Manual do Usuário Compilador D+

Lucas Ferreira de Almeida

13/10/2021

Conteúdo

1 Sobre o compilador

1.1 Árvore de diretorios e Arquivos

```
   -- DS_PLUS_COMPILER
------- Entrada (Pasta aonde fica o arquivo .d para ser compilado)
------- Src (Classes da aplicação)
------------ File.cs (Classe para manusear arquivos)
------------ LEXICA.cs (Classe para realizar a analise Léxica)
------------ MEPA.cs (Classe para realizar a geração de código MEPA)
------------ SEMANTICO.cs (Classe para realizar a analise Semântica)
------------ Simbolos.cs (Classe com os objetos utilizados na TS
------------ SINTATICO.cs (Classe para realizar a analise Sintática)
------------ Tokens.cs (Classe com os tokens)
------- doc
------------ DTs (Diagramas de transição)
------------------ Img (Diagramas de transição em imagem)
------------ Latex (Arquivo latex para gerar a documentação)
------------ mu.pdf (Manual do usuário)
------- Config.cs (Arquivo com as configurações globais da aplicação)
------- DS_PLUS_COMPILER.cs (Main)
```

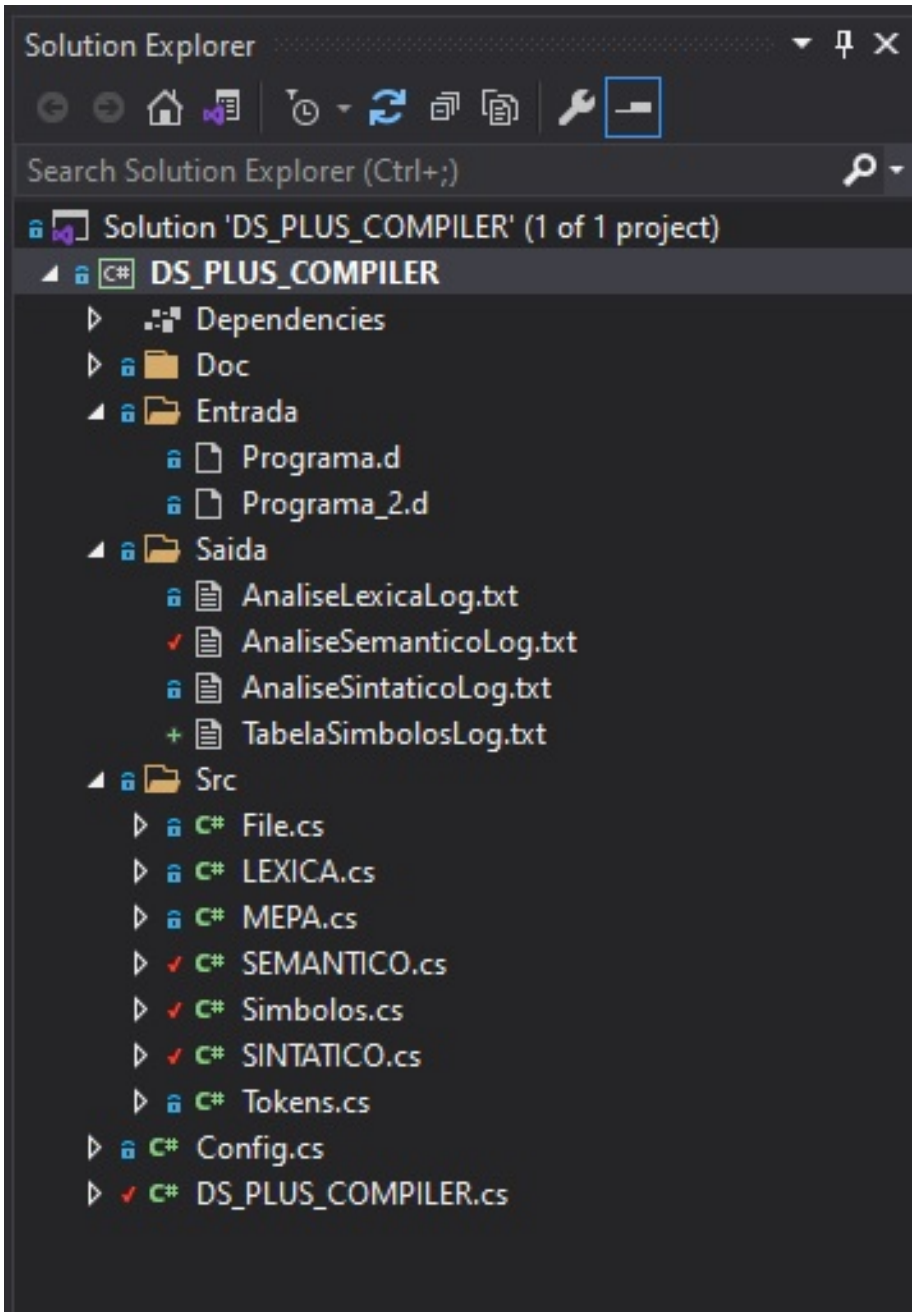

Figura 1: Árvore da aplicação.

1.2 Como utilizar

O arquivo de entrada deverá ser copiado para a pasta "Entrada", após isso é necessário rodar a aplicação. Todas as etapas das análises serão printadas no console e serão gerados arquivos com cada respectiva etapa.

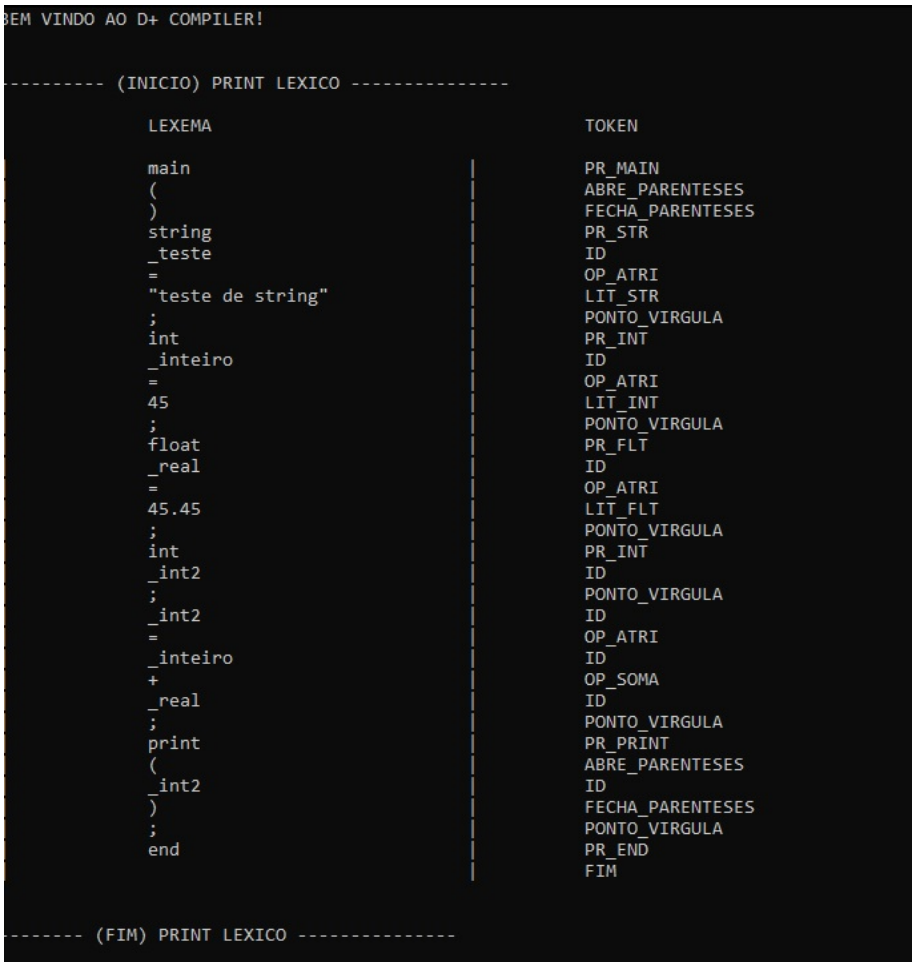

Figura 2: Print no console da analise Léxica.

