**项目说明文档**

**数据结构课程设计**

**——表达式计算**

作 者 姓 名： 戴仁杰

学 号： 1951650

指 导 教 师： 张颖

学院、 专业： 软件学院 软件工程

同济大学

Tongji University

目 录

[1 分析 - 3 -](#_Toc91188725)

[1.1 背景分析 - 3 -](#_Toc91188726)

[1.2 功能分析 - 3 -](#_Toc91188727)

[2 设计 - 3 -](#_Toc91188728)

[2.1 数据结构设计 - 3 -](#_Toc91188729)

[2.2 类结构设计 - 3 -](#_Toc91188730)

[2.3 成员与操作设计 - 4 -](#_Toc91188731)

[2.3.1 Vector类 - 4 -](#_Toc91188732)

[2.3.2 Stack类 - 4 -](#_Toc91188733)

[2.3.3 BTreeNode类 - 5 -](#_Toc91188734)

[2.3.4 BTree类 - 5 -](#_Toc91188735)

[2.3.5 BTreeCalculateSystem类 - 6 -](#_Toc91188736)

[2.4 系统设计 - 6 -](#_Toc91188737)

[3 实现 - 6 -](#_Toc91188738)

[3.1 根据表达式建立二叉树功能的实现 - 6 -](#_Toc91188739)

[3.1.1 根据表达式建立二叉树功能流程图 - 6 -](#_Toc91188740)

[3.1.2 根据表达式建立二叉树功能核心代码 - 7 -](#_Toc91188741)

[3.2.1 总体功能流程图 - 8 -](#_Toc91188742)

[3.4.2 总体功能核心代码 - 8 -](#_Toc91188743)

[3.4.3 总体功能截屏示例 - 8 -](#_Toc91188744)

[4 测试 - 9 -](#_Toc91188745)

[4.1 功能测试 - 9 -](#_Toc91188746)

[4.1.1 不带括号表达式功能测试 - 9 -](#_Toc91188747)

[4.1.2 带括号表达式功能测试 - 9 -](#_Toc91188748)

[4.1.3 含小数表达式功能测试 - 9 -](#_Toc91188749)

[4.2 边界测试 - 9 -](#_Toc91188750)

[4.2.1 表达式中无运算符 - 9 -](#_Toc91188751)

[4.2.2 整个表达式被括号包围 - 10 -](#_Toc91188752)

[4.3 出错测试 - 10 -](#_Toc91188753)

[4.3.1 表达式左右括号不匹配 - 10 -](#_Toc91188754)

[4.3.2 表达式中出现未定义符号 - 10 -](#_Toc91188755)

[4.3.2 表达式中出现空括号 - 10 -](#_Toc91188756)

1 分析

1.1 背景分析

在日常生活中，我们经常会遇到需要计算算术表达式的情况，如果是遇到比较简单的算术表达式还好，但如果遇到的是比较复杂的表达式，那么我们必须要手动进行繁杂计算，并且还很有可能算错，这给日常生活带来了不便。因此，编写一个可以用来计算算术表达式的程序十分重要。

在计算机科学中，表达式求值是程序设计语言编译中的一个最基本问题，就是将一个表达式转化为逆波兰表达式并求值。而求值首先要完成对表达式的表达方式转换，转换为方便计算机理解的形式。具体要求是以字符序列的形式从终端输入表达式，并利用给定的优先关系实现对算术四则混合表达式转换成前缀、中缀、后缀形式。

1.2 功能分析

作为一个算术表达式转换的程序，要将输入中缀表达式转化为前缀表达式、后缀表达式。我们日常遇到与使用的表达式都为中缀表达式，即二元运算符处于两个运算数中间。这种表达式人读起来非常容易理解，但是用计算机计算中缀表达式的时候会出现符号优先级的问题，还会出现括号带来的问题，因此我们要将输入的中缀表达式转化为计算机容易理解的形式。

在本程序中，使用了栈和树的数据结构存储操作数和操作符，支持加减乘除，取余，乘方运算。还支持带有括号的表达式，以及运算数或运算结果中含有小数的表达式。

综上，为了实现表达式求值，本项目要求首先读入表达式（包括括号）并创建对应二叉树，其次对二叉树进行前序遍历，中序遍历，后续遍历，输出对应的波兰式，中序表达式和逆波兰表达式。

