**递归算法及经典递归例子**

**递归（recursion）**：程序调用自身的编程技巧。

  递归满足2个条件：

    1）有反复执行的过程（调用自身）

    2）有跳出反复执行过程的条件（递归出口）

 递归**[算法](http://lib.csdn.net/base/datastructure)**是一种直接或者间接调用自身函数或者方法的算法，我们就可以利用大道至简的思想，把一个大的复杂的问题层层转换为一个小的和原问题相似的问题来求解的这样一种策略。递归往往能给我们带来非常简洁直观的代码 ，从而使我们的编码大大简化，然而递归的思维确实很我们的常规思维相逆的，我们通常都是从上而下的思维问题， 而递归趋势从下往上的进行思维。这样我们就能看到我们会用很少的语句解决了非常大的问题，所以递归策略的最主要体现就是小的代码量解决了非常复杂的问题。

递归算法解决问题的特点：

1）递归就是方法里调用自身。

2）在使用递增归策略时，必须有一个明确的递归结束条件，称为递归出口。

3）递归算法解题通常显得很简洁，但递归算法解题的运行效率较低。所以一般不提倡用递归算法设计程序。

4）在递归调用的过程当中系统为每一层的返回点、局部量等开辟了栈来存储。递归次数过多容易造成栈溢出等，所以一般不提倡用递归算法设计程序。

在做递归算法的时候，一定要把握住出口，也就是做递归算法必须要有一个明确的递归结束条件。这一点是非常重要的。其实这个出口是非常好理解的，就是一个条件，当满足了这个条件的时候我们就不再递归了。

递归**[算法](http://lib.csdn.net/base/datastructure)**一般用于解决三类问题：

(1)数据的定义是按递归定义的。(Fibonacci函数)

(2)问题解法按递归算法实现。([回溯](http://www.hudong.com/wiki/%E5%9B%9E%E6%BA%AF))

(3)数据的结构形式是按递归定义的。([树的遍历](http://www.hudong.com/wiki/%E6%A0%91%E7%9A%84%E9%81%8D%E5%8E%86)，图的搜索)

递归的缺点：

递归算法解题的运行效率较低。在递归调用的过程当中系统为每一层的返回点、局部量等开辟了栈来存储。递归次数过多容易造成[栈溢出](http://www.hudong.com/wiki/%E6%A0%88%E6%BA%A2%E5%87%BA)等。

递归与迭代都是基于控制结构：迭代用重复结构，而递归用选择结构。

**递归例子：**

**（1）阶乘**

n! = n \* (n-1) \* (n-2) \* ...\* 1(n>0)

//阶乘

int recursive(int i)

{

int sum = 0;

if (0 == i)

return (1);

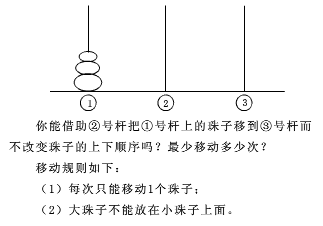
else

sum = i \* recursive(i-1);

return sum;

}

**（2）汉内塔问题**



//汉内塔

void hanoi(int n,int p1,int p2,int p3)

{

if(1==n)

cout<<"盘子从"<<p1<<"移到"<<p3<<endl;

else

{

hanoi(n-1,p1,p3,p2);

cout<<"盘子从"<<p1<<"移到"<<p3<<endl;

hanoi(n-1,p2,p1,p3);

}

}

**（3）全排列**

  从n个不同元素中任取m（m≤n）个元素，按照一定的顺序排列起来，叫做从n个不同元素中取出m个元素的一个排列。当m=n时所有的排列情况叫全排列。

  如1,2,3三个元素的全排列为：

  1,2,3

  1,3,2

  2,1,3

  2,3,1

  3,1,2

  3,2,1

//全排列

inline void Swap(int &a,int &b)

{

int temp=a;

a=b;

b=temp;

}

void Perm(int list[],int k,int m)

{

if (k == m-1)

{

for(int i=0;i<m;i++)

{

printf("%d",list[i]);

}

printf("n");

}

else

{

for(int i=k;i<m;i++)

{

Swap(list[k],list[i]);

Perm(list,k+1,m);

Swap(list[k],list[i]);

}

}

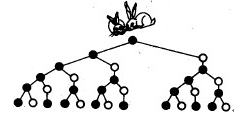
}

**（4）斐波那契数列**

  斐波纳契数列，又称黄金分割数列，指的是这样一个数列：1、1、2、3、5、8、13、21、……

  这个数列从第三项开始，每一项都等于前两项之和。

  有趣的兔子问题：



  一般而言，兔子在出生两个月后，就有繁殖能力，一对兔子每个月能生出一对小兔子来。如果所有兔子都不死，那么一年以后可以繁殖多少对兔子？

  分析如下：

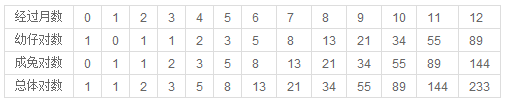
  第一个月小兔子没有繁殖能力，所以还是一对；

  两个月后，生下一对小兔子，总数共有两对；

  三个月以后，老兔子又生下一对，因为小兔子还没有繁殖能力，总数共是三对；

  ……

  依次类推可以列出下表：



//斐波那契

long Fib(int n)