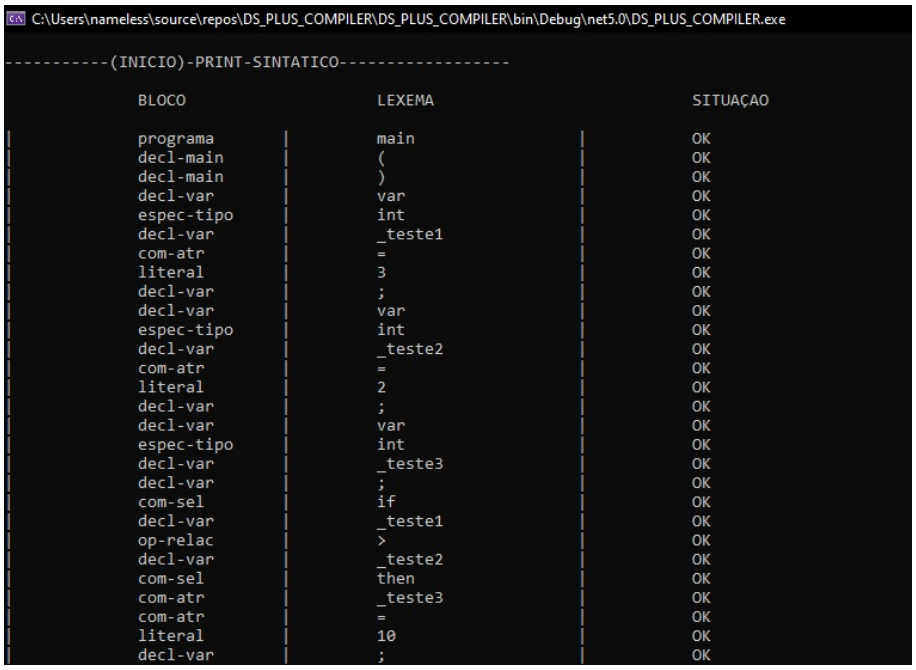

Figura 3: Print no console da analise Sintática.

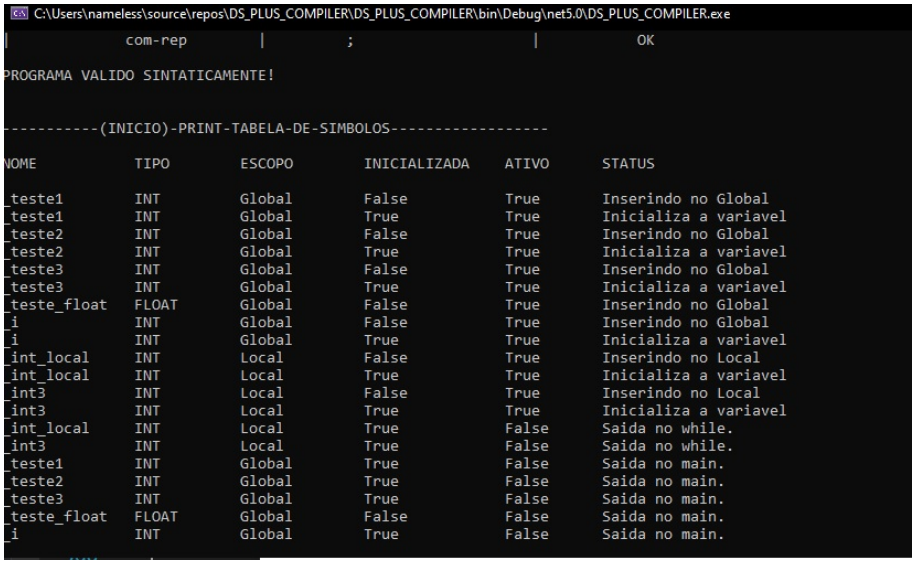

Figura 4: Print no console da Tabela de simbolo.

