



Universidade Federal do Ceará  
Campus Crateús

Ciência da Computação  
Compiladores

# **Expressões Regulares da Linguagem C**

Francisco David Nascimento Sousa 412772  
Lucas Chaves Evangelista 427671

1 de Setembro de 2020

# 1 Expressões regulares da linguagem C

Descrição das expressões regulares:

Palavra:

```
W = [A-Za-z]
```

Número:

```
D = [0-9]
```

Identificador:

```
id = <W>(<W> + <D>)*
```

Tipo de dados:

```
DT = int | float | char | void
```

Char literal:

```
CL = '<W>'
```

String literal:

```
SL = "<W> + <D>*"
```

Inteiro literal:

```
IL => (<D>)+
```

Flutuante literal:

```
FL => (<D>)+.(<D>)+
```

Operadores booleana:

```
boolean_operator = == | != | ! | >= | > | <= | < | && | ||
```

Operadores aritméticos:

```
arithmetic_operator = + | - | * | / | % | ++ | -- | += |  
*= | /=
```

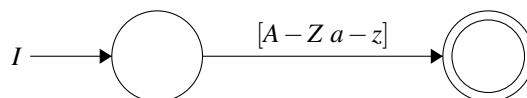
Delimitadores:

```
delimiter = , | ;
```

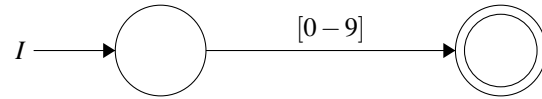
## 2 Autômatos C

Autômatos das expressões acima:

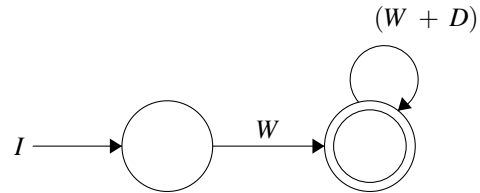
Word (W):



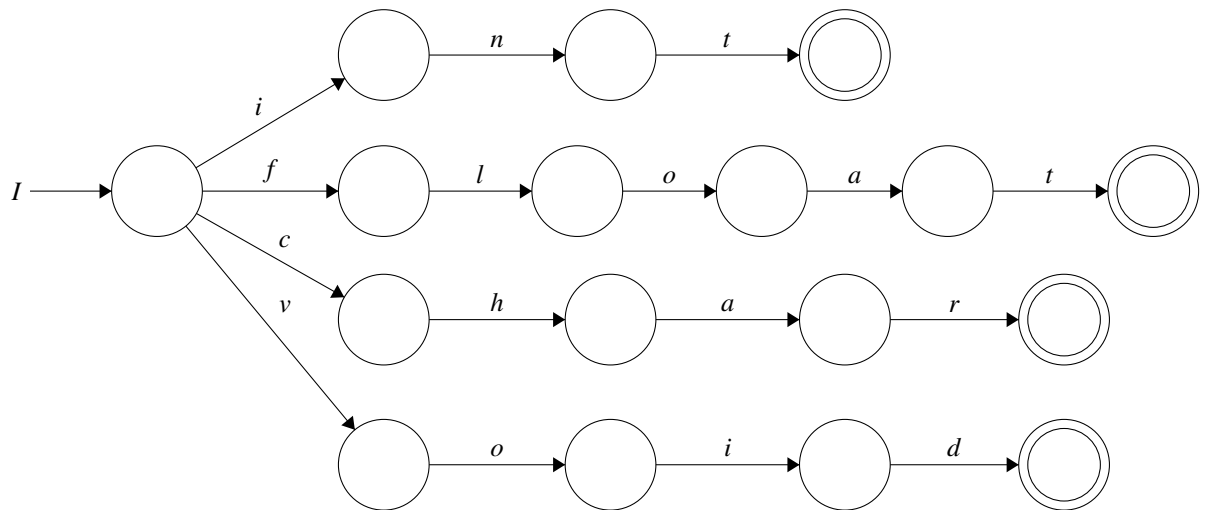
Digito (D):



