数据结构课程设计文档

题目：表达式转换

姓名：赵卓冰

学号：2252750

专业：软件工程

年级：2023 级

指导教师：张颖

2024 年 10 月 18 日

1. 运行环境与开发工具

本项目支持在以下开发环境和编译运行环境中运行：

- **Windows 操作系统**：
  - 版本：Windows 10 x64
  - IDE：Visual Studio 2022 (Debug 模式)
  - 编译器：MSVC 14.39.33519
- **Linux 操作系统**：
  - 版本：Ubuntu 20.04.6 LTS
  - IDE：VS Code
  - 编译器：gcc version 9.4.0 (Ubuntu 9.4.0-1ubuntu1~20.04.2)

2. 项目要求

本项目旨在实现一个表达式二叉树，支持从中缀表达式的输入转化为表达式树，并生成前缀表达式和后缀表达式。用户通过输入合法的中缀表达式，程序构建相应的二叉树并输出树结构和前序、后序遍历的结果。

2.1. 功能要求

1. **输入说明**：输入合法的中缀表达式，表达式可以包含括号和四则运算符（`+`, `-`, `*`, `/`），操作数可以是多位的正数。
2. **输出说明**：
    - 输出表达式二叉树的可视化结构。
    - 输出相应的前缀表达式和后缀表达式。
    - 若中缀表达式不合法，输出提示信息并结束程序。

3. 项目简介

本项目实现了一个表达式树，支持从中缀表达式输入构建二叉树，并生成对应的前缀和后缀表达式。项目的核心功能是根据用户输入的中缀表达式生成后缀表达式，随后使用后缀表达式构建二叉树，并对树进行遍历，最终输出树结构以及前缀和后缀表达式。

项目的主要功能包括：

- **中缀表达式输入**：支持多位数的正数及四则运算符和括号。
- **表达式树的构建**：基于后缀表达式的栈操作实现树的构造。
- **前缀、后缀表达式的输出**：通过对树进行前序和后序遍历生成相应表达式。
- **中缀表达式的合法性检查**：对输入的中缀表达式进行语法检查，确保操作数、操作符、括号匹配正确。

4. UI界面展示

4.1. Windows 界面

- **测试用例1**：

```
请输入中缀表达式:
2 + 3 * ( 7 - 4 ) + 8 / 4

表达式的二叉树为:
--------------------表达式的二叉树------------------------
                                +
                     +                     /
            2            *            8            4
                      3     -
                          7   4
----------------------------------------------------------

前缀表达式为:
+ + 2 * 3 - 7 4 / 8 4

后缀表达式为:
2 3 7 4 - * + 8 4 / +

按回车键退出...
```

- **测试用例2**：

```
请输入中缀表达式:
( ( 2 + 3 ) * 4 - ( 8 + 2 ) ) / 5

表达式的二叉树为:
--------------------表达式的二叉树------------------------
                                /
                    -                          5
           *            +
       +       4    8       2
   2   3
----------------------------------------------------------

前缀表达式为:
/ - * + 2 3 4 + 8 2 5

后缀表达式为:
2 3 + 4 * 8 2 + - 5 /

按回车键退出...
```

- **测试用例3**:

```
请输入中缀表达式:
1314 + 25.5 * 12

表达式的二叉树为:
--------------------表达式的二叉树----------------------
        +
     1314      *
            25.5    12
------------------------------------------------------------

前缀表达式为:
+ 1314 * 25.5 12

后缀表达式为:
1314 25.5 12 * +

按回车键退出...
```

- **测试用例4**:

```
请输入中缀表达式:
-2 * ( +3 )

表达式的二叉树为:
--------------------表达式的二叉树----------------------
    *
  -2  3
------------------------------------------------------------

前缀表达式为:
* -2 3

后缀表达式为:
-2 3 *

按回车键退出...
```

- **还可以支持带变量的表达式**：

```
请输入中缀表达式：
a + ( b - c / d ) - x * y

表达式的二叉树为：
--------------------表达式的二叉树------------------------
                                   -
                     +                      *
           a                 -                    x            y
                        b          /
                               c    d
----------------------------------------------------------

前缀表达式为：
- + a - b / c d * x y

后缀表达式为：
a b c d / - + x y * -

按回车键退出...
```

4.2. Linux 界面

```
bing@bing-virtual-machine:~$ cd ds
bing@bing-virtual-machine:~/ds$ g++ expressionConvert.cpp -o expressionConvert
bing@bing-virtual-machine:~/ds$ ./expressionConvert
请输入中缀表达式：
2 + 3 * ( 7 - 4 ) / 9

表达式的二叉树为：
--------------------表达式的二叉树------------------------
                                +
                     2                     /
                                     *            9
                                      3       -
                                          7   4
----------------------------------------------------------

前缀表达式为：
+ 2 / * 3 - 7 4 9

后缀表达式为：
2 3 7 4 - * 9 / +

按回车键退出...
bing@bing-virtual-machine:~/ds$
```

