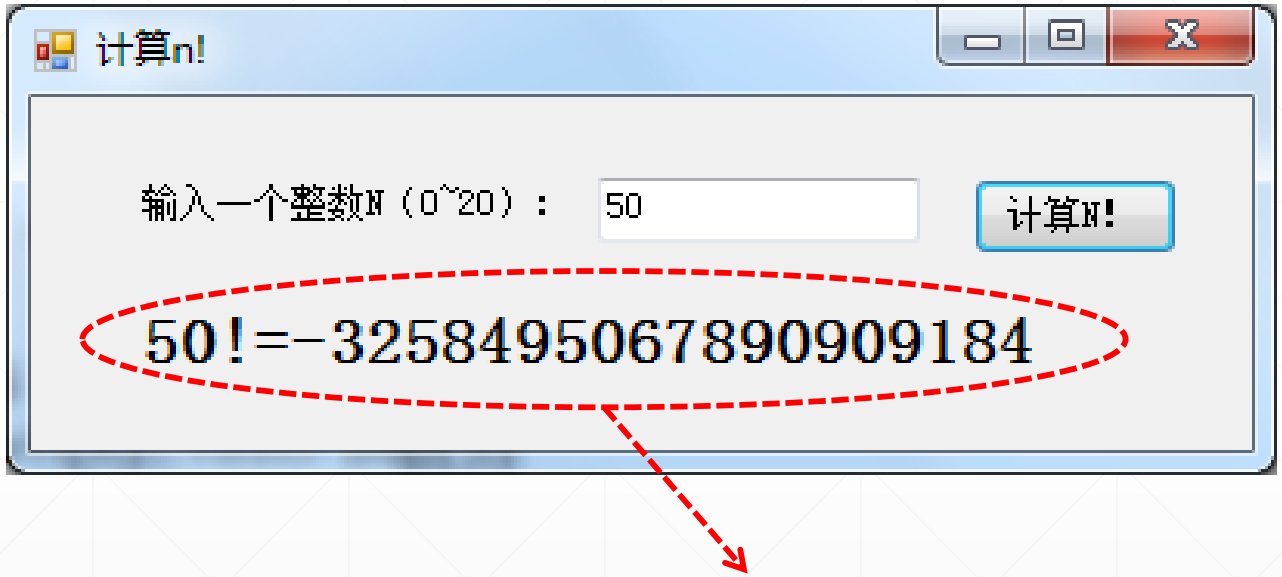
（1）按照代码注释中的提示，移动一下示例中的两句代码位置，发生了什么？

（2）使用设置断点的方法，单步执行，弄清楚代码的执行次序

关于递归，再多说几句……

- 在软件科学中，递归这种思想有着重要的应用，比如，许多计算机算法都是用递归实现的。
- 在具体的软件开发实践中，递归也用得非常多。
- 对于初学者而言，要一下子理解递归比较困难，只能在开发实践中慢慢体会，最终方能灵活地应用递归解决实际问题。

警告！警告！程序出错！



晕！阶乘数怎么可能出现负数？

参看示例：CalculateN

问题的根源：计算机能表示的数是有范围的！

C#中int类型的数值占32位，是有符号的，最高一位是符号位，表示“正负”

正数：000……0 ~ 011……1 即 32个全“0”到 “0”+31个全“1”
负数：100……0 ~ 111……1 即 “1”+31个全“0” 到 32个全“1”

$int.MaxValue = 2^{31}-1 = 2,147,483,647$
$int.MinValue = -2^{31} = -2,147,483,648$

类似地，long类型的数值占64位：

$long.MaxValue = 2^{63}-1 = 9,223,372,036,854,775,807$
$long.MinValue = -2^{63} = -9,223,372,036,854,775,808$

示例程序计算阶乘出现错误的原因

由于计算机使用固定的位数来保存数值，因此，能处理的数值大小是有限的，当要处理的数值超过了这一范围时，计算机将会自动截断数值的二进制表示为它所能处理的最多位数，这将导致错误的处理结果。

