



UNIVERSIDADE FEDERAL DE ALAGOAS – UFAL

INSTITUTO DE COMPUTAÇÃO

CIÊNCIA DA COMPUTAÇÃO

## **COMPILADORES**

**Professor Alcino Dall'Igna Júnior**

**AUDREY EMMELY RODRIGUES VASCONCELOS**

**NATÁLIA DE ASSIS SOUSA**

**WILLIAM PHILIPPE LIRA BAHIA**

**MACEIÓ, AL 21 DE DEZEMBRO DE 2021**

## 1. DEFINIÇÕES DA GRAMÁTICA LIVRE DE CONTEXTO

As alternativas são separadas por barras verticais: ou seja, ' $a \mid b$ ' significa " $a$  ou  $b$ ";

Os colchetes indicam opcionalidade:  $[a]$  representa um  $a$  opcional, ou seja, " $a \mid \varepsilon$ " (onde  $\varepsilon$  se refere à sequência vazia);

As chaves indicam repetição:  $\{a\}$  significa " $\varepsilon \mid a \mid aa \mid aaa \mid \dots$ ";

O 'ch' denota qualquer caractere ASCII imprimível;

## 2. Terminais

```
letter = a | b | ... | z | A | B | ... | Z
digit = 0 | 1 | ... | 9
const = PI | EULER | NEW_LINE
PI = 3.14159
EULER = 2.71
NEW_LINE = '\n'
cteBool =
id = [a-z]([a-z][A-Z][0-9])*
str = "{ch}", onde ch denota qualquer caractere ASCII
imprimível
ch = 'ch' | '\0'
```

## 3. Produções Gramaticais

### a. Símbolo Inicial

$S = \text{Principal } S \mid \varepsilon$

### b. Declaração de escopo global e de funções

```
Principal = 'start' '{' Decl [Fun] '}' ';'
Fun = 'function' Type 'id' '(' Parameters ')' '{' Decl
      '}' ';' | ε
Parameter = Type 'id' Parameter {',' Type 'id'
Parameter} | ε
```

### c. Declaração e atribuição de variáveis

```
Decl = VarDecl
VarDecl = Type Id ';' Decl
Id = 'id' [Vector]
Vector = '[' digit ']' | ε
Atrib = Type Id '=' Value ';' | Id '=' Value ';' | ε
Value = letter | digit | 'ch'
```

### d. Tipos de dados

```
Type = 'int' | 'float' | 'char' | 'string' | 'bool' |
      'undefined'
```

### e. Instruções e comandos permitidos

```
Instruction = Command Instruction | ε
Command = If | Elseif | Else | While | DoWhile | For |
Loop | Return | FunctionCall
If = 'if' '(' Eb ')' '{' Instruction '}' Elseif Else
Elseif = {'elseif' '(' Eb ')' '{' Instruction '}}
Else = 'else' '{' Instruction '}' | ε
While = 'while' '(' Eb ')' '{' Instruction '}'
DoWhile = 'while' '(' Eb ')' '{' Instruction '}'
For = 'for' '(' [Expr] ';' [Expr] ';' [Expr] ')' '{'
Instruction '}'
Loop = 'loop' '(' [Expr] ';' [Expr] ';' [Expr] ';'
[Expr] ')' '{' Instruction '}'
Return = 'return' Expr ';'
FunctionCall = Id '(' Parameters ')'
```

### f. Entrada e saída de dados

```
Get = 'get' '(' Id [',' Id] ')' ';'
Put = 'put' '(' Id [',' Id] ')' ';'

```

### g. Expressões

```
Expr = Expr '+'
Expr = Eb
Eb = Eb '|' Tb
Eb = Tb
Tb = Tb '&' Fb
Tb = Fb
Fb = '!' Eu
Fb = Eu
Eu = Opu Ei
Opu = '-' | '+' | ε
Ei = Er Opig Er
Ei = Er
Opig = '==' | '!='
Er = 'false' | 'true'
Er = Ea Oprel Ea
Er = Ea
Oprel = '>' | '>=' | '<' | '<='
Ea = Ta Ear
Ear = Opa Ta Ear | ε
Ta = Fa Tar
Tar = Opm Fa Tar | ε
Fa = ' ' Fa
Fa = '(' Ea ') '
Fa = Id | FunctionCall | Const
Opa = '+' | '-'
Opm = '*' | '/'
```

#### 4. Associatividade e precedência de operadores

A tabela a seguir fornece as associatividades de vários operadores e suas precedências relativas. As precedências diminuem à medida que descemos na tabela.

Operadores (precedência)	Associatividade
!	direita para esquerda
- e + (unário)	direita para esquerda
* e /	esquerda para direita
+ e -	esquerda para direita
< <= > e >=	não associativo
== e !=	não associativo
&	esquerda para direita
	esquerda para direita