Todas as funções estarão abaixo de todos os comandos do programa!

program ::= command\* functionDeclaration\*  
  
command ::=  identifier **=** expression

| **while** expression **do** (command)\*  **end**  
 | **if** expression **then** (command)\* (**else** (command)\*)? **end**

| procedureCall

| **break**

| **print** **(** (expression)? **)**

Pode ou não ter return associado

functionDeclaration ::= **function** identifier **(** (parameters)? **)** (command)\* (return (expression)?**;**)? **end**

parameters ::= identifier ( **,** identifier)\*

**Função sem retorno!**

procedureCall ::= **call** identifier **(** (arguments)? **)**  
  
arguments ::=  expression ( **,** expression)\*  
  
expression ::= relationalExpression

relationalExpression ::= additiveExpression (relationalOperator additiveExpression)\*

additiveExpression ::= multiplicativeExpression (additiveOperator multiplicativeExpression)\*

multiplicativeExpression ::= baseExpression (multiplicativeOperator baseExpression)\*

baseExpression ::= **(** expression **)** | number | identifier (nil | **(** (arguments)? **)** )

digit -> [**0**-**9**]  
  
number -> digit+(. digit+)?  
  
letter -> [**a**-**zA**-**Z\_**]  
  
identifier -> letter (letter | digit)\*  
  
relationalOperator -> **>** | **>=** | **<** | **<=** | **==** | **~=**  
  
additiveOperator -> **+** | **-**   
  
multiplicativeOperator -> **\*** | **/**  
  
reserved ->  **break** | **do** | **else** | **end** | **function** | **if** | **return** | **then** | **while** | **print**   
token -> number | identifier | relationalOperator | additiveOperator | multiplicativeOperator | reserved | **(** | **)** | **=** | **,**  
  
separator -> **#** (graphic)\* **\n** | **\n** | **\t** | ‘ ’