**语法分析（实验二）：**

我们构造了两种语法分析程序：通过文件输入和输出分别实现输出语法分析树和最左规约的顺序输出派生的产生式序列。

对于第一种，需要在每个低层语句的动作,也就是翻译模式中新建终结符节点并把树结构回传给非终结符。在最顶层语句对应的动作中深度遍历构建的语法分析树，按格式打印在终端和输出文件。

对于第二种，由于bison使用自下而上的处理顺序，直接打印对应规约语句即可

### 3.2.1语法分析系统结构

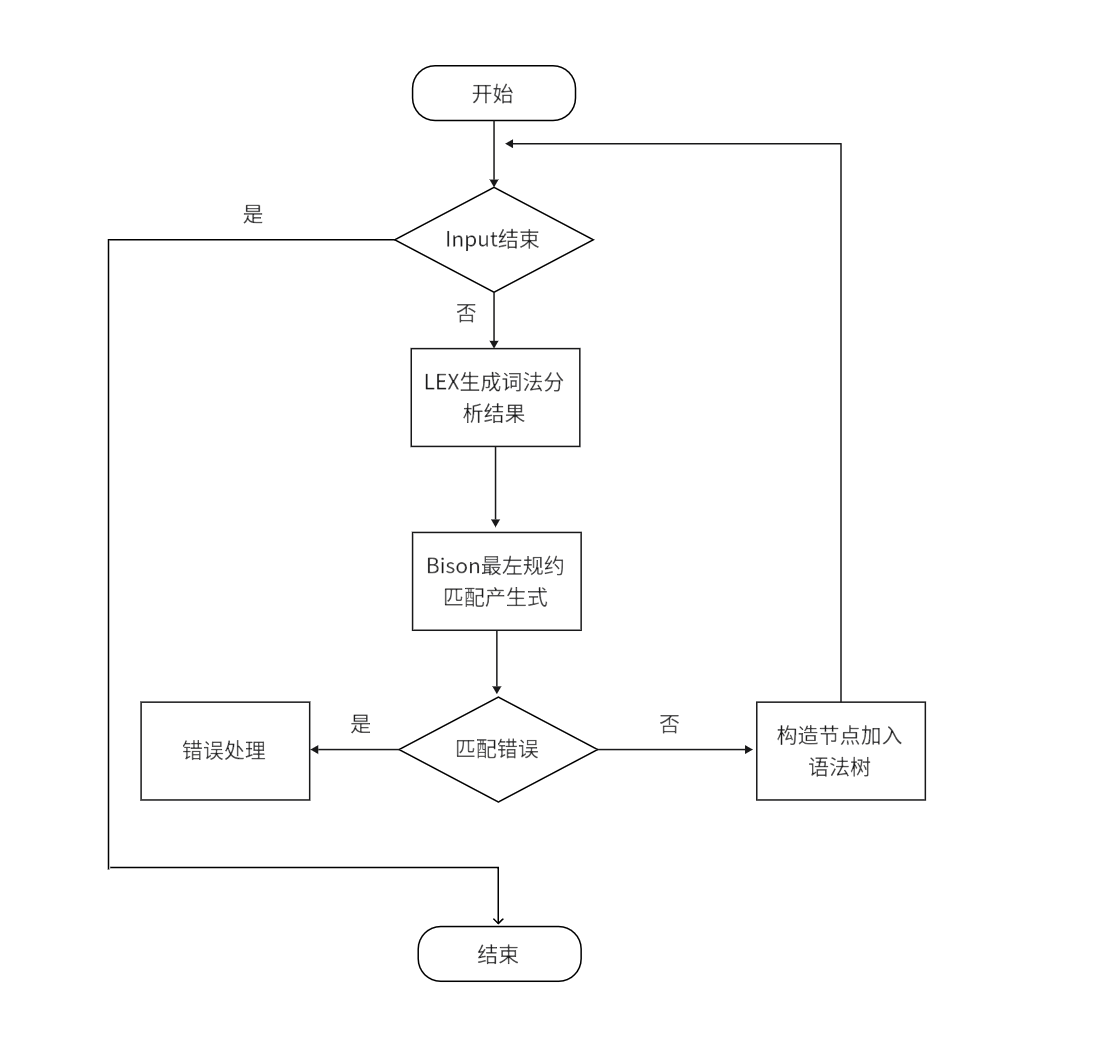


图3-1

### 3.2.2语法分析子系统的主要数据结构与算法

①print2file 函数：接收一个字符串参数 s，将字符串打印到控制台并写入输出文件 o 中。

②yyerror 函数：接收一个字符串参数 s，将字符串错误消息打印到控制台中。

③newnode 函数：接收三个参数，分别是 len，name 和 brothers。其中，len 表示 brothers 数组的长度，name 表示新节点的名称，brothers 是一个指向结构体 node 的指针数组，表示新节点的兄弟节点们。函数通过 malloc 分配新节点的内存，然后将 name 设置为新节点的名称，将 brothers[0] 设置为新节点的第一个孩子，将其中的兄弟节点连接起来，最后返回指向新节点的指针。

④newleaf 函数：接收一个参数 name，表示新叶子节点的名称。函数通过 malloc 分配新节点的内存，然后将 name 设置为新节点的名称，将该节点的兄弟和孩子都设置为 NULL，最后返回指向新节点的指针。

⑤traverse 函数：接收两个参数，分别是 depth 和 root。其中，depth 表示当前打印字符时需要缩进的空格数，root 是要遍历的根节点。函数首先将 depth 个空格写入到输出文件中，然后将 root 节点的名称也写入到输出文件中，并在结尾处加一个换行符。接着，如果 root 节点有孩子，就递归地调用 traverse 函数，将深度加 2。如果 root 节点有兄弟，则递归地调用 traverse 函数，将深度保持不变。

