program → macros classes

macros → macros macro

| ε

macro → reference

reference → REFERENCE STRING

classes → classes class

| ε

class → CLASS ID { symbol_decs }

symbol_decs → symbol_decs symbol_dec

| ε

symbol_decs → var_dec

| func_dec

var_dec → var_type var_list ;

var_type → return_type

| STATIC return_type

return_type → INT

| REAL

| STRING

var_list → var_list , var_list_item

| var_list_item

var_list_item → ID

| ID = exp

func_dec → func_type ID (formal_arguments) block

func_type → return_type

| STATIC return_type

| VOID

| STATIC VOID

formal_arguments → formal_arguments_list

| ε

formal_arguments_list → formal_arguments_list , formal_argument

| formal_argument

formal_argument → return_type ID

block → { statements_list }

| statement

statements_list → statements_list statement

| ε

statement → ;

| exp ;

| assignment

| print

| statement_var_dec

| if

| for

| while

| return

| break

| continue

assignment → ID = exp ;

print → PRINT (STRING) ;

statement_var_dec → return_type var_list ;

if → IF (exp) block elseif_blocks else_block

elseif_blocks → elseifs

| ε

elseifs → elseifs elseif

| elseif

elseif → ELSEIF block

else_block → ELSE block

| ε

for → FOR (ID IN exp TO exp STEPS exp) block

while → WHILE (exp) block

return → RETURN exp ;

break → BREAK ;

continue → CONTINUE ;

exp → INTEGER

| REAL

| STRING

| ID

| binary_operation

| logical_operation

| comparison_operation

| bitwise_operation

| unary_operation

| (exp)

| function_call

binary_operation → exp + exp

| exp – exp

| exp * exp

| exp / exp

| exp % exp

| exp ^ exp

| exp << exp

| exp >> exp

logical_operation → exp && exp

| exp || exp

comparison_operation → exp < exp

exp <= exp

exp > exp

exp >= exp

exp == exp

exp != exp

bitwise_operation → exp & exp

| exp | exp

unary_operation → - exp

| ! exp

function_call → ID (actual_arguments)

actual_arguments → actual_arguments_list

| ε

actual_arguments_list → actual_arguments_list , exp

| exp