Data Structures and Algorithms

Laboratory Projects

Minimum Requirements on Writing a

Project Report

**Linear List**

王喆

Date: 2017-11-02

**Chapter 1: Introduction**

这个程序实现了+ - \* / #(单目负) 能够通过优先级进行浮点数计算，若有语法错误会具体指出是哪种错误，错误共分：

1. error:有非法字符请检查
2. 操作数与操作符不匹配(操作数过少)
3. 缺失(
4. ()内无操作数
5. 操作数与操作符不匹配(操作数过多)

当出现错误后可立即知道错误的位置，同时，还支持多次输入，每次输入以‘=’结束，输入’b’程序结束。

**Chapter 2: Algorithm Specification**

* 该程序共引入了三个头文件，分别为：

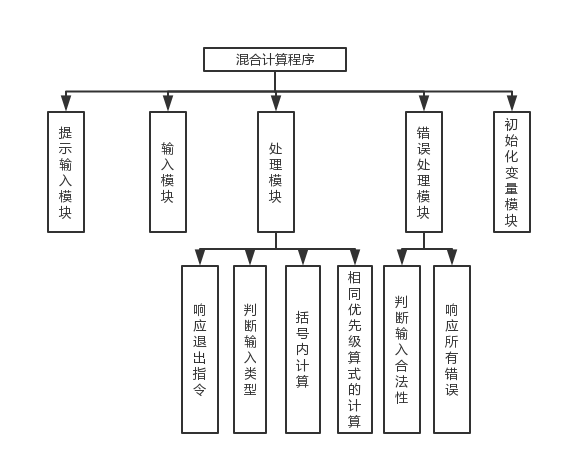
#include<iostream>

#include<stack>

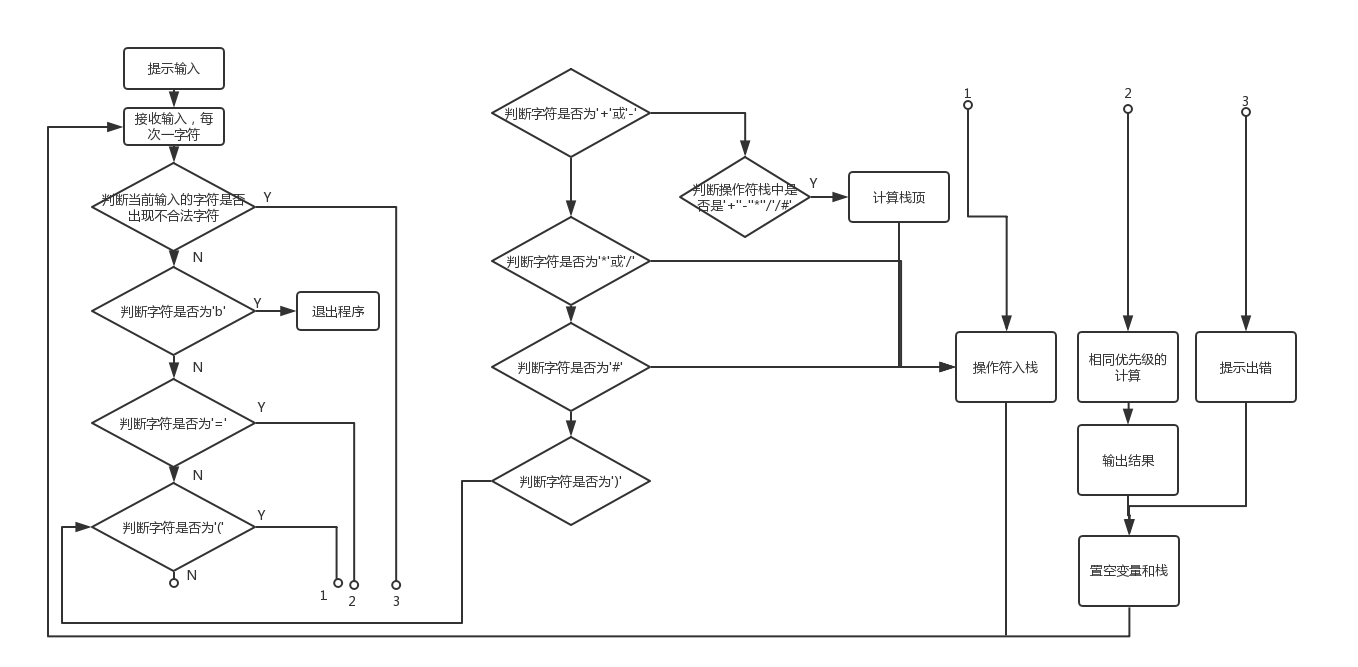
#include<stdlib.h>

程序中使用了标准库中的Stack<char>类和Stack<double>类，创建了两个栈ops和vals用以存取操作符和操作数，使用了栈的top(),pop(),push()方法，以对两个栈进行操作

* 系统设计思想：



* 程序流程图



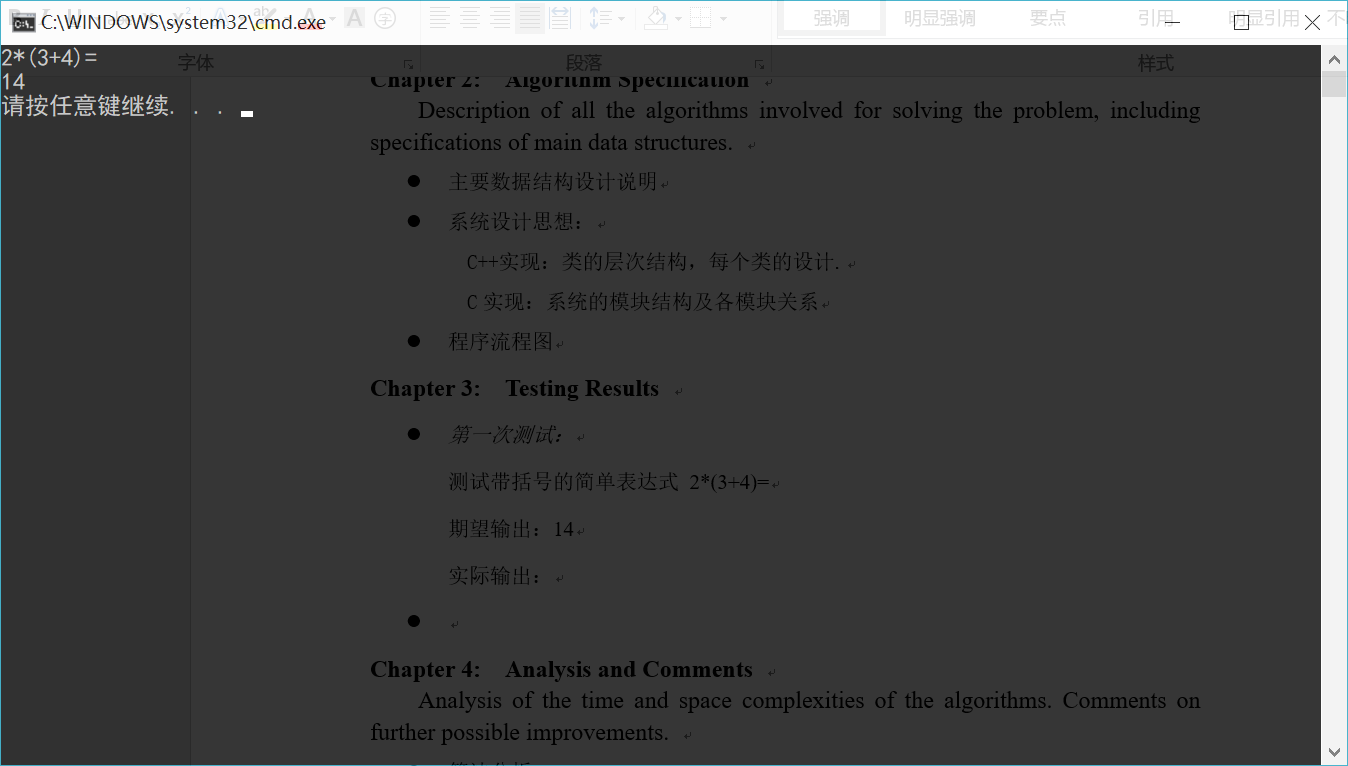
**Chapter 3: Testing Results**

* 第一次测试：(pass)

