南京大学

计算机科学与技术系

编译原理实验报告

实验名称：L1-词法分析与语法分析

组长学号：211830115

组长姓名：郑九铭

组员学号：211180213

组员姓名：胡德谔

实验时间：2.29-3.29

报告撰写：郑九铭

一、 程序功能介绍

1. 必做部分

本实验完成了对于给定程序，分析其词法单元和语法单元结构，从而检查出程序中的词法错误和语法错误并输出对应的错误类型及行号。若程序无错误，则打印出符合要求的语法树。

其中，检测词法错误和语法错误通过在 lexical.l 中定义词法单元和在 syntax.y 中定义语法单元，借助 flex 和 bison 实现词法分析和语法分析，并从中识别出词法和语法错误。同时我定义了 treenode 这一多叉树数据结构，从而在词法分析和语法分析中实现了对语法树的构建和维护，以及后续对语法树的打印。

2. 选做部分

本实验完成了全部选做部分的要求。为：

- 识别八进制数和十六进制数。
- 识别指数形式的浮点数。
- 识别“//”和“/*…*/”形式的注释。

实现方式均为在词法分析过程中对上述选做部分的词法单元进行特殊识别与处理。例如八进制数，其在用正则表达式匹配后会被用自定义的 otoi 函数转化为 int 类型，再转为字符串常量传入语法树中处理。

3. 实验亮点

本人认为该实验的设计亮点有二：

- **支持变长参数列表传入的多叉树节点构建函数**：本实验中定义了如下函数 `TreeNode* insertNode(int linenum, NodeType type, char* name, char* val, int argc, ...)`，其可以在语法分析过程中一步到位构建语法树的同时维护行号与属性值，与助教在实验课上展示的语法树设计相比大大减少了代码量，同时提高了可维护性。（具体代码见 syntax.y 文件中对该函数的定义与使用）

- **利用 git.nju.edu.cn 代码托管平台进行代码管理**：本实验由于是多人组队，因此采用了南京大学提供的代码托管平台进行代码管理，这样做便于代码版本维护和代码传输，提高了合作效率，同时降低了代码出现不可逆转的问题而无法恢复的风险。

二、 程序编译方式

1. 编译

在命令行中执行以下指令：

cd CODE_PATH

这里 CODE_PATH 是指项目文件夹 Code 所在的路径，确保执行后所处的位置在 Code 文

件夹中；

make

此时即完成了对代码的编译，可以看到可执行文件 parser 已经生成。

2. 运行

在命令行中执行以下指令：

./parser ../Test/test1.cmm

即可分析 test 文件夹中的 test1.cmm 程序。分析其余程序同理，只需将./parser 后的路径修改为待分析程序的路径即可。

三、 个人感想与问题

1. 个人感想

实验难度适中，但是有一定工作量，确实需要花费大量时间完成。

2. 问题

个人认为实验指导手册中对于某些可能遇到的输入情况下程序应该表现的行为是未定义的，这大大增加了不必要的实验难度，如：

- 若程序中只有一段注释而无其他内容，语法树中 program 的行号应当为第一行还是注释后的第一行？
- 对于只有“/*”而没有“*/”的样例的话，是应该将“/*”后的内容全部视为注释，并报一个词法错误；还是将“/*”视为 DIV MUL，并报一个语法错误？

希望这些疑问可以得到解答，十分感谢。