Compilers-lab2 note

实现功能

- 本次实验实现了对`SysY`语言的语法分析和高亮，将语法元素和词法元素按照深度和先后顺序输出为一个语法树结构（如下图）。

```
Program
  CompUnit
    FuncDef
      FuncType
          int INT[orange]
        main IDENT[red]
      Block
    FuncDef
      FuncType
          void VOID[orange]
        fun IDENT[red]
      Block
    FuncDef
      FuncType
          int INT[orange]
        max IDENT[red]
      FuncFParams
        FuncFParam
          BType
              int INT[orange]
            a IDENT[red]
      Block
```

代码设计

- 在SysYParser.g4文件中，设计的特点是利用ANTLR的左递归语法特点，将`exp`和`cond`语法元素设计为左递归的形式。
- 在`Visitor`类中，新增了`displayLexer()`和`displayParser()`两个方法，分布用来输出语法单元和词法单元，因为两个单元的大小写和高亮不同，所以不能合并为一个方法。

```java
public void displayParser(String text, int depth){
  for (int i = 1; i < depth; i++) {
    System.err.print("  ");
  }
  System.err.println(text.substring(0,1).toUpperCase() + text.substring(1));
}
```

```java
public void displayLexer(String text, String word, int depth){
  //excluded
  String [] excludedWords = {"L_PAREN", "R_PAREN", "L_BRACE", "R_BRACE", "L_BRACKT", "R_BRACKT", "SEMICOLON", "COMMA"};
  if(text.equals("<EOF>")){
    return;
  }
  for(String excludedWord:excludedWords){...}
  //color
  String [] reservedWords = {"const", "int", "void", "if", "else", "while", "break", "continue", "return"};
  String [] operators = {"plus", "minus", "mul", "div", "mod", "assign", "eq", "neq", "lt", "gt", "le", "ge", "not", "and", "or"};
  String color = "";
  for (String reservedWord : reservedWords) {...}
  for (String operator : operators) {...}
  if(word.equals("IDENT")){ color = "[red]"; }
  if(word.equals("INTEGR_CONST")){ color = "[green]"; }

  // 处理8进制,16进制
  if (text.startsWith("0x") || text.startsWith("0X")){...} else if (text.startsWith("0") && text.length() > 1){...}

  //output
  for (int i = 1; i < depth + 1; i++) {
      System.err.print("  ");
  }
  System.err.println(text + " " + word + color);

}
```

遇到的问题

1. 设计g4文件时，不太理解左递归的含义，在查阅ANTLR手册后解决该问题。
2. 在输出词法单元时，未找到获得`depth`属性的方法，经过类型强制转换后成功（如下图）。

```java
RuleContext ruleContext = new RuleContext((RuleContext) node.getParent(), invokingState: 0);
depth = ruleContext.depth();
displayLexer(node.getSymbol().getText(), word, depth);
```