**词法分析实验**

**1. 实验目的**

(1)熟悉 C 语言的词法规则,了解编译器词法分析器的主要功能;

(2)掌握典型词法分析器构造的相关技术和方法,设计并实现 C 语言词法分析器;

(3)掌握编译器从前端到后端各个模块的工作原理,词法分析模块与其他模块之间的交互过程。

**2. 实验内容**

根据 C 语言的词法规则, 设计识别 C 语言所有单词类的词法分析器的确定有限状态自动机, 并使用Java采用程序中心法或者数据中心法设计并实现词法分析器。词法分析器的输入为经过预处理宏替换与注释消去的 C 语言源程序,输出为属性字流。

**3. 实验分析**

C语言的词法由常量、标识符、关键字、操作符与分隔符构成，由128个ASCII码与无数其它字符构成，若以整个C语言设计一个确定性有限自动机，这个自动机将非常的庞大。因此，可以将这个大自动机拆分为很多个小的自动机，将简化它的设计工作。

1．常量：常量分为整形常量、浮点型常量、字符型常量、字符串型常量、其它型常量暂时不考虑。

1.1 整形常量是只由数字字符组成的字符串

1.2 符点型常量由整形+小数点+整形组成

1.3 字符型常量由单引号+字符串+单引号组成

1.4 字符串常量由双引号+字符串+双引号组成

1.5 布尔型常量有true与false

1.6 空常量NULL

1.7 其它进制数

1.8 其它常量暂时不考虑

2. 标识符：标识符是由字母或下划线开头，由字母或数字或下划线组成的字符串。

3. 关键字：关键字为一种特殊的标识符，是在C中保留的有特殊含义的标识符，因此关键字集合是标识符的一个子集。常用的保留字有{auto, static, register, extern, char, double, enum, float, int, long, short, signed, union, unsigned, void, for, do, while, break, continue, if, else, goto, switch, case, default, const, sizeof, typedef, volatile, return, include, define, true, false, NULL }

4. 操作符：操作符比较特殊，有单目操作符如 ++,--,\*,&,!,~ , 有双目运算符如 +,-,\*,/,%,<<,>>,>,>=,<,<=,==,!=,&,^,|,&&,||, 三目运算符如?:, 特殊运算符如 (),[],.,+=,-=,\*=,/=,%=,<<=,>>=,&=,|=,!=。确定性有限自动机只能识别一个连续的字符串，因此不连续的运算符如 ?: , () , [] 需要语法分析时识别。

5. 分隔符：常用的分隔符如 : , ; , { , } , , 还有上面某些分隔的操作符如 ( , ) , [ , ]

6. 空白符：无特别含义的字符如 空格，换行，TAB。识别出这些空白符可以直接跳过。

词法分析器分析输出的属性字流无特别的标准，因此结果是仁者见仁，智者见智的。实际上属性字类型分类方法需要结合后面的语法分析进行更改、优化，词法分析应该为语法分析提供方便，而非为了输出与已知的语法分析器相同的属性字流，因此，我这里就暂时只使用一种简单的token分类方法，等到语法分析时再进行标准优化。Token的类型分为