{

 if (n == 0)

  return 0;

 if (n == 1)

  return 1;

 if (n > 1)

  return Fib(n-1) + Fib(n-2);

}

**（4）判定一系列字符串中是否有相同的内容**

public class T {

public static void main(String[] args) {

String[] a = {"a1","a2","a3","b3","c","b","33","33"};

boolean b = new T().fun(0, a);

System.out.println(b);

}

public boolean fun(int n,String[] a){

boolean b = false;

if(n == a.length){

b = true;

}else{

for(int i = n; i < a.length-1; i++){

System.out.println(n+" "+(i+1));

if(a[n].equals(a[i+1])){

return false;

}

}

n++;

fun(n,a);

}

return b;

}

}

**（5）求数组中的最大值**

用递归算法求数组中的最大值。

**/\*\***

**\* 用递归算法求数组中的最大值**

**\* @param a 数组**

**\* @param low 数组下标**

**\* @param heigh 数组上标**

**\* @return**

**\*/**

**public static int Max(int[] a, int low, int heigh) {**

**int max;**

**if(low > heigh-2) {**

**if(a[low] > a[heigh]) max = a[low];**

**else max = a[heigh];**

**}else {**

**int mid = (low + heigh)/2;**

**int max1 = Max(a, low, mid);**

**int max2 = Max(a, mid+1, heigh);**

**max = max1>max2 ? max1 : max2;**

**}**

**return max;**

**}**

**（6）**

**数字塔问题**

用递归算法求解数字塔问题。

n=1时

1

n=2时

1

2      2

n=3时

1

2      2

3      3      3

n=4时

1

2      2

3      3      3

4      4      4      4

**/\*\***

**\* 用递归算法求解数字塔问题**

**\* @param n 数字塔的行数**

**\* @return 数字塔的字符串**

**\*/**

**public static String tourData(int n) {**

**String str = new String();**

**if(1 == n) {**

**str = rowData(n) + "\n";**

**return str;**

**}**

**else {**

**str = tourData(n-1) + rowData(n) + "\n";**

**}**

**return str;**

**}**

**private static String rowData(int n) {**

**String str = new String();**

**for(int i=0; i<n; i++) {**

**str = str+ n + " ";**

**}**

**return str;**

**}**

汉诺塔问题递归算法分析：

　　一个庙里有三个柱子，第一个有64个盘子，从上往下盘子越来越大。要求庙里的老和尚把这64个盘子全部移动到第三个柱子上。移动的时候始终只能小盘子压着大盘子。而且每次只能移动一个。

　　1、此时老和尚（后面我们叫他第一个和尚）觉得很难，所以他想：要是有一个人能把前63个盘子先移动到第二个柱子上，我再把最后一个盘子直接移动到第三个柱子，再让那个人把刚才的前63个盘子从第二个柱子上移动到第三个柱子上，我的任务就完成了，简单。所以他找了比他年轻的和尚（后面我们叫他第二个和尚），命令：

          ① 你丫把前63个盘子移动到第二柱子上

          ② 然后我自己把第64个盘子移动到第三个柱子上后

          ③ 你把前63个盘子移动到第三柱子上

      2、第二个和尚接了任务，也觉得很难，所以他也和第一个和尚一样想：要是有一个人能把前62个盘子先移动到第三个柱子上，我再把最后一个盘子直接移动到第二个柱子，再让那个人把刚才的前62个盘子从第三个柱子上移动到第三个柱子上，我的任务就完成了，简单。所以他也找了比他年轻的和尚（后面我们叫他第三和尚），命令：

          ① 你把前62个盘子移动到第三柱子上

          ② 然后我自己把第63个盘子移动到第二个柱子上后

          ③ 你把前62个盘子移动到第二柱子上

　　3、第三个和尚接了任务，又把移动前61个盘子的任务依葫芦话瓢的交给了第四个和尚，等等递推下去，直到把任务交给了第64个和尚为止（估计第64个和尚很郁闷，没机会也命令下别人，因为到他这里盘子已经只有一个了）。

　　4、到此任务下交完成，到各司其职完成的时候了。完成回推了：

第64个和尚移动第1个盘子，把它移开，然后第63个和尚移动他给自己分配的第2个盘子。

第64个和尚再把第1个盘子移动到第2个盘子上。到这里第64个和尚的任务完成，第63个和尚完成了第62个和尚交给他的任务的第一步。

　　从上面可以看出，只有第64个和尚的任务完成了，第63个和尚的任务才能完成，只有第2个和尚----第64个和尚的任务完成后，第1个和尚的任务才能完成。这是一个典型的递归问题。 现在我们以有3个盘子来分析：

