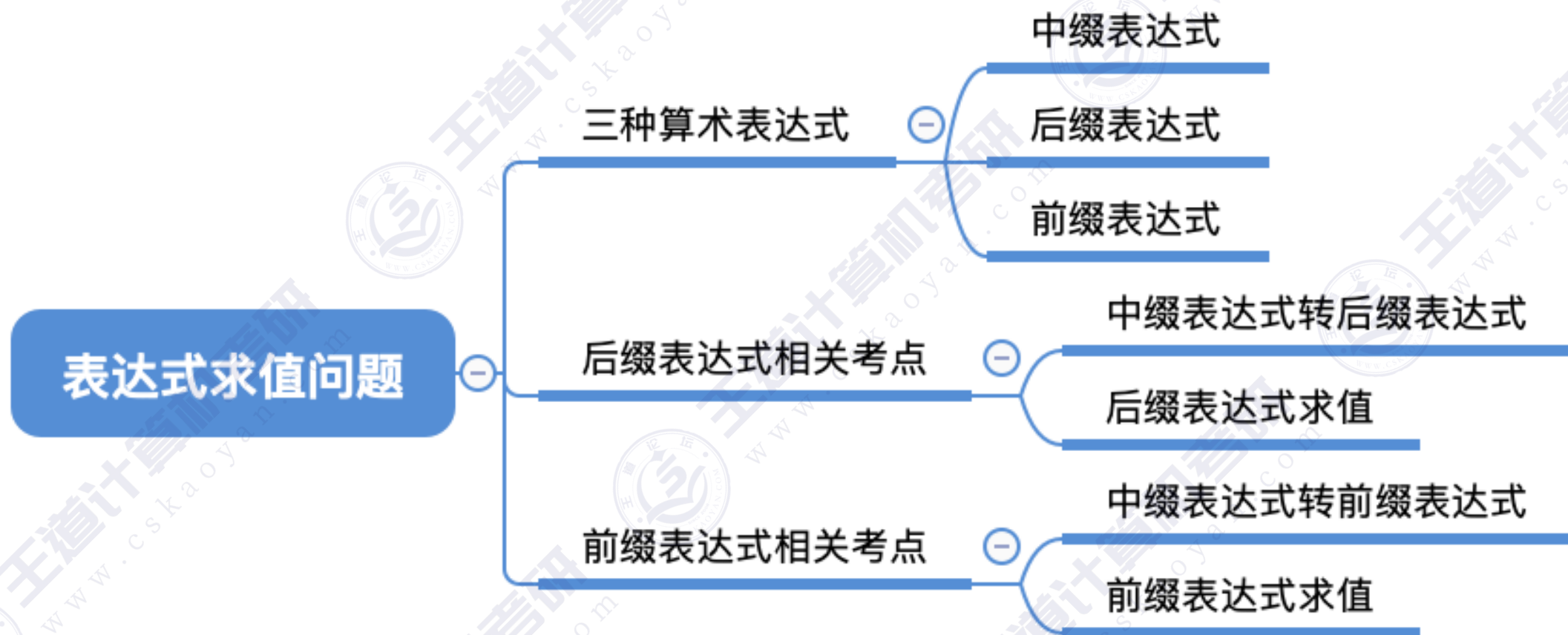


本节内容

# 栈的应用

——表达式求值

# 知识总览



# 知识总览

## 表达式求值问题（第二季）

中缀表达式转后缀表达式（机算，用栈实现）

中缀表达式的计算（用栈实现）

## 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

“左优先”原则：只要左边的运算符能先计算，就优先算左边的

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

①

④

②

③

⑤

$AB + CD * E / - F +$

①

②

③

④

⑤

## 中缀表达式转后缀表达式（机算）

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。

**\* / 优先级高于 + -**

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。

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

①

④

②

③

⑤

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

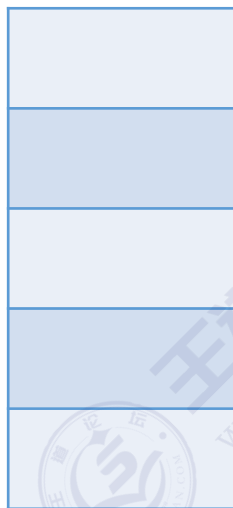
①

②

③

④

⑤



A

## 中缀表达式转后缀表达式（机算）

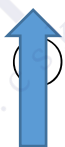
初始化一个栈，用于保存暂时还不能确定运算顺序的运算符。

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。

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



④

②

③

⑤

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

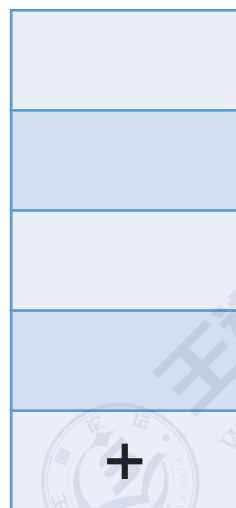
①

②

③

④

⑤



A

## 中缀表达式转后缀表达式（机算）

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。

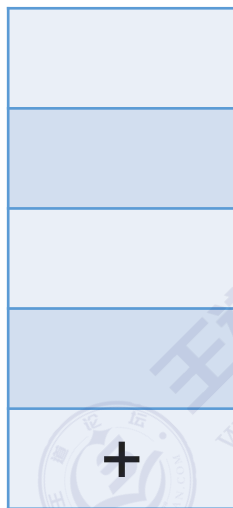
按上述方法处理完所有字符后，将栈中剩余运算符依次弹出，并加入后缀表达式。

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

① ↑ ④ ② ③ ⑤

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

① ② ③ ④ ⑤



A B

## 中缀表达式转后缀表达式（机算）

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。

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

①



②

③

⑤

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

①

②

③

④

⑤



A B

对比：“左优先”原则

## 中缀表达式转后缀表达式（机算）

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。

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

①

