



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:

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

Número:

```
digit = [0-9]
```

Identificador:

```
id = <word>(<word> + <digit>)*
```

Tipo de dados:

```
data_type = int | float | char | void
```

Char literal:

```
char_literal = '<word>'
```

String literal:

```
string_literal = "(<word> + <digit>)*"
```

Inteiro literal:

```
integer_literal => (<digit>)+
```

Flutuante literal:

```
float_literal => (<digit>)+.(<digit>)+
```

Operadores booleana:

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

Operadores aritméticos:

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

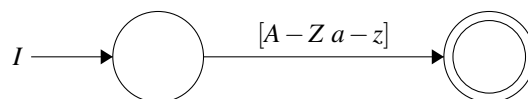
Delimitadores:

```
delimiter = , | ;
```

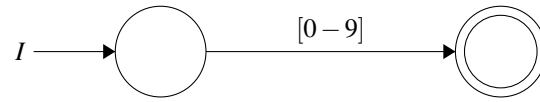
## 2 Autômatos C

Autômatos das expressões acima:

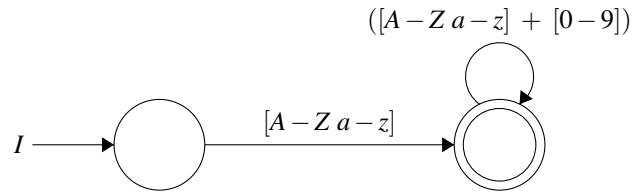
Letra:



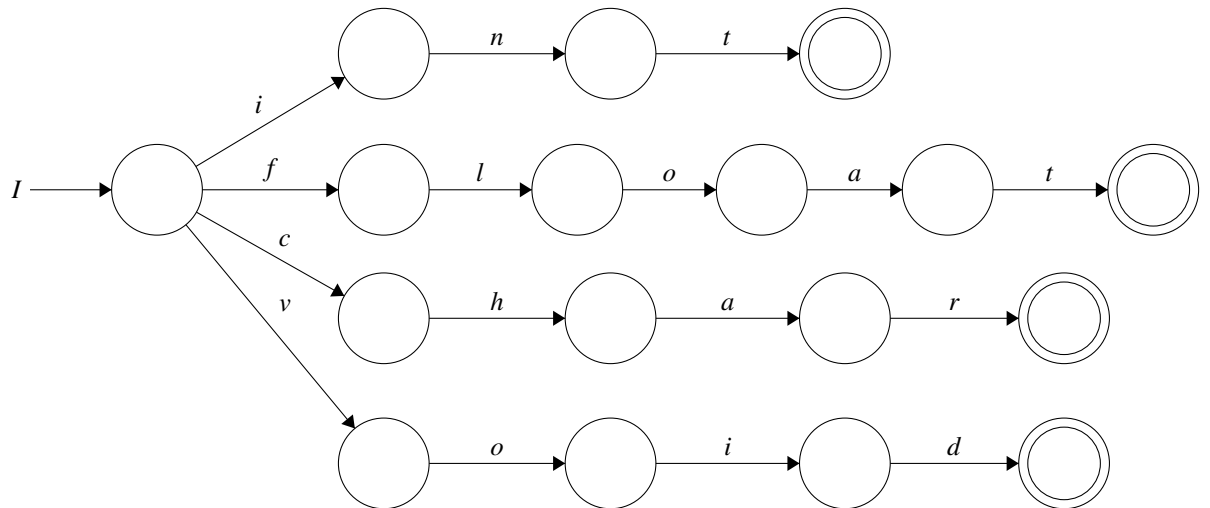
Número:



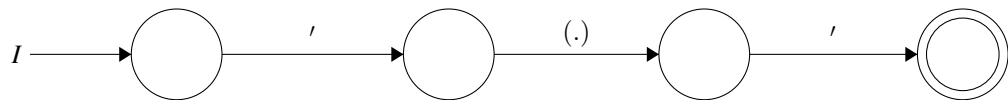
Identificador:



