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 21 DE DEZEMBRO DE 2021

1. DEFINIÇÕES DA GRAMÁTICA LIVRE DE CONTEXTO

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" (onde ε se refere à sequência vazia);

As chaves indicam repetição: {*a*} significa "ε | *a* | *aa* | *aaa* | ...";

O ‘ch’ denota qualquer caractere ASCII imprimível;

2. Produções Gramaticais

a. Símbolo Inicial

```
S = Principal
```

b. Declaração de escopo global e de funções

```
Principal = 'start' '{' Programa '}' ';'
Programa = DeclaraVar Programa | Funcao Programa |
      Instrucao Programa | ε
```

c. Declaração e atribuição de variáveis

```
DeclaraVar = Type Id ArrayDecl IniciarId ListaId ';'
DeclaraVarFor = Type Id ArrayDecl IniciarId ListaId
ListaId = ',' Id ArrayDecl IniciarId ListaId | ε
ArrayDecl = '[' ExpNum ']' | ε
IniciarId = '=' Expressao | ε
```

d. Tipos de dados

```
Type = 'int' | 'float' | 'char' | 'string' | 'bool'
TypeFuncao = 'undefined' | Type ArrayType
ArrayType = '[' ']' | ε
```

e. Funções

```
Funcao = 'function' TypeFuncao Id '(' DeclParametros ')'
      '{' Programa '}'
DeclParametros = Type ArrayType Id ListaParametros | ε
ListaParametros = ',' Type ArrayType Id ListaParametros
      | ε
Argumentos = Expressao ArgumentosLista | ε
ArgumentosLista = ',' Expressao ArgumentosLita | ε
ChamadaFuncao = Id '(' Argumentos ')'
```

f. Instruções e comandos permitidos

```
Instrucao = Command Instrucao | ε
Command = If | While | DoWhile | For | Loop | Retorno |
      AtrOuFunInst | Get | Put
If = 'if' '(' Expressao ')' '{' Programa '}' Elseif Else
Elseif = 'elseif' '(' Expressao ')' '{' Programa '}'
     Elseif | ε
Else = 'else' '{' Programa '}' | ε
While = 'while' '(' Expressao ')' '{' Programa '}'
DoWhile = 'do' '{' Programa '}' 'while' '(' Expressao
')'
For = 'for' '(' [DeclaraVarFor] ';' [Expressao] ';'
     [AtribuicaoFor] ')' '{' Programa '}'
AtribuicaoFor = Id '=' Expressao ListaAtriFor
ListaAtriFor = ',' Id '=' Expressao ListaAtriFor | ε
Loop = 'loop' '(' 'int' Id ';' ExpNum ';'
     ExpNum ';' [ExpNum] ')' '{'
     Programa '}'
Retorno = 'return' Expressao ';'
AtrOuFunInst = Id ArrayDecl EqualOuCall
EqualOuCall = Atribuicao | FuncaoInstrucao
Atribuicao = '=' Expressao ';'
FuncaoInstrucao = '(' Argumentos ')' ';'
```

g. Entrada e saída de dados

```
Get = 'get' '(' Argumentos ')' ';'
Put = 'put' '(' Argumentos ')' ';'
```

h. Expressões

```
Expressao = ExpBool
ExpBool = ExpA ExpOr
ExpOr = '|' ExpA ExpOr | ε
ExpA = ExpN ExpAnd
ExpAnd = '&' ExpN ExpAnd | ε
ExpN = '!' ExpressaoLogica | ExpressaoLogica
ExpressaoLogica = ValorLogico OperacaoLogica
OperacaoLogica = OperadoresLogicos ValorLogico | ε
ValorLogico = ExpNum
```

```
ExpNum = ExpM ExpAditiva
ExpAditiva = '+' ExpM ExpAditiva | '-' ExpM ExpAditiva
     | ε
```

```
ExpM = ExpUnaria ExpMulti
ExpMulti = '*' ExpUnaria ExpMulti | '/' ExpUnaria
     ExpMulti | ε
ExpUnaria = '-' Valor | '+' Valor | Valor
Valor = IdOuFuncCall | Array OperadorSize |
     CastingOuExpressao | 'constanteInteira' |
     'constanteFloat' | 'constanteBool' | 'cte' |
     ExpString |

IdOuFuncCall = Id Call
Call = '(' Argumentos ')' | OperadorSize | ArrayDecl | ε

CