Tokens Utilizados

Palavras Reservadas

void	PR_VOID
int	PR_INT
float	PR_FLOAT
char	PR_CHAR
bool	PR_BOOL
if	PR_IF
else	PR_ELSE
for	PR_FOR
while	PR_WHILE
do	PR_DO
return	PR_RETURN
break	PR_BREAK
continue	PR_CONTINUE
goto	PR_GOTO
true	PR_TRUE
false	PR_FALSE

Operadores

+	OP_ADICAO
-	OP_SUBTRACAO
*	OP_MULTIPLICACAO
/	OP_DIVISAO
%	OP_MODULO
?	OP_TERNARIO
:	OP_DOISPONTOS
!	OP_NEGACAO
&	OP_ENDERECO
.	OP_PONTO
->	OP_FLECHA
<	OP_MENOR
>	OP_MAIOR
==	OP_IGUALDADE
!=	OP_DIFERENCA
<=	OP_MENORIGUAL
>=	OP_MAIORIGUAL

=	OP_IGUAL
+=	OP_ADICAOIGUAL
-=	OP_SUBTRACAOIGUAL
*=	OP_MULTIPLICACAOIGUAL
/=	OP_DIVISAOIGUAL
%=	OP_MODULOIGUAL
++	OP_INCREMENTO
&&	OP_AND
||	OP_OR

Sinais de Pontuação

,	SP_VIRGULA
;	SP_PONTOEVIRGULA
(	SP_ABREPARENTESES
)	SP_FECHAPARENTESES
[	SP_ABRECOLCHETES
]	SP_FECHACOLCHETES
{	SP_ABRECHAVES
}	SP_FECHACHAVES

Literais Básicos

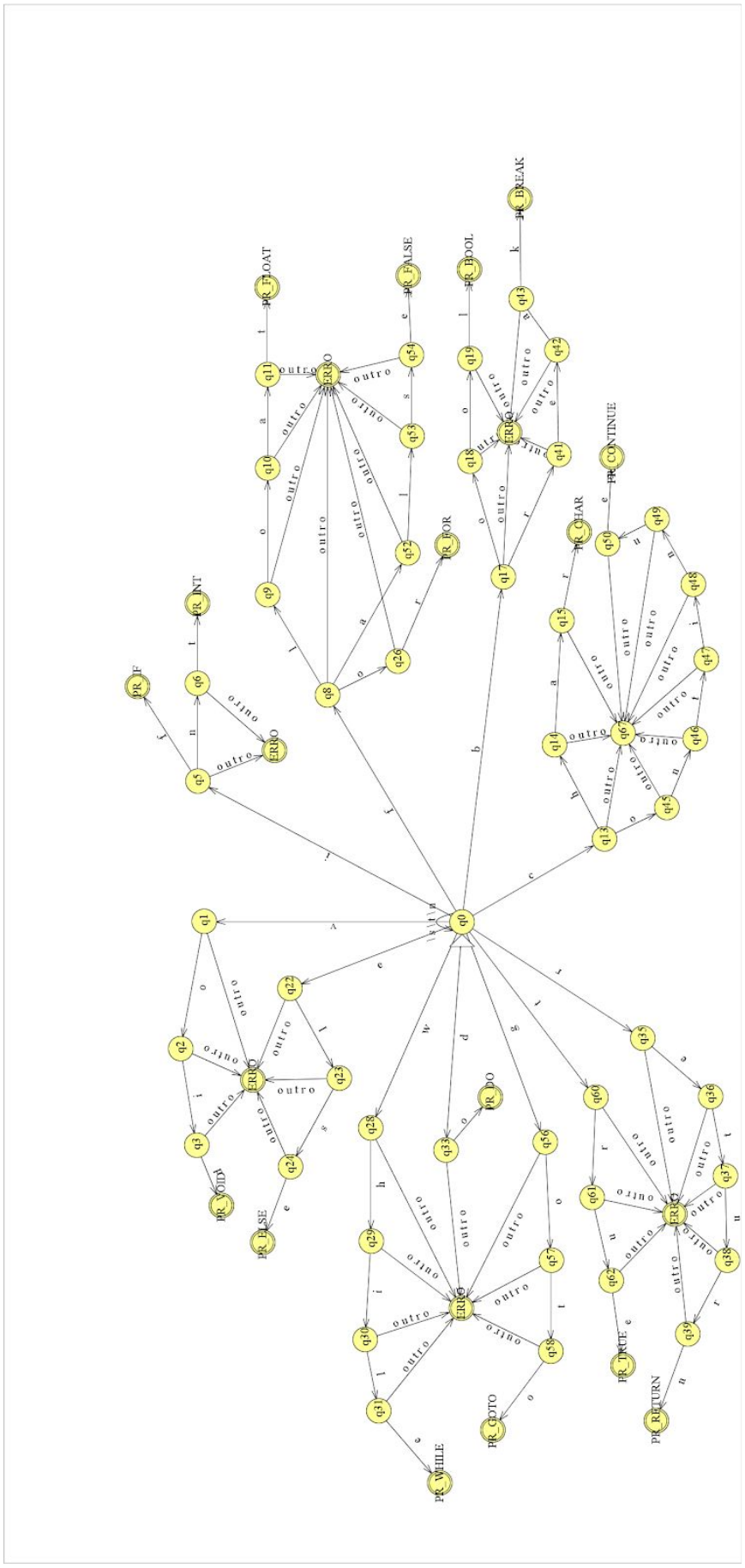
inteiros	LB_INT
reais	LB_FLOAT
caracteres	LB_CHAR
strings	LB_STRING
booleanos	LB_BOOL

