Nomes: *Gustavo Pioli Resende* - *12111BCC010*
Natan Gonçalves de Lyra - *12111BCC006*
Vinícius Dias Martins Rocha - *12111BCC039*

1ª Etapa do Projeto

1. Especificação Da Linguagem

1.1 Definição da Gramática Livre de Contexto (GLC)

Uma GLC G pode ser representada por uma quádrupla:

G(V, T, P, S)

- **V** = Conjunto dos Símbolos Não-Terminais
- **T** = Conjunto dos Símbolos Terminais
- **P** = Conjunto de Produções (regras de transição)
- **S** = INICIO

Símbolos Terminais:

- **main**
- **ID**
- **begin**
- **end**
- **int**
- **char**
- **float**
- **if**
- **then**
- **else**
- **while**
- **do**
- **repeat**
- **until**
- **número**
- **símbolo**
- **letra**

- **relop**
- **operador**
- **ws**
- **{**
- **}**
- **(**
- **)**
- **:=**
- **,**
- **;**
- **:**

Símbolos Não-Terminais:

- **INÍCIO**
- **tipo**
- **bloco**
- **declaração de variáveis**
- **declaração de variável**
- **lista_ids**
- **sequência de comandos**
- **comando**
- **condição**
- **texto_comentário**
- **expressão**

Produções Gramaticais (regras de transição)

- **<INICIO> ::= main ID** <bloco>
- **<bloco> ::= begin <**declaração de variáveis> <sequência de comandos> **end**
- **<declaração de variáveis> ::= <**declaração de variável> <declaração de variáveis> | ε
- **<declaração de variável> ::= <**tipo> : **<**lista_ids> **;**
- **<tipo> ::= char | int | float**
- **<lista_ids> ::= ID | ID, <**lista_ids>
- **<sequência de comandos> ::=** <comando> **|** <comando> **<**sequência de comandos>
- **<comando> ::=** <comando_seleção> | <comando_repetição> | <comando_atribuição>
- **<comando_seleção> ::= if (<**condição> **) then <**comando_ou_bloco> | **if (<**condição> **) then <**comando_ou_bloco> **else <**comando_ou_bloco>
- **<comando_ou_bloco> ::=** <comando> | <bloco>
- **<comando_repetição> ::=** <comando_while> | <comando_repeat>
- **<comando_atribuição> ::= ID := <**expressão**> ;**
- **<comando_while> ::= while (** <condição> **) do** <comando_ou_bloco>
- **<comando_repeat> ::= repeat** <comando_ou_bloco> **until (** <condição> **);**

- **<condição> ::=** <expressão> **relop** <expressão>
- **<texto_comentário> ::= {** comentário **}**
- **<expressão> ::= num | ID |** (<expressão>) **|** <expressão> <operador> <expressão>

1.2 Definição da Gramática Livre de Contexto (GLC)

LEXEMA	TOKEN	ATRIBUTO
main	main	-
begin	begin	-
end	end	-
int	int	-
char	char	-
float	float	-
if	if	-
then	then	-
else	else	-
while	while	-
do	do	-
repeat	repeat	-
until	until	-
Qualquer número	número	Posição na tabela de símbolos
‘Qualquer caractere’	símbolo	Posição na tabela de símbolos
Qualquer ID	ID	Posição na tabela de símbolos
>	relop	GT
<	relop	LT
>=	relop	GE
<=	relop	LE
==	relop	EQ
!=	relop	NE

:=	:=	-
{	{	-
}	}	-
(	(	-
)	)	-
+	operador	sum
-	operador	sub
/	operador	div
*	operador	mult
**	operador	exp
,	,	-
;	;	-
:	:	-
Texto entre chaves { }	-	-
ws	-	-

