1 设计思路

关于 expression_evaluator 的设计思路：

我首先设计了一个类 ExpressionEvaluator。其中设置 evaluator 为类的公共借口，用来评估传入的中缀表达式；设置私有成员函数 parseExpression 用来解析加法和减法运算，parseTerm 用来解析乘法和除法运算，parseFactor 用来解析数字和括号，parseNumber 用来解析整数、小数和科学计数法，skipWhitespace 用来跳过表达式中的空白字符。

关于 main 的设计思路：

首先要对输入表达式进行处理，通过初始化 pos 为 0，调用 parseExpression 方法解析表达式并计算结果，调用 skipWhitespace 去掉空白字符，检查是否解析到了表达式的末尾，如果没有，抛出异常"ILLEGAL"。
我利用 parseExpression 方法解析加法和减法运算，调用 parseTerm 解析第一个项，进入循环并跳过空白字符，并通过判断是否到达表达式末尾和当前字符 op 是否是 + 和 − 来结束循环，移动到下一个字符同理解析下一个项，并且根据 op 更新结果 result。

再利用 parseTerm 方法解析乘法和除法运算，调用 parseFactor 解析第一个因子，进入循环并跳过空白字符，并通过判断是否到达表达式末尾和当前字符 op 是否是 ∗ 和 / 来结束循环，移动到下一个字符同理解析下一个项，并且根据 op 更新结果 result。

再利用 parseFactor 方法解析数字、正负号和括号。跳过空白字符，检查是否到达表达式末尾，如果是，抛出异常"ILLEGAL"；如果当前字符是左括号，解析括号内的表达式，并检查是否有匹配的右括号。如果当前字符是 + 或 −，解析下一个因子，并根据符号返回正或负的因子值。否则，调用 parseNumber 方法解析数字。

接着利用 parseNumber 方法解析整数、小数和科学计数法。跳过空白字符，记录起始位置 start，检查是否有正负号。然后解析数字和小数点，如果有多个小数点，抛出异常"ILLEGAL"。然后解析科学计数法的指数部分，如果格式不正确，抛出异常"ILLEGAL"。如果没有解析到任何数字，抛出异常"ILLEGAL"。最后返回解析的数字。
最后利用 skipWhitespace 方法跳过空白字符。

2 测试的结果

测试结果一切正常。详细的结果见照片。

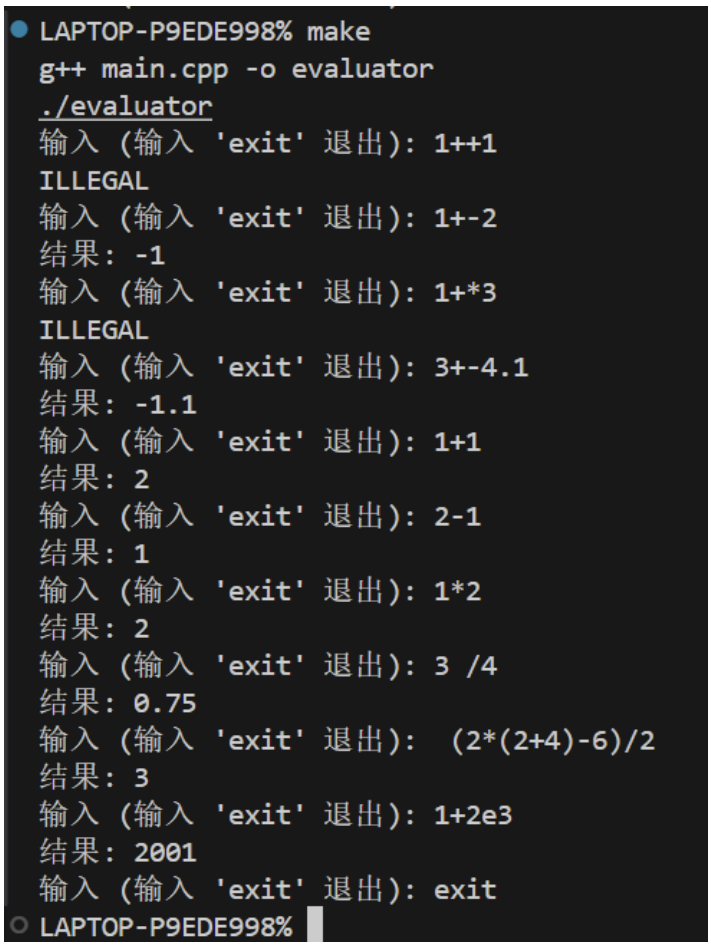
LAPTOP-P9EDE998% make
g++ main.cpp -o evaluator
./evaluator
输入 (输入 'exit' 退出): 1++1
ILLEGAL
输入 (输入 'exit' 退出): 1+-2
结果: -1
输入 (输入 'exit' 退出): 1+*3
ILLEGAL
输入 (输入 'exit' 退出): 3+-4.1
结果: -1.1
输入 (输入 'exit' 退出): 1+1
结果: 2
输入 (输入 'exit' 退出): 2-1
结果: 1
输入 (输入 'exit' 退出): 1*2
结果: 2
输入 (输入 'exit' 退出): 3 /4
结果: 0.75
输入 (输入 'exit' 退出): (2*(2+4)-6)/2
结果: 3
输入 (输入 'exit' 退出): 1+2e3
结果: 2001
输入 (输入 'exit' 退出): exit
LAPTOP-P9EDE998%

图 1: Test