**Compilers Principles Lab**

**Lab 1 Lex**

**71118415 叶宏庭**

# Motivation

该实验的目的是为了自行编写一个词法分析器Lex，可以针对输入的字符流进行词法分析，返回结果的Token序列。

# Content description

1. Input

Stream of characters

REs(The number of REs is decided by yourself)

1. Output

Sequence of tokens

3) Classes of words are defined by yourself

4) Error handling may be included

# Ideas/Methods

1. 定义正则表达式RE
2. 自定义Res -> NFAs -> NFA -> DFA -> DFA°
3. 基于DFA°进行编码

# Assumptions

1. 本实验完成的为java的词法分析器，因此输入必须为xxx.java文件。
2. 定义RE如下：

**digit** -> 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

**letter** -> 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 | 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

**operator** -> + | - | \* | / | % | > | < | = | & | | | ~ | >= | <= | == | != | && | || | ++ | — | += | -= | ( | ) | [ | ] | . | ”

**separator**-> , | ; | { | } | \ | \\ | ’ | ” | \t | \n

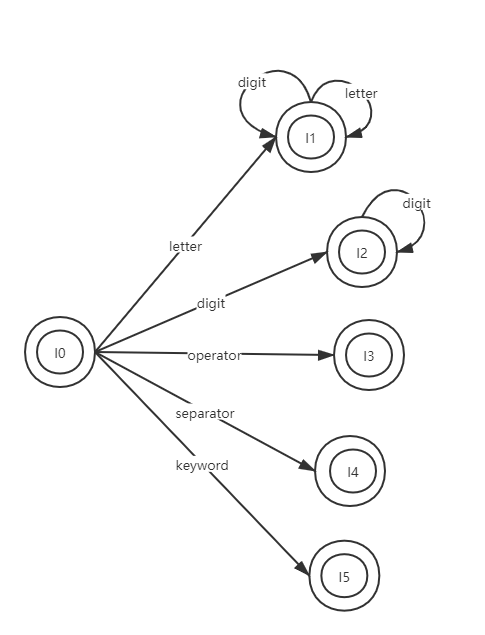
**keyword** -> abstract | boolean | break | byte | case | catch | char | class | const | continue | default | do | double | else | enum | extends | false | final | finally | float | for | if | implements | import | int | interface | long | new | null | package | private | protected | public | return | short | static | super | switch | this | throw | throws | try | true | void | while

