实验二、 自动生成词法分析程序（ JFlex）

19335019 陈泷

Oberon-0 语言的词汇表

以表格形式列出 Oberon-0 语言的词汇表：

类别	内容
数值常量	(1-9)+(1-9)* | 0(0-7)*
算术运算符	+, -, *, DIV, MOD
关系运算符	=, #, <, <=, >, >=
逻辑运算符	&, OR, ~
其他运算符	:= , : , (,) , [,]
标识符	letter {letter | digit}*
标点符号	; , .
保留字	MODULE, PROCEDURE, BEGIN, END, IF, THEN, ELSIF, ELSE, WHILE, DO, OF
关键字	INTEGER, BOOLEAN, Read, Write, WriteLn, CONST, ARRAY, VAR, TYPE, RECORD
注释	(* , *)

（1）单词分类的理由：主要依据在Oberon-0 语言充当的角色进行分类。 另外，以下四类其实可以归属为运算符这一大类，为了精细划分才拆分为4类。

运算符类	内容
算术运算符	+, -, *, DIV, MOD
关系运算符	=, #, <, <=, >, >=
逻辑运算符	&, OR, ~
其他运算符	:= , : , (,) , [,]

（2）对于保留字与关键字的分类：主要还是依据实验一中的思路，与程序功能性相关的归属到关键字，与划分程序结构层次相关的归属到保留字。

Oberon-0 语言的词法规则

用正则定义式描述 Oberon-0 语言词法规则； 若你使用纯文本书写正则定义式， 其中的元符号“定义为”使用“->”表示，空串用“epsilon”表示。

```
digit -> [0-9]
letter -> [a-z A-Z]

Number -> Decimal | Octal
Decimal -> [1-9](digit)*
Octal -> 0[0-7]*

Identifier -> letter( letter | digit )*

ReservedWord ->  "module" | "procedure" | "begin" | "end" | "if" | "then" | "elseif" | "else" | "while" | "do" | "of"
Keyword -> "Integer" | "write" | "read" | "writeln" | "boolean" | "const" | "arary" | "var" | "type" | "record"

Comment -> "(*"  ([^\*] | "*"+[^\)])*  "*)"

Punctuation -> ";" | ","

Operator -> "+"|"-"|"*"|"div"|"mod"|":="| "="|"#"|">"|"<"|">="|"<="| "("|")"|"&"|"or"|"~"|"["|"]"|"."|":"
```

Oberon-0 语言与其他高级语言的词法规则的异同比较

与 Pascal 、 C/C++、 Java 等常见高级程序设计语言的词法规则相比， Oberon-0语言的词法规则有何异同?

（1）不同之处

- 部分运算对应的运算符不同：例如 or表示或， ~表示非， :=为赋值， DIV为除法， MOD取余，# 表示不相等等。
- 注释的方式不同。
- 标识符中不可以有下划线
- 数据只能是整数类型和Boolean类型，且不支持书写 BOOLEAN 类型的常量 ，没有浮点数类型、字符串类型等其他类型。

（2）相同之处：除了上述提及的区别之外，其他的词法规则与 Pascal 、 C/C++、 Java 等常见高级程序设计语言的词法规则基本保持一致。

3 种不同 lex 族软件工具的输入文件中， 词法规则定义的差异或特点

1. 运行的环境不同： GUN Flex 运行环境是 C 语言平台，jFlex 和 JLex 运行环境是 java 平台。
2. 词法和语法规则上存在差异。
3. jflex 与 jlex 需要将代码写入 `%{` 和 `%}` 之间；gnu flex 则直接编写代码，没有另外的格式要求。
4. 源文件的结构不同：jflex 与 jlex分为用户代码、选项声明、词法规则三个部分，GNU Flex 则分为定义