测试带括号的简单表达式 2\*(3+4)=

期望输出：14

实际输出：14

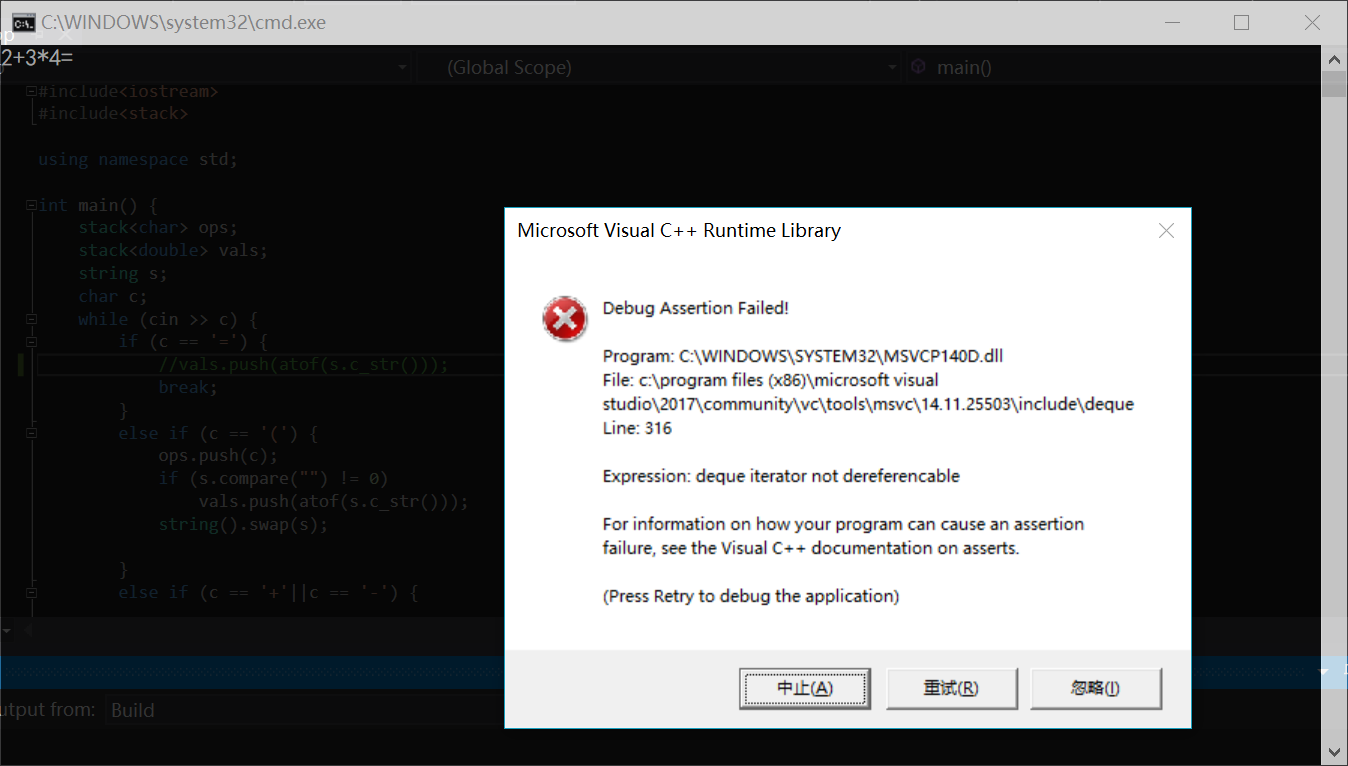


* 第二次测试：(fail)

