数据结构实验报告

软件学院 软件工程 4 班 2113850 李鹏

——实验题目：表达式求值

一、整体思路及算法：

使用数据结构——栈的方式，将中缀表达式转为后缀表达式，对后缀表达式求值，来实现对给定的中缀表达式求值。另外，还要考虑表达式不合法的情况。

1.整体思路：

设置两个栈，用于存储表达式中涉及到的运算符

逐个字符分析表达式，直到全部字符都已分析完

若当前字符为数字，则判断是否后续字符也为数字，若为数字则进行拼接，直到下一个数字为运算符为止，此时将拼接好的多位数字压入数字栈中。（如果已经是最后一个字符则直接压入栈）

若当前字符为算数运算符，如果运算符栈为空则直接压入栈中；如果运算符不为空，则对运算符优先级进行判断：

如果当前运算符优先级大于等于栈顶运算符则直接压入栈中；如果优先级低于栈顶运算符，则从数字栈中取出两个数据，将当前栈顶运算符弹出进行运算，将结果压入数字栈中，将当前运算符压入运算符栈中。

2.具体操作：

使用两个栈，一个是操作数栈 stack1_num，另一个为操作符栈 stack2_op，分别存储操作数和运算符（此处可直接使用栈的类模板）。使用字符串 str 存储待求值的表达式数据。

接下来，从字符串 str 中逐个读取字符，并对该字符分析处理，操作数入栈 stack1_num，运算符入栈 stack2_op。其要求如下：

①如果碰到操作数：则直接将其压入操作数栈 stack1_num 中。

②如果碰到运算符：

•若运算符栈 stack2_op 为空或运算符栈 stack2_op 栈顶元素为左括号’(’或当前运算符为左括号’(’，则直接将其压入运算符栈 stack2_op 中。

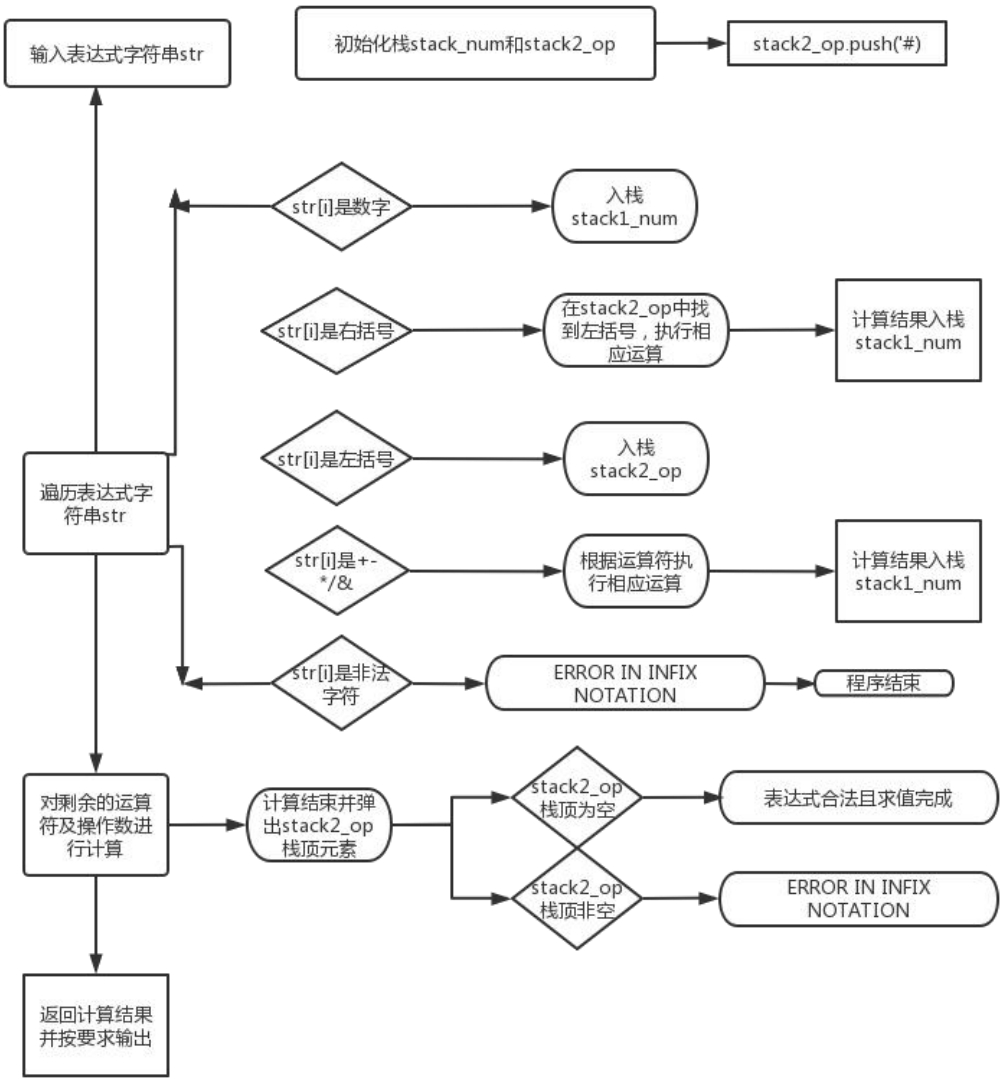
• 若当前运算符为右括号’)’，则循环执行“运算操作”，直至碰到左括号’(’，最后再将左括号’('出栈。