2 设计

2.1 数据结构设计

如上述功能分析所述，该表达式转换系统要求能够对输入的中缀表达式进行存储转换为前缀、中缀、后缀三种形式，对应二叉树的三种遍历结果。在转换时，用栈暂时存储输入的表达式，并建立起与之匹配的二叉树。之后对该二叉树进行前序、中序、后续遍历，即可得到波兰式、中缀表达式以及后缀表达式。

2.2 类结构设计

首先，本程序使用二叉树来保存表达式的信息，所以需要有一个二叉树（BTree）类，并与之配套一个二叉树结点（BTreeNode）类。其次，在建立表达式二叉树时，我们使用到了栈（Stack）类，本程序中是用Vector类实现的栈类。最后，还设计了一个二叉树运算系统（BTreeCalculateSystem）类，将表达式字符串的解析，计算，展示结果等功能整合在一起，更加方便管理。为了使向量类和栈类更具有泛用性，本程序将Vector类、Stack类、BTreeNode类和BTree类都设计为了模板类。

2.3 成员与操作设计

2.3.1 Vector类

Vector类为向量类，采用了类模板提高泛用性。类内实现了下标索引，插入元素，删除元素，获得数组大小等操作。通过动态内存申请调整大小，避免直接声明数组造成的空间浪费。