测试不带括号的加减乘除混合运算 2+3\*4=

期望输出：14

实际输出：错误



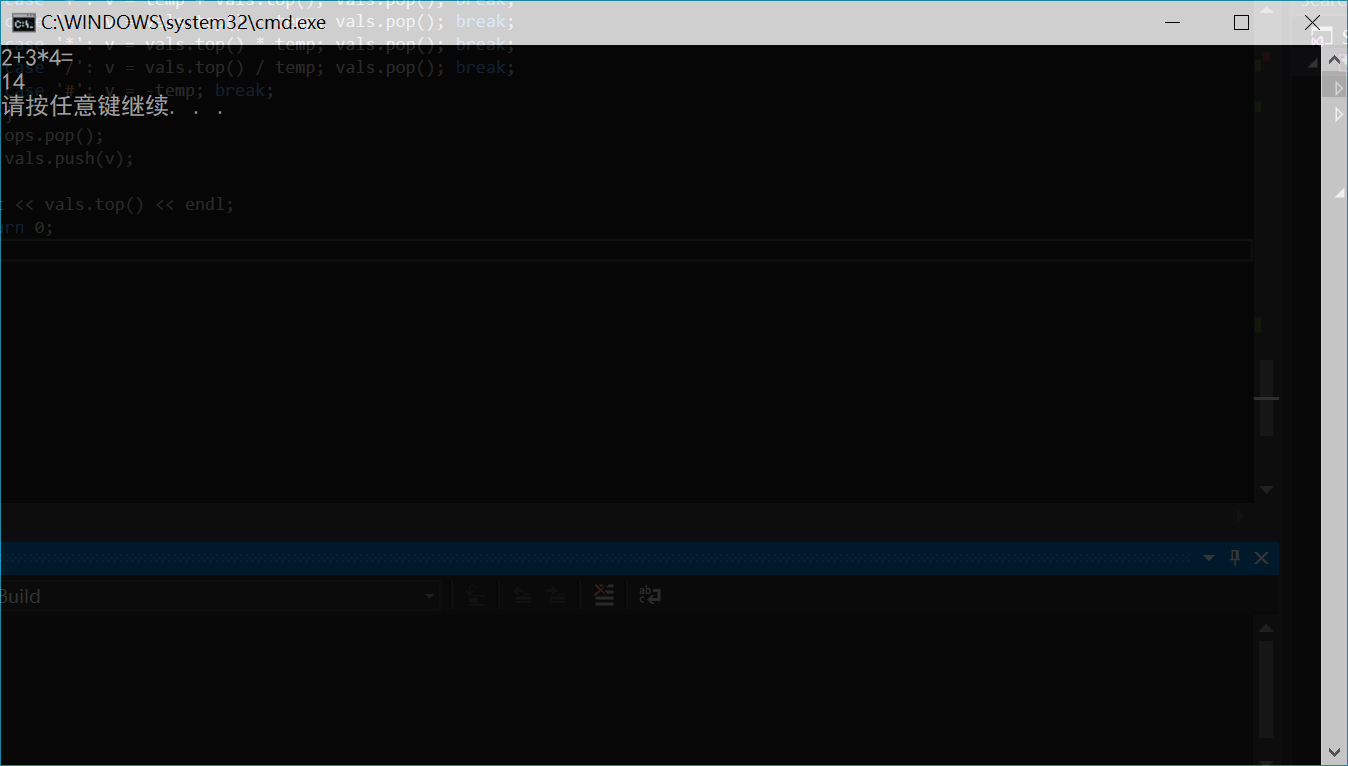
可能原因：运算符，操作数入栈出栈错误

* 第三次测试：(pass)

