编译原理编译器前端实验报告

10132510232-吴炎

实验内容

1、 词法分析器

词法分析器的工作是低级别的分析：将字符或者字符序列转化成相应的词法单元，分析得到的词法单元以供后续语法分析使用。

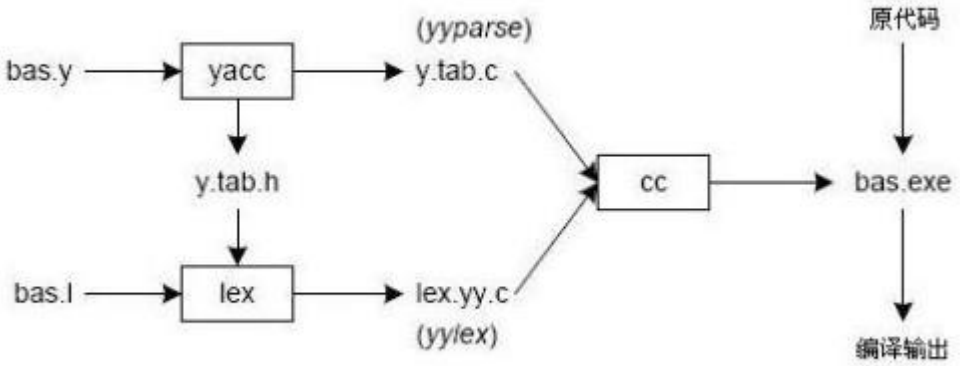
在本次实验分析时，我们把词法分析器当成编译程序的独立部分。在前一种情况下，词法分析器不断地被语法分析器调用，每调用一次词法分析器将从源程序的字符序列拼出一个单词，并将其 Token 值返回给语法分析器。

2、 语法分析器

语法分析其通过获取词法分析器的词法单元，对应相应产生式进行语法检查、并构建由输入的单词组成的语法树。

在本次实验中，我们组选用 Flex 与 Yacc 作为词法分析器与语法分析器的工具，并且采用 Vim 与 Java 的 UI 界面两种方式，对于结果做出展示，其中，在 Vim 方面，我们组也通过本地配置对相应的代码进行高亮处理，从而使得代码更加美观易读。

Flex 与 Yacc 工作流程如下图：



实验结果

本次实验在词法方面，对于从文件读取或者从编辑器中编写的字符流，在 UI 层进行了词法单元的输出，使得结果清晰明了。

在语法方面，根据定义的产生式规则，在 UI 层对于每一个错误信息，根据“行号-列号：错误信息”等方式，对于结果进行了精确的说明，让使用者能够快速根据结果对代码进行修正。

此外，我们呈现了一个简洁实用的 UI 界面，让使用者能够快速进行词法与语法的分析。

实验心得

本次实践中，主要负责的部分

1、 参与词法分析器的编写（共同完成）
2、 负责 UI 界面的搭建，进行 Swing window 观感的设置
3、 参与 JNI 后台逻辑与界面结合的工作（共同完成）
4、 负责文件流的处理
5、 参与项目汇报的说明（共同完成）

本次实验中，主要学习到了 Lex 和 Yacc 工具的使用，lex 通过正则表达式匹配的方式，对于输入流进行分析，同时依据最长匹配的规则，保证获取最优解，在这过程中，也锻炼了正则表达式的使用能力。Yacc 的使用，因为没有负责语法分析部分，所以只是了解了些基础，Yacc 通过 LALR 语法，读取 lex 传递的词法单元，对产生式进行匹配，结果通过 yyerror 进行呈现，通过阅读代码，我发现我们组在错误处理这块，通过对错误类进行封装，从而保证了 error 的精确显示。同时，在学习 yacc 的过程中，遇到了 shift/reduce 之类的冲突，而通过定义运算符优先级等策略可以帮助我们很好的解决这一类问题。

在 UI 界面上，对于用 C 实现的 lex 与 yacc，如何与 Java 的 UI 界面整合是我们在过程中面临的一个问题，通过资料阅读，我们发现了使用 JNI 调用本地库（dll，so）的方法，通过对 lex，yacc 等 c 文件的打包，生成动态链接库，然后在 Java 中通过 JNI 方式编写相应代码，即可在 Java 中调用相应的 c 接口完成我们的工作。所以，通过这次实验，也学习到了 Java 和 c 的结合使用。

此外，在这次实验中，对于编译原理课程所学知识也进行了一系列巩固，特别是编译器前端方面，词法语法部分的相结合工作，正则表达式的使用，LALR 语法的工作原理，冲突的解决方案等，都通过此次试验进行了实践，进而加深了理解。