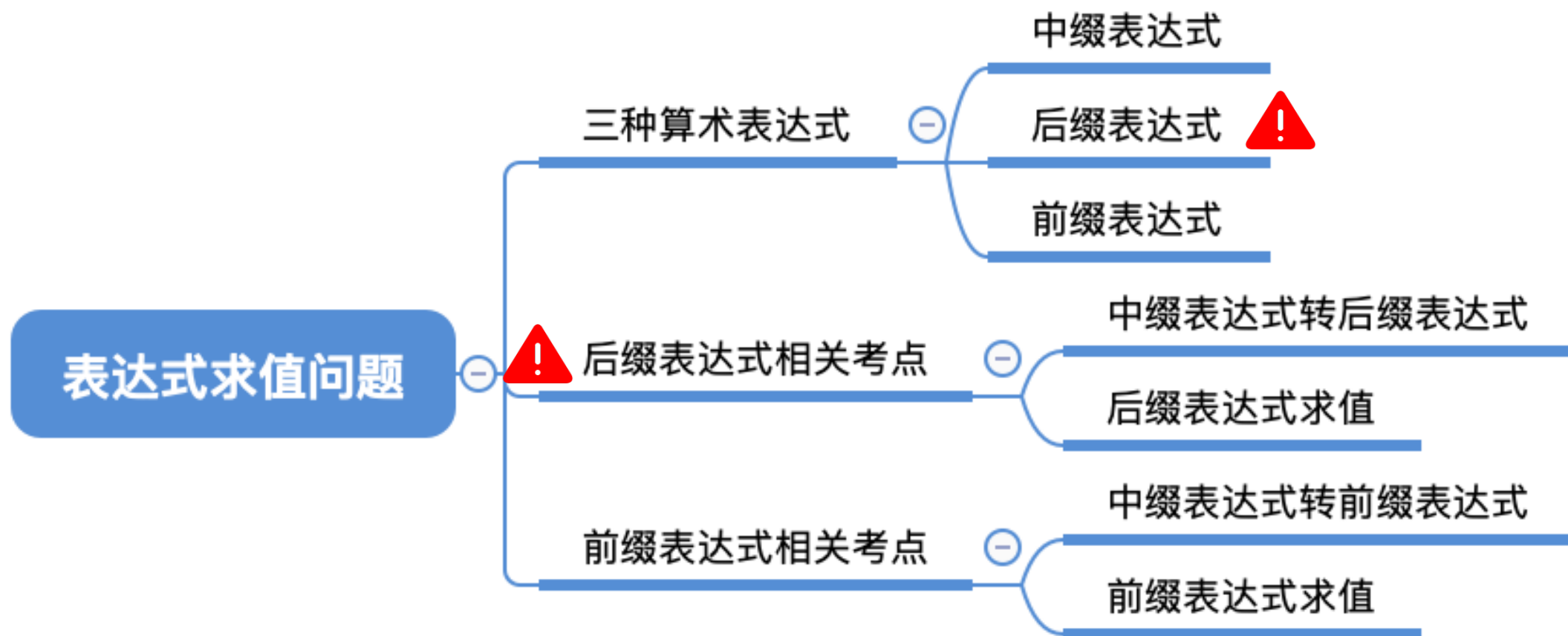


本节内容

栈的应用

——表达式求值

知识总览



大家熟悉的算数表达式



Reference: Wikipedia
——Reverse Polish notation

$$\begin{array}{ccccccc} ((15 \div (7 - (1 + 1))) \times 3) - (2 + (1 + 1)) \\ \textcircled{3} \quad \textcircled{2} \quad \textcircled{1} \quad \textcircled{4} \quad \textcircled{7} \quad \textcircled{6} \quad \textcircled{5} \end{array}$$

$$\begin{array}{ccccccc} 15 \div 7 - 1 + 1 \times 3 - 2 + 1 + 1 \\ \textcircled{1} \quad \textcircled{2} \quad \textcircled{4} \quad \textcircled{3} \quad \textcircled{5} \quad \textcircled{6} \quad \textcircled{7} \end{array}$$

