编译原理 Lab1 实验报告

191250111 裴为东

1：实现功能

本程序完成了 Lab1 文档中的 5 个要求，能够识别出附录 A7.1.1 中的所有 tokens，并且可以识别 8 进制和 16 进制并以 10 进制输出，同时可以识别浮点数和正确的注释，并对无法识别的词给出错误提示。

2：实现过程

首先我先在 CmmLexer.g4 中编写了 7.11 中所有 tokens 的 antlr4 表达式，其中将 int 的 10 进制、8 进制、16 进制表示分开识别。

```
INT_10 : '0' | [1-9][0-9]*;
INT_8 : '0'[0-7]+;
INT_16 : '0'[xX][0-9a-fA-F]+;
```

在写 ID 的表达式时，需要将 return、struct、type 等特殊“ID”写在 ID 前面实现优先匹配

```
TYPE : 'int' | 'float';
RETURN : 'return';
STRUCT : 'struct';
IF : 'if';
ELSE : 'else';
WHILE : 'while';
ID : (CHAR | '_')(CHAR | '_' | [0-9])*;
```

多行注释的匹配需要用到非贪婪匹配，因要求中没有对嵌套注释做出报错要求，所以匹配多行注释时直接匹配最短的情况

```
SL_COMMENT : '//' .*? '\n' -> skip;
ML_COMMENT : '/*' .*? '*/' -> skip;
```

接着在 main 函数中读取文件内容，并传入上面构造的 lexer 对象中

```
CmmLexer lexer = new CmmLexer(CharStreams.fromFileName(path)){
```

同时需要对 lexer 中的 notifiListeners（）方法进行重载，以便可以自定义报错内容，err 为一个静态布尔类型变量，用于判断有无遇到无法识别的词

```
@Override
public void notifyListeners(LexerNoViableAltException e){
    String text = this._input.getText(Interval.of(this._tokenStartCharIndex, this._input.index()));
    String msg = "Error type A at Line " + this._tokenStartLine + ": Mysterious character '" + this.getErrorDisplay(text) + "'.";
    System.err.println(msg);
    err = true;
}
```

在没有报错的情况下，需要对识别的结果进行打印，识别之后的 tokens 列表可通过 getAllTokens()获取，其中整数部分通过 Integer.parseInt(str,radix)转换为 10 进制输出，浮点数通过 Stirng.format(“%.6f”,f)实现保留小数点后 6 位输出。

3：实验感受

印象最深的是刚开始做的时候每次修改 g4 文件的词法后都要重启 idea 才能变更修改。

一开始表达式写得不够精简，有很多可以简化的地方