

《数据结构》上机实验报告

第 2 次上机

学号： 202011140104

姓名： 李馨

学院： 物理学系

专业： 物理学

教师： 郑新

日期： 2022.9.16

1. 实验要求

1.上机之前应做好充分准备，认真思考所需的上机题目，提高上机效率。

2.独立上机输入和调试自己所编的程序，切忌抄袭、拷贝他人程序。

3.上机结束后，整理出实验报告。书写报告时，重点放在实验的方法、思路以及总结反思上，以达到巩固课堂学习、提高动手能力的目的。

1. 实验过程
   1. 问题描述：合理运用栈，按照教材中的运算优先级，编程实现任意中缀算术表达式（可以只包含’+’、’-‘、’\*’、’/’等双目运算符、小括号和结束符）的求值运算。
   2. 问题分析与关键代码
      1. **中缀表达式转化为后缀表达式（**Infix\_to\_Postfix.cpp**）**

直接使用中缀表达式计算，需要用到两个栈：操作符栈OPTR (operator)，操作数栈OPND(operand)。使用后缀表达式计算时需要一个操作数栈。这里我们考虑先将前缀表达式转化为后缀表达式，然后利用后缀表达式计算，这一步也需要用到一个栈。

* + - 1. 对中缀表达式进行扫描，读到一个操作数时，将之输出到结果字符串中；读到操作符时不立即输出，根据情况存到栈中，等待输出处理。
         1. 当读到左括号时，无论何时都直接存到栈中，无需其他操作。

左括号当且仅当处理右括号时会弹出，不会输出。

* + - * 1. 当读到右括号时，不将之存到栈中（意味着不会被输出），将栈顶元素弹出并输出，直到弹完左括号，左括号弹出但不输出。
        2. 当读到其他操作符时，暂时不将之存到栈中，将栈顶优先级更高的元素弹出并输出，直到遇到优先级比该读入字符低的元素；如果遇到和读入字符相同的操作符，也进行退栈和输出操作。退栈完成后将此操作符压入栈中。（这意味着栈中双目运算符的优先级从底到顶是递增的。）
        3. 如果读到输入的中缀表达式的末尾，将栈顶元素弹出并输出，直到变成空栈。

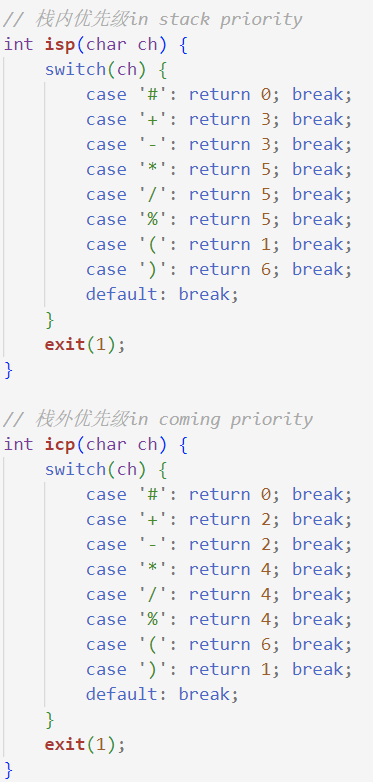
为了判断表达式末尾我们设立结束符“#”。

读入’#’时进行的操作与读到右括号进行的操作（括号配对前不断退栈输出）有相似之处，为统一处理，初始时在栈底压入“#”。

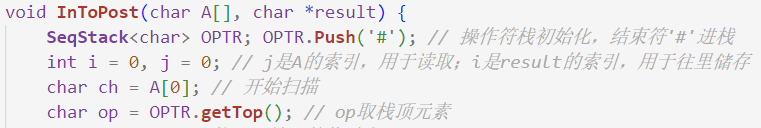
* + - 1. 为实现上述算法的运行，我们设立如下所示的算术操作符的栈外优先级（icp）和栈内优先级（isp）：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | # | + - | \* / | ( | ) |
| icp | 0 | 2 | 4 | 6 | 1 |
| isp | 0 | 3 | 5 | 1 | 6 |

* + - * 1. 为了处理1(a)和1(b)的情况，将左括号的栈外优先级icp设得足够高，右括号栈外优先级icp足够低；为了处理栈外双目操作符与栈内相同时退栈输出的情况，双目运算符isp都大于icp。
        2. 操作符优先数相等的情况只出现在括号配对或栈底的“#”号与输入流最后的“#”号配对时。
      1. 返回优先级的函数：