由三个部分组成：操作数、运算符、界限符

界限符是必不可少的，
反映了计算的先后顺序

波兰数学家的灵感

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



一个灵感：可以不用界限符也能无歧义地表达运算顺序

Reverse Polish notation（逆波兰表达式 = 后缀表达式）

Polish notation（波兰表达式 = 前缀表达式）

中缀、后缀、前缀表达式

运算符在两个
操作数中间

中缀表达式

$a + b$

$a + b - c$

$a + b - c * d$

规则：运算符在
两个操作数后面

后缀表达式

$a b +$

$a b + c -$

$a b + c d * -$

规则：运算符在
两个操作数前面

前缀表达式

$+ a b$

$- + a b c$

$- + a b * c d$

中缀表达式转后缀表达式（手算）

中缀转后缀的手算方法：

- ① 确定中缀表达式中各个运算符的运算顺序
- ② 选择下一个运算符，按照「左操作数 右操作数 运算符」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

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

③ ② ① ④ ⑦ ⑥ ⑤

中缀表
达式

15 7 1 1 ^①+ ^②- ^③÷ 3 ^④× 2 1 1 ^⑤+ ^⑥+ ^⑦-

后缀表
达式

中缀表达式转后缀表达式（手算）

中缀转后缀的手算方法：

- ① 确定中缀表达式中各个运算符的运算顺序
- ② 选择下一个运算符，按照「左操作数 右操作数 运算符」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

运算顺序不唯一，因此对应的
后缀表达式也不唯一

$A + B * (C - D) - E / F$

③ ② ① ⑤ ④



① ② ③ ④ ⑤

$A B C D - * + E F / -$

$A + B * (C - D) - E / F$

⑤ ③ ② ④ ①



② ③ ① ④ ⑤

$A B C D - * E F / - +$

私房菜：“左优先”原则，不要FreeStyle，保证手算和机算结果相同

“左优先”原则：只要左边的运算符能先计算，就优先算左边的

客观来看两种都正确，只是“机算”结果是前者

中缀表达式转后缀表达式（手算）

中缀转后缀的手算方法：

- ① 确定中缀表达式中各个运算符的运算顺序
- ② 选择下一个运算符，按照「左操作数 右操作数 运算符」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

运算顺序不唯一，因此对应的
后缀表达式也不唯一

“左优先”原则：只要左边的运算符能先计算，就优先算左边的

可保证运算顺序唯一

$A + B - C * D / E + F$

① ④ ② ③ ⑤

$A B + C D * E / - F +$

① ② ③ ④ ⑤

后缀表达式的计算（手算）

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

③

②

①

④

⑦

⑥

⑤

中缀表
达式

$$15 \quad 7 \quad 1 \quad 1 \quad \overset{\textcircled{1}}{+} \quad \overset{\textcircled{2}}{-} \quad \overset{\textcircled{3}}{\div} \quad 3 \quad \overset{\textcircled{4}}{\times} \quad 2 \quad 1 \quad 1 \quad \overset{\textcircled{5}}{+} \quad \overset{\textcircled{6}}{+} \quad \overset{\textcircled{7}}{-}$$

