编译原理 Lab 2 实验报告

学号：201250185

姓名：王福森

我实现的功能

本次实验我通过 antlr 工具，以 SysYParser.g4 作为输入，得到一个语法分析器 SysYParser.java，结合上次实验得到的词法分析器，实现了打印语法树功能，实现了语法高亮功能，实现了语法报错功能。

我是这么实现的

通过实验一我得到了 SysY 语言的词法分析器。

我通过阅读 SysY 语言的文法，将其中的文法规则写成 g4 文件的格式，并且对其中的表达式和条件语句进行 “左递归” 改造，通过 antlr 工具为上述的 g4 文件生成对应的语法分析器 SysYParser.java。

我自己定义了一个错误处理类 ParserErrorListener，该类的功能是在语法分析处理时，对无法分析的内容输出特定的错误内容，这个部分的内容与实验一基本相同。

我自己还定义了一个继承自 SysYParserBaseVisitor<Void> 且 override 了 visitChild 和 visitTerminal 两个成员方法的类 HighLightVisitor。visitChild 方法用于输出缩进和规则名称，缩进的大小通过传入参数 node.getRuleContext().depth()，规则名称则通过 ruleContext.getRuleIndex() 获得。visitTerminal 方法用于终结符输入缩进，文本，类型及高亮的颜色。缩进通过 node.getParent() 得到父节点的缩进后再加一即可得到缩进的大小；文本直接通过 node.getSymbol().getText() 获得；类型则先获得类型索引后查 SysYLexer.java 中的ruleNames 表即可得到；颜色通过表驱动实现，该表 terminalNode 的 key 是类型，value 是颜色。唯一需要特殊处理的就是，当类型是 “INTEGR_CONST” 时，将该类型对应的文本都转化成 10 进制输出。

在 main 方法中，我先对从参数中得到输入的文件并传入词法分析器 SysYLexer.java，得到 token 流，再将该流传入　语法分析器 SysYParser.java，并且设置词法分析器的 errorListener 为我自己定义的 ParserErrorListener。通过调用 sysYParser 对象的 program() 成员方法即可对输入程序进行语法分析并且得到相应的语法分析树 ParserTree。此时查看 parserErrorListener 的 hasError 字段，如果没有出错则用调用 HighLightVisitor 对象 visitor的 visit() 方法遍历语法树，同时输出相应内容。

我的精巧设计

我比较朴实没有啥精巧的设计 /(ㄒoㄒ)/~~

过程中有趣现象和印象很深的 bug

我第一次提交后的分数是 3872，一个非常奇怪的数字，因为测试用例没有公开，我只能去群上看各路 dalao 们遇到的 bug 是什么，然后看看是否能刚好解决我的问题，结果发现大家的分数都是非常整的，比如 3600 什么的，寻找未果 /(ㄒoㄒ)/~~。所以我一遍又一遍地看要求文档，看看是不是哪里写错了，看来好几遍都觉得没啥问题；然后又怀疑是不是 g4 文件写错了，所以就自己又一个字一个字地敲了一遍 g4 文件，结果还是没问题；又怀疑是不是 lexer 写错了，替换了助教 gg 上传的 lexer 文件，提交后还是 3872 分呜呜呜。最后吃了个饭回来发现是terminalNode 表中少了 “NEQ”！！！ 上传了就 AC 了。总共 de 了 5 h，结果发现是少抄了一行……