自动生成词法分析程序（JFlex）

Oberon-0 语言的词汇表

类别	预定义单词
保留字	MODULE, BEGIN, END, CONST, TYPE, VAR, PROCEDURE, ARRAY, OF, RECORD, IF, THEN, ELSE, ELSIF, WHILE, DO
关键字	INTEGER,BOOLEAN,READ,WRITE,WRITELN
标识符	以字母开头，由字母和数字组成的标识符
常量	
数字常量	十进制或八进制整数
布尔常量	TRUE, FALSE
运算符	
关系运算符	=, <, >, <=, >=,#
算术运算符	+,-,*, DIV,MOD
逻辑运算符	& ,OR , ~
其他符号	
赋值	:=
类型定义	:
括号	(,),[,]
分隔符	;, , , .
注释	(* {不含*）的注释内容} *)

在处理保留字和关键字时，通常需要在词法分析阶段就将它们与标识符进行区分。保留字是语言规范中预定义的关键字，具有特殊含义，不能用作标识符；而关键字是具有特殊意义但可以用作标识符的单词。通常需要在解析标识符时检查是否为关键字，避免与保留字冲突

抽取 Oberon-0 语言的词法规则

将词法定义从 Oberon-0 语言的 BNF 中分离出来，并写成正则定义式

```
digit   -> [0-9]
letter  -> [a-z]|[A-Z]
iteger  -> [1-9](digit)* | 0 ([0-7])*
identifier -> letter (letter|digit)*
```

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

- 标识符命名规则基本相同，都是由字母、数字和下划线组成，且必须以字母或下划线开头，不能以数字开头
- 都具有整数、布尔、数组等数据类型，但Oberon-0参与算术表达式的运算量必须是 INTEGER 类型，否则会产生类型不兼容错误，且Oberon-0 不支持实数除法运算（仅支持整除运算），算术表达式的运算结果总是 INTEGER 类型
- 都使用常见的运算符，比如`+`、`-`、`*`、`=`、`<`、`>`，不同的是Oberon-0的除法使用`DIV`，取模使用`MOD`
- Oberon-0使用的注释为`(*`、`*)`

总的来看不同语言的语法规则都有一定相似性，同时包含着自己独特的一些语法

下载词法分析程序自动生成工具 JFlex

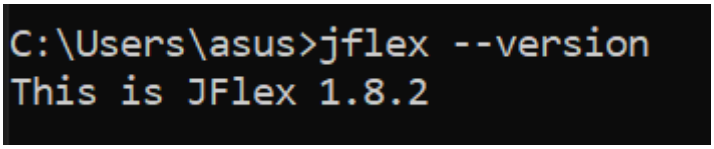
在Releases · jflex-de/jflex (github.com)下载`JFlex 1.8.2`，修改`bin/jflex.bat`文件如下：

```
@echo off
REM Please adjust JFLEX_HOME and JFLEX_VERSION to suit your needs
REM (please do not add a trailing backslash)

if not defined JFLEX_HOME set JFLEX_HOME=E:\jflex-1.8.2
if not defined JFLEX_VERSION set JFLEX_VERSION=1.8.2
set JAVA_HOME=E:\Java\jdk-21

java -Xmx128m -jar "%JFLEX_HOME%\lib\jflex-full-%JFLEX_VERSION%.jar" %*
```

将`\jflex-1.8.2\bin`加入环境变量，打开命令行输入`jflex --version`验证安装成功：



生成 Oberon-0 语言的词法分析程序

直接使用lab2实验的exceptions，加入有关Oberon-0的异常类。

编写主程序`LexicalAnalysis`：

```
public class LexicalAnalysis {
    /**
     * 词法分析器的入口
     * @param args 读入oberon-0程序
     * @throws Exception 抛出异常
     */
    public static void main(String[] args) throws Exception {
        // jflex生成的OberonScanner类
        OberonScanner scanner = new OberonScanner(new
java.io.FileReader(args[0]));
        int flag = 0;
        while (!scanner.yyatEOF()) {
            try {
                String lex = scanner.yylex();
                // 无异常且未扫描结束则打印词法分析结果
                if(lex != null){
                    String text = String.format("%-25s",scanner.yytext());
```

```
                    System.out.println(text + lex);
                }
            } catch (LexicalException e) {
                System.out.print("########## Error happen ##########\n");
                System.out.print(scanner.yytext() + " : " + e + "\n");
                flag += 1;
            }
        }
        System.out.println("Lexical analysis done. With " + flag +" lexical
error");
    }
}
```

