编译原理lab01 实验报告

161240056 谈婧

本次实验完成了对于输入代码进行基本词法分析和语法分析的功能，实验编写环境mac os。

完成功能点：

词法分析部分支持以下功能

- 利用正则表达式完成了cmm的token识别（基本）；
- 在整数范围支持识别十进制数、八进制数和十六进制数；
- 支持识别小数形式浮点数和指数形式浮点数；
- 支持多种错误处理：
  - 对于非法八进制和十六进制整数的识别和报错；
  - 对于超出32位十进制整数范围(-2147483648 ~ 2147483648)的十进制数的识别和报错；
  - 对于未定义字符的识别和报错；

语法分析部分支持以下功能：

- 支持对cmm语法的识别
- 支持语法树的打印
- 支持基本错误识别和原因分析

实现方法与数据结构表示：

- 词法分析的实现利用了flex工具和正则匹配。
  - 对于非法十进制/八进制/十六进制的处理也使用了正则匹配，通过猜测可能的错误形式并用正则匹配获取来移除错误，并且仍旧return INT类型以保证不影响语法分析。
  - 在词法分析中设置了指示变量 Lexical_flag 来提示词法错误的存在，使得存在词法错误时不打印语法树。
- 语法分析的实现利用了bison工具。
  - 根据附录A中给出的语法信息和bison语法实现了语法分析；
  - 构建语法树时实现了多叉树。本多叉树的组织形式是，对于每一个非叶结点，有一个child指针指向其第一个子结点，其余子结点都一一用sibling指针相连。
  - 多叉树的结点结构如下：

```
struct TreeNode{
    int lineno;
    int type; // 0 for nonterminals, 1 for id, 2 for decint, 3 for float, 4 for Specifiers, 5 for type
    char * name;
    int type_int;
    float type_float;
    struct TreeNode * child;
    struct TreeNode * sibling;
}TreeNode;
```

    - type用于说明该节点的类型，具体定义见注释。

- name中存储id/保留字/数字的字符串形式信息
- type_int和type_float是为整型数和浮点数设计的，用来存储具体数字。
- child和sibling是TreeNode型指针，用于维持树形结构。
- 使用type来控制在打印语法树时是输出yytext中匹配到的字符串，还是给出整型数/浮点数。

- 多叉树包含的函数方法：
  - struct TreeNode * create_node(char * name, int lineno, char * text)
    - 由于需要用到create_node的情况一定是非终结符号在创建节点，因此实现create_node的逻辑是，创建一个type为0的节点，name记录这个非终结符号的名字，text好像没啥用。
  - void add_child(struct TreeNode * a, struct TreeNode * b)
    - add_child的逻辑是给a加上子结点b，如果a的child指针为空，则直接将b赋值给a->child。如果a->child不为空，则从a->child指向的结点开始，找该结点的兄弟结点，直到找到空为止。将b添加为a的最后一个子结点的兄弟。
  - void add_sibling(struct TreeNode * a, struct TreeNode * b)
    - 由于语法树的特殊性，如果a需要添加sibling，则其sibling一定为空，否则不符合语法产生式的定义。因此只需要简单的将b赋值给a->sibling即可。

编译运行方法：

可以使用make指令，也可以使用命令行：

```
bison -d syntax.y
flex lexical.l
gcc main.c syntax.tab.c -ll -ly -o parser [for mac os]
gcc main.c syntax.tab.c -lfl -ly -o parser [for linux]
```

如此会生成parser可执行文件，可以调用 `./parser [test url]` 进行测试。

实验总结：

- 在实现词法分析中对于注释的处理时，我一开始使用了正则匹配进行整个注释和注释块的提取，然后在action处放空。在对optional-test6进行测试的时候发现虽然都会对nested comments报错，但是这样处理的报错在line 3，而给出的结果在line 8。我觉得其实都可以，但是出于对实验讲义的尊重，还是改成了使用正则匹配comments的开头，用input读入comments达到忽略效果的实现。
- bison有很多隐藏的功能，比如加上#define YYERROR_VERBOSE 就可以输出详细syntax error的内容，这些就需要自己探索。
- 写实验的时候感觉自己对词法分析和语法分析有了更清晰的认识，尤其是在调整syntax conflicts的时候就会认真的思考语法产生式之间的逻辑。我遇到一个bug是，刚刚给语法产生式加上动作之后，由于觉得既然都是空就不需要处理而没有给空/*empty*/的地方也加上动作，因此bison给出了no action的warning。于是我先删除了所有的/*empty*/语句，紧接着warning了一半的nonterminals和rules都是useless的。仔细思考了语法结构之后发现，如果真的删除了所有是空的地方，那么好几句产生式就变成了要获得A只需要A和B，而如果存在A就会存在更多的A；如果一开始不存在A，那么永远不会存在A的逻辑。因此意识到了空的重要性。最后经过思考，加上了空的情况，也明白了即使是空也需要加上action: create_node。