④

②

③

⑤

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

①

②

③

④

⑤



A B + C

## 中缀表达式转后缀表达式（机算）

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。

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

①

④



③

⑤

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

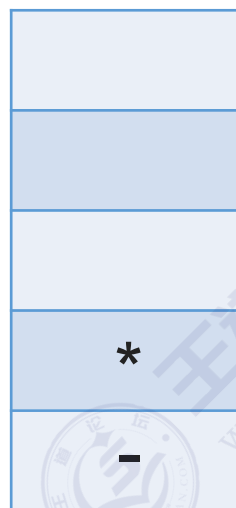
①

②

③

④

⑤



A B + C

## 中缀表达式转后缀表达式（机算）

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。

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

①

④

②

③

⑤



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

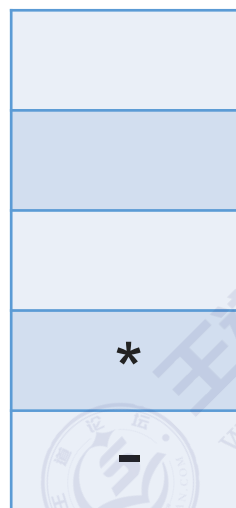
①

②

③

④

⑤



A B + C D

## 中缀表达式转后缀表达式（机算）

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。

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

①

④

②



⑤

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

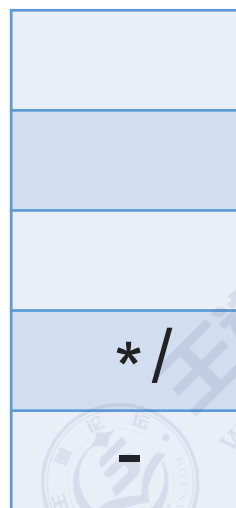
①

②

③

④

⑤



A B + C D

## 中缀表达式转后缀表达式（机算）

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。

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

①

④

②

③

⑤



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

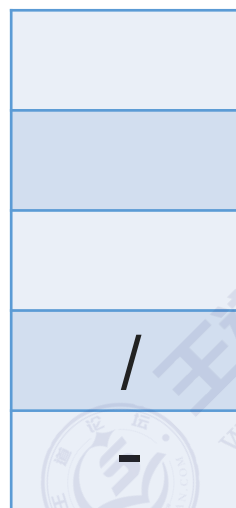
①

②

③

④

⑤



A B + C D \* E

## 中缀表达式转后缀表达式（机算）

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。

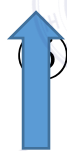
A + B - C \* D / E + F

①

④

②

③



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

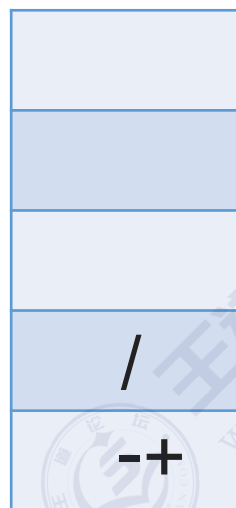
①

②

③

④

⑤



A B + C D \* E

## 中缀表达式转后缀表达式（机算）

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。

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

①

④

②

③

⑤



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

