



数据结构与算法 (Python版)

表达式转换 (上)

陈斌 北京大学 gischen@pku.edu.cn

中缀表达式

- ❖ 我们通常看到的表达式象这样： $B * C$ ，很容易知道这是B乘以C
- ❖ 这种操作符 (operator) 介于操作数 (operand) 中间的表示法，称为“**中缀**”表示法
- ❖ 但有时候中缀表示法会引起混淆，如
“ **$A + B * C$** ”
是 $A + B$ 然后再乘以C
还是 $B * C$ 然后再去加A？

中缀表达式中的优先级

❖ 人们引入了操作符“**优先级**”的概念来消除混淆

规定高优先级的操作符先计算

相同优先级的操作符从左到右依次计算

这样 $A+B*C$ 就没有疑义是A加上B与C的乘积

❖ 同时引入了**括号**来表示**强制**优先级，括号的优先级最高，而且在嵌套的括号中，内层的优先级更高

这样 $(A+B)*C$ 就是A与B的和再乘以C

全括号中缀表达式

- ❖ 虽然人们已经习惯了这种表示法，但计算机处理最好是能明确规定所有的计算顺序，这样无需处理复杂的优先规则
- ❖ 引入**全括号表达式**：在所有的表达式项两边都加上括号
 $A+B*C+D$ ，应表示为 $((A+(B*C))+D)$
- ❖ 可否将表达式中操作符的位置稍**移动**一下？

前缀和后缀表达式

❖ 例如中缀表达式 $A + B$

将操作符移到前面, 变为 “ $+AB$ ”

或者将操作符移到最后, 变为 “ $AB+$ ”

❖ 我们就得到了表达式的另外两种表示法:

“**前缀**” 和 “**后缀**” 表示法

以操作符相对于操作数的位置来定义

前缀和后缀表达式

❖ 这样 $A + \underline{B * C}$ 将变为前缀的 “ $+$ $\underline{A * B C}$ ” ,
后缀的 “ $\underline{A B C} * +$ ”

为了帮助理解，子表达式加了下划线

中缀表达式	前缀表达式	后缀表达式
$A + B$	$+ A B$	$A B +$
$A + B * C$	$+ A * B C$	$A B C * +$

前缀、中缀和后缀表达式

- ❖ 再来看中缀表达式 $(A+B)*C$ ，按照转换的规则，前缀表达式是 $*+ABC$ ，而后缀表达式是 $AB+C*$
- ❖ 神奇的事情发生了，在中缀表达式里必须的括号，在前缀和后缀表达式中消失了？
- ❖ 在前缀和后缀表达式中，**操作符的次序完全决定了运算的次序，不再有混淆**
所以在很多情况下，表达式的计算机表示都避免用复杂的中缀形式

前缀、中缀和后缀表达式

❖ 下面看更多的例子

中缀表达式	前缀表达式	后缀表达式
$A + B * C + D$	$++A*BCD$	$ABC*+D+$
$(A + B) * (C + D)$	$*+AB+CD$	$AB+CD+*$
$A * B + C * D$	$+*AB*CD$	$AB*CD*+$
$A + B + C + D$	$+++ABCD$	$AB+C+D+$

中缀表达式转换为前缀和后缀形式

❖ 目前为止我们仅手工转换了几个中缀表达式到前缀和后缀的形式

一定得有个算法来转换任意复杂的表达式

❖ 为了分解算法的复杂度，我们从“全括号”中缀表达式入手

我们看 $A+B*C$ ，如果写成全括号形式：

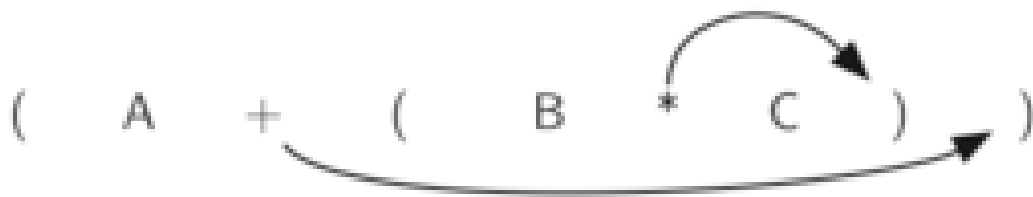
$(A+(B*C))$ ，显式表达了计算次序

我们注意到每一对括号，都包含了一组完整的操作符和操作数

中缀表达式转换为前缀和后缀形式

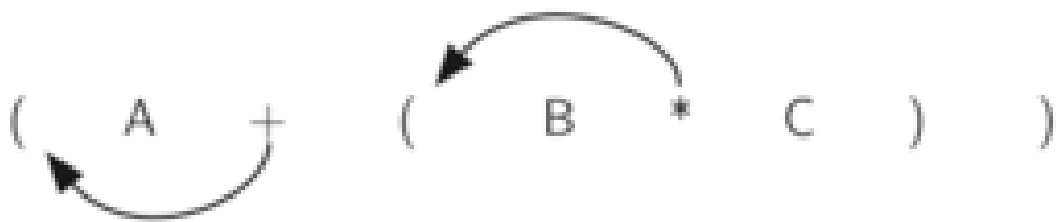
❖ 看子表达式 $(B * C)$ 的右括号，如果把操作符 $*$ 移到右括号的位置，替代它，再删去左括号，得到 $BC*$ ，这个正好把子表达式转换为后缀形式

进一步再把更多的操作符移动到相应的右括号处替代之，再删去左括号，那么整个表达式就完成了到后缀表达式的转换



中缀表达式转换为前缀和后缀形式

- ❖ 同样的，如果我们把操作符移动到**左括号**的位置替代之，然后删掉所有的右括号，也就得到了**前缀**表达式



中缀表达式转换为前缀和后缀形式

❖ 所以说，无论表达式多复杂，需要转换成前缀或者后缀，只需要两个步骤

将中缀表达式转换为全括号形式

将所有的操作符移动到子表达式所在的左括号（**前缀**）或者右括号（**后缀**）处，替代之，再删除所有的括号

