****

编译原理实验

报告一

**实验名称：Lexical Analyzer**

**学生姓名： 柳沿河**

**学生学号： 71117230**

东南大学计算机科学与工程学院、软件学院

School of Computer Science & Engineering College of Software Engineering

Southeast University

二 0 一 九 年 十 二 月

1. 目的
   * 了解词法分析的原理
   * 掌握通过自定义正规表达式构造最少状态DFA的方法
   * 掌握基于正则表达式转化得到的DFA编写词法分析程序的
2. 内容描述

编写一个词法分析程序，要求如下：

1. 输入：字符流、正则表达式集合（数量自定义）
2. 输出：token序列
3. 单词类别自定义
4. 考虑错误处理

任选一下两种实现方式实现：

1. 基于FA
2. 基于Lex
3. 方法

本次实验采用第一种实现方式，即通过将正则表达式转换为最少状态DFA后基于该DFA来进行词法分析程序的构建。

1. 假设
   * 分析语言：类C语言词法
   * 词法对应的正则表达式集合：
     + 数字：digit 0|1|2|3|4|5|6|7|8|9
     + 字母：

letter→a|b|c|d|e|f|g|h|i|g|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z|A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z

* + - 标识符：identifier →{letter}({letter}|{digit})\*
    - 十进制数值：

number→0|(1|2|3|4|5|6|7|8|9)({digit})\*|(1|2|3|4|5|6|7|8|9)({digit})\*.{digit}({digit})\*|(0).{digit}({digit})\*

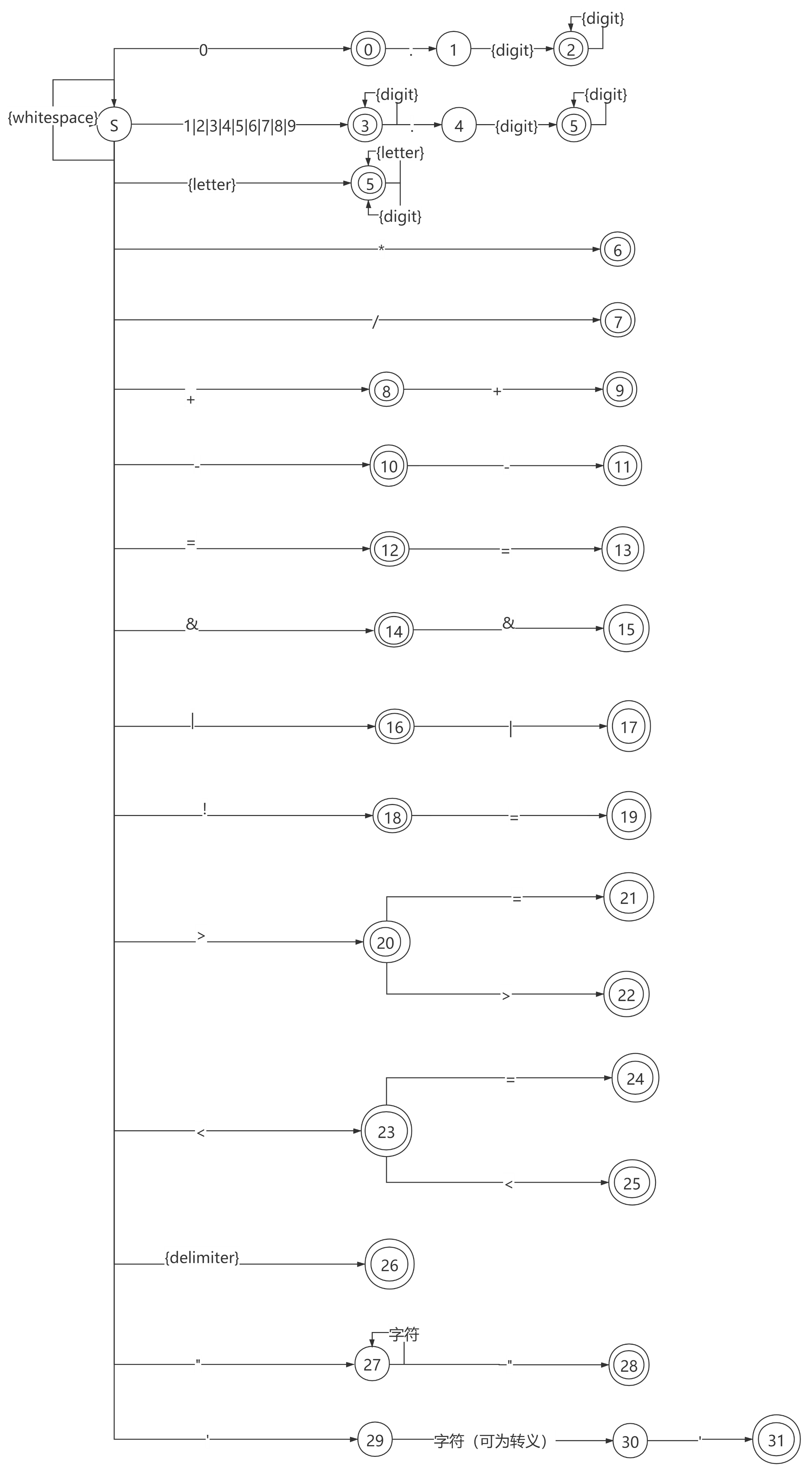
* + - 运算符：operator→+|-|\*|/|…
    - 界符：delimiter→,|;|“|(|…
    - 字符常量：character