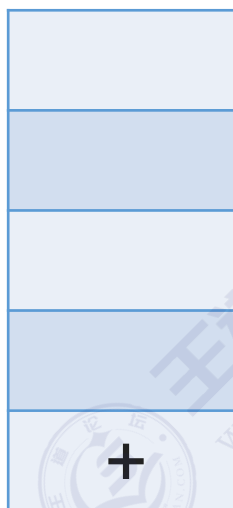
①

②

③

④

⑤



A B + C D \* E / - F

## 中缀表达式转后缀表达式（机算）

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。

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

①

④

②

③

⑤



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

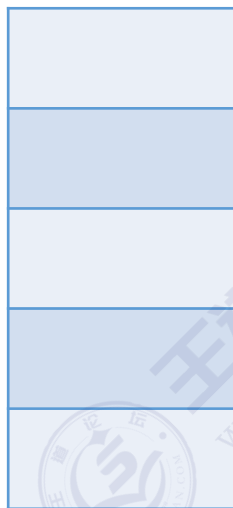
①

②

③

④

⑤



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

## 中缀表达式转后缀表达式（机算）

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。

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

③      ②      ①      ⑤      ④

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



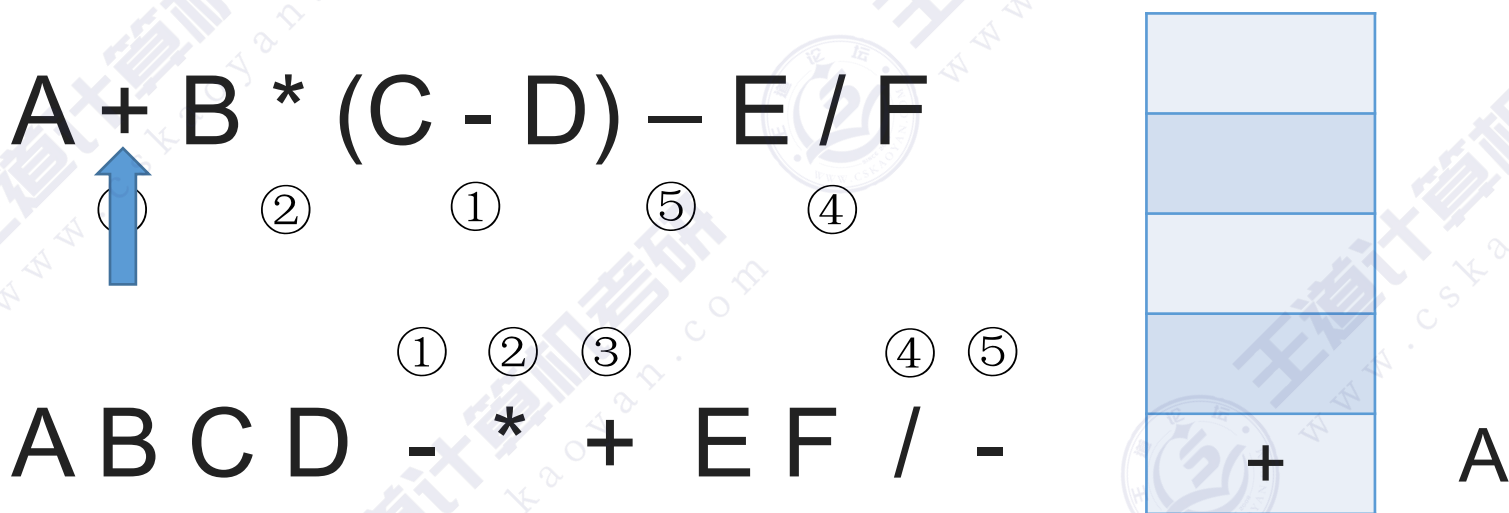
## 中缀表达式转后缀表达式（机算）

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。



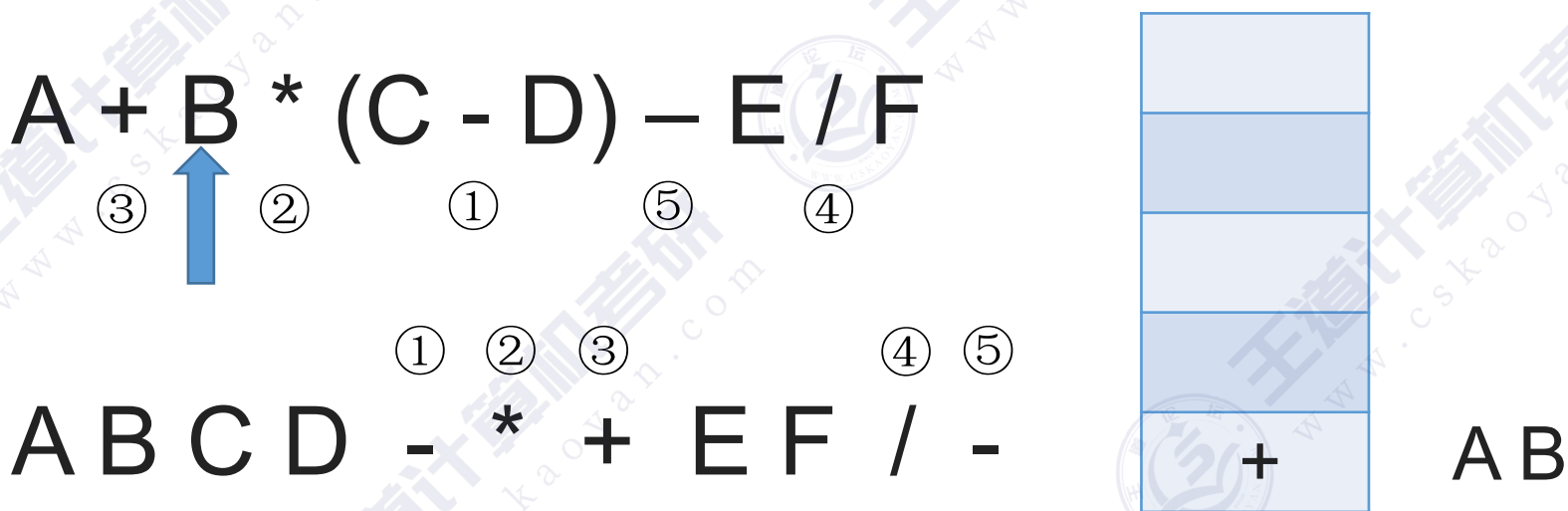
## 中缀表达式转后缀表达式（机算）

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。



## 中缀表达式转后缀表达式（机算）

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。

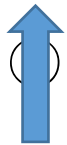
从左到右处理各个元素，直到末尾。可能遇到三种情况：

- 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。

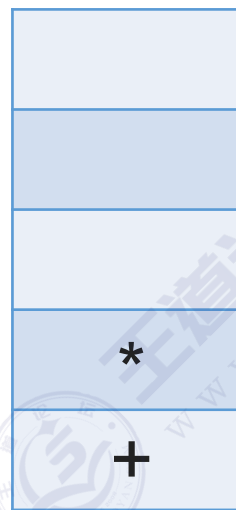
A + B \* (C - D) - E / F

③      ①      ⑤      ④



①   ②   ③                      ④   ⑤

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



A B

## 中缀表达式转后缀表达式（机算）

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。

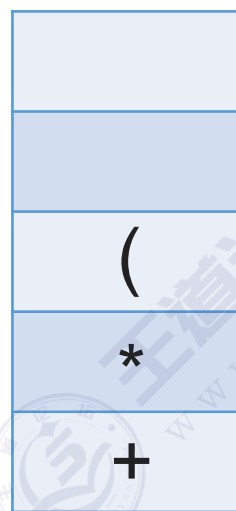
$A + B * (C - D) - E / F$

③      ②      ①      ⑤      ④



①   ②   ③                      ④   ⑤

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



$A B$

## 中缀表达式转后缀表达式（机算）

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。

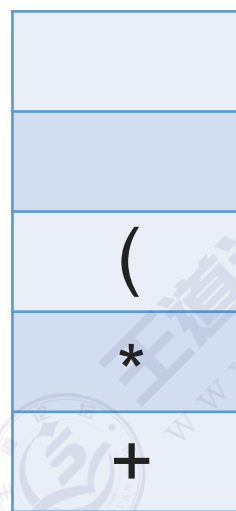
按上述方法处理完所有字符后，将栈中剩余运算符依次弹出，并加入后缀表达式。