`oberon.jflex` 实现如下：

```
import java.io.*;
import exceptions.*;

%%

%public
%class OberonScanner
%ignorecase
%unicode
%type String
%line
%column
%yylexthrow LexicalException

ReservedWord =
"module"|"begin"|"end"|"const"|"type"|"var"|"procedure"|"array"|"of"|"record"|"i
f"|"then"|"else"|"elsif"|"while"|"do"|"or"|"div"|"mod"
Keyword = "integer"|"boolean"|"read"|"write"|"writeln"
Operator = "="|"#"|"<"|"<="|">"|">="|"*"|"div"|"mod"|"+"|"-
"|"&"|"or"|"~"|":="|":"|"("|")"|"["|"]"
Delimiter = ";"|","|"."
Comment = "(*"~"*)"
Identifier = [a-zA-Z_][a-zA-Z0-9_]*
Integer = 0[0-7]* | [1-9]+[0-9]*

IllegalOctal = 0[0-7]*[8-9]+[0-9]*
IllegalInteger  = {Integer}+{Identifier}+
MismatchedComment= "(*" ([^\*]|"*"+[^\)])* | ([^\(]|"("+[^\*])* "*)"
WhiteSpace  = " "|\r|\n|\r\n|\t

%%

<YYINITIAL>
{
    {ReservedWord}  {return "ReservedWord";}
    {Keyword}       {return "Keyword";}
    {Operator}      {return "Operator";}
    {Delimiter}     {return "Delimiter";}
    {Comment}       {return "Comment";}
    {Identifier}    {
```

```
                        if (yylength() > 24)
                            throw new IllegalIdentifierLengthException();
                        else
                            return "Identifier";
                    }
    {Integer}       {
                        if (yylength() > 12)
                            throw new IllegalIntegerRangeException();
                        else
                            return "Integer";
                    }
    {IllegalOctal}      {throw new IllegalOctalException();}
    {IllegalInteger}    {throw new IllegalIntegerException();}
    {MismatchedComment} {throw new MismatchedCommentException();}
    {WhiteSpace}        {}
     .                  {throw new IllegalSymbolException();}
}
```

点击`gen.bat`生成OberonScanner类，点击`build.bat`在bin文件夹生成class文件

运行正确程序点击`run.bat`如下：

C:\Windows\system32\cmd.exe

```
:=                      Operator
0                       Integer
TO                      Identifier
maxNumber               Identifier
-                       Operator
1                       Integer
DO                      ReservedWord
factorials              Identifier
[                       Operator
n                       Identifier
]                       Operator
:=                      Operator
Factorial               Identifier
(                       Operator
n                       Identifier
)                       Operator
END                     ReservedWord
;                       Delimiter
(* 输出阶乘数组 *)            Comment
PrintArray              Identifier
(                       Operator
factorials              Identifier
,                       Delimiter
maxNumber               Identifier
)                       Operator
END                     ReservedWord
FactorialExample        Identifier
.                       Delimiter
Lexical analysis done. With 0 lexical error
Press any key to continue . . .
```

正常点击运行`test.bat`如下：

C:\Windows\system32\cmd.exe

```
0                       Integer
TO                      Identifier
maxNumber               Identifier
-                       Operator
1                       Integer
DO                      ReservedWord
factorials              Identifier
[                       Operator
n                       Identifier
]                       Operator
:=                      Operator
Factorial               Identifier
(                       Operator
n                       Identifier
)                       Operator
END                     ReservedWord
;                       Delimiter
(* 输出阶乘数组 *)            Comment
PrintArray              Identifier
(                       Operator
factorials              Identifier
,                       Delimiter
maxNumber               Identifier
)                       Operator
END                     ReservedWord
FactorialExample        Identifier
.                       Delimiter
Lexical analysis done. With 1 lexical error
================================================
Press any key to continue . . .
```

