

Lecture 10: 栈在表达式求值中的应用

10.1 三种表达式

基本概念

- 前缀表达式 (Polish notation) : 波兰表达式
 - 规则: 运算符在两个操作数前面;
 - $+ab$
 - $-+abc$
 - $-+ab*cd$
- 后缀表达式 (Reverse Polish notation) : 逆波兰表达式;
 - 规则: 运算符在两个操作数后面;
 - $ab+$
 - $ab+c-$
 - 把 $a+b$ 的结果作为一个整体和 c 相减, 因此是 $ab+$ 在前面, 然后要减 c , 然后操作符往后放, 变成 $ab+c-$
 - $ab+cd*-$
 - 因为原本的是 $a+b-c*d$, 所以应该先计算 $*$ 的操作, 然后再 $-$, 所以是 $ab+cd*-$
- 中缀表达式: 我们平时数学运算时的运算规则;

中缀转后缀运算方法

- 确定中缀表达式中各个运算符的运算顺序;
- 选择下一个运算符, 按照**操作数右操作数运算符**的方式组合成一个新的操作数;
- 如果还有运算符没被处理, 就继续 (2);

- 图示:

$$((15 \div (7 - (1 + 1))) \times 3) - (2 + (1 + 1))$$

③ ② ① ④ ⑦ ⑥ ⑤

中缀表达式

$$15 \ 7 \ 1 \ 1 \ + \ - \ \div \ 3 \ \times \ 2 \ 1 \ 1 \ + \ + \ -$$

① ② ③ ④ ⑤ ⑥ ⑦

后缀表达式

- 过程
 - 从左往右扫描, 每遇到一个运算符, 就让运算符前面最近的两个操作数执行对应运算;
 - 图示:

$$((15 \div (7 - (1 + 1))) \times 3) - (2 + (1 + 1))$$

③
②
①
④
⑦
⑥
⑤

$$15 \quad 7 \quad 1 \quad 1 \quad + \quad - \quad \div \quad 3 \quad \times \quad 2 \quad 1 \quad 1 \quad + \quad + \quad -$$

①
②
③
④
⑤
⑥
⑦

• 减号左边

• 1 1 +

• 7 1 1 + -

• 15 7 1 1 + - /

• 减号右边

• 1 1 +

• 2 1 1 + +

• 2 1 1 + +

• 结果（去掉括号）： (15 7 1 1 + - /) (2 1 1 + +) -

• 注意：

- 1. 左优先原则：只要左边的运算符能计算，就优先计算左边的；
- 2. 运算符生效的顺序和后缀表达式相同（左优先原则）；

补充：中缀转前缀的手算方法

• 总结：

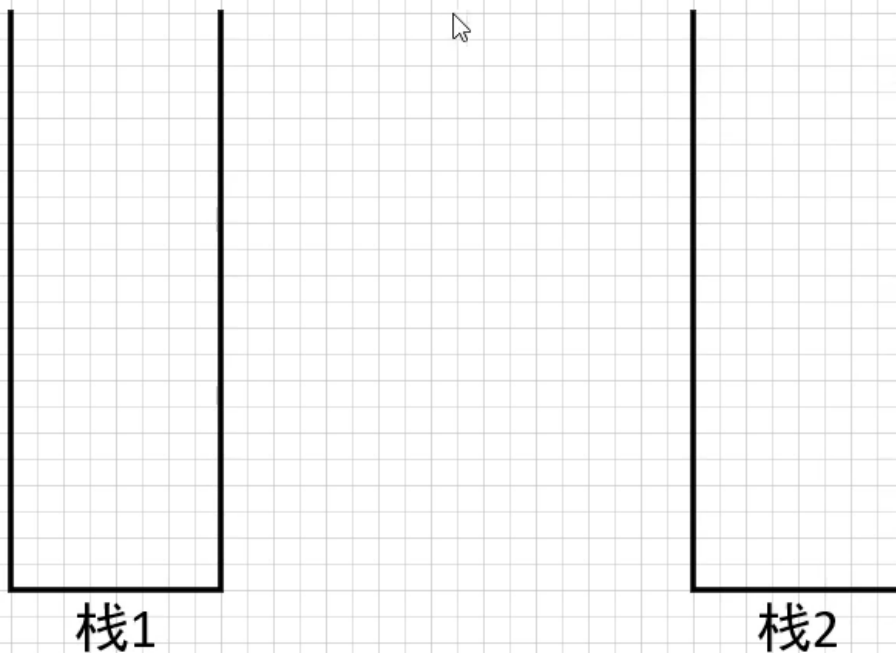
- 1. 确定中缀表达式中的各个运算符的运算顺序；
- 2. 选择下一个运算符，按照**运算符|左操作数|右操作数**的方式组合成一个新的操作数；
- 3. 如果还有运算符没被处理，就继续 (2)；