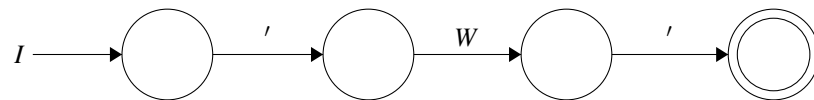
Identificador (id):



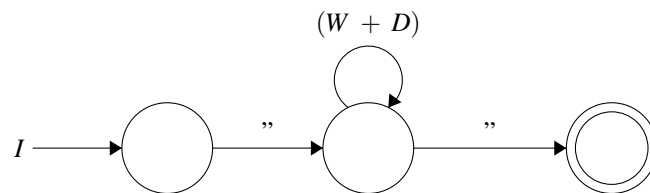
Tipo de dados (DT):



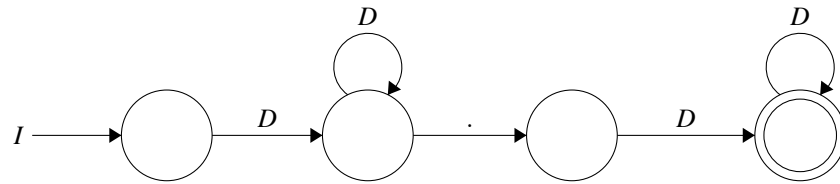
Char literal (CL):



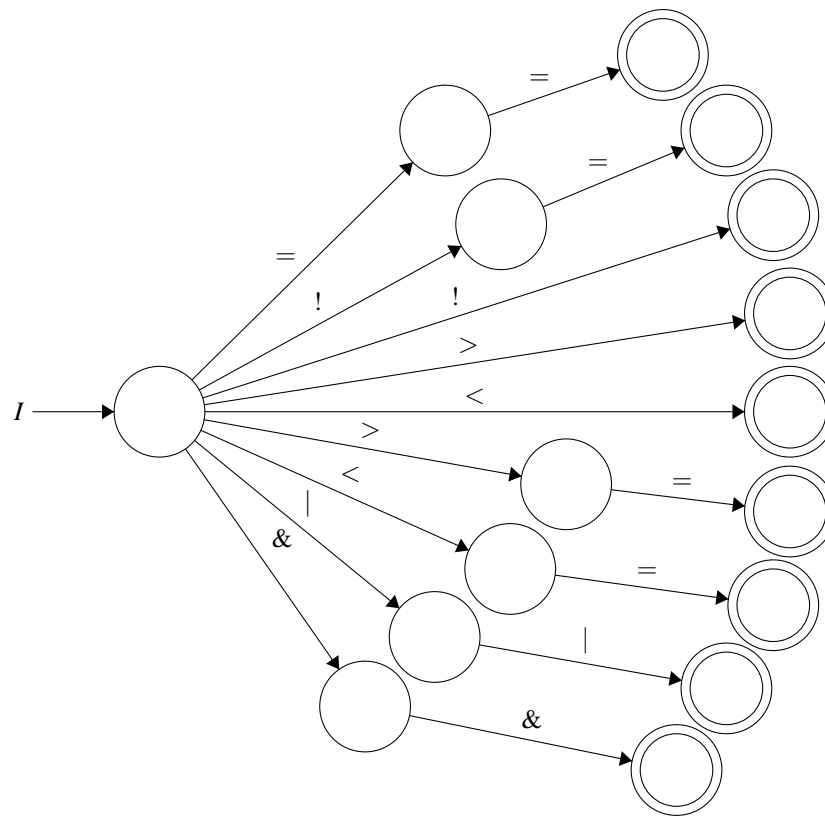
String literal (SL):