Identificadores

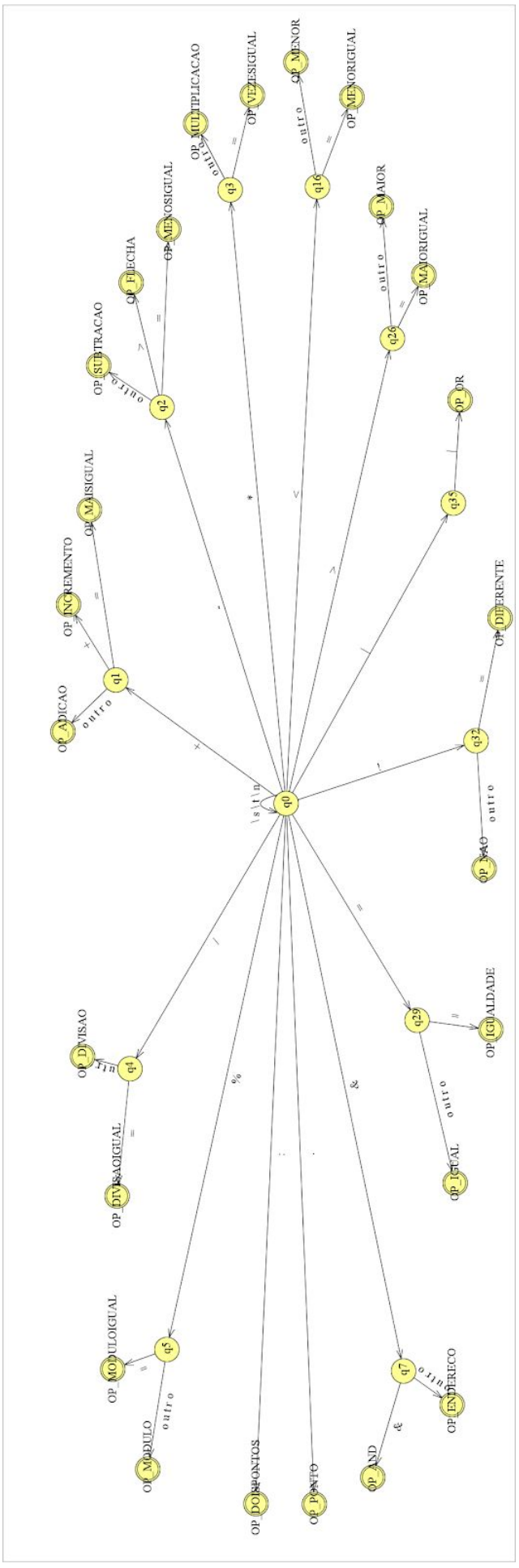
identificadores	ID

Diagramas de Transição

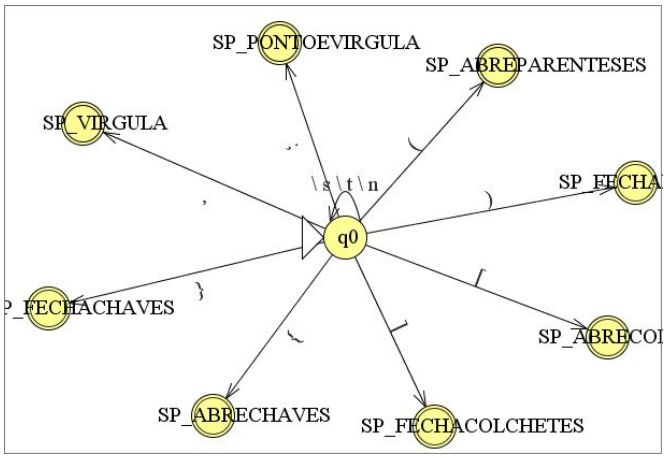
Palavras Reservadas



Operadores

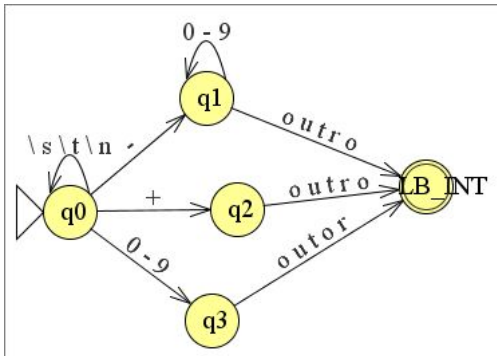

q0
q1
q2
q3
q4
q5
q7
q16
q26
q29
q32
q35
OP_ADICAO
OP_INCREMENTO
OP_MAISIGUAL
OP_SUBTRACAO
OP_SETA
OP_MENOSIGUAL
OP_MULTIPLICACAO
OP_VEZESIGUAL
OP_MENOR
OP_MENORIGUAL
OP_MAIOR
OP_MAIORIGUAL
OP_OR
OP_DIFERENTE
OP_IGUALDADE
OP_IGUAL
OP_DIVISAO
OP_DIVISAOIGUAL
OP_MODULOIGUAL
OP_MODULO
OP_AND
OP_ENDERECO
OP_DOISPONTOS
OP_PONTO

