项目报告：中缀表达式转后缀表达式计算器

罗俊勋

2023 年 10 月 28 日

1 项目介绍

这个项目是一个中缀表达式转后缀表达式并计算表达式值的计算器。它包括了三个关键文件：'main.cpp','stringPrepoce.h', 和'calculator.h'。下面是对项目的详细解释。

2 设计思路

在'main.cpp' 文件中, 我们首先读取命令行参数的中缀表达式, 并通过'stringPreproce' 函数进行预处理。预处理包括去除无关字符、检查括号和操作符的合法性。然后, 使用'strToQueue' 函数将预处理后的中缀表达式转换为后缀表达式队列, 同时处理负数的情况。最后, 使用'evalPostfix' 函数计算后缀表达式的值, 将结果输出到标准输出。

在'stringPrepoce.h' 文件中, 我们定义了一系列用于预处理中缀表达式的函数, 包括错误提示函数、判断字符是否为操作符的函数、检查括号合法性的函数、检查操作符合法性的函数等。这些函数协同工作以确保中缀表达式的合法性。

在'calculator.h' 文件中, 我们定义了用于计算后缀表达式的函数。这包括判断字符串是否为数字的函数和执行后缀表达式计算的函数。'evalPostfix' 函数遍历后缀表达式队列并使用双栈来执行计算。

2.1 main.cpp

'main.cpp' 是项目的入口文件, 负责接收用户输入的中缀表达式, 处理它并输出计算结果。以下是'main.cpp' 的关键设计思路:

- 引入必要的头文件:'iostream'、'stack'、'queue'、'string' 等, 以及自定义的'stringPreproce.h' 和'calculator.h', 以支持项目所需的功能。
- 实现了 op_Priority 函数, 用于判断操作符的优先级, 根据操作符的最后一个字符来确定其优先级。这是在处理操作符时非常重要的功能。
- 实现了'countDots' 函数, 用于统计字符串中小数点的个数。这对于检测小数点的合法性很有帮助。
- 主要逻辑包括将中缀表达式转化为后缀表达式, 然后计算后缀表达式的值, 并输出结果。
- 处理了中缀表达式中的数字、操作符和括号。使用队列'infixQueue' 来存储中缀表达式, 栈'opStack' 用于操作符的处理, 队列'Postfix' 用于存储后缀表达式。
- 通过多次遍历中缀表达式, 根据操作符的优先级将操作符转换为后缀表达式。最终, 计算后缀表达式的值并输出。

2.2 calculator.h

'calculator.h' 包含了计算后缀表达式的函数, 以及一些辅助函数。以下是'calculator.h' 的设计思路:

- 实现了'isnum' 函数, 用于判断一个字符串是否为数字, 这是后缀表达式计算中的重要判断条件。
- 实现了'evalPostfix' 函数, 该函数用于执行后缀表达式的计算。它使用一个操作数栈和遍历后缀表达式的方式来计算结果。

- 实现了'eval' 函数, 用于执行二元运算, 包括加法、减法、乘法和除法。根据操作符执行相应的运算, 并返回结果。
- 这些函数协同工作, 将后缀表达式队列的计算结果返回给'main.cpp', 并输出到标准输出。

2.3 stringPrepoce.h

'stringPrepoce.h' 包含了一系列用于中缀表达式预处理的函数。以下是'stringPrepoce.h'的设计思路:

- 实现了错误提示函数 err_exp, 用于在遇到非法表达式时输出错误信息并退出程序。
- 实现了一系列函数, 包括判断字符是否是操作符、四则运算符、小数点, 以及检查括号合法性和操作符合法性的函数。
- 定义了总判断函数 strPreproce, 它将中缀表达式进行预处理, 去除无关字符, 并检查括号和操作符的合法性。
- 实现了'strToQueue' 函数, 用于将预处理后的中缀表达式字符串转换为队列, 以便后续处理。它还处理了负数的情况, 确保负号与数字正确解析。
- 这些函数协同工作, 确保中缀表达式的合法性, 并将处理后的表达式传递给'main.cpp'。

3 测试说明

3.1 单项测试

- 如需测试输入"2+9*(9-7)"是否合法, 在终端中输入"./test "2+9*(9-7)" " 即可
- 如需大量测试, 将测试用例按行输入 input.txt 后运行"bash run", 即可在生成的 output.txt 文件中获得对应的输出