**中山大学软件学院本科生实验报告**

|  |  |  |  |
| --- | --- | --- | --- |
| 年级 | **10级** | 专业（方向） | **软件工程（电子政务）** |
| 学号 | **10389455** | 姓名 | **霍俞蓉** |
| 电话 | **15800021493** | Email | **380217075@qq.com** |
| 开始日期 | **2012.10.29** | 完成日期 | **2011.10.30** |

课程名称：**数据结构** 任课教师：**林倞**

1. **需求分析**

**总概：**将一个中缀表达式转换成后缀表达式，在转换过程中要检查此表达式是否有误，如果出现错误，程序会与相应的错误提示。表达式是简单的对0-9的数字进行加与减的运算，表达式不允许有制表符、空格等符号

**详细：（1）输入值：**一个0-9的加减运算中缀表达式

**输出值:**如果输入值正确则为完整的后缀表达式

如果错误，则提示响应错误信息

**程序运行：**在将中缀表达式转换为后缀表达式的过程中进行词法和语法的分析

1. **实验内容（过程）**
2. **静态成员与非静态成员**

程序中lookahead的类型为静态，原因是**：**可以在不创建对象的时候就可以调用静态成员，并且它只占用一份存储空间。

在实验过程中将lookahead的类型改为非静态并没有影响程序的运行。

当你的程序有许多静态方法或者你希望在不创建对象的时候就使用某种方法和成员（你的方法和成员跟任一个类都没有关系时），静态成员就是最好的选择。

1. **消除程序尾递归**

在此过程中将rest（）部分用循环来替换。代码框架如下：

**void** expr() **throws** IOException

{

**if**( ! term()){}

**while**(**true**)

{

**if** ((**char**)*lookahead* == '+')

{

**if**( ! match('+')){**break**;}

**if**( ! term()){**break**;}

System.*out*.print('+');

}

**else** **if** ((**char**)*lookahead* == '-')

{

**if**( ! match('-')){**break**;}

**if**( ! term()){**break**;}

System.*out*.print('-');

}

**else**

{

**if**((**char**)*lookahead* == '\r'){**break**;}

**if**(Character.*isDigit*((**char**)*lookahead*)){}

**else** {}

**}**

**try**

{

*lookahead* = System.*in*.read();

} **catch** (IOException e) {e.printStackTrace();}

**if**((**char**)*lookahead* == '\r') {**break** ;}

}

}

在用循环替换了递归方法之后为了证实是否有效率提高的结果，我编写了比较程序分别用两种方法计算时间复杂度。

**设计方案：**

**测试数据的获得：**创建类Produceinfix，实现创建10个.infix文档，第一个文档规定中缀表达式长度为501，每创建一个文档，表达式长度都会增加500，0-9的数字为随机产生，“+”“-“符号也为随机。（详细内容请看具体类）

**尾递归方法：**类Parser\_TailRecursion

**循环方法：**类Parser\_Loop

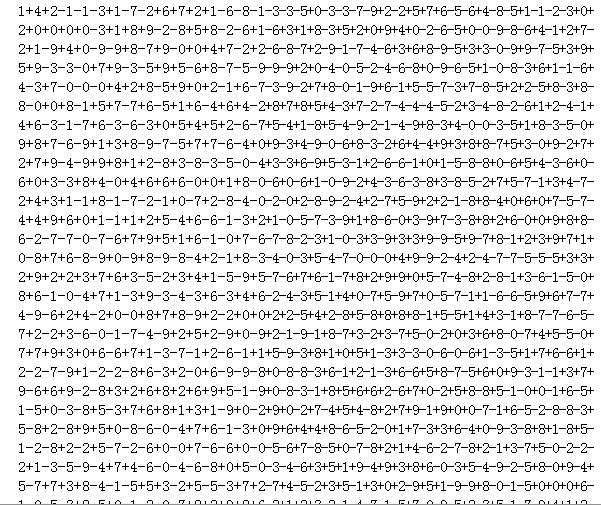
**主程序：**类Compare\_two\_method，在这个类中，我们读取每一个infix文档作为输入，分别调用两种方法类，对于每一个文档我们都会用两种方法分别执行20次，并计算costtime，最后分别将两个方法执行20次的costtime做平均得到最终结果，并将它写入result.txt中。（详细内容请看result文档）