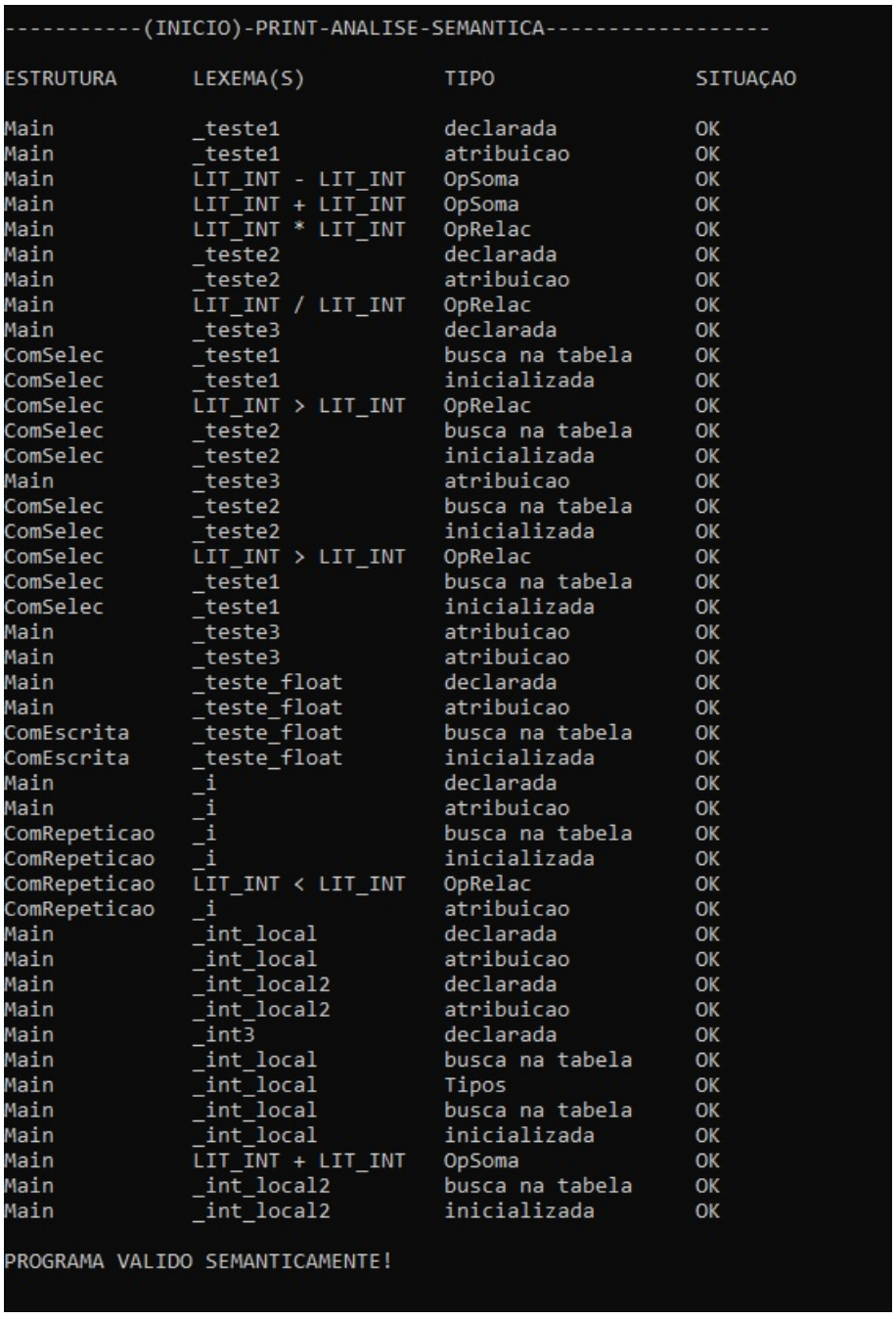

Figura 5: Print no console da analise Semântica.

2 Tokens

2.1 Identificadores

Identificador	ID

2.2 Literais

Inteiros	LIT_INT
Reais	LIT_FLT
Caractere	LIT_CHAR
String	LIT_STR
Boolean	LIT_BOOL

2.3 Palavras Reservadas

void	PR_VOID
int	PR_INT
float	PR_FLT
char	PR_CHAR
bool	PR_BOOL
if	PR_IF
then	PR_THEN
else	PR_ELSE
end	PR_END
for	PR_FOR
while	PR_WHILE
do	PR_DO
loop	PR_LOOP
return	PR_RETURN
true	PR_TRUE
false	PR_FALSE
var	PR_VAR
main	PR_MAIN
scan	PR_SCAN
print	PR_PRINT

2.4 Pontuação

,	VIRGULA
;	PONTO_VIRGULA
(	ABRE_PARENTESES
)	FECHA_PARENTESES
[	ABRE_COLCHETES
]	FECHA_COLCHETES
{	ABRE_CHAVES
}	FECHA_CHAVES

2.5 Operadores

+	OP_SOMA
-	OP_SUB
*	OP_MULT
/	OP_DIV
%	OP_MOD
.	OP_PONTO
<	OP_MENOR
>	OP_MAIOR
==	OP_IGUAL
!=	OP_DIF
<=	OP_MEN_IGUAL
>=	OP_MAI_IGUAL
=	OP_ATRI

3 Diagramas de transição

3.1 Identificadores

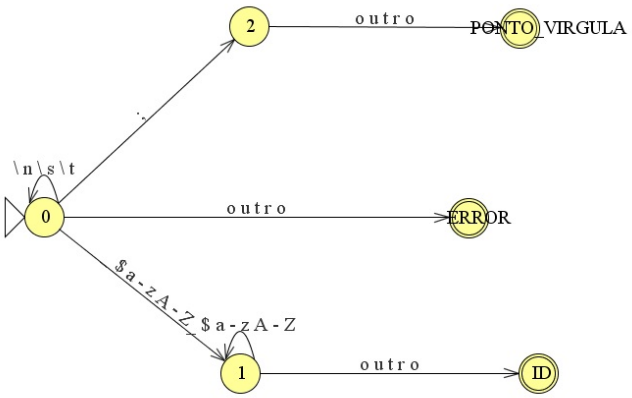

Figura 6: Indetificadores.

3.2 Literais

Figura 7: Literais (String).

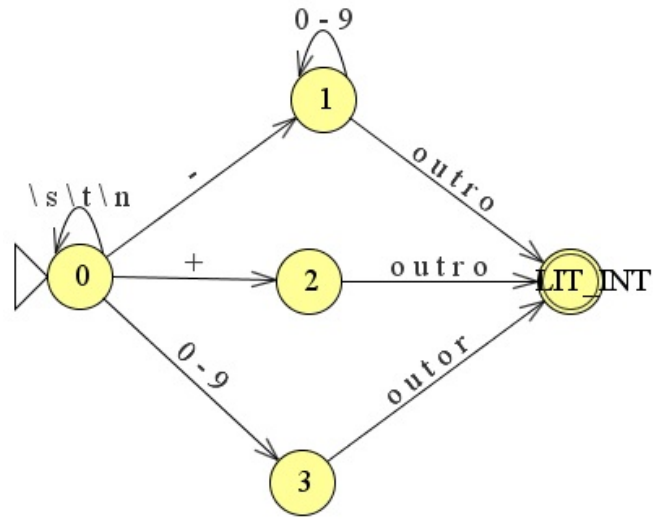

Figura 8: Literais (Inteiro).

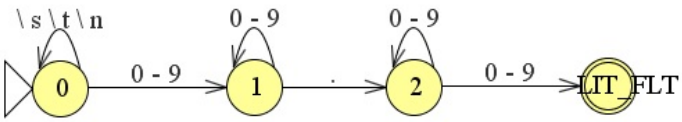

Figura 9: Literais (Reais).

Figura 10: Literais (Char).

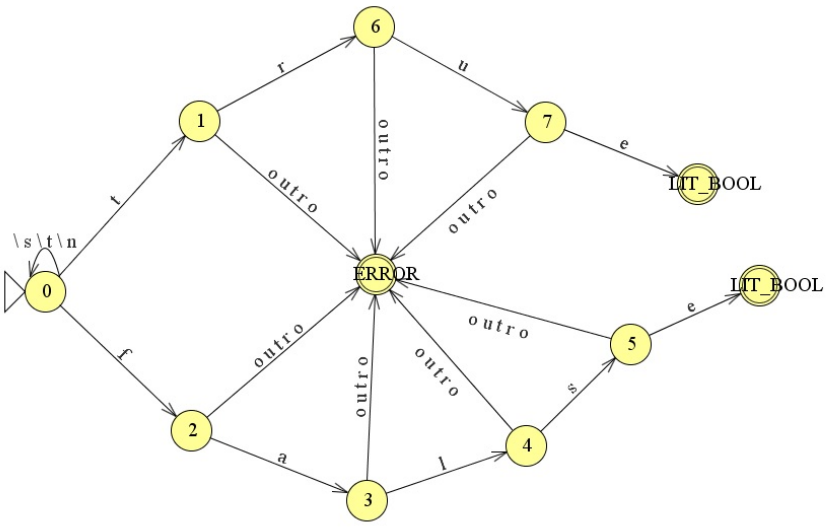

Figura 11: Literais (Boolean).

