# Lexical Analyzer编译报告

## Motivation/Aim

编写一个词法分析程序，通过编写词法分析程序，掌握记号、模式与单词，掌握正规式与正规集，掌握有限自动机，掌握如何从正规式到词法分析器的各种算法。

## Content description

1)Input

Stream of characters

REs(The number of REs is decided by yourself)

2)Output

Sequence of tokens

3)Classes of words are defined by yourself

4)Error handling may be included

## Ideas/Methods

G -> RE -> NFA -> DFA -> 最小化DFA -> 完成

程序需要对输入的字符串进行判断，是否被最小化DFA接受，若是，则输出token，不是，则程序结束。

## Assumption

Token分为标识符、关键字、操作符、分隔符、注释符、整数和浮点数。

关键字：public|class|static|void|main|while|if|else|for|switch|case|package|String|int|float

|double|char；

操作符：[,],~,(,),?,^,%,\*,/,.,’,”,:,+,-,<,>,=,!,&,|,+=,-=,<=,>=,!=,&&,||,<<,>>,++,--；

分隔符：;|{|}；

注释符：//|/\*|\*/；

另外，空格、\n、\t、\r在词法分析阶段忽略。

## Related FA descriptions

将Token构造正则表达式：

标识符：letter(letter|digit)\*

关键字：public|class|static|void|main|while|if|else|for|switch|case|package|String|int|float

|double|char

操作符：[,],~,(,),?,^,%,\*,/,.,’,”,:,+,-,<,>,=,!,&,|,+=,-=,<=,>=,!=,&&,||,<<,>>,++,--

分隔符：;|{|}

注释符：//|/\*|\*/

整数：digit (digit)\*

浮点数：digit (digit)\*. digit (digit)\*

## Description of important Data Structures

状态：

STATE{

DONE,INVAR,INADD,INMINUS,INLESS,INMORE,INEQUAL,INEXCLAMATORY,INAND,INDIGIT\_DOT,INOR,INSOLIDUS,INDIGIT,INDECIMALS,ANNOTATION\_ONE\_LINE,ANNOTATION\_MULTI\_LINE,ANNOTATION\_MULTI\_LINE\_ASTERISK,SINGLE\_QUOTE\_MARK,DOUBLE\_QUOTE\_MARK

}

记号：

TOKEN{

KEYWORDS,IDENTIFIER,OPERATOR,DELIMITER,INT,DOUBLE,ANNOTATION

}

记号和符号表的结构：

HashMap<TOKEN,String> map=new HashMap<TOKEN,String>();

initMap() {

map.put(TOKEN.KEYWORDS, "关键字");

map.put(TOKEN.IDENTIFIER, "标识符");

map.put(TOKEN.OPERATOR, "操作符");

map.put(TOKEN.DELIMITER, "分隔符");

map.put(TOKEN.INT, "整数");

map.put(TOKEN.DOUBLE, "浮点数");

map.put(TOKEN.ANNOTATION, "注释符");

}

关键字数组结构：

String KEYWORDS[]={"public","class","static","void",

"main","while","if","else","for","switch","case","package","String","int","float","double","char"};

## Description of core Algorithms

本算法的要求是从字符串表示的代码段中识别具有独立意义的单词、符号，要做到根据扫描到单词符号的第一个字符的种类，拼出相应的单词符号。关键字也作为标识符。当扫描程序扫描到标识符时，优先查找一张事先建立好的关键字表，若能匹配，则将该单词识别为关键字，否则为一般性质的标识符。

识别出源程序中的各个单词符号，并转换成内部编码形式，各个单词符号分为：KEYWORDS, IDENTIFIER, OPERATOR, DELIMITER, INT, DOUBLE, ANNOTATION这几种。删除无用的空白字符回车字符以及其他非实质性字符。删除注释。进行词法检查，报告所发现的错误。基本都是运用条件判断和比较，最后分门别类输出。每一个单词后会表明这是什么类型的单词符号。

## Use cases on running

输入程序：input.txt

public class Hello{

public static void main(String[] argv){

System.out.println("Hello World!");

}

}

**输出结果：**

关键字,"public"

关键字,"class"

Hello,"标识符 "

非运算符号,"{"

关键字,"public"

关键字,"static"

关键字,"void"

关键字,"main"

非运算符号,"("

String,"标识符 "

非运算符号,"["

非运算符号,"]"

argv,"标识符 "

非运算符号,")"

非运算符号,"{"

System,"标识符 "

非运算符号,"."

out,"标识符 "

非运算符号,"."

println,"标识符 "

非运算符号,"("

非运算符号,""Hello World!""

非运算符号,")"

非运算符号,";"

非运算符号,"}"

非运算符号,"}"