编译原理
实验一

成员： 组长 161130118 尹浚宇
组员 171850524 刘扬

一、实现功能

- 对于正确的程序，通过标准输出打印语法树；
- 对于包含错误的信息，输出相关的语法错误和词法错误；
- 识别源代码中的注释(选做 1.3)

二、实现过程

(1)词法单元与语法单元的识别

词法单元和语法单元参考附录 A 进行实现． 依据附录 A 在 syntax.y 中定义文法所有的非终结符和终结符，并描述文法．

终结符对应 Tokens，在 lexical.l 中使用正则表达式匹配 Tokens 识别词法单元，之后为其分配语法树结点并返回．

在 syntax.y 中，依据文法的描述，创建文法表达式左侧非终结符对应的语法树结点，并将出现在表达式右边的语法单元对应的树结点作为子女链接到左侧语法单元的树节点上．

(2)词法错误和语法错误的识别

对于词法错误， 在 lexical.l 中规则部分的最后使用通配符”.”来匹配任意不能被其它规则匹配的词素，这样的词素不属于文法描述的语言，被视为词法错误，直接向 stderr 输出词法错误的信息：行号+未知符号．行号对应 yylineno，它需要我们手动更新：正则表达式匹配到换行符后执行 yylineno++的操作．

对于语法错误， 函数 yyerror(char* msg)将在发现语法错误时自动被调用输出语法错误， 此处只需要重写 yyerror 函数让其向 stderr 打印语法错误的信息： 行号+错误信息．msg 默认为”syntax error”，可以通过开启以下两个宏(测试环境 ubuntu 16.04)来打印较为准确的语法错误．不过在版本为 ubuntu 12.02.2 的系统上，这两个宏会在编译时报错，我们在提交时选择将其注释．

```
%define parse.error verbose
%define parse.lac none
```

识别到语法错误后， 我们在 ExtDef VarDec FunDec CompSt Stmt Def Exp 中将 SEMI RP RC 等作为同步符号实现语法错误的恢复．

词法错误和语法错误都会引发 nr_Error++的操作， 我们在 main 函数中将其置为 0，若最终 nr_Error 不为零，则说明代码存在错误，不会打印语法树．

特别地， 对于缺少 SEMI RP RC 这一类语法错误，其报错总会将行号定义为下一个语法单元的行号．

(3)对于移入规约冲突的解决

对于操作符优先级引起的移入规约冲突，根据附录 A 中 C–语言中运算符的优先级和结合性进行定义便可消除． 操作符优先级中较为特殊的是对于负号和减号的区分， 处理方式是令 SUB 代表减号， MINUS 代表负号， 并规定二者优先级和结合性， 在文法描述中通过

Exp : SUB Exp %prec MINUS 来进行区分.

对于文法二义性产生的移入规约冲突， 我们的方案是在发生移入/规约冲突时选择移入， 实现方式来源于实验教材． 需要注意的是二义性不仅 ifelse 这一处冲突， 在 Dec DecList 的文法描述中都各包含一个冲突，处理方式同 ifelse.

(4)语法树

语法树的数据结构选择为子女-兄弟树， 便于处理语法树是一棵多叉树的问题． 结构定义如下： lineno 是树结点对应的语法单元的行号，空语法单元的行号为-1． isSyntax 用于区分树结点是词法单元还是语法单元． type 存储语法单元名称． Info 用于存储词法单元的内容.

```
union Info{
    char name[MAXSIZE];
    int intVal;
    float floatVal;
};
typedef union Info Info;
struct syntaxTreeNode{
    struct syntaxTreeNode* child;
    struct syntaxTreeNode* brother;
    int lineno;
    int isSyntax;
    char type[MAXSIZE];
    Info info;
};
typedef struct syntaxTreeNode Node;
```

addChild 函数实现将语法单元加入语法树， showTree 将在语法分析结束后用于打印语法树，deleteTree 用于释放树结点的内存.

三、编译方式

使用自带的 makefile 进行编译.