# **参考网页：https://www.cnblogs.com/hsrzyn/archive/2009/12/21/1629274.html**

# **[“中序表达式”转换为“前序表达式”、“后序表达式”](https://www.cnblogs.com/hsrzyn/archive/2009/12/21/1629274.html)**

首先看下面所示表格：

|  |  |
| --- | --- |
| 中序表达式 | 2\*3/(2-1)+3\*(4-1) |
| 前序表达式 | +/\*23-21\*3-41 |
| 后序表达式 | 23\*21-/341-\*+ |

　中序表达式对我们而言是很直观的（我们平时接触的就是这个），但计算机处理起来比较麻烦（括号、优先级之类的），前序和后序表达式中没有括号，而且在计算中只需单向扫描，不需要考虑运算符的优先级。  
　　以前序表达式“+/\*23-21\*3-41”为例，从右往左，先取出两个操作数“1”、“4”和一个运算符“-”，计算“4-1”，将结果3回填到字符串中，现在字符串变为“+/\*23-21\*33”。  
　　再从右至左取两个数“3”、“3”和“\*”，计算“3\*3”，将结果“9”回填到字符串，得“+/\*23-219’”,

　　再取数，连续取出“9”、“1”、“2”，直到取出一个运算符“-”，将与运算符最近的两个操作数进行计算，即“2-1”得“1”，回填字符串中，现在为“+/\*239”  
　　重复上述步骤，取出“2\*3”=6，回填字符串得到“+/619”，

　　再取“6/1”=6，得到“+69”,

　　再取“6+9”=15。运算完毕。

　　即**从右至左取数，直到取出一个运算符，将刚取出的紧挨着运算符的两个操作数按运算符进行计算，结果回填至运算符。重复该步骤，直到最后只剩下一个字符串则剩下的字符串即为结果。  
　　后序表达式的字符串扫描方式正好和前序相反，是从左往右扫描，规则类似。**

[IMG_256](javascript:void(0);)

中序表达式转前序表达式步骤  
1、反转输入字符串，如“2\*3/(2-1)+3\*(4-1)” 反转后为“ )1-4(\*3+)1-2(/3\*2”，  
2、从字符串中取出下一个字符  
　　2.1.如果是操作数，则直接输出  
　　2.2.如果是“)”，压入栈中  
　　2.3.如果是运算符但不是“(”，“)”,则不断循环进行以下处理  
　　　　2.3.1.如果栈为空，则此运算符进栈，结束此步骤  
　　　　2.3.2.如果栈顶是“)”,则此运算符进栈，结束此步骤  
　　　　2.3.2.如果此运算符与栈顶优先级相同或者更高，此运算符进栈，结束此步骤  
　　　　2.3.4.否则，运算符连续出栈，直到满足上述三个条件之一，然后此运算符进栈  
　　2.4、如果是“(”，则运算符连续出栈，直到遇见“)”为止,将“)”出栈且丢弃之  
3、如果还有更多的字符串，则转到第2步  
4、不在有未处理的字符串了，输出栈中剩余元素  
5、再次反转字符串得到最终结果

[IMG_257](javascript:void(0);)