• 若当前运算符不为上述两种情况，且不为括号。若当前运算符的优先级大于(>)运算符栈 stack2_op 栈顶元素的优先级，则直接将当前运算符入栈；若当前运算符的优先级小于或等于(<=)运算符栈 stack2_op 栈顶元素的优先级，则执行一次“运算操作”。

优先级：

C_1 \ C_2	+	−	*	/	(	)	^	#
+	>	>	<	<	<	>	<	>
−	>	>	<	<	<	>	<	>
*	>	>	>	>	<	>	<	>
/	>	>	>	>	<	>	<	>
(	<	<	<	<	<	=	<	
)	>	>	>	>		>	>	>
^	>	>	>	>	<	>	>	>
#	<	<	<	<	<		<	=

二、流程图：



三、部分关键代码分析：

1.运算符优先级判断，用于运算符入栈与出栈

根据优先级顺序返回不同的值，根据返回值的大小可以得出优先级顺序。

+ - 为 1　　* / %为 2　　^为 3　　（为 0　　#为-1

```
int getprior(char op)
{
    switch (op)
    {
    case '+':
    case '-':
        return 1;
    case '*':
    case '/':
    case '%':
        return 2;
    case '^':
        return 3;
    case '(':
        return 0;
    case '#':
        return -1;
    }
}
```

2.关键求值运算：

```
double calculate(double a,char op,double b)
{
    double c;
    switch (op)
    {
    case '+':
    {c = b + a; break; }
    case '-':
    {c = b - a; break; }
    case '*':
    {c = b * a; break; }
    case '/':
    {c = b / a; break;}
    case '%':
    {c = (int)b % (int)a; break; }
    case '^':
    {c = pow( _X: b,  _Y: a); break; }
    }
    return c;
}
```

