

Unisul	Universidade do Sul de Santa Catarina.
Curso	Ciência da Computação
Disciplina	Linguagens Formais e Autômatos
Capítulo	Trabalho Final

## 1. Introdução

- Entrega: **Dia 10.06.2020 – Avaliação - EVA**
- Programação: Java Fonte
- Programa Gerado: 2020a\_<Iniciais dos Integrantes da Equipe>Lexico.jar
- Exportar pelo Eclipse
- Nome Projeto: 2020a\_<Iniciais dos Integrantes da Equipe>Léxico
- Diagrama de Classe do Programa (Enviar PDF de forma legível)
- Autômato Finito (Exemplo abaixo)
- 3 programas de teste com resultado do analisador léxico (tabela de símbolos)
- Documentação padrão Unisul

## 2. Especificação da Linguagem LMS

### 2.1 Elementos Léxicos

O desenvolvimento do compilador foi feito através destes elementos léxicos.

Descrição da Linguagem

**a) Símbolos especiais:** virgula (,); pontovirgula (;); ponto (.); ponto´ponto (..); doispontos(:); abrepar(()); fechapar()); abrecolch ([); fechacolch(]);

#### b) Palavras reservadas

AND – ARRAY – BEGIN – CALL – CASE – CONST – DO – ELSE – END  
– FOR – GOTO – IF – INTEGER – LABEL - NOT - OF - OR -  
PROCEDURE - PROGRAM – READLN - REPEAT - THEN - TO -  
UNTIL - VAR - WHILE - WRITELN

#### c) Identificadores

É definido por um conjunto de caracteres, alfanuméricos (**máx. 30**), sendo o primeiro caractere um alfabético, seguido de um conjunto de **dígitos**(0..9) e/ou **letras**(a..z, A..Z).

#### d) Números inteiros

Conjuntos de números definidos na faixa de **-32.767 a 32.767**. Somente inteiros são aceitos na linguagem LMS.

Ex: 19672 ok  
**32800 ILEGAL**, valor fora da escala  
-1 ok

Unisul	Universidade do Sul de Santa Catarina.
Curso	Ciência da Computação
Disciplina	Linguagens Formais e Autômatos
Capítulo	Trabalho Final

**19.67 ILEGAL**, não aceita ponto decimal

#### e) Literais

Sequência de caracteres (letras/símbolos/números) delimitados por aspas, contendo não mais do que **255 caracteres**.

Ex.: “ Compilador TESTE para a LMS 1234<sup>a</sup> teste“

#### f) Operadores

Os operadores sejam eles aritméticos ou lógicos, dividem-se em categorias:

- 1- menos unário (-)
- 2- operador de negação: NOT
- 3- operadores de multiplicação/divisão : \* /
- 4- operadores lógicos: AND OR
- 5- operador de adição/subtração: + -
- 6- operadores relacionais : < > = <= >= <>

### OBSERVAÇÃO:

**DELIMITADORES** – Os **caracteres branco**, final de arquivo ou um comentário podem ser usados como separadores de tokens.

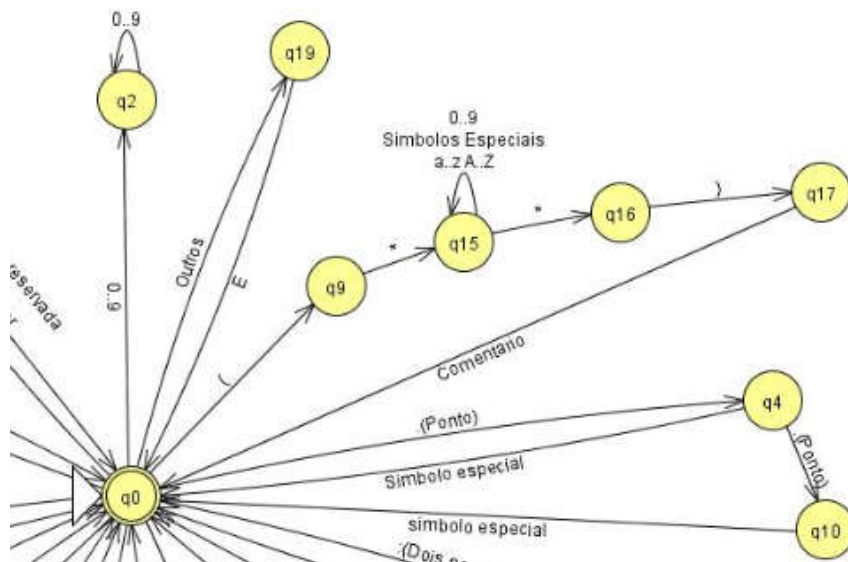
**COMENTÁRIOS** – Um comentário pode ser inserido em qualquer lugar do programa onde um delimitador é válido. **É delimitado por (\* e \*)**. Exemplo: (\* compilador LMS \*)

#### 2.1.1 Tabela de código (para implementação) para os símbolos terminais (tokens)

Unisul	Universidade do Sul de Santa Catarina.
Curso	Ciência da Computação
Disciplina	Linguagens Formais e Autômatos
Capítulo	Trabalho Final

Código	Símbolo	Código	Símbolo	Código	Símbolo	Código	Símbolo
1	Program	14	Then	27	For	40	=
2	Label	15	Else	28	To	41	>
3	Const	16	While	29	Case	42	>=
4	Var	17	Do	30	+	43	<
5	Procedure	18	Repeat	31	-	44	<=
6	Begin	19	Until	32	*	45	< >
7	End	20	Readln	33	/	46	,
8	Integer	21	Writeln	34	[	47	;
9	Array	22	Or	35	]	48	Literal (cadeia de caracteres entre aspas)
10	Of	23	And	36	(	49	.
11	Call	24	Not	37	)	50	..
12	Goto	25	Identificador	38	:=	51	\$ (delimitador – final de arquivo)
13	If	26	Inteiro	39	:		

### Modelo Autômato Finito



### Exemplo Geração Programa

Unisul	Universidade do Sul de Santa Catarina.
Curso	Ciência da Computação
Disciplina	Linguagens Formais e Autômatos
Capítulo	Trabalho Final

The screenshot shows the PL/SQL Editor with a program on the left and its tokenization on the right.

**Program teste3:**

```

Const c:=2;
Var a:integer;

Procedure p;
Var x:integer;

  Procedure q;
  Var t:integer;
  Begin (" inicio da q ")
    x:= x - 80 ; t:=x*c;
    if t > 80 then call q else writeln(t)
    end; (" fim de q ")

  begin (" inicio da P ")
    x:= a+b*t;
    if x> 100 then call q else writeln(x);
  end; (" fim da p ")

begin (" programa principal ")

readln(a,b)
if a>1000 then a:= 900
  else a:= b+10;
while a>b do

```

**Confirmar**

Codigo	Token	Descricao
22	PROGRAM	Palavra reservada
19	teste3	Identificador
14	;	Ponto e virgula
23	CONST	Palavra reservada
19	c	Identificador
6	=	Operador relacional
20	2	Numero inteiro
14	;	Ponto e virgula
24	VAR	Palavra reservada
19	a	Identificador
15	,	Virgula
19	y	Identificador
13	:	Dois Pontos
28	INTEGER	Palavra reservada
14	;	Ponto e virgula
25	PROCEDURE	Palavra reservada
19	p	Identificador
14	;	Ponto e virgula