CompUnit → Decl | FuncDef | CompUnit Decl | CompUnit FuncDef

Decl → ConstDecl | VarDecl

ConstDecl → 'const' BType ConstDef ConstDefList ';'

ConstDefList → ',' ConstDef ConstDefList | ε

BType → 'int' | 'float'

ConstDef → Ident ArrayDef '=' ConstInitVal

ArrayDef → '[' ConstExp ']' ArrayDef | ε

ConstInitVal → ConstExp | '{' ConstInitVal InitValList '}'

InitValList → ',' ConstInitVal InitValList | ε

VarDecl → BType VarDef VarDefList ';'

VarDefList → ',' VarDef VarDefList | ε

VarDef → Ident ArrayDef | Ident ArrayDef '=' InitVal

InitVal → Exp | '{' InitVal InitValList '}'

InitValList → ',' InitVal InitValList | ε

FuncDef → FuncType Ident '(' FuncFParams ')' Block

FuncType → 'void' | 'int' | 'float'

FuncFParams → FuncFParam FuncFParamList | ε

FuncFParamList → ',' FuncFParam FuncFParamList | ε

FuncFParam → BType Ident ArrayDefExt

ArrayDefExt → '[' ']' ArrayDef | ε

Block → '{' BlockItems '}'

BlockItems → BlockItem BlockItems | ε

BlockItem → Decl | Stmt

Stmt → LVal '=' Exp ';'

| Exp ';'

| Block

| 'if' '(' Cond ')' Stmt ElseStmt

| 'while' '(' Cond ')' Stmt

| 'break' ';'

| 'continue' ';'

| 'return' Exp ';'

ElseStmt → 'else' Stmt | ε

Exp → AddExp

Cond → LOrExp

LVal → Ident LValArrayDef

LValArrayDef → '[' Exp ']' LValArrayDef | ε

PrimaryExp → '(' Exp ')' | LVal | Number

Number → IntConst | floatConst

UnaryExp → PrimaryExp | Ident '(' FuncRParams ')' | UnaryOp UnaryExp

UnaryOp → '+' | '−' | '!'

FuncRParams → Exp FuncRParamList | ε

FuncRParamList → ',' Exp FuncRParamList | ε

MulExp → UnaryExp MulOpExp | UnaryExp

MulOpExp → ('\*' | '/' | '%') UnaryExp MulOpExp

AddExp → MulExp AddOpExp | MulExp

AddOpExp → ('+' | '−') MulExp AddOpExp

RelExp → AddExp RelOpExp | AddExp

RelOpExp → ('<' | '>' | '<=' | '>=') AddExp RelOpExp

EqExp → RelExp EqOpExp | RelExp

EqOpExp → ('==' | '!=') RelExp EqOpExp

LAndExp → EqExp AndOpExp | EqExp

AndOpExp → '&&' EqExp AndOpExp

LOrExp → LAndExp OrOpExp | LAndExp

OrOpExp → '||' LAndExp OrOpExp

ConstExp → AddExp

Comment → SingleLineComment | MultiLineComment

SingleLineComment → '//' AnyCharacterExceptNewline '\n'

MultiLineComment → '/\*' CommentContent '\*/'

CommentContent → AnyCharacterUntilStarSlash CommentContent | ε

integer-const → decimal-const | octal-const | hexadecimal-const

decimal-const → nonzero-digit decimal-const | nonzero-digit

octal-const → '0' octal-const | '0'

hexadecimal-const → hexadecimal-prefix hexadecimal-digit hexadecimal-const | hexadecimal-prefix hexadecimal-digit

hexadecimal-prefix → ‘0x’ | ‘0X’

nonzero-digit 为以下之一

1 2 3 4 5 6 7 8 9

octal-digit 为以下之一

0 1 2 3 4 5 6 7

hexadecimal-digit 为以下之一

0 1 2 3 4 5 6 7 8 9 a b c d e f A B C D E F

上下文无关文法实现的全部的语言特性：

普通要求实现

1)数据类型：int

2)变量声明、常量声明，常量、变量的初始化

3)语句：赋值（=）、表达式语句、语句块、if、while、return

4)表达式：算术运算（+、-、\*、/、%，其中+、-都可以是单目运算符）、关系运算（==，>，<，>=，<=，!=）和逻辑运算（&&（与）、||（或）、!（非））

5)注释

6)输入输出（实现连接SysY运行时库，参见文档《SysY运行时库》）

加分规则实现：

1)函数、语句块（1分）

a)函数（0.5分）：函数声明、函数调用

b)变量、常量作用域（0.5分）：在函数中、语句块（嵌套）中包含变量、常量声明的处理，break、continue语句

2)数组（1.5分）：数组（一维、二维、…）的声明和数组元素访问

3)浮点数（1.5分）：浮点数常量识别、变量声明、存储、运算

4)代码优化（3分）

a)寄存器分配优化方法（1.5分）

b)基于数据流分析的强度削弱、代码外提、公共子表达式删除、无用代码删除等（1.5分）

5)自由发挥（2分）：自动向量化等