GRAMMAR

```
program ::= | function program

function ::= FUNCTION ident SEMICOLON BEGIN_PARAMS help_dec_semi END_PARAMS
BEGIN_LOCALS help_dec_semi END_LOCALS BEGIN_BODY help_state_semi END_BODY

help_dec_semi ::= | declaration SEMICOLON help_dec_semi

help_state_semi ::= statement SEMICOLON | statement SEMICOLON help_state_semi

declaration ::= help_id_comma COLON help_array INTEGER

help_id_comma ::=       ident COMMA help_id_comma | ident

help_array ::= | ARRAY L_SQUARE_BRACKET number R_SQUARE_BRACKET OF

statement ::= IF bool_expr THEN help_if_then ENDIF
                     | WHILE bool_expr BEGINLOOP help_state_semi ENDLOOP
             | DO BEGINLOOP help_state_semi ENDLOOP WHILE bool_expr
             | FOR var ASSIGN number SEMICOLON bool_expr SEMICOLON var ASSIGN
expression BEGINLOOP help_state_semi ENDLOOP
             | READ help_var_comma
             | WRITE help_var_comma
             | CONTINUE
             | RETURN expression
             | var ASSIGN expression

bool_expr ::= relation_and_expr help_or_rae

help_if_then ::= help_state_semi | help_state_semi ELSE help_state_semi

help_or_rae ::= | OR relation_and_expr help_or_rae

help_var_comma ::= var COMMA help_var_comma | var

relation_and_expr ::= relation_expr help_and_re

help_and_re ::= | AND relation_expr help_and_re

relation_expr ::= NOT help_re_choices | help_re_choices

help_re_choices ::= expression comp expression
             | TRUE
             | FALSE
             | L_PAREN bool_expr R_PAREN

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

```
expression ::= multiplicative_expr help_pm_me

help_pm_me ::= | ADD multiplicative_expr help_pm_me | SUB multiplicative_expr
help_pm_me

multiplicative_expr ::= term help_mdm_term

help_mdm_term ::=
            | MULT term help_mdm_term
            | DIV term help_mdm_term
            | MOD term help_mdm_term

term ::= SUB help_vne_choices
            | help_vne_choices
            | ident L_PAREN help_expr R_PAREN
            | ident L_PAREN R_PAREN

help_vne_choices ::= var | number | L_PAREN expression R_PAREN

help_expr ::= expression | expression COMMA help_expr

var ::= ident | ident L_SQUARE_BRACKET expression R_SQUARE_BRACKET

ident ::= IDENT

number ::= NUMBER
```