注释掉正常词法分析过程，只显示词法错误信息，打包后点击`test.bat`结果如下：可以看到报错结果符合预期

```
Running Testcase 001: IllegalSymbolException
================================================
########## Error happen ##########
@ : exceptions.IllegalSymbolException: Unknown character.
Lexical analysis done. With 1 lexical error
================================================
Press any key to continue . . .

C:\Users\asus\Desktop\大三下\编译原理\lab_3\21307347陈欣宇\ex2>call test002.bat
Running Testcase 002: IllegalIntegerException
================================================
########## Error happen ##########
1A : exceptions.IllegalIntegerException: Illegal Integer, no blank between interger and letters.
Lexical analysis done. With 1 lexical error
================================================
Press any key to continue . . .

C:\Users\asus\Desktop\大三下\编译原理\lab_3\21307347陈欣宇\ex2>call test003.bat
Running Testcase 003: IllegalIntegerRangeException
================================================
########## Error happen ##########
1234567890123 : exceptions.IllegalIntegerRangeException: Illegal IntegerRange: more than 12.
Lexical analysis done. With 1 lexical error
================================================
Press any key to continue . . .
```

```
C:\Users\asus\Desktop\大三下\编译原理\lab_3\21307347陈欣宇\ex2>call test004.bat
Running Testcase 004: IllegalOctalException
==========================================
########## Error happen ##########
01289 : exceptions.IllegalOctalException: Illegal Octal number.
Lexical analysis done. With 1 lexical error
==========================================
Press any key to continue . . .

C:\Users\asus\Desktop\大三下\编译原理\lab_3\21307347陈欣宇\ex2>call test005.bat
Running Testcase 005: IllegalIdentifierLengthException
==========================================
########## Error happen ##########
resultaaaaaaaaaaaaaaaaaaaaaaaa : exceptions.IllegalIdentifierLengthException: Illegal Identifier Length: more than 24.
Lexical analysis done. With 1 lexical error
==========================================
Press any key to continue . . .

C:\Users\asus\Desktop\大三下\编译原理\lab_3\21307347陈欣宇\ex2>call test006.bat
Running Testcase 001: MismatchedCommentException
==========================================
########## Error happen ##########

 计算整数 n 的阶乘 *) : exceptions.MismatchedCommentException: Mismatched Comment.
Lexical analysis done. With 1 lexical error
==========================================
Press any key to continue . . .
```

讨论不同词法分析程序生成工具的差异

比较以下 3 种流行的词法分析程序自动生成工具之间的差异：JFlex、JLex 和 GNU Flex。主要讨论这些软件工具接收输入源文件时，在词法规则定义方面存在的差异。

- JFlex 是基于 Java 编写的用于生成 Java 词法分析器的工具，支持 Unicode 和复杂的正则表达式，与 Java 的整合度高，输入文件格式是 .flex 文件。支持正则表达式和扩展正则表达式（包括 Unicode 字符类）。支持多种动作和状态管理，可以嵌入 Java 代码。
- JLex 也是用于 Java 的词法分析器生成器，类似 JFlex，但功能上不如 JFlex 强大。输入文件格式是 .lex 文件，支持正则表达式，但只支持 ASCII，不支持 Unicode，语法与GNU Flex类似，但生成的是Java代码。
- GNU Flex 是一个用于生成 C 语言词法分析器的工具，广泛用于 Unix 和 Linux 系统中，用于生成高效的 C 语言词法分析器，输入文件格式是 .l 文件，也支持正则表达式和扩展正则表达式。不支持 Unicode，仅支持 ASCII。支持 C 代码嵌入，通常与 Bison（生成语法分析器）一起使用。

总的来看，JFlex 适合复杂的 Java 项目；JLex 简单易用，适合从 C/C++ 过渡到 Java 的项目；GNU Flex 与 Bison 配合良好，适合 C 项目特别是 Unix/Linux 环境中的项目。