



**UNIVERSIDADE FEDERAL DE ALAGOAS – UFAL
CAMPUS A. C. SIMÕES**

CIÊNCIA DA COMPUTAÇÃO – COMP263

MAXWELL ESDRA ACIOLI SILVA

ANALISADOR LÉXICO LINGUAGEM 2M

**Trabalho desenvolvido sob a
orientação do professor Alcino
Dall'Igna, como requisito parcial
para obtenção de nota na
disciplina Compiladores do curso
de Ciência da Computação da
Universidade Federal de Alagoas.**

**MACEIÓ - AL
MARÇO DE 2016**

ESPECIFICAÇÃO DOS TOKENS DA LINGUAGEM DE PROGRAMAÇÃO 2M

1. ESPECIFICAÇÕES DA LINGUAGEM DE IMPLEMENTAÇÃO

A linguagem de programação que será utilizada para implementação do analisador léxico e sintático da Linguagem 2M será JAVA. A escolha dessa linguagem foi feita baseada nos recursos que a mesma oferece para este fim.

2. ENUMERAÇÃO COM AS CATEGORIAS DOS TOKENS

```
public enum TokenCategory{

    major, id, tEmpty, tInt, tLong, tLogic, tChar, tCchar, tDec, escBegin, escEnd,
    paramBegin, paramEnd, arrayBegin, arrayEnd, comment, term, sep1, sep2, constNumInt,
    constNumDec, constLogic, constChar, constCchar, prReadin, prPrintout, prIf, prElse,
    prElseif, prIterator, prDo, prWhile, prReturn, opAtrib, opLogicAnd, opLogicOr,
    opNegLogic, opAritAdit, opAritMult, opAritExp, opNegUn, opRel1, opRel2, opConc;

}
```

3. ESPECIFICAÇÃO DOS TOKENS DA LINGUAGEM

Expressões Regulares Auxiliares:

```
letter = 'a' | 'b' | 'c' | 'd' | 'e' | 'f' | 'g' | 'h' |
'i' | 'j' | 'k' | 'l' | 'm' | 'n' | 'o' | 'p' | 'q' | 'r'
| 's' | 't' | 'u' | 'v' | 'w' | 'x' | 'y' | 'z' | 'A' |
'B' | 'C' | 'D' | 'E' | 'F' | 'G' | 'H' | 'I' | 'J' | 'K'
| 'L' | 'M' | 'N' | 'O' | 'P' | 'Q' | 'R' | 'S' | 'T' |
'U' | 'V' | 'W' | 'X' | 'Y' | 'Z';
```

```
digit = '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' |
'8' | '9';
```

```
symbol = ' ' | '.' | ',' | ':' | ';' | '?' | '!' | '+' |
'-' | '*' | '\' | '/' | '_' | '%' | '@' | '&' | '#' | '$'
| '<' | '>' | '=' | '(' | ')' | '[' | ']' | '{' | '}' |
'|' | ''' | '"';
```

Lexemas:

Major:

```
major = 'major';
```

Identificador:

```
id = ('letter' | '_') ('letter' | '_' | 'digit')*;
```

Tipos primitivos:

```
tEmpty = 'empty';  
tInt = 'int';  
tLong = 'long';  
tLogic = 'logic';  
tChar = 'char';  
tCchar = 'cchar';  
tDec = 'dec';
```

Delimitadores:

Escopo:

```
escBegin = '[';  
escEnd = ']';
```

Parâmetros:

```
paramBegin = '(';  
paramEnd = ')';
```

Array:

```
arrayBegin = '{';  
arrayEnd = '}';
```

Comentários:

```
comment = '/$';
```

Terminador:

```
term = '#';
```

Separadores:

```
sep1 = ',';
```

```
sep2 = ';;
```

Constantes de tipos:

```
constNumInt = ('digit')('digit')*;
constNumDec =
('digit')('digit')*('.')('digit')('digit')*;
constLogic = ('truth' | 'false');
constChar = ('')('letter' | 'digit' | 'symbol')('');
constCchar = ('')('letter' | 'digit' | 'symbol')*('');
```

Palavras Reservadas de Comando de Entrada ou Saída:

```
prReadin = 'readin';
prPrintout = 'printout';
```

Palavras Reservadas de Comando de Iteração ou Seleção:

```
prIf = 'if';
prElse = 'else';
prElseif = 'elseif';
prIterator = 'iterator';
prDo = 'do';
prWhile = 'while';
```

Palavra Reservada de Retorno de Função:

```
prReturn = 'return';
```

Operador Atributivo:

```
opAtrib = '=';
```

Operadores Lógicos:

```
opLogicAnd = 'and';
opLogicOr = 'or';
opNegLogic = 'not';
```

Operadores Aritméticos:

```
opAritAdit = '+' | '-';
opAritMult = '*' | '/';
opAritExp = '^';
```

Operador Unário:

```
opNegUn = '-';
```

Operadores Relacionais:

```
opRel1 = '<' | '>' | '<=' | '>='  
opRel2 = '==' | '~=';
```

Operador Concatenação:

```
opConc = '++';
```