**Universidade Federal de Alagoas**

# Especificação dos Tokens

**Aluno**: David Silva Alexandre

**Professor**: Alcino Dall'lgna Júnior

**2021**

## Linguagem utilizada

A linguagem de programação que iremos usar para implementar os analisadores léxico e sintático da linguagem AVI será C++.

## Especificação dos Tokens

Como solicitado, especificaremos os tokens em dois grupos distintos, confira:

- **Enumeração de Tokens**

```
enum TokenType
{
    comma,
    semiColon,
    headerCons,
    headerName,
    reMain,
    intType,
    intConst,
    floatType,
    floatConst,
    boolType,
    boolConst,
    stringType,
    stringCons,
    addOp,
    subOp,
    mulOp,
    divOp,
    andOpLog,
    andOpBin,
    orOpLog,
    orOpBin,
    xorOpBin,
    negOp,
    negUn,
    eqRel,
    greEqRel,
    lowEqRel,
    lowRel,
```

notEqRel,
concaten,
increOp,
decreOp,
openBrack,
closeBrack,
openPar,
closePar,
openBrace,
closeBrace,
endLine,
reIf,
reElse,
reFor,
reWhile,
ternOp1,
ternOp2,
reElseIf,
id,
greRel,
reVoid,
atrib,
unknown
};

- **Expressões Regulares dos Lexemas**

comma = ‘,’;
semiColon = ‘;’;,
reMain = ‘main’;
intType = ‘int’;
intConst = ‘[:digit:] +’;
floatType = ‘float’;
floatConst ‘([:digit:]) + \. ([:digit:]) +’;
boolType = ‘bool’;
boolConst = ‘true | false’;
stringType = ‘string’;
stringCons = ‘\”([:ascii:])*\”’;
addOp = ‘+’;
subOp = ‘-’;
mulOp = ‘*’;
divOp = ‘/’;

andOpLog = ‘&&’;
andOpBin = ‘&’;
orOpLog = ‘||’;
orOpBin = ‘|’;
xorOpBin = ‘^’;
negOp = ‘!’;
eqRel = ‘=’;
greRel = ‘>’;
lowRel = ‘<’;
greEqRel = ‘>=’;
lowEqRel = ‘<=’;
notEqRel = ‘!=’;
concaten = ‘+=’;
increOp = ‘++’;
decreOp = ‘--’;
openPar = ‘\(’;
closePar = ‘\)’;
openBrace = ‘\[’;
closeBrace = ‘\]’;
openBrack = ‘\{’;
closeBrack = ‘\}’;
endLine = ‘;’;
reIf = ‘if’;
reElse = ‘else’;
reFor = ‘for’;
reWhile = ‘while’;
ternOp1 = ‘?’;
ternOp2 = ‘:’;
reElseIf = ‘elseIf’;
id = ‘[:alpha:]([:alnum:] | ‘_’)*’;