identifier、separator、operator、keyword

constant细分为constr、conchar、conintN (N进制)、confloat、conbool、connull。

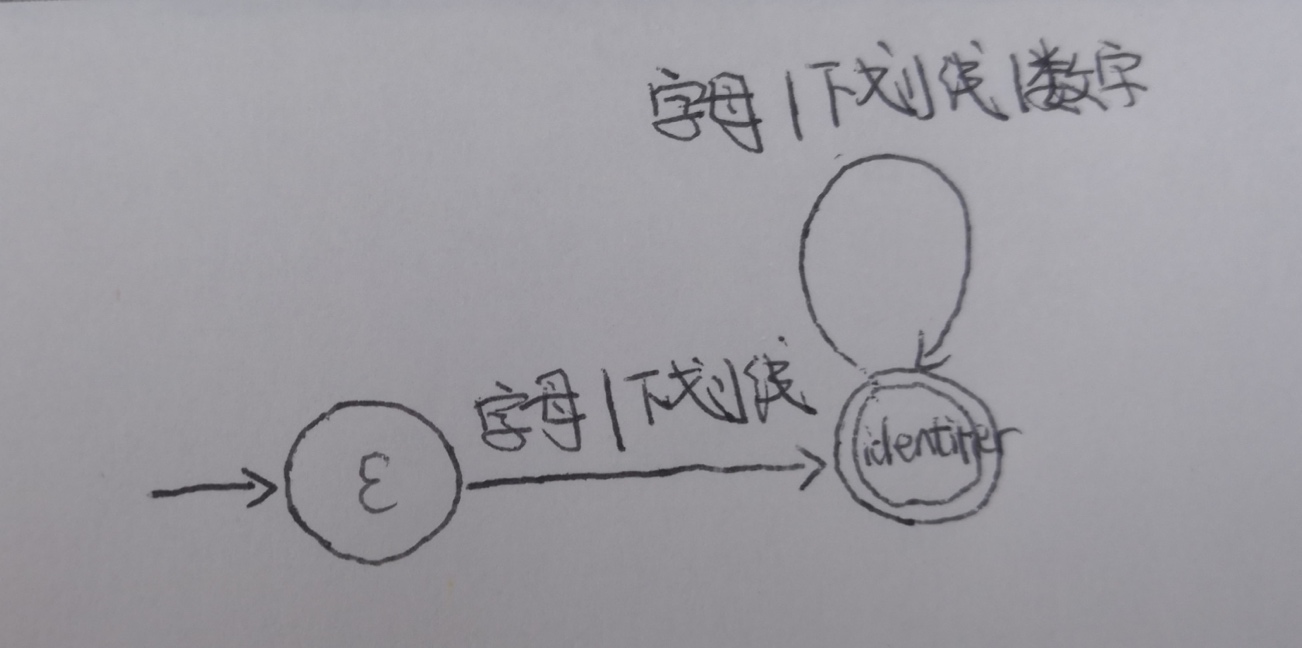
**4. 自动机与分析器设计**

4.1 NFA设计

keyword是identifier的子集，separator全部是单个字符，operator最长的只有两个字符，并且个数较少，各个操作符之间没有很强的相似性，使用枚举法或者哈希效率甚至会更高，不需要单独设计一个NFA。因此需要设计的NFA只有identifier、constant

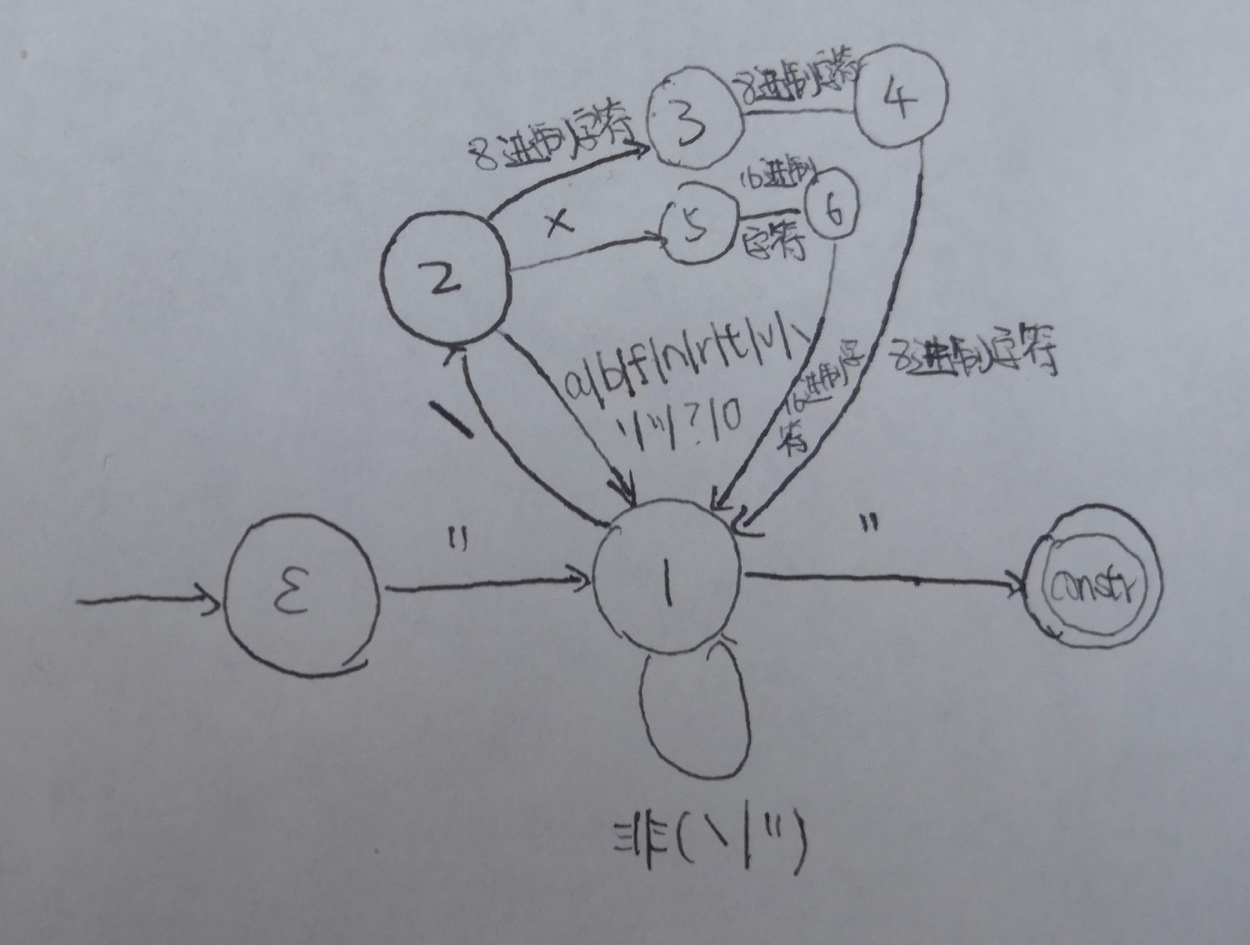
4.1.1 标识符不确定性有限自动机

终态为identifier



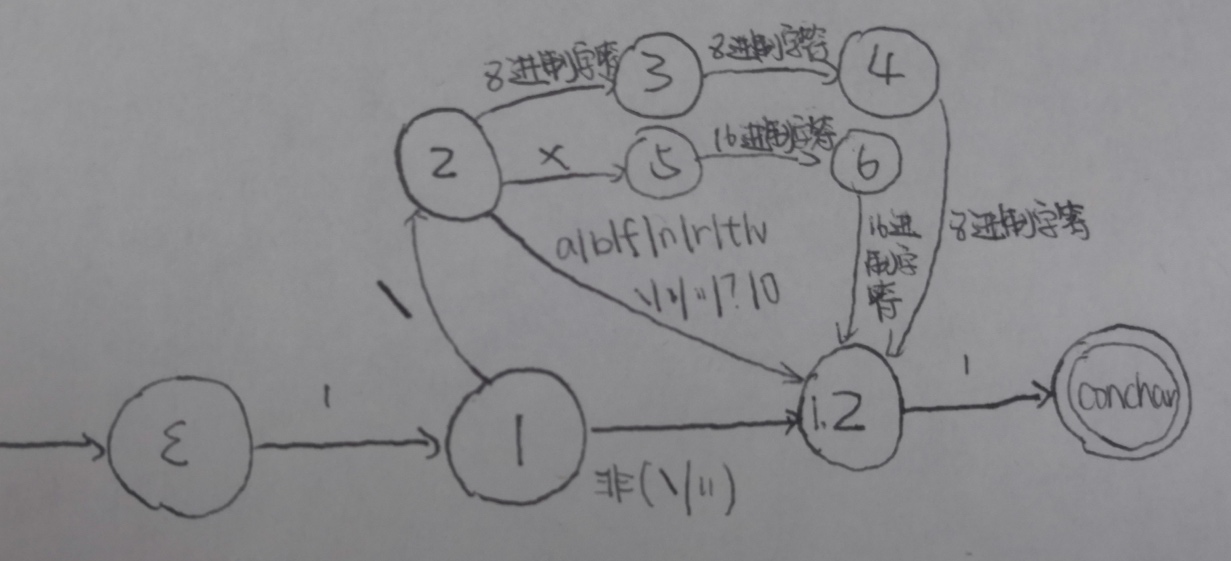
4.1.2 字符串不确定性有限自动机

终态为conchar



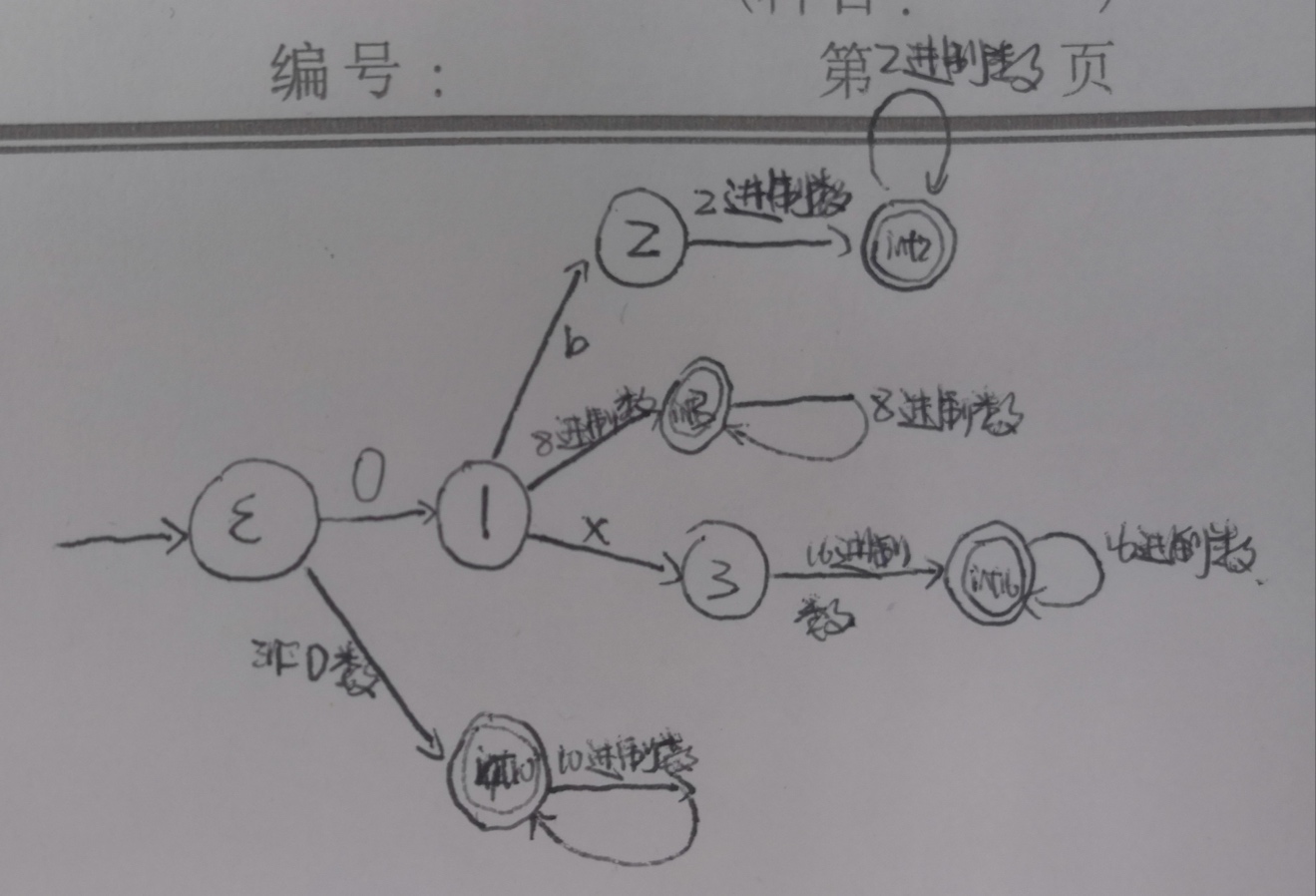
4.1.3 字符型不确定性有限自动机

