programa → **programa** ID { corpo } **fimprog**

corpo → declaracao corpo

| comando corpo

| declaracao

| comando

declaracao → tipo listaVar ;

listaVar → ID ( = expressão )? listaVarContinuacao

listaVarContinuacao → , ID ( = expressao )? listaVarContinuacao

| ε

tipo → **inteiro**

| **decimal**

| **texto**

| **logico**

comando → atribuicao ;

| chamadaFuncao ;

| cmdImprimir ;

| cmdLer ;

| cmdSe

| cmdEnquanto

| cmdPara

| cmdRetornar ;

| { corpo }

atribuicao → ID = expressao

| ID é expressao

chamadaFuncao → ID ( listaArg )

| ID ( )

listaArg → expressao listaArgRest

listaArgRest → , expressao listaArgRest

| ε

cmdImprimir → **imprimir** ( expressao )

cmdLer → **ler** ( ID )

cmdSe → **se** ( expressao ) { corpo } ( **senao** { corpo } )?

cmdEnquanto → **enquanto** ( expressao ) { corpo }

cmdPara → **para** ( inicializacao ; condicao ; incremento ) { corpo }

| **para** ( inicializacao ; condicao ; ) { corpo }

| **para** ( inicializacao ; ; incremento ) { corpo }

| **para** ( ; condicao ; incremento ) { corpo }

| **para** ( ; ; ) { corpo }

inicializacao → declaracao

| atribuicao

condicao → expressao

incremento → atribuicao

| expressao

cmdRetornar → **retornar** expressao

| **retornar**

expressao → exprOu

exprOu → exprE exprOuRest

exprOuRest → || exprE exprOuRest

| ε

exprE → exprIgual exprERest

exprERest → && exprIgual exprERest

| ε

exprIgual → exprRel exprIgualRest

exprIgualRest → == exprRel exprIgualRest

| != exprRel exprIgualRest

| ε

exprRel → exprAdd exprRelRest

exprRelRest → < exprAdd exprRelRest

| <= exprAdd exprRelRest

| > exprAdd exprRelRest

| >= exprAdd exprRelRest

| ε

exprAdd → exprMult exprAddRest

exprAddRest → + exprMult exprAddRest

| - exprMult exprAddRest

| ε

exprMult → exprUnario exprMultRest

exprMultRest → \* exprUnario exprMultRest

| / exprUnario exprMultRest

| ε

exprUnario → ! primario

| - primario

| primario

primario → ( expressao )

| ID ( listaArg? )

| ID ( )

| NUM

| TEXTO

| **verdadeiro**

| **falso**

| **nulo**

NUM → [0-9]+ ( ‘ . ’ [0-9]+ )?

TEXTO → ‘ " ’ ( qualquer caractere exceto aspas ou quebra de linha, com suporte a \" \\ \n \t ) ‘ " ‘

ID → letra ( letra | dígito | ‘ \_ ‘ )\*