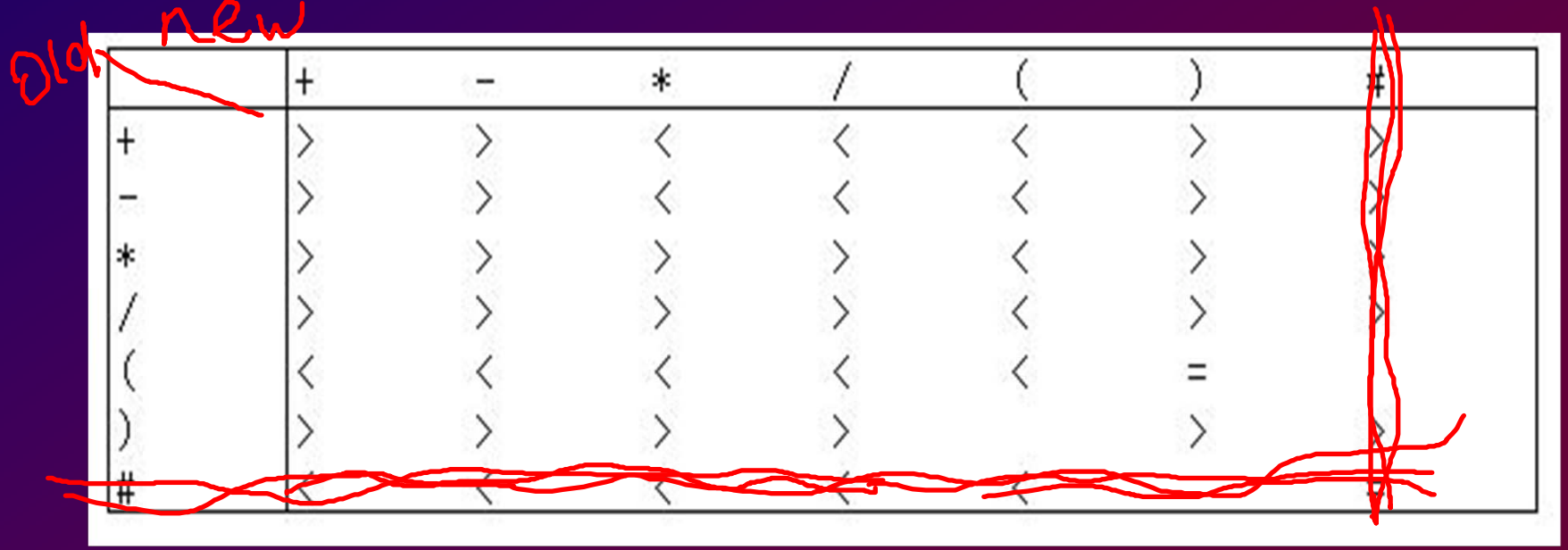
表达式求解

- 假设：
  - 只涉及＋、－、*和/四种双目运算符号（其余可自己扩展）
  - 不考虑左、右括号的匹配，假设已经保证是匹配的
  - 设只出现小括号（其余括号可自己扩展）
  - 设表达式尾标记为＃
- 分析：
  - 表达式由三部分构成：运算符、操作数和界限符（括号和表达式尾标记）
  - 将括号也看成一种运算符，得到优先级别表

表达式求解

- 运算符之间的优先级



	+	-	*	/	(	)	#
+	>	>	<	<	<	>	>
-	>	>	<	<	<	>	>
*	>	>	>	>	<	>	>
/	>	>	>	>	<	>	>
(	<	<	<	<	<	=	
)	>	>	>	>		>	>
#	<	<	<	<	<		=

表达式求解

- 分析：
  - 运算根据运算优先级进行，而优先级与操作数无关——两者可独立存储；
  - 当读到的当前运算符优先级低于前一个时，可以计算前一个运算符——因此必须记录前一个运算符，即有一个读入运算符的逆序（栈）
  - 计算某个运算符时，它所对应的操作数应该是最后两个操作数——因此必须记录操作数的逆序（栈）

表达式求解

- 结论：
  - 设定两个栈：
    - 操作数栈——保存读入操作数
    - 运算符栈——保存读入运算符
- 算法：
  - 初始化两个栈：运算符栈**OPTR**和操作数栈**OPND**
  - 将起始标记‘#’入**OPTR**；
  - 循环读入表达式字符**ch**，直到遇到尾标记‘#’为止，按**ch**分情况讨论：
    - **Ch**为操作数：入操作数栈**OPND**

表达式求解

- 算法：（续）
  - **ch为运算符，取出OPTR运算符栈中的栈顶运算符ch1，比较ch与ch1的优先级：**
    - **Ch==ch1**：去除括号,运算符栈进行出栈操作**pop(OPTR)**，读入下一个符号**ch**；
    - **Ch>ch1**：将**ch**入栈**push(OPTR,ch)**，读入下一个符号**ch**；
    - **Ch<ch1**：先计算表达式中前一个子式
      - 运算符栈出栈theta=pop(OPTR)；
      - 操作数栈出栈两次: rop=pop(OPND)； lop=pop(OPND)；其中第一次出栈的是右操作数，第二次是左操作数；
      - 进行运算：x=operate(lop theta rop)；并将结果入操作数栈push(OPND,x)；

表达式求解

- 算法：（续）
  - 读到‘#’时，从**OPND**操作数栈中取出结果，该结果就是表达式的值；
  - 释放两个栈
- 例如：
  - 表达式：**3*(4+5)/2-3-(5+2)**