数据结构项目七——表达式计算

同济大学 软件学院 15级2班 1552651 王依睿

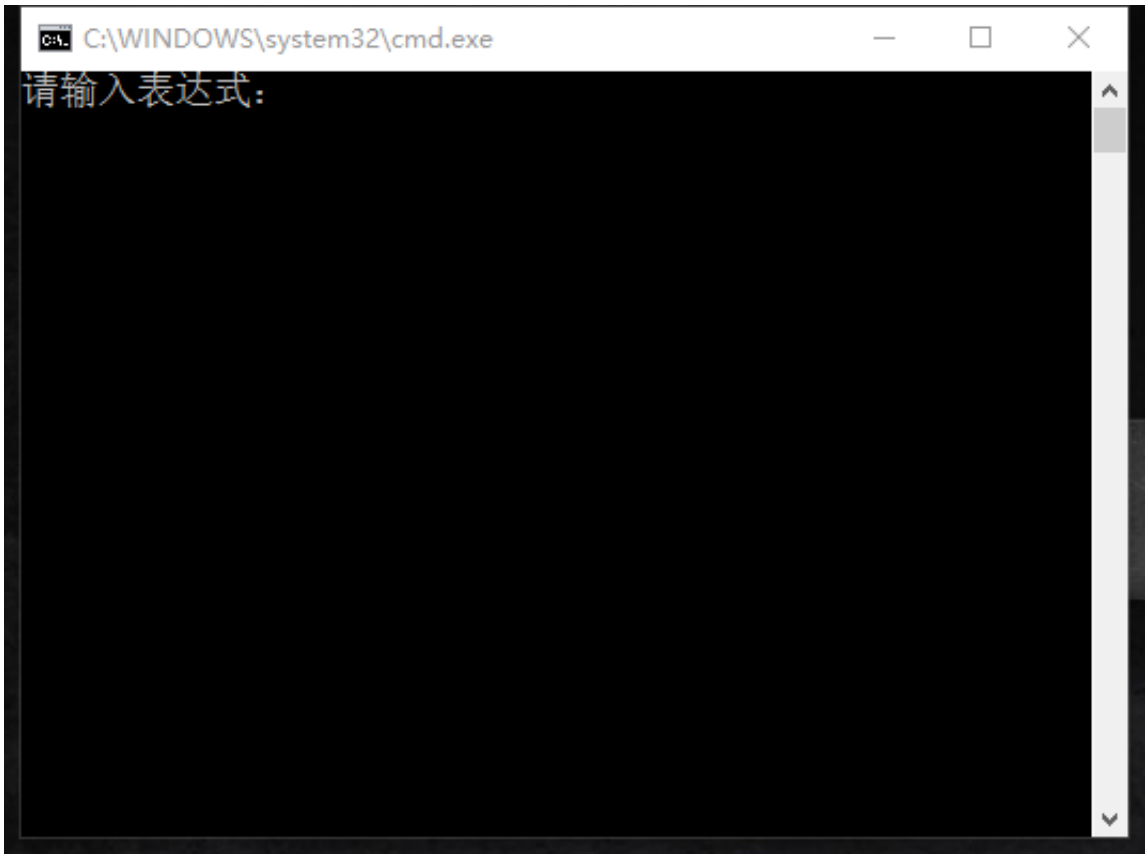
C++ 11 数据结构 二叉树

目录

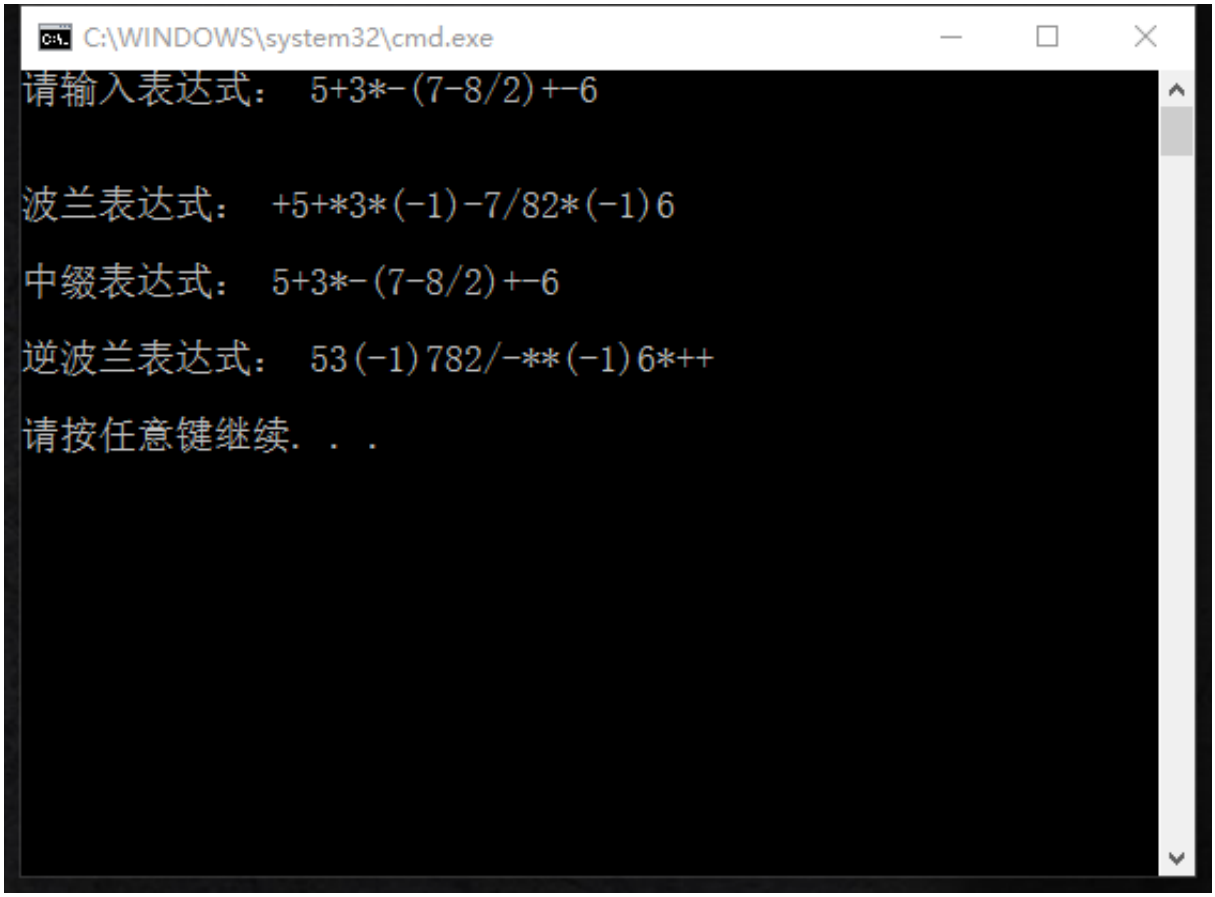
使用说明

操作手册

- 运行程序后，请求用户输入表达式。



- 输出波兰表达式、中缀表达式、逆波兰表达式。



注意事项

- 输入的表达式字符长度请勿超过100。
- 请输入合法的中缀表达式。
- 表达式中仅允许 '+' 、 '-' （作为减号或负号）、 '*' 、 '/' 、匹配的前后括号（ '(' 、 ')' ）以上六种运算符出现。
- 单个数字的取值在 -2147483648 ~ +2147483647 之间。

概述

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，输出对应的逆波兰式，中序表达式和波兰表达式。

项目简介