修改对读取到等号后将操作数入栈的逻辑，再次尝试 2+3\*4=

期望结果：14

实际输出：14

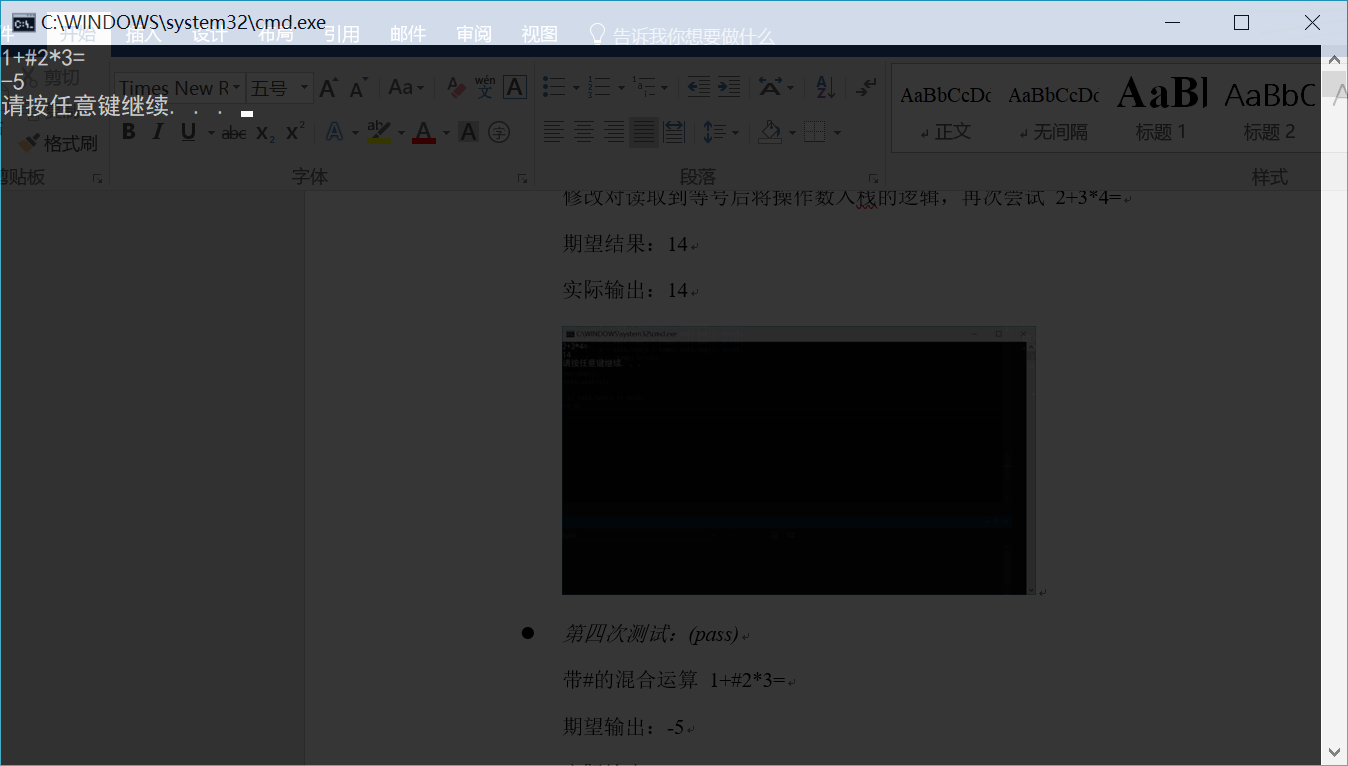


* 第四次测试：(pass)

