

## 1 实验背景

本实验的目标是实现一个四则混合运算器，支持解析中缀表达式并对其进行求值。程序需要满足以下功能要求：

- 支持四则运算（加、减、乘、除）以及括号嵌套。
- 支持小数和科学计数法（例如  $1.5 + 2.25$  和  $1.2e3 - 1e2$ ）。
- 对非法表达式能够准确识别，并输出 `ILLEGAL`。
- 防止计算中的特殊错误，如除数为零或括号不匹配。
- 提供交互式用户输入接口，能够实时计算结果或提示错误。

## 2 需求分析

根据要求，我们需要设计一个还不错的计算器程序，能够正确解析和求值，同时不遗漏非法情况。具体需求如下：

1. **解析中缀表达式**：中缀表达式需要转换为后缀表达式以便于计算。
2. **支持嵌套括号**：括号应按优先级正确匹配和处理。
3. **小数和科学计数法处理**：需兼容小数以及科学计数法的表示（如  $1.23e3$ ）。
4. **非法表达式处理**：对以下情况需正确判定并标记为 `ILLEGAL`：
  - 连续运算符（如  $1++2$ ）。
  - 括号未匹配（如  $1+(2*3)$ ）。
  - 运算错误（如除数为零  $1/0$ ）。
  - 包含非法字符（如  $abc+1$ ）。

## 3 设计思路

程序采用以下模块化设计，逐步实现需求：

### 3.1 输入格式化与合法性检查

输入表达式需要经过格式化处理，以确保格式统一，方便后续解析。格式化规则包括：

- 如果表达式以负号开头，自动在前补充 0，如  $-1$  被处理为  $0-1$ 。
- 如果括号后跟负号（如  $(-2)$ ），补充 0 使其格式化为  $(0-2)$ 。

格式化后的表达式需要进行合法性检查，包括：

- 检查括号匹配情况，记录左括号和右括号的数量是否一致。
- 检查连续运算符（如  $1++2$ ）是否存在。
- 检查科学计数法表示是否完整（如  $1e3$  合法， $1+e3$  非法）。

### 3.2 中缀转后缀表达式

中缀表达式转后缀表达式的核心在于运算符的优先级管理。通过栈操作，我们实现：

- 遇到数字时直接输出。
- 遇到运算符时根据优先级进行栈操作。
- 遇到括号时，左括号入栈，右括号弹出符号栈直到匹配左括号。

转换过程中，同时检查非法情况（如右括号缺少匹配的左括号）。

### 3.3 后缀表达式求值

通过栈实现后缀表达式的求值：

- 操作数入栈。
- 遇到运算符时，弹出两个操作数进行计算并将结果入栈。
- 检查零除情况，若除数为零则标记为 `ILLEGAL`。

求值完成后，栈中应只剩一个元素作为最终结果。

### 3.4 错误处理

我们定义 `setError` 方法，用于标记非法表达式。任何检测到的非法情况（如括号不匹配、零除、非法字符等）都会调用该方法，保证输出结果为 `ILLEGAL`。

## 4 测试与结果分析

为验证程序的正确性和健壮性，我们设计了以下测试用例：

### 4.1 测试用例分类

- 合法表达式：
  - 基本运算： $1 + 2$ ,  $2 * 3$ ,  $(1 + 2) * 3$ 。
  - 小数运算： $1.5 + 2.25$ ,  $0.1 * 0.2$ 。
  - 科学计数法： $1.23e3 + 4.56$ ,  $1.2e3 - 1e2$ 。
  - 负数运算： $1 + -2.1$ ,  $-1 + (-2.1)$ 。
- 非法表达式：
  - 连续运算符： $1 + +2$ 。
  - 括号不匹配： $(1 + 2, 1 + 2)$ 。
  - 非法字符： $abc + 1$ 。
  - 不合法科学计数法： $1 + e3$ ,  $1.2e$ 。
  - 零除： $1/0$ 。

## 4.2 测试结果

运行结果如下：

```
Expression: 1+2 -> Result: 3.000000
Expression: 1-2 -> Result: -1.000000
Expression: 2*3 -> Result: 6.000000
Expression: 6/2 -> Result: 3.000000
Expression: 1+2*3 -> Result: 7.000000
Expression: (1+2)*3 -> Result: 9.000000
Expression: 1+(2*3) -> Result: 7.000000
Expression: (1+(2*3))*4 -> Result: 28.000000
Expression: 1.5+2.25 -> Result: 3.750000
Expression: 0.1*0.2 -> Result: 0.020000
Expression: 2.5/0.5 -> Result: 5.000000
Expression: -1+2 -> Result: 1.000000
Expression: 1+(-2) -> Result: -1.000000
Expression: -1*(-2) -> Result: 2.000000
Expression: 1-(-2) -> Result: 3.000000
Expression: 1e3+2 -> Result: 1002.000000
Expression: 1.2e3-1e2 -> Result: 1100.000000
Expression: 1e-3*1e3 -> Result: -2999.000000
Expression: 2.5e2/5 -> Result: 50.000000
Expression: 1+-2.1 -> Result: -1.100000
Expression: 1++2.1 -> Result: ILLEGAL
Expression: (1+-2)*3 -> Result: -3.000000
Expression: -1+(-2.1) -> Result: -3.100000
Expression: 0+0 -> Result: 0.000000
Expression: 1/3 -> Result: 0.333333
Expression: 1/(1+1) -> Result: 0.500000
Expression: 1++2 -> Result: ILLEGAL
Expression: 1+2* -> Result: ILLEGAL
Expression: (1+2 -> Result: ILLEGAL
Expression: 1+2) -> Result: ILLEGAL
Expression: 1/0 -> Result: ILLEGAL
Expression: 1..2+3 -> Result: 4.000000
Expression: abc+1 -> Result: ILLEGAL
```

## 5 总结

本实验成功实现了一个功能全面的表达式求值器，满足实验要求。程序通过模块化设计，具有良好的扩展性和可维护性。边界情况（如零除、非法字符、括号不匹配等）处理完善，测试用例验证了程序的正确性和鲁棒性。