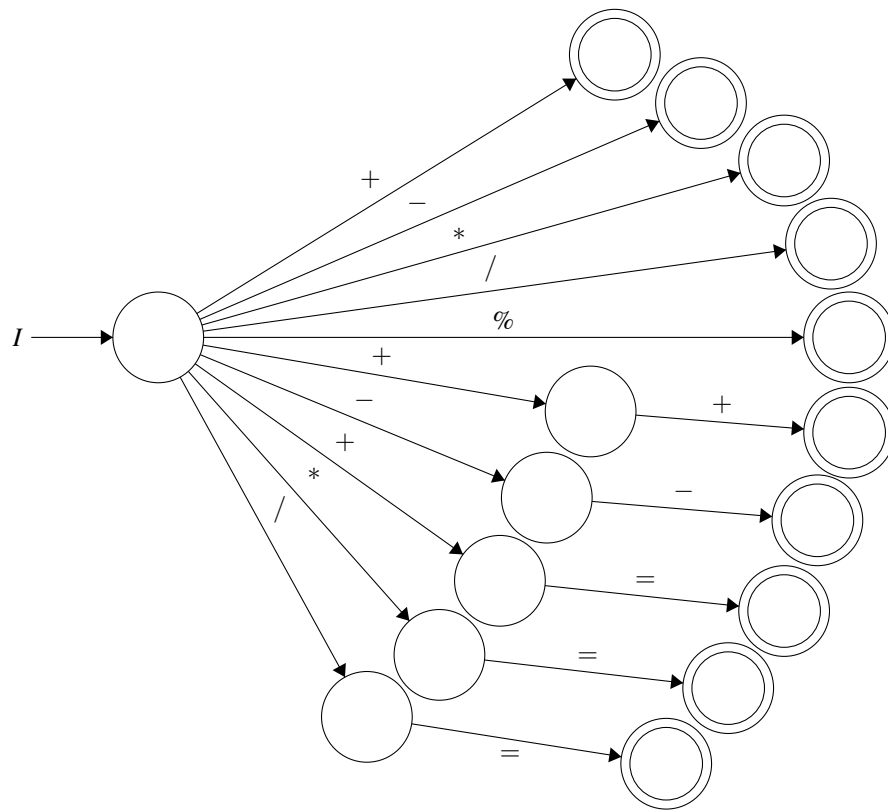
Float literal (FL):



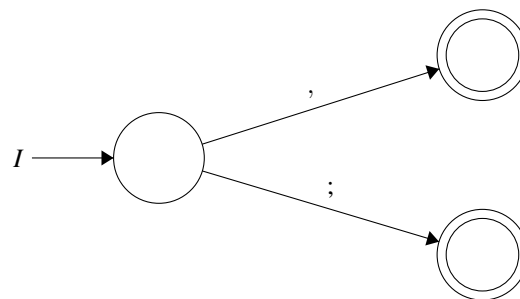
Operadores booleanos:



Operadores aritméticos:



Delimitadores:



### 3 Definições da gramática da linguagem C

$P \Rightarrow \text{void main } () \langle B \rangle \text{ (1)}$

$E \Rightarrow \langle \text{id} \rangle = \langle E \rangle \text{ (2)}$   
 $\Rightarrow ! \langle E \rangle \langle E_1 \rangle \text{ (3)}$   
 $\Rightarrow - \langle E \rangle \langle E_1 \rangle \text{ (4)}$   
 $\Rightarrow + \langle E \rangle \langle E_1 \rangle \text{ (5)}$   
 $\Rightarrow ( \langle E \rangle ) \langle E_1 \rangle \text{ (6)}$   
 $\Rightarrow \langle \text{IL} \rangle \langle E_1 \rangle \text{ (8)}$   
 $\Rightarrow \langle \text{CL} \rangle \langle E_1 \rangle \text{ (9)}$   
 $\Rightarrow \langle \text{FL} \rangle \langle E_1 \rangle \text{ (10)}$   
 $\Rightarrow \langle \text{SL} \rangle \langle E_1 \rangle \text{ (11)}$