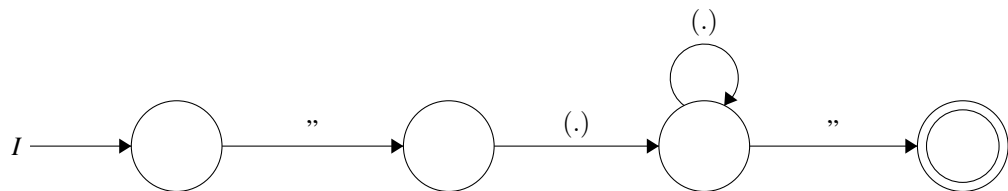
Tipo de dados:



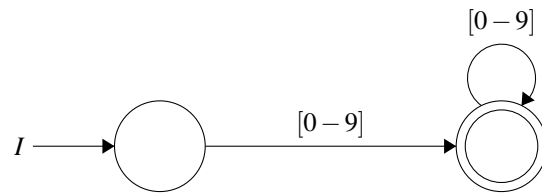
Caractere:



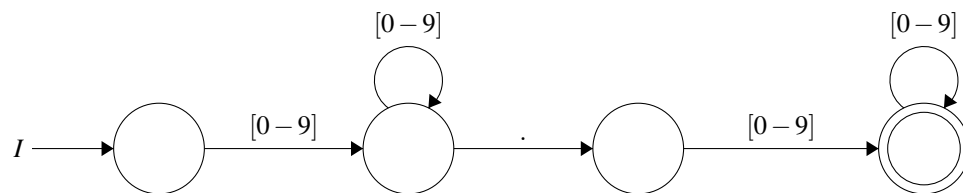
String:



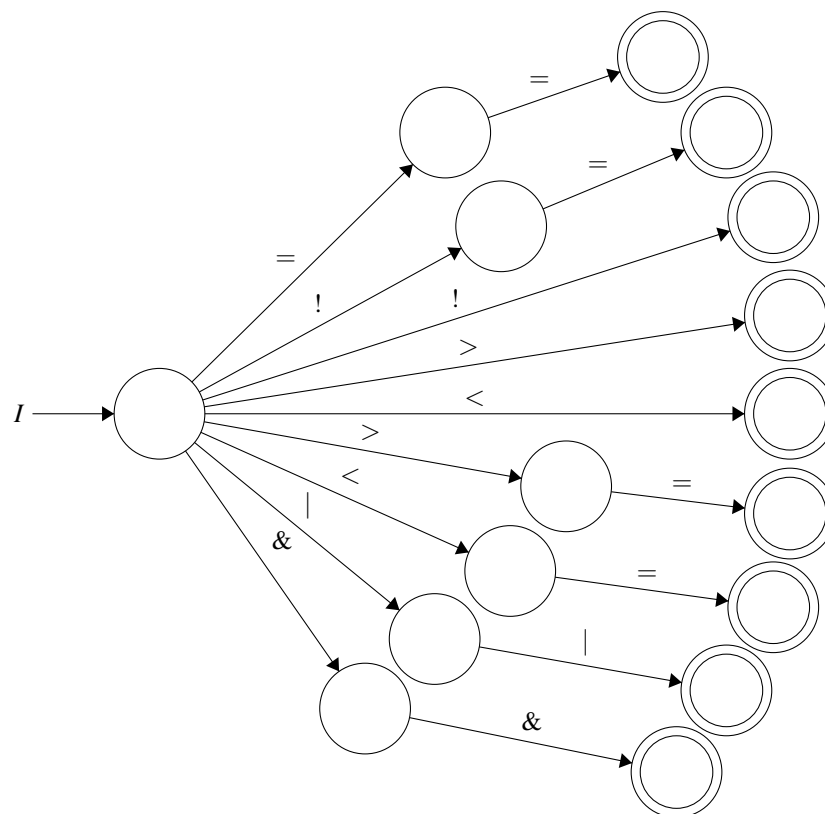
Inteiro literal:



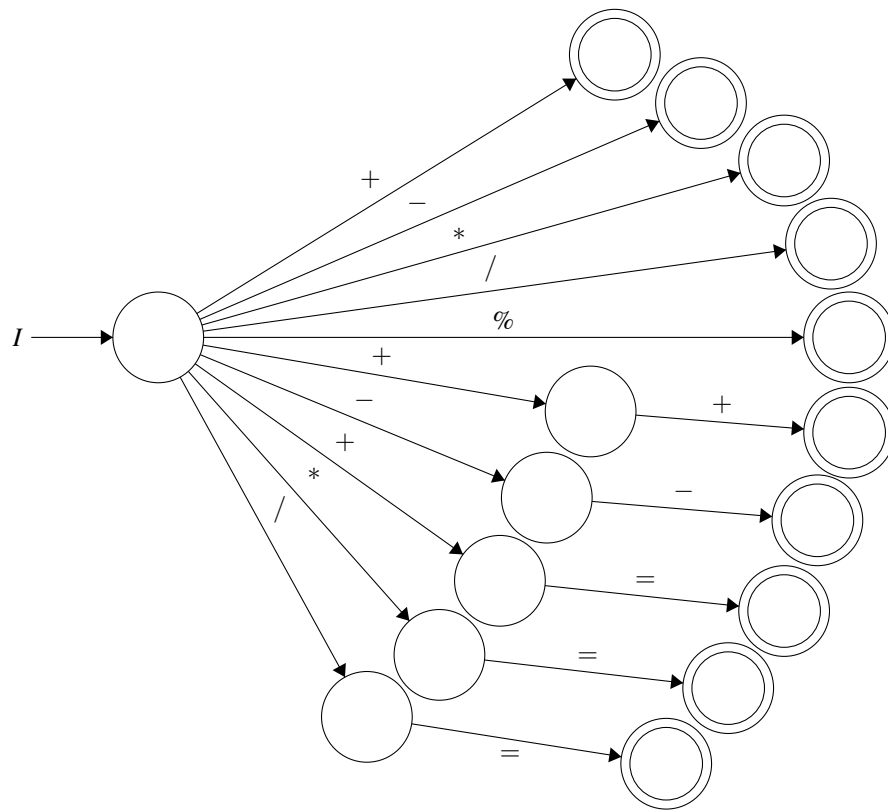
Ponto flutuante literal:



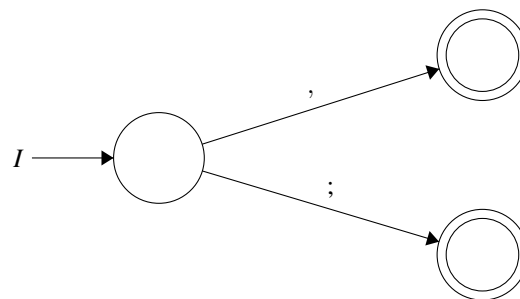
Operadores booleanos:



Operadores aritméticos:



Delimitadores:



### 3 Definições da gramática da linguagem C

Programa principal:

```
program => void main() <{> <decl_block> <}>
```

Expressão:

```
expr => <id> = <expr><expr'>
      => !<expr><expr'>
      => -<expr><expr'>
      => +<expr><expr'>
      => (<expr>)<expr'>
      => <id><expr'>
      => <integer_literal><expr'>
      => <char_literal><expr'>
      => <float_literal><expr'>
      => <string_literal><expr'>
expr' => || <expr><expr'>
       => == <expr><expr'>
       => != <expr><expr'>
       => <= <expr><expr'>
       => < <expr><expr'>
       => >= <expr><expr'>
       => > <expr><expr'>
       => && <expr><expr'>
       => + <expr><expr'>
       => - <expr><expr'>
       => * <expr><expr'>
       => / <expr><expr'>
       => % <expr><expr'>
```

Declaração de expressão:

```
expr_stmt => <expr>; | e
```

Declaração de condicional:

```
if_stmt => if (<expr>) <stmt> | if (<expr>) <stmt> else <
      stmt>
```

Declaração:

```
stmt => <expr_stmt> | <if_stmt> | <decl_block> | <var_decl>
```

Bloco de declarações:

```
<decl_block> => <{> <stmt_list> <}>
```

Declaração de variável:

```
var_decl => <data_type> <id>;
```

Lista de declarações:

```
stmt_list => <stmt_list'>
stmt_list' => <stmt><stmt_list'>
```

