

EX 2

20308003 曾伟超

词汇表

这里使用了一些扩展的正则表达，例如 `a-z` 表示所有的小写字母，`a-zA-Z` 表示所有字母

类型	正则表达
十进制数字	<code>(1-9)(0-9)*</code>
八进制数字	<code>0(0-7)*</code>
标识符	<code>(a-zA-Z)(a-zA-Z 0-9)*</code>
关键字(忽略大小写)	<code>integer boolean write writeln read</code>
保留字(忽略大小写)	<code>if elsif then else while begin do end of var const type procedure record array module</code>
标点	<code>, ;</code>
算数运算符	<code>+ - * div mod</code>
关系运算符	<code>> >= < <= = #</code>
逻辑运算符	<code>& or ~</code>
赋值运算符	<code>:=</code>
选择运算符	<code>\. []</code>
括号	<code>()</code>
类型声明	<code>:</code>
注释	<code>\(* . * ? * \)</code>

分类依据以及一些说明

关键字和保留字按照了 `ex1` 中的处理办法，将在语法中存在的作为保留字，其它的放入关键字

还有一些是我个人处理中存在些许疑惑的，例如 `div` 是放在算术运算符还是保留字中，`mod` 也是，但是最终考虑到一致性，还是将其放入了算数运算符中，这点对于后面的 `or` 也是相同的道理

而在注释，由于 `*` 和 `()` 本身在正则中有其含义，从而将 `*` 使用 `\` 做转义，`*` 的类似，加上 `?` 是为了采取非贪婪的匹配，防止出现注释嵌套，这在本次实验中是不允许的

词法规则

这里为了简化，将所有的运算符都统一使用 `Operator` 代替

$$\begin{aligned}
Number &\rightarrow DEC \mid OCT \\
DEC &\rightarrow 0(0-9)^* \\
OCT &\rightarrow (1-9)(0-9)^* \\
Identifier &\rightarrow (a-zA-Z)(a-zA-Z|0-9)^* \\
Marks &\rightarrow , \mid ; \\
Operator &\rightarrow + \mid - \mid * \mid div \mid mod \mid > \mid >= \mid < \mid <= \mid = \mid \# \mid \& \mid or \mid \sim \mid := \mid \backslash \cdot \mid [\mid] \mid \backslash (\mid \backslash) \mid : \\
Comment &\rightarrow \backslash (\backslash * . * ? \backslash * \backslash) \\
Keywords &\rightarrow integer \mid boolean \mid write \mid writeln \mid read \\
ReservedWords &\rightarrow if \mid elsif \mid then \mid else \mid while \mid begin \mid do \mid end \mid of \mid var \mid const \mid type \mid procedure \mid record \mid array \mid module
\end{aligned}$$

和高级语言的对比

1. 高级语言如 C/C++, Java 等, 其标识符的正则会更加的负责, 可以包含如下划线(`_`)存在, 而不是仅仅有字母和数字
2. 一些运算符的不同, 例如逻辑与在 C/C++ 中是 `&&`, `&` 在 C/C++ 中是按位与, 还有 `or` 在 C/C++ 中是 `||`, 不等号和相等也都有区别
3. 赋值的区别, 赋值在 C/C++ 等高级语言中是 `=`, 而在这里是 `:=`, 因为这里的 `=` 沿用了数学的相等
4. 注释的不同, 这里没有严格的区分单行和多行注释, 这里都是用统一的注释, 任意被括住的都是被注释部分
5. 类型符号, C/C++ 中没有这个类型符号, 靠的是类型前缀声明

JFlex

由于之前曾经使用过 `flex`, 而 `JFlex` 在使用上有很多相似的地方, 一些区别可以通过阅读文档很容易的区分, `JFlex` 文件分为三部分, 每两个部分之间使用 `%%` 进行分割, 从上到下依次为用户代码, 一些可选的选项设置, 以及最后的词法规则

用户代码部分, 主要包含了 `exceptions` 错误类, 用于抛出异常

这里由于大小写不敏感, 所以需要开启 `ignorecase`, 又由于后续需要使用 `javaCUP` 作为语法分析器, 需要开启 `cup`, 同时手动指定 `class` 为 `OberonScanner`

而对于各个操作符, 则是进行了区分, 实际上, 每一个无论是符号还是保留字还是关键字, 都会有独一无二的一个标识符用来区分, 具体可以参考 `TokenType.java`, 编写完成后进行测试, 以 `debug` 模式进行测试, 命令如下

```
./gen.sh src/oberon.flex && ./build.sh && ./run.sh ../ex1/testcases/fib.obr
```

将末尾的文件换成测试的文件, 如下图

```
..sysu/exp3/ex2
action [95] { return TokenType.tok_space; }
tok_space
line: 67 col: 9 match: --END--
action [48] { return TokenType.tok_end; }
tok_end
line: 67 col: 12 match: --\u000A--
action [95] { return TokenType.tok_space; }
tok_space
line: 68 col: 1 match: --END--
action [48] { return TokenType.tok_end; }
tok_end
line: 68 col: 4 match: -- --
action [95] { return TokenType.tok_space; }
tok_space
line: 68 col: 5 match: --Fib--
action [99] { if (yylength() > 24)
               throw new IllegalIdentifierLengthException();
               return TokenType.tok_identifier; }
tok_identifier
line: 68 col: 8 match: --.--
action [62] { return TokenType.tok_dot; }
tok_dot
line: 68 col: 9 match: --;--
action [64] { return TokenType.tok_semicolon; }
tok_semicolon
tok_eof

~/minijava_sysu/exp3/ex2 on main ?1 > py base at 03:41:22
```

可以看到，能够被成功的进行解析，之后，参考之前的变异，其中设计词法错误的为 001, 002, 012, 013, 014, 015，依次进行测试，如下图

001

```
..sysu/exp3/ex2
tok_space
line: 6 col: 5 match: --PROCEDURE--
action [46] { return TokenType.tok_procedure; }
tok_procedure
line: 6 col: 14 match: -- --
action [95] { return TokenType.tok_space; }
tok_space
line: 6 col: 15 match: --Fib--
action [99] { if (yylength() > 24)
               throw new IllegalIdentifierLengthException();
               return TokenType.tok_identifier; }
tok_identifier
line: 6 col: 18 match: --(--
action [65] { return TokenType.tok_lparen; }
tok_lparen
line: 6 col: 19 match: --n--
action [99] { if (yylength() > 24)
               throw new IllegalIdentifierLengthException();
               return TokenType.tok_identifier; }
tok_identifier
line: 6 col: 20 match: --@--
action [103] { throw new IllegalSymbolException(); }
Unexpected exception:
exceptions.IllegalSymbolException: Unknown Character Detected
    at OberonScanner.yylex(OberonScanner.java:899)
    at OberonScanner.main(OberonScanner.java:1317)

~/minijava_sysu/exp3/ex2 on main ?1 > py base at 03:44:30
```

符合报错预期

002

```
..sysu/exp3/ex2
line: 22 col: 7 match: -- --
action [95] { return TokenType.tok_space; }
tok_space
line: 22 col: 8 match: -- --
action [95] { return TokenType.tok_space; }
tok_space
line: 22 col: 9 match: --c--
action [99] { if (yylength() > 24)
               throw new IllegalIdentifierLengthException();
               return TokenType.tok_identifier; }
tok_identifier
line: 22 col: 10 match: -- --
action [95] { return TokenType.tok_space; }
tok_space
line: 22 col: 11 match: ==--
action [70] { return TokenType.tok_equal; }
tok_equal
line: 22 col: 12 match: -- --
action [95] { return TokenType.tok_space; }
tok_space
line: 22 col: 13 match: --019--
action [92] { throw new IllegalOctalException(); }
Unexpected exception:
exceptions.IllegalOctalException: Illegal Octal Detected
    at OberonScanner.yylex(OberonScanner.java:1066)
    at OberonScanner.main(OberonScanner.java:1317)

~/minijava_sysu/exp3/ex2 on main ?1 > py base at 03:46:24
```