- 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。具体要求是以字符序列的形式从终端输入语法正确的，不含变量的整数表达式，并利用给定的优先关系实现对算术四则混合表达式的求值，并延时在求值过程中运算符栈，操作数栈，输入字符和主要操作变化过程。 要把一个表达式翻译成正确求值的一个机器指令序列，或者直接对表达式求值，首先要能正确解释表达式。任何一个表达式都是由操作符，运算符和界限符组成，我们称它们为单词。一般来说，操作数既可以是常数，又可以是被说明为变量或常量的标识符；运算符可以分成算术运算符，关系运算符和逻辑运算符3类；基本界限符有左右括号和表达式结束符等。为了叙述的简洁，我们仅仅讨论简单算术表达式的求值问题。这种表达式只包括加，减，乘，除4种运算符。 人们在书写表达式时通常采用的是“中缀”表达形式，也就是将运算符放在两个操作数中间，用这种“中缀”形式表示的表达式称为中缀表达式。但是，这种表达式表示形式对计算机处理来说是不大合适的。对于表达式的表示还有另一种形式，称之为“后缀表达式“，也就是将运算符紧跟在两个操作书的后面。这种表达式比较合适计算机的处理方式，因此要用计算机来处理，计算表达式的问题，首先要将中缀表达式转化成后缀表达式，又称为逆波兰表达式。