3.3 Pontuação

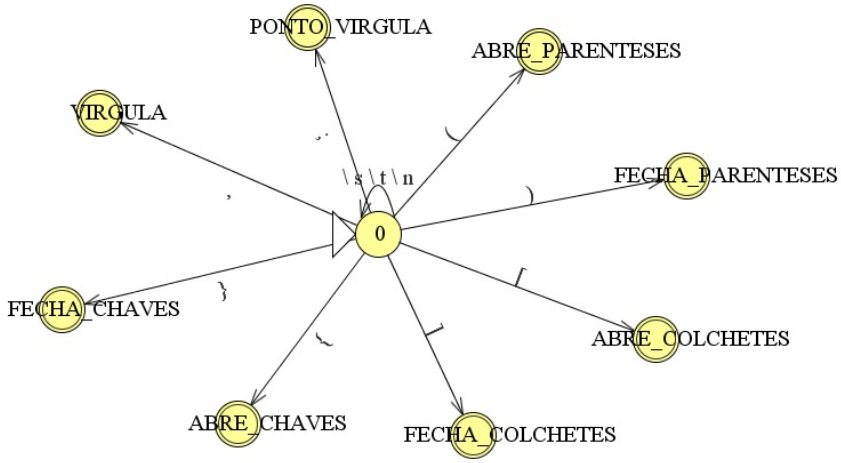

Figura 12: Literais (Boolean).

3.4 Palavras Reservadas

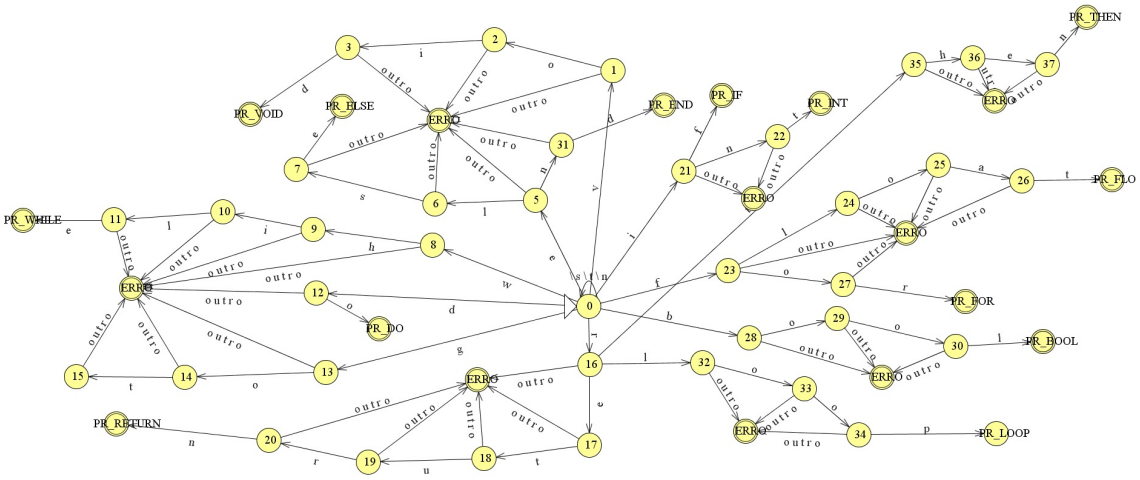

Figura 13: Literais (Boolean).