$E_1 \Rightarrow \parallel \langle E \rangle \text{ (12)}$   
 $\Rightarrow == \langle E \rangle \text{ (13)}$   
 $\Rightarrow != \langle E \rangle \text{ (14)}$   
 $\Rightarrow \leq \langle E \rangle \text{ (15)}$   
 $\Rightarrow < \langle E \rangle \text{ (16)}$   
 $\Rightarrow \geq \langle E \rangle \text{ (17)}$   
 $\Rightarrow > \langle E \rangle \text{ (18)}$   
 $\Rightarrow \&\& \langle E \rangle \text{ (19)}$   
 $\Rightarrow + \langle E \rangle \text{ (20)}$   
 $\Rightarrow - \langle E \rangle \text{ (21)}$   
 $\Rightarrow * \langle E \rangle \text{ (22)}$   
 $\Rightarrow / \langle E \rangle \text{ (23)}$   
 $\Rightarrow \% \langle E \rangle \text{ (24)}$   
 $\Rightarrow ; \langle E \rangle \text{ (25)}$

$IF\_ST \Rightarrow \text{if } ( \langle E \rangle ) \langle \text{STMT} \rangle \text{ else } \langle \text{STMT} \rangle \text{ (26)}$

$B \Rightarrow \{ \langle \text{STMT\_LIST} \rangle \} \text{ (27)}$

$VAR \Rightarrow \langle \text{DT} \rangle \langle \text{id} \rangle ; \text{ (28)}$

$STMT \Rightarrow \langle E \rangle \text{ (29)}$   
 $\Rightarrow \langle \text{IF\_ST} \rangle \text{ (30)}$   
 $\Rightarrow \langle B \rangle \text{ (31)}$   
 $\Rightarrow \langle \text{VAR} \rangle \text{ (32)}$

$STMT\_LIST \Rightarrow \langle \text{STMT} \rangle \langle \text{STMT\_LIST} \rangle \text{ (33)}$   
 $\Rightarrow \epsilon \text{ (34)}$

### Descrição dos não terminais

Não terminais	Descrição
P	Programa principal
E	Expressão da linguagem
E.1	Auxiliar das Expressões
IF_ST	Declaração de comando if
B	Declaração de bloco de código
VAR	Declaração de variável
STMT	Declaração
STMT_LIST	Lista de declarações

### Conjunto First

Não terminal	First
P	{ <i>void</i> }
E	{ <i>id</i> ,!, -, +, (, <i>IL</i> , <i>CL</i> , <i>FL</i> , <i>SL</i> }
E.1	{  , ==, !=, <=, <, >=, >, &&, +, -, *, /, %, ;}
IF_ST	{ <i>if</i> }
B	{{}}
VAR	{ <i>DT</i> }
STMT	{ <i>id</i> ,!, -, +, (, <i>IL</i> , <i>CL</i> , <i>FL</i> , <i>SL</i> , <i>if</i> , {, <i>DT</i> }
STMT_LIST	{ <i>id</i> ,!, -, +, (, <i>IL</i> , <i>CL</i> , <i>FL</i> , <i>SL</i> , <i>if</i> , {, <i>DT</i> }

### Conjunto Follow

Não terminal	Follow
P	{ <i>\$</i> }
E	{  , ==, !=, <=, <, >=, >, &&, +, -, *, /, %, )}
E.1	{  , ==, !=, <=, <, >=, >, &&, +, -, *, /, %, )}
IF_ST	{}
B	{}
VAR	{}
STMT	{}
STMT_LIST	{})}

Tabela prediviva

	void	DT	(	)	{	}	if	!	else	id	IL	CL	FL	SL
P	1													
E			6					3		2	8	9	10	11
$E_1$														
IF_ST							26							
B					27									
VAR		28												
STMT		32	29		31		30	29		29	29	29	29	29
STMT_LIST		33	33		33	34	33	33		33	33	33	33	33

	!=	;	≤	<	≥	>	+	−	*	/	&&	==		%
P														
E							5	4						
$E_1$	14	25	15	16	17	18	20	21	22	23	19	13	12	24
IF_ST														
B														
VAR														
STMT							29	29						
STMT_LIST							33	33						



## 4 Diagramas de transição

Abaixo está os diagramas para as gramáticas.