带#的混合运算 1+#2\*3=

期望输出：-5

实际输出：-5

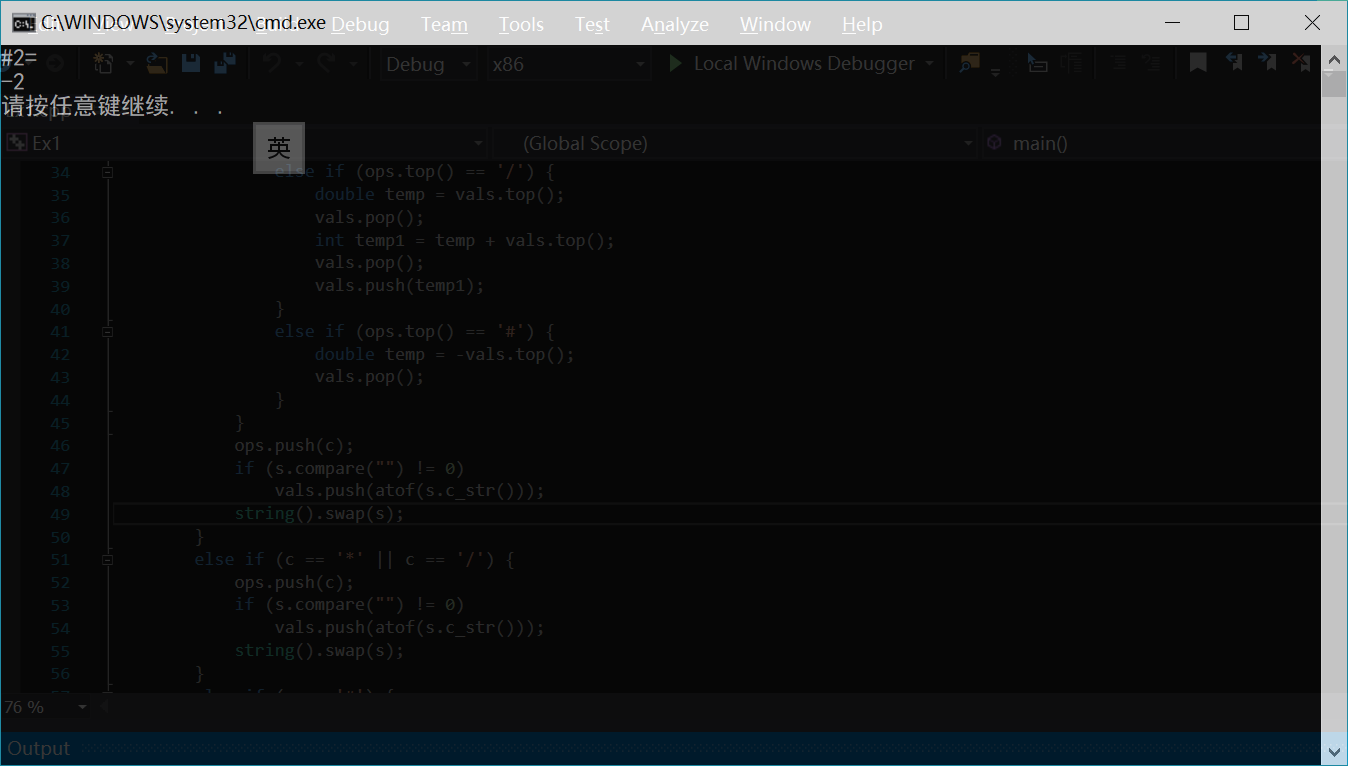


* 第五次测试：(pass)