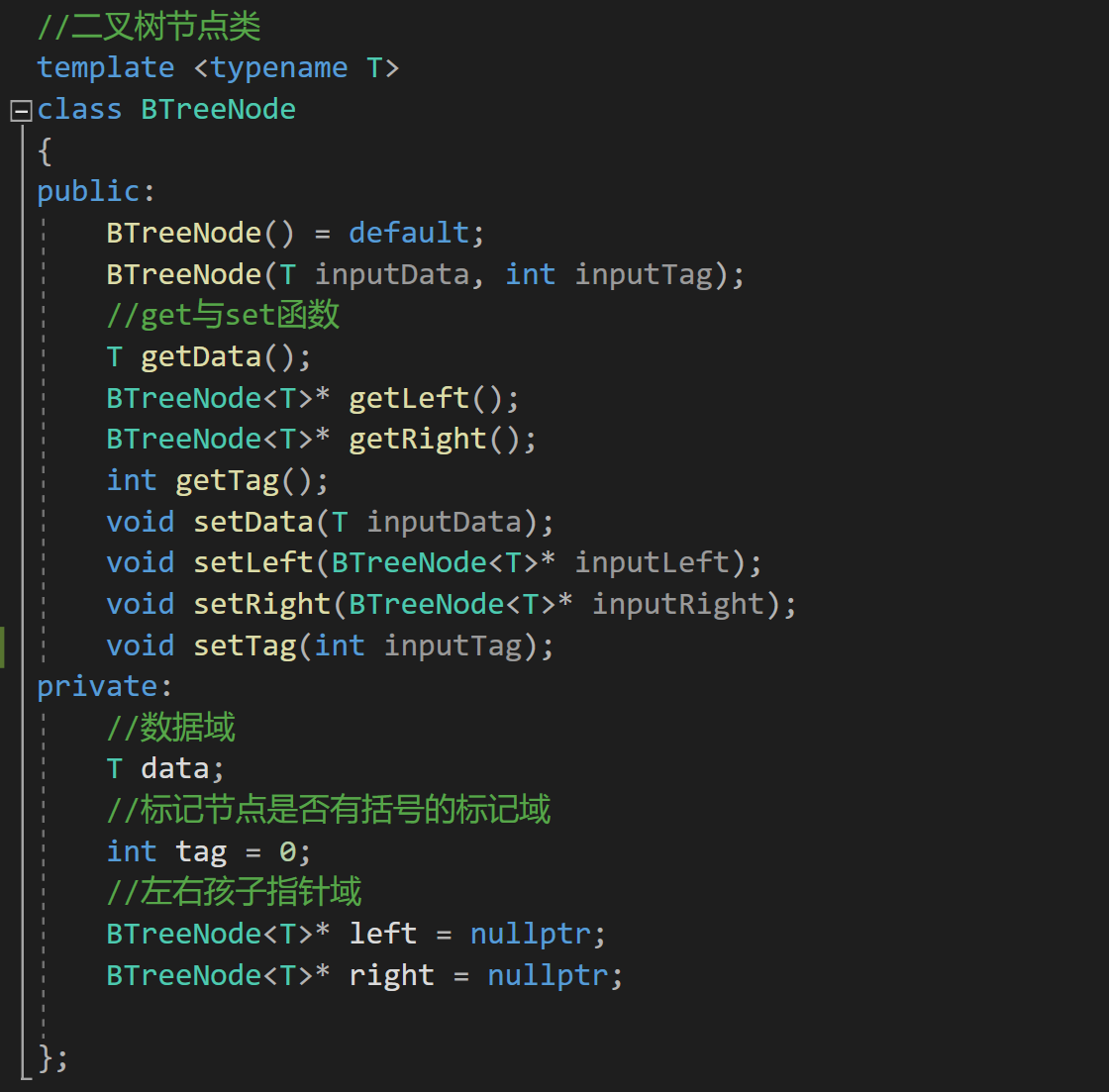
2.3.2 Stack类

Stack类为栈类，采用了类模板提高泛用性。类内实现了栈的增删元素，获得大小、获得栈顶元素、判断是否为空和清空栈操作。类内成员为一个Vector类型的向量数组元素。



2.3.3 BTreeNode类

BTreeNode类为二叉树结点类，采用了类模板提高泛用性。提供默认构造函数、带参构造函数以及与外接数据交流的一系列get与set函数。类内数据为data数据域、左右孩子指针以及标记结点是否有括号的标记值，存储了结点信息并与孩子结点建立联系。



2.3.4 BTree类

BTree类为二叉树类，采用了类模板提高泛用性。提供析构函数释放内存。提供插入左右孩子的函数、清空二叉树的函数。此外，提供了get和set系列函数与外界进行数据交流。类内数据为二叉树根节点。



2.3.5 BTreeCalculateSystem类

BTreeCalculateSystem类为二叉树表达式计算系统类。对二叉树类进行封装增强，丰富了交互提示信息，增强了输入错误、边界错误处理性能。类内封装了判断表达式优先级、判断表达式是否合法的函数、遇到各种符号的处理函数和输出表达式函数。类内数据为一个二叉树类型变量，用于保存表达式的操作符和操作数信息。



