编译原理实验一

211220028 党任飞

一、实现功能

1. 识别词法错误。识别出没有在附录定义的 c-- 语法，并报出错误类型A。
2. 识别语法错误。识别出不符合附录中产生式隐含的语法错误，并报出错误类型B。
3. 选做3：识别 // 和 /*.....*/ 类型的注释。

二、运行方式

进入 Lab1/Code/ 文件夹之后顺序执行以下指令：

```
flex lexical.l
bison -d syntax.y
gcc main.c syntax.tab.c syntaxTree.c -lfl -ly -o parser
./parser [test file path]
```

三、核心实现方法

3.1 词法分析

主要参考实验手册给出的 flex 语法进行分析，其中需要注意的是：

- 针对形如 6_wrong 的错误ID，专门设置规则 WRONGID {digit}{ID} 进行匹配，否则一定会匹配为一个数字加一个ID，然后报语法错误。
- 用通配符 . 来兜底，进行词法报错。
- [选做] 注释：
  - 如匹配到 // ，循环读取下一个字符直到换行符 \n 。
  - 如匹配到 /* ，循环读取下两个字符，直到读取到 */ 或文件结尾结束。

3.2 语法分析

按照实验手册指示进行，其中在合适的位置插入 error 进行错误恢复。

其中最令人困扰的是 `CompSt : LC DefList StmtList RC` ，一旦有 `Stmt` 出现在 `Def` 之前，此时按照文法会在最开始创造空的 `DefList` ，因此在后面的就是 `StmtList` 了，此时需要使语法分析器将后面的所有 `DefList` 视为error并丢弃。

3.3 语法树

选择带有 `child` 和 `sibling` 两个指针的树格式，即多叉树的二叉树表示。因为这样的格式方便从一个 `child` 开始遍历其兄弟节点，方便输出。

建树：

- 在词法分析的过程中，为每一个 token 申请空间，形成一个树节点。
- 在语法分析的过程中建树。一旦有一个产生式成功匹配，则为产生式头生成一个树节点，其子节点为产生式体中的元素。这样自底向上的过程保证产生式匹配成功的时候，子节点都已经被创建。
    - 这里用到了 `<stdarg.h>` 库中的不定个数输入工具 `va_list` ，处理子结点个数不定的问题。

打印树。按照实验手册要求从root结点开始遍历并打印树。