

**院 系：计 算 机 学 院**

**实验课程：编译原理课程实验项目**

**实验项目：编译原理课程项目一**

**指导老师：王欣明**

**专 业：数据库**

**班 级：15级 2 班**

**学 生：李明育**

**学 号：20152100084**

**华南师范大学教务处**

**一、实验内容**

1. 实验题目

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。并为以上的 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，可以被上面的程序读取。

2）实验代码

①**StringToArithmetic.java**

package InfixToPostfix;

import java.util.Stack;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

import java.util.ArrayList;

import java.util.HashSet;

import java.io.File;

import java.io.FileReader;

import java.io.BufferedReader;

public class StringToArithmetic {

private StringToArithmetic() {}

public static String infixToSuffix(String infix) {

Stack<Character> stack = new Stack<Character>();

String suffix = "";

int length = infix.length();

for (int i = 0; i < length; i++) {

Character temp;

char c = infix.charAt(i);

switch (c) {

// 忽略空格

case ' ':

break;

// 碰到'('，push到栈

case '(':

stack.push(c);

break;

// 碰到'+''-'，将栈中所有运算符弹出，送到输出队列中

case '+':

case '-':

while (stack.size() != 0) {

temp = stack.pop();

if (temp == '(') {

stack.push('(');

break;

}

suffix +=temp;

}

stack.push(c);

break;

// 碰到'\*''/'，将栈中所有乘除运算符弹出，送到输出队列中

case '\*':

case '/':

while (stack.size() != 0) {

temp = stack.pop();

if (temp == '(' || temp == '+' || temp == '-') {

stack.push(temp);

break;

}

else {

suffix +=temp;

}

}

stack.push(c);

break;

// 碰到右括号，将靠近栈顶的第一个左括号上面的运算符全部依次弹出，送至输出队列后，再丢弃左括号

case ')':

while (stack.size() != 0) {

temp = stack.pop();

if (temp == '(')

break;

else

suffix +=temp;

}

break;

case ';':

break;

//如果是数字，直接送至输出序列

default:

suffix += c;

}

}

//如果栈不为空，把剩余的运算符依次弹出，送至输出序列。

while (stack.size() != 0) {

suffix +=stack.pop();

}

return suffix;

}

public static boolean isNumber(String str)

{

Pattern p=Pattern.compile("\\d+||(\\d+\\.\\d+)");

Matcher n=p.matcher(str);

if(!n.matches())

return false;

return true;

}

public static void main(String[] args)throws Exception{

File f=new File("test.txt");

FileReader fr=new FileReader(f);

BufferedReader br=new BufferedReader(fr);

String str =null;

String s=null;

int i=1;

while ((str=br.readLine()) !=null)

{

s=str.substring(0,str.length()-1);

System.out.println(StringToArithmetic.infixToSuffix(str));

i++;

}

br.close();

}

}

②**lianxi.java**

package InfixToPostfix;

import java.io.BufferedWriter;

import java.io.File;

import java.io.FileWriter;

import java.io.IOException;

import java.util.ArrayList;

import java.util.Scanner;

import java.util.HashSet;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

public class lianxi {

public static boolean isNumber(String str)

{

Pattern p=Pattern.compile("\\d+||(\\d+\\.\\d+)");

Matcher n=p.matcher(str);

if(!n.matches())

return false;

return true;

}

public static boolean getNumberStringFromList(ArrayList<Character>list)

{

if(list!=null) {

StringBuffer s=new StringBuffer();

for(Character c:list)

s.append(c);

boolean r=isNumber(s.toString());

if(!r)

{

return false;

}

return r;

}

return false;

}

public static boolean fun(String str)

{

HashSet<Character>os=new HashSet<>();

os.add('-');

os.add('+');

os.add('/');

os.add('\*');

ArrayList<Character>d=new ArrayList<>();

int cn=0;

int length = str.length();

for(int j=0;j<length;j++)

{

char c = str.charAt(j);

if(Character.isDigit(c)||c=='.')

d.add(c);

else

{

if(d.size()>0)

{

boolean r=getNumberStringFromList(d);

if(r)

d.clear();

else

return false;

}

if(c=='+'||c=='\*'||c=='/')