3.5 Operadores

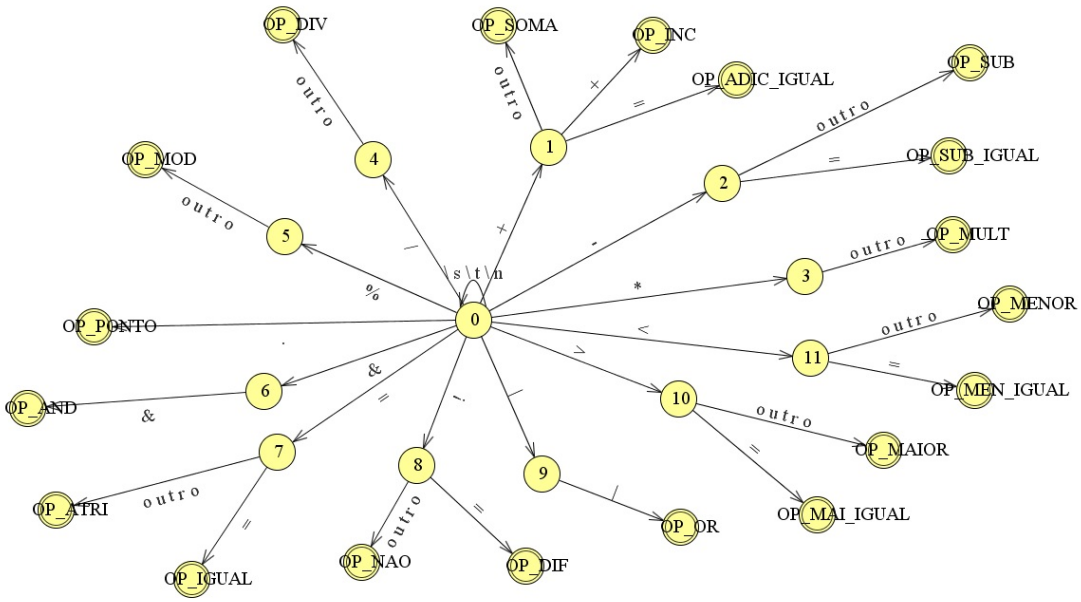

Figura 14: Literais (Boolean).

4 Fatoração

```
programa → lista-decl

// lista-decl → decl lista-decl | decl
lista-decl → decl lista-decl'
lista-decl' → decl | &

decl → decl-var | decl-main
decl-var → VAR espec-tipo var ;
decl-main → MAIN ( ) bloco END

espec-tipo → INT | REAL | CHAR
lista-com → comando lista-com | &
comando → decl-var | com-atrib | com-selecao | com-repeticao | com-leitura | com-escrita

com-atrib → var = exp ;
com-leitura → SCAN ( var ) ;
com-escrita → PRINT ( exp ) ;

// com-selecao → IF exp THEN bloco END | IF exp THEN bloco ELSE bloco END
com-selecao → IF exp THEN bloco com-selecao'
com-selecao' → END | ELSE bloco END

bloco → lista-com
com-repeticao → WHILE exp DO bloco LOOP

// exp → exp-soma op-relac exp-soma | exp-soma
exp → exp-soma exp'
exp' → op-relac exp-soma | &

op-relac → <= | < | > | >= | == | <>
// exp-soma → exp-mult op-soma exp-soma | exp-multi
exp-soma → exp-mult exp-soma'
exp-soma' → op-soma exp-soma |
op-soma → + | -

// exp-mult → exp-simples op-mult exp-mult | exp-simples
exp-mult → exp-simples exp-mult'
exp-mult' → op-mult exp-mult | &
op-mult → * | / | DIV | MOD
exp-simples → ( exp ) | var | literal

literal → NUMINT | NUMREAL | CHAR | STRING
var → ID
```

5 Analise do Primeiro Símbolo

APS	P	P+	t
programa	lista-decl	P(lista-decl)	VAR, MAIN()
list-decl	decl	P(decl)	VAR, MAIN()
lista-decl'	decl, &	P(decl), S(lista-decl')-> S(lista-decl)-> S(programa)-> #	VAR, MAIN()
decl	decl-var, decl-main	P(decl-var), P(decl-main)	VAR, MAIN()
decl-var	VAR	VAR	VAR
decl-main	MAIN()	MAIN()	MAIN()
espec-tipo	INT, REAL, CHAR	INT, REAL, CHAR	INT, REAL, CHAR
bloco	lista-com	P(lista-com)	VAR, ID, IF, WHILE, SCAN(, PRINT(, END
lista-com	comando, &	P(comando), S(lista-com)-> S(bloco)-> #	VAR, ID, IF, WHILE, SCAN(, PRINT(, END
comando	decl-var, com-atrib, com-selecao, com-repeticao, com-leitura, com-escrita	P(decl-var), P(com-atrib), P(com-selecao), P(com-repeticao), P(com-leitura), P(com-escrita)	VAR, ID, IF, WHILE, SCAN(, PRINT(
com-atrib	var	P(var)	ID
com-leitura	SCAN(	SCAN(	SCAN(
com-escrita	PRINT(	PRINT(	PRINT(
com-selecao	IF	IF	IF
com-selecao'	END, ELSE	END, ELSE	END, ELSE
com-repeticao	WHILE	WHILE	WHILE
exp	exp-soma	P(exp-soma)	(, INT, REAL, CHAR, STR, ID

