1 计算器的基本设计思路

我们使用两个栈来处理表达式的计算，一个用来存放数，记为 nums，另一个用来存放运算符，记为 ops。从前往后依次处理字符：若为当前位为数字，则继续向后读取字符组成数字，途中若判断非法则返回 false；若为运算符，则将当前 ops 中所有运算优先级大于等于当前运算符的运算符与相应的两个数取出、运算后放回栈中，并将该运算符放入栈中。对于括号，我使用了一个变量 cnt 记录当前应配对的左括号的个数。若为左括号，则 cnt++；若为右括号，先判断当前 cnt 是否为 0，若为 0 则表达式非法，否则需要不断进行计算操作直到 ops 弹出第一个左括号。

2 合法表达式的判别

本程序的合法表达式规则同项目要求一致，能够识别识别非法的表达式，如括号不匹配、运算符连续使用、表达式以运算符开头或结尾以及除数是 0 等。同时，本程序考虑了负数，比如：1+-2.1, 1——-2 是合法的，但 1++2.1 是非法的。具体的处理方法是当遇见连续的运算符时判断后者是否为-，若是则视为将后面的数乘以-1 并 continue。对于表达式开头的-号，我选择的是在 eval 最开头进行判断，若为-号则在 nums 和 ops 中分别添加 0，'+'。本程序还考虑了科学计数法，比如：-1+2e2 是合法的，但 11e2 与 1e-2.2 是非法的。

3 Expression 类

本程序设计了 Expression 类用于处理输入与计算。Expression 类要求输入一个字符串进行初始化，并且出于处理方便起见，在初始化时会将字符串中的空格全部删除。Expression 类有 eval() 和 get_result() 两个 public 类函数。eval() 会对表达式进行运算，若合法则会将结果保存在成员变量 result 中并返回 true，否则返回 false。get_result 则可返回当前变量中存储的 result。

eval 函数中最复杂的地方为处理操作数的读取，对应 _compose_num 函数。因为表达式总共只需扫描一遍，整个 _compose_num 函数与其辅助函数共享一个指针 i，停留在每一步读取数的最后一位处。我们可以先读取整数部分，若后续存在. 与 e 则进一步进行处理。大致的流程为依次读取整数部分-> 若 i+1 处为小数点，则读取小数部分-> 若 i+1 处为 e，则读取科学计数法。同时，因为需要返回是否合法，该函数即其辅助函数均需要传入指针以保存结果。

4 检验的设计思路

据我所知，C++ 的标准库中不存在像 python 那样直接解析字符串的函数，所以在不借助其他库的情况下应该无法生成随机表达式大规模检测程序的正确性。因此本测试程序仅通过手动输入表达式与其预期结果来检测程序的正确性。具体来说，我分别对合法的表达式和非法的表达式进行了测试。程序的最后会输出成功的测试点与失败的测试点的个数。本想借助 LLM 扩充一下测试点范围，但 chatgpt 似乎并没有办法很好地推理四则运算，遂作罢。两种可以选择的改进方式是: 1. 借用三方库 eval 字符串。2. 利用 python 在本地生成测试点与正确结果，并在测试程序中读入并比较结果。

5 实验结果

在最外层目录输入 make run 可以运行计算器。输入 make test 可以进行测试。测试点全部通过。(浮点数计算的误差设置在了 1e-9 内)。