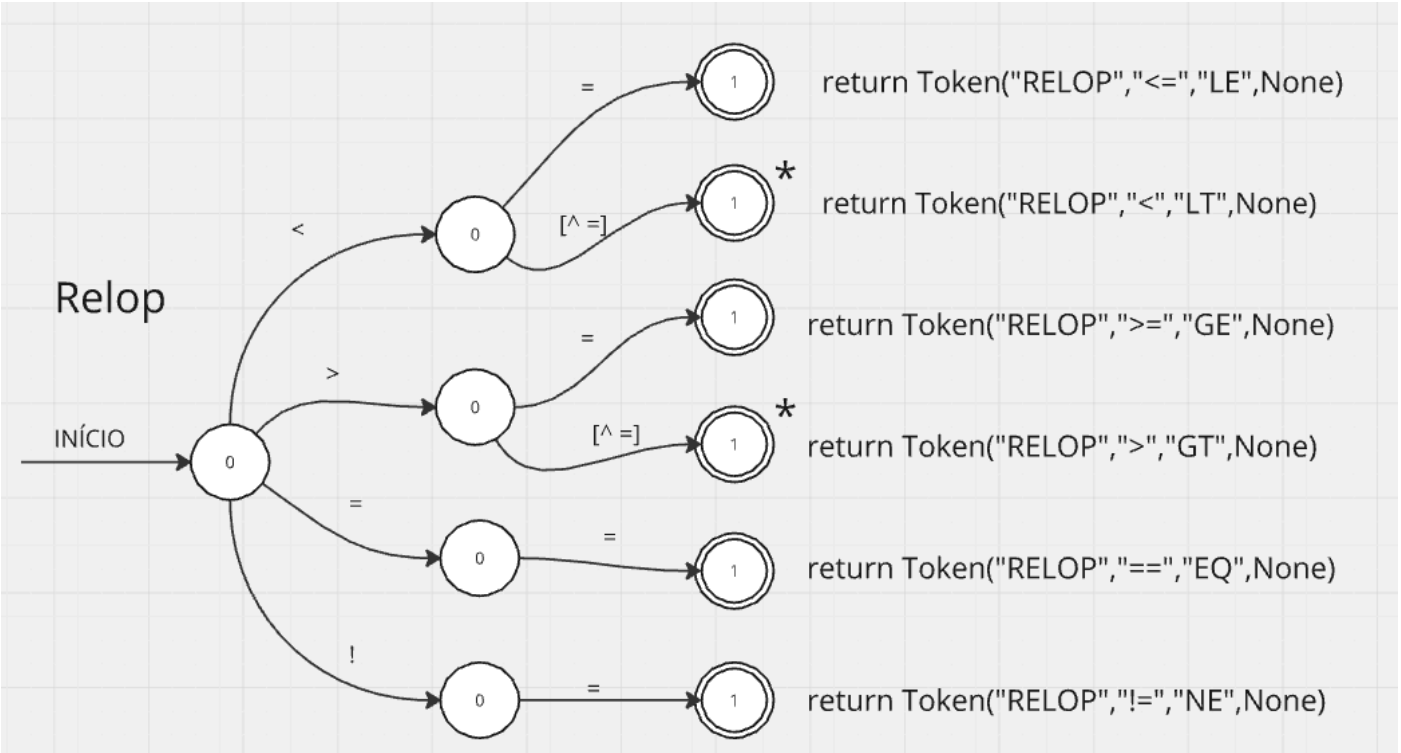
1.3 Definição dos padrões de cada token

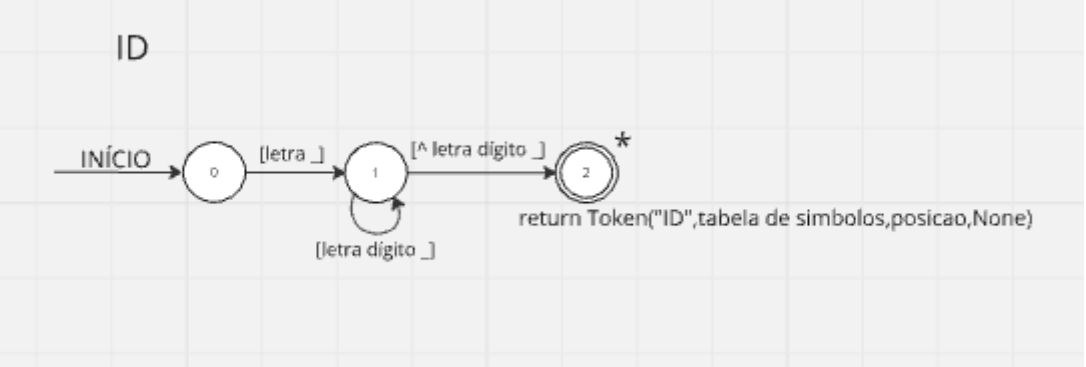
- main □ main
- begin □ begin
- end □ end
- int □ int
- char □ char
- float □ float
- if □ if
- then □ then
- else □ else
- while □ while
- do □ do
- repeat □ repeat
- until □ until
- { □ {
- } □ }
- (□ (
-) □)

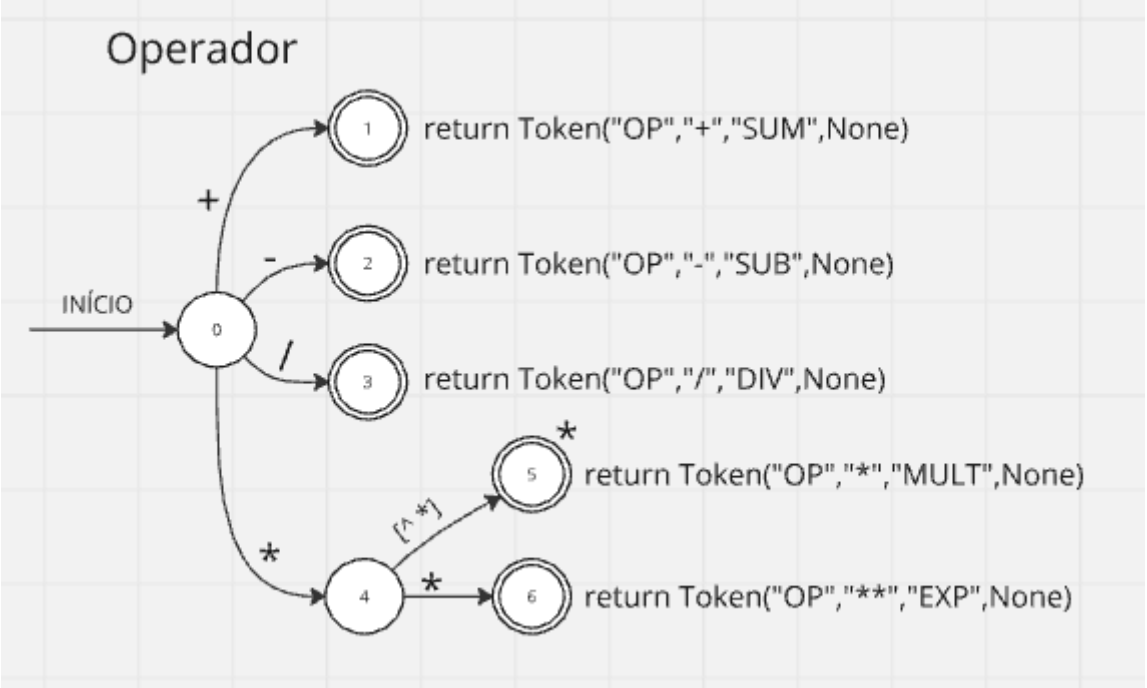
- := □ :=
- , □ ,
- ; □ ;
- : □ :
- dígito□ [0-9]
- dígitos □ dígito+
- número □ -?dígitos(\.dígitos)?(E-?dígitos)?
- letra □ [a-zA-Z]
- ID □ [letra _][digito letra _] *
- símbolo □ ’.’
- relop □ < | <= | > | >= | == | !=
- operador □ + | - | / | ** | *
- comentário □ [^ }]*
- ws □ [\t\n]

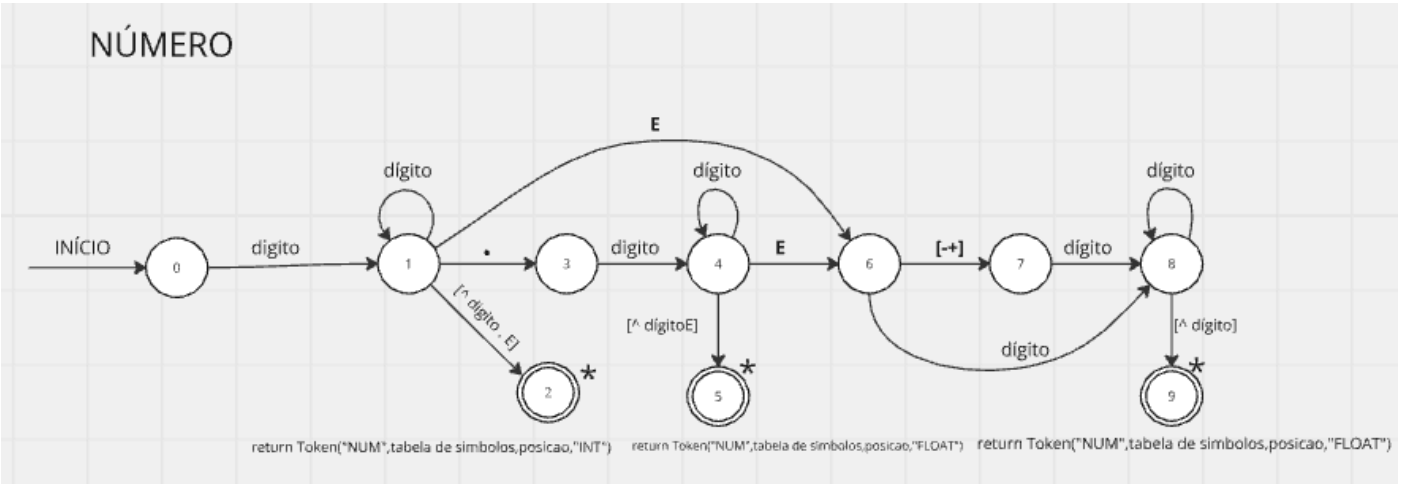
2. Análise Léxica

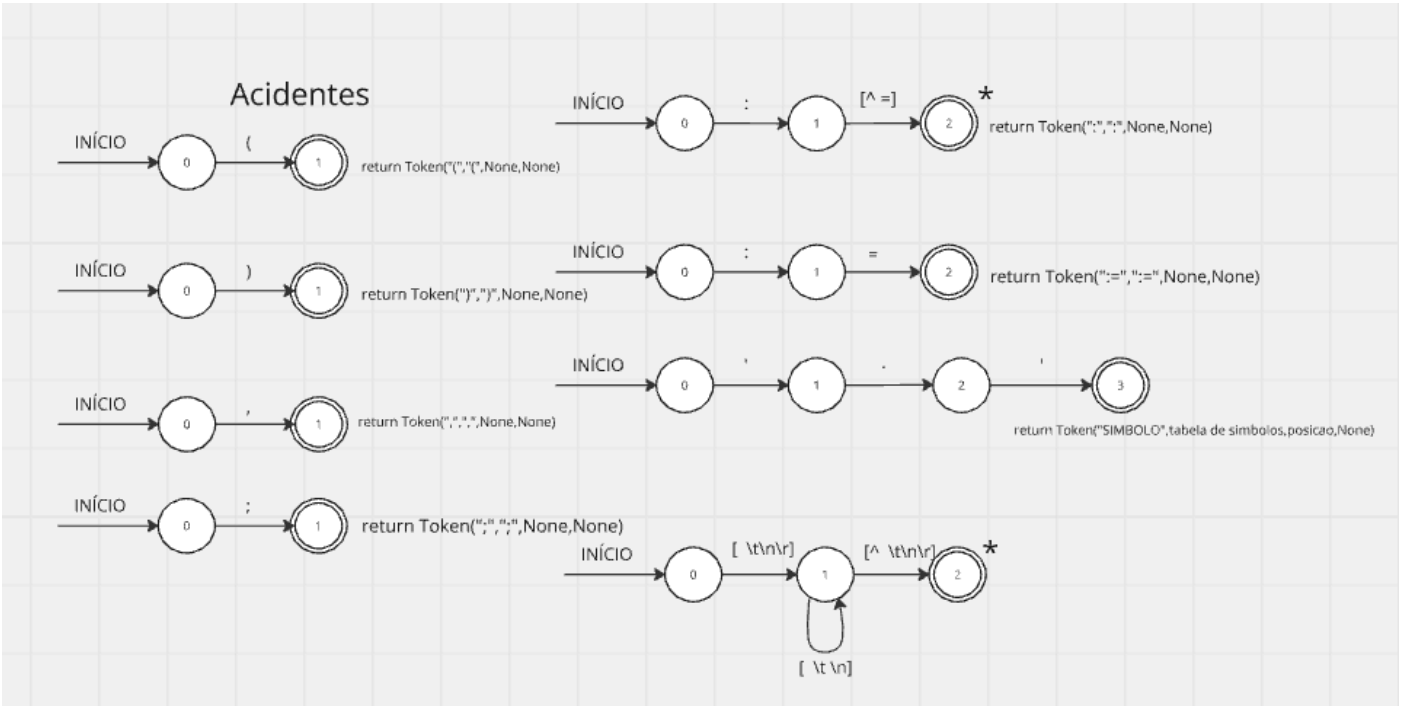
2.1 Diagramas de Transição para os Tokens

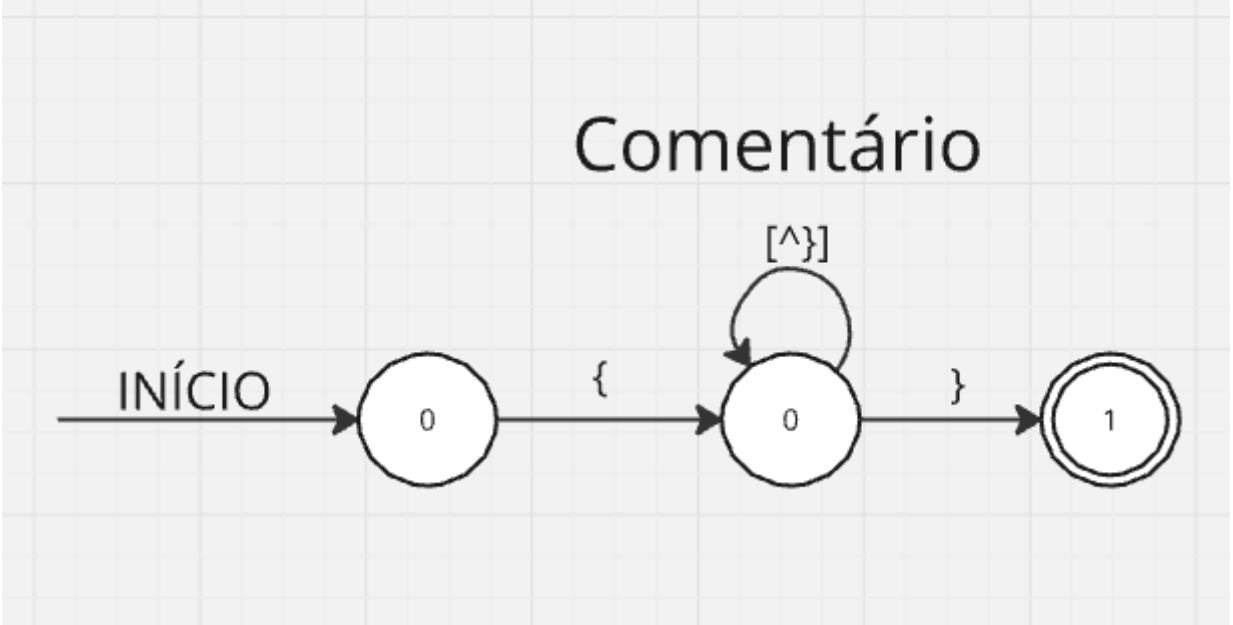


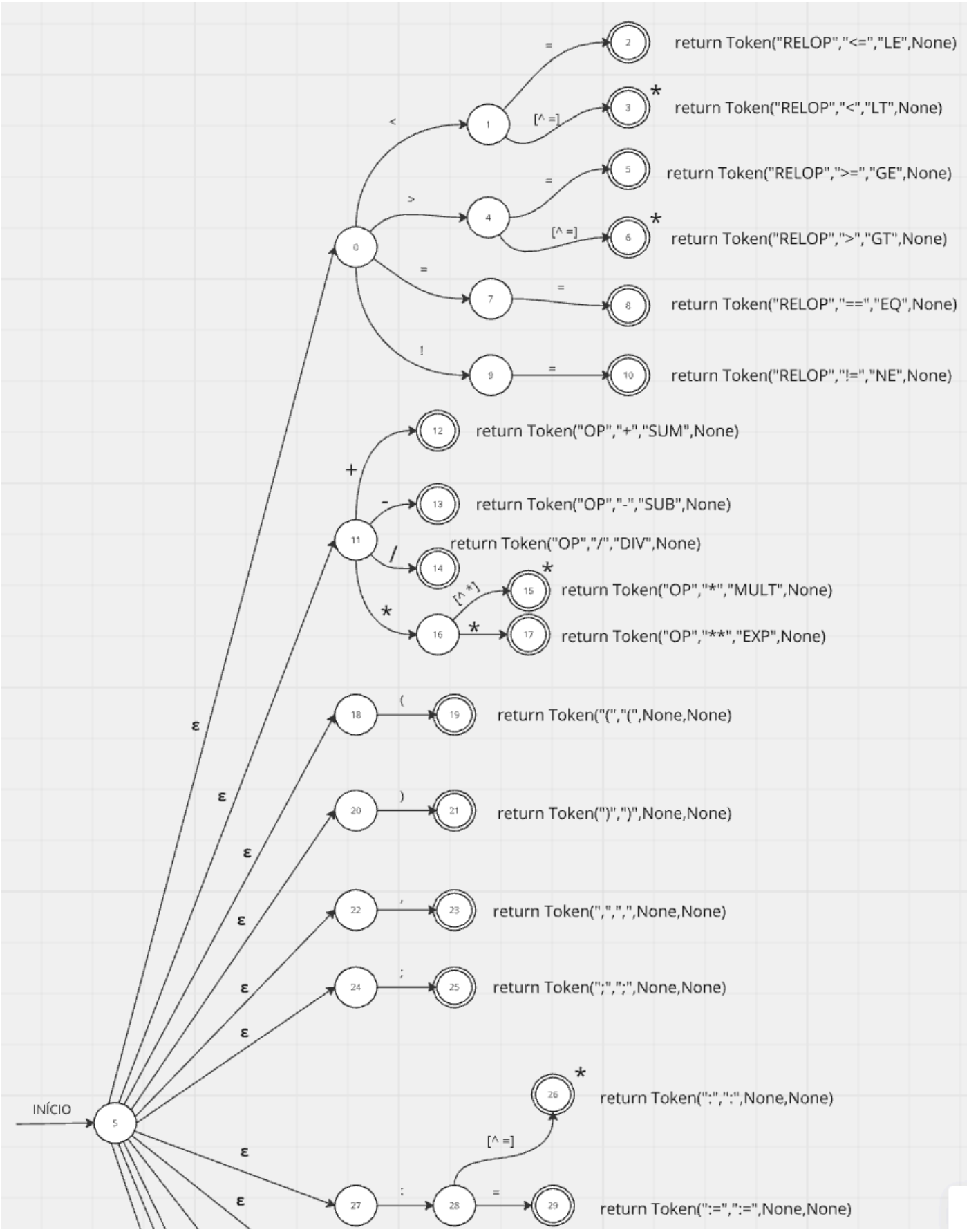

ID
INÍCIO
0
[letra _]
1
[^ letra dígito _]
2
*
return Token("ID",tabela de simbolos,posicao,None)
[letra dígito _]