后缀表
达式

后缀表达式的手算方法：

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，合体为一个操作数

注意：两个操作数的左右顺序

后缀表达式的计算（手算）

后缀表达式的手算方法：


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，合体为一个操作数

注意：两个操作数的左右顺序

$$A + B * (C - D) - E / F$$

③ ② ① ⑤ ④

① ② ③ ④ ⑤

$$A \ B \ C \ D \ - \ * \ + \ E \ F \ / \ -$$


后缀表达式的计算（手算）

后缀表达式的手算方法：

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，合体为一个操作数

注意：两个操作数的左右顺序

$A + B - C * D / E + F$
① ④ ② ③ ⑤

① ② ③ ④ ⑤
 $A B + C D * E / - F +$
└──────────┘
└──────────┘
└──────────┘
└──────────┘



认真思考

特点：最后出现的操作数先被运算



耶！

LIFO（后进先出）

栈！！

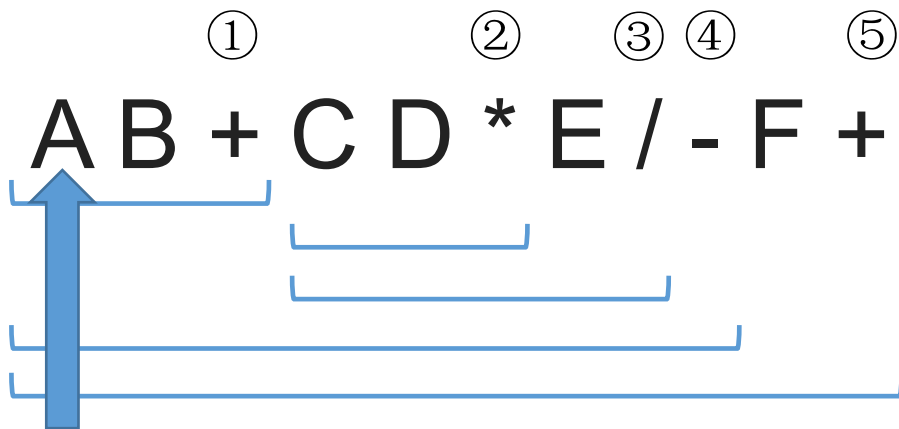
后缀表达式的计算（机算）

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

$A + B - C * D / E + F$

① ④ ② ③ ⑤



栈



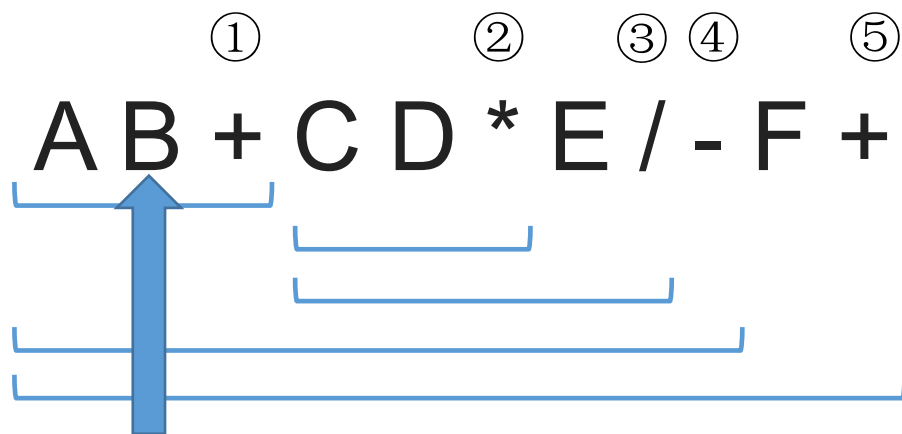
后缀表达式的计算（机算）

用栈实现后缀表达式的计算：

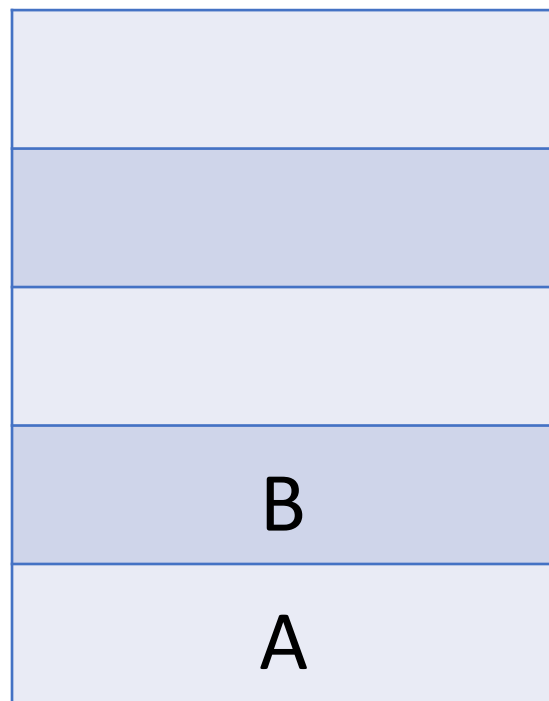
- ①从左往右扫描下一个元素，直到处理完所有元素
- ②若扫描到操作数则压入栈，并回到①；否则执行③
- ③若扫描到运算符，则弹出两个栈顶元素，执行相应运算，运算结果压回栈顶，回到①