终态：conchar



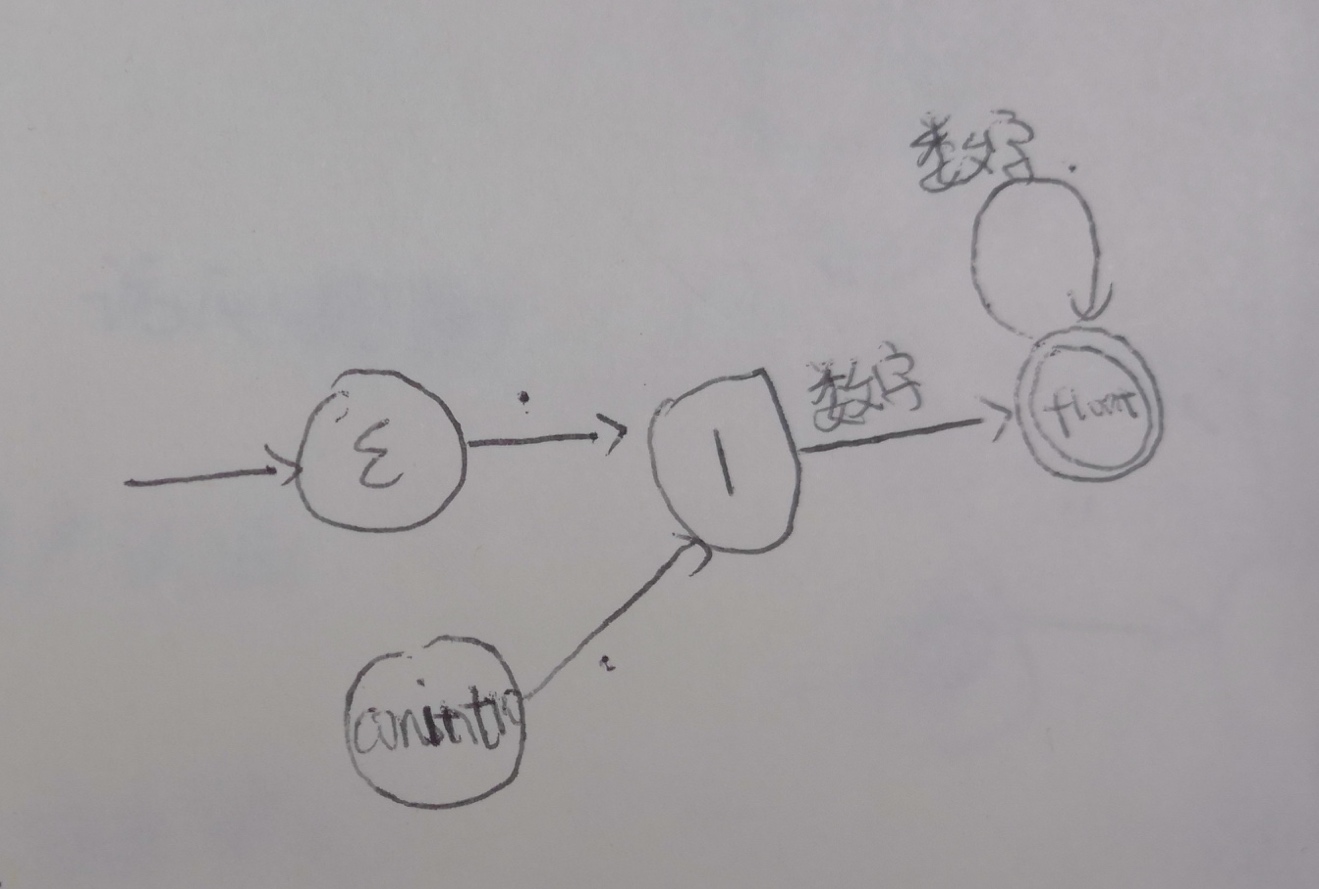
4.1.4 整形不确定性有限自动机

终态：conint2、conint8、conint16、conint10



符点型非确定性有限自动机

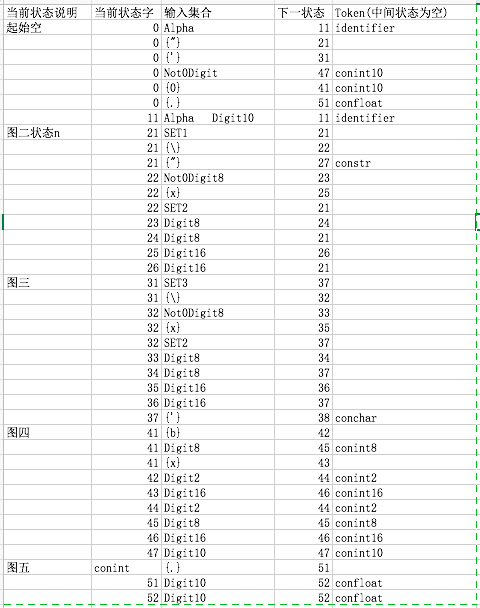
终态：confloat

