

编译器构造实验

Lab5—实验 2

自动生成词法分析程序 (JFlex)

姓名：郝裕玮

班级：计科 1 班

学号：18329015

目录

1 编写一个正确的 Oberon-0 源程序	3
2 编写上述 Oberon-0 源程序的变异程序	4
3 讨论 Oberon-0 语言的特点	5
4 讨论 Oberon-0 文法定义的二义性	6
5 实验心得	15

1 总结 Oberon-0 语言的词汇表

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

单词分类的理由：根据单词本身起到的功能和作用来进行分类即可。

区分保留字和关键字的方法：正如我在实验 1 的 Oberon-0 报告中所说：

3 讨论 Oberon-0 语言的特点

保留字与关键字的区别：

- (1) 保留字是程序中预先定义的特殊的字或词，程序员不能再将这些字作为变量名或者过程名使用。主要用于对不同程序块进行组织划分。
- (2) 关键字是编译器保留使用的字或词，主要用于提供部分预定的函数和变量。

所以保留字的主要功能是对代码块进行范围划分（例如 MODULE, PROCEDURE, BEGIN, END, IF, ELSE 等都是用于范围界定），关键字的主要功能是为提供部分预定的函数和变量（例如 INTEGER, Read, Write 都是具有特定功能，而非用于代码范围划分）。

2 抽取 Oberon-0 语言的词法规则

- Keyword \rightarrow "INTEGER" | "BOOLEAN" | "CONST" | "TYPE" | "VAR" | "RECORD" | "ARRAY" | "Read" | "Write" | "WriteLn"
- ReservedWord \rightarrow "MODULE" | "PROCEDURE" | "OF" | "BEGIN" | "END" | "IF" | "THEN" | "ELSE" | "ELSIF" | "WHILE" | "DO"
- Decimal \rightarrow [1-9][0-9]*
- Octal \rightarrow 0[0-7]*
- Number \rightarrow Decimal | Octal
- Operator \rightarrow "+" | "-" | "*" | "DIV" | "MOD" | "&" | "OR" | "~" | "=" | "#" | ">" | ">=" | "<" | "<=" | ":=" | "[" | "]" | "." | "(" | ")" | ":"
- Letter \rightarrow [a-zA-z]
- Digit \rightarrow [0-9]
- Identifier \rightarrow Letter(Letter | Digit)*
- Punctuation \rightarrow ";" | ","
- Comment \rightarrow "(*" [^*] ~"*)" | "(*" "*" + ")"

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的异同点：

(1) Oberon-0 不区分变量名大小写。JAVA、C/C++区分变量名大小写。

(2) Oberon-0 的不同过程 PROCEDURE 使用 BEGIN 和 END 来区分。Java、C/C++中不同代码块用{}来区分。

(3) Oberon-0 没有浮点数，只支持整除和取余。而 Pascal 、 C/C++、 Java 等常见高级程序设计语言支持浮点数除法。

(4) Oberon-0 的不等于是"#"， 而 Pascal 、 C/C++、 Java 等常见高级程序设计语言的不等于是"!="。

(5) Oberon-0 的注释方法为(* *)。而 Pascal 、 C/C++、 Java 等常见高级程序设计语言的注释方法为"/"。

3 下载词法分析程序自动生成工具 JFlex

验证 JFlex 是否安装成功：

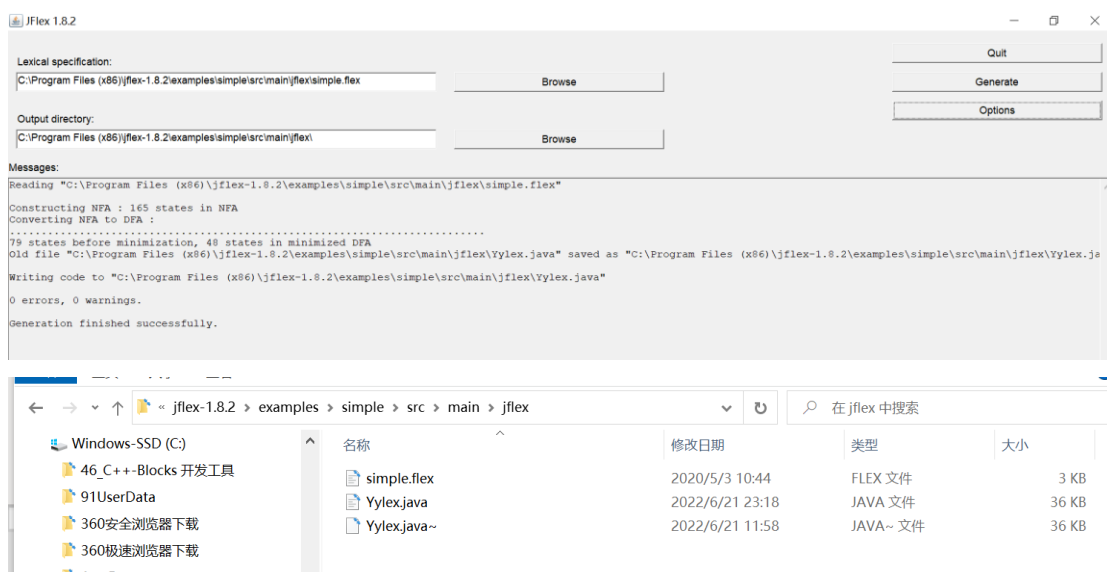


```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [版本 10.0.19044.1766]
(c) Microsoft Corporation。保留所有权利。

C:\Users\93508>jflex --version
Picked up JAVA_TOOL_OPTIONS: -Dfile.encoding=UTF-8
This is JFlex 1.8.2

C:\Users\93508>_
```

运行 JFlex 附带的输入源文件例子：



易知成功生成 Java 语言的词法分析程序源代码。

4 生成 Oberon-0 语言的词法分析程序

JFlex 输入源文件代码如下所示（包含注释）：

```
//用户代码
import java.io.*;
import exceptions.*;

//选项与声明
%%

//定义生成词法分析器 Java 文件的文件名
%class OberonScanner
//使生成的类是 public
%public
//设置扫描函数的返回类型
%type String
//使扫描函数声明抛出异常
%yylexthrow LexicalException
//其中用户代码部分直接被复制到扫描函数中，并且在每次文件结束时执行。这个用户代码应该返回表示文件结束的值
%eofval{
    return "EOF";
}
%eofval}
//支持字符集，防止溢出
```

```

%unicode
//行计数器, yyline 记录当前行数
#line
//列计数器, yycolumn 记录当前列数
%column
//不区分字符大小写, 符合 oberon-0 的词法规则
%ignorecase

//获取当前行数或列数
%{
    int get_line(){ return yyline;}
    int get_column(){ return yycolumn;}
}%

//正则表达式
Keyword = "INTEGER" | "BOOLEAN" | "CONST" | "TYPE" | "VAR" | "RECORD" |
"ARRAY" | "Read" | "Write" | "WriteLn"
ReservedWord = "MODULE" | "PROCEDURE" | "OF" | "BEGIN" | "END" | "IF" |
"THEN" | "ELSE" | "ELSIF" | "WHILE" | "DO"
Decimal = [1-9][0-9]*
Octal = 0[0-7]*
Number = {Decimal} | {Octal}
Operator = "+" | "-" | "*" | "DIV" | "MOD" | "&" | " OR " | "~" | "=" |
"#" | ">" | ">=" | "<" | "<=" | ":@" | "[" | "]" | "." | "(" | ")" |
":"
Identifier = [:jletter:][:jletterdigit:]*
Punctuation = ";" | ","
Comment = "(*" [^*] ~"*)" | "(*" "*" + ")"
WhiteSpace = " " | \r | \n | \r\n | [ \t\f]
//异常情况的正则表达式
WrongInteger = {Number} + {Identifier} +
WrongOctal = 0[0-7]*[9|8][0-9]*
WrongComment = "(*" ([^*]|"*" + [^\\])*)*|([^\(|"(" + [^*])"*)"

//词法规则
%%
<YYINITIAL>{
    {Keyword}                {return "Keyword";}
    {ReservedWord}           {return "ReservedWord";}
    {Number}                  {
                                if(yylength() > 12){
                                    throw new
IllegalIntegerRangeException();
                                }

```

```

        else{
            return "Number";
        }

    }

    {Operator}
    {Identifier}
    {
        if(yylength() > 24){
            throw new
IllegalIdentifierLengthException();

        }
        else{
            return "Identifier";
        }
    }

    {Punctuation}
    {Comment}
    {WhiteSpace}
    {WrongInteger}
    {throw new
IllegalIntegerException();}
    {WrongOctal}
    {throw new
IllegalOctalException();}
    {WrongComment}
    {throw new
MismatchedCommentException();}
    {throw new
IllegalSymbolException(); }
}