**预期结果：**因为递归程序会有创建和销毁的过程所以预计会消耗更多时间

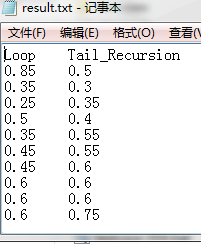
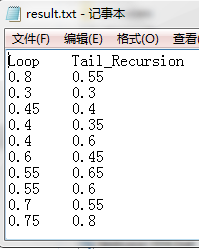
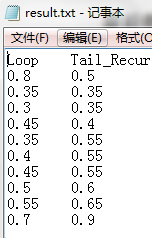
**执行方案：**

**Infix文件（选取一个）：**

Test004.infix

****

**最后得到的数据为：**

****

从上图可以看出循环方法在有大量数据的情况下并没有对性能有明显的提升。

1. **扩展错误处理**

**错误类型：**

**（1）词法错误**

缺少操作符—一个数字的右边是非法操作符（非+，\_符号）

缺少操作符—两个0-9的数字连在一起

**（2）语法错误**

表达式为空

缺少操作符—数字之前没有“+、\_”符号

缺少右操作数—符号右边没有数字

缺少左操作数—当一个非法操作符（非+、\_）和+，\_符号连在一起时

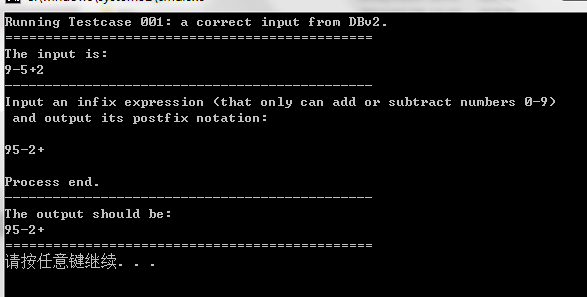
**测试结果：**

**Testcase-001：正确测试**

输入：test001.infix

输出：test001.postfix

结果正确

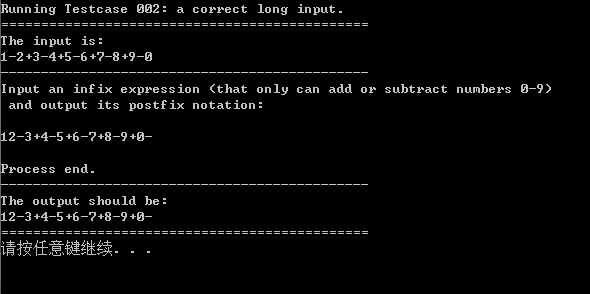
****

**Testcase-002：正确测试**

输入：test002.infix

输出：test002.postfix

结果正确

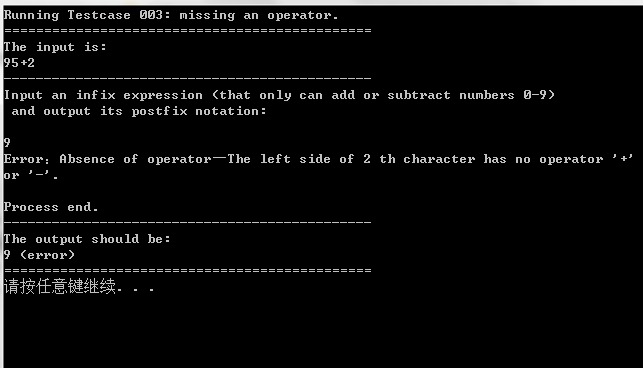