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

③      ②      ①      ⑤      ④

①   ②   ③                      ④   ⑤

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



$A B C$

## 中缀表达式转后缀表达式（机算）

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。

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

③

②



⑤

④

①

②

③

④

⑤

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



A B C

## 中缀表达式转后缀表达式（机算）

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。

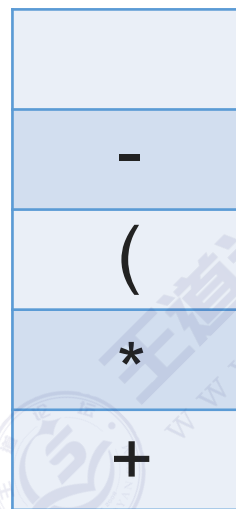
按上述方法处理完所有字符后，将栈中剩余运算符依次弹出，并加入后缀表达式。

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

③      ②      ①      ⑤      ④

①   ②   ③                      ④   ⑤

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



A B C D

## 中缀表达式转后缀表达式（机算）

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。

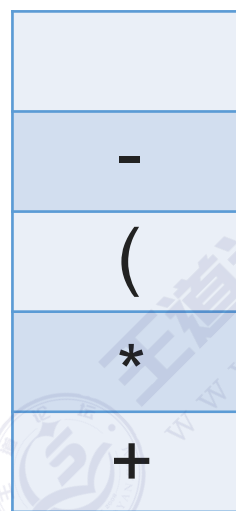
按上述方法处理完所有字符后，将栈中剩余运算符依次弹出，并加入后缀表达式。

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

③      ②      ①      ⑤      ④

①   ②   ③      ④   ⑤

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



A B C D

## 中缀表达式转后缀表达式（机算）

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。

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

③

②

①

④

⑤

①

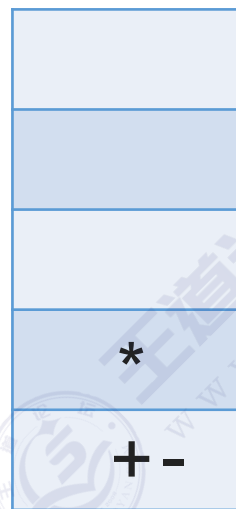
②

③

④

⑤

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



A B C D -

## 中缀表达式转后缀表达式（机算）

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。

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

③      ②      ①      ⑤      ④



①   ②   ③      ④   ⑤

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



$A B C D - * + E$

## 中缀表达式转后缀表达式（机算）

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。

