

- Linguagem Karloff, definições léxicas e sintáticas:

Linguagem Karloff

~~~~~

v 0.4

KARLOFF -> MAIN FUNC?

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

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

TIPO -> "float" | "boolean" | "void"

SEQCOMANDOS -> SEQCOMANDOS COMANDO | vazio

COMANDO -> TOKEN\_id "=" EXP ";"  
 | TOKEN\_id "(" LISTAEXP? ")" ";"  
 | "if" EXP "then" "{" SEQCOMANDOS "}" ";"  
 | "while" EXP "{" SEQCOMANDOS "}" ";"  
 | TOKEN\_id = "readInput" "(" ")" ";"  
 | "return" EXP ";"  
 | "printOut" EXP ";"

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

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

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

LISTAEXP -> EXP | LISTAEXP "," EXP

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