Figura 1: Programa principal

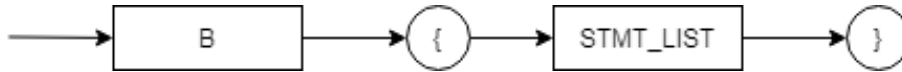


Figura 2: Bloco de comandos

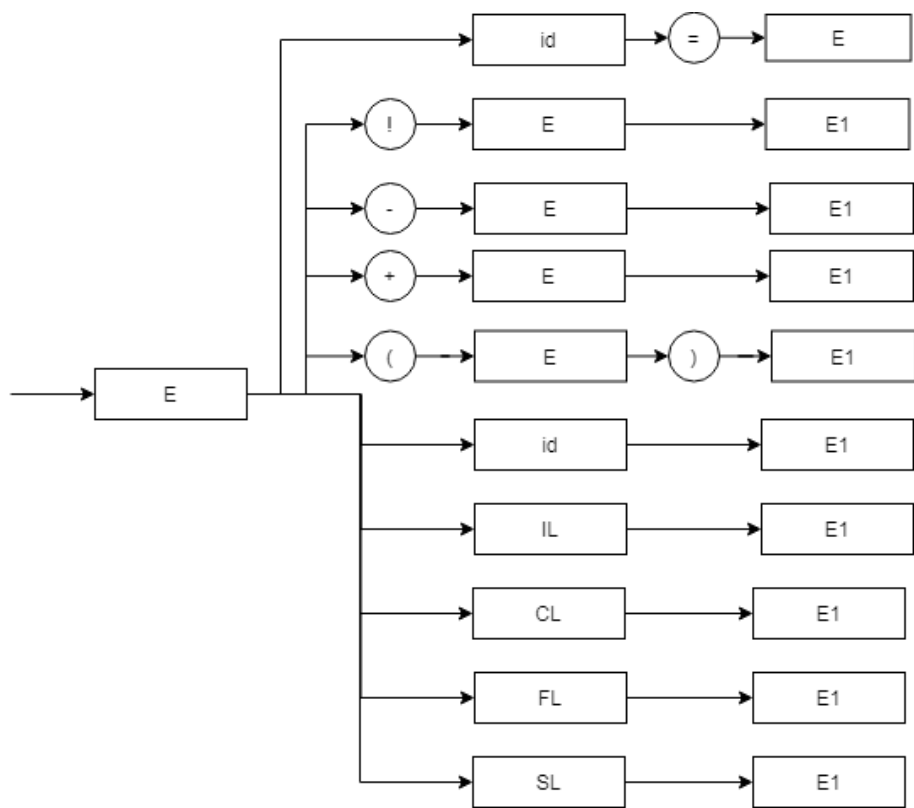


Figura 3: Expressão

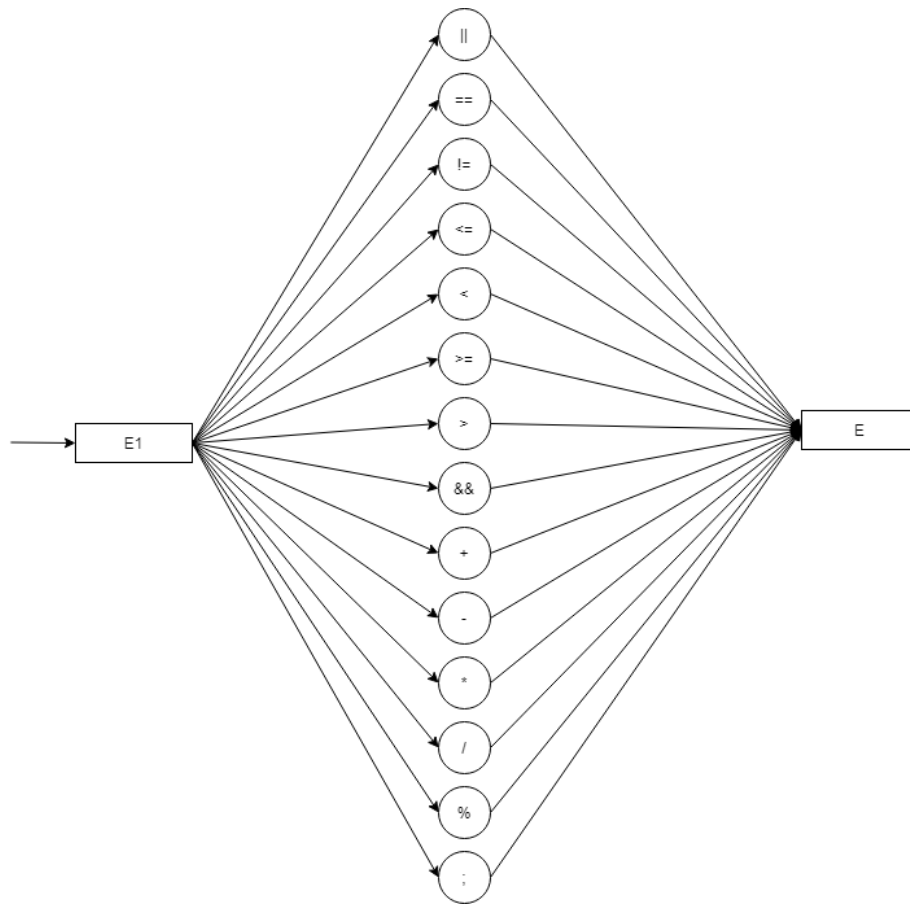


Figura 4: Auxiliar da Expressão