$A + B - C * D / E + F$

① ④ ② ③ ⑤



栈



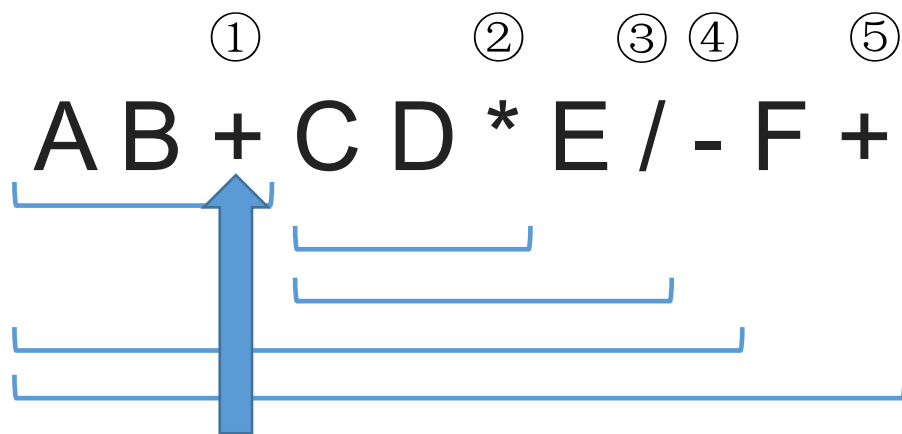
后缀表达式的计算（机算）

用栈实现后缀表达式的计算：

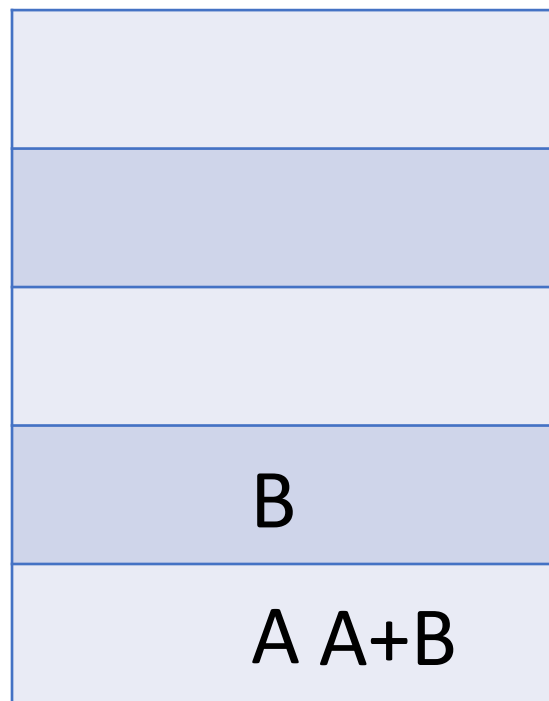
- ①从左往右扫描下一个元素，直到处理完所有元素
- ②若扫描到操作数则压入栈，并回到①；否则执行③
- ③若扫描到运算符，则弹出两个栈顶元素，执行相应运算，运算结果压回栈顶，回到①

$A + B - C * D / E + F$

① ④ ② ③ ⑤



栈



注意：先出栈的是“右操作数”