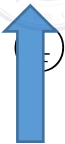
A + B \* (C - D) - E / F

③

②

①

⑤



①

②

③

④

⑤

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



A B C D - \* + E

## 中缀表达式转后缀表达式（机算）

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。

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

③

②

①

⑤

④



①

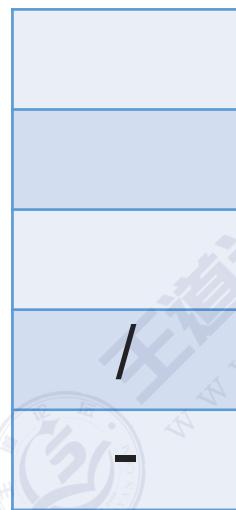
②

③

④

⑤

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



A B C D - \* + E F

## 中缀表达式转后缀表达式（机算）

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。

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

③      ②      ①      ⑤      ④

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



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

## 中缀表达式转后缀表达式（机算）

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。

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

③

②

①

④

⑦

⑥

⑤

15 7 1 1 + - ÷ 3 × 2 1 1 + + -

①

②

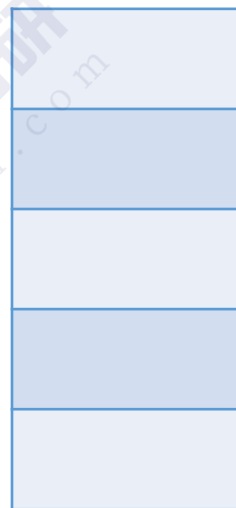
③

④

⑤

⑥

⑦



栈是否会  
溢出？

## 中缀表达式的计算（用栈实现）

中缀转后缀  
+  
后缀表达式求值  
两个算法的结合

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）

## 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



## 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

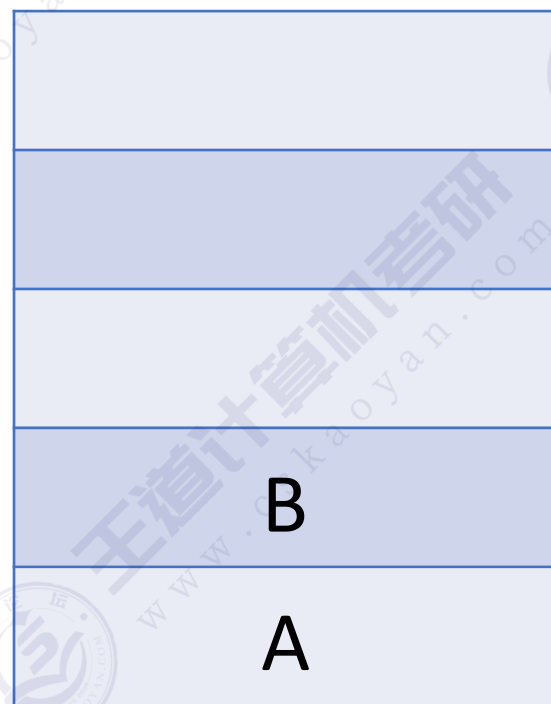
$A + B - C * D / E + F$

①      ④      ②      ③      ⑤

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



栈



## 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

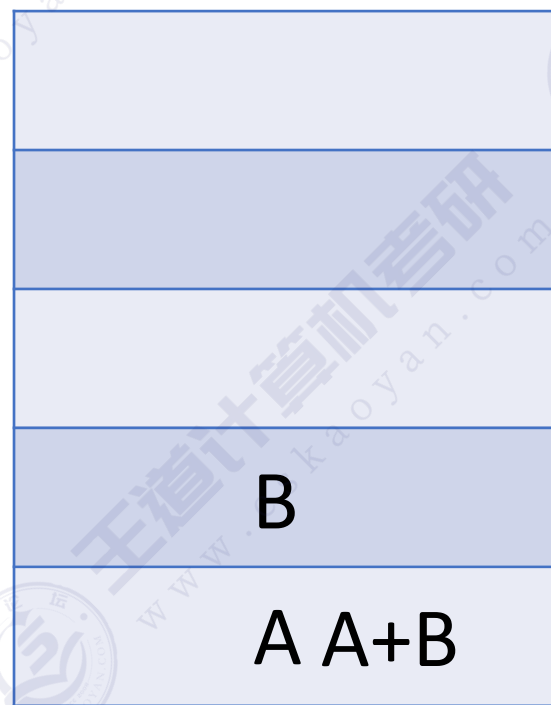
$A + B - C * D / E + F$

①      ④      ②      ③      ⑤

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



栈



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

