literal: literal\_prim | array\_literal | struct\_literal;

literal\_prim:

INT\_LIT

| FLOAT\_LIT

| STRING\_LIT

| NIL

| TRUE

| FALSE;

struct\_literal: ID LBRACE list\_elements? RBRACE;

list\_elements: element COMMA list\_elements | element;

element: ID COLON expression;

list\_statement: statement list\_statement | statement;

statement: (

declared\_statement

| assign\_statement

| if\_statement

| for\_statement

| break\_statement

| continue\_statement

| call\_statement

| return\_statement

) SEMICOLON;

declared\_statement: variables | constants;

variables:

VAR ID type\_minigo

| VAR ID (ASSIGN expression)

| VAR ID type\_minigo (ASSIGN expression);

constants: CONST ID ASSIGN expression;

assign\_statement: (lhs) (

SEMICOLON\_ASSIGN

| ADD\_ASSIGN

| SUB\_ASSIGN

| MUL\_ASSIGN

| DIV\_ASSIGN

| MOD\_ASSIGN

) expression;

lhs: lhs LBRACK expression RBRACK | lhs DOT ID | ID;

if\_statement:

IF LPAREN expression RPAREN LBRACE list\_statement? RBRACE list\_else\_if? else\_statement?;

list\_else\_if: else\_if list\_else\_if | else\_if;

else\_statement: ELSE LBRACE list\_statement? RBRACE;

else\_if:

ELSE IF LPAREN expression RPAREN LBRACE list\_statement? RBRACE;

for\_statement: basic\_for | for\_loop | for\_array;

basic\_for: FOR expression LBRACE list\_statement? RBRACE;

for\_loop:

FOR (assign\_for | variables\_for) SEMICOLON expression SEMICOLON assign\_for LBRACE list\_statement

RBRACE;

for\_array:

FOR ID COMMA ID SEMICOLON\_ASSIGN RANGE expression LBRACE list\_statement? RBRACE;

variables\_for: VAR ID type\_minigo? (ASSIGN expression);

assign\_for: (ID) (

SEMICOLON\_ASSIGN

| ADD\_ASSIGN

| SUB\_ASSIGN

| MUL\_ASSIGN

| DIV\_ASSIGN

| MOD\_ASSIGN

) expression;

break\_statement: BREAK;

continue\_statement: CONTINUE;

call\_statement: (lhs DOT)? ID LPAREN list\_expression? RPAREN;

return\_statement: RETURN expression?;

SEMICOLON:

';'

| '\r'? '\n' {

tk = self.preType;

if (tk):

list = [self.INT\_LIT, self.FLOAT\_LIT, self.ID,

self.STRING\_LIT, self.TRUE, self.NIL\_LIT,self.FALSE, self.NIL,

self.RETURN, self.CONTINUE, self.BREAK, self.RBRACE, self.RBRACK, self.RPAREN]

if tk in list:

self.text = ';'

else:

self.skip()

else:

self.skip()

};

program: (declared SEMICOLON)+ EOF;

declared:

variables

| constants

| function

| method

| struct\_type\_declared

| interface\_type\_declared;

function:

FUNC ID LPAREN list\_param? RPAREN type\_minigo? LBRACE list\_statement? RBRACE;

method:

FUNC LPAREN ID ID RPAREN ID LPAREN list\_param? RPAREN type\_minigo? LBRACE list\_statement? RBRACE

;

struct\_type\_declared: TYPE ID STRUCT LBRACE list\_fields RBRACE;

list\_fields: fields SEMICOLON list\_fields | fields SEMICOLON;

fields: ID type\_minigo;

interface\_type\_declared:

TYPE ID INTERFACE LBRACE list\_meth\_interface RBRACE;

list\_meth\_interface:

meth\_interface SEMICOLON list\_meth\_interface

| meth\_interface SEMICOLON;

meth\_interface: ID LPAREN list\_param? RPAREN type\_minigo?;

list\_param: param list\_param | param;

param: list\_ID type\_minigo COMMA param | list\_ID type\_minigo;

list\_ID: ID COMMA list\_ID | ID;