符合报错预期

012

```
..sysu/exp3/ex2
action [95] { return TokenType.tok_space; }
tok_space
line: 22 col: 11 match: ==--
action [70] { return TokenType.tok_equal; }
tok_equal
line: 22 col: 12 match: -- --
action [95] { return TokenType.tok_space; }
tok_space
line: 22 col: 13 match: --019--
action [92] { throw new IllegalOctalException(); }
Unexpected exception:
exceptions.IllegalOctalException: Illegal Octal Detected
    at OberonScanner.yylex(OberonScanner.java:1066)
    at OberonScanner.main(OberonScanner.java:1317)

~/minijava_sysu/exp3/ex2 on main ?1 > ./run.sh ../ex1/testcases/fib.012 py base at 03:46:24
line: 1 col: 1 match: --(* Unmatched comment *)--
action [94] { return TokenType.tok_comment; }
tok_comment
line: 1 col: 24 match: --\u000A\u000AA Fib Calculate Program which is used to show how to write program in Obe
ron0 *)--
action [97] { throw new MismatchedCommentException(); }
Unexpected exception:
exceptions.MismatchedCommentException: Mismatched Comment Detected
    at OberonScanner.yylex(OberonScanner.java:1059)
    at OberonScanner.main(OberonScanner.java:1317)

~/minijava_sysu/exp3/ex2 on main ?1 > py base at 03:46:39
```

符合报错预期

013

```
..sysu/exp3/ex2
line: 9 col: 9 match: --IF--
action [49] { return TokenType.tok_if; }
tok_if
line: 9 col: 11 match: -- --
action [95] { return TokenType.tok_space; }
tok_space
line: 9 col: 12 match: --n--
action [99] { if (yylength() > 24)
              throw new IllegalIdentifierLengthException();
              return TokenType.tok_identifier; }
tok_identifier
line: 9 col: 13 match: -- --
action [95] { return TokenType.tok_space; }
tok_space
line: 9 col: 14 match: ----
action [70] { return TokenType.tok_equal; }
tok_equal
line: 9 col: 15 match: -- --
action [95] { return TokenType.tok_space; }
tok_space
line: 9 col: 16 match: --0b--
action [91] { throw new IllegalIntegerException(); }
Unexpected exception:
exceptions.IllegalIntegerException: Illegal Integer Detected
    at OberonScanner.yylex(OberonScanner.java:1073)
    at OberonScanner.main(OberonScanner.java:1317)

~/minijava_sysu/exp3/ex2 on main ?1 > py base at 03:46:54
```

符合报错预期

014

```
..sysu/exp3/ex2
tok_if
line: 9 col: 11 match: -- --
action [95] { return TokenType.tok_space; }
tok_space
line: 9 col: 12 match: --n--
action [99] { if (yylength() > 24)
              throw new IllegalIdentifierLengthException();
              return TokenType.tok_identifier; }
tok_identifier
line: 9 col: 13 match: -- --
action [95] { return TokenType.tok_space; }
tok_space
line: 9 col: 14 match: ----
action [70] { return TokenType.tok_equal; }
tok_equal
line: 9 col: 15 match: -- --
action [95] { return TokenType.tok_space; }
tok_space
line: 9 col: 16 match: --1234545667889--
action [88] { if (yylength() > 12)
              throw new IllegalIntegerRangeException();
              return TokenType.tok_decimal; }
Unexpected exception:
exceptions.IllegalIntegerRangeException: Integer Is Out of Range
    at OberonScanner.yylex(OberonScanner.java:986)
    at OberonScanner.main(OberonScanner.java:1317)

~/minijava_sysu/exp3/ex2 on main ?1 > py base at 03:47:04
```

符合报错预期

015

