# 前缀、中缀、后缀表达式

【<https://blog.csdn.net/Antineutrino/article/details/6763722>】

【https://www.cnblogs.com/ysocean/p/7910432.html】

【<https://blog.csdn.net/walkerkalr/article/details/22798365>】

【https://blog.csdn.net/yu757371316/article/details/48456413】

**前言：**

所谓的前缀、中缀、后缀表达式都是针对运算符与两个操作的位置而言的，如果运算符在两个操作数之前，则叫做前缀表达式；如果运算符在两个操作数中间，则叫做中缀表达式；如果运算符在两个操作数之后，则叫做后缀表达式。

## 人如何解析算数表达式？

如何解析算数表达式？或者换个说法，遇到某个算数表达式，我们是如何计算的？

### 求值：3+4-5

|  |  |  |
| --- | --- | --- |
| 读取元素 | 解析后的表达式 | 备注 |
| 3 | 3 |  |
| + | 3+ |  |
| 4 | 3+4 |  |
| - | 7 | 看到 - 后，可以计算3+4的值 |
|  | 7- |  |
| 5 | 7-5 |  |
| End | 2 | 达到表达式的末端，可以计算7-5的值 |

这个表达式，我们在看到3+4后不能直接计算其值，直到看到4后面的 - 号，因为减号的优先级和前面的加号一样，所以可以计算3+4的值了，如果4后面是\*或者/，那么就要在乘除过后才能做加法操作，比如：

### 2、求值：3+4\*5

|  |  |  |
| --- | --- | --- |
| 读取元素 | 解析后的表达式 | 备注 |
| 3 | 3 |  |
| + | 3+ |  |
| 4 | 3+4 |  |
| \* | 3+4\* | 不能计算3+4的值，因为4后面的\*号比+号优先级高 |
| 5 | 3+4\*5 | 看到5后可以计算4\*5的值 |
|  | 3+20 |  |
| End | 23 | 达到表达式的末端，可以计算3+20的值 |

这个不能先求3+4的值，因为4后面的\*号运算级别比前面的+号高。

通过这两个表达式的说明，我们可以总结解析表达式的时候遵循的几条规则：

1. 从左到右读取算式；
2. 已经读到了可以计算值的两个操作数和一个操作符时，可以计算，并用计算结果代替那两个操作数和一个操作符；
3. 继续这个过程，从左到右，能算就算，直到表达式的结尾；

## 计算机如何解析算数表达式？

对于前面的表达式3+4-5，我们人是有思维能力的，能根据操作符的位置，以及操作符的优先级别能算出该表达式的结果，但是计算机怎么算？

计算机必须要向前（从左到右）来读取操作数和操作符，等到读取足够的信息来执行一个运算时，找到两个操作数和一个操作符进行运算，有时候如果后面是更高级别的操作符或者括号时，就必须推迟运算，必须要解析到后面级别高的运算，然后回头来执行前面的运算。

我们发现这个过程是机器繁琐的，而计算机是一个机器，只认识高低电平，想要完成一个简单的表达式的计算，我们可能要设计出很复杂的逻辑电路来控制计算过程，那更不用说很复杂的算数表达式，所以这样来解析算术表达式时不合理的，那么我们应该采取什么办法呢？

请大家先看什么是前缀表达式、中缀表达式、后缀表达式：这三种表达式其实就是算术表达式的三种写法，以3+4-5为例：

1. 前缀表达式：操作符在操作数的前面，比如：+ - 5 4 3；
2. 中缀表达式：操作符在操作数的中间，这也是人类最容易识别的算术表达式，比如： 3 + 4 - 5；
3. 后缀表达式：操作符在操作数的后i按，比如：3 4 + 5 -

上面我们讲的是人是如何解析算术表达式，也就是解析中缀表达式，这是人最容易识别

的，但是计算机不容易识别，计算机容易识别的是前缀表达式和后缀表达式，将中缀表达式转换成前缀表达式或者后缀表达式之后，计算机能很快计算出表达式的值，那么中缀表达式是如何转换成前缀表达式和后缀表达式，以及计算机如何解析前缀表达式、中缀表达式、后缀表达式来得到结果呢？

下面先来看一下程序是如何将中缀表达式和后缀表达式，然后再看一看计算机时如何计算前缀表达式、中缀表达式、后缀表达式，以及性能又是如何：

## 前缀表达式（波兰表达式）：

### 什么是前缀表达式：

前缀表达式，指的是不包含括号，运算符放在两个运算对象的前面，严格从右向左进行（不再考虑运算符的有限规则），所有的计算按运算符出现的顺序。

**【注意】**前缀表达式是对中缀表达式从右向左解析，后缀表达式是对中缀表达式从左向右解析（这是计算机这样解析）。

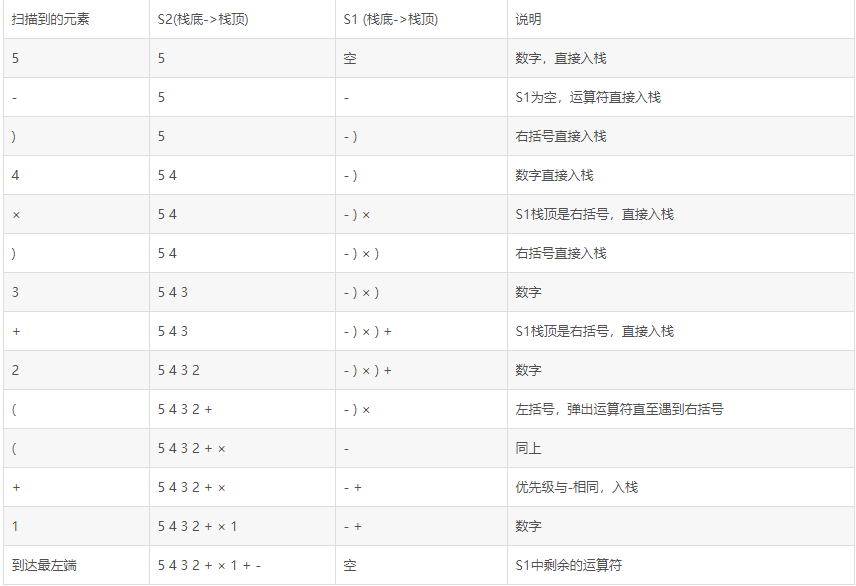
### 中缀表达式转换成前缀追表达式（程序操作）：

遵循以下步骤：

|  |  |
| --- | --- |
| **中缀表达式转换为前缀表达式** | |
| 1）初始化两个栈：运算符栈S1和存储中间结果的栈S2； | |
| 2）从右至左扫描中缀表达式； | |
| 3）遇到操作数时，直接将其压入S2； | |
| 4）遇到运算符时，比较其与S1栈顶运算符的优先级： | |
|  | 4-1）如果S1为空，或栈顶运算符为右括号，则直接将此运算符入栈； |
| 4-2）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S1； |
| 4-3）否则，将S1栈顶运算符弹出并压入到S2中，再次转到S1中新的栈顶运算符相比较； |
| 5）遇到括号时： | |
|  | 5-1）如果是右括号，则直接压入S1； |
| 5-2）如果是左括号，则依次弹出S1栈顶的运算符，并压入S2，直到遇到右括号为止，此时将这一对括号丢弃； |
| 6）重复步骤（2）至（5），直到表达式的最左边； | |
| 7）将S1中剩余的运算符依次弹出并压入S2； | |
| 8）依次弹出S2中元素并输出，结果即为中缀表达式对应的前缀表达式； | |

### 3、示例：

将中缀表达式“1+（（2+3）\*4）-5”转换为前缀表达式的过程如下：



转换成的前缀表达式是：“- + 1 x + 2 3 4 5”【“5 4 3 2 + x 1 + -”是栈底到栈顶】

### 4、代码：

|  |
| --- |
|  |

【DataStructureAndAlgorithmDemo/src/dataStructure/ds\_04\_expression/exp\_1\_prefixExpression/\*】

### 计算机怎样解析前缀表达式？

#### 1）步骤：

方法一）从右至左扫描前缀表达式，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，并将结果入栈；重复上述过程，直到表达式最左端，最后得出的值即为表达式的结果。