## 4 Conjunto First e Follow

Não terminais	First	Follow
program	{ <i>main</i> }	
expr	{ <i>id</i> ,!, -, +, (, <i>integer_literal</i> , <i>char_literal</i> , <i>float_literal</i> , <i>string_literal</i> }	
expr'	{ , ==, !=, <=, >, >=, &&, +, -, *, /, %}	
expr_stmt	{ <i>id</i> ,!, -, +, (, <i>integer_literal</i> , <i>char_literal</i> , <i>float_literal</i> , <i>string_literal</i> , $\epsilon$ }	
if_stmt	{ <i>if</i> }	
stmt	{ <i>id</i> ,!, -, +, (, <i>integer_literal</i> , <i>char_literal</i> , <i>float_literal</i> , <i>string_literal</i> , $\epsilon$ <i>if</i> , {, <i>data_type</i> }	
decl_block	{ { }	
var_decl	{ <i>data_type</i> }	
stmt_list	{ <i>id</i> ,!, -, +, (, <i>integer_literal</i> , <i>char_literal</i> , <i>float_literal</i> , <i>string_literal</i> , $\epsilon$ <i>if</i> , {, <i>data_type</i> }	
stmt_list'	{ <i>id</i> ,!, -, +, (, <i>integer_literal</i> , <i>char_literal</i> , <i>float_literal</i> , <i>string_literal</i> , $\epsilon$ <i>if</i> , {, <i>data_type</i> }	

## 5 Diagramas de transição

Abaixo está os diagramas para as gramáticas.

