

# 编译原理 Lab1

## 词法编译器实验报告

151250084

李晓冬

2017/10/26

## 1. 实验目的

在第三章中，我们学习了如何将一段代码分析为合法的词法单元。我们首先提供词法的正则表达式，根据正则表达式构造出 **NFA**（不确定的有穷自动机），再将 **NFA** 优化到 **DFA**（确定的有穷自动机）。借助 **DFA** 判断单词的合法性，将程序转换成合法的词法单元供后续语法分析使用。

在这个实验中，我构造一个词法分析器。

## 2. 内容描述

这个词法分析器实现一下功能。

- 1) 读取输入程序代码，在 `Lexcial Analyer/input/Input.txt`
- 2) 忽略程序中的空白符
- 3) 识别大部分词法单元（不包括注释，以及带非字母或数字的标识符）
- 4) 简单识别词法错误，标识为 `error`
- 5) 输出词法单元序列，在 `Lexcial Analyer/result/Output.txt`

## 3. 想法 / 方法

根据词法的 **REs**（正则表达式）生成 **NFA**（不确定的有穷自动机），再转化为 **DFA**（确定的有穷自动机）。

## 4. 假设

假设输入为 `java` 程序，无注释，标识符命名皆采用数字和字母。

- 1) 关键字：

```

abstract 4
boolean 5
break 6
byte 7
case 8
catch 9
char 10
class 11
continue 12
default 13
do 14
double 15
else 16
extends 17
false 18
final 19
finally 20
float 21
for 22
if 23
implements 24
import 25
instanceof 26
int 27
interface 28
long 29
native 30
new 31
null 32
package 33
private 34
protected 35
public 36
return 37
short 38
static 39
super 40
switch 41
synchronized 42
this 43
throw 44
throws 45
transient 46
try 47
true 48
void 49
volatile 50
while 51

```

2) 操作符和界符:

```

+ 52
- 53
* 54
/ 55
% 56
& 57
~ 58
| 59
^ 60
! 61
< 62
> 63
" 64
' 65
, 66
; 67
: 68
[ 69
] 70
. 71
= 72
{ 73
} 74
? 75
( 76
) 77
\ 78

```

3) 空白符: " ", \t, \n

4) 整数: INT

INT -> digit·digit\*

digit -> [0-9]

5) 小数: DOUBLE

```
DOUBLE ->digit·digit*·/.digit*  
digit ->[0-9]
```

6) ID:

```
ID -> letter (letter | digit)  
digit -> 0|1|2|3|4|5|6|7|8|9  
letter ->[a-z]||[A-Z]
```

7) 其他: error

## 5. 重要的数据结构

关键字、操作符和界符的状态采用 4.1 和 4.2 的表驱动法，二目操作符的状态转换也采用表驱动，

```
52 = 79  
53 = 80  
54 = 81  
55 = 82  
56 = 83  
57 & 84  
59 | 85  
61 = 86  
62 = 87  
62 < 88  
62 > 89  
63 = 90  
63 > 91  
72 = 92
```

程序初始化时将三组数据分别放入数组 initState、keyWordState、transState 中。

读入的程序代码剔除空白字符存入数组 pro 中，token 用于暂存词素，p 是遍历指针，syn 持有状态值，pro\_len 是程序的长度。

## 6. 核心算法

- 1) 算法很简单，每次扫描不断以字符为单元读入程序，如果第一个字符为字母则不断读入后续的字母或数字，直到非字母或数字停止，组成词素，先判断词素是否为关键字，不是则为标识符。
- 2) 其次判断输入首字符是否为数字，若是则继续读入数字直到非数字为止，词此词法单元名字为 INT（整数），如果中间出现一个小数点，且后续跟着若干数字，则此词法单元为 DOUBLE（小数）。
- 3) 接下来判断是否为操作符或者是界符，若是，则继续读入下一字符，看两个字符可否组成双目操作符。
- 4) 以上皆不是则标识为错误。