Sinais de Pontuação

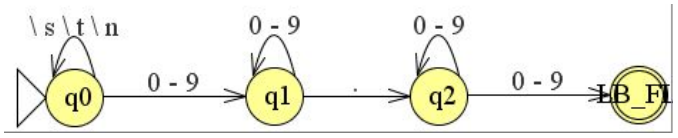


Literais Básicos

Inteiros

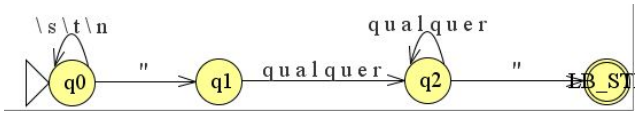


Reais

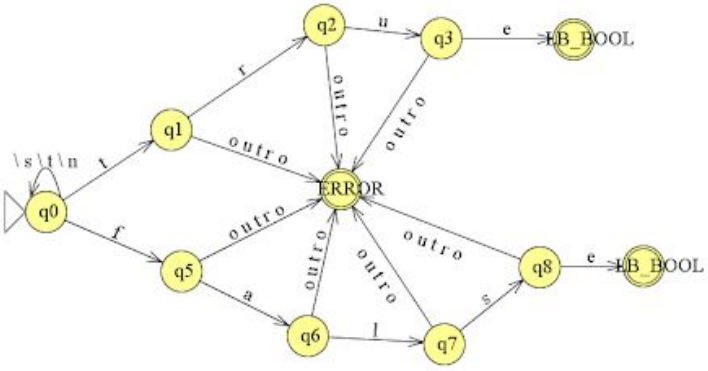


Caracteres

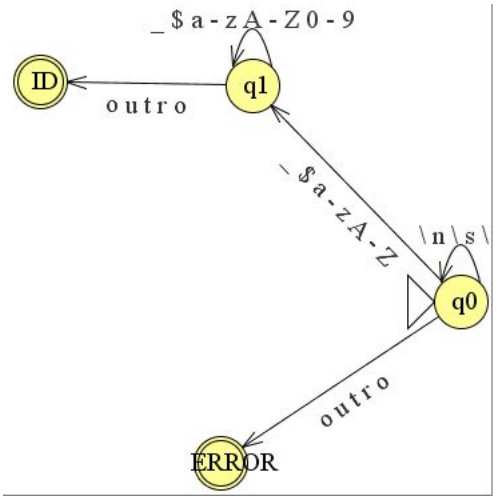
Strings



Booleanos



Identificadores



Técnicas utilizadas na gramática

```
programa → lista-decl
lista-decl → 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
bloco → lista-com
lista-com → comando lista-com | ε
comando → decl-var | com-atrib | com-selecao | com-repeticao | com-leitura | com-escrita
com-atrib → var = exp ;
com-leitura → SCAN ( var ) ; | SCANLN ( var ) ;
com-escrita → PRINT ( exp ) ; | PRINTLN ( exp ) ;
com-selecao → IF exp THEN bloco END-IF | IF exp THEN bloco ELSE bloco END-IF
com-repeticao → WHILE exp DO bloco LOOP
exp → exp-soma op-relac exp-soma | exp-soma
op-relac → <= | < | > | >= | == | <>
exp-soma → exp-mult op-soma exp-soma | exp-multi
op-soma → + | -
exp-mult → exp-simples op-mult exp-mult | exp-simples
op-mult → * | / | DIV | MOD
exp-simples → ( exp ) | var | literal
literal → NUMINT | NUMREAL | CARACTERE | STRING
var → ID
```

APS	P	P+	t
programa	lista-decl	Prim(lista-decl)	VAR, MAIN
lista-decl	decl	Prim(decl)	VAR, MAIN