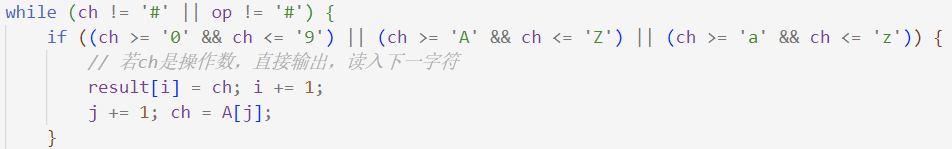


* + - 1. 算法实现：
         1. 操作符栈OPTR初始化，将结束符“#”进栈，op取栈顶元素。然后读入中缀表达式字符串的首字符ch。



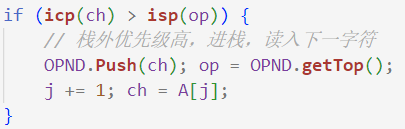
* + - * 1. 重复执行以下步骤，直到ch = ‘#’，同时栈顶操作符也为‘#’，停止循环。

若ch是操作数，直接输出，读入下一个字符ch。

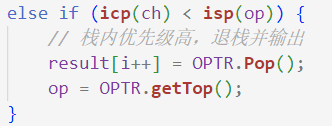


若ch是操作符，判断icp(ch)和isp(op)：

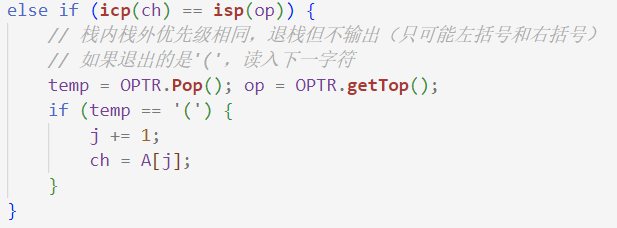
若 icp (ch) > isp (op)，令ch进栈，读入下一个字符ch。



若 icp (ch) < isp (op)，退栈并输出。



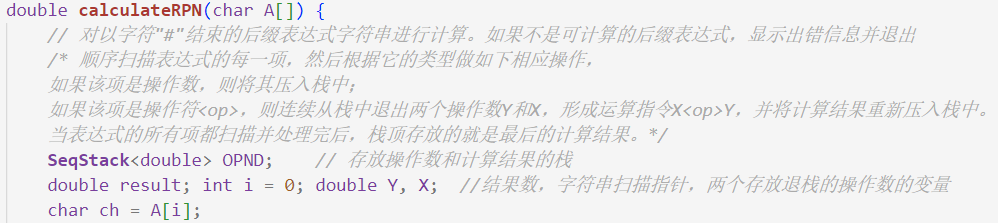
若 icp (ch) == isp (op)，退栈但不输出，若退出的是“(”号读入下一个字符ch。



为了用输出的字符串计算后缀表达式，在输出的字符串末尾加上‘#’。



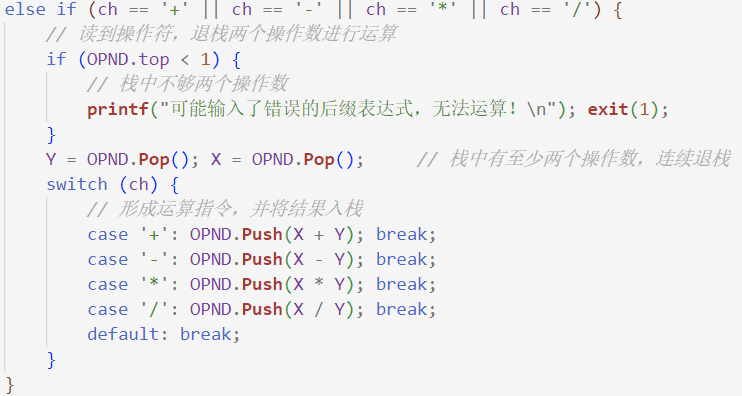
* + 1. **利用栈对后缀表达式求值（**calculateRPN.cpp**）**
       1. 初始化操作数栈OPND，开始用ch扫描输入的字符串A。



* + - 1. 顺序扫描后缀表达式（以“#”结尾）的每一项，然后根据它的类型做如下相应操作，
         1. 如果该项是操作数，则将其压入栈中；



* + - * 1. 如果该项是操作符<op>，则连续从栈中退出两个操作数Y和X，形成运算指令X<op>Y，并将计算结果重新压入栈中。



* + - * 1. 当表达式的所有项都扫描并处理完后（即扫描到“#”），栈顶存放的就是最后的计算结果。



* 1. 运行结果（main.cpp）
     1. 测试用中缀表达式：

char A[] = "1+2\*(3-4)-5/6#";

char B[] = "2-3+4\*5/(6+7)#";

* + 1. 分别转化为后缀表达式后进行计算，注释为预期的输出结果，实际运行结果与预期相符：



三、总结（实验中遇到的问题、取得的经验、感想等）

①实验中遇到了同时需要以double为数据类型和char为数据类型的两个栈（操作数栈和操作符栈），对于同一个结构体通过两个def实现这一点是不行的，SelemType只能是double或char其中之一的别名。如果定义两个结构体，则函数又需要都重写一遍，且不能重名。最后通过使用c++的class和template解决了这一问题。

②实验中加深了对栈的把握和印象，对其应用有了更深刻的感受。它在往往用于数据的暂时储存。