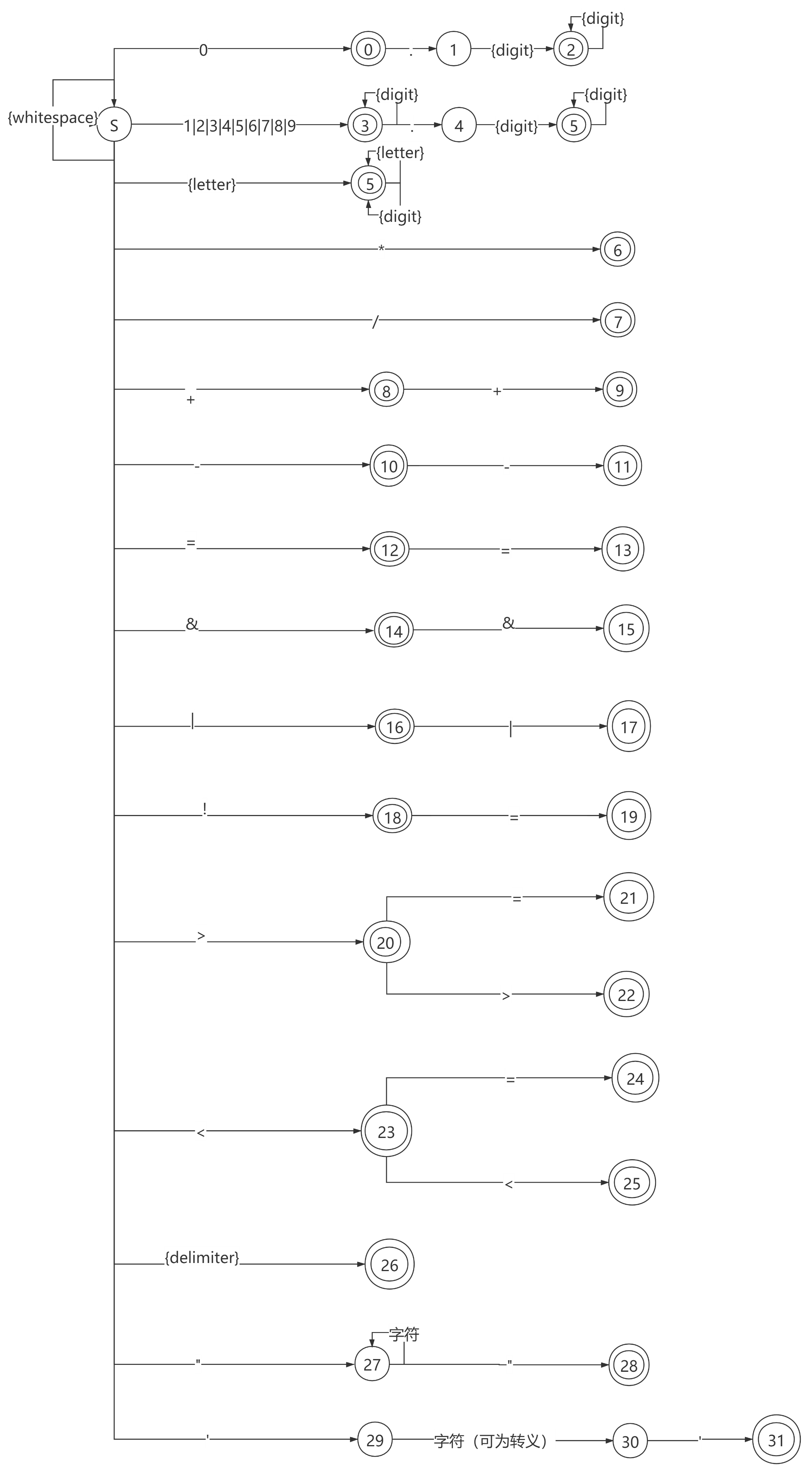
→’{letter}’|’{digit}’|’{delimiter}’|’{operator}’|’\t’|’\r’|’\n’|’\”’|’\’’|’\\’