decl	decl-var, decl-main	Prim(decl-var), Prim(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	Prim(lista-com)	VAR, ID, IF, IF, WHILE, SCAN, SCANL, PRINT, PRINTLN, END
lista-com	comando, ε	Prim(comando), Segue(lista-com) ->Segue(bloco) -> END	VAR, ID, IF, IF, WHILE, SCAN, SCANL, PRINT, PRINTLN, END
comando	decl-var, com-atrib, com-selecao, com-repeticao, com-leitura, com-escrita	Prim(decl-var), Prim(com-atrib), Prim(com-selecao), Prim(com-repeticao), Prim(com-leitura), Prim(com-escrita),	VAR, ID, IF, IF, WHILE, SCAN, SCANL, PRINT, PRINTLN
com-atrib	var	Prim(var)	ID
com-leitura	SCAN, SCANLN	SCAN, SCANLN	SCAN, SCANLN
com-escrita	PRINT, PRINTLN	PRINT, PRINTLN	PRINT, PRINTLN
com-selecao	IF, IF	IF, IF	IF, IF
com-repeticao	WHILE	WHILE	WHILE
exp	exp-soma	Prim(exp-soma)	(, NUMINT, NUMREAL, CARACTER, STRING, ID

op-relac	<=, <, >, >=, ==, <>	<=, <, >, >=, ==, <>	<=, <, >, >=, ==, <>
exp-soma	exp-mult	Prim(exp-mult)	(, NUMINT, NUMREAL, CARACTER, STRING, ID
op-soma	+, -	+, -	+, -
exp-mult	exp-simples	Prim(exp-simples)	(, NUMINT, NUMREAL, CARACTER, STRING, ID
op-mult	*, /, DIV, MOD	*, /, DIV, MOD	*, /, DIV, MOD
exp-simples	(, var, literal	(, Prim(var), Prim(literal)	(, NUMINT, NUMREAL, CARACTER, STRING, ID
literal	NUMINT, NUMREAL, CARACTER, STRING	NUMINT, NUMREAL, CARACTER, STRING	NUMINT, NUMREAL, CARACTER, STRING
var	ID	ID	ID

Mensagens de Erro