+

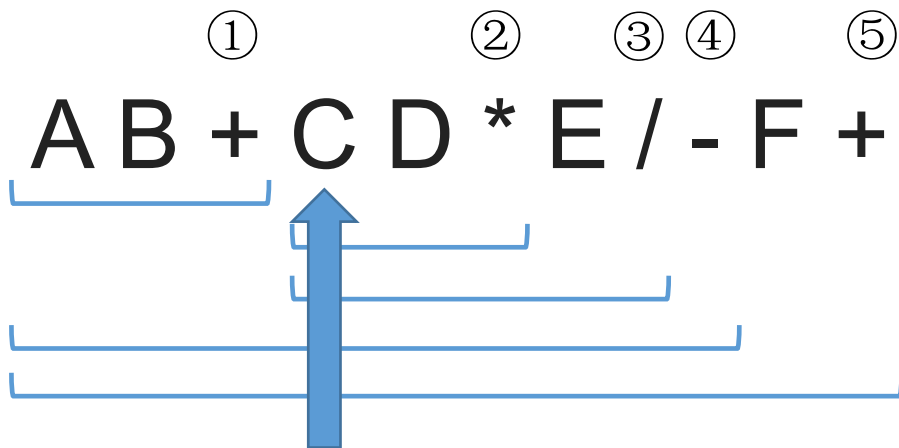
后缀表达式的计算（机算）

用栈实现后缀表达式的计算：

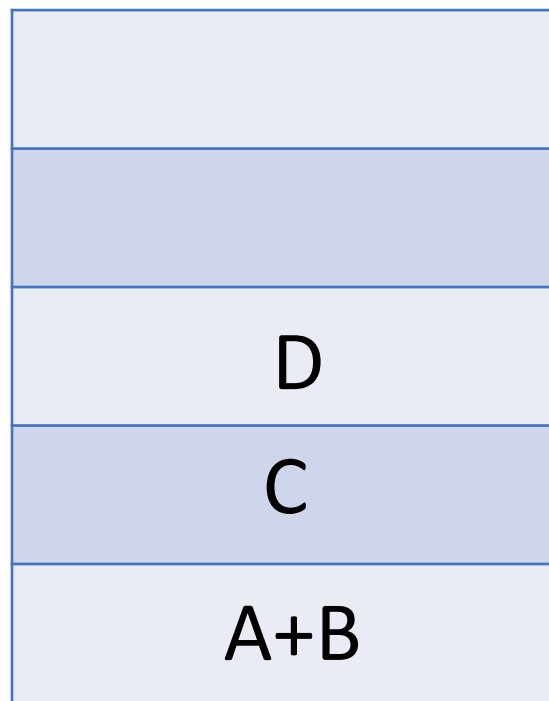
- ①从左往右扫描下一个元素，直到处理完所有元素
- ②若扫描到操作数则压入栈，并回到①；否则执行③
- ③若扫描到运算符，则弹出两个栈顶元素，执行相应运算，运算结果压回栈顶，回到①

