program -> “prog” id ; block .

declaration -> “var” id declaration\_prime : type

| “proc” id ( parameters ) ; block

| “func” id ( parameters ) : type ; block

| ε

declaration\_prime -> , id declaration\_prime

| ε

parameters -> id : type parameters\_prime

| var id : type parameters\_prime

| ε

parameters\_prime -> , id : type

| , “var” id : type

| ε

type -> datatype | array\_type | ε

array\_type -> “array” [ ] “of” datatype

| “array” [ expr ] “of” datatype

| ε

datatype -> “int” | “char” | “real”

block -> “beg” statement block\_prime “end”

block\_prime -> ; statement block\_prime

| ;

| ε

statement -> structured\_statement

| simple\_statement

| declaration

simple\_statement -> assignment\_statement

| call

| return\_statement

| read\_statement

| write\_statement

| assert\_statement

assignment\_statement -> variable := expr

call -> id ( arguments )

arguments -> expr arguments\_prime

arguments\_prime -> , expr arguments\_prime | ε

return\_statement -> “return”

| “return” expr

read\_statement -> “read” ( variable read\_statement\_prime )

read\_statement\_prime -> , variable read\_statement\_prime | ε

write\_statement -> “writeln” ( arguments )

assert\_statement -> “assert” ( expr )

structured\_statement -> block

| if\_statement

| while\_statement

if\_statement -> “if” expr “then” statement

| “if” expr “then”statement “else” statement

while\_statement -> “while” expr “do” statement

expr -> simple\_expr

| simple\_expr relop simple\_expr

simple\_expr -> term siple\_expr\_prime

| addop term simple\_expr\_prime

simple\_expr\_prime -> add\_operator term simple\_expr\_prime | ε

term -> factor term\_prime

term\_prime -> mulop factor term\_prime | ε

factor -> call

| variable

| literal

| ( expr )

| “not” factor

| id . “size”

variable -> id

| id ( expr )

add\_operator -> addop

| “or”

literal -> integer

| real\_literal

| string\_literal

real\_literal -> integer . Integer

| integer . Integer “e” integer

| integer . Integer “e” integer

addop -> “+” | “-”

mulop -> “/” | “\*” | “ %”

relop -> “<” | “<=” | “>” | “>=” | “==” | “!=”

integer -> [0-9]+

id ->[a-zA-z][a-zA-z0-9\_]\*

string\_literal -> \”(\\.|