牛客网-华为机试练习题 54

题目描述

给定一个字符串描述的算术表达式，计算出结果值。

输入字符串长度不超过100，合法的字符包括”+, -, *, /, (,)”，”0-9”，字符串内容的合法性及表达式语法的合法性由做题者检查。本题目只涉及整型计算。

输入描述:

输入算术表达式

输出描述:

计算出结果值

示例1

输入

400+5

输出

405

解决代码:

```java
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import java.util.Stack;

public class Main {

    public static void main(String[] args) throws Exception {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        String line = "";
        while((line=br.readLine())!=null)
        {
            Stack<Character> stack = new Stack<Character>();
            List<Object> list = new ArrayList<Object>();

            //利用中缀表达式构建后缀表达式
            for(int i=0;i<line.length();++i)
            {
                String T = "";
                while(i<line.length() && line.charAt(i)>='0'&&line.charAt(i)<='9')
                    T = T + line.charAt(i++);
                if(!T.equals("")){  //T不等于""，说明T是一个数字字符串
                   list.add(new Integer(T));  //转为一个Integer对象
                    --i;
                }
                else  // T等于空说明当前的charAt(i)不是数字 ,是操作符号
                {
                    if(line.charAt(i)=='(')//如果是 ( 则先入栈
                        stack.push(line.charAt(i));
                    else if(line.charAt(i)=='+'||line.charAt(i)=='-
'||line.charAt(i)=='*'||line.charAt(i)=='/') /*是符号 并且优先级大于*/

                    {
                        if(stack.isEmpty())//若果栈是空的，直接加入第一个符号
                            stack.push(line.charAt(i));
                        else if(isUpperPro(line.charAt(i), stack.peek()))//新符号优先级大于栈顶
                            stack.push(line.charAt(i));
                        else{//新符号优先级低于栈顶
                            while(!stack.isEmpty() && stack.peek()!='(' &&
!isUpperPro(line.charAt(i),stack.peek()))
                                list.add(stack.pop());
                            stack.push(line.charAt(i));
                        }
                    }
                    else if(line.charAt(i)==')'){
                        while(stack.peek()!='(')
                            list.add(stack.pop());
                        stack.pop();
                    }
                }
            }
            while(!stack.isEmpty())
                list.add(stack.pop());



            //利用后缀表达式求值
            Stack<Integer> pStack = new Stack<Integer>();
            Iterator<Object> it= list.iterator();
            while(it.hasNext())
            {
                Object temp = it.next();
                if(temp instanceof Integer)
                    pStack.push((Integer)temp);
                else if(temp instanceof Character){
                    int temp2 = pStack.pop();   //要注意出栈进栈的顺序，使得操作数也不一样
                    int temp1 = pStack.pop();
                    int res = getOP(temp1,temp2,(char)temp);
                    pStack.push(res);
                }
            }
            System.out.println(pStack.pop());
        }
    }

    private static int getOP(int temp1, int temp2, char charAt){
        if(charAt == '+') return temp1+temp2;
        if(charAt == '-') return temp1-temp2;
        if(charAt == '*') return temp1*temp2;
        if(charAt == '/') return temp1/temp2;
        return 0;
    }

    private static boolean isUpperPro(char charAt, char peek){
        if(peek=='(')  //如果栈顶元素是（,那么新字符优先级大于栈顶的（
            return true;
        if((charAt=='+'||charAt=='-')&&(peek=='*'||peek=='/'))
            return false;
        if((charAt=='*'||charAt=='/')&&(peek=='-'||peek=='+'))
            return true;
        if((charAt=='+'||charAt=='-')&&(peek=='+'||peek=='-'))
            return false;
        if((charAt=='*'||charAt=='/')&&(peek=='*'||peek=='/'))
            return false;
        return false;
    }
}
```