```

在当前文件夹下运行 gen.bat 来生成词法分析程序

OberonScanner.java

18329015郝裕玮 > ex2 >

在 ex2 中搜索

名称	修改日期	类型	大小
 bin	2022/6/22 21:11	文件夹	
 jflex	2022/6/22 0:32	文件夹	
 lib	2022/6/22 21:10	文件夹	
 src	2022/6/22 21:49	文件夹	
 build.bat	2022/6/22 1:26	Windows 批处理文件	1 KB
 gen.bat	2022/6/22 18:57	Windows 批处理文件	1 KB
 lexgen.docx	2022/6/22 3:43	Microsoft Word 文档	269 KB
 readme.txt	2022/6/21 11:18	文本文档	1 KB
 run.bat	2022/6/22 2:13	Windows 批处理文件	1 KB
 test.bat	2022/6/22 2:10	Windows 批处理文件	1 KB


```
C:\WINDOWS\system32\cmd.exe
Picked up JAVA_TOOL_OPTIONS: -Dfile.encoding=UTF-8
Reading "..\src\oberon.flex"
Constructing NFA : 410 states in NFA
Converting NFA to DFA :
.....
168 states before minimization, 96 states in minimized DFA
Writing code to "..\src\OberonScanner.java"
请按任意键继续. . .
```

运行 build.bat 对词法分析程序进行编译

```
C:\WINDOWS\system32\cmd.exe
Picked up JAVA_TOOL_OPTIONS: -Dfile.encoding=UTF-8
请按任意键继续. . .
```

运行 run.bat 来处理我编写的正确 Oberon-0 例子程序：Test.obr，

以下截图仅展示部分分析结果：

```
C:\WINDOWS\system32\cmd.exe
C:\Users\93508\Desktop\18329015郝裕玮\ex2>cd bin
C:\Users\93508\Desktop\18329015郝裕玮\ex2\bin>java Main ../src/Test.obr
Picked up JAVA_TOOL_OPTIONS: -Dfile.encoding=UTF-8
../src/Test.obr
ReservedWord : MODULE
Identifier : Test
Punctuation : ;
Comment : (* 计算阶乘 *)
ReservedWord : PROCEDURE
Identifier : Factorial
Punctuation : ;
Keyword : VAR
Identifier : n
Punctuation : ,
Identifier : result
Operator : :
Keyword : INTEGER
Punctuation : ;
ReservedWord : BEGIN
Identifier : result
Operator : :=
Number : 1
```

```
Comment : (* 计算两数之和与两数之差 *)
ReservedWord : PROCEDURE
Identifier : AddSub
Punctuation : ;
Keyword : TYPE
Identifier : res
Operator : =
Keyword : INTEGER
Punctuation : ;
Identifier : sum
Operator : =
Keyword : RECORD
Identifier : a
Punctuation : ,
Identifier : b
Operator : :
Keyword : INTEGER
Punctuation : ;
Identifier : minus
```

```
Operator : ]
Operator : .
Identifier : d
Punctuation : ;
Keyword : Write
Operator : (
Identifier : addres
Operator : )
Punctuation : ;
Keyword : Write
Operator : (
Identifier : subres
Operator : )
Punctuation : ;
Keyword : WriteLn
ReservedWord : END
Identifier : AddSub
Punctuation : ;
ReservedWord : END
Identifier : Test
Operator : .
```

扫描完毕，该程序未出现词法错误！

再运行 test.bat 来检测各种异常的测试用例：

(1) IllegalSymbolException

当识别一个单词时遇到不合法的输入符号（譬如@、\$等符号）

则抛出该异常。

```
C:\WINDOWS\system32\cmd.exe

C:\Users\93508\Desktop\18329015郝裕玮\ex2>cd bin

C:\Users\93508\Desktop\18329015郝裕玮\ex2\bin>java Main ../src/testcases/Test.*
Picked up JAVA_TOOL_OPTIONS: -Dfile.encoding=UTF-8

..\src\testcases\Test.001

Comment : (* IllegalSymbolException *)
ReservedWord : MODULE
Identifier : Test
Punctuation : ;
Comment : (* 计算阶乘 *)
ReservedWord : PROCEDURE
Identifier : Factorial
Punctuation : ;
Keyword : VAR
Identifier : n
Punctuation : ,
Identifier : result
Operator : :
Keyword : INTEGER
Punctuation : ;
ReservedWord : BEGIN
Identifier : re

Line 6, Column 10: @ 存在异常
LexicalException :
Illegal Symbol.
Illegal Symbol.
```

(2) IllegalIntegerException

当整数常量（无论是十进制还是八进制） 与其后的标识符之间

无空白分隔时抛出该异常。

```
..\src\testcases\Test.002

Comment : (* IllegalIntegerException *)
ReservedWord : MODULE
Identifier : Test
Punctuation : ;
Comment : (* 计算阶乘 *)
ReservedWord : PROCEDURE
Identifier : Factorial
Punctuation : ;
Keyword : VAR

Line 4, Column 6: 1n 存在异常
LexicalException :
Illegal Integer, no blank between interger and letters.
Illegal Integer, no blank between interger and letters.
```