**Testcase-003：error测试—缺少操作符**

输入：test003.infix

输出：test003.postfix

结果正确

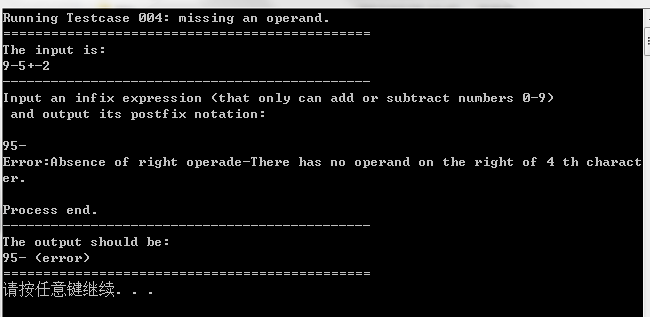
****

**Testcase-004：error测试—缺少操作数**

输入：test004.infix

输出：test004.postfix

结果正确

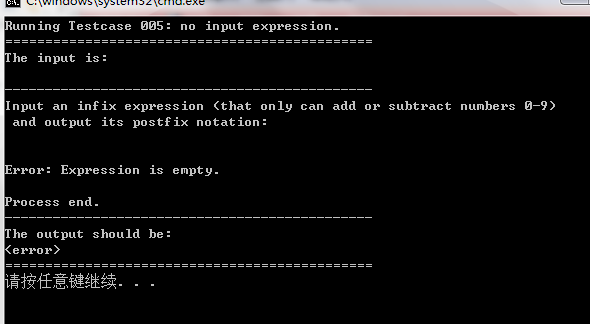
****

**Testcase-005：error测试—输入为空**

输入：test005.infix

输出：test005.postfix

结果正确

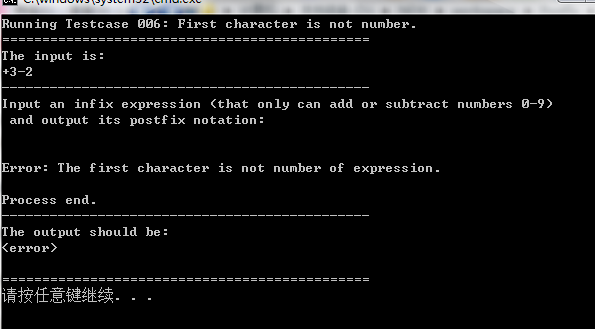
****

**Testcase-006：error测试—input表达式第一个字符不为数字**

输入：test006.infix

输出：test006.postfix

结果正确

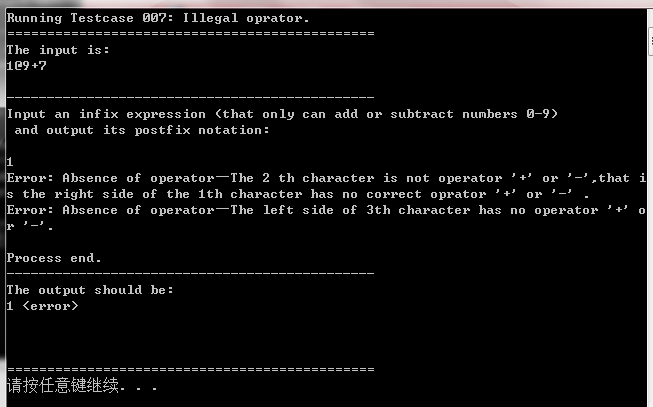
****

**Testcase-007：error测试—表达式有非法操作符**

输入：test007.infix

输出：test007.postfix

结果正确

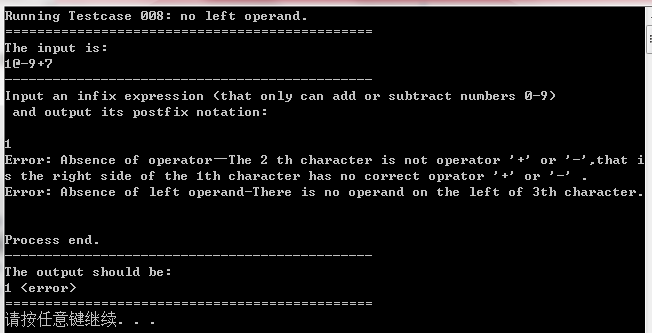
****

**Testcase-008：error测试—非法操作符之后连接操作符**

输入：test008.infix

输出：test008.postfix