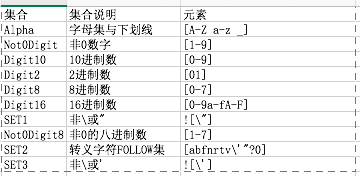


4.2确定性有限自动机设计

由于将所有NFA进行确定化集中在一个DFA中，DFA图将非常的庞大。因此，我们使用一个状态表来表示DFA的状态设计

具体请看词法分析状态转移表





4.3 词法分析器设计

4.3.1 设计原理

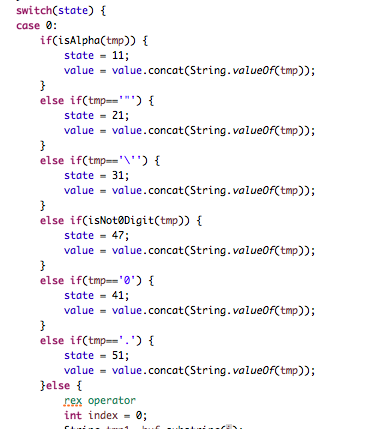
词法分析器本质上是一个自动机，有了状态转移表后这个自动机的实现就非常容易了。实现了DFA后，就可以识别出Token的种类有identifier、constr、conchar、conint2、conint8、conint10、conint16、confloat。

识别出了identifier后再依次匹配是否为保留字，如果是true或false则Token为conbool，如果是NULL则为connull，如果是sizeof则为operator，其它都为关键字