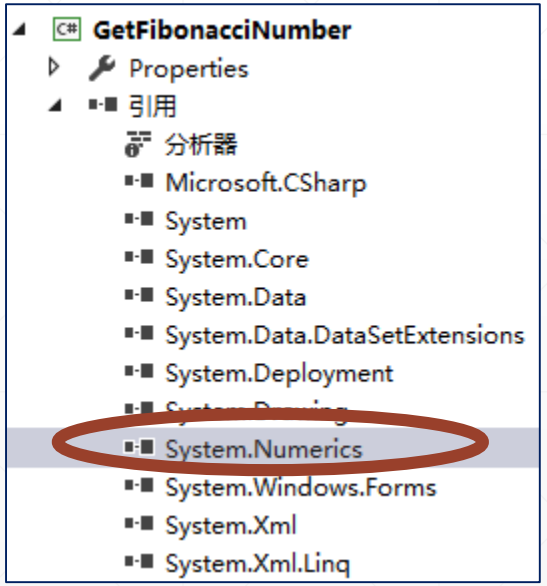
如果我们的确需要处理很大的整数，其数值大到甚至超过了long类型的最大值$2^{63}-1$，是不是就没有办法了呢?

处理巨大的整数

- .NET 4.0以上版本提供了一个BigInteger类，支持大整数的加减乘除运算。

```
BigInteger bi=long.MaxValue;
bi *= 2;
Console.WriteLine(bi);
Console.WriteLine(bi / 4);
```

注意：
BigInteger类型定义于System.Numerics中，需要为项目添加对这一程序集的引用



GetFibonacciNumber示例

计算机的怪脾气……

请看以下代码：

```
double i = 0.0001;
double j = 0.000100000000000000001;
Console.WriteLine(i==j);  //输出：true
```

好奇怪啊！“i”和“j”并不相等啊，为什么计算机告诉我这两个数是相等的？

揭秘！

计算机不能精确地表达浮点数（特殊形式的除外），因此，当需要比较两个浮点数是否相等时，应该比较其差的绝对值是否在某个允许范围之内即可，无法做到像数学那样的精确比较。

科学计数法的计算机表达方式，表示10^{-10}

```
if ( Math.Abs(i - j) < 1e-10 )
    Console.WriteLine("true");
else
     Console.WriteLine("false");
```

参看示例： AreYouEqual

课后作业

光学不练一场空

请找出满是灰尘的高中数学书，复习一下杨辉三角形与组合数的相关知识

$\binom{0}{0}$ 1

$\binom{1}{0}\binom{1}{1}$ 1 1

$\binom{2}{0}\binom{2}{1}\binom{2}{2}$ 1 2 1

$\binom{3}{0}\binom{3}{1}\binom{3}{2}\binom{3}{3}$ 1 3 3 1

杨辉三角形可用来计算组合数

作业：使用计算机编程计算组合数

（1）使用组合数公式利用n!来计算

$$C_n^k = \frac{n!}{k!(n-k)!}$$

（2）使用递推的方法用杨辉三角形计算

$$C_{n+1}^k = C_n^{k-1} + C_n^k$$

（3）使用递归的方法用组合数递推公式计算

学霸挑战极限！

- **题目：编程列出指定文件夹下的所有文件**

分析与提示：

（1）一个文件夹下可以有多个文件，也可以有多个子文件夹，而每个子文件夹下又可以有多个文件和子文件夹……天然地构成一个递归结构。
（2）.NET提供一个**Directory**类用于访问文件夹，它的**GetFiles()**方法可以列出某文件夹下所有的文件，而**GetDirectories()**方法则可以列出此文件夹下的所有子文件夹，此类在**System.IO** 命名空间中。
（3）你能用**递推**与**递归**两种方式实现这一程序吗?