结果正确

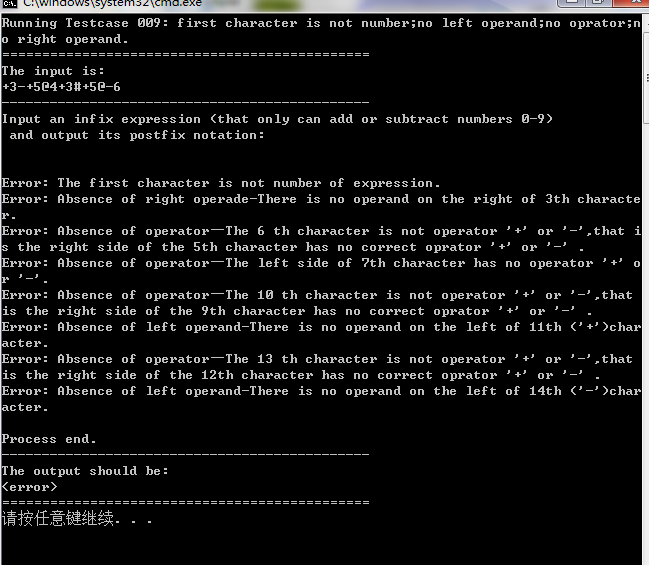
****

**Testcase-009：error测试—input表达式第一个字符不为数字,；无左操作数（非法操作符与合法操作符相连）；无操作符；无右操作数**

输入：test009.infix

输出：test009.postfix

结果正确

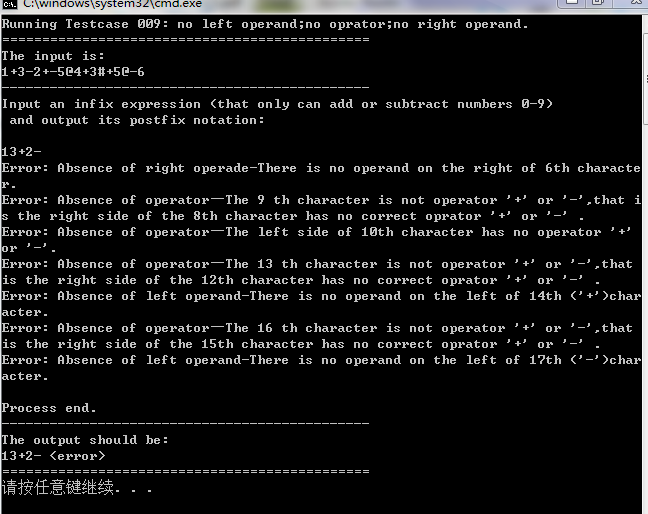
****

**Testcase-010：error测试—无左操作数（非法操作符与合法操作符相连）；无操作符；无右操作数**

输入：test010.infix

输出：test010.postfix

结果正确

****

3. **附录**

项目工程：Postfix

**目录Bin：**

存放编译后生成的JAVA字节码Postfix.class，parser.class，compare包文件中生成的可执行JAVA字节码（Compare\_two\_method.class）

**目录Src：**

Postfix.java

Compare包：

**c**ompare\_two\_method.java（计算递归与循环方法时间复杂度主程序）

**目录Doc：**

文档注释生成的HTML文档

**目录testcase：**

10个infix文档（输入表达式），10个postfix文档（对应的预期输出）

**目录compare\_time：**

10个infix文档（输入表达式）

**Result.txt文档：**

递归方法与loop方法转换10个输入表达式，每一个所用时间