

# Java狂飙之路2-递归算法

2020年12月10日 木曜日 10:12

★ 概述:简单的说就是函数自身直接或间接**调用函数本身**.

★ 递归算法的基本思路

1. 对于一个复杂的问题,把原问题分解为若干个相对简单的子问题,继续下去直到子问题简单到能够直接求解,也就是说找到了递归的**出口**
2. 在做递归算法的时候,一定要把握住出口,也就是递归算法必须要有一个**明确**的递归**结束条件**.当满足了这个条件的时候我们就结束递归.

★ 应用场景

1. 一个功能被重复使用,并每次使用时,参与运算的结果和上一次调用有关,这是就可以使用递归来解决这个问题.
2. 使用要点⚠
  1. 递归一定要明确结束条件
  2. 注意递归的次数

## 自我理解(洋葱,套娃...)

1. 核心条件结果的逆向包围,如同洋葱,洋葱芯

2.



...

第二次不符合核心条件  
后输出结果(包含前两次结果)

第一次不符合条件后的  
输出结果(包含核心条件结果)

核心条件的输出结果

## ★ 难点 ⚠

1. 核心结束条件的精准剖析
2. 实际运用场景



```
1 package algorithm;
2
3 /**
4  * @author Yuki RecursionAlgorithm 递归算法实例
5  */
6 public class RecursionAlgorithm {
7     public static void main(String[] args) {
8         int i = 4;
9         System.out.println(num(i)); // 打印输出结果
10    }
11
12    /**
13     * num 自定义输出指定方法结果值,采用递归算法
14     * @param i给定值
15     */
16    private static int num(int i) {
17        int n = 1;
18        if (i >= 0) { // 保证递归算法有意义
19            if (i == 1 || i == 0) { // 递归方法的核心条件
20                return n;
21            }
22            return num(i - 1) + num(i - 2); // 递归算法实际运用
23        } else {
24            return i;
25        }
26    }
27 }
28
29 /**
30  * 运行思路:
31  * num(4)=num(3)+num(2) 即3+2=5;
32  * num(3)=num(2)+num(1) 即2+1=3; ↑
33  * num(2)=num(1)+num(0) 即1+1=2; ↑
34  */
```

结果逆向包围

Console

