

- Linguagem Karloff, definições léxicas e sintáticas:

Linguagem Karloff

~~~~~

20/12/2017

KARLOFF -> MAIN FUNC?

MAIN -> "void" "main" "{" VARDECL SEQCOMANDOS "}"

VARDECL -> VARDECL "newVar" TIPO TOKEN\_id ";" | vazio

TIPO -> "int" | "bool"

SEQCOMANDOS -> SEQCOMANDOS COMANDO | vazio

COMANDO -> TOKEN\_id "=" EXP ";"  
 | TOKEN\_id "(" LISTAEXP? ")" ";"  
 | "if" "(" EXP ")" "then" "{" SEQCOMANDOS "}" ";"  
 | "while" "(" EXP ")" "{" SEQCOMANDOS "}" ";"  
 | "repeat" "{" SEQCOMANDOS "}" "until" "(" EXP ")" ";"  
 | "return" EXP ";"  
 | "System.output" "(" EXP ")" ";"

EXP -> "(" EXP OP EXP ")" | FATOR

FATOR -> TOKEN\_id | TOKEN\_id "(" LISTAEXP? ")"  
 | TOKEN\_numliteral | "true" | "false"

OP -> "+" | "-" | "\*" | "/" | "&" | "|" | "<" | ">" | "=="

LISTAEXP -> EXP | LISTAEXP "," EXP

FUNC -> FUNC "func" TIPO TOKEN\_id "(" LISTAARG? ")" "{" VARDECL SEQCOMANDOS "}"  
 | "func" TIPO TOKEN\_id "(" LISTAARG? ")" "{" VARDECL SEQCOMANDOS "}"

LISTAARG -> TIPO TOKEN\_id | LISTAARG "," TIPO TOKEN\_id

=====

Convenções léxicas

~~~~~

TOKEN_id -> letra letraoudigito* finalsublinhado*

TOKEN_numliteral -> digitos facao_opcional expoente_opcional

onde:

letra -> [a-zA-Z]

digito -> [0-9]

```
digitos -> digito+
facao_opcional -> (.digitos)?
expoente_opcional -> (E (+ | -)? digitos)?
letraoudigito -> letra | digito
finalsublinhado -> _letraoudigito+
letra -> [a-zA-Z]
digito -> [0-9]
```