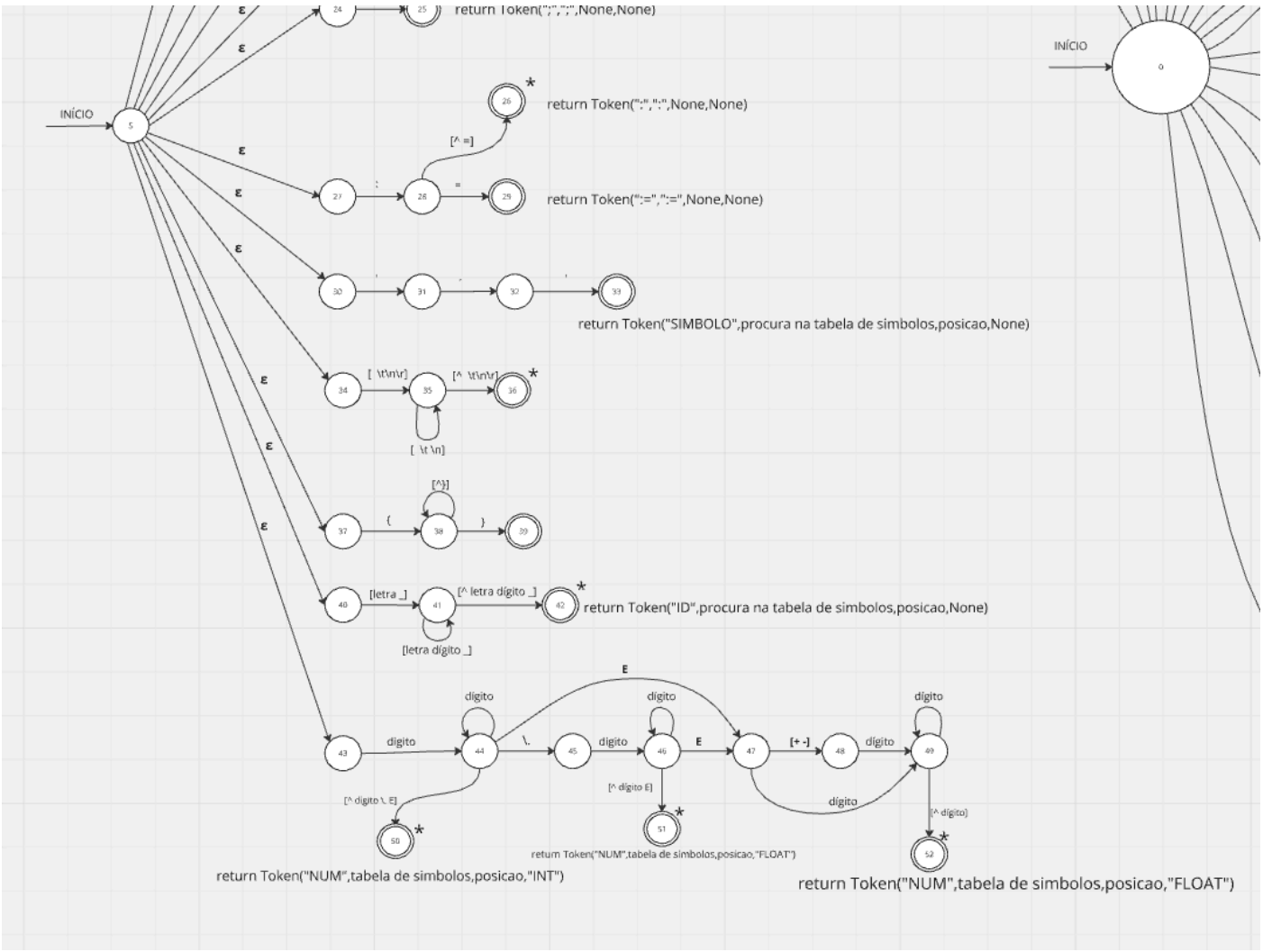

Operador
INÍCIO
0
+
1
return Token("OP","+","SUM",None)
-
2
return Token("OP","-","SUB",None)
/
3
return Token("OP","/","DIV",None)
*
4
[^ *]
5
*
return Token("OP","*","MULT",None)
*
6
return Token("OP","**","EXP",None)