* + - 字符串：string→”({letter}|{digit}|{whitespace}|{delimiter}|{operator})\*”
    - 部分关键字：keyword→class|include|int|…
    - 空白符：whitespace→\t|\n|\r|

1. 相关FA描述

根据正则表达式集合构造的DFA中的部分示意图如下：





1. 重要的数据结构描述
   * Token类：从字符流中识别出的token对象
     + 成员
       - String word：识别出的单词
       - String attribute：对应属性值
       - int innerCode：单词对应token的内部码
     + 方法
       - 重载toString函数：用于向文件中以一定的格式写入token对象
   * TranslationTable类：转换表对象
     + 成员
       - List<String> keyWordsList：存放所有已定义的合法关键字的列表，索引即内部码
       - List<String> operatorsList：存放所有已定义的合法运算符的列表，索引即内部码
       - List<String> delimitersList：存放所有已定义的界符的列表，索引即内部码
       - Map<Integer, String> attributeMap：存放token对应的属性值的列表，内部码为键，属性值为值
     + 方法
       - boolean isKeyWord(String token)：判断是否为已定义的关键字
       - Token getKeyWordAttributeValue(String token)：创建对应的Token对象
2. 核心算法描述

算法：类C词法分析算法analyze

输入：待分析的代码文件，TranslationTable对象

输出：存放识别出的token序列的文本文件

执行过程：

1. 创建待分析文件对象inputFile和输出文件对象outputFile；
2. 为文件对象连接输入流对象reader和输出流对象writer
3. 创建读入字符对象c和字符缓冲区对象buffer
4. 从输入流中读入一个字符，存放在c中；若c不为文件终结符且小于最大可读入字符数（65535），则执行下一步；否则停止读入，判断三种括号的栈中有无剩余的括号，若有则报错，否则分析成功，结束程序
5. 若c为空白字符（\t, \n, \r, space），返回4；否则执行下一步
6. 若c为数字，执行下一步；否则到8；
7. 若c为0，则读入下一个字符，判断是不是小数点，若是则继续读入
   1. 若c为0，缓冲区中不含小数点，或有多个小数点，或小数点不在第一个0之后，则报错；否则为该数字构造Token对象并写入文件后清空缓冲区，返回4；
   2. 若c不为0，缓冲区中有多个小数点，或小数点在最后，则报错，否则为该数字构造Token对象并写入文件后清空缓冲区，返回4
8. 若c为字母，执行下一步，否则到10
9. 若c为关键字的首字母，则依次读入相应的字符到缓冲区中，判断是否为关键字，若是关键字则构造相应的token对象，写入文件中并清空缓冲区；否则持续读入字符或数字，按照ID构造token对象，写入文件中并清空缓冲区；若超出文件可读范围，则报错
10. 若c为运算符，则执行下一步，否则到12
11. 将c写入缓冲区.
    1. 若c为可以与另一个运算符组合成一个运算符的运算符，则将c存入beg，从输入流中再读入一个字符到c中，并将c写入缓冲区
       1. 若c为运算符，则根据beg的值判断c是否可以与beg构成运算符，若不能则报错。再读入一个字符到c中，若c为操作符，则报错；否则将c退回输入流，对c+beg构造相应的token对象并写入文件，之后清空缓冲区，到4
       2. 否则将c退回输入流，对c构造相应的token对象并写入文件，之后清空缓冲区，到4
    2. 对c构造相应的token对象并写入文件，之后清空缓冲区，返回4
12. 若c为界符，则执行下一步，否则到14
13. 若c为左括号，则将其和所在的行号压入栈中，构造相应的Token对象并写入文件；若c为右括号，若对应的栈为空，则报错；否则弹栈，构造相应Token对象并写入文件中；若c为单引号，则读入字符并判断是否合法（普通字符或转义字符），若合法则构造Token对象并写入文件，之后清空缓冲区，否则报错；若c为双引号，则持续读入字符，直到遇到下一个双引号，若不存在下一个双引号则报错，否则构造相应的Token对象并写入文件。之后清空缓冲区，返回4
14. c为非法字符，报错
15. 用例

测试用例如下：

int i = 0;

char c = 'a';

char \*lowercase = " ";

char \*uppercase = " ";

while( i < 51 )

