项目作业：四则混合运算器设计报告

王珩宁 3230104148

2024/11/10

1 项目简介

我们这个项目实现了一个基于 C++ 的四则运算表达式求值程序，支持中缀表达式的计算，包括加法、减法、乘法、除法和括号的多层嵌套。此外，程序支持输入合法性检测并输出结果，非法输入返回 `ILLEGAL`。

2 设计思路

2.1 总体框架

程序的核心由以下模块构成：

- **表达式解析**：将输入表达式逐字符解析为操作数和操作符。
- **运算模块**：通过栈结构计算表达式的值，支持操作符优先级。
- **合法性检测模块**：确保输入表达式的格式正确，包括括号匹配、科学计数法支持、操作符的合理性等。
- **交互模块**：支持用户输入表达式和运行测试集。

处理数字和运算符：

- `isValidNumber` 函数用于验证数字的格式是否有效，确保数字中只有一个小数点和一个科学计数法标志。
- `processNumber` 函数用于从表达式中提取数字,并将其转换为 `double` 类型后存入 `values` 栈中。
- `processOperator` 函数用于处理运算符，根据运算符的优先级进行计算，并将结果存入 `values` 栈中。

处理括号：

- isMatchingParenthesis 函数用于检查括号是否匹配。
- processLeftParenthesis 函数处理左括号，将其压入 ops 栈中。
- processRightParenthesis 函数处理右括号，执行相应的计算直到匹配的左括号，然后将左括号弹出。

解析表达式：

- parseExpression 函数是整个表达式计算的核心，它遍历表达式并根据数字、运算符和括号的不同情况调用相应的处理函数。
- 在解析表达式时，会处理空格、数字、括号和运算符，并根据情况调整 expectOperator 和 isNegative 标志位。

执行计算：

- compute 函数执行实际的计算操作，根据栈顶两个操作数和当前运算符进行计算，并将结果存回 values 栈中。
- computeResult 函数循环执行计算，直到所有运算符都被处理完毕，最终将结果存入 result 中。

清空操作数栈和操作符栈：

- clear 函数用于清空 values 和 ops 栈，以便下一次表达式计算。

2.2 运算模块

伪代码如下

```
函数 parseExpression(字符串 expression):
初始化 expectOperator = isNegative = F
遍历字符串:
    如果是空格:
        跳过
    如果是数字或小数点:
        调用 processNumber 对数字进行tokenize
        如果失败:
            报错
        设置 expectOperator = T
    如果是左括号:
        调用直接入栈
    如果是右括号:
```

```
            调用一直出栈直到遇到左括号
            如果失败:
                报错
            设置 expectOperator = T
        如果是运算符:
            调用 processOperator
            如果失败:
                报错
        否则:
            报错
    返回 T
```

3 测试结果分析

3.1 测试方法

通过文件 test_cases.txt 我提供了足量多测试用例，覆盖了边界条件，基本运算，可行性和各类边界条件等。程序将运行并验证计算结果是否正确。以下为部分测试用例。

测试编号	测试用例	期望结果	实际结果
1	3+5	8	8
2	(2+3)*4	20	20
3	1e2+3	103	103
4	(3	ILLEGAL	ILLEGAL
5	2+*3	ILLEGAL	ILLEGAL
6	1/0	ILLEGAL	ILLEGAL
7	5.2e1+3.4	53.4	53.4
8	-2.5e-2+4.1	4.075	4.075
9	(1+2)*(3-4)	-3	-3
10	3++4	ILLEGAL	ILLEGAL

表 1: 部分测试用例及其结果分析

3.2 测试覆盖情况

- 支持的表达式类型：四则运算、括号嵌套运算、科学计数法、小数、负数。
- 非法表达式检测：括号不匹配、连续操作符、不合法数字格式。

4 不足和疑点

- 1e308*1e-308 这种情况下会返回 false，但是按理来说应该能做到计算出 1，可能是 double 的性质比较奇怪，但是我暂时没有找到解决办法
- 当前实现未对极端情况（如超长表达式）进行性能优化。本程序为了全面的报错和非法处理，在效率上有所让步。
- 在测试用例时，检验答案在字符串意义上的正确性，和在数学意义上的正确性各有优缺点，这里只用了前者，后者可能对无穷有理数的检查更准确
- 无法处理超长的数，思路上，应该可以通过把浮点数变成三个可拓展的 vector<int> 来解决，但是我在试图实现除法的时候发现太难了就搁置了。由 1. 计算机输入的小数一定有限，所以一定有理数，2. 只有 +-*/没有根号。可得计算机的结果一定是有理数，或许可以想办法用整数 (vector<short>，一个数组表示一条整数) 之商来表示，更精确。但是加减乘除法会变成分式运算。但是能解决 double 的量程问题，且不会发生浮点数导致的 underflow 和 overflow。恳请助教老师指点和建议。