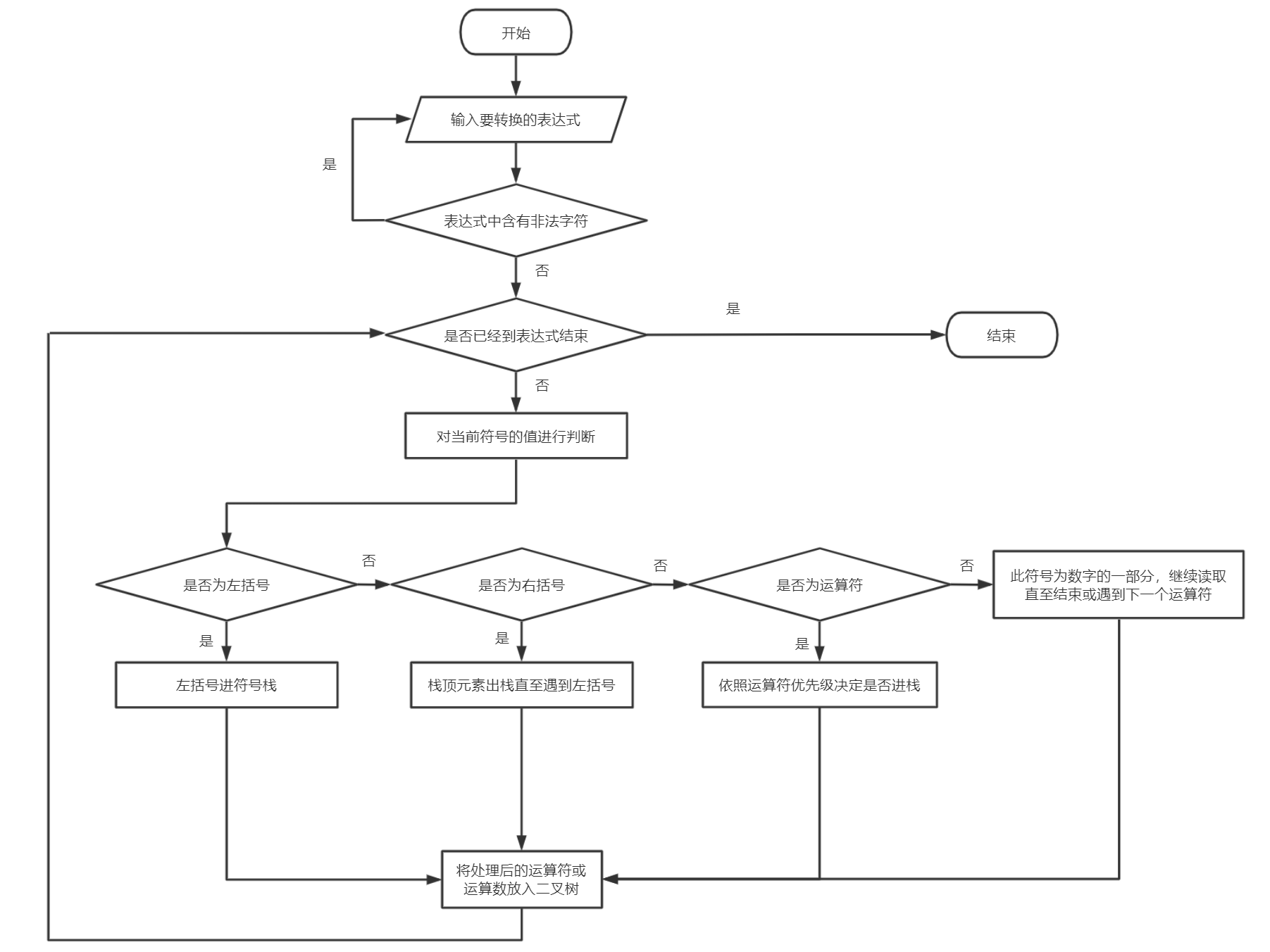
2.4 系统设计

程序在使用时，首先输入表达式字符串，然后对字符串进行解析，建立起二叉树表达式系统。之后对存储了表达式信息的二叉树进行前序、中序、后序遍历，如果表达式合法，则会返回并展示对应的表达式转换结果，如果计算失败则会显示错误提示。

