1 expression 函数的设计思路

代码中首先设计了整体的 Calculator 类，负责封装整个运算流程的相关函数。在之中又嵌套了 ExprTree 类来实现对中缀表达式的处理，中缀表达式处理操作主要如下：

将表达式拆分成树状结构，根据表达式树的性质，自下往上计算，操作数为叶子节点任意节点的优先级小于等于其子节点的优先级，优先级从高到低依次为: 操作数; 乘除; 加、减。

所以在对树进行插入操作时，如果表达式树为空则可直接将当前节点作为根节点。如果表达树不为空，则需要考虑优先级与上一节点的关系了。如果优先级更大，则这一节点作为上一个节点的右子节点。如果优先级小，则从上一个节点沿路径向根节点回溯，找到第一个优先级小于当前节点的节点，再将上一个节点作为当前节点的左子节点。

如果有括号则可以先计算括号内子表达式的结果再将其作为一个数插入树中。

按照要求本项目也考虑了非法输入的各种情况，通过 if 的结构对其中可能出现的非法情况进行列举，最终打印 ILLEGAL 并 throw 出来。

另外本运算器还设计了对负数的运算操作，如果当前字符是 - 且它是表达式的第一个字符，或者它的前一个字符是左括号，则认为它是负号而不是减号。最后我们会将负数作为一个完整的字符串进行操作。

2 测试程序的设计思路

测试程序我们考虑了许多情况，首先是对运算结果的计算。随机设计了一些具有括号的较复杂的中缀运算式以及较为简单的运算式作为验证的例子。

其次我们验证了负数的合法输入以及正确计算结果的功能。

最后就是设计各种非法输入来检验运算器对非法表达式的识别。

整体而言表现是好的。设计结果如图所示

```
12      expr = "2*(1+3)-5*(15.23)/(1+2)*3-5";
13      cout << "input1:  " << expr << endl;
14      cal.parse(expr);
15      cout << "output1: ";
16      cal.solve();
17
18      expr = "1.25+(3*(1+2)*3-43)/(4-2)";
19      cout << "input2:  " << expr << endl;
20      cal.parse(expr);
21      cout << "output2: ";
22      cal.solve();
23
24      expr = "-1.25+(-3*(1+2)*3-43*(-0.5))/(4-2)";
25      cout << "input3:  " << expr << endl;
26      cal.parse(expr);
27      cout << "output3: ";
28      cal.solve();
29
30      expr = "2*(1+3))-5*(15.23)/(1+2)*3-5";
31      cout << "input4:  " << expr << endl;
32      cal.parse(expr);
33      cout << "output4: ";
34      cal.solve();
35
36      expr = "2*(1+3)-5*(15.23)/(1-1)*3-5";
37      cout << "input5:  " << expr << endl;
38      cal.parse(expr);
39      cout << "output5: ";
40      cal.solve();
41
42      expr = "2*(1+3)-5*(15.23)(1+2)*3-5";
43      cout << "input6:  " << expr << endl;
44      cal.parse(expr);
45      cout << "output6: ";
46      cal.solve();
47
48      expr = "2*(1+3)()-5*(15.23)/(1+2)*3-5";
49      cout << "input7:  " << expr << endl;
50      cal.parse(expr);
51      cout << "output7: ";
52      cal.solve();
53
54      expr = "2*(1+3c)-5*(15.2z3)/(1+x2)*3-5";
55      cout << "input8:  " << expr << endl;
56      cal.parse(expr);
57      cout << "output8: ";
58      cal.solve();
59
60      expr = "2*(1+3)-5*(15.2.3)/(1+2)*3-5";
61      cout << "input9:  " << expr << endl;
62      cal.parse(expr);
63      cout << "output9: ";
64      cal.solve();
65
66      expr = "2*(1+3)-5*(15.23)/(1+2)+*3-5";
67      cout << "input10:  " << expr << endl;
68      cal.parse(expr);
69      cout << "output10: ";
70      cal.solve();
```

图 1: output

3 测试的结果

测试结果如图所示。非法表达式全部输出 ILLEGAL，合法表达式全部输出正确结果，我已用计算器进行过验证。

```
danbao@danbao-virtual-machine:~/Desktop/DataStructures/project$ ./test
input1:  2*(1+3)-5*(15.23)/(1+2)*3-5
output1: -73.15
input2:  1.25+(3*(1+2)*3-43)/(4-2)
output2: -6.75
input3:  -1.25+(-3*(1+2)*3-43*(-0.5))/(4-2)
output3: -4
input4:  2*(1+3))-5*(15.23)/(1+2)*3-5
output4: ILLEGAL
input5:  2*(1+3)-5*(15.23)/(1-1)*3-5
output5: ILLEGAL
input6:  2*(1+3)-5*(15.23)(1+2)*3-5
output6: -225.45
input7:  2*(1+3)()-5*(15.23)/(1+2)*3-5
output7: ILLEGAL
input8:  2*(1+3c)-5*(15.2z3)/(1+x2)*3-5
output8: ILLEGAL
input9:  2*(1+3)-5*(15.2.3)/(1+2)*3-5
output9: ILLEGAL
input10:  2*(1+3)-5*(15.23)/(1+2)+*3-5
output10: ILLEGAL
```

图 2: output

4 bug 报告

一切正常。