exp'	op-relac, &	P(op-relac), S(exp')-> S(exp)->);	<=, <, >, >=, ==, <>,);
op-relac	<=, <, >, >=, ==, <>	<=, <, >, >=, ==, <>	<=, <, >, >=, ==, <>
exp-soma	exp-mult	P(exp-mult)	(, INT, REAL, CHAR, STR, ID
exp-soma'	op-soma, &	P(op-soma), S(exp-soma')-> S(exp-soma)-> S(exp')-> S(exp)->);	+, -,);
op-soma	+, -	+, -	+, -
exp-mult	exp-simples	P(exp-simples)	(, INT, REAL, CHAR, STR, ID
exp-mult'	op-mult, &	P(op-mult), S(exp-mult')-> S(exp-mult)-> P(exp-soma))	*,/,DIV, MOD,+,-,);
op-mult	*,/,DIV,MOD	*,/,DIV,MOD	*,/,DIV,MOD
exp-simples	(, var, literal	(, P(var), P(literal)	(, INT, REAL, CHAR, STR, ID
literal	INT, REAL, CHAR, STR	(, P(var), P(lit)	(, INT, REAL, CHAR, STR, ID
var	ID	ID	ID

6 Tabela de Derivação

6.1 Declarações

TD	VAR	MAIN()	INT	REAL	CHAR
programa	lista-decl	lista-decl	-	-	-
lista-decl	decl	decl	-	-	-
decl	decl-var	decl-main	-	-	-
decl-var	VAR	-	-	-	-
decl-main	-	MAIN()	-	-	-
espec-tipo	-	-	INT	REAL	CHAR

6.2 Comandos

TD	SCAN(	PRINT(	IF	WHILE	VAR	ID	END
bloco	lst-com	lst-com	lst-com	lst-com	lst-com	lst-com	lst-com
lst-com	comando	comando	comando	comando	comando	comando	comando
comando	com-lei	com-lei	com-lei	com-lei	com-atr	-	-
com-atr	-	-	-	-	var	-	-
com-lei	SCAN(	-	-	-	-	-	-
com-esc	-	PRINT(	-	-	-	-	-
com-sel	-	-	IF	-	-	-	-
com-rep	-	-	-	WHILE	-	-	-
var	-	-	-	-	-	ID	-

6.3 Expressões

TD	<=	<	>	>=	==	<>	+	-	*	/
exp	-	-	-	-	-	-	-	-	-	-
op-rel	<=	<	>	>=	==	<>	-	-	-	-
exp-som	-	-	-	-	-	-	-	-	-	-
op-som	-	-	-	-	-	-	+	-	-	-
exp-mul	-	-	-	-	-	-	-	-	-	-
op-mul	-	-	-	-	-	-	-	-	*	/
exp-sim	-	-	-	-	-	-	-	-	-	-
lit	-	-	-	-	-	-	-	-	-	-
var	-	-	-	-	-	-	-	-	-	-

TD	DIV	MOD	INT	REAL	CHAR	STR	ID	(	);
exp	-	-	exp-som	exp-som	exp-som	exp-som	exp-som	exp-som	-
op-rel	-	-	-	-	-	-	-	-	-
exp-som	-	-	exp-mul	exp-mul	exp-mul	exp-mul	exp-mul	exp-mul	-
op-som	-	-	-	-	-	-	-	-	-
exp-mul	-	-	exp-sim	exp-sim	exp-sim	exp-sim	exp-sim	exp-sim	-
op-mul	DIV	MOD	-	-	-	-	-	-	-
exp-sim	-	-	lit	lit	lit	lit	var	(	-
lit	-	-	INT	REAL	CHAR	STR	-	-	-
var	-	-	-	-	-	-	ID	-	-

7 Validações

7.1 Lexica

Tipo	Mensagem
Erro	'Erro': Operador ou pontuação inválida. Lexema = '{Lexema}'.
Erro	'Erro': Char inválido. Lexema = '{Lexema}'.
Erro	'Erro': Comando não identificado. Lexema = '{Lexema}'.

7.2 Sintática & Semântica

Mensagem
Erro na estrutura '{estrutura}', esperado '{lexemas_esperados}' encontrado lexema '{lexema}'.
Erro na linha '{linha}' -> Tipo diferente da variavel declarada '{lexima}'.
Erro na linha '{linha}' -> Váriavel '{lexima}' nao inicializada.
Erro na linha '{linha}' -> Váriavel '{lexima}' já declarada.
Erro na linha '{linha}' -> Operaçao '{literal}' * '{literal}' inválida semânticamente.