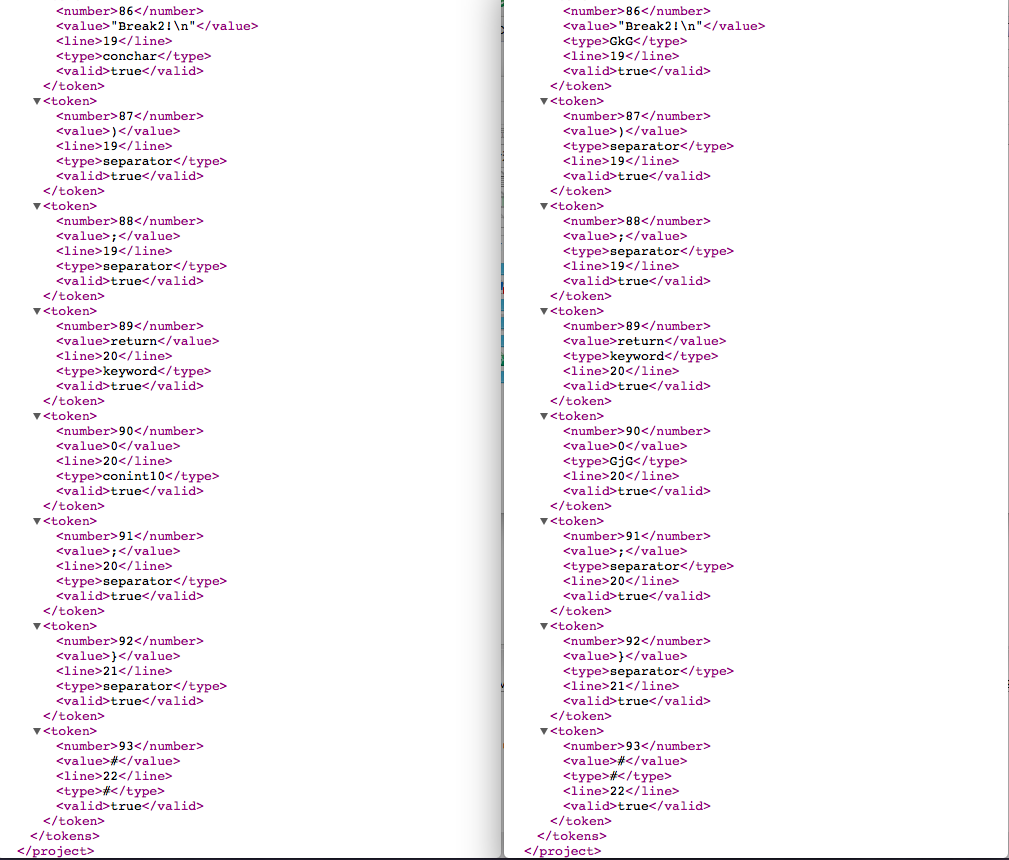
如果没有被自动机识别，则匹配是否为separator与operator。

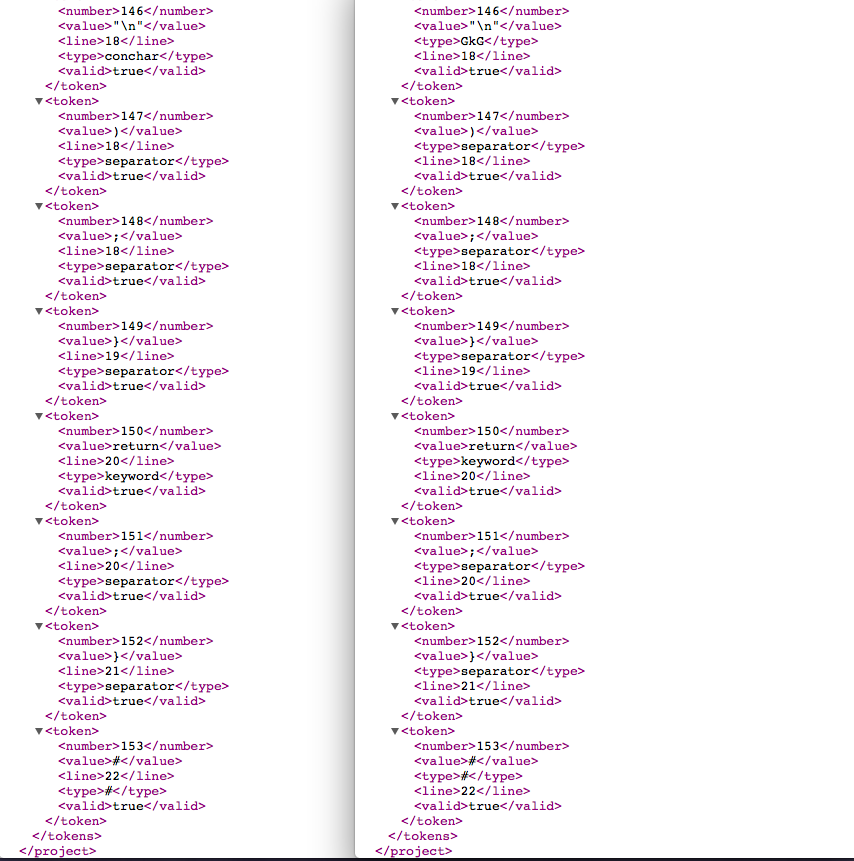
4.3.2 程序实现与结果

本程序使用状态机实现，实现完全依托于上面的状态转移表。这里就不再赘述。



程序运行结果除一些token的type类型为我自定义的类型，其他完全一致





**5. 实验心得体会**

我的词法分析器考虑了绝大多数C语言的词法，也有一些极为偏门的词法如L123这种长整形变量或U123这种无符号数未予考虑，但是倘若以后要用到，可以很容易的将这些词法加上去。而且词法分析并不是完全独立的，它需要为语法分析提供方便，因此倘若以后做语法分析时，可能会对词法分析器做出一些修改。