

04-C 进制转换

#数据结构邓神

典型应用场合

逆序输出	<ul style="list-style-type: none">• conversion.• <u>输出次序与处理过程颠倒</u>；<u>递归深度和输出长度不易预知</u>
递归嵌套	<ul style="list-style-type: none">• stack permutation + parenthesis• 具有自相似性的问题可递归描述，但分支位置和嵌套深度不
延迟缓冲	<ul style="list-style-type: none">• evaluation• 线性扫描算法模式中，在预读足够长之后，方能确定可处理

我们这里以进制的转换

进制转换实现就可以用栈

进制转换

短除法：

$80(10) \rightarrow (2)$

对2取余数

进制转换

$89_{(10)} \sim \boxed{}_{(2)}$

89	1
44	0
22	0
11	1
5	1
2	0
1	1
0	

将所得到的这一串比特位记录下来

反向拿出 1011001

进制转换

$89_{(10)} \sim 1011001_{(2)}$

89	1
44	0
22	0
11	1
5	1
2	0
1	1
0	

$2013_{(10)} \sim 31023_{(5)}$

2013	3
402	2
80	0
16	1
3	3
0	

Data Structures & Algorithms (Fall 2013), Tsinghua University

我们发现这个问题可以用栈很好的解决

实现

```
// 将输入整数n (10) 转换成 base类型
void convert(stack<char> & S, int n, int base){
    static char digit[] =
    {'0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B', 'C', 'D', 'E', 'F'};
    while(n > 0){
        S.push(digit[n%base]);
        n /= base;
    }
}
```