(3) IllegalIntegerRangeException

当识别出的整数常量（无论是十进制还是八进制）大于本文档约定的整数常量值最大限制时抛出此异常。

```
..\src\testcases\Test.003  
  
Comment : (* IllegalIntegerRangeException *)  
ReservedWord : MODULE  
Identifier : Test  
Punctuation : ;  
Comment : (* 计算阶乘 *)  
ReservedWord : PROCEDURE  
Identifier : Factorial  
Punctuation : ;  
Keyword : VAR  
Identifier : n  
Punctuation : ,  
Identifier : result  
Operator : :  
Keyword : INTEGER  
Punctuation : ;  
ReservedWord : BEGIN  
Identifier : result  
Operator : :=  
  
Line 6, Column 18: 111111111111 存在异常  
LexicalException :  
Illegal IntegerRange: more than 12.  
Illegal IntegerRange: more than 12.
```

(4) IllegalOctalException

当 0 开头的整数常量中含有 0~7 之外的符号（包括 8 和 9）时抛出该异常。（图见下页）

```

..\src\testcases\Test.004

Comment : (* IllegalOctalException *)
ReservedWord : MODULE
Identifier : Test
Punctuation : ;
Comment : (* 计算阶乘 *)
ReservedWord : PROCEDURE
Identifier : Factorial
Punctuation : ;
Keyword : VAR
Identifier : n
Punctuation : ,
Identifier : result
Operator : :
Keyword : INTEGER
Punctuation : ;
ReservedWord : BEGIN
Identifier : result
Operator : :=

Line 6, Column 18: 09 存在异常
LexicalException :
Illegal Octal number.
Illegal Octal number.

```

(5) IllegalIdentifierLengthException

当识别出的一个标识符长度超过最大限制时抛出该异常。

```

..\src\testcases\Test.005

Comment : (* IllegalIdentifierLengthException *)
ReservedWord : MODULE
Identifier : Test
Punctuation : ;
Comment : (* 计算阶乘 *)
ReservedWord : PROCEDURE
Identifier : Factorial
Punctuation : ;
Keyword : VAR
Identifier : n
Punctuation : ,
Identifier : result
Operator : :
Keyword : INTEGER
Punctuation : ;
ReservedWord : BEGIN

Line 6, Column 8: resultabcdefghijklmnopqrstuvwxyz 存在异常
LexicalException :
Illegal Identifier Length: more than 24.
Illegal Identifier Length: more than 24.

```

(6) MismatchedException

当 “(” 开头的注释直至扫描到最后一个符号都找不到配对的 “)” 时抛出该异常。

```
.. \src\testcases\Test.006

Comment : (* MismatchedException *)
ReservedWord : MODULE
Identifier : Test
Punctuation : ;

Line 2, Column 4: (* 计算阶乘 *)
PROCEDURE Factorial;
    VAR n,result: INTEGER;
BEGIN
    result := 1;
    Read(n);
    IF n = 0 THEN
        result := 1;
    END
    WHILE n >= 1 DO
        result := n * result;
        n := n - 1
    END;
    Write(result); WriteLn
END Factorial;

(* 计算两数之和与两数之差
PROCEDURE AddSub;
TYPE
    res = INTEGER;
    sum = RECORD
        a, b : INTEGER;
    minus = RECORD
        c, d : INTEGER;
    END;
VAR
    add: ARRAY 1 OF sum;
    sub: ARRAY 1 OF minus;
    addres, subres: res;
BEGIN
    READ(add[0].a);
    READ(add[0].b);
    READ(sub[0].c);
    READ(sub[0].d);
    addres := add[0].a + add[0].b;
    subres := sub[0].c - sub[0].d;
    Write(addres); Write(subres); WriteLn
END AddSub;
END Test. 存在异常
LexicalException :
Mismatched Comment.
Mismatched Comment.
```

综合上述 6 个词法异常的测试样例可知，JFlex 生成的 Java 词法分析程序可成功检测出实验要求的 6 种不同的词法异常。

5 讨论不同词法分析程序生成工具的差异

- (1) JFlex 和 JLex 生成 Java 语言，GNU Flex 生成 C 语言。所以这三者的词法规则必定有所不同（C 和 Java 的差异）。
- (2) 三者的代码书写规则有不同之处。
- (3) JFlex 和 JLex 将代码分为用户代码，选项与声明，词法规则。GNU Flex 将代码分为定义段(definitions)，规则段(rules)、用户代码段(user code)。

6 实验心得

通过实验 2 让我对 JFlex 的使用更加熟练，并对正则表达式和词法规则的结合有了更进一步的认识，对词法分析这一步骤也更加理解其本质。