①

②

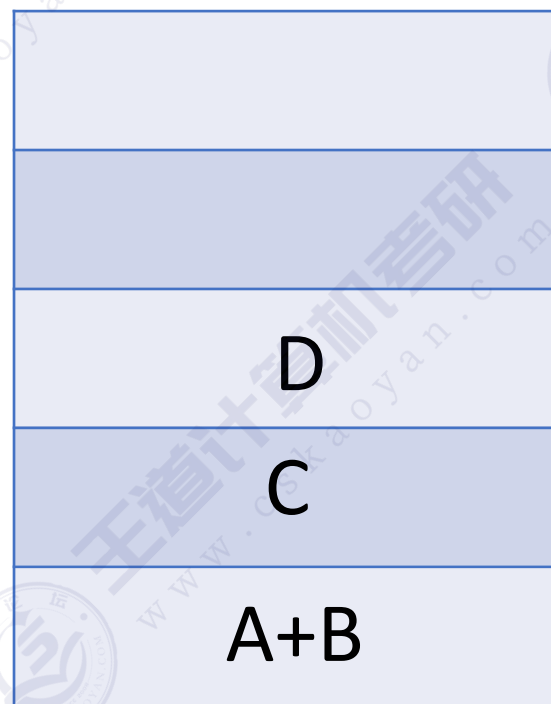
③

④

⑤

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

栈



## 后缀表达式的计算（机算）

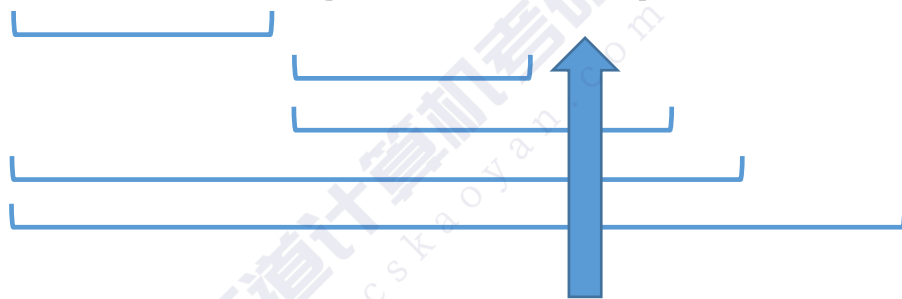
用栈实现后缀表达式的计算：

- 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

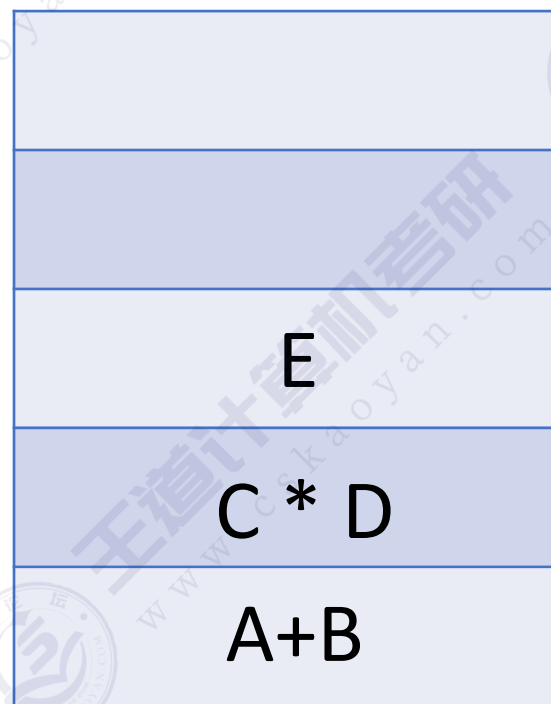
$A + B - C * D / E + F$

①      ④      ②      ③      ⑤

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



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

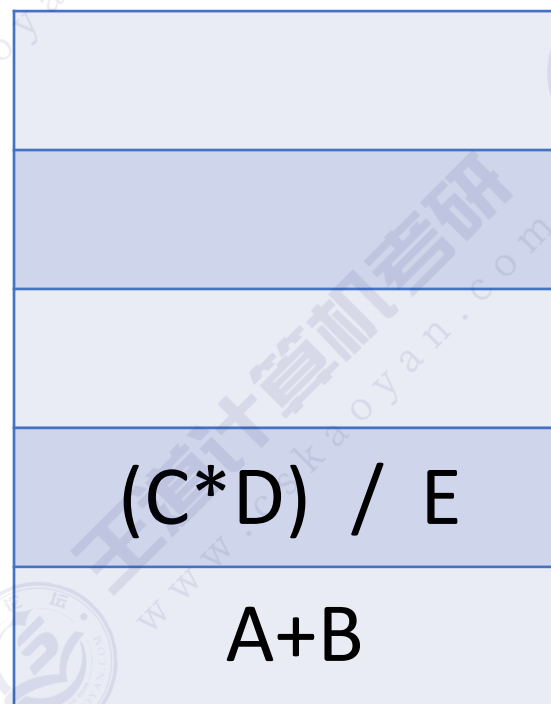
$A + B - C * D / E + F$

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

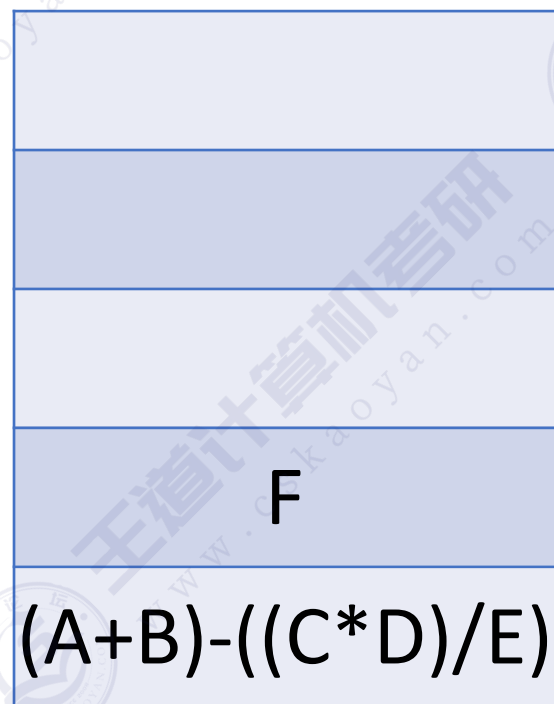
A + B - C \* D / E + F

①      ④      ②      ③      ⑤

①      ②      ③      ④      ⑤  
A B + C D \* E / - F +



栈



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

## 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

用于存放当前暂时还不能确定运算次序的操作数

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

## 中缀表达式转后缀表达式（机算）

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。

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

①

④

②

③

⑤



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

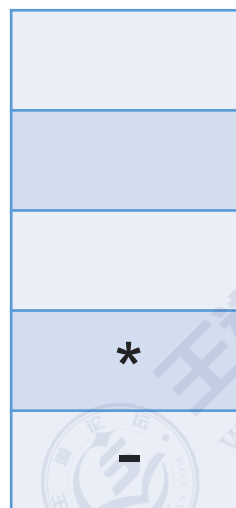
①

②

③

④

⑤



用于存放当前暂时还不能确定运算次序的运算符

A B + C D

## 中缀表达式的计算（用栈实现）

中缀转后缀  
+  
后缀表达式求值  
两个算法的结合

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）

## 中缀表达式的计算（用栈实现）

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）

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

①

④

②

③

⑤

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

①

②

③

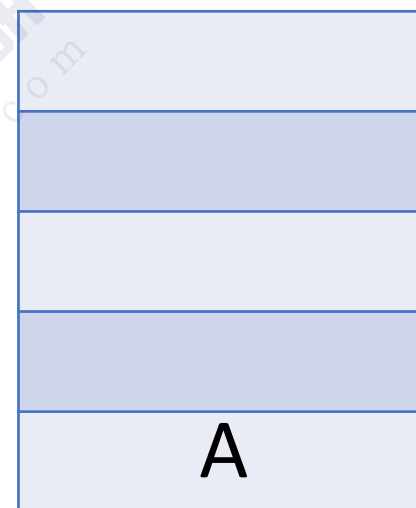
④

⑤

运算符栈



操作数栈



## 中缀表达式的计算（用栈实现）

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）

