
UNIVERSIDADE FEDERAL
DE ALAGOAS

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 17 DE DEZEMBRO DE 2021**

1. **EBNF**

   As alternativas são separadas por barras verticais: ou seja, '*a* | *b* ' significa "*a* ou *b*".

   Os colchetes indicam opcionalidade: '[*a*]' representa um *a* opcional, ou seja, "*a* | ε" (ε se refere à sequência vazia).

   Os colchetes indicam repetição: '{*a*}' significa "ε | *a* | *aa* | *aaa* | ..."

2. **Regras Léxicas**

```
letter ::= a | b | ... | z | A | B | ... | Z
digit ::= 0 | 1 | ... | 9
ch ::= 'ch'| '\n' | '\0', onde ch denota qualquer caractere
ASCII imprimível
str :== "{ch}", onde ch denota qualquer caractere ASCII
imprimível
id ::= letter { letter | digit | _ }
```

3. **Produções Gramaticais**

```
prog : {func}

func : type id '(' parameters ')' '{' { type var_decl { ','
var_decl } ';' } { statement } '}' ';'
| undefined id '(' parameters ')' '{' { type var_decl { ','
var_decl } ';' } { statement } '}' ';'

var_decl : id [ '[' digit ']' ]

type : int
| float
| char
| string
| bool

parameters : undefined
| type id [ '[' ']' ] { ',' type id [ '[' ']' ] }

statement  : if '(' expr ')' '{' statement '}' [ elseif {'
statement '}'][ else {' statement '}' ] ';'
| while '(' expr ')' '{' statement '}' ';'
| do '{' statement '}' while '(' expr ')' ';'
| for '(' [ assg ] ';' [ expr ] ';' [ assg ] ')' '{'
statement '}' ';'
| loop '(' [ expr ] ';' [ expr ] ';' [ expr ] ')' '{'
statement '}' ';'
```

```
| return [ expr ] ';'
| assg ';'
| id '(' [expr { ',' expr } ] ')' ';'
| '{' { statement } '}'
| ';'

assg : id [ '[' expr ']' ] = expr

expr : simpleExp

simpleExp : simpleExp orop andExp | andExp

orop : |

andExp : andExp andop unaryRelExp | unaryRelExp

andop : &

relExp : relExp relop sumExp | sumExp

relop : ==
| !=
| <=
| <
| >=
| >

sumExp : sumExp sumop mulExp | mulExp

sumop : +
| -

mulExp : mulExp mulop unaryExp | unaryExp

mulop : *
| /

unaryExp : unaryop unaryExp |  exp

unaryop : –

unaryRelExp : notop unaryRelExp | relExp

notop : !
```

**4. Associatividades e precedências 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.

| Operador | Associatividade |
|---|---|
| !, - (unário) | direita para esquerda |
| *, / | esquerda para direita |
| +, - (binário) | esquerda para direita |
| <, <=, >, >= | esquerda para direita |
| ==, != | esquerda para direita |
| & | esquerda para direita |
| \| | esquerda para direita |