



Curso de Ciência de Computação

COMPILADORES

Prof. Tarcísio Lucas

Compilador Arretado

Análise Léxica

Seu time terá a missão de desenvolver um Analisador Léxico para uma linguagem de programação inspirada em C em seus comandos e inspirada no vocabulário pernambucano em suas mensagens de erro. Segue a descrição dos tipos de tokens que devem ser considerados.

Inteiro	Sequência de dígitos entre 0 e 9. Desconsiderar uso de sinal por simplicidade Ex: 100, 123, 932000, 1.
Real	Sequência de dígitos, seguido de ponto, seguido de outra sequência de dígitos. Desconsiderar uso de sinal por simplicidade Ex: 100.25, 5.7, 132.85
Char	Aspa simples, seguida de letra ou número, seguida de aspa simples. Ex: 'a', '1'
Identificador	Letra, seguido de letras ou números. Pode ser apenas uma letra. Apenas letras minúsculas e sem acento. Ex: n, num, num1, n1
operador relacional	<, >, <=, >=, ==, <>
operador aritmético	+, -, *, /, %
operador atribuição	=
caracter especial), (, {, }, ;, ,
Palavras reservadas	int, float, char, while, main, if, else

Expressão regular de cada token:

Considere:

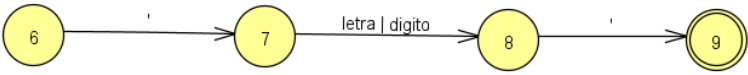
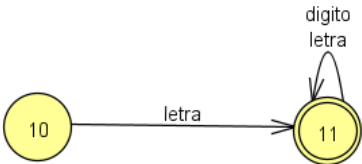
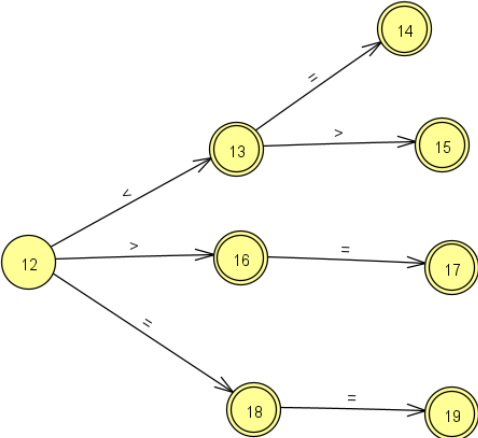
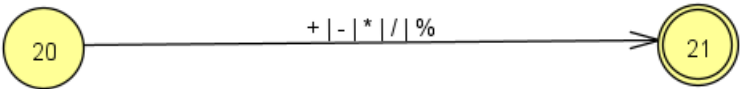
- dígito -> [0-9]
- letra -> [a-zA-Z]

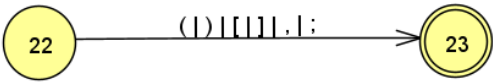
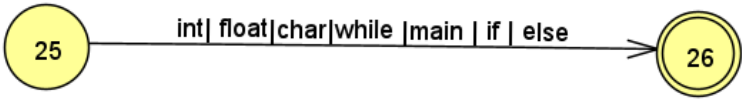

Descreva as expressões regulares para os tokens descritos na tabela anterior.

Tipo Token	Expressão regular
Inteiro	digito+
Real	(digito+ . digito+)
Char	'(letra digito)'
Identificador	letra(letra digito)*
operador relacional	[<>]=?
operador aritmético	[+.*/%]
operador atribuição	[=]
caracter especial	[(){};,]
palavras reservadas	(int float char while main if else)

Máquina de estado correspondente a cada expressão regular:

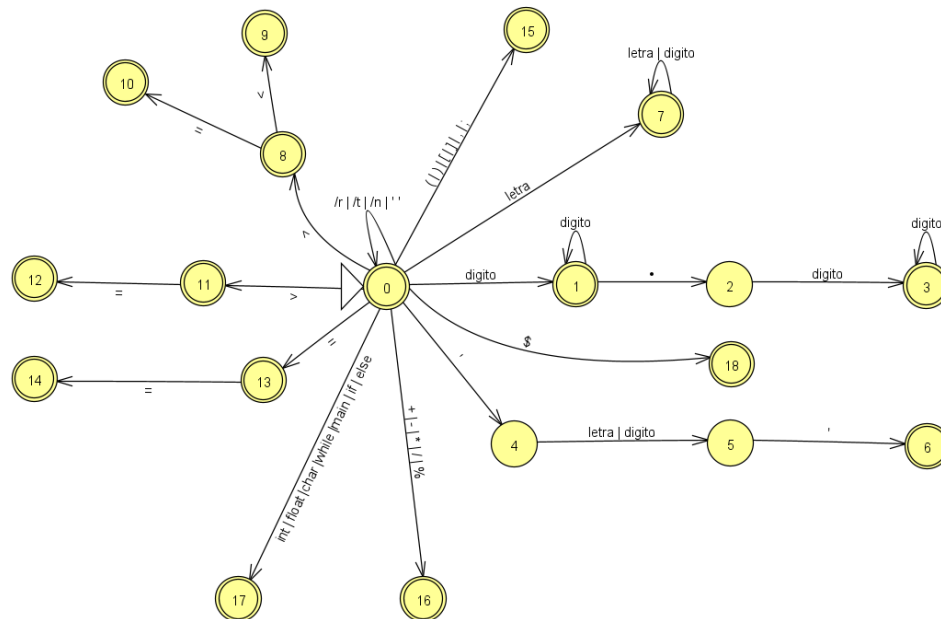
Expressão regular token	Máquina de estado
Inteiro	<pre> graph LR Start(()) --> 0((0)) 0 -- digito --> 1(((1))) 1 -- digito --> 1 </pre>
Real	<pre> graph LR 2((2)) -- digito --> 3(((3))) 3 -- digito --> 3 3 -- "." --> 4((4)) 4 -- digito --> 5(((5))) 5 -- digito --> 5 </pre>

Char	 <p>A finite state machine with four states: 6, 7, 8, and 9. State 6 is the start state, and state 9 is the final state. Transitions are: 6 to 7 on input "'", 7 to 8 on input "letra digito", and 8 to 9 on input "'".</p>
Identificador	 <p>A finite state machine with two states: 10 and 11. State 10 is the start state, and state 11 is the final state. Transitions are: 10 to 11 on input "letra", and a self-loop on state 11 for inputs "digito" and "letra".</p>
operador relacional	 <p>A finite state machine with eight states: 12, 13, 14, 15, 16, 17, 18, and 19. State 12 is the start state, and states 14, 15, 17, and 19 are final states. Transitions are: 12 to 13 on "<", 12 to 16 on ">", 12 to 18 on "=", 13 to 14 on "=", 13 to 15 on ">", 16 to 17 on "=", and 18 to 19 on "=".</p>
operador aritmético	 <p>A finite state machine with two states: 20 and 21. State 20 is the start state, and state 21 is the final state. The transition is from 20 to 21 on inputs "+", "-", "*", "/", and "%".</p>

operador atribuição	O Estado 18 do operador <u>relacional</u> já trata a atribuição
caracter especial	 <pre> graph LR 22((22)) -- "() [] { } , ;" --> 23(((23))) </pre>
palavras reservadas	 <pre> graph LR 25((25)) -- "int float char while main if else" --> 26(((26))) </pre>
espaço em branco	 <pre> graph LR 24(((24))) -- "\n \t /n \\' " --> 24 </pre>

Máquina de estado única, unindo todas as máquinas de estado criadas anteriormente e adicionando as modificações necessárias. Onde cada estado deve ser identificado por um número inteiro distinto.

Máquina de estado Geral



Bom trabalho!