题目：堆栈应用: 逆波兰式解决四则运算

涉及知识点：字符串处理、堆栈操作、基本运算

要求： 编写一个程序实现四则运算。 程序运行时输入算术表达式, 运算符包括 +, -, *, /, (,)，输出表达式的值。

比如输入： 1 + 4 * （4 - 1.5）- （5 / 2.）

输出结果：8.5

其他：（1）如果你希望降低难度，可只处理整数； （2）输出格式不限制，数据正确即可。比如 8.500000，8.5， 8.50 都可以。

算法： 网上搜索 逆波兰式 （后缀表达式）的应用，这里是搜索的某个网页，供参考。

逆波兰表达式又叫做后缀表达式，它将复杂表达式转换为可以依靠简单的操作得到计算结果的表达式，解决了四则运算中括号改变运算符优先级的问题。四则运算的表达式一般都是中缀表达式如 1+2*(3-4)+5，即操作符在两个操作数之间。四则运算需要两个步骤，一是把中缀表达式转为后缀表达式，二是由后缀表达生成结果

中缀表达式转为后缀表达式算法描述：
(1)首先有个包含中缀表达式元素列表 sourceList，然后创建一个符号列表 destList 保存最终后缀表达式，创建一个操作符堆栈 opStack
(2)从 sourceList 取出一个元素 A
(3a)如是数字则加入到 destList 中
(3b)如果元素 A 是运算符，将操作符 A 与操作符堆栈 opStack 栈顶的运算符的优先关系相比较。如果，优先关系高于 opStack 栈顶的运算符，则将 该运算符压入操作符堆栈 opStack。倘若不是(低于或等于)的话，则将运算符栈 opStack 栈顶的运算符从栈中弹出保存到 destList，重复此 步骤，直到作符 A 压入操作符堆栈 opStack。
(3c)若元素 A 是左括号"("，则压入操作符堆栈 opStack
(3d)若元素 B 是右括号")"，则操作符堆栈 opStack 弹出操作符并加入到 destList 中，直到弹出左括号"("。
(5)从步骤 2 重复上述操作，所有元素处理完毕后将操作符堆栈 opStack 弹出操作符并加入到 destList 中，这样中缀式表示的简单算术表达式转化为逆波兰表示的简单算术表达式。

示例：
中缀表达式如 1+2*(3-4)+5，构造元素列表 1,+,2,*,(,3,-,4,),5,构造一个空最终后缀表达式 destList，一个操作符堆栈 opStack
1、取出“1”，destList【1】，opStack【】
2、取出“+”，destList【1】，opStack【+】
3、取出“2”，destList【1,2】，opStack【+】
4、取出“*”，destList【1,2】，opStack【+,*】
5、取出“(”，destList【1,2】，opStack【+,*,(】
6、取出“3”，destList【1,2,3】，opStack【+,*,(】
7、取出“-”，destList【1,2,3】，opStack【+,*,(,-】
8、取出“4”，destList【1,2,3,4】，opStack【+,*,(,-】
9、取出“)”，destList【1,2,3,4,-】，opStack【+,*】 //操作符堆栈 opStack 弹出操作符并加入到 destList 中，直到弹出左括号"("
10、取出“+”，destList【1,2,3,4,-,*,+】，opStack【+】 //加号优先级不大于【+,*】
11、取出“5”，destList【1,2,3,4,-,*,+,5】，opStack【+】
12、处理完毕，destList【1,2,3,4,-,*,+,5,+】，opStack【】

后缀表达式到计算结果算法描述：

遍历储存后缀表达式的列表，将元素依次进栈，当遇到操作符时，连续出栈两个元素，进行运算，再将结果进栈，最后栈内留下的元素就是计算结果
示例：
后缀表达式 destList【1,2,3,4,-,*,+,5,+】，结果堆栈 resultStatck【】
格式
输入-->:结果
[1,2,3,4]-->:resultStatck【1,2,3,4】
[-]-->:resultStatck【1,2,3-4】
[*]-->:resultStatck【1,2*(3-4)】
[+]-->:resultStatck【1+2*(3-4)】
[5]-->:resultStatck【1+2*(3-4),5】
[+]-->:resultStatck【1+2*(3-4)+5】