$A + B - C * D / E + F$

① ④ ② ③ ⑤



栈



$C * D$

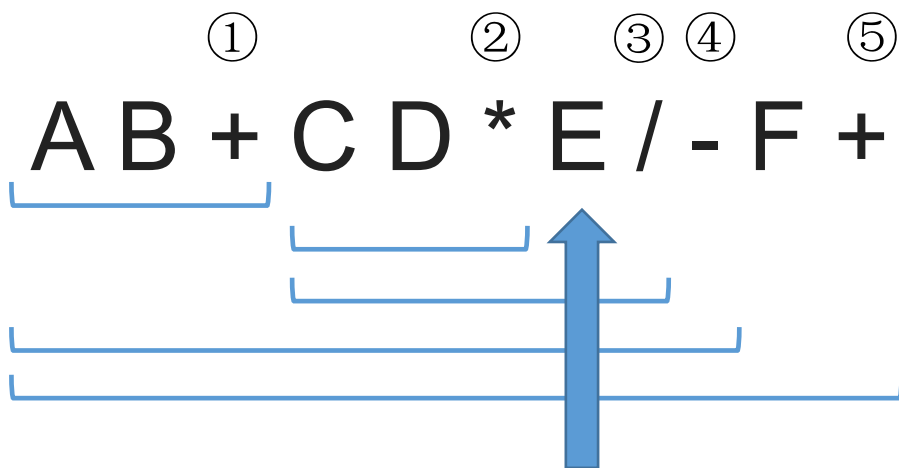
后缀表达式的计算（机算）

用栈实现后缀表达式的计算：

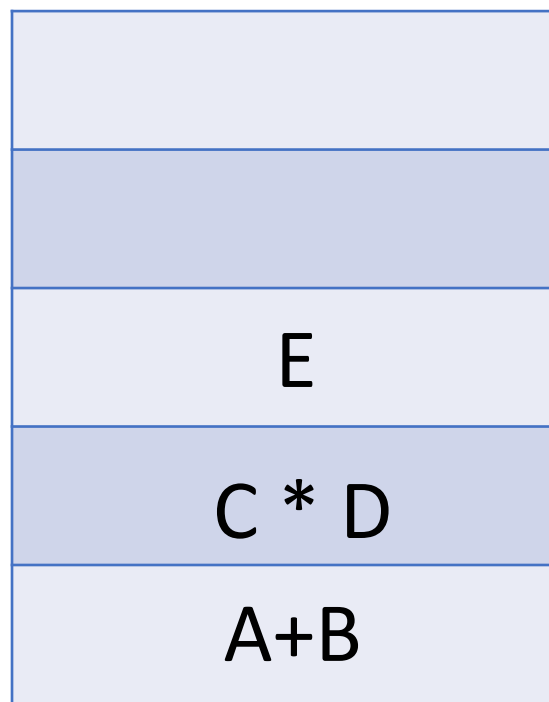
- ①从左往右扫描下一个元素，直到处理完所有元素
- ②若扫描到操作数则压入栈，并回到①；否则执行③
- ③若扫描到运算符，则弹出两个栈顶元素，执行相应运算，运算结果压回栈顶，回到①

