语法分析

文法改写

为了消除文法的左递归性，便于理解，将文法改写如下

```
CompUnit → {Decl} {FuncDef} MainFuncDef
Decl → ConstDecl | VarDecl
ConstDecl → 'const' BType ConstDef { ',' ConstDef } ';'
BType → 'int'
ConstDef → Ident { '[' ConstExp ']' } '=' ConstInitVal
ConstInitVal → ConstExp
    | '{' ConstInitVal { ',' ConstInitVal } '}'
    | '{' '}'
VarDecl → BType VarDef { ',' VarDef } ';'
VarDef → Ident { '[' ConstExp ']' }
    | Ident { '[' ConstExp ']' } '=' InitVal
InitVal → Exp
    | '{' InitVal { ',' InitVal } '}'
    | '{' '}'
FuncDef → FuncType Ident '(' [FuncFParams] ')' Block
MainFuncDef → 'int' 'main' '(' ')' Block
FuncType → 'void' | 'int'
FuncFParams → FuncFParam { ',' FuncFParam }
FuncFParam → BType Ident ['[' ']' { '[' ConstExp ']' }]
Block → '{' { BlockItem } '}'
BlockItem → Decl | Stmt
Stmt → LVal '=' Exp ';'
    | [Exp] ';'
    | Block
    | 'if' '(' Cond ')' Stmt [ 'else' Stmt ]
    | 'for' '(' [ForStmt] ';' [Cond] ';' [ForStmt] ')' Stmt
    | 'break' ';'
    | 'continue' ';'
    | 'return' [Exp] ';'
    | LVal '=' 'getint''('')'';'
    | 'printf''('FormatString{','Exp}')'';'
ForStmt → LVal '=' Exp
Exp → AddExp
Cond → LOrExp
LVal → Ident {'[' Exp ']'}
PrimaryExp → '(' Exp ')' | LVal | Number
Number → IntConst
UnaryExp → PrimaryExp | Ident '(' [FuncRParams] ')'
    | UnaryOp UnaryExp
UnaryOp → '+' | '−' | '!'
FuncRParams → Exp { ',' Exp }
MulExp → UnaryExp { ('*' | '/' | '%') UnaryExp }
AddExp → MulExp { ('+' | '−') MulExp }
RelExp → AddExp { ('<' | '>' | '<=' | '>=') AddExp }
EqExp → RelExp { ('==' | '!=') RelExp }
LAndExp → EqExp { '&&' EqExp }
LOrExp → LAndExp { '||' LAndExp }
ConstExp → AddExp
```

偷看

为了消除多个选项的问题，提出如下“偷看”思路

1,

```
CompUnit → {Decl} {FuncDef} MainFuncDef
Decl → 'const' ... | 'int' ...
FuncDef → 'void' ... | 'int' ... '('
MainFuncDef → 'int' 'main' '('
```

一次性偷看三个单词

```
'const' ** ** -> Decl
'void' ** ** -> FuncDef
'int' 'main' '(' -> MainFuncDef
'int' ** '(' -> FuncDef
others -> Decl
```

2,

```
Stmt → LVal '=' Exp ';' 标识符开头 PLAN_A
| Exp ';' 标识符， "+","-","!", Number，'(' PLAN_B
| LVal '=' 'getint''('')'';' 标识符开头 PLAN_C
```

- 第一位是 标识符
  - 第二位是 (B
  - 第二位是 [A C B
  - 第二位是 = A C
  - 第二位是 其他 B