第1个和尚命令：

          ① 第2个和尚你先把第一柱子前2个盘子移动到第二柱子。（借助第三个柱子）

          ② 第1个和尚我自己把第一柱子最后的盘子移动到第三柱子。

          ③ 第2个和尚你把前2个盘子从第二柱子移动到第三柱子。

　　　很显然，第二步很容易实现（哎，人总是自私地，把简单留给自己，困难的给别人）。

其中第一步，第2个和尚他有2个盘子，他就命令：

          ① 第3个和尚你把第一柱子第1个盘子移动到第三柱子。（借助第二柱子）

          ② 第2个和尚我自己把第一柱子第2个盘子移动到第二柱子上。

          ③ 第3个和尚你把第1个盘子从第三柱子移动到第二柱子。

　　　同样，第二步很容易实现，但第3个和尚他只需要移动1个盘子，所以他也不用在下派任务了。（注意：这就是停止递归的条件，也叫边界值）

第三步可以分解为，第2个和尚还是有2个盘子，命令：

          ① 第3个和尚你把第二柱子上的第1个盘子移动到第一柱子。

          ② 第2个和尚我把第2个盘子从第二柱子移动到第三柱子。

          ③ 第3个和尚你把第一柱子上的盘子移动到第三柱子。

分析组合起来就是：1→3 1→2 3→2 借助第三个柱子移动到第二个柱子 |1→3 自私人留给自己的活| 2→1 2→3 1→3借助第一个柱子移动到第三个柱子|共需要七步。

如果是4个盘子，则第一个和尚的命令中第1步和第3步各有3个盘子，所以各需要7步，共14步，再加上第1个和尚的1步，所以4个盘子总共需要移动7+1+7=15步，同样，5个盘子需要15+1+15=31步，6个盘子需要31+1+31=64步……由此可以知道，移动n个盘子需要（2的n次方）-1步。

　　　从上面整体综合分析可知把n个盘子从1座（相当第一柱子）移到3座（相当第三柱子）：

（1）把1座上（n-1）个盘子借助3座移到2座。

     （2）把1座上第n个盘子移动3座。

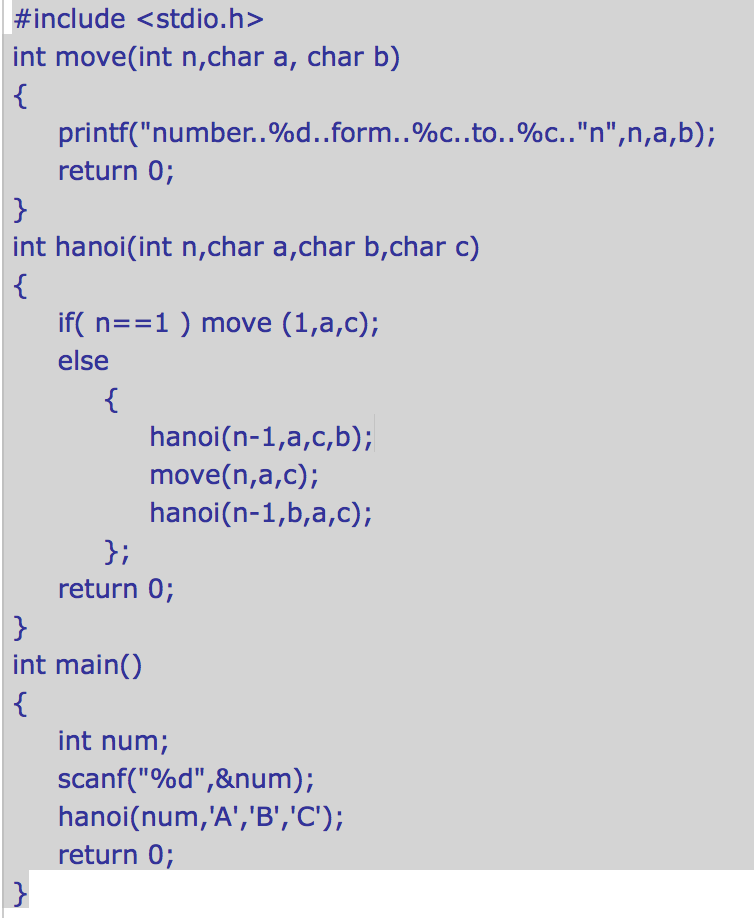
（3）把2座上（n-1）个盘子借助1座移动3座。

下面用hanoi（n,a,b,c）表示把1座n个盘子借助2座移动到3座。

很明显:    (1)步上是 hanoi(n-1,1,3,2)

               (3)步上是 hanoi(n-1,2,1,3)

用C语言表示出来，就是：



习题：

1 趣味问题——猴子吃桃。海滩上有一堆桃子，五只猴子来分。第一只猴子把这堆桃子凭据分为五份，多了一个，这只猴子把多的一个扔入海中，拿走了一份。第二只猴子把剩下的桃子又平均分成五份，又多了一个，它同样把多的一个扔入海中，拿走了一份，第三、第四、第五只猴子都是这样做的，问海滩上原来最少有多少个桃子？

2 趣味问题——年龄。有5个人坐在一起，问第五个人多少岁？他说比第4个人大2岁。问第4个人岁数，他说比第3个人大2岁。问第三个人，又说比第2人大两岁。问第2个人，说比第一个人大两岁。最后问第一个人，他说是10岁。请问第五个人多大？用递归算法实现。