$A + B - C * D / E + F$

① ④ ② ③ ⑤



栈



$(C * D) / E$

后缀表达式的计算（机算）

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

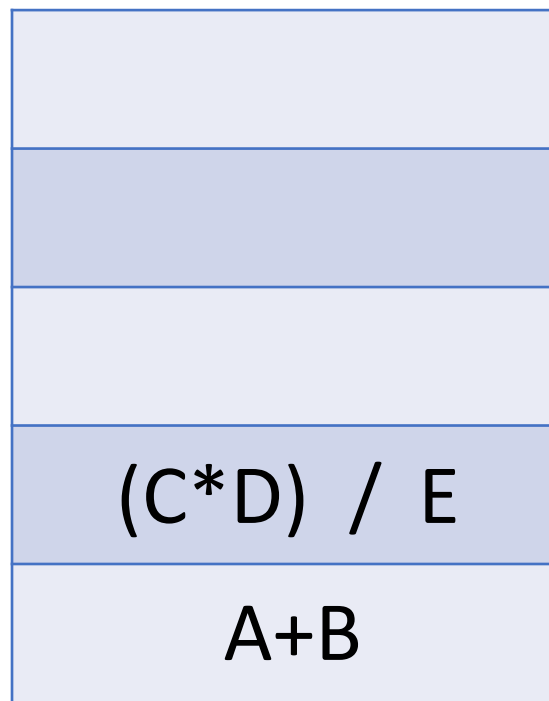
$A + B - C * D / E + F$

① ④ ② ③ ⑤

① ② ③ ④ ⑤
 $A B + C D * E / - F +$



栈

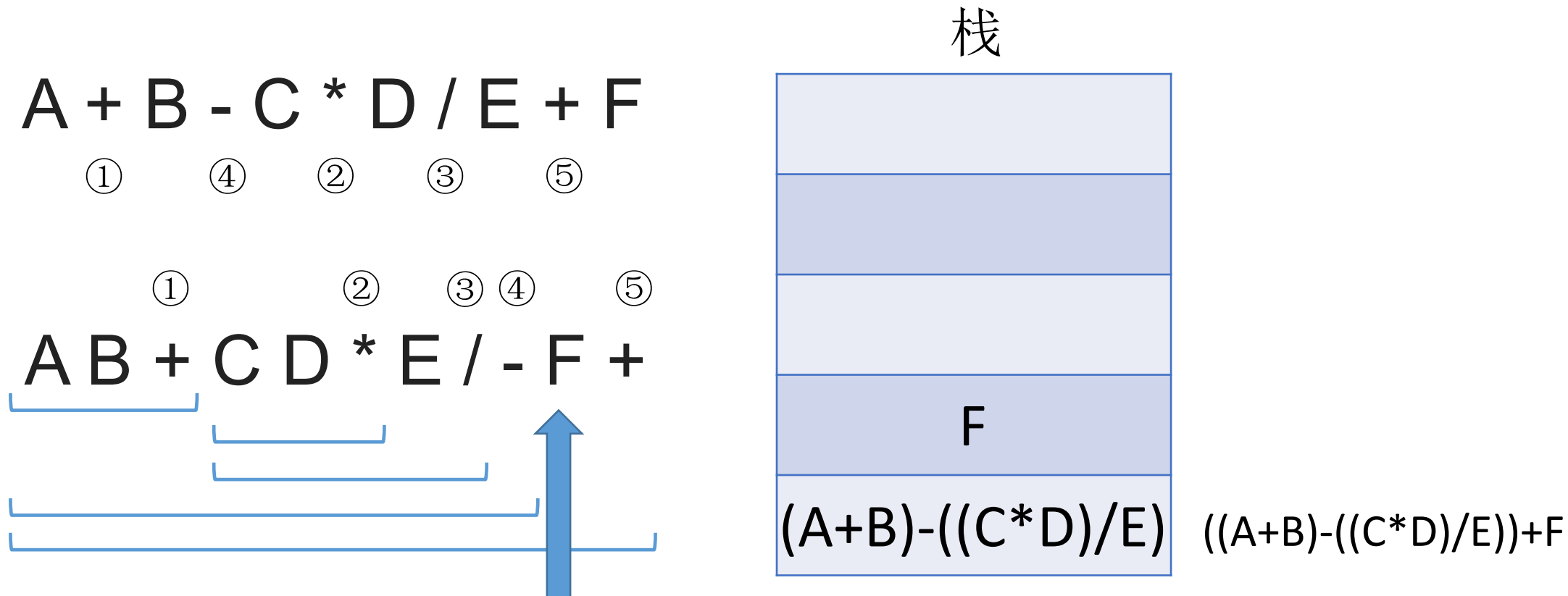


$(A+B)-((C*D)/E)$

后缀表达式的计算（机算）

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



后缀表达式的计算（机算）

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”

$A + B - C * D / E + F$

① ④ ② ③ ⑤

① ② ③ ④ ⑤
 $A B + C D * E / - F +$



栈



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

后缀表达式的计算（机算）

后缀表达式适用于基于栈的编程语言（**stack-oriented programming language**），如：Forth、PostScript

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”

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

③ ② ① ④ ⑦ ⑥ ⑤

15 7 1 1 + - ÷ 3 × 2 1 1 + + -

栈



思考：后缀表达式怎么转中缀？

放弃思考



中缀表达式转前缀表达式（手算）

中缀转前缀的手算方法：

- ① 确定中缀表达式中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

“右优先”原则：只要右边的运算符能先计算，就优先算右边的

$$\begin{array}{ccccccccc} A & + & B & * & (C & - & D) & - & E / F \\ \textcircled{3} & & \textcircled{2} & & \textcircled{1} & & \textcircled{5} & & \textcircled{4} \end{array}$$

$$\begin{array}{ccccccccc} A & + & B & * & (C & - & D) & - & E / F \\ & & \textcircled{5} & & \textcircled{3} & & \textcircled{2} & & \textcircled{4} & & \textcircled{1} \end{array}$$

$$\begin{array}{ccccccccc} \textcircled{5} & \textcircled{3} & & \textcircled{2} & & \textcircled{1} & & & \textcircled{4} \\ - & + & A & * & B & - & C & D & / E F \end{array}$$

$$\begin{array}{ccccccccc} \textcircled{5} & & \textcircled{4} & \textcircled{3} & & \textcircled{2} & & & \textcircled{1} \\ + & A & - & * & B & - & C & D & / E F \end{array}$$

中缀表达式转前缀表达式（手算）

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

③ ② ① ④ ⑦ ⑥ ⑤

中缀转后缀：
“左优先”

15 7 1 1 + - ÷ 3 × 2 1 1 + + -

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

⑤ ④ ③ ⑥ ⑦ ② ①

中缀转前缀：
“右优先”

- × ÷ 15 - 7 + 1 1 3 + 2 + 1 1

⑦ ⑥ ⑤ ④ ③ ② ①

前缀表达式的计算

用栈实现前缀表达式的计算：

① 从右往左扫描下一个元素，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“左操作数”