   段 (definitions) 、规则段 (rules)、用户代码段 (user code)三部分。

生成 Oberon-0 语言的词法分析程序

(1) 测试实验一中自己写的源代码`account.obr`：

```
ReservedWord : MODULE
Identifier : program
PUNCTUATION : ;
COMMENT : (* define the type--AccountRecord *)
Keyword : TYPE
Identifier : AccountRecord
OPERATOR : =
Keyword : RECORD
Identifier : id
OPERATOR : :
Keyword : INTEGER
PUNCTUATION : ;
Identifier : salary
OPERATOR : :
Keyword : INTEGER
ReservedWord : END
PUNCTUATION : ;
COMMENT : (*Find the max salary among n accounts *)
ReservedWord : PROCEDURE
Identifier : FindMax
PUNCTUATION : ;
Keyword : VAR
Identifier : x
PUNCTUATION : ,
Identifier : n
PUNCTUATION : ,
Identifier : i
OPERATOR : :
Keyword : INTEGER
PUNCTUATION : ;
Identifier : accounts
OPERATOR : :
Keyword : ARRAY
NUMBER : 100
ReservedWord : of
Identifier : AccountRecord
PUNCTUATION : ;
ReservedWord : BEGIN
Keyword : Read
OPERATOR : (
Identifier : n
OPERATOR : )
PUNCTUATION : ;
Identifier : i
OPERATOR : :=
NUMBER : 1
PUNCTUATION : ;
Identifier : x
OPERATOR : :=
NUMBER : 0
PUNCTUATION : ;
ReservedWord : if
Identifier : n
OPERATOR : >
NUMBER : 0
```

```
ReservedWord : THEN
Identifier : accounts
OPERATOR : [
NUMBER : 0
OPERATOR : ]
PUNCTUATION : .
Identifier : id
OPERATOR : =
NUMBER : 0
PUNCTUATION : ;
Identifier : x
OPERATOR : :=
Keyword : Read
OPERATOR : (
Identifier : accounts
OPERATOR : [
NUMBER : 0
OPERATOR : ]
PUNCTUATION : .
Identifier : salary
OPERATOR : )
ReservedWord : END
PUNCTUATION : ;
ReservedWord : WHILE
OPERATOR : (
Identifier : i
OPERATOR : <
Identifier : n
OPERATOR : &
Identifier : i
OPERATOR : <
NUMBER : 100
OPERATOR : )
ReservedWord : DO
Identifier : accounts
OPERATOR : [
Identifier : i
OPERATOR : ]
PUNCTUATION : .
Identifier : id
OPERATOR : =
Identifier : i
PUNCTUATION : ;
Keyword : Read
OPERATOR : (
Identifier : accounts
OPERATOR : [
Identifier : i
OPERATOR : ]
PUNCTUATION : .
Identifier : salary
OPERATOR : )
PUNCTUATION : ;
ReservedWord : if
Identifier : accounts
OPERATOR : [
Identifier : i
OPERATOR : ]
```

```
PUNCTUATION : .
Identifier : salary
OPERATOR : >
Identifier : x
ReservedWord : THEN
Identifier : x
OPERATOR : :=
Identifier : accounts
OPERATOR : [
Identifier : i
OPERATOR : ]
PUNCTUATION : .
Identifier : salary
ReservedWord : ELSE
Identifier : x
OPERATOR : :=
Identifier : x
ReservedWord : END
PUNCTUATION : ;
Identifier : i
OPERATOR : :=
Identifier : i
OPERATOR : +
NUMBER : 1
PUNCTUATION : ;
ReservedWord : END
PUNCTUATION : ;
Keyword : Write
OPERATOR : (
Identifier : x
OPERATOR : )
PUNCTUATION : ;
Keyword : Write
OPERATOR : (
Identifier : n
OPERATOR : )
PUNCTUATION : ;
Keyword : Write
OPERATOR : (
Identifier : i
OPERATOR : )
PUNCTUATION : ;
Keyword : WriteLn
ReservedWord : END
Identifier : FindMax
PUNCTUATION : ;
ReservedWord : END
Identifier : program
PUNCTUATION : .
EOF :
Scanning Finished. No Error Found.
```

(2) 测试词法错误的类型以及测试结果：

1. Illegal Symbol Exception：识别单词时遇到不合法的输入符号(譬如@、 $等符号)则抛出该异常。
   出错内容：`sal@ary: INTEGER;`

```
Identifier : sal
..\src\testcases\account.001 : Compiler error. (details:Lexical error. (details:IllegalSymbol error. (details:IllegalSym
bol error.)))
Line 19, Colume 34: @
```

2. Illegal Octal Exception：当 0 开头的整数常量中含有 0~7 之外的符号(包括 8 和 9或其他字母)时抛出该异常。出错内容：`x := 08;`

```
Identifier : x
OPERATOR : :=
..\src\testcases\account.002 : Compiler error. (details:Lexical error. (details:IllegalOctal error. (details:IllegalOcta
l error.)))
Line 16, Colume 6: 08
```

3. 含有不合法的常量：常量中数字的个数超出限制。出错内容：`i := 10000000000000000000000000000001;`

```
Identifier : i
OPERATOR : :=
..\src\testcases\account.003 : Compiler error. (details:Lexical error. (details:IllegalIntegerRange error. (details:Ille
galIntegerRange error.)))
Line 15, Colume 6: 10000000000000000000000000000001
```

4. 标识符长度不允许超过 24 个字符。出错内容：`VAR x, n, izxcvbnmasdfghjklqwert: INTEGER;`

```
Keyword : VAR
Identifier : x
PUNCTUATION : ,
Identifier : n
PUNCTUATION : ,
..\src\testcases\account.004 : Compiler error. (details:Lexical error. (details:IllegalIdentifierLength error. (details:
IllegalIdentifierLength error.)))
Line 11, Colume 11: izxcvbnmasdfghjklqwertyuiop
```

实验心得与体会

实验二内容相比实验一多了一些。通过实验二，我熟悉了 jflex 的使用，了解了从生成词汇表到确定词法规则，再到利用现有工具生成词法分析器的生成过程。另外，通过与其他高级语言的比较，也加深了对几种语言的认识与辩证性思考。总之，在这一过程中有一定的收获。