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



④

②

③

⑤

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

①

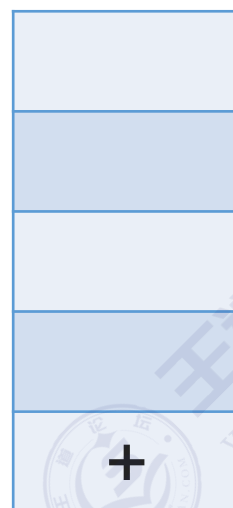
②

③

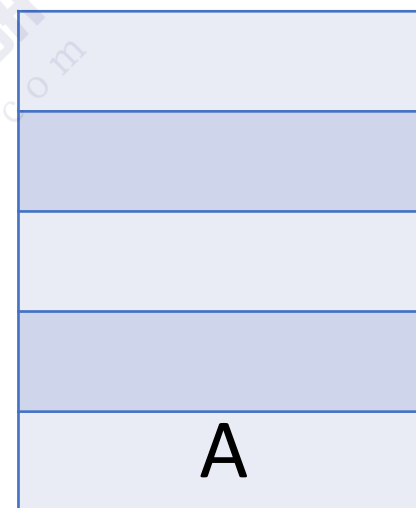
④

⑤

运算符栈



操作数栈



## 中缀表达式的计算（用栈实现）

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）

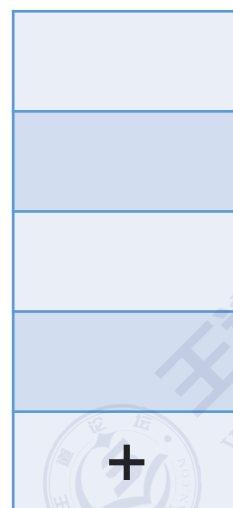
A + B - C \* D / E + F

① ↑ ④ ② ③ ⑤

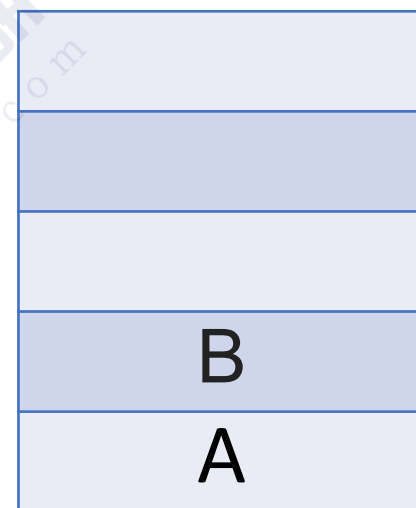
A B + C D \* E / - F +

① ② ③ ④ ⑤

运算符栈



操作数栈



## 中缀表达式的计算（用栈实现）

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）

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

①



②

③

⑤

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

①

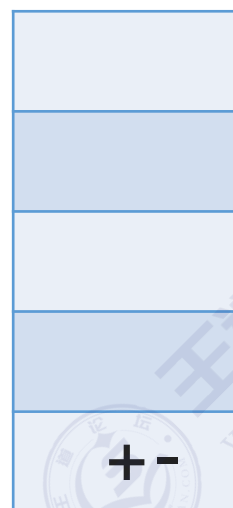
②

③

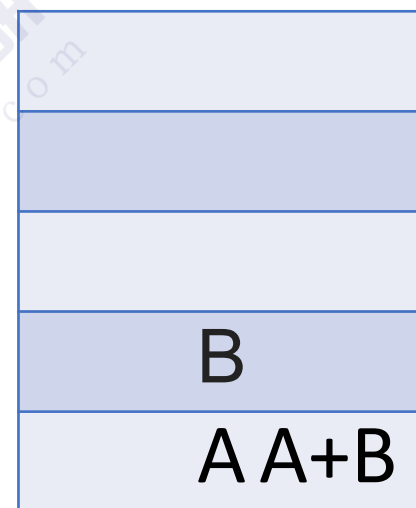
④

⑤

运算符栈



操作数栈



## 中缀表达式的计算（用栈实现）

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）

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

①

④

②

③

⑤

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

①

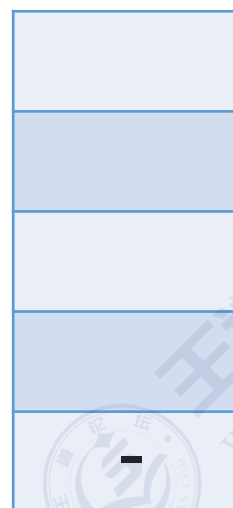
②

③

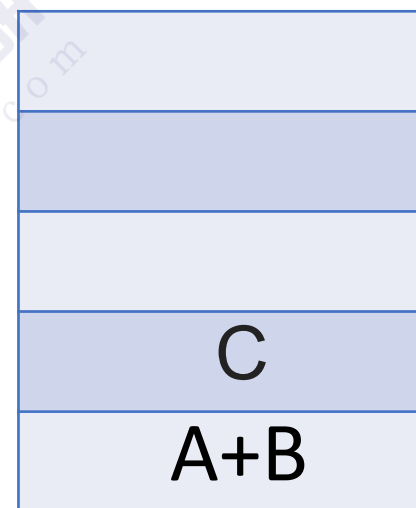
④

⑤

运算符栈



操作数栈



## 中缀表达式的计算（用栈实现）

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）

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

①

④



③

⑤

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

①

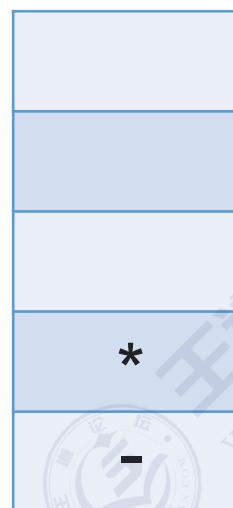
②

③

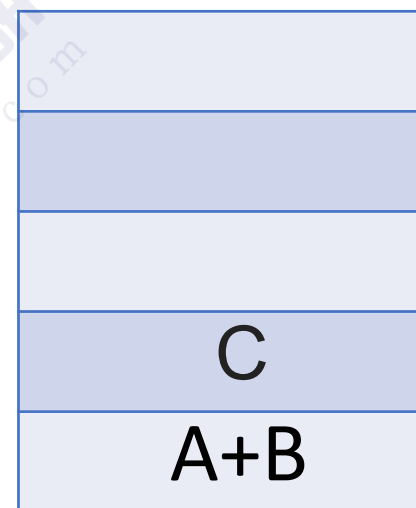
④

⑤

运算符栈



操作数栈



## 中缀表达式的计算（用栈实现）

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）

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

①

④

②

③

⑤



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

①

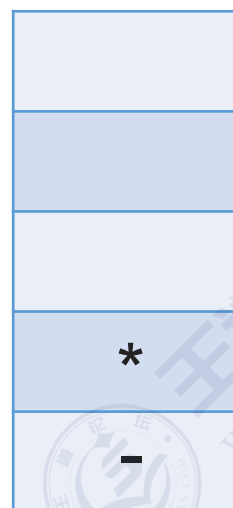
