```
program ::=
    | function program


function ::= FUNCTION ident SEMICOLON BEGIN_PARAMS dec_list END_PARAMS BEGIN_LOCALS
dec_list END_LOCALS BEGIN_BODY sta_loop END_BODY


dec_list ::=
    | declaration SEMICOLON dec_list


sta_loop ::= statement SEMICOLON
    | statement SEMICOLON sta_loop


declaration ::= dec_help COLON array_size INTEGER


dec_help ::= ident
    | ident COMMA dec_help


array_size ::=
    | ARRAY L_SQUARE_BRACKET number R_SQUARE_BRACKET OF


statement ::= var ASSIGN expression
    | IF bool_expr THEN conditional ENDIF
    | WHILE bool_expr BEGINLOOP sta_loop ENDLOOP
    | DO BEGINLOOP sta_loop ENDLOOP WHILE bool_expr
    | FOR var ASSIGN number SEMICOLON bool_expr SEMICOLON var ASSIGN expression
BEGINLOOP sta_loop ENDLOOP
    | READ var_list
    | WRITE var_list
    | CONTINUE
    | RETURN expression


conditional ::= sta_loop
    | sta_loop ELSE sta_loop


var_list ::= var
    | var COMMA var_list


bool_expr ::= relation_and_expr
    | relation_and_expr OR relation_and_expr


relation_and_expr ::= relation_expr
    | relation_expr AND relation_and_expr


relation_expr ::= relation_expr_help
    | NOT relation_expr_help


relation_expr_help ::= expression comp expression
    | TRUE
    | FALSE
```

```
    | L_PAREN bool_expr R_PAREN


comp ::= EQ
    | NEQ
    | LT
    | GT
    | LTE
    | GTE


expression ::= multiplicative_expr
    | multiplicative_expr expression_help


expression_help ::= ADD multiplicative_expr
    | SUB multiplicative_expr
    | ADD multiplicative_expr expression_help
    | SUB multiplicative_expr expression_help


multiplicative_expr ::= term
    | term multiplicative_expr_help


multiplicative_expr_help ::= MULT term
    | DIV term
    | MOD term
    | MULT term multiplicative_expr_help
    | DIV term multiplicative_expr_help
    | MOD term multiplicative_expr_help


term ::= term_help
    | SUB term_help
    | ident L_PAREN term_ident R_PAREN


term_help ::= var
    | number
    | L_PAREN expression R_PAREN


term_ident ::=
    | expression
    | expression COMMA term_ident


var ::= ident
    | ident L_SQUARE_BRACKET expression R_SQUARE_BRACKET


ident ::= IDENT


number ::= NUMBER
```