编译原理课程实验报告

> 实验 2 - 词法分析器
>
> 实验时间: 28th Oct, 2016

1) 实验内容与目的

实验 2 要求使用 Flex 或手工构造一个针对特定语言（见实验说明文档）的词法分析器，藉此了解词法分析器，初步熟悉 Flex 的使用方法或手工编写词法分析器的要点。

2) 实现分析

.lex 源文件由定义部分、规则部分以及用户附加的 C 代码部分构成。在定义部分我们可以进行全局声明，`%{` 与 `%}` 之间的部分将被原样复制到生成的 C 代码中。之后到 `%%` 之前的部分都将是基于正则表达式的 lex 的标记声明，这些标记可以在规则部分使用。

在本次实验中，我们在定义部分定义了各词素的返回值、声明了在附加代码部分调用的函数并给出了在规则部分中将要使用的正则表达式标记。实验中将词素分类为变量与数字（ID/NUM）、保留字（BASIC/SEMICOLON/COMMA）、操作符（+-*/=）、逻辑关系符(>/</==/<>/!=/>=/<=)、括号（()[]{}）、条件语句保留字（IF/ELSE/DO/WHILE/BREAK）、布尔常量保留字（TRUE/FALSE），同时在定义部分给出了分隔符 delim、空白符 ws、字母 letter、数字 digit、类型保留字 basic、操作符 ops、变量名 id、 数 number 的正则定义。

在第一个 `%%` 标记之后，lex 文件进入规则部分，我们在此部分规定了词法分析器读取到各词素时的对应动作，此部分到第二个 `%%` 标记之前结束。

在第二个 `%%` 标记之后，lex 文件进入用户附加 C 代码部分规定主函数等内容。需要注意的是这一段必须包括 `yywrap()` 函数，一般简单地定义为 `int yywrap() { return 1; }` .

3) 实验结果

实验所采用的 flex 版本为 flex 2.5.35, C 编译器为 LLVM version 8.0.0, 系统环境为 macOS Sierra 10.12.2.

实验采用的测试文件为 `testfile.c` ，采用 flex 生成的 `lex.yy.c` 文件编译的词法分析器可正常编译测试文件。

4) 尚需考虑的问题

1. 注释匹配所使用的正则表达式：在本次实验中，对于注释字段的匹配沿用了 `[\/][\*]([^\*])*[\*]([^\*\/](([^\*])*)[\*]|[\*])*(\/)` 这个表达式，但是能够注意到 `[\*]` 可以规约到后面的部分，从而将表达式变为 `[\/][\*]([^\*])*([^\*\/](([^\*])*)[\*]|[\*])+(\/)` 这样的形式，但此想法的正确性尚待确认（两正则表达式是否等价）。
2. 错误词素的捕捉问题：当前 .lex 源文件没有找到正确的定义以捕捉错误的词素，因而会造成失配的词素将被词法分析器原样输出。