• 注意：

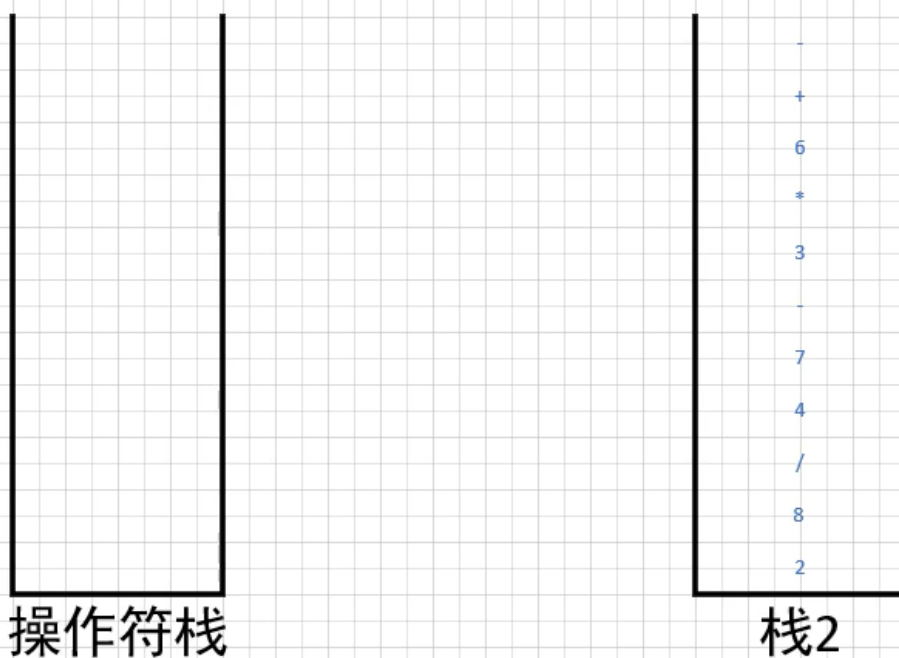
- 这里采用右优先原则；

• 图示：

中缀表达式: $6+3*(7-4)-8/2$
转为前缀表达式。



中缀表达式: $6+3*(7-4)-8/2$
转为前缀表达式。



10.2 后缀表达式算法

用栈实现后缀表达式的计算

1. 从左往右扫描下一个元素，直到处理完所有元素
 2. 若扫描到**操作数则压入栈**，并回到 (1)，否则执行 (3)
 3. 若扫描到**运算符**，则**弹出**两个栈顶元素，**执行相应运算**，运算结果压回栈顶，回到 (1)
- 注意：先出栈的是右操作数。若表达式合法，则最后栈中只会留下一个元素，就是最终结果；

10.3 中缀表达式算法

用栈实现中缀表达式的计算

- 初始化 2 个栈，**操作数栈**和**运算符栈**
1. 若扫描到 **操作数**，压入操作数栈
 2. 若扫描到 **运算符** 和 **界限符**，则按照**中缀转后缀**，相同的逻辑压入运算符栈（期间也会弹出运算符，每当弹出一个运算符时，就需要再弹出两个操作数栈的栈顶元素并执行相应运算，运算结果再压回操作数）

10.3.1 中缀表达式转后缀表达式

方法

- 初始化一个栈，用保存暂时还不能确定运算顺序的运算符。
 - 从左到右处理各个元素，直到末尾。可能遇到 3 种情况：
1. 遇到 **操作数**。直接加入后缀表达式。
 2. 遇到 **界限符**。遇到 **(** 直接入栈，遇到 **)** 则依次弹出栈内运算符并加入后缀表达式，直到弹出“**(**”为止
 3. 遇到 **运算符**。依次弹出栈中优先级高于或等于运算符的所有运算符，并**加入后缀表达式**，若碰到 **(** 或栈空则停止。之后再把当前运算符入站。
- 按上述方法处理完所有字符后，将栈中剩余运算符依次弹出，并加入后缀表达式；
 - 图示：

$$((15 \div (7 - (1 + 1))) \times 3) - (2 + (1 + 1))$$

③ ② ① ④ ⑦ ⑥ ⑤

$$15 \ 7 \ 1 \ 1 \ + \ - \ \div \ 3 \ \times \ 2 \ 1 \ 1 \ + \ + \ -$$

① ② ③ ④ ⑤ ⑥ ⑦

中缀表达式的计算：先中缀转后缀，后进行后缀表达式求值；

用栈实现中缀表达式转后缀表达式：

初始化一个栈，用于保存暂时还不能确定运算顺序的运算符。

从左到右处理各个元素，直到末尾。可能遇到三种情况：

- ① 遇到**操作数**。直接加入后缀表达式。
- ② 遇到**界限符**。遇到“(”直接入栈；遇到“)”则依次弹出栈内运算符并加入后缀表达式，直到弹出“(”为止。注意：“(”不加入后缀表达式。
- ③ 遇到**运算符**。依次弹出栈中优先级高于或等于当前运算符的所有运算符，并加入后缀表达式，若碰到“(”或栈空则停止。之后再把当前运算符入栈。

按上述方法处理完所有字符后，将栈中剩余运算符依次弹出，并加入后缀表达式。

用栈实现后缀表达式的计算：

① 从左往右扫描下一个元素，直到处理完所有元素

② 若扫描到操作数则压入栈，并回到①；否则执行③

③ 若扫描到运算符，则弹出两个栈顶元素，执行相应运算，运算结果压回栈顶，回到①

用栈实现中缀表达式的计算：

初始化两个栈，**操作数栈**和**运算符栈**

若扫描到操作数，压入操作数栈

若扫描到运算符或界限符，则按照“中缀转后缀”相同的逻辑压入运算符栈（期间也会弹出运算符，每当弹出一个运算符时，就需要再弹出两个操作数栈的栈顶元素并执行相应运算，运算结果再压回操作数栈）

获取笔记持续更新可加微信: Wd...