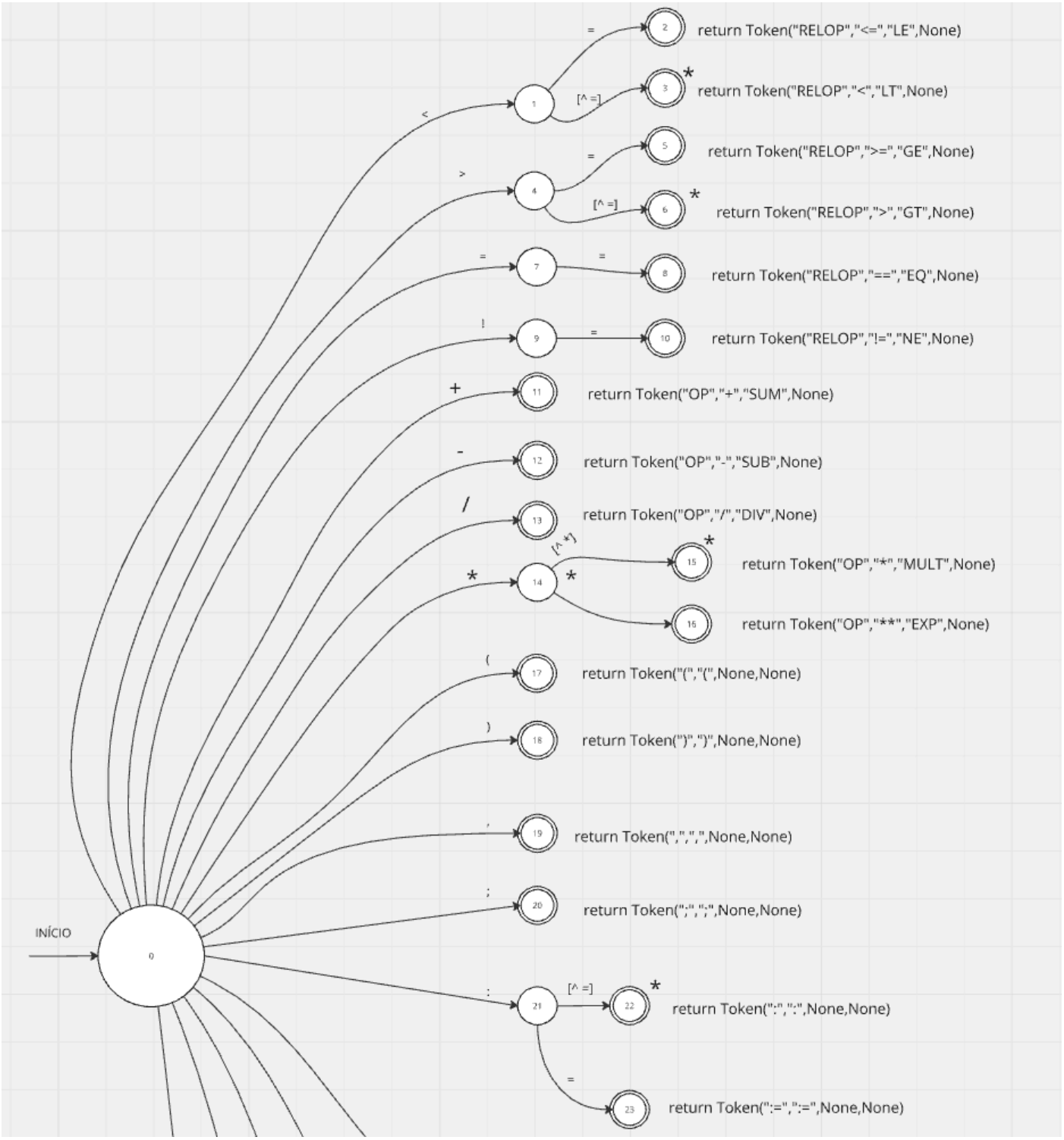

NÚMERO
INÍCIO
0
digito
1
dígito
.
3
digito
4
dígito
E
6
E
[-+]
7
dígito
8
dígito
[^ digito . E]
2
*
return Token("NUM",tabela de simbolos,posicao,"INT")
[^ dígitoE]
5
*
return Token("NUM",tabela de simbolos,posicao,"FLOAT")
dígito
[^ dígito]
9
*
return Token("NUM",tabela de simbolos,posicao,"FLOAT")