3 实现

## 3.1 根据表达式建立二叉树功能的实现

### 3.1.1 根据表达式建立二叉树功能流程图

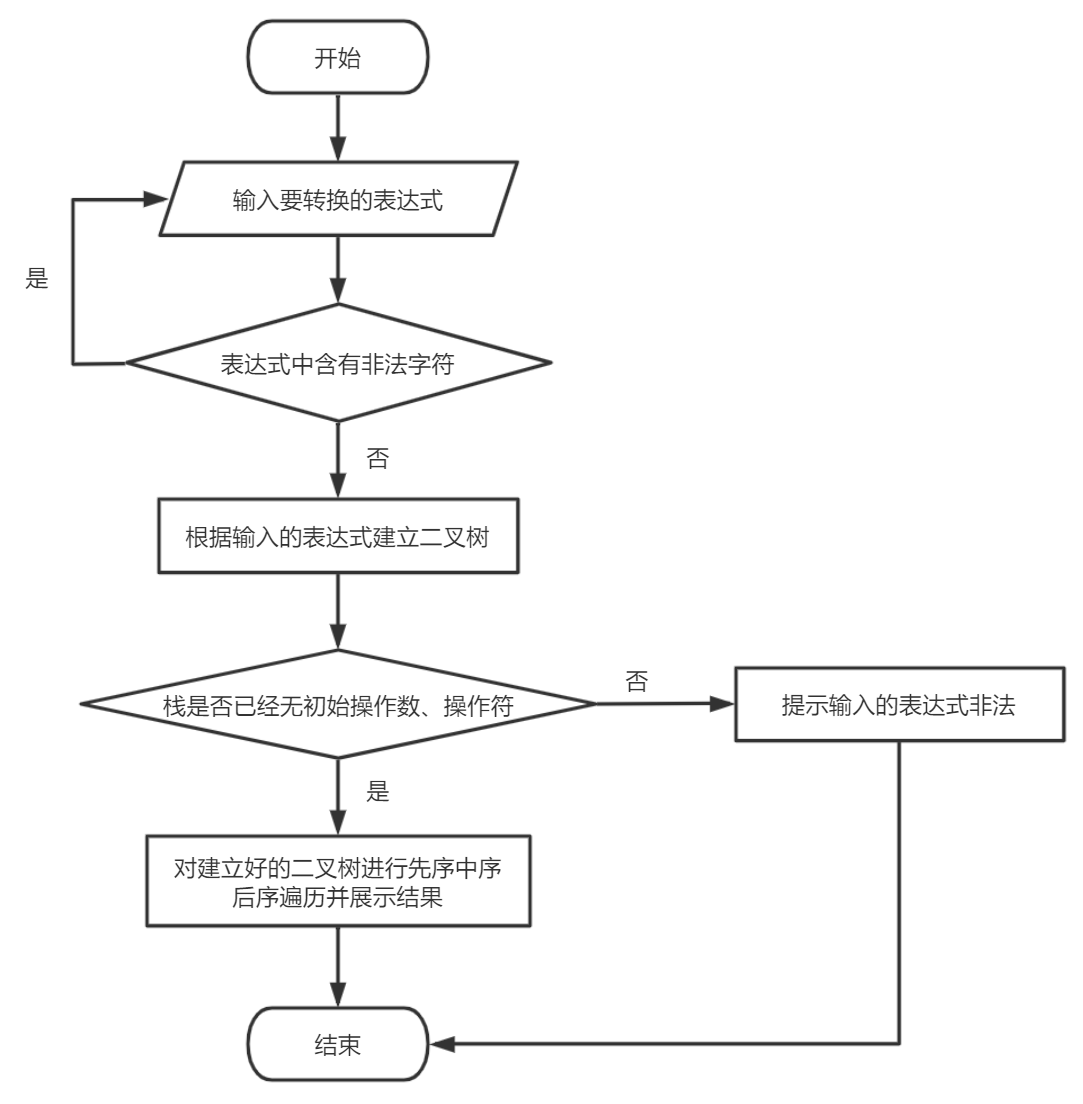


### 3.1.2 根据表达式建立二叉树功能核心代码

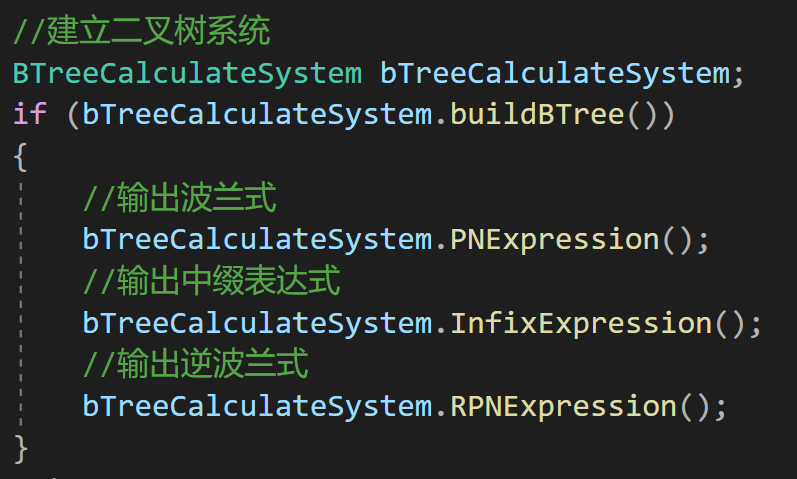
BTreeCalculateSystem类中的buildTree()函数



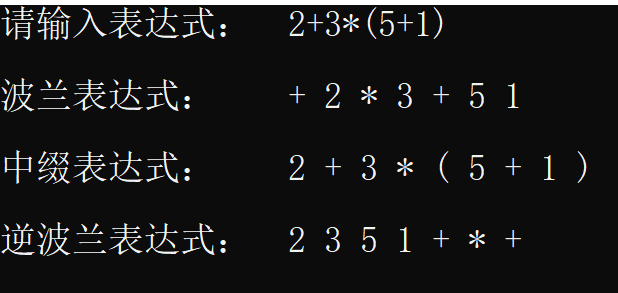
3.2.1 总体功能流程图



3.4.2 总体功能核心代码



3.4.3 总体功能截屏示例



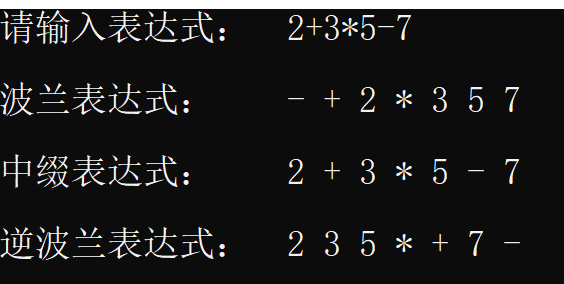
4 测试

4.1 功能测试

4.1.1 不带括号表达式功能测试

**测试用例**：2+3\*5-7

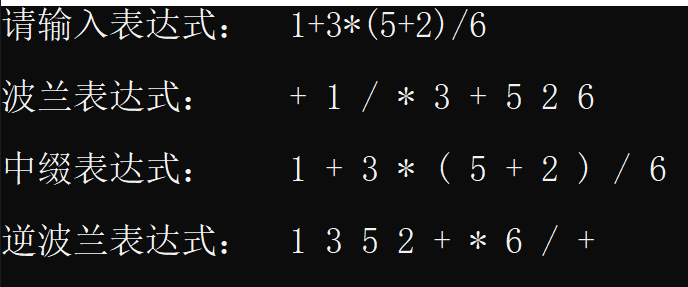
**实验结果：**结果正确



4.1.2 带括号表达式功能测试

**测试用例：**1+3\*(5+2)/6

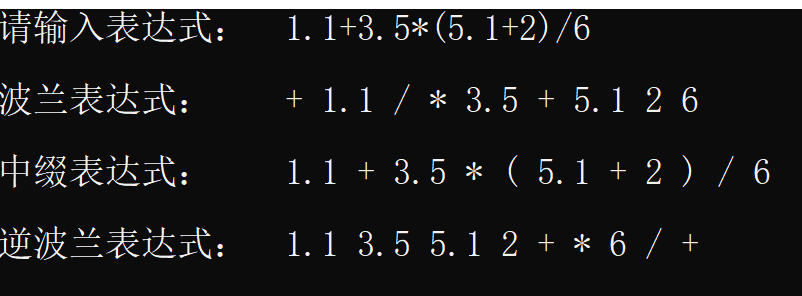
**实验结果：**结果正确



4.1.3 含小数表达式功能测试

