# 实验3中缀算术表达式求值的指导

## 1题目

输入一个中缀算术表达式，计算其结果。对输入的表达式，做如下假设：

1. 只考虑+、-、\*、/这四种运算符，中缀表达式中只有一种括号（）；

（2）输入的中缀表达式中数字只有整数，没有小数；

（3）假定输入表达式是合法的。

如果加入乘方'^'运算，如何设计？

## 2测试用例

中缀表达式：（1）1+(2-3)\*4+4/2 （2）a+(b-c)\*d

后缀表达式：（1）1 2 3-4\*+4 2 / + （2）abc-d\*+

## 3算法

计算中缀算术表达式是有关栈的经典算法题，很多公司面试或者笔试的时候都会把计算中缀表达式作为一个考察点。这里先讨论如何将中缀表达式转化为后缀表达式，再讨论如何计算后缀表达式，最后讨论如何一步到位，直接计算中缀表达式的值。

### 3.1中缀表达式转为后缀表达式

中缀转化为后缀，要先设一个栈存储运算符，从左到右遍历表达式，主要遵循以下原则：

（1）读到操作数，直接输出；

（2）栈为空时，遇到运算符，入栈；

（3）遇到左括号，将其入栈；

（4）遇到右括号，执行出栈操作，并将出栈的元素输出，直到弹出栈的是左括号，左括号不输出， 左括号也要弹出；

（5）遇到其他运算符’+”-”\*”/’时，弹出所有优先级大于或等于该运算符的栈顶元素，然后将该运算符入栈， 遇到左括号时不能再弹出；

（6）最终将栈中的元素依次出栈，输出。

经过上面的步骤，得到的输出就是转换得到的后缀表达式。

说明：输出的后缀表达式可以一个队列存储。

### 3.2后缀表达式求值

设后缀表达式的栈为post,求值计算采用辅助栈result实现，算法如下：

(1)从后缀表达式栈post中弹出栈顶元素，若是操作数，压入result；若是运算符oprater，则从result中取出两个元素，进行oprator运算，结果压入result。

(2)若post为空，结束，result存放的结果即为运算结果；否则转向（1）。

假定待求值的后缀表达式为：6 5 2 3 + 8 \* + 3 + \*，则其求值过程如下：

* 遍历表达式，遇到数字首先放入栈，此时栈如下 6 5 2 3
* 接着读到+，则弹出3和2，执行3+2，将结果5压栈 6 5 5
* 读到8，压栈 6 5 5 8
* 读到 \*, 弹出8和5，执行8\*5，将结果40压栈 6 5 40
* 读到 +,弹出40和5，执行40+5，将结果45压栈 6 45
* 读到 3,压栈 6 45 3
* 读到 +,弹出3和45，执行3+45，将结果48压栈 6 48
* 读到 \*,弹出48和6，执行48\*6，将结果288压栈 288
* 最后结果288

链接：https://www.jianshu.com/p/fcd2b521a3e2

### 3.3直接计算中缀表达式

设置两个工作栈，一个操作数栈，一个操作符栈，在（自左至右）扫描算术表达式时，遇到操作数直接入操作数栈，若遇到操作符，则根据操作符优先级判断下一步操作（“操作符优先级规则”）：若其优先级高于栈顶操作符，则入栈，否则（相等或小于），弹出栈顶算符并从操作数栈弹出两个操作数，计算，将计算结果入操作数栈，继续比较与栈顶操作符的优先级，若仍然等于或低于之，则计算，直至大于之，则将此操作符入栈；左括号一定入栈，且其优先级低于后续来到的任何操作符，右括号一定出栈并计算直至遇到左括号，另外栈的开始以“#”开始，算术表达式以“#”结束，做结束标志。当栈空时，计算结束。

### 4 算法对数据的操作过程

**例子：**中缀表达式 a + b \* c + (d \* e + f) \* g ，其转换成后缀表达式则为a b c \* + d e \* f + g \* +。

| **步骤 扫描** | **stack 栈** | **输出** |
| --- | --- | --- |
| 1 a | 空 | a |
| 2 a+ | + | a |
| 3 a+b | + | ab |
| 4 a+b\* | +\* | ab |
| 5 a+b\*c | +\* | abc |
| 6 a+b\*c+ | + | abc\*+ |
| 7 a+b\*c+( | +( | abc\*+ |
| 8 a+b\*c+(d | +( | abc\*+d |
| 9 a+b\*c+(d\* | +(\* | abc\*+d |
| 10 a+b\*c+(d\*e | +(\* | abc\*+de |
| 11 a+b\*c+(d\*e+ | +(+ | abc\*+de\* |
| 12 a+b\*c+(d\*e+f | +(+ | abc\*+de\*f |
| 13 a+b\*c+(d\*e+f) | + | abc\*+de\*f+ |
| 14 a+b\*c+(d\*e+f)\* | +\* | abc\*+de\*f+ |
| 15 a+b\*c+(d\*e+f)\*g | +\* | abc\*+de\*f+g |
| 16 a+b\*c+(d\*e+f)\*g# | 空 | abc\*+de\*f+g\*+ |

注意：第6步，读到+,因为栈顶元素\*的优先级高，所以\*出栈，栈中下一个元素+优先级与读到的操作符+一样，所以也要弹出,然后再将读到的+压入栈中。第13步，读到)，则直接将栈中元素弹出直到遇到(为止。这里左括号前只有一个操作符+被弹出。

### 5 扩展部分设计

如果加入乘方'^'运算，要注意乘方运算是右结合的,需要修改两处：

(1)将乘方添加到优先级中：

1：（

2：+ -

3：\* /

4：^

5：）

(2)在读中缀表达式的时候，如果读到乘方^，就将它放进符号栈中。因为乘方的优先级是最高的，而且是右结合的，所以无论它前面出现的是什么运算，这些运算都不能执行。而且它本身能否执行也是不知道的，因此只能进栈。https://blog.csdn.net/qq\_26286193/article/details/80214805