Compilers Lab1: Lexer 实验报告

程序功能

在 `antlr` 辅助下实现了 `SysY` 的词法分析器 `SysYLexer`，能够将 `SysY` 源程序文本字符流转化为遵循定义的词法规约的词法单元流，如果没有发现错误，则输出每个词法单元的信息（`Symbolic name`, `text`, `line number`），如果发现错误，则报告其中的语法错误（此处仅限词法分析阶段发现的错误）。

实现和设计细节

实现包含两个部分：`SysYLexer.g4`，`SysYErrorListener.java`

`SysYLexer.g4` 规定了 `SysY` 程序词法单元的规约，大部分都在实验文档中给出，我的实现中仅添加了对 `INTEGR_CONST`(整数常量，不考虑负数) 和 `IDENT` (标识符)。

对于 `INTEGR_CONST`，要考虑的情况有三种，十进制数，八进制数（以 0 为前缀），十六进制数 (以 0x 或0X 为前缀)

为了方便复用，对于每一种进制数，我先用 fragment 来表示它们的数位，比如十进制数位(`DECIMAL_DIGIT`)是 `[0-9]`，八进制数位(`OCT_DIGIT`)是 `[0-7]`，十六进制数位(`HEX_DIGIT`)是 `[0-9a-fA-F]`。

接着，对于十进制数来说，处理 0 的情况（单独拿出来, regex: 0）和非零的情况（非零开头，后面是 0 次或若干次十进制位,regex: `[1-9]DECIMAL_DIGIT*`），对于八进制，就是前缀 0 加上一个及以上的八进制位(regex: `0OCTDIGIT+`)，对于十六进制数，就是前缀 0x 或者 0X 加上一个及以上的十六进制位(regex:`(0x | 0X) HEX_DIGIT+`)。

对于标识符来说则比较简单,匹配下划线或字母开头，后面跟 0 个或若干个字母(ALPHA: [a-zA-Z])，数字，下划线的字符即可(regex: `(_ | ALPHA) (_ | ALPHA | DECIMAL_DIGIT)*`)

为了实现错误报告和在监测到错误的时候不继续输出解析到的词法单元，我编写了 `BaseErrorlistener` 的子类，覆盖了 `syntaxError` 方法，在其中输出了定制的错误报告信息，并将该 `listener` 对象的一个属性 `hasSyntaxError` 置为 `true`，以便主程序中检查是否发现错误。

实验中遇到的有趣的现象或印象深刻的 bug

刚开始忘记了八进制和十六进制还能有 000, 0x00, 0x001 之类的这样的形式，产生了一些错误，其他的好像没遇到什么问题。