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

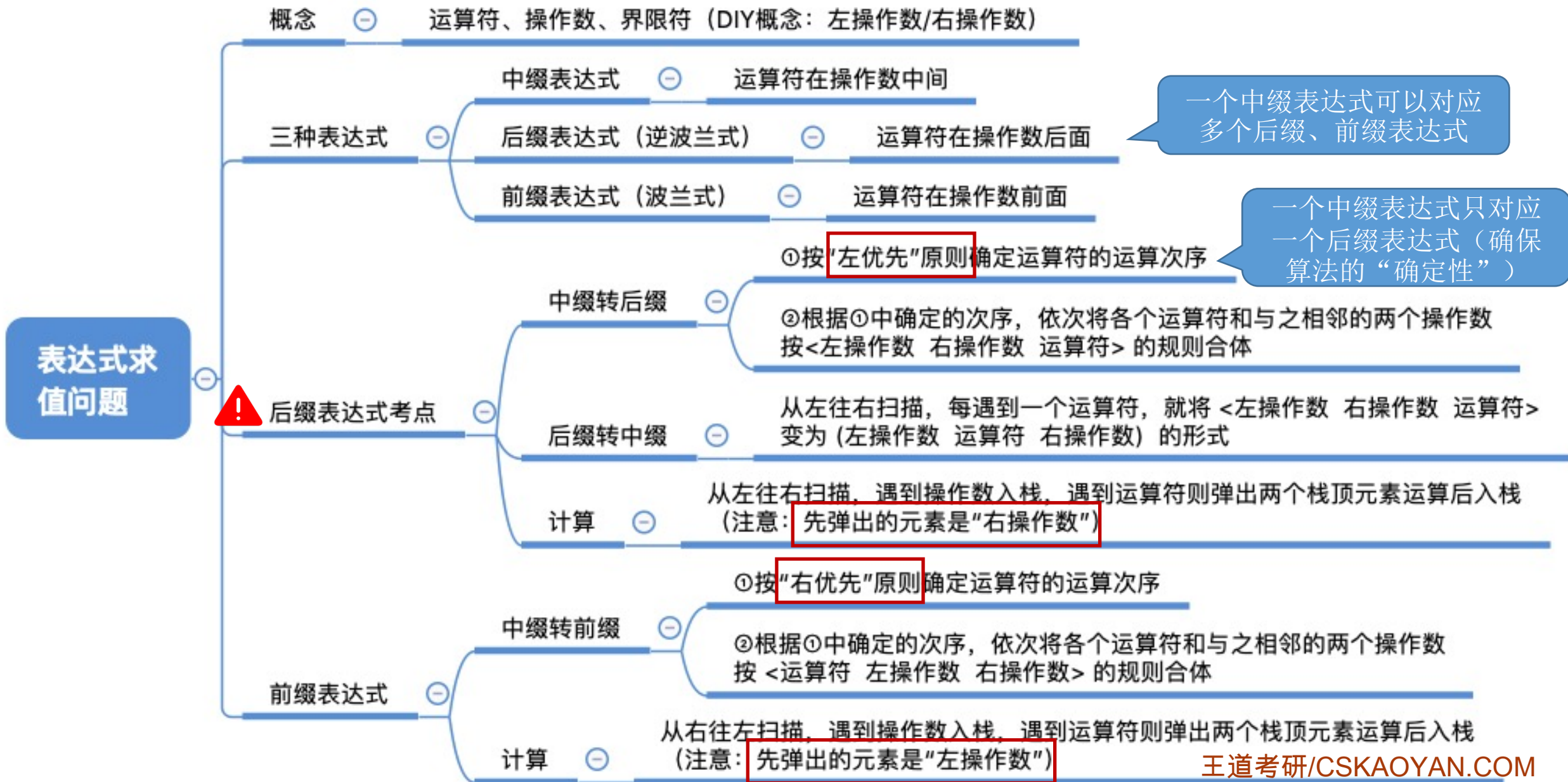
⑤ ④ ③ ⑥ ⑦ ② ①

- × ÷ 15 - 7 + 1 1 3 + 2 + 1 1

⑦ ⑥ ⑤ ④ ③ ② ①



知识回顾与重要考点



欢迎大家对本节视频进行评价~



学员评分：3.3.2_1 栈...

扫一扫二维码打开或分享给好友



- 腾讯文档 -

可多人实时在线编辑，权限安全可控



公众号：王道在线



b站：王道计算机教育



抖音：王道计算机考研