先出栈的元素 a 与后出栈的元素 b 在运算符 op 的作用下运算。

使用 switch 语句，根据运算符 op 选择对应的运算方式，并返回最终运算结果。

四、样例测试：

测试输入	(2-4)^3 (3*5*(4+8)%2) 1+2(（2.5-3.6）^3 （-5.3+3.3）^2 ((3-5+4*8/2)%3)^2 6+8$
测试目的	第一行与第二行测试代码能否解决 oj 所给的样例 第三行测试程序能否判断输入的表达式合法 第四行测试程序能否处理含有小数的表达式 第五行测试程序能否处理含有复数的表达式 第六行测试能否正确处理多个运算符的表达式 第七行测试程序能否处理其他的操作符
正确输出	**-1.00** **0.00** ERROR IN INFIX NOTATION -1.33 -4.00 4.00 ERROR IN INFIX NOTATION
实际输出	**-1.00** **0.00** ERROR IN INFIX NOTATION -1.33 -4.00 4.00 ERROR IN INFIX NOTATION
分析	开始时，程序没有考考虑为负数的情况，在输入含有负数的中缀表达式后，程序将负号当做减号转为转为后缀表达式，在计算含有负号的后缀表达式时，程序将其当做减号进行运算，最后后缀表达式会多出一个双目运算符，而操作数只有一个，程序发生了某种漏洞，导致输出为 0.00。程序在判断输入当然表达式是否为合法的表达式时，只考虑了括号是否匹配的情况，故在括号匹配的情况下输入其他运算符都是合法的，但是在转为后缀表达式时，程序中缺少处理其他符号的代码，故在将中缀表达式转为后缀表达式时，遇到了程序未知的运算符时，程序不能处理，跳不出转为后缀表达式的子程序，所以不能进行后缀表达式时的计算，从而没有结果的输出。
目前状况	程序目前的状况能够满足 oj 系统上所测试样例，即能够处理括号是否匹配的中缀表达式以及能够处理+、-、*、/、()、%

	六种运算符的计算，并且能够处理小数以及一位数以上的表达式的计算。

测试结果分析：

1. 根据测试样例的输出结果，可知该程序能够正确处理+、-、*、/、（）、^、%的运算，以及能够正确判断括号是否匹配的中缀表达式，并且能够正确地从相对简单的表达式到含有多种运算符的相对复杂的表达式的运算。对于小数以及多位数都够做到正确地处理，但是程序未能处理含有负数和其他不在题目范围内的操作符，还有待进一步优化。

2. 该程序直接使用 STL 库中的<stack>可达到缩减代码和提高效率的目的，免去了自己编写栈类型可能出现的错误，对系统提供的模板直接调用有了更深的了解，也体会到了系统函数的便捷。在以后的编程实例中，可多用已有类模板对问题操作实现简化。

五、实验总结：

通过本次数据结构的应用——表达式求值的编程实例作业，熟悉了对栈的创建与使用（尤其是入栈，出栈等过程更加形象化），了解并使用了栈的类模板，进而加深了对数据结构的认知，为接下来的学习奠定基础。

本次编程作业耗时较长，漏洞百出。在很快就有解体想法和思路之后，花费了大量时间才得以完成具体代码的实现，在实现过程概念中，也经历了不断的调试与修正，最终才得以通过全部测试样例。让我印象最深的便是 if，switch 语句。Switch 语句长期不用，有些生疏，忘了跳出语句的 break。而 if 语句看似简单，但搭配上 else if，else 等之后，逻辑稍微混乱便错误百出，仍然记得无论我输入什么测试数据，总会输出 ERROR IN INFIX NOTATION，就是由于 if 语句的错用。

因此，对于编程的学习，懂得思路和原理是基础，还要具备将思路转化为代码的能力，才能解决具体问题，对写代码，读代码的能力也要逐步锻炼提高。另外，编程讲究逻辑，逻辑清晰才不会导致代码混乱，当然，编程也离不开不断的调试与修正，才能得出更加优化的代码。

总结：本次实验对加、减、乘、取模、取余、乘方、括号等操作符和操作数组成的算术表达式算法进行了研究，通过数据结构栈分析并实现了运算符的优先关系和浮点数转换为整型数值的计算，最终实现了中缀表达式求值。不足的是，对于运算过程中出现负数，以及表达式非法的情况考虑还不够周全，还需进一步思考，优化完善程序。