实验报告

181850236 张子辰（基础班） 181850236@smail.nju.edu.cn

1. 程序运行和本地环境：

进入”/Lab/Code”目录，直接在终端运行 make 命令，即可编译生成 parser 文件，可以使用该文件对.cmm 文件进行分析。

或者通过在目录下依次执行命令“bison -d syntax.y”, “flex lexical.l”, “gcc main.c syntax.tab.c tree.c -lfl -ly -o parser”来生成二进制文件。

我的本地运行环境为 Ubuntu amd64 18.04, flex 2.6.4, bison 3.0.4, gcc 7.5.0。

2. 程序功能

我的程序实现了本次实验的必做任务和所有选做任务。

本次实验的必做任务的重点在于多叉树的创建和维护，实现起来难度不大，我将这部分代码的实现写在了 tree.c 和 tree.h 中。

三个选做任务的正确实现全部是在 lexical.l 文件中：

① 对于八进制、十六进制和浮点数的检测，和第一种注释”//”的识别，我是通过正则表达式直接匹配的。

② 对于第二种注释”/**/”，我在检测到”/*”后令 flex 跳转到注释状态，检测到”*/”后再回到初始状态，而如果检测到了<<EOF>>，就会报错。

对于错误情况的识别各有不同：

① 对于十进制、八进制和十六进制整数的错误识别，我在词法分析中定义了相应的错误正则表达式，匹配错误的输入，并作为错误类型 A 输出。

② 对于浮点数，在最初的实现中，我采用了和①中相同的方法，但在测试时，发现当成员操作符(.)和 e 连接在一起时会误报。所以我将这种错误交给了语法分析，如果出现错误会输出错误类型 B。

③ 对于注释错误，主要涉及的是”/**/”类型的注释，如果缺少”*/”，我将在 lexical.l 中通过检测到<<EOF>>识别，报错类型为 A，并调用 yyterminate() 终止程序，防止死循环出现；而如果缺少”/*”，我将其交给语法分析，处理的错误是”*”与”/”相邻着出现，错误类型是 B。

3. 程序亮点

① 我使用了上一级拔尖班同学维护的测试库进行了测试，可以通过所有 L1 分支的样例。但该测试库对于正确样例采用逐行匹配，对于错误样例只要求产生任意错误输出即可，所以无法判定报错及行号是否正确。

② 为了解决①中的问题，我从中挑选了一些错误样例进行语法分析，对于程序输出不正确的情况开启 DEBUG 模式调试，添加了一些错误产生式。但

我并不认为我的程序在所有错误样例上都能正确输出，等面临错误时再对程序进行修改。

③ 我从 Flex 和 Bison 中巧用单双引号提升语法文件的可读性这篇博客中学到了 bison 别名的使用，使代码在一定程度上更易读。

④ 通过课本上实例的学习以及实验讲义中对 ELSE 的处理，我通过%prec 解决了“减号”和“负号”优先级冲突的问题。

⑤ 对于语法错误的报错，最初我是想修改向 yyerror()中传递的参数来解决。但我在 StackOverFlow 上找到了名为 How to change parameter of yyerror function 的文章，认识到这是不可能实现的，因为 yyerror()是 bison 在遇到错误后自动调用的。最后我是在 bison 中开启了%error-verbose 同时在 yyerror()中使用 yytext 来获得比较完善的报错。

⑥ 在 syntax.y 中建立语法分析树时，因为子节点数目是可变的，所以我使用了 stdarg.h 头文件来支持可变参数的函数 AddChildren，我认为通过减少函数调用次数，可以使程序性能更高。但实现中也遇到了 Segmentation fault (core dumped)的问题，后来在 stdarg.h reads too many arguments 这篇文章中得到了解决。

⑦ 为了在全是注释的文件中输出正确的 Program 行号（EOF 位置），我在创建 Program 节点时会检测 ExtDefList 是否存在，如果他是 NULL，则意味着后面没有任何语句，这时就将 Program 节点的行数设置为 ExtDefList 为 NULL 的那一行。