4. 栈与队列

(c5) 栈应用:逆波兰表达式

将欲去之,必固举之

将欲夺之,必固予之

将欲灭之,必先学之

邓俊辉

deng@tsinghua.edu.cn

RPN)

- ❖ 逆波兰表达式 (Reverse Polish Notation)
 - J. Lukasiewicz (12/21/1878 02/13/1956)
- ❖ 在由运算符(operator)和操作数(operand)组成的表达式中不使用括号(parenthesis-free),即可表示带优先级的运算关系
- ❖例如: 0!+123+4*(5*6!+7!/8)/9
 - 0 ! 123 + 4 5 6 ! * 7 ! 8 / + * 9 / +
- ❖又如: 0!-(1+23-4-56)*7*8-9
 - 0 ! 1 23 + 4 56 7 * 8 * 9 -
- ❖ 相对于日常使用的中缀式(infix), RPN亦称作后缀式(postfix)
- ❖作为补偿,须额外引入一个起分隔作用的元字符(比如空格) //较之原表达式,未必更短

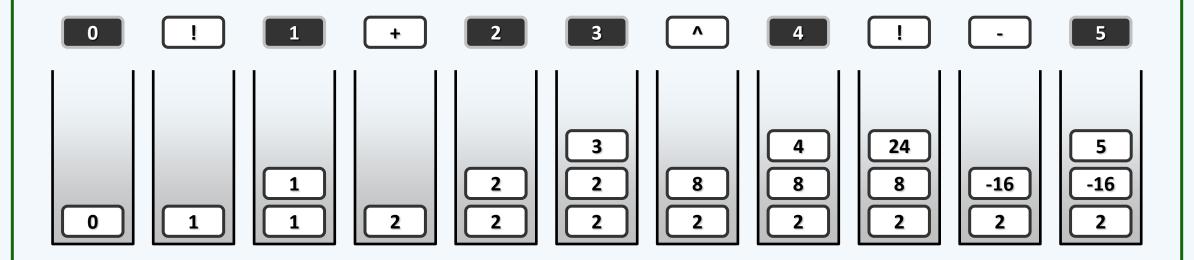
求值算法

```
❖rpnEvaluate(expr) //假定RPN表达式expr的语法正确
   引入栈S,用以存放操作数 //故亦称作栈式求值
   while ( expr尚未扫描完毕 ) {
     读入expr的下一元素x
     if (x是操作数)
       将x压入S
     else { //x是运算符
       从栈S中弹出运算符x所需数目的操作数
       对弹出的操作数实施x运算,并将运算结果重新压入S
     } //else
   } //while
   返回栈顶 //也是栈底
```

实例

0 | ! | 123 | + | 4 | 5 | 6 | ! | * | 7 | ! | 8 | / | + | * | 9 | / | +

0 ! 1 + 2 3 ^ 4 ! - 5 ! 6 / - 7 * 8 * - 9 -



!	6	/	7	*	8	* -	9	-)
1		1.1	 	- 11	- 11	- 11	-11	1.1	

120 -16	6 120 -16	20	-36	7 -36	-252	8 -252	-2016		9	
2	2	2	2	2	252	252	2	2018	2018	2009

0 ! 1 + 2 3 ! 4 - 5 ^ - 67 * - 8 - 9 +



32

2010

2004

infix到postfix: 手工转换

❖例如:(0!+1)^(2*3!+4-5)

假设:事先未就运算符之间的优先级关系做过任何约定

1) 用括号显式地表示优先级

```
{([0!] +1) ^([(2*[3!]) +4] -5)}
```

2) 将运算符移到对应的右括号后

```
{([0]!1)+([(2[3]!)*4]+5)-}^
```

3) 抹去所有括号

```
0 ! 1 + 2 3 ! * 4 + 5 - ^
```

4) 稍事整理,即得

```
0 ! 1 + 2 3 ! * 4 + 5 - ^
```

❖中缀式求值算法evaluate()略做扩展,亦可同时完成RPN转换...

infix到postfix:转换算法

```
❖float <u>evaluate</u>( char* S, char* & RPN ) { //RPN转换
  /* ..... */
  while (!optr.empty()) { //逐个处理各字符,直至运算符栈空
    if ( isdigit(*S) ) //若当前字符为操作数,则直接将其
     { readNumber(S, opnd); append(RPN, opnd.top()); } //接入RPN
    else //若当前字符为运算符
     switch( orderBetween( optr.top(), *S ) ) {
      /* ..... */
       case '>': { //且可立即执行,则在执行相应计算的同时将其
        char op = optr.pop(); append(RPN, op ); //接入RPN
        /* ..... */
       } //case '>'
```

Data Structures & Algorithms (Fall 2013), Tsinghua University

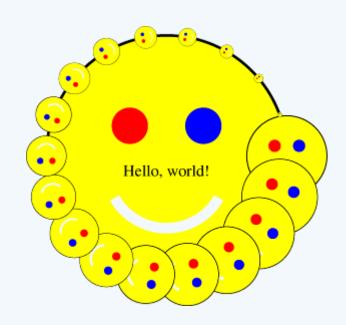
/ *

PostScript

- ❖ 诞生于1985,支持设备独立的图形描述
- ❖ (1个解释器 + 5个栈) × RPN语法
- ❖ operand stack: 存放操作数及运算结果
- ❖ 一旦遇到操作符,则 弹出相应数目的元素

实施计算,并

将(可能多个、一个或零个)结果入栈



- ❖ dictionary / execution / graphics state / clipping path stacks
- ❖实例: 4 4 mul 5 5 mul add 7 mul 7 mul
- ❖ 提供基础且强大的图形功能,支持数据类型、变量、函数/宏 ...
 newpath 300 600 80 22.5 -22.5 arc stroke