

中缀、前缀、后缀表达式简介

一、中缀、前缀、后缀表达式

1、中缀表达式

简介：中缀表达式就是常见的运算表达式。

如 $(3+4) \times 5 - 6$ 。

2、前缀表达式

简介：前缀表达式又称波兰式，前缀表达式的运算符位于操作数之前。

比如： $- \times + 3 4 5 6$

3、后缀表达式

简介：后缀表达式又称逆波兰表达式，与前缀表达式相似，只是运算符位于操作数之后。

比如： $3 4 + 5 \times 6 -$

二、前缀表达式的计算机求值

从右至左扫描表达式，遇到数字时，将数字压入堆栈，遇到运算符时，弹出栈顶的两个数，用运算符对它们做相应的计算（栈顶元素 op 次顶元素），并将结果入栈；重复上述过程直到表达式最左端，最后运算得出的值即为表达式的结果

例如： $- \times + 3 4 5 6$

- 从右至左扫描，将 6、5、4、3 压入堆栈
- 遇到+运算符，因此弹出 3 和 4（3 为栈顶元素，4 为次顶元素，注意与后缀表达式做比较），计算出 $3+4$ 的值，得 7，再将 7 入栈
- 接下来是 \times 运算符，因此弹出 7 和 5，计算出 $7 \times 5 = 35$ ，将 35 入栈
- 最后是 $-$ 运算符，计算出 $35 - 6$ 的值，即 29，由此得出最终结果

三、将中缀表达式转换为前缀表达式

转换步骤如下：

- 初始化两个栈：运算符栈 s_1 ，储存中间结果的栈 s_2
- 从右至左扫描中缀表达式
- 遇到操作数时，将其压入 s_2
- 遇到运算符时，比较其与 s_1 栈顶运算符的优先级
 - 如果 s_1 为空，或栈顶运算符为右括号 “ $)$ ”，则直接将此运算符入栈
 - 否则，若优先级比栈顶运算符的较高或相等，也将运算符压入 s_1
 - 否则，将 s_1 栈顶的运算符弹出并压入到 s_2 中，再次转到 (4-1) 与 s_1 中新的栈顶运算符相比较
- 遇到括号时
 - 如果是右括号 “ $)$ ”，则直接压入 s_1
 - 如果是左括号 “ $($ ”，则依次弹出 s_1 栈顶的运算符，并压入 s_2 ，直到遇到右括号 “ $)$ ” 为止，此时将这一对括号丢弃
- 重复步骤 2 至 5，直到表达式的最左边
- 将 s_1 中剩余的运算符依次弹出并压入 s_2
- 依次弹出 s_2 中的元素并输出，结果即为中缀表达式对应的前缀表达式

四、后缀表达式计算机求值

与前缀表达式类似，只是顺序是从左至右：

从左至右扫描表达式，遇到数字时，将数字压入堆栈，遇到运算符时，弹出栈顶的两个数，用运算符对它们做相应的计算（次顶元素 op 栈顶元素），并将结果入栈；重复上述过程直到表达式最右端，最后运算得出的值即为表达式的结果

例如后缀表达式 $3\ 4\ +\ 5\ \times\ 6\ -$

- 从左至右扫描，将 3 和 4 压入堆栈；
- 遇到+运算符，因此弹出 4 和 3（4 为栈顶元素，3 为次顶元素，注意与前缀表达式做比较），计算出 $3+4$ 的值，得 7，再将 7 入栈；
- 将 5 入栈；
- 接下来是 \times 运算符，因此弹出 5 和 7，计算出 $7\times 5=35$ ，将 35 入栈；
- 将 6 入栈；
- 最后是-运算符，计算出 $35-6$ 的值，即 29，由此得出最终结果。

五、将中缀表达式转换为后缀表达式

与转换为前缀表达式相似，步骤如下：

- 初始化两个栈：运算符栈 s1 和储存中间结果的栈 s2；
- 从左至右扫描中缀表达式；
- 遇到操作数时，将其压 s2；
- 遇到运算符时，比较其与 s1 栈顶运算符的优先级：
 - 如果 s1 为空，或栈顶运算符为左括号“（”，则直接将此运算符入栈；
 - 否则，若优先级比栈顶运算符的高，也将运算符压入 s1（**注意转换为前缀表达式时是优先级较高或相同，而这里则不包括相同的情况**）；
 - 否则，将 s1 栈顶的运算符弹出并压入到 s2 中，再次转到(4-1)与 s1 中新的栈顶运算符相比较；
- 遇到括号时：
 - 如果是左括号“（”，则直接压入 s1；
 - 如果是右括号“）”，则依次弹出 s1 栈顶的运算符，并压入 s2，直到遇到左括号为止，此时将这一对括号丢弃；
- 重复步骤 2 至 5，直到表达式的最右边；
- 将 s1 中剩余的运算符依次弹出并压入 s2；
- 依次弹出 s2 中的元素并输出，**结果的逆序即为中缀表达式对应的后缀表达式（转换为前缀表达式时不用逆序）**

例如，将中缀表达式 $1+(2+3)\times 4-5$ 转换为后缀表达式的过程如下：

扫描到的元素	s2(栈底->栈顶)	s1 (栈底->栈顶)	说明
1	1	空	数字，直接入栈
+	1	+	s1 为空，运算符直接入栈
(1	+ (左括号，直接入栈
(1	+ ((同上
2	1 2	+ ((数字
+	1 2	+ ((+	s1 栈顶为左括号，运算符直接入栈
3	1 2 3	+ ((+	数字
)	1 2 3 +	+ (右括号，弹出运算符直至遇到左括号
×	1 2 3 +	+ (×	s1 栈顶为左括号，运算符直接入栈
4	1 2 3 + 4	+ (×	数字
)	1 2 3 + 4 ×	+	右括号，弹出运算符直至遇到左括号
-	1 2 3 + 4 × +	-	-与+优先级相同，因此弹出+，再压入-
5	1 2 3 + 4 × + 5	-	数字
到达最右端	1 2 3 + 4 × + 5 -	空	s1 中剩余的运算符