program ::= **init** program-suffix

program-suffix ::= id decl-assign | other-stmt

decl-assign ::= ":=" simple-expr ";" stmt-list-tail **stop**

| ident-list-tail is type ";" decl-list-tail stmt-list **stop**

other-stmt := stmt-prime ";" stmt-list-tail **stop**

stmt-prime := if-stmt | do-stmt

| read-stmt | write-stmt

decl-list-tail ::= decl ";" decl-list-tail

| λ

decl ::= ident-list **is** type

ident-list ::= identifier ident-list-tail

ident-list-tail ::= "," identifier ident-list-tail

| λ

type ::= **integer** | **string**

stmt-list ::= stmt ";" stmt-list-tail

stmt-list-tail ::= stmt ";" stmt-list-tail

| λ

stmt ::= assign-stmt | if-stmt | do-stmt

| read-stmt | write-stmt

assign-stmt ::= identifier ":=" simple\_expr

if-stmt ::= **if** "(" condition ")" **begin** stmt-list **end** if-suffix

if-suffix ::= **else** **begin** stmt-list **end**

| λ

condition ::= expression

do-stmt ::= **do** stmt-list do-suffix

do-suffix ::= **while** "(" condition ")"

read-stmt ::= **read** "(" identifier ")"

write-stmt ::= **write** "(" writable ")"

writable ::= simple-expr

expression ::= simple-expr expression-suffix

expression-suffix ::= relop simple-expr

| λ

simple-expr ::= term simple-expr-prime

simple-expr-prime ::= addop term simple-expr-prime

| λ

term ::= factor-a term-prime

term-prime ::= mulop factor-a

| λ

factor-a ::= factor | **not** factor | "-" factor

factor ::= identifier | constant | "(" expression ")"

relop ::= "=" | ">" | ">=" | "<" | "<=" | "<>"

addop ::= "+" | "-" | **or**

mulop ::= "\*" | "/" | **and**

constant ::= integer\_const | literal

integer\_const ::= nozero {digit} | “0”

literal ::= " “ " {caractere} " ” "

identifier ::= (letter) {letter | digit | " \_ " }

letter ::= [A-Za-z]

digit ::= [0-9]

nozero ::= [1-9]

caractere ::= um dos 256 caracteres do conjunto ASCII, exceto “ " ” e quebra de linha