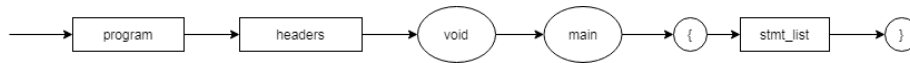


Figura 1: Programa principal

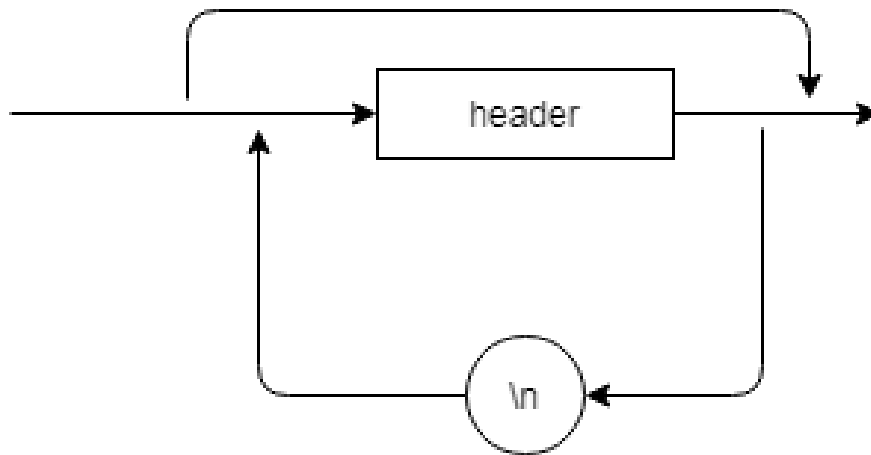


Figura 2: Cabeçalhos

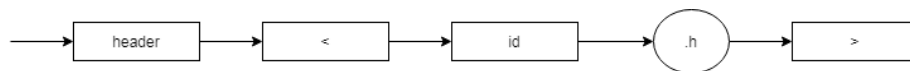


Figura 3: Cabeçalho





Figura 4: Declaração break

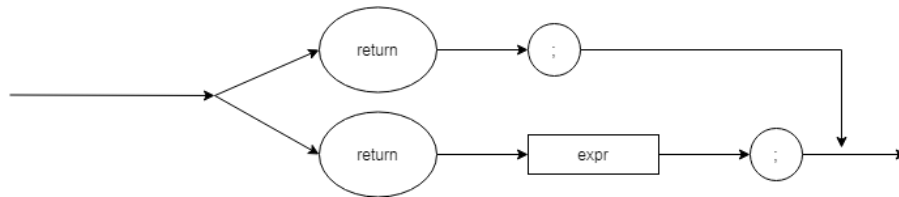


Figura 5: Declaração return

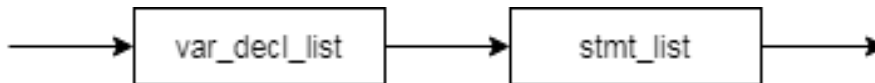


Figura 6: Declaração de bloco de comandos

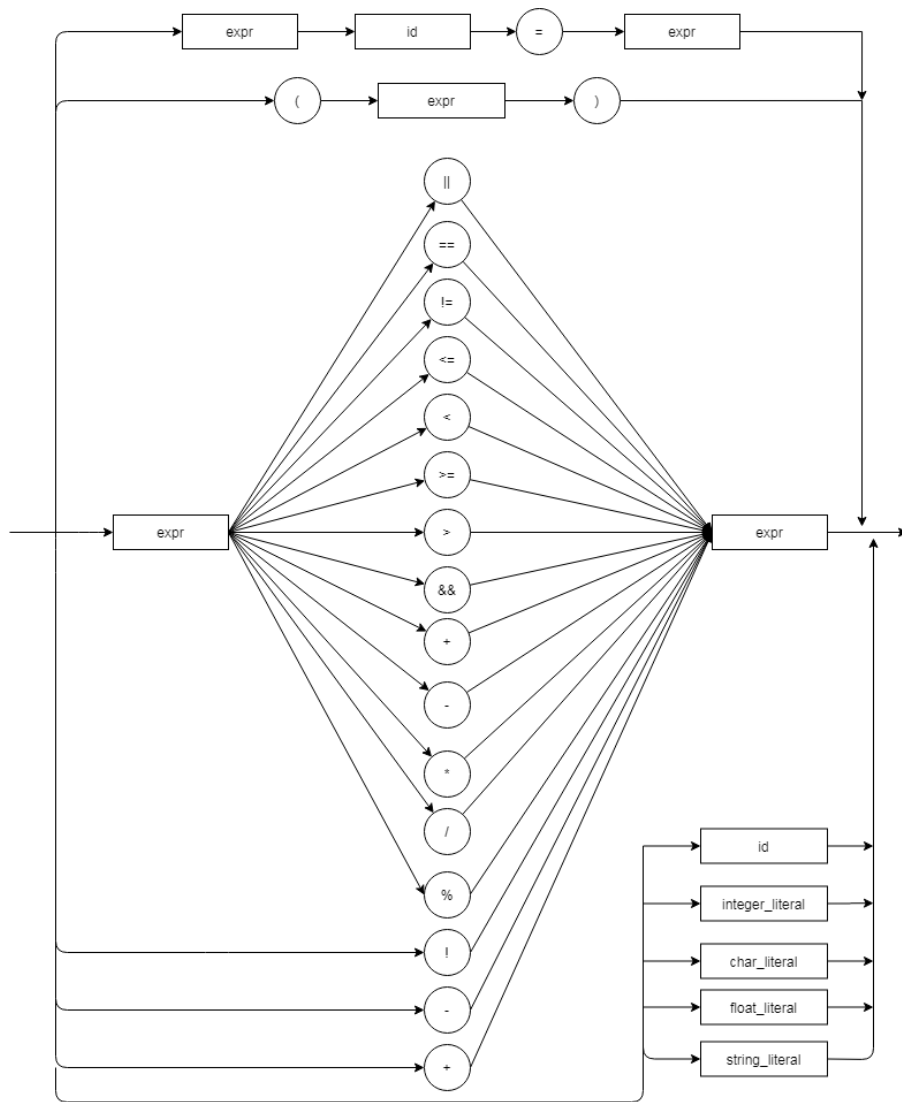


Figura 7: Expressão

