華南師範大學

本科学生实验（实践）报告

院　　系：计 算 机 学 院

实验课程：编译原理实验

实验项目：中缀表达式转后缀表达式

指导老师：王欣明

开课时间：2017 ~ 2018 年度第 2 学期

专　　业：软件技术与应用

班　　级：2015 级软工 4 班

学　　生：詹萍

学　　号：20152100027

华南师范大学教务处

目录

一、实验内容

1、使用 Java 语言实现中缀转后缀

实现一个完整的 Java 程序，它读取文件中的中缀表达式（每个表达式以分号结束，文件中可以有多个表达式）并转换为等价的后缀表达式后输出到屏幕上。表达式中的运算量可以是任意整数或者小数，支持加、减、乘、除、取负运算以及小括号，表达式中的空格、制表符等空白符号可以被忽略。

若用户输入的表达式有误，则提示用户错误的位置。譬如两个运算量之间缺少运 算符、或运算符缺少左（或右）运算量等。错误处理功能的最低要求是当输入表 达式有错时，给出一个报错信息，提示错误的位置和类别。学有余力的学生还可 考虑尝试如何实现出错恢复（Error Recovery），即当程序发现一个错误时不是立 马停下来，而是能够从跌倒的地方爬起来，继续分析下去，从而一次运行即可发现更多的错误。

2、提供一个随机测试数据发生器

为以上的 Java 程序提供一个随机测试数据发生器（用 Java 语言来写），生成若干随机的正确表达式和不正确表达式（通过命令行参数决定是生成正确的还是 不正确的以及生成的数量）。生成的测试数据要求写入文件，可以被 1）中的程序 读取。

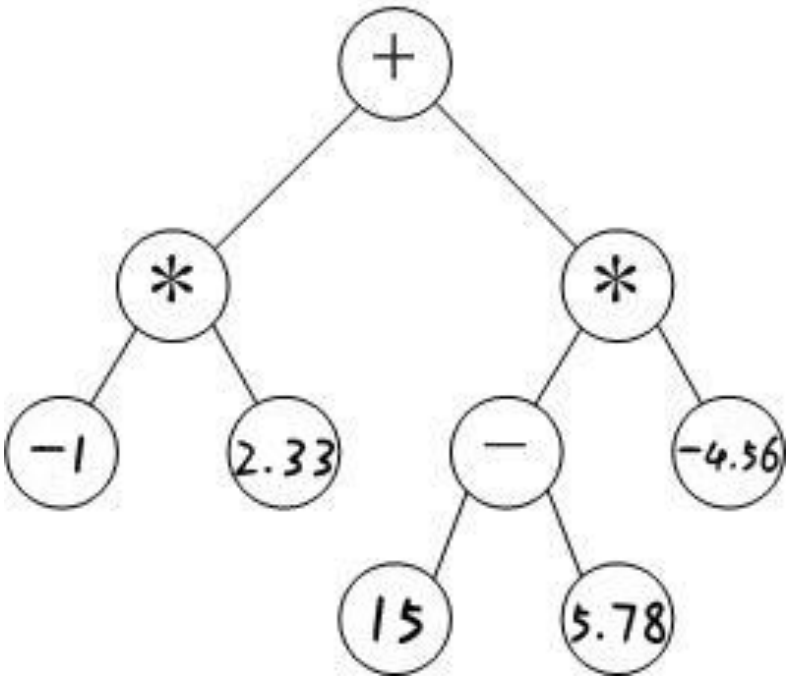
二、实现思路

1、随机测试数据发生器

（1）实现思路

算术表达式的表示方式有很多种，其中有一种方式是将表达式用二叉树的方式存储起来，其中叶子结点存放的是操作数，非叶子结点存放的是运算符。

如下图所示：



二叉树存储表达式的方式可以方便地生成中缀表达式（中序遍历）和后缀表达式（后序遍历）。其中二叉树的结点包含三部分，一个是数据本身，以及它的左右子树。

实现随机生成表达式的思路就是从一棵二叉树的树根出发，对于每个结点，使其随即成为运算符或者操作数。如果是运算符就继续生成其左右结点，如果是操作数，就作为叶子结点，直到整棵树建立完毕。

其中运算符和操作数（正数或者负数、整数或者浮点数）都随机生成。

二叉树建立好之后，通过中序遍历整棵二叉树来生成表达式。其中，如果当前结点的运算符是 + 或者 - 时，如果它的父结点是 * 或者 / 时，则需要对当前结点的运算加上一对括号。

（2）表达式的正误

生成正确表达式的方法如上所示，只需要中序遍历整棵二叉树将结点的数据连接起来即可。

生成错误的表达式我采用的方法是中序遍历整棵树先生成正确的表达式，再在表达式的运算符旁边随机插入另一个运算符或者随机将运算符替换成非运算符来生成错误的表达式。两个连续的运算符中间缺少操作数，两个操作数中间的运算符被替换成其他符号则缺少运算符。

