

program \rightarrow macros classes

macros \rightarrow macros macro

| ϵ

macro \rightarrow reference

reference \rightarrow REFERENCE STRING

classes \rightarrow classes class

| ϵ

class \rightarrow CLASS ID { symbol_decs }

symbol_decs \rightarrow symbol_decs symbol_dec

| ϵ

symbol_decs \rightarrow var_dec

| func_dec

var_dec \rightarrow var_type var_list ;

var_type \rightarrow return_type

| STATIC return_type

return_type \rightarrow INT

| REAL

| STRING

| ID

var_list \rightarrow var_list , var_list_item

| var_list_item

var_list_item \rightarrow ID

| ID = exp

$\text{func_dec} \rightarrow \text{var_type func_body}$

| VOID func_body

| STATIC VOID func_body

$\text{func_body} \rightarrow \text{ID (formal_arguments) block}$

$\text{formal_arguments} \rightarrow \text{formal_arguments_list}$

| ϵ

$\text{formal_arguments_list} \rightarrow \text{formal_arguments_list , formal_argument}$

| formal_argument

$\text{formal_argument} \rightarrow \text{return_type ID}$

$\text{block} \rightarrow \{ \text{statements_list} \}$

| statement

$\text{statements_list} \rightarrow \text{statements_list statement}$

| ϵ

$\text{statement} \rightarrow ;$

| exp ;

| assignment

| print

| statement_var_dec

| if

| for

| while

| return

| break

| continue

assignment \rightarrow lvalue = exp ;

lvalue \rightarrow ID

| ID . ID

print \rightarrow PRINT (STRING) ;

statement_var_dec \rightarrow return_type var_list ;

if \rightarrow IF (exp) block elseif_blocks else_block

elseif_blocks \rightarrow elseifs

| ϵ

elseifs \rightarrow elseifs elseif

| elseif

elseif \rightarrow ELSEIF (exp) block

else_block \rightarrow ELSE block

| ϵ

for \rightarrow FOR (ID IN exp TO exp STEPS exp) block

while \rightarrow WHILE (exp) block

return \rightarrow RETURN exp ;

break \rightarrow BREAK ;

continue \rightarrow CONTINUE ;

$\text{exp} \rightarrow \text{INTEGER}$

| REAL

| STRING

| lvalue

| binary_operation

| logical_operation

| comparison_operation

| bitwise_operation

| unary_operation

| (exp)

| function_call

$\text{binary_operation} \rightarrow \text{exp} + \text{exp}$

| $\text{exp} - \text{exp}$

| $\text{exp} * \text{exp}$

| exp / exp

| $\text{exp} \% \text{exp}$

| $\text{exp} ^ \text{exp}$

| $\text{exp} << \text{exp}$

| $\text{exp} >> \text{exp}$

$\text{logical_operation} \rightarrow \text{exp} \&\& \text{exp}$

| $\text{exp} || \text{exp}$

comparison_operation \rightarrow exp < exp

exp <= exp

exp > exp

exp >= exp

exp == exp

exp != exp

bitwise_operation \rightarrow exp & exp

| exp | exp

unary_operation \rightarrow - exp

| ! exp

| ~ exp

function_call \rightarrow ID function_call_body

| ID . ID function_call_body

function_call_body \rightarrow (actual_arguments)

actual_arguments \rightarrow actual_arguments_list

| ϵ

actual_arguments_list \rightarrow actual_arguments_list , exp

| exp