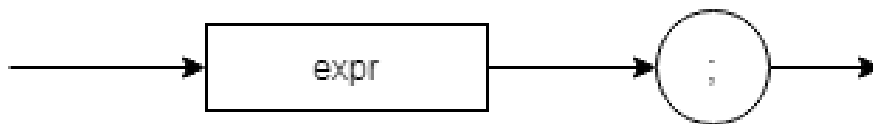


Figura 8: Declaração de expressão

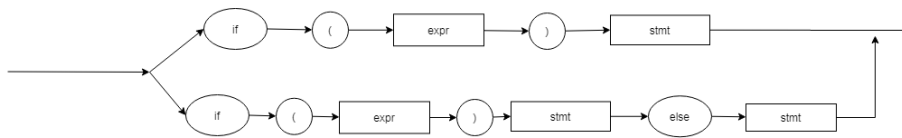


Figura 9: Declaração de condicional

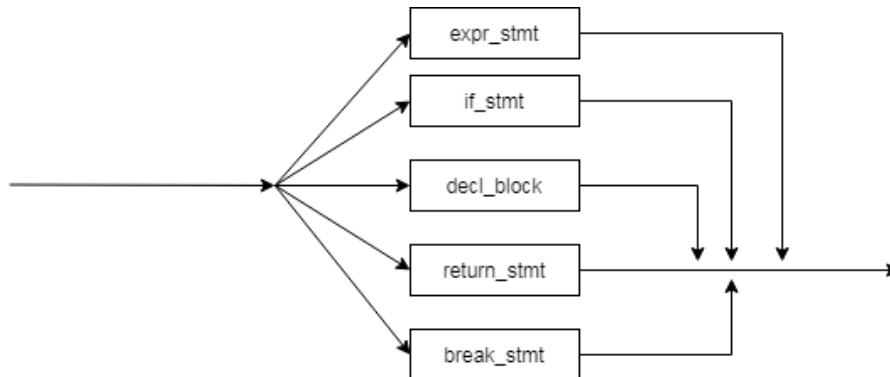


Figura 10: Declaração

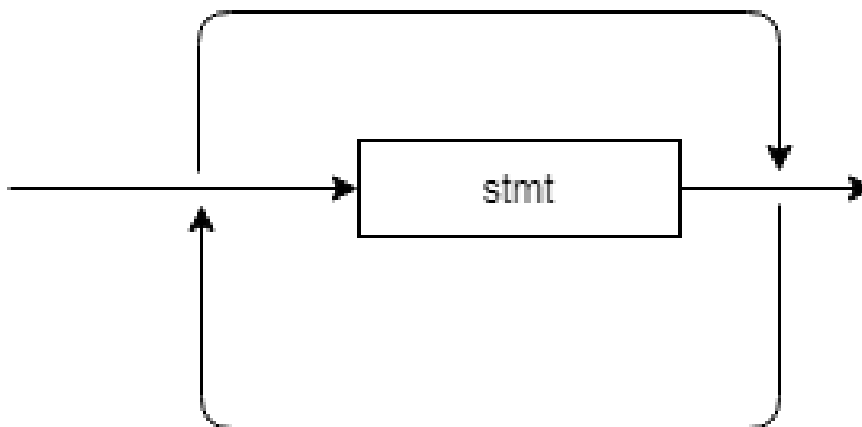


Figura 11: List de declaração

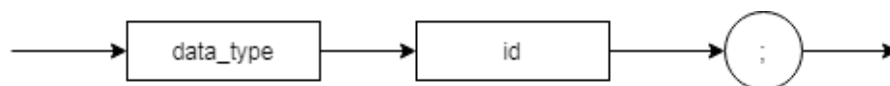


Figura 12: Declaração de variável

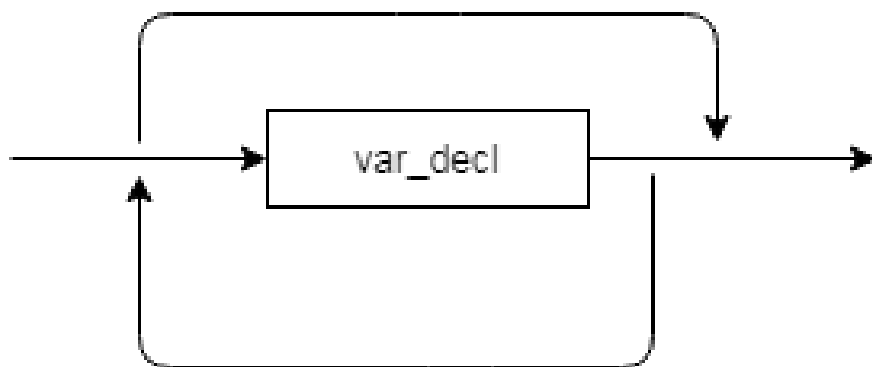


Figura 13: List de declaração de variáveis