栈的应用:四则运算表达式值:

四则运算的口诀是:先乘除,后加减,从左算到右,先括号内后括号外.那计算器是怎样实现这个功能的呢?

仔细观察后发现，括号都是成对出现的,有左括号就一定会有右括号,对于多重括号,最终也是完全嵌套匹配的.这和栈正好合适,只要碰到左括号,就将左括号入栈,不管表达式有多少重括号，反正遇到左括号就进栈,而后面出现右括号时,就让栈顶的左括号出栈,期间让数字运算,这样,最终有括号的表达式就从左到右巡查一遍,栈应该是由空到有元素,最终再因全部匹配成功后成为空栈的结果.

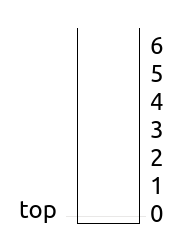
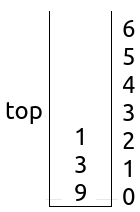
但对于四则运算,括号也只是当中的一部分,先乘除后加减使得问题依然复杂.

波兰的逻辑学家,想到了一种不需要括号的后缀表达式,我们将其称为逆波兰表示.

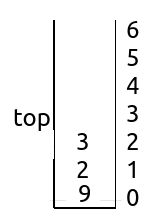
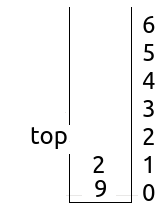
例如”9+(3-1)\*3+10/2” 如果用后缀表达式法应该是这个样子”9 3 1 - 3 \* +”叫后缀的原因是所有的运算符都是要在运算数字的后面出现.显然这里没了括号.

后缀表达式运算规则:从左到右遍历表达式的每个数字和符号,遇到是数字就进栈,遇到是符号,就将处于栈顶两个数字出栈,进行运算,运算结果进栈，一直到最终获得结果.

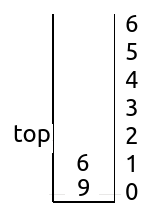
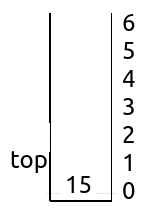
1. 初始化一个空栈,利用此栈来对要运算的数字进出使用.如图1:
2. 后缀表达式中前三个都是数字,所以9,3,1进栈,如图2:

1. 接下来是”-”,所以将栈中的1出栈做为减数，3出栈做为被减数,运算3-1得到2，再将2进栈,如图３所示:
2. 接着是数字3进栈,如右图所示:

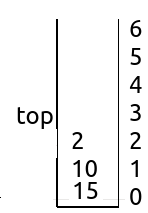
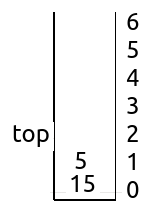


1. 后面是”\*”,也就意味着栈中３和２出栈,2与３相乘,得到６,并将6进栈。
2. 下面是”+”,所以栈中6和9出栈，９与６相加，得到15,将15进栈,如右图:

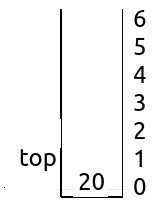
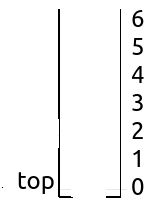
 

７.接下来是10与２进栈,如图所示:

8.接下来是符号”/”，因此,栈顶的2与10出栈,10与２相除,得到５,将５进栈,如图所示:

1. 最后一个是符号”+”,所以15与５出栈并相加,得到20,将20进栈,如图所示:
2. 结果是20出栈,栈变为空,如图所示:

果然,后缀表达式可以很顺利解决计算的问题,那么后缀表达式是怎样出来的呢?这个问题搞不清楚,等于没有解决.

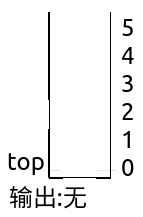
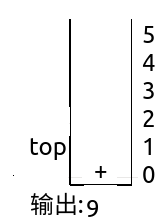
中缀表达式转后缀表达式:

我们把平时所用的标准四则运算表达式,即”9+(3-1)\*3+10/2”叫做中缀表达式,因为所有的运算符号都在两数字的中间,现在我们的问题是中缀到后缀的转化.

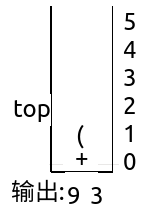
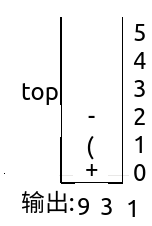
中缀表达式”9+(3-1)\*3+10/2”转化为后缀表达式”9 3 1 - 3 \* + 10 2/ +”

规则:从左到右遍历中缀表达式的每个数字和符号,若是数字就输出,即成为后缀表达式的一部分:若是符号,则判断其与栈顶符号的优先级,是右括号或优先级低于栈顶符号(乘除优先加减)　则栈顶元素依次出栈并输出,并将当前符号进栈,一直到最终输出后缀表达式为止.

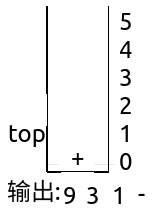
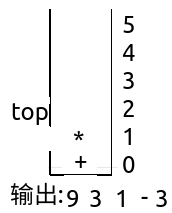
1. 一旦初始化空栈,用来对符号进出栈使用.
2. 第一个字符是数字9，输出9，后面是符号”+”,进栈。

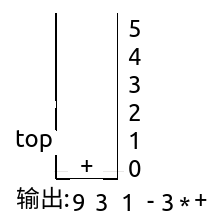
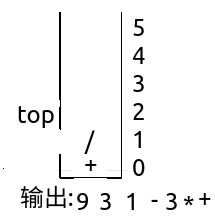
1. 第三个字符是”(”,依然是符号,因其只是左括号,还未配对,故进栈.
2. 第四个是数字3,输出,总表达式为9 3,接着是”-”，进栈,

1. 接着是数字1，输出,总表达式为9 3 1,后面是符号”)”,此时我们需要去匹配此前的”(”,所以栈顶依次出栈,并输出,直到”(”出栈为止.此时左括号上方只有”-”,因此输出”-”.总的输出表达式为9 3 1-
2. 接着是数字3,输出,总的表达式为9 3 1 -3。紧接着是符号”\*”,因为此时的栈顶元素＝＝符号为”+”号,优先级低于”\*”号,因此不输出,“\*”号进栈.

1. 之后是符号”+”,此时当前栈顶元素”\*”,比这个”+”的优先级高,因此栈中元素出栈并输出(没有比”+”号更低的优先级，所以全部出栈),总输出表达式为9 3 1 - 3\* +。然后将当前这个符号”+”进栈.也就是说,前6张图的栈底”+”是指中缀表达式开头的9后面的那个”+”,
2. 紧接着数字10,输出,总表达式变为9 3 1 - 3 \* + 10.后是符号”/” ,所以”/”进栈.

1. 最后一个数字2,输出,总的表达式为9 3 1 - 3 \* + 10 2。
2. 因已经到最后,所以将栈中符号全部出栈并输出.最终输出的后缀表达式为 9 3 1 - 3 \* + 10 2 / +。