{

char temp = c + i;

if ( i < 26 )

lowercase[i] = temp;

else

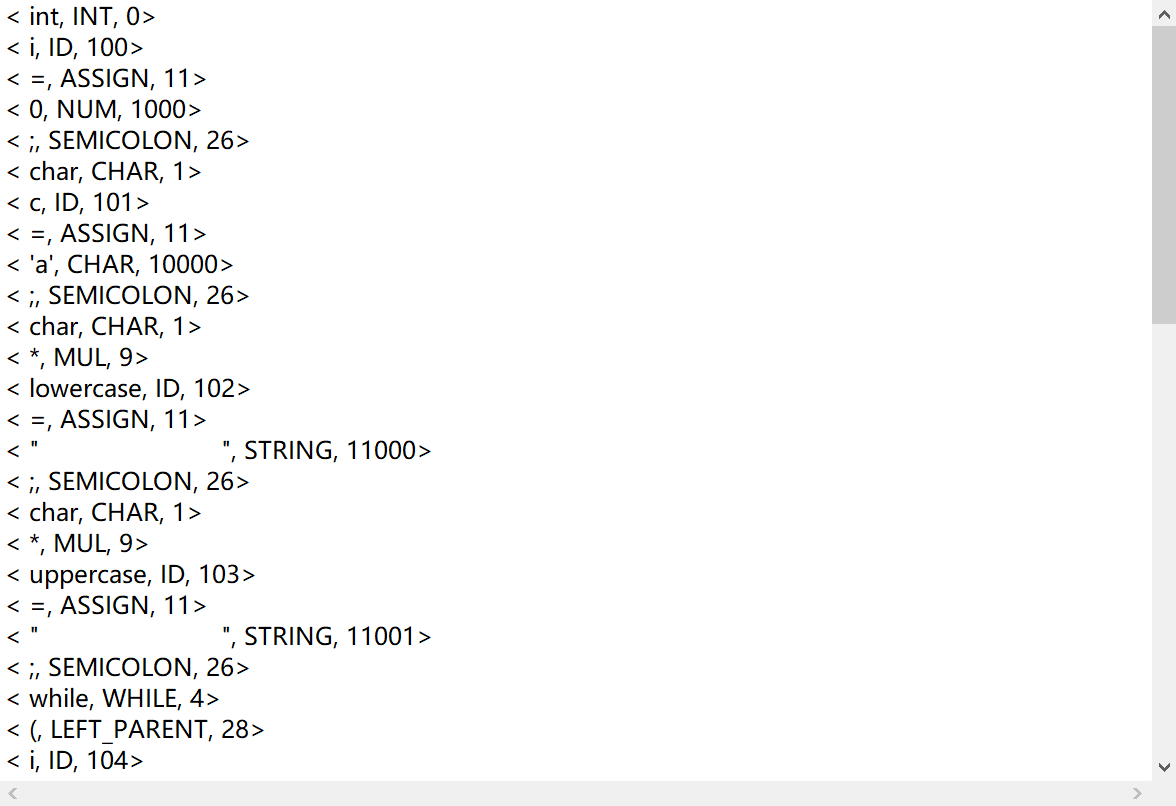
uppercase[i - 26] = temp;

}

运行程序方法：

* java LexicalAnalyzer test.cpp tokens.txt
* IDE中运行测试类LexAnalyzerTest

运行结果如下：



1. 发生的问题和相关的解决方案

* 正则表达式集合→DFA：先为每个正则表达式构建NFA，再合并这些NFA形成总体的NFA，最后进行优化
* 词法分析算法的代码实现：将DFA中的状态转换边作为分支，将回路转换为循环
* 将字符退回输入流：使用java.io包中的PushbackReader对象连接文件的输入流，并使用unread方法将不为转换边上的终结符的字符退回输入流

1. 感受和评论

刚开始进行这个实验时没有头绪，无从下手。阅读了两种实现方式后，首先尝试采用了第二种方式——利用lex和自定义.l文件生成词法分析程序，发现比较简单，但使用的是cygwin下的flex，后来咨询老师后才得知lex需要自己实现，其核心是实现正则表达式→NFA→最小DFA的转换算法，挑战较大，遂放弃了这种方式，但是还是切身体验了用lex和.l文件生成词法分析程序的过程，掌握了Windows下使用lex的方法，可谓卒或有所闻。

之后开始转战第一种实现方式——自定义正则表达式、构造相应的最小状态DFA、基于DFA来编写词法分析程序。首先在自定义正则表达式上遇到了困难，不知道应该构建怎样的正则表达式来正确地表示C++的部分词法，在参考了老师给出的资料后得到了自己的正则表达式集合；然后利用课堂中学到的知识将正则表达式集合转换为了最少状态DFA；最后基于DFA实现了词法分析。有了DFA后，编码的思路也就比较清晰简单了，除了构造主干的数据结构外，剩余的问题主要是一些细节上的问题，如字符的回退、缓冲区的创建和清空等等。

通过这次实验，基本了解了词法分析的流程，掌握了词法分析程序的构造方法，加深了对所学知识的印象。本次实验还有很多待改进的地方，如token的识别；未来可以尝试用第二种方法实现。