**ID** -> <letter>(<letter> | <digit>)\*

**NUMBER** -> <digit> (<digit>)\*

# Related FA descriptions

最终DFA°如下图所示：I0为初始状态

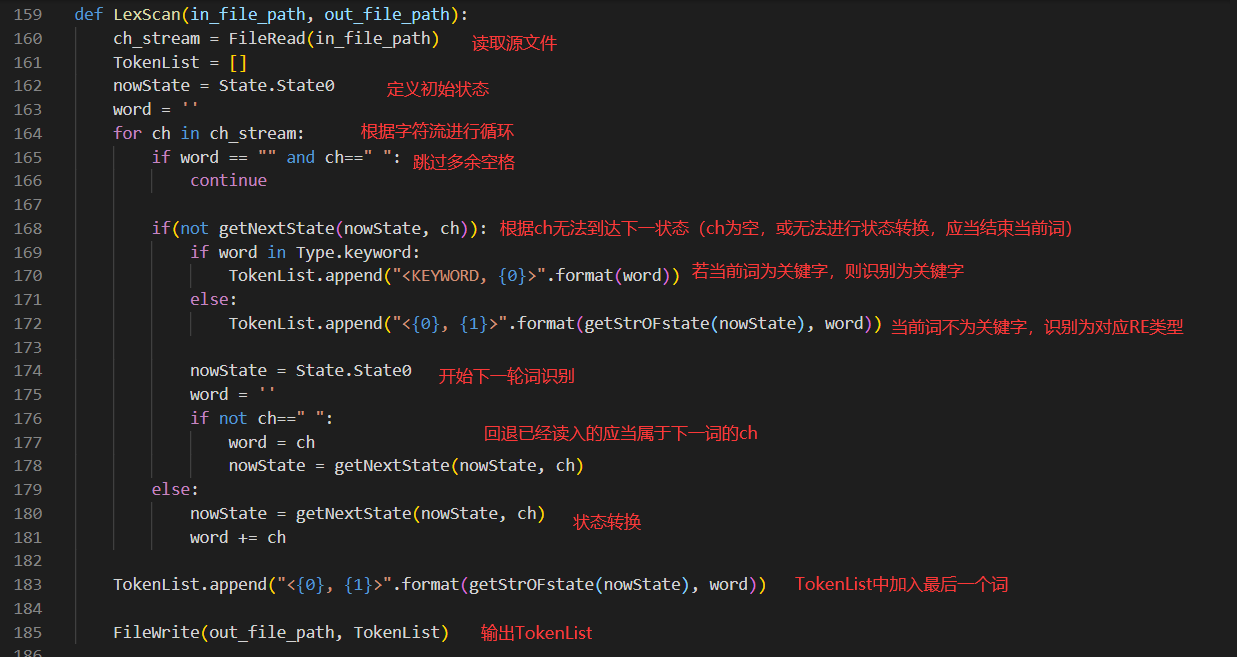


# Description of important Data Structures

1. **Enum\_Type:** 定义了统一的枚举类型，包括digit、letter、operator、separator、keyword。
2. **Type：**存储上诉五种类型的具体list，包括判断ch所属类型的函数TypeJudge()。
3. **State：**定义了当前状态的枚举类型，根据DFA°，共包含六个状态。
4. **tansM字典**：定义状态表（Soft coding）。

# Description of core Algorithms

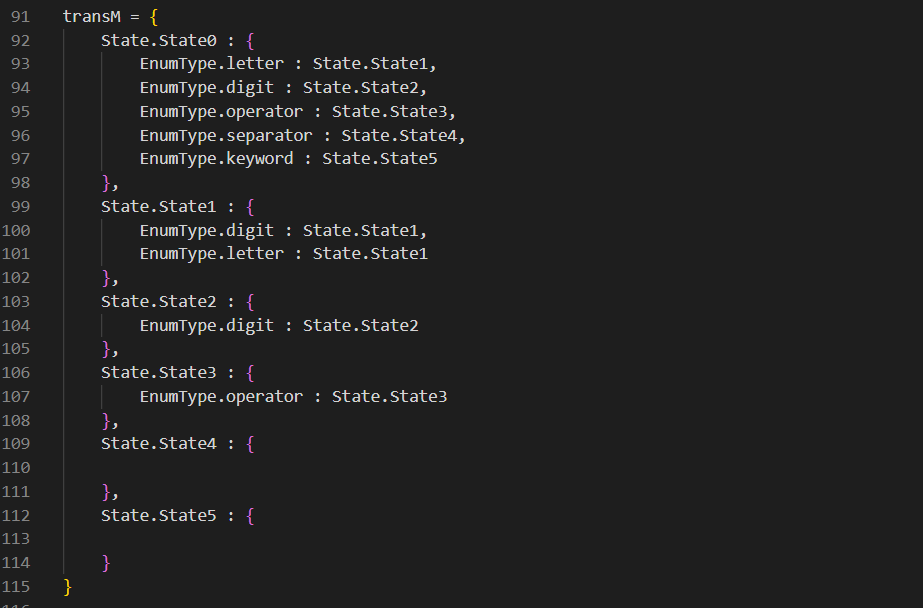
**LexScan()方法为词法分析主程序，下图中已经针对程序中的每一步做出解释。**



**输入输出模块：**

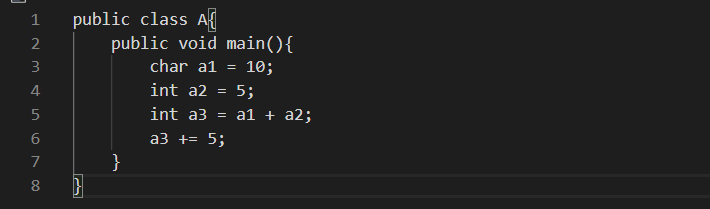


**状态转换表（Soft coding关键）：**



# Use cases on running

s,java作为源文件输入，进行词法分析，下图为s.java的内容。



下面给出词法分析结果out.txt：

<KEYWORD, public>

<KEYWORD, class>

<ID, A>

<SEPARATOR, {>

<KEYWORD, public>

<KEYWORD, void>

<ID, main>

<OPERATOR, ()>

<SEPARATOR, {>

<KEYWORD, char>

<ID, a1>

<OPERATOR, =>

<DIGIT, 10>

<SEPARATOR, ;>

<KEYWORD, int>

<ID, a2>

<OPERATOR, =>

<DIGIT, 5>

<SEPARATOR, ;>

<KEYWORD, int>

<ID, a3>

<OPERATOR, =>

<ID, a1>

<OPERATOR, +>

<ID, a2>

<SEPARATOR, ;>

<ID, a3>

<OPERATOR, +=>

<DIGIT, 5>

<SEPARATOR, ;>

<SEPARATOR, }>

<SEPARATOR, }>

# Problems occurred and related solutions

1. 在第一次编码时，忘记做ch的回退操作，导致部分字串的词法分析丢失。后来补上相应代码后，成功完成词法分析
2. 在定义数据结构时，没有很好组织结构关系，可以进行更好的结构优化。

# Your feelings and comments

通过本次实验，更加深入理解了Lex词法分析器的工作原理，同时完成了自己的一个简易的词法分析器，更好的掌握了相关的理论知识，在未来的学习和工作中都会有很大的帮助，也希望我能够有时间继续去完善这个词法分析器