单#运算 #2=

期望输出：-2

实际输出：-2

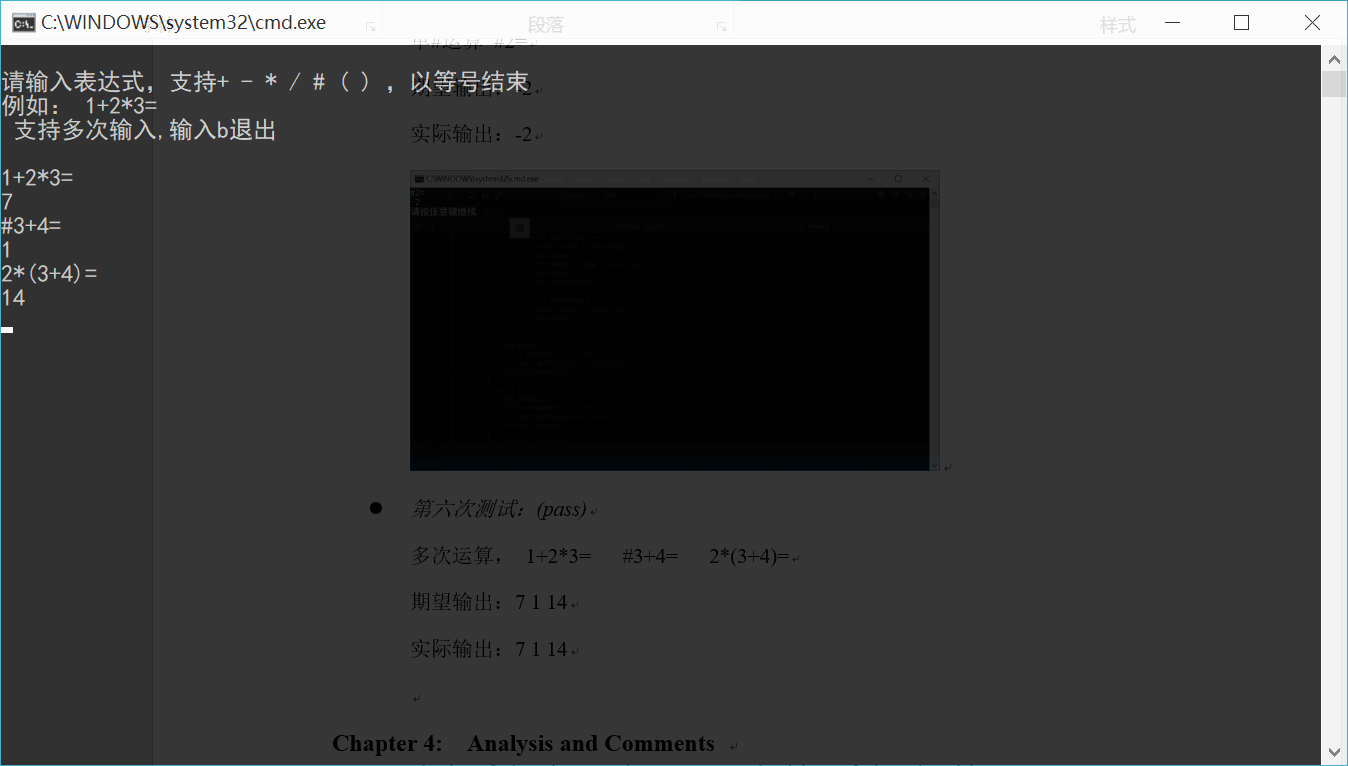


* 第六次测试：(pass)

多次运算， 1+2\*3= #3+4= 2\*(3+4)=

期望输出：7 1 14

实际输出：7 1 14



**Chapter 4: Analysis and Comments**

Analysis of the time and space complexities of the algorithms. Comments on further possible improvements.

* 算法分析

本算法共使用了两个栈，一个操作数栈和一个操作符栈，然后程序每次循环读取一个字符，若为数字则追加到字符串末尾，直到遇到操作符时，将操作数的字符串转换为double类型，当读取到的操作符优先级比栈顶操作符运算优先级高时直接入栈，否则从栈中取出一个操作符和与之匹配的操作数，并进行计算后入栈，当遇到左括号时入栈，遇到右括号则计算括号内的值直到遇到左括号。当全部操作完成后操作数栈的剩余的唯一浮点数即为所求的值。

算法复杂度:O(n)

* 算法特色

算法实现了实时计算，而不是传统的先将操作符、操作数按顺序全部入栈后再进行计算，相比前者**空间占用率较低，**同时对栈的操作次数也会相应减少，达到**加快程速度**的目的。

* 不足

错误判断模块没有与计算模块分离，较难维护。

**Declaration**

***We hereby declare that all the work done in this project titled "XXX" is of our independent effort as a group.***

**Duty Assignments:**

**Programmer: XXX**

**Tester: XXX**

**Report Writer: XXX**

Please keep in mind that these are the “minimum” requirements. Other requirements will be specified according to each project assignment.