{ if(j==0||j==(length-1)||os.contains(str.charAt(j+1))||str.charAt(j+1)==')')

{

return false;

}

}

else if(c=='-')

{

if(j==(length-1)||os.contains(str.charAt(j+1))||str.charAt(j+1)==')')

{

return false;

}

}

else if(c=='(')

{

cn++;

if(j==(length-1)||str.charAt(j+1)=='+'||str.charAt(j+1)=='\*'||str.charAt(j+1)=='/'|str.charAt(j+1)==')'||(j!=0&&Character.isDigit(str.charAt(j-1))))

{

return false;

}

}

else if(c==')')

{

cn--;

if(j==0||j<(length-1)&&str.charAt(j+1)=='('||cn<0)

{

return false;

}

}

else if(c!=';')

{

return false;

}

}

}

if(d.size()>0)

{

boolean r=getNumberStringFromList(d);

if(r)

d.clear();

else

return false;

}

if(cn!=0)

return false;

return true;

}

public static String createxpression()

{

StringBuffer str=new StringBuffer();

char Op[]= {'+','-','\*','/'};

int length=(int)(Math.random()\*10)+2;

boolean d=false;

int bc=0;

for(int i=0;i<length;i++)

{

if(d)

{

d=false;

double num=Math.random()\*10;

while(num==0.0)

num=Math.random()\*10;

str.append(num);

}

else str.append((int)(Math.random()\*10));

int t=bc;

for(int j=0;j<t;j++)

{

if(Math.random()\*5>2)

{

bc--;

str.append(')');

}

}

int op=(int)(Math.random()\*4);

str.append(Op[op]);

if(op==3)

d=true;

if(Math.random()\*2>1)

{

str.append('(');

bc++;

str.append((int)(Math.random()\*10));

op=(int)Math.random()\*4;

str.append(Op[op]);

if(op==3)

d=true;

}

}

if(d)

{

d=false;

double num=Math.random()\*10;

while(num==0.0)

num=Math.random()\*10;

str.append((int)num);

}

else str.append((int)(Math.random()\*10));

for(int i=0;i<bc;i++)

{

str.append(')');

}

str.append(';');

return str.toString();

}

}

③**Test.java （表达式生成器）**

package InfixToPostfix;

import java.io.BufferedWriter;

import java.io.File;

import java.io.FileWriter;

import java.io.IOException;

import java.util.Scanner;

public class Test {

public static void main(String[] args) throws IOException {

File f=new File("test.txt");

f.createNewFile();

BufferedWriter out=new BufferedWriter(new FileWriter(f));

Scanner s=new Scanner(System.in);

System.out.println("请输入表达式生成个数:");

int n=s.nextInt();

String ss=null;

for(int i=0;i<n;i++)

{

out.write(lianxi.createxpression()+"\r\n");

}

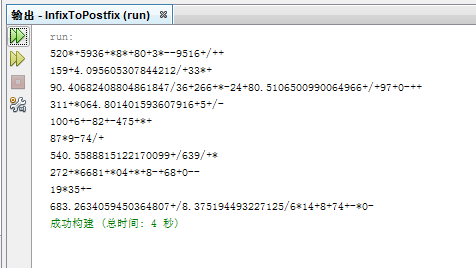
out.close();

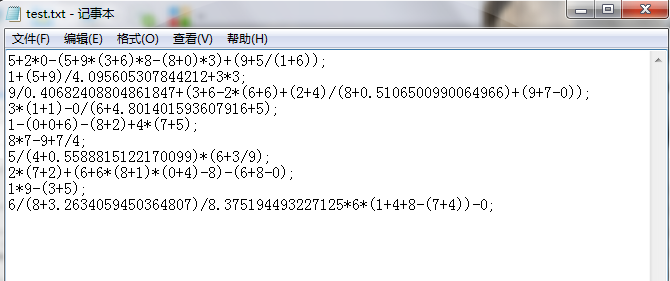
s.close();

}

}

3）运行结果





**二、实验小结**

本次实验出现了一些我未曾掌握的内容，比如生成不正确的表达式，以及判断如何检查错误位置，因此本次实验代码并没有涉及这两个方面，希望在之后，我可以更好地了解这次实验项目，从而可以写出符合项目要求的代码。