2、检查表达式的正误

（1）括号匹配

可以用栈来检查括号匹配是否有错误。

当遇到左括号时，将其序号（是第几个左括号）压入栈中，遇到右括号时，从栈中弹出栈顶的左括号。

如果栈中已经为空时，出现新的右括号或者已经没有右括号而栈中仍然有左括号时，报错。

如果括号匹配没有错误，则进行算术表达式的语法检查。

（2）正则表达式匹配

先用正则表达式判断算术表达式是否存在语法错误。如果没有错误，则可以进行下一步中缀转后缀。如果存在错误，进行错误位置及类型检查。

在进行正则表达式匹配之前先去掉表达式中所有的括号，再写出算术表达式的正则表达式：(-?\\d+)(\\.\\d+)?((\\+|\\-|*|\\/)(-?\\d+)(\\.\\d+)?)*。使用 java 中包装好的类来进行匹配。根据匹配结果决定是否需要再进行检错。

（3）错误位置及类型检测

检查错误的方法设计得不太好，基本思路就是根据表达式是由：操作数 运算符 操作数 ……组成的这一规则来查找错误，如果不符合这样一个交替出现的规则，则报错。

具体思路：设一个标记来记录当前符号应该是操作数还是运算符，如果在该出现操作数的时候出现了运算符，说明当前的运算符后面缺少操作数，如果在该出现运算符的时候出现了操作数或者其他符号，说明两个操作数中间缺少运算符。

根据上述情况，报告出相应的错误位置以及错误类型。

3、中缀转后缀

中缀表达式转为后缀表达式可以使用栈来协助完成。

依次读取整个表达式的内容，有如下情况：

① 遇到操作数，直接加到后缀表达式中。

② 遇到运算符，与栈顶的运算符进行比较，如果栈顶运算符优先级比较低，就将当前运算符压入栈中；如果栈顶运算符优先级比较高，就从栈中弹出栈顶的运算符，从栈中弹出的运算符加到后缀表达式中；如果栈内外的运算符的优先级一样，说明这是一对括号，直接将括号弹出栈外即可，后缀表达式中不需要括号。

在这里对运算符引入了优先级的概念，并且栈内与栈外运算符的优先级不同。同样的运算符在栈内会比栈外的优先级高一些。这是因为算术表达式是从左到右计算的，对于同样的运算符，栈内的运算符是处于左边的所以优先级会高一点，可以先进行计算。

遇到运算符还有一种特殊情况就是它是操作数的符号，即操作数是负数，遇到这种情况只需要判断负号 - 前面是否连着另一个运算符即可。

三、实验结果

1、输入正确和错误的表达式的数目

```
请输入要生成的正确的表达式的数目：
5
请输入要生成的错误的表达式的数目：
5
```

2、展示随机生成的表达式

```
生成如下表达式：
(-17)/(-10.31)
23*((36+40-37)/20.41+22)/(-0.35)/6.51
19-(-9.31)
((0.58-1.60+(-4)+(-4)/((-4.20)+3*(-7.15))*11+25.57)/39.37/(-38)+(-12.78))/13+7.84-0.45
31+7
7.14/((-16.04)
8.85(*4.01
(-12.25)a(-5.75)
39a(-11)
(a47)/9
```

3、进行检错以及中缀转后缀处理

```
中缀表达式：(-17)/(-10.31)
后缀表达式：-17 -10.31 /

中缀表达式：23*((36+40-37)/20.41+22)/(-0.35)/6.51
后缀表达式：23 36 40 + 37 - 20.41 / 22 + * -0.35 / 6.51 /

中缀表达式：19-(-9.31)
后缀表达式：19 -9.31 -

中缀表达式：((0.58-1.60+(-4)+(-4)/((-4.20)+3*(-7.15))*11+25.57)/39.37/(-38)+(-12.78))/13+7.84-0.45
后缀表达式：0.58 1.60 - -4 + -4 -4.20 3 -7.15 * + / 11 * + 25.57 + 39.37 / -38 / -12.78 + 13 / 7.84 + 0.45 -

中缀表达式：31+7
后缀表达式：31 7 +

中缀表达式：7.14/((-16.04)
括号冗余，第1个括号多余

中缀表达式：8.85(*4.01
括号冗余，第1个括号多余

中缀表达式：(-12.25)a(-5.75)
第1个操作数后面缺少运算符

中缀表达式：39a(-11)
第1个操作数后面缺少运算符

中缀表达式：(a47)/9
第1个运算符前面缺少操作数!
```

四、实验小结

这次的实验主要是对表达式进行中缀转后缀的处理，在这之前需要进行表达式的生成以及检错的步骤。

这次主要还是用到了以前学习到的用二叉树表示表达式以及用运算符的优先级来处理表达式的做法。

在这次实验中也使用到了一小部分的正则表达式的应用，在读入表达式时，使用正则表达式去匹配表达式中的空格和制表符等空白符号，并将它们替换为这样”“一个空的串，来实现剔除空白符号的功能。在对表达式进行进行检错时，一开始的思路是让表达式去匹配算术表达式的正则表达式，如果匹配则没有错误；如果有错误，再找出其中的错误类型和错位位置，不过由于对于 java 的两个类 Pattern 和 Matcher 使用不熟练，没办法很好地利用起来，所以还是选择了比较笨的方法来查找错误。