**测试用例：**1+3\*(5+2)/6

**实验结果：**结果正确

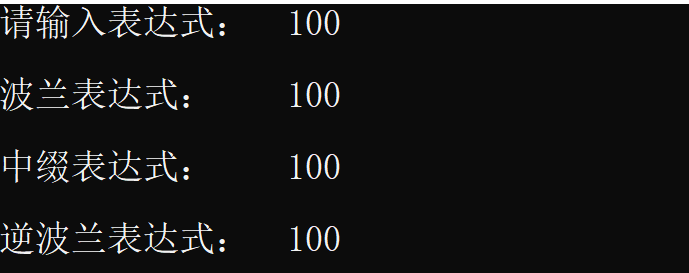


4.2 边界测试

4.2.1 表达式中无运算符

**测试用例：**100

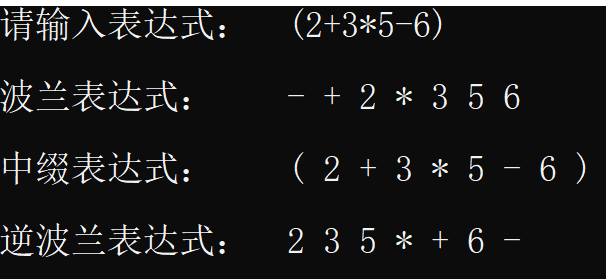
**实验结果：**结果正确



4.2.2 整个表达式被括号包围

**测试用例：**(2+3\*5-6)

**实验结果：**结果正确



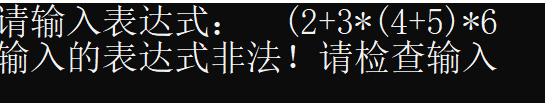
4.3 出错测试

4.3.1 表达式左右括号不匹配

**测试用例：**(2+3\*(4+5)\*6

**预期结果：**程序给出提示信息，程序正常运行不崩溃。

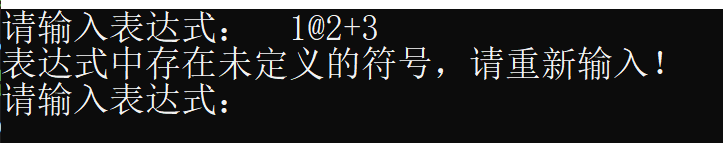
**实验结果：**



4.3.2 表达式中出现未定义符号

**测试用例：**1@2+3

**实验结果：**程序给出提示信息，程序正常运行不崩溃



4.3.2 表达式中出现空括号

**测试用例：**2+()+5

**实验结果：**程序给出提示信息，程序正常运行不崩溃