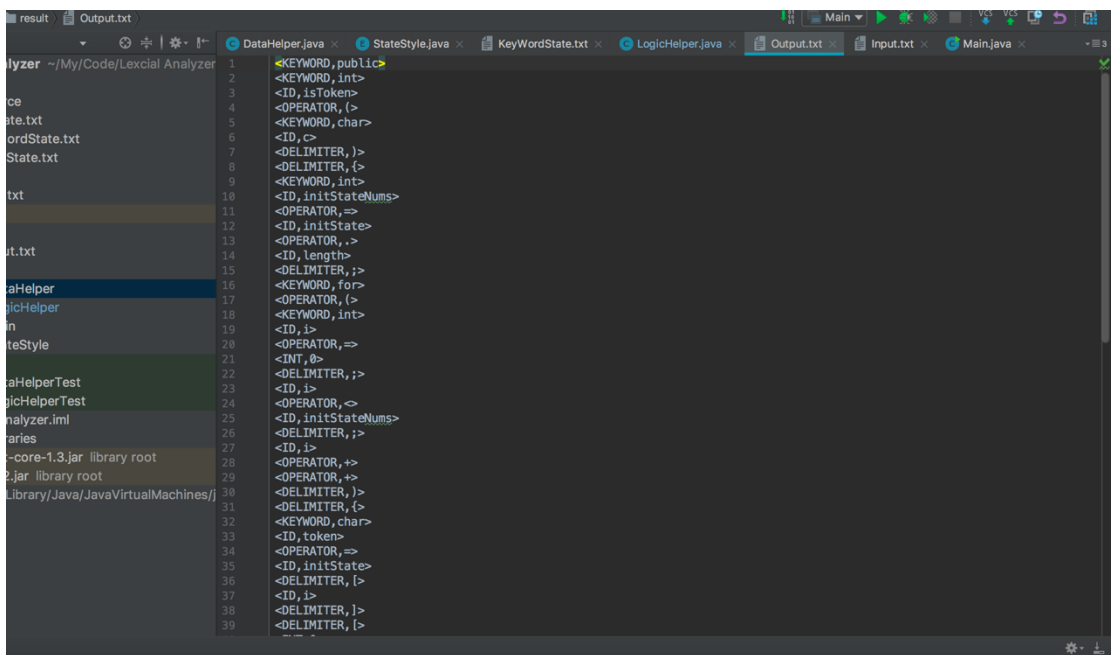
## 7. 运行实例

输入



```
1 int initStateNums = initState.length;
2 for(int i=0;i<initStateNums;i++){
3     char token = initState[i][0].charAt(0);
4     if(c == token){
5         return Integer.parseInt(initState[i][1]);
6     }
7 }
8 return -1;
9
10 }
```

## 输出



```
1 <KEYWORD,public>
2 <KEYWORD,int>
3 <ID,isValid>
4 <OPERATOR,{>
5 <KEYWORD,char>
6 <ID,c>
7 <DELIMITER,)>
8 <DELIMITER,{>
9 <KEYWORD,int>
10 <ID,initStateNums>
11 <OPERATOR,=>
12 <ID,initState>
13 <OPERATOR,.>
14 <ID,length>
15 <DELIMITER,;>
16 <KEYWORD,for>
17 <OPERATOR,{>
18 <KEYWORD,int>
19 <ID,i>
20 <OPERATOR,=>
21 <INT,0>
22 <DELIMITER,;>
23 <ID,i>
24 <OPERATOR,<
25 <ID,initStateNums>
26 <DELIMITER,;>
27 <ID,i>
28 <OPERATOR,+>
29 <OPERATOR,+>
30 <DELIMITER,)>
31 <DELIMITER,{>
32 <KEYWORD,char>
33 <ID,token>
34 <OPERATOR,=>
35 <ID,initState>
36 <DELIMITER,)>
37 <ID,i>
38 <DELIMITER,)>
39 <DELIMITER,)>
```

## 8. 遇到的问题

基本没有什么重大的实现上的困难，比较有难度还是数据结构的设计，看能不能想到，能想到，有思路就顺水推舟，迎刃而解了。主要参考博客资料

<http://www.cnblogs.com/yanlingyin/archive/2012/04/17/2451717.html>

## 9. 感受

通过这次实验温习了 java 编程，周到了感觉，也对词法分析有了小认识。现在觉得在开始写程序前的设计是很重要的，特别是设计到一些复杂的数据结构和算法的，设计的好实现起来也很容易了，码代码是技术难度较小的部分。