②

③

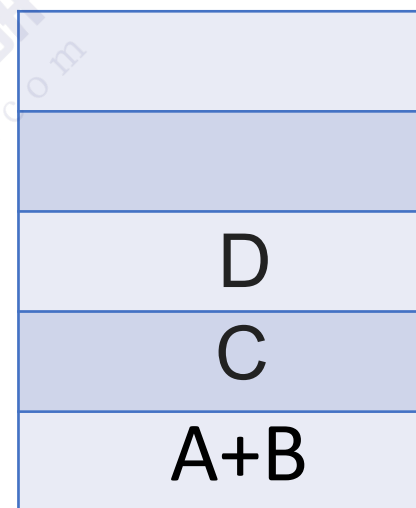
④

⑤

运算符栈



操作数栈



## 中缀表达式的计算（用栈实现）

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）

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

①

④

②



⑤

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

①

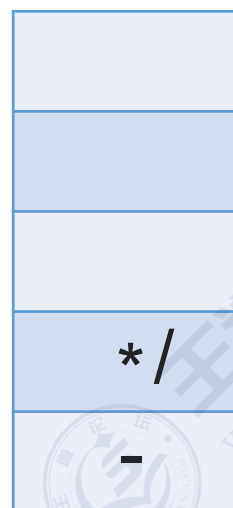
②

③

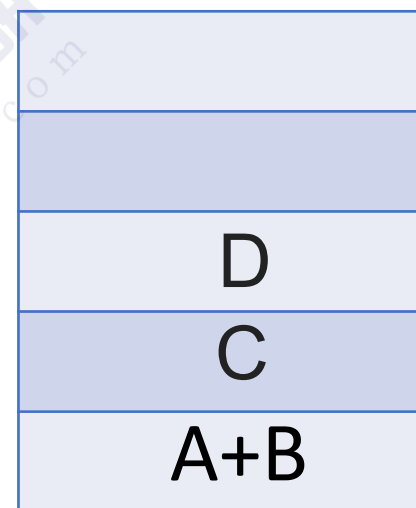
④

⑤

运算符栈



操作数栈



## 中缀表达式的计算（用栈实现）

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）

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

①

④

②

③

⑤



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

①

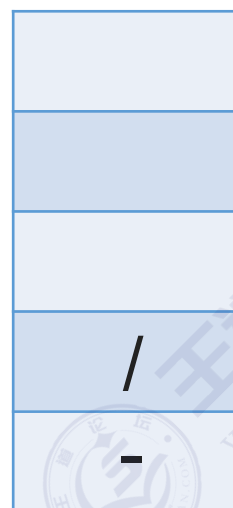
②

③

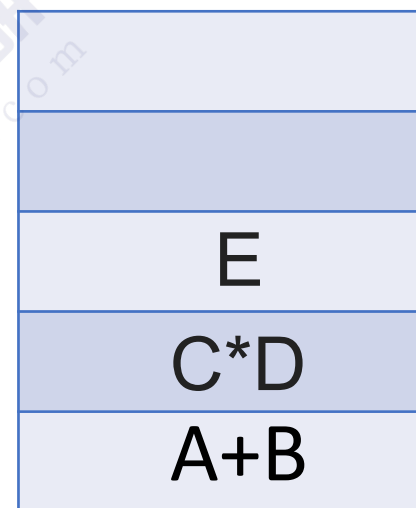
④

⑤

运算符栈



操作数栈



## 中缀表达式的计算（用栈实现）

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）

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

①

④

②

③



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

①

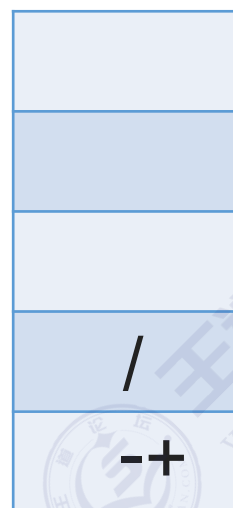
②

③

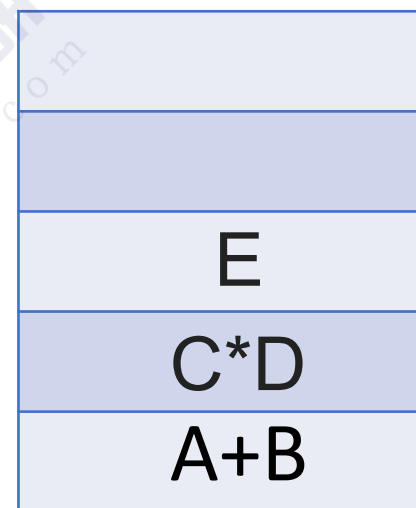
④

⑤

运算符栈



操作数栈



## 中缀表达式的计算（用栈实现）

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）

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

①

④

②

③

⑤



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

①

②

③

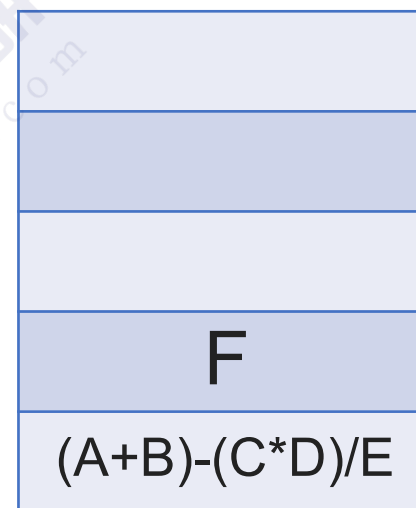
④

⑤

运算符栈



操作数栈



## 中缀表达式的计算（用栈实现）

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）



就我这小暴脾气哟~

关于Why的思考：搞这么复杂有毛意义？

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

①

④

②

③

⑤



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

①

②

③

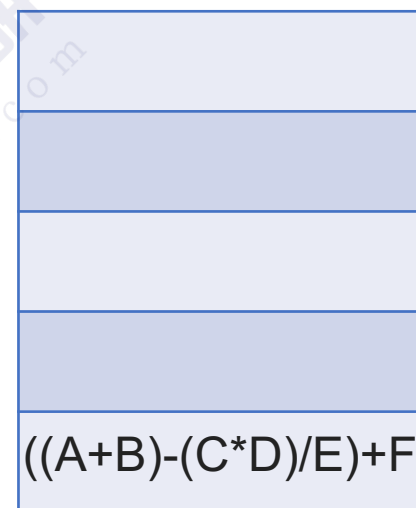
④

⑤

运算符栈



操作数栈



## 知识回顾与重要考点

**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)