计算器的实现设计思路和测试说明

October 26, 2021

1. 项目设计思路:

   程序的实现主要分为以下几个步骤:

   1. **无效字符的过滤**：将输入的字符串除数字、小数点、加减乘除符号和括号外的符号过滤，形成新的字符串。这样可以方便之后的所有运算。
   2. **数字和操作符的分离**：在这一步中，我们希望最后能得到一个存储在vector<string>中的中缀表达式。由于一个中缀表达式每个数和其他符号是分离的，因此我们需要在第一步过滤后的字符串中提取出数字(和小数点)作为一个整体的数存储，而不是一个个拆散的字符。本程序中通过在每个数字起点后截取第一个非数字(和小数点)前的字符串来实现。**特别地**，对于负数前面的负号，程序通过判断条件将其和普通的减号区分开来，把负号看作数的一部分。
   3. **中缀到后缀的转换**：这一步骤是利用stack实现的。具体的实现方法在课本中已经有非常详细的说明，本程序完成了其代码实现。值得注意的是，本程序使用了priLevel()函数给每个符号赋予了不同的优先级，并通过给"("最高的优先级和")"最低的优先级来巧妙地解决了括号的输入输出问题。最后的结果存储在vector<string>中。
   4. **后缀表达式的求值**：同样地，课本中也有对这一步详细的算法说明。每次遇到加减乘除符号时，程序都将栈顶的两个元素弹出并进行相应的运算。最后栈中存储的就是运算的结果。
   5. **中缀表达式合法性的判断：这一步在程序里并没有作为独立的过程实现，而是在第2-4步中对表达式的合法性进行判断**。下面将叙述程序是如何识别不合法的表达式的。

      I 数的不合法：数由小数点和数字构成。数的不合法主要是排除数内部出现多个小数点，以及小数点出现在数的首尾两端的情况。这些都在第二步截取数的时候进行了判断。

      II 操作符(含括号，下同)之间的不合法：这一类的不合法现象主要涉及括号的匹配问题，以及多个操作符连续出现的问题。括号匹

配在中缀转后缀的过程中得到判断(因为不合法的括号会找不到匹配的括号或者留在栈中)。多个操作符连续出现在第四步中得到判断。因为不合法的操作符会使得栈中没有足够的数进行运算。

III 操作符与数的结合不合法：这一类不合法现象主要有除数为0，以及左(右)括号的左(右)侧是数的不合法表达。它们分别在第四步和第二步得到了判断。

2．测试说明：

1．测试输入：

I 运行 make 命令，自动编译 main.cpp 并生成可执行文件 test；

II 运行 bash run 命令即可；

III **测试数据的说明**：测试的数据都放在"input.txt"中，总共有二十行(即二十个算数表达式)。其中前四行的数据与老师给出的输入输出样例对应，后十六行数据针对各种复杂的运算和各种可能报错的情况进行测试。**为了便于验算，表达式的长度都不算太长，测试者可以自行修改。**

2．测试输出：

I 测试输出在"output.txt"中，每一行对应着输入的一个结果。经验算，程序计算的结果都是正确的。