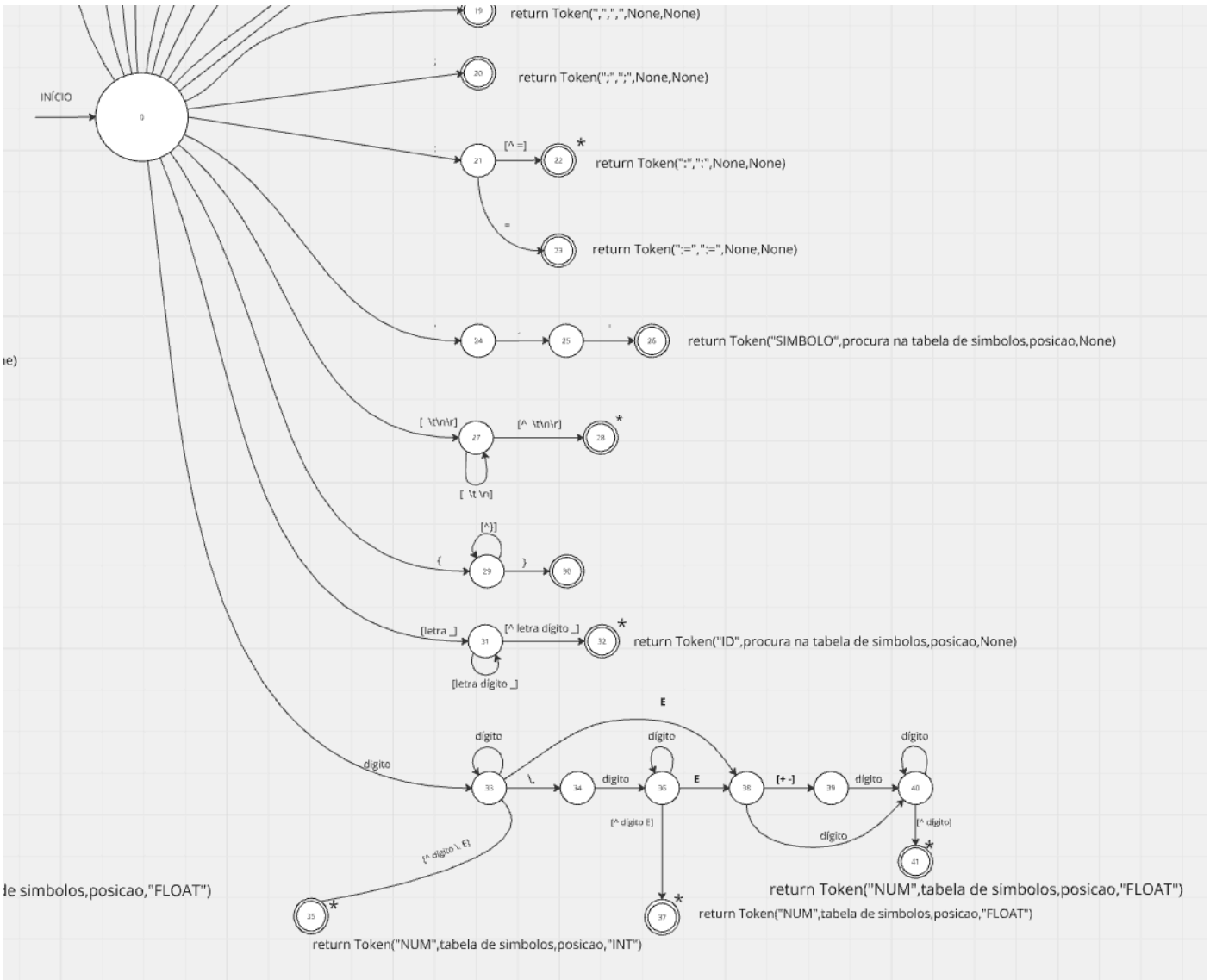

Acidentes
INÍCIO
0
(
1
return Token("(","(",None,None)
INÍCIO
0
)
1
return Token(")",")",None,None)
INÍCIO
0
,
1
return Token(",",",",None,None)
INÍCIO
0
;
1
return Token(";",";",None,None)
INÍCIO
0
:
1
[^ =]
2
*
return Token(":",":",None,None)
INÍCIO
0
:
1
=
2
return Token(":=",":=",None,None)
INÍCIO
0
'
1
.
2
'
3
return Token("SIMBOLO",tabela de simbolos,posicao,None)
INÍCIO
0
[\t\n\r]
1
[^ \t\n\r]
2
*
[\t \n]


Comentário
[^}]
INÍCIO
0
{
0
}
1


INÍCIO
5
ε
0
<
1
=
2
return Token("RELOP","<=","LE",None)
[^ =]
3
*
return Token("RELOP","<","LT",None)
>
4
=
5
return Token("RELOP",">=","GE",None)
[^ =]
6
*
return Token("RELOP",">","GT",None)
=
7
=
8
return Token("RELOP","==","EQ",None)
!
9
=
10
return Token("RELOP","!=","NE",None)
11
+
12
return Token("OP","+","SUM",None)
-
13
return Token("OP","-","SUB",None)
/
14
return Token("OP","/","DIV",None)
*
16
[^ *]
15
*
return Token("OP","*","MULT",None)
*
17
return Token("OP","**","EXP",None)
18
(
19
return Token("(","(",None,None)
20
)
21
return Token(")",")",None,None)
22
,
23
return Token(",",",",None,None)
24
;
25
return Token(";",";",None,None)
26
*
return Token(":",":",None,None)
[^ =]
27
:
28
=
29
return Token(":=",":=",None,None)


INÍCIO
ε
return Token(";",";",None,None)
return Token(":",":",None,None)
[^ =]
return Token(":=",":=",None,None)
return Token("SIMBOLO",procura na tabela de simbolos,posicao,None)
[\t\n\r]
[^ \t\n\r]
[\t \n]
[^}]
{
}
[letra _]
[^ letra dígito _]
return Token("ID",procura na tabela de simbolos,posicao,None)
[letra dígito _]
dígito
\.
E
[+ -]
[^ dígito \. E]
[^ dígito E]
[^ dígito]
return Token("NUM",tabela de simbolos,posicao,"INT")
return Token("NUM",tabela de simbolos,posicao,"FLOAT")
return Token("NUM",tabela de simbolos,posicao,"FLOAT")


INÍCIO
0
<
1
=
2
return Token("RELOP","<=","LE",None)
[^ =]
3
*
return Token("RELOP","<","LT",None)
>
4
=
5
return Token("RELOP",">=","GE",None)
[^ =]
6
*
return Token("RELOP",">","GT",None)
=
7
=
8
return Token("RELOP","==","EQ",None)
!
9
=
10
return Token("RELOP","!=","NE",None)
+
11
return Token("OP","+","SUM",None)
-
12
return Token("OP","-","SUB",None)
/
13
return Token("OP","/","DIV",None)
*
14
[^ *]
15
*
return Token("OP","*","MULT",None)
*
16
return Token("OP","**","EXP",None)
(
17
return Token("(","(",None,None)
)
18
return Token(")",")",None,None)
,
19
return Token(",",",",None,None)
;
20
return Token(";",";",None,None)
:
21
[^ =]
22
*
return Token(":",":",None,None)
=
23
return Token(":=",":=",None,None)


INÍCIO
0
19
return Token(",",",",None,None)
;
20
return Token(";",";",None,None)
:
21
[^ =]
22
*
return Token(":",":",None,None)
=
23
return Token(":=",":=",None,None)
'
24
.
25
'
26
return Token("SIMBOLO",procura na tabela de simbolos,posicao,None)
[\t\n\r]
27
[^ \t\n\r]
28
*
[\t \n]
{
[^}]
29
}
30
[letra _]
31
[^ letra digito _]
32
*
return Token("ID",procura na tabela de simbolos,posicao,None)
[letra digito _]
digito
33
digito
\.
34
digito
35
digito
E
E
36
[+ -]
39
digito
40
digito
[^ digito]
41
*
digito
[^ digito \. E]
35
*
return Token("NUM",tabela de simbolos,posicao,"INT")
[^ digito E]
37
*
return Token("NUM",tabela de simbolos,posicao,"FLOAT")
return Token("NUM",tabela de simbolos,posicao,"FLOAT")