1 项目背景与目标

本项目旨在设计并实现一个程序，用于计算包含加法、减法、乘法、除法和括号的中缀表达式。用户可以输入一个合法的数学表达式，程序会输出计算结果。如果输入的表达式不合法，程序将输出"ILLEGAL"并给出错误提示。

2 项目要求

项目需要满足以下要求：

- 支持加法、减法、乘法、除法运算。
- 支持多重括号及运算符优先级。
- 支持有限位小数的运算。
- 支持负数的运算。
- 识别非法表达式，如括号不匹配、运算符连续使用、除数为零等。
- 输出非法表达式的错误信息，提示具体的错误类型。

3 设计与实现

本程序使用C++语言实现。下面详细介绍了各个功能模块的实现过程。

3.1 前置声明与使用的库

程序使用了以下C++标准库来实现核心功能：

- `include <iostream>`：用于输入输出操作。
- `include <stack>`：用于栈的数据结构，实现操作符和操作数的存储与处理。
- `include <sstream>`：用于字符串流的操作，主要用于解析数字。
- `include <cctype>`：用于字符处理，如检查字符是否为数字等。

3.2 运算符优先级的定义

程序中定义了一个函数 `precedence(char op)` 来返回操作符的优先级。优先级如下：

- 加法 (+) 和减法 (-) 的优先级为 1。
- 乘法 (*) 和除法 (/) 的优先级为 2。
- 左括号的优先级为 0。

该函数在比较当前操作符与栈中的操作符时用于决定是否需要将栈中的操作符先应用于操作数。

3.3 执行运算的函数

程序通过 `applyOp(int a, int b, char op)` 函数来执行加、减、乘、除操作。该函数根据给定的操作符执行相应的运算。例如：

- 如果操作符是 "+"，则执行加法：`a + b`。
- 如果操作符是 "-"，则执行减法：`a - b`。
- 如果操作符是 "*"，则执行乘法：`a * b`。
- 如果操作符是 "/"，则执行除法，并且当除数为 0 时抛出异常。

如果遇到除数为 0 的情况，程序会抛出异常，指示该操作无法执行。

3.4 表达式求值的核心逻辑

`evaluate(const string &expression)` 函数是程序的核心。该函数接受一个字符串类型的表达式，遍历其中的每个字符并按照中缀表达式的规则进行计算。具体步骤如下：

- **空格处理：** 如果当前字符是空格，程序会跳过它继续处理下一个字符。
- **数字处理：** 如果当前字符是数字，程序会继续读取字符直到构成一个完整的数字（包括整数和小数）。解析完成后，将数字压入 `values` 栈中。
- **左括号处理：** 如果遇到左括号 `(`，程序将其压入 `ops` 栈中，表示接下来需要计算的子表达式。
- **右括号处理：** 如果遇到右括号 `)`，程序会将栈中的操作符依次应用于操作数，直到遇到左括号为止。如果栈中没有匹配的左括号，程序会抛出异常。
- **操作符处理：** 如果遇到加法、减法、乘法、除法等操作符，程序会根据操作符的优先级决定是否先处理栈中的操作符。首先将栈中优先级较高的操作符应用于操作数，然后将当前操作符压入栈中。
- **非法字符检查：** 如果遇到非法字符（即不属于数字、空格、括号或四则运算符），程序会抛出异常。

3.5 解析剩余的操作符

在遍历完表达式后，可能栈中还存在未处理的操作符。这些操作符会按照优先级依次应用于操作数，直到栈为空。

3.6 错误处理

在程序运行过程中，如果栈中剩余的元素个数不为 1，表示表达式不合法，程序会抛出异常并输出"ILLEGAL+具体错误原因"。通过使用 `try-catch` 机制，程序能够捕获异常并正确处理非法表达式。

3.7 主函数 (main)

主函数首先测试了一些测试用例，再从用户输入读取一个表达式，然后调用 `evaluate` 函数计算结果。如果输入表达式合法，程序会输出计算结果；如果表达式不合法，程序会捕获异常并输出 `"ILLEGAL"`。

4 测试用例与结果

为了验证程序的正确性，我们设计了一系列测试用例。测试结果如下：

4.1 合法表达式

- 输入：$1+2*3$，输出：7
- 输入：$(1+2)*3$，输出：9
- 输入：$3+(4*5)$，输出：23
- 输入：$(10-3)/(2+1)$，输出：2.33333
- 输入：$1+-2.1$，输出：-1.1
- 输入：$3.5+2.1$，输出：5.6
- 输入：$4*-1.5$，输出：-1.5
- 输入：$10/2.5$，输出：4
- 输入：$1+-2.1*2$，输出：-3.2

4.2 非法表达式

- 输入：$1++2$，输出：ILLEGAL: 操作数不足
- 输入：$1+2*$，输出：ILLEGAL: 操作数不足
- 输入：$+1+2$，输出：ILLEGAL: 操作数不足
- 输入：$(1+2$，输出：ILLEGAL: 操作数不足
- 输入：$10/0$，输出：ILLEGAL: 除数不能为0
- 输入：$1.2+3.4.5$，输出：ILLEGAL: 小数格式错误
- 输入：$10/0.0$，输出：ILLEGAL: 除数不能为0
- 输入：$2*(3/0)$，输出：ILLEGAL: 除数不能为0

5 负数处理

代码中，我们加入了对负数的处理。

5.1 负号的判断

我们定义了一个函数 `isminus` 来判断当前字符是否表示负数。具体来说，负号 '-' 如果出现在表达式的起始位置，或者前一个字符是运算符（如加号 '+'、减号 '-'、乘号 '*'、除号 '/'）或左括号 '('，则该负号表示一个负数符号，而非减法操作符。

5.2 处理

当我们识别出负号时，压入一个零值，然后将负号当作减法运算符继续运算。

6 总结

通过本项目的实现，我们成功地设计了一个四则运算表达式求值程序。程序不仅能够处理基本的四则运算和多重括号，还能够识别并处理非法表达式，确保计算结果的准确性和程序的健壮性，以及还能处理负数。