方法二）从做至右扫描前缀表达式，运算符和操作数依次入栈，连续出现的两个操作数和在它们之前且紧靠它们的运算符构成一个最小表达式。

#### 示例：

例如前缀表达式 “- x + 3 4 5 6”【中缀表达式为“（3+4）x5-6”】

①从右至左扫描，将6，5，4，3压入堆栈；

②遇到+运算符，因此弹出3和4（3为栈顶，4为次栈顶，注意与后缀表达式做比较），计算3+4的值，得7，再将7入栈；

③接下来是x运算符，因此弹出7和5，计算7x5的值为35，将35入栈；

④最后是-运算符，计算出35-6的值，为29，由此得出最终结果；

可以看出，用计算机计算前缀表达式的值是很容易的。

#### 3）代码：

|  |
| --- |
|  |

## 四、中缀表达式：

中缀表达式是一种通用的算术或逻辑公式表示方法，操作符以中缀形式处于操作数的中间，中缀表达式是人们常用的算术表示方法。

### 中缀表达式转换成前缀表达式：

以 “a + b x c - (d + e)”和 “3 + 4 -5”为例

### 中缀表达式转换成后缀表达式：

## 五、后缀表达式（逆波兰表达式）：

### 1、什么是后缀表达式？

后缀表达式，指的是不包含括号，运算符放在两个运算对象的后面，所有的计算按运算符出现的顺序，严格从左向右执行（不再考虑运算符的优先级）。

由于后缀表达式的运算符在两个操作数的后面，那么计算机在解析后缀表达式的时候，只需要从左向右扫描，也就是只需向前扫描，而不用回头扫描，遇到运算符就将运算符放在前面两个操作数的中间（这里先不考虑乘方类似的单目运算），一直运算到最右边的运算符，那么就得出运算结果了。

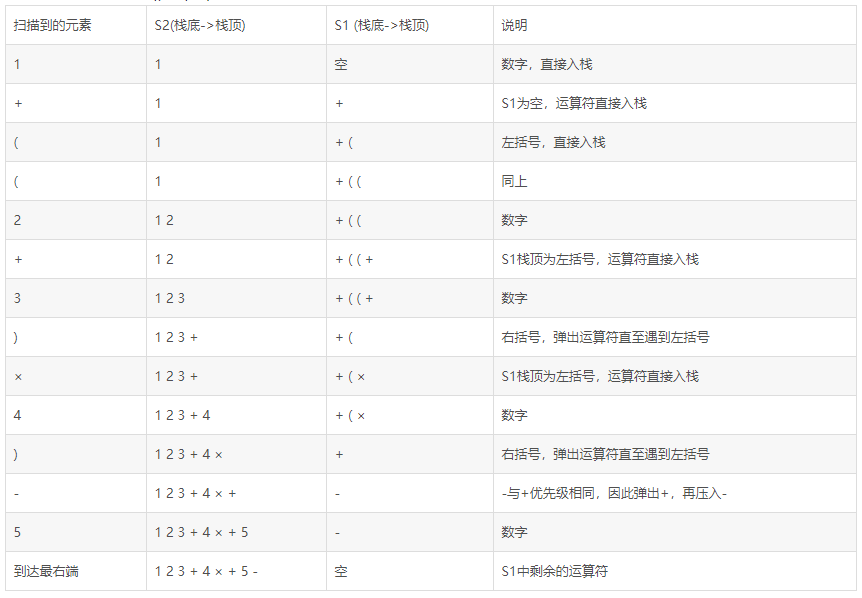
### 2、中缀表达式转换成后缀表达式（程序操作）：

遵循以下步骤：

|  |  |
| --- | --- |
| **中缀表达式转换为后缀表达式** | |
| 1）初始化两个栈：运算符栈S1和存储中间结果的栈S2； | |
| 2）从左至右扫描中缀表达式； | |
| 3）遇到操作数时，直接将其压入S2； | |
| 4）遇到运算符时，比较其与S1栈顶运算符的优先级： | |
|  | 4-1）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； |
| 4-2）否则，若优先级比栈顶运算符的较高，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，而这里是较高】 |
| 4-3）否则，将S1栈顶运算符弹出并压入到S2中，再次转到S1中新的栈顶运算符相比较； |
| 5）遇到括号时： | |
|  | 5-1）如果是左括号“(”，则直接压入S1； |
| 5-2）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； |
| 6）重复步骤（2）至（5），直到表达式的最左边； | |
| 7）将S1中剩余的运算符依次弹出并压入S2； | |
| 8）依次弹出S2中元素并输出，结果的逆序即为中缀表达式对应的后缀表达式； | |

### 3、示例：

将中缀表达式“1+（（2+3）\*4）-5”转换为后缀表达式的过程如下：



### 4、代码：

|  |
| --- |
|  |

【DataStructureAndAlgorithmDemo/src/dataStructure/ds\_04\_expression/exp\_2\_suffixExpression/SuffixExpression.java】

### 5、计算机怎么解析后缀表达式？

#### 1）步骤：

运算符在式中出现的顺序恰好为表达式的运算顺序；

每个运算符和在它之前出现且紧靠它的两个操作数构成一个最小表达式；

#### 2）示例：

例如后缀表达式“3 4 + 5 \* 6 -”

#### 3）代码：

|  |
| --- |
|  |