程序设计目的

- 实现一个将中缀表达式转换成波兰表达式和逆波兰表达式的简单程序，练习二叉树的编写实现，熟悉面向对象程序设计。 ### 算法思路
- 将读入的操作数和操作符进行预处理，将单个操作符或操作数以及它们的优先级存入一个结构体里（注意：如果遇到 '-' 作为负号使用，程序将其处理为 "(-1)*" 这个等价操作）。
- 利用一个操作符栈和一个操作数栈将表达式存入一个二叉树中：
  - 从表达式左端开始，如遇到操作数将其压入操作数栈。
  - 如遇到操作符，将其与栈顶操作符相比较：
    - 若栈顶操作符优先级高于当前待入栈的操作符，栈顶操作符出栈，操作数栈进行两次栈顶操作数出栈，并将操作符作为根节点，操作数分别作为左右子节点构造一棵二叉树，并将此压入操作数栈作为一个操作数。重复以上做法，直至栈顶操作符优先级低于或等于当前操作符待入栈的操作符。

        - 若栈顶操作符优先级低于或等于当前待入栈的操作符，将其压入操作符栈。

    - 处理完表达式最后一项后，用与上述相同的方法构造二叉树直至操作符栈弹空。
    - 从操作数栈中取出的唯一一个元素即是我们所要的表达式树。

- 利用二叉树的先序遍历输出波兰表达式，利用二叉树的后序遍历输出逆波兰表达式。 ###数据结构
- 采用两个栈，一个作为操作数栈，一个作为操作符栈
- 将二叉树节点封装成一个结构体
- 将二叉树（表达式树）封装成一个类

文件目录

- 1*1552651*wangyirui.cpp（主文件）
- BT_class.h（二叉树类）
- BT_class.cpp（二叉树类函数实现）
- 1*1552651*wangyirui.exe（可执行文件）
- 1*1552651*wangyirui.pdf（项目文档）

函数接口

二叉树节点接口（Node）

成员函数名	功能	参数	返回值类型
void setNode(char opr, int pr)	设置节点内部操作符及其优先级	操作符opr，优先级pr	/
void setNode(int opd, int pr)	设置节点内部操作数及其优先级（其优先级表示节点内部存储的是操作数）	操作数opd，优先级pr	/

二叉数节点类

```
class Node {

public:

    char opr_= NULL;//操作符
    int opd_ = 0;//操作数
    int pr_ = -2;//优先级,操作数的优先级为-1
    Node* lc_ = NULL;
    Node* rc_ = NULL;

    void setNode(char opr, int pr) {
        opr_ = opr;
        pr_ = pr;
    }

    void setNode(int opd, int pr) {
        opd_ = opd;
        pr_ = pr;
    }

};
```

二叉树接口（BT）

成员函数名	功能	参数	返回值类型
void visit(pNode x)	访问并打印节点	根节点pNode	/
void travPre_R(pNode x)	通过先序遍历打印节点	根节点pNode	/
void travIn_R(pNode x)	通过后序遍历打印节点	根节点pNode	/
void empty(pNode x)	用于清空二叉树	根节点pNode	/

二叉树类

```
class BT {
public:

    pNode root_ = NULL;//表达式树的根节点

    void visit(pNode x);//访问并打印打印节点
    void travPre_R(pNode x);//先序遍历
    void travIn_R(pNode x);//后序遍历
    void empty(pNode x);//用于清空二叉树

    ~BT() { empty(root_); }
};
```

栈处理函数接口

成员函数名	功能	参数	返回值类型
void stackprocess(pNode *expr, int cnt, pNode &root)	对表达式进行栈处理	表达式数组pNode *，表达式数组长度int，根节点 pNode	

构造二叉树函数接口

成员函数名	功能	参数	返回值类型
void buildtree(pNode opr, pNode sucopd, pNode preopd)	构造二叉树	根节点（操作符）pNode，左儿子（操作数1）pNode，右儿子（操作数2）pNode	