**计算器题目涉及到的中缀转后缀表达式：**

**1. 前缀表达式**(Prefix Notation)是指将运算符写在前面操作数写在后面的不包含括号的表达式

* **后缀表达式**(Postfix Notation)与之相反，是指运算符写在操作数后面的不包含括号的算术表达式，也叫做逆波兰表达式
* **中缀表达式**(Infix Notation)就是常用的将操作符放在操作数中间的算术表达式。前缀表达式和后缀表达式相对于中缀表达式最大的不同就是去掉了表示运算符优先级的括号

**2. 中缀转后缀**

人工转换的方法，假设有一个中缀表达式a+b\*c-(d+e)

1. 首先将这个中缀表达式的所有运算加括号((a+(b\*c))-(d+e))
2. 然后将所有运算符放到括号后面，这样就变成了((a(bc)\* )+ (de)+ )-
3. 把所有括号去掉abc\*+de+ -，最后得出的结果就是后缀表达式

**5. 后缀表达式计算机求值**

从左至右扫描表达式，遇到数字时，将数字压入堆栈，遇到运算符时，弹出栈顶的两个数，用运算符对它们做相应的计算（次顶元素 op 栈顶元素），并将结果入栈；重复上述过程直到表达式最右端，最后运算得出的值即为表达式的结果

**例如后缀表达式“3 4 + 5 × 6 -”：**

1. 从左至右扫描，将3和4压入堆栈；
2. 遇到+运算符，因此弹出4和3（4为栈顶元素，3为次顶元素，注意与前缀表达式做比较），计算出3+4的值，得7，再将7入栈；
3. 将5入栈；
4. 接下来是×运算符，因此弹出5和7，计算出7×5=35，将35入栈；
5. 将6入栈；
6. 最后是-运算符，计算出35-6的值，即29，由此得出最终结果。

**为什么要设计后缀表达式，有什么好处？**  
便于用栈实现计算，而且不需要表示运算符优先级的括号。

1. enum是枚举数据类型

2. template**模板**是c++支持**参数化多态**的工具，使用模板可以使用户为类或者函数声明一种一般模式，使得类中的某些数据成员或者成员函数的参数、返回值取得任意类型。函数模板针对仅参数类型不同的函数；类模板针对仅数据成员和成员函数类型不同的类。

3. **c++** stack(堆栈）是一个容器的改编，**它实现了一个先进后出的数据结构（FILO）**

使用该容器时需要包含#include<stack>头文件；

定义stack对象的示例代码如下：

stack<int>s1;

stack<string>s2;

stack的基本操作有：

1.入栈：如s.push(x);

2.出栈:如 s.pop().**注意：出栈操作只是删除栈顶的元素，并不返回该元素。**

3.访问栈顶：如s.top();

4.判断栈空：如s.empty().当栈空时返回true。

5.访问栈中的元素个数，如s.size（）;