实验一 词法分析与语法分析

完成人：陈越琦(121160005)
小组编号：11

实验环境

ubuntu 14.04 LTS kernel version 3.16.0-50 generic
gcc version 4.8.4
flex version 2.5.35
bison 3.0.2

程序功能

读取.cmm 文件，分析文件中代码的词法和语法是否符合 c--文法。分组选做内容为任务 2（识别指数形式的浮点数）。
程序能够正确定位词法错误（Error type A）行数给出正确贴切的说明文字。
程序能够正确定位语法错误（Error type B）行数给出正确贴切的说明文字。
如果测试文件中代码完全符合 c—文法，能够按输出格式正确输出“语法树”。

编译方式

在目录 lab/Code 下
make parser 或者 make 生成可执行文件 parser 在目录 lab/下
make test 测试 parser 的正确性， 测试文件在目录 lab/Test 下，按字典排序依次测试，要求测试文件扩展名为.cmm

实验过程和代码释义

实验是按照实验手册一步一步做下来的，先熟悉 flex 和 bison 的用法，然后完成词法分析和语法分析部分。

第一步实现词法分析,修改 makefile 文件生成 scanner 可执行文件，通过测试样例测试 scanner 的正确性，为语法分析打下基础。

第二步实现语法分析，在 bison 自动进行语法分析的过程中，通过语义动作建立语法树。经过词法分析和语法分析检查没有错误的就按格式输出。

语法单元的属性值类型是结构指针，指向语法树中结点类型，每个结点需要记录词法单元的行号，类型， 语法单元名称和语法单元的词素(针对终结符)，子结点数，和子结点指针数组（终结符的子结点数为 0）。

其中需要解释的是类型属性，类型分成 3 类， 根据最后的输出分类。INT , FLOAT， ID，TYPE 这四个需要输出词法单元词素的为一类，其他的终结符为一类，非终结符为一类。

在词法分析部分，通过函数 creat_node_token()生成所有终结符的结点，返回给语法分析部分。在语法分析部分，通过函数 creat_node()将产生式右边所有语法单元的属性值（即语法树结点指针）以及产生式左边语法单元的词法单元名称和位置作为输入一步一步自底向上构成语法树，形成的语法树跟结点为 root。函数 creat_node()是可变参数函数。因为每个产生式右边的语法单元数量是不确定的。

通过全局变量 error_num 来记录出现错误的个数。词法分析的过程中，识别出词法错误就在 action 中对 error_num 加一；语法分析的过程中，在 yyerror()函数中，对 error_num 加一。完成一个测试文件的分析后，判读 error_num 是否为 0 来决定是否输出语法树，并且将 error_num 清零，yylineno 重置为 1。

最后，在没有任何词法和语法错误的前提下， 通过前序遍历的方式输出语法树。

二义性消除方面，一方面是规定操作符的结合性和优先级，另外一方面，为了避免移入-归约冲突，需要定义一个优先级比 else 低的语法单元，从而使得任何 if 语句在遇到 else 语句的情况下，选择移入 else 而不是归约。此外，为了区别取负和减法，定义一个语法单元为 UMINUS，其优先级和结合性与取负操作相同。

错误恢复方面，基本是让行号和括号作为错误恢复的同步符号。其中有一个特殊情况，如果在测试程序中相应位置出现缺少分号或者括号的情况会可能会对后续的错误识别造成影响，这也是程序需要进一步优化的地方。

错误信息输出，A 类错误是在词法分析中使用通配符”.”直接判断出来。B 类错误是在在 bison 文件中添加%error-verbose 选项，在 yyerror()函数中输出准确的信息，这里所谓的准确是大致准确，即能够准确无误得输出错误行号但是说明文字并不是 100%得正确，语法分析器按照文法理解出来的错误与实际的错误会有偏差。这是很正常的现象。此外，如果测试程序中出现缺少分号或者括号的情况，定位出来的错误行号是下一行的行号。因为语法分析器在分析的时候，无法意识到这是一个句子或者是一个语句块的结束，只会接着向下看，所以报告出来的错误行号是下一行的行号，其实这也是语法分析器按照文法理解出来的错误与实际错误有偏差的一个表现。

实验中遇到的困难和解决方案

实验中遇到的困难是可变参数函数 creat_node()的实现。一开始并不知道可以使用头文件<stdarg.h>中定义的宏。因此是通过手动判断函数参数栈中的情况来访问参数。但是 c99 标准中的栈的使用与之前学习内容略有不同，其参数压栈方式恰好相反。左边的参数在高地址，右边的参数在低地址。此外，即使能够定位到对应参数的地址，从地址中取出参数的值也不是轻而易举的事情，因此最后是使用 va_start , va_arg 等宏来实现这一功能。

实验总结

程序实现基本的词法分析和语法分析功能，通过实验熟悉了 flex 和 bison 的用法。此外还掌握了实现可变参数函数的方法。同时，巩固加深了对课堂教学内容的理解。
程序在一下方面还需要进一步的优化：
1.对于程序中出现的相应位置缺少分号和括号的情况还要更深一步的细化处理。
2.使用输入缓冲区提高词法分析效率
3.增加更多测试样例

参考资料

http://www.cplusplus.com/reference/cstdarg/va_start/

2015/10/9