⑥treefree 函数：接收一个参数，表示要释放其内存的根节点。函数首先判断该节点的孩子和兄弟是否都为空节点，如果是，就释放这些空节点的内存，并将它们的指针设置为 NULL。然后，递归地调用 treefree 函数，继续处理该节点的孩子和兄弟。

⑦main 函数：程序的主函数，通过调用 yyparse 函数解析输入文件中的语法，并将其转换为抽象语法树。在函数开头，先打开输入文件并设置为 yyin 的值，打开输出文件并将其指针存储到全局变量 o 中。最后，返回 yyparse 函数的值。

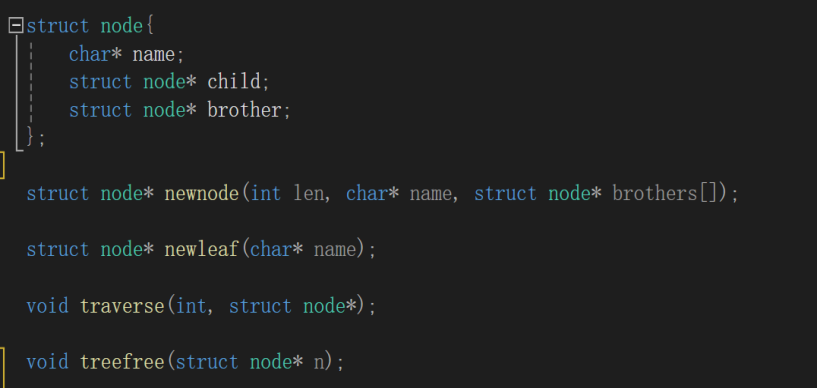


图3-2

⑧错误处理函数：函数首先将全局变量 errflag 设置为 1，表示当前解析出现了错误。然后，通过全局变量 yylloc 记录下当前词法分析器所读取的 Token 的位置信息，从中获取到当前错误所在的列数 start 和结束的列数 end。最后，函数将错误消息、行号和列号写入到输出文件 errorFile 中。具体地，通过 fprintf 函数输出 "Error: %s on Line: %d:c%d to %d:c%d\n" 这样一个格式化的字符串，使用参数 s 替换第一个 %s，使用 yylineno 替换第二个 %d，使用 start 替换第三个 %d，使用 yylineno 替换第四个 %d，使用 end 替换第五个 %d。其中，":c" 是用于表示列数的格式化信息。

### 3.2.3 采用的自动生成技术为Bison自动生成工具

Bison通过获得flex分析出的token以及用户定义的产生式进行归约，每当根据一条产生式进行归约时，执行用户预先定义好的一系列动作。

关于bison：bison不分析二义性文法。需要事先告诉优先级和分组规则。

bison使用两种分析方法，一种是LALR(1)，一种是LR我们采用默认的LALR分析：不能处理有歧义的语法，不能前向查看多个token确定是否匹配规则。合并同心闭包集。而判断语句“if else”和“if else then”需要前向查看多个符号。而bison好处是在编译环节自动报告语法存在的冲突。

其中，Bison核心代码如下：

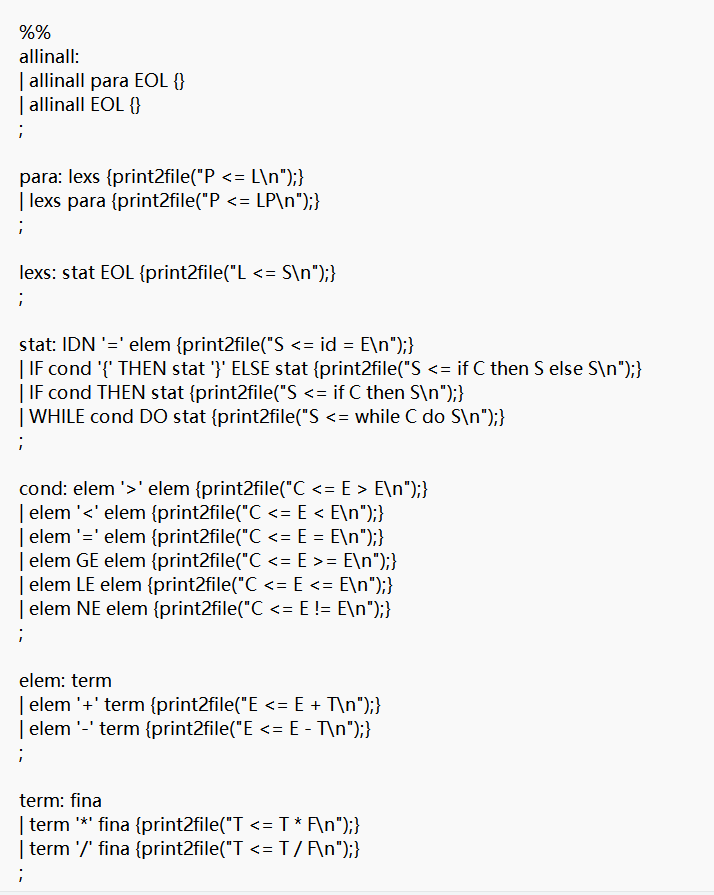


图3-3

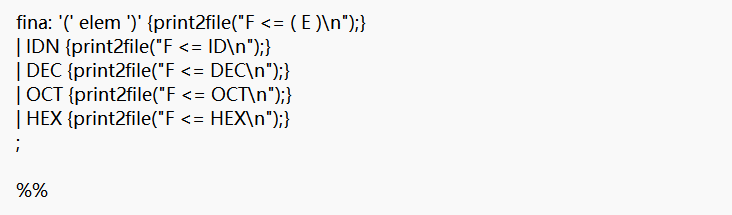


图3-3续

### 3.2.4实验结果

徽标, 公司名称

描述已自动生成

图3-2：第一句最左规约的顺序输出派生的产生式序列

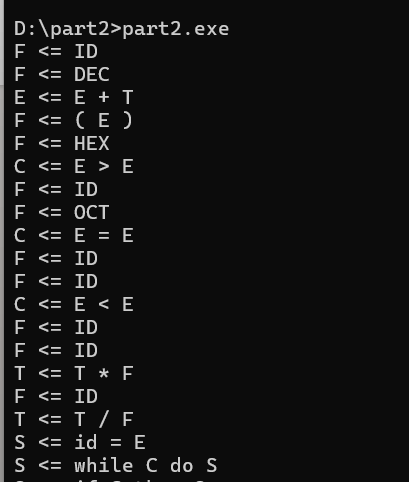


图3-4

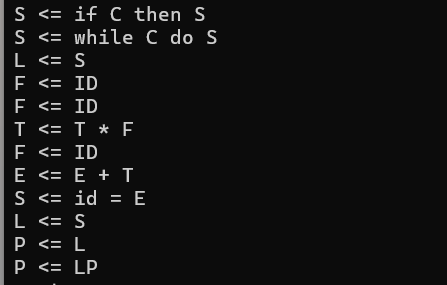


图3-4 续