UNIVERSIDADE FEDERAL DE ALAGOAS
INSTITUTO DE COMPUTAÇÃO - IC
CIÊNCIA DA COMPUTAÇÃO

JOÃO VICTOR DE ALARCÃO AYALLA ALCÂNTARA
ASCANIO SAVIO DE ARAUJO NEVES
JACKSON BARBOSA DA SILVA

COMPILADORES

# ESPECIFICAÇÃO DA LINGUAGEM AJA++

# Sumário

## 1 - ERs auxiliares

[:upper:] = Letras maiúsculas do alfabeto
[:lower:] = Letras minúsculas do alfabeto
[:digit:] = Algarismos de base decimal
[:alun:] = Caracteres que são Letras maiúsculas ou minúsculas do alfabeto ou algarismos de base decimal
Letter = ‘[:upper:]’ | ’[:lower:]’
Digits = ’[:digit:]+’
Symbol = ‘ ’ | ‘;’ | ‘,’ | ‘.’ | ‘:’ | ‘?’ | ‘!’ | ’+’ | ‘-’ | ‘*’ | ‘\\’ | ‘/’ | ‘_’ | ‘%’ | ‘&’ | ‘\#’ | ‘@’ | ‘$’ | ‘<’ | ‘>’ | ‘=’ | ‘(‘ | ‘)’ | ‘|’ | ‘[‘ | ‘]’ | ‘(‘ | ‘)’ | ‘{‘ | ‘}’ | ‘\’’ | ‘\”’ | ‘^’ | ‘\\n’

## 2 - Terminais

Id = (‘{Letter}’) ((‘{Letter}’ | ‘[:digit:]’)*)
Attribution = ‘=’
ArithmeticOperator = ‘+’ | ‘-’ | ‘/’ | ‘*’ | ‘++’ | ‘--’ | ‘+=’
BitwiseOperator = ‘^’ | ‘|’ | ‘&’
Relation = ‘>’ | ‘<’ | ‘>=’ | ‘<=’ | ‘==’ | ‘!=’
Integer = ((‘+’ | ‘-’)?) ‘{Digits}’
Double = ((‘+’ | ‘-’)?) (‘{Digits}’) (’.’) (‘{Digits}’)
Boolean = ‘true‘ | ‘false’
Character = (‘\’’) (‘[:alnum:]’ | ‘{Symbol}’) (‘\’’)
String = (‘\"’) (‘{Character}+’) (‘\"’)
ListType = ‘list’
Type = ‘void’ | ‘itg’ | ‘dbl’ | ‘chr’ | ‘bool’ | ‘string’

## 3 - Gramática

S = MainFunction | Function | Declaration
MainFunction = (‘function’) (‘void’) (‘main’) (‘{‘) (‘}’) (‘[‘) (‘{CommandsBlock}’) (‘]‘)
Function = (‘function’) (‘void’) (‘{Id}’) (‘{‘) (‘{ParametersList}’) (‘}’) (‘[‘) (‘{CommandsBlock}’) (‘]‘)
ParametersList = ((‘{VariableParameter}’ | ‘{ListDeclaration}’)*)
VariableParameter = (‘{Type}’) (‘{Id}’)
CommandsBlock = ‘{Command}*’

```
Command = FunctionCall | AttributionExpression | Declaration | Condition | Loop |
Output | Input | AppendList | Return
FunctionCall = (‘{Id}’) (‘{‘) (‘{IdList}) (‘}‘)
IdList = (‘{Id’}*)
AttributionExpression = (‘id’) (‘=’) (‘{Expression}’)
Expression =  (‘{ArithmeticExpression}’) | (‘{Boolean}’) |  (‘{Character}’) |
(‘{String}’)
ArithmeticExpression = (‘{ArithmeticTerm}’) ((‘{Operator}’
‘{ArithmeticExpression}’)?)
ArithmeticTerm = (‘{ArithmeticValue}’) ((‘{Operator}’ ‘{ArithmeticValue}’)?)
ArithmeticValue = ‘{Id}’ | ‘{ArithmeticConstant}’
ArithmeticConstant = ‘{Integer}’ | ‘{Double}’
Operator = ‘{ArithmeticOperator}’ | ‘{BitwiseOperator}’
Declaration = ‘{VariableDeclaration}’ |  ‘{ListDeclaration}’
ListDeclaration = (‘{ListType}’) (‘(‘) (‘{Type}’) (‘)‘) (‘{Id}’)
VariableDeclaration =  (‘{type}’) (‘{AttributionExpression}’ | ‘{id}’)
Condition = ‘{IfCommand}’ ‘{ElseCommand}’
ElseCommand = (‘else’)  (‘[‘) (‘{CommandsBlock}’) (‘]‘)
IfCommand = (‘if’ | ‘elseif’) (‘{‘) (‘{LogicalExpression}’) (‘}‘) (‘[‘)
(‘{CommandsBlock}’) (‘]‘)
LogicalExpression = (‘{LogicalTerm}’) ((‘{Relation}’ ‘{LogicalExpression}’)?)
LogicalTerm =  ‘{Value}’ ‘{Relation}’ ‘{Value}’
Value = ‘{Id}’ | ‘{Constant}’
Constant = ‘{Integer}’ | ‘{Double}’ | ‘{Boolean}’ | ‘{Character}’ | ‘{String}’
Loop = ‘{WhileStatement}’ | ‘{ForStatement}’
WhileStatement = (‘while’)  (‘{‘) (‘{LogicalExpression}’) (‘}‘) (‘[‘)
(‘{CommandsBlock}’) (‘]‘)
ForStatement = (‘for’)  (‘{‘) (‘{AttributionExpression}’) (‘;’) ('{Logical
Expression}') (‘;’) (‘{ArithmeticTerm}’) (‘}‘) (‘[‘) (‘{CommandsBlock}’) (‘]‘)
Output = (‘write’) (‘{‘) (‘{Value}‘) (‘}‘)
Input = (‘read’) (‘{‘) (‘{Value}‘) (‘}‘)
AppendList = (‘id’) (‘.’) (‘append’) (‘{‘) (‘{Value}‘) (‘}‘)
Return = (‘read’) (‘{Value}?’)
```