Lab2实验报告

201250141 刘屿

实验思路

1. 首先根据SysY语法规则编写`SysYParser.g4`，然后为其生成语法分析器`SysYParser.java`
2. main方法接收文件路径，并将文件内容传给词法分析器，从词法分析器获取 Token 流后再传递给语法分析器
3. 实现一个继承自BaseErrorListener的MyParserErrorListener并添加给SysYParser
    - 之后遇到错误时会向MyParserErrorListener发送错误信息，调用`synTaxError()`方法，因此只需要重写`synTaxError()`方法，按要求的格式输出错误信息即可
    - 因为需要输出全部错误信息，所以在MyErrorListener类中定义一个boolean成员变量`used`，用来判断是否发生了语法错误
        - 通过在`syntaxError()`方法中将used设为true实现
4. 通过`sysYParser.program()`函数生成语法树，并进行错误检查
5. 如果`myParserErrorListener.used`为false，即没有发生语法错误，正常打印语法树，通过 ANTLR 提供的 Visitor 机制来实现，将**词法规则和语法规则作为成员变量**传递给继承自`SysYParserBaseVisitor` 的 `Visitor`类
    - 将需要打印的每个词法规则的高亮颜色组织成数组`highLight`，作为 Visitor 的成员变量在构造函数中初始化
        - 后面打印时就可以通过**表驱动**机制来获取高亮颜色
6. Visitor在访问每个节点的子节点前会调用`visitChildren()`函数，通过重写这个函数来实现语法树的打印
    - 通过获取节点在语法树中的深度计算出该行打印时的缩进应该有多长
    - 通过成员变量的 `parserRuleNames` 和该节点的 `getRuleIndex()` 可以获取到对应的 ruleName，首字母变为大写后打印即可
7. Visitor在访问每个终结符节点时会调用`visitTerminal()`函数，通过重写这个函数来打印终结符及终结符的高亮
    - 因为本身没有 `depth()` 方法，需要通过获取父节点的深度来计算出本节点的深度，然后计算出缩进距离
    - 通过`node.getSymbol().getType()`获取到该节点所属的类型，只有 `1 ⩽ type ⩽ 24 || type==33 || type==34` 时才需要打印
    - 通过成员变量的 `lexerRuleNames, highLight` 获取到对应的词法规则名与高亮颜色，按格式打印即可

8. 在遇到INTEGR_CONST时需要将 8 进制与 16 进制转化为 10 进制，通过编写静态函数实现，之后按要求输出即可

精巧的设计

- 通过深度来计算缩进距离
- 通过在 MyParserErrorListener 中添加成员变量 used 来判断输入文件是否有语法错误
- 将需要打印的每个词法规则的高亮颜色组织成数组 highLight，作为 Visitor 的成员变量在构造函数中初始化，后面需要获取到对应词法规则的颜色时，可以通过表驱动机制获取
- 通过 TerminalNode 类无法获取到所处深度，但是可以获取到父节点，强转为 RuleContext 类获取到父节点深度，+1 即可

遇到的困难及解决办法

问题不是很多，主要就是一开始不清楚要如何获取到规则名，之后从SysYLexer和SysYParser中分别获取词法和语法规则名数组，然后作为成员变量传递给 Visitor 就可以了