5. 代码架构

项目主要包含三个核心类：`String` 类、`Vector` 类以及 `ExpressionTree` 类，分别用于表示字符串、动态数组以及表达式树。

5.1. String 类

`String` 类用于存储和操作字符串。它支持字符串的动态内存分配、拼接、删除等常用操作。

- **成员函数**：
  - `Clear()`：清空字符串。
  - `Append()`：向字符串末尾添加字符。
  - `Erase()`：从指定位置开始删除字符。
  - `operator[]`：重载索引运算符，用于访问字符串中的字符。
  - `operator+=`：拼接字符串或字符。
  - `C_str()`：返回 C 风格字符串。

5.2. Vector 类

`Vector` 类是一个动态数组，用于存储元素并支持动态扩容。

- **主要方法**：
  - `PushBack()`：向数组末尾添加元素。
  - `PopBack()`：从数组末尾删除元素。
  - `Reverse()`：反转数组中的元素顺序。
  - `operator[]`：重载索引运算符，用于访问数组中的元素。

5.3. ExpressionTree 类

`ExpressionTree` 类用于构建表达式树并生成相应的前缀和后缀表达式。继承自 `BinaryTree` 类。

- **成员函数**：
  - `BuildFromPostfix()`：通过后缀表达式构建表达式二叉树。
  - `InfixToPostfix()`：将中缀表达式转化为后缀表达式。
  - `PreOrderTraversal()`：前序遍历表达式树，生成前缀表达式。
  - `PostOrderTraversal()`：后序遍历表达式树，生成后缀表达式。
  - `PrintTree()`：可视化输出表达式树结构。

5.4. 其他辅助类

- **TreeNode**：二叉树节点类，包含节点值以及左右子节点。
- **Stack**：栈类，用于处理表达式构建时的中间存储。
- **Queue**：队列类，用于层次遍历二叉树。

6. 功能模块介绍

6.1. 中缀表达式输入

用户通过输入中缀表达式，项目会将其转化为后缀表达式，并构建二叉树。支持的操作包括四则运算符 `+`, `-`, `*`, `/` 和括号。

6.2. 中缀转后缀表达式

`InfixToPostfix()` 方法实现了中缀表达式到后缀表达式的转换，使用了操作符优先级规则和栈来处理括号及操作符顺序。

6.3. 表达式树构建

通过 `BuildFromPostfix()` 方法，基于后缀表达式构造表达式树。栈用于保存子树结构，当遇到操作符时，弹出左右子树并将操作符作为新节点。

6.4. 前缀和后缀表达式输出

通过对表达式树进行前序和后序遍历，生成前缀和后缀表达式。

6.5. 表达式树的可视化输出

`PrintTree()` 方法使用广度优先搜索（BFS）算法输出表达式树的结构，方便用户查看树的整体结构。

7. 用户交互

用户在命令行中输入合法的中缀表达式，系统将进行以下操作：

1. 验证输入的中缀表达式是否合法（括号是否匹配、操作符与操作数是否正确等）。
2. 将中缀表达式转换为后缀表达式，并构建表达式二叉树。
3. 输出表达式树的结构。
4. 输出前缀表达式（通过前序遍历树）。
5. 输出后缀表达式（通过后序遍历树）。

7.1. 示例交互

- **输入**：

```
3 + ( 5 * 2 ) - 8 / 4
```

- **输出**：

```
表达式二叉树：
    +
   / \
  3   -
     / \
    *   /
   / \ / \
  5  2 8  4
前缀表达式：
+ 3 - * 5 2 / 8 4
后缀表达式：
3 5 2 * 8 4 / -
```

8. 性能考虑

表达式树的构建和遍历时间复杂度为 `O(n)`，其中 `n` 是表达式中的元素数量（操作数和操作符的数量）。由于中缀表达式的转换和树的构建涉及栈和队列操作，算法性能较为高效，适合处理中小规模的表达式。

9. 错误处理和输入验证

为了保证输入的合法性，项目对中缀表达式进行了严格的验证：

- **括号匹配检查**：确保每个左括号都有对应的右括号。
- **操作符与操作数检查**：检查操作符与操作数的数量关系，确保操作数不缺失，操作符不多余。
- **正号去除**：如果表达式中的操作数前有正号（例如 `+5`），程序会自动去除正号。

若检测到非法输入，程序会输出提示信息并结束运行。

10. 未来改进

1. **增加更多操作符支持**：目前项目仅支持基本的四则运算符，未来可以扩展支持指数、取模等运算。
2. **优化输出形式**：目前的树结构输出较为简单，可以通过图形化工具进一步优化树的展示效果。
3. **支持小数运算**：当前版本只支持整数运算，未来可扩展为支持浮点数计算。

4. **支持用户输入的表达式保存与加载**：将表达式树保存为文件格式，方便后续加载和进一步分析。

11. 心得体会

通过本项目的开发，我加深了对二叉树和栈、队列数据结构的理解，掌握了如何将中缀表达式转换为后缀表达式并通过后缀表达式构建表达式树。项目的核心挑战在于如何处理表达式的优先级和括号匹配，以及如何通过树的遍历生成前缀和后缀表达式。