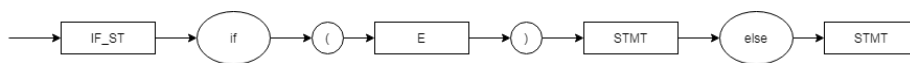


Figura 5: Declaração de condicional

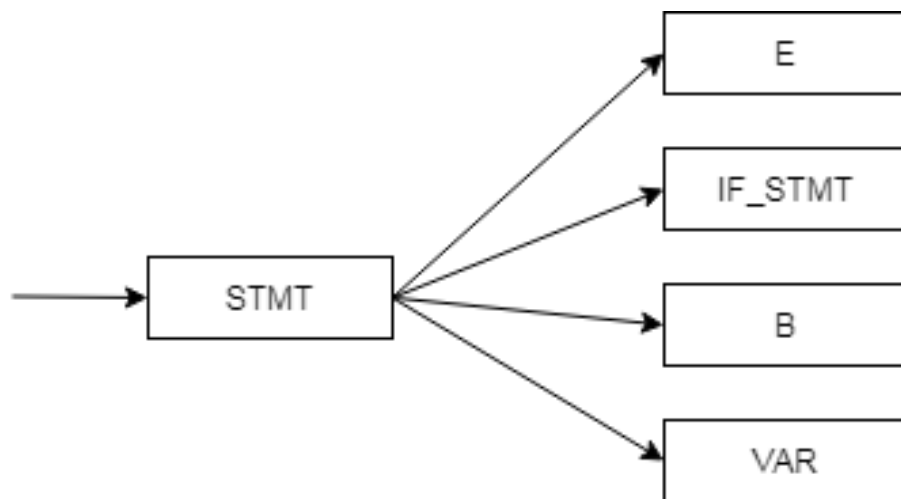


Figura 6: Declaração

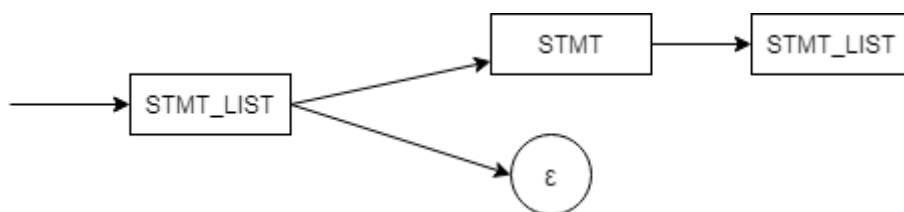


Figura 7: List de declaração



Figura 8: Declaração de variável