Lab1 实验报告

201240069 曹语桐　201240060 林彦葶

一、如何编译运行程序

在 Code 目录下执行:
make clean
make
生成可执行文件 parser。
假设测试文件为./test.c，运行./parser test.c 就可以得到输出结果。

PS: 我们设置了简洁打印模式与详细打印模式, 在 main.c 中令 sim=0 则输出详细词法信息，令 bisonsim=0 则输出详细语法信息，令 errorsim=0 则输出详细错误恢复信息，将这些变量置为 1 则输出信息是实验要求的标准输出。

二、实现的主要功能

1.

首先，我们实现了讲义上所要求的全部必做和选做内容：查出 C--语言测试文件中的词法错误和语法错误并输出报错信息，如果没有词法和语法错误，输出程序的语法树信息。其中:

lexical.l 负责实现词法分析功能，出现未定义符号时报错，输出错误类型 Error Type A 及对应行号。出现单行注释//时，利用 flex 的库函数 input()将该行所有符号丢弃。出现/*时，将丢弃从/*开始到识别到的第一个*/之内的所有字符。没有错误时，返回词法单元给 syntax.y。

syntax.y 负责实现语法分析功能，出现语法错误时，输出 Error Type B 及对应语法单元的第一行行号；没有错误时，添加每个终结符和非终结符信息到语法树上，最终输出语法树信息。

tree.h 和 tree.c 负责实现语法树，输出没有语法和词法错误时程序的语法树结构，其中 tree.h 定义 NODE 结构体信息，声明添加节点和打印语法树的函数。tree.c 定义添加终结符到语法树的函数、添加非终结符到语法树的函数以及打印语法树的函数。

2.

在此基础上，设计错误恢复方案，使语法错误发生时，程序能够丢弃尽可能少的词法单元，尽快同步成功。

具体的错误恢复方案是:

a.主要在语句级别上进行错误恢复，即在 ExtDef，Stmt，Def 上进行错误恢复，error 后跟的同步符有两种：一种是是非终结符的 First（）和 Follow（）集合中的符号；另一种是标志语句结束的符号 SEMI，RC。目的在于尽量在下一个语句结束前完成错误恢复，从而之后的语句不受影响。

```
ExtDef->error SEMI
    | error Specifier
Stmt->error SEMI
    | error Exp
    | error RETURN
    | error IF
```

| error WHILE

| error RC

| error Specifier　//Specifier 虽然不属于 Stmt 的 First（）和 Follow（）集合，但是由于在错误恢复中可能出现 Specifier 被吞掉，语法分析提前进入 StmtList，但是代码实际上仍处于 DefList 的情况（见图 1），此时的 Specifier 可以帮助尽早实现同步。

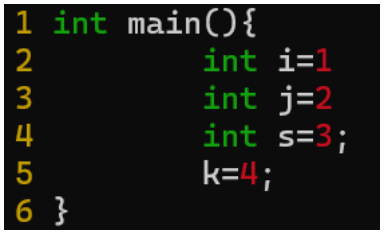

图 1

（第 3 行的 int 被第 2 行的错误恢复吞掉，语法分析识别 j=2 为 Stmt，从而进入 Stmt 的错误恢复，如果没有 Specifier 作为同步符，会将第 4 行的 int 和 s 都丢弃掉，加入 Specifier 为同步符后，s 不会被丢弃）

Def->error SEMI

| error Specifier

| error Exp

| error RETURN

| error IF

| error WHILE

| error RC

b.在 FunDec 这样产生式较少，且产生式结尾的终结符相同的非终结符产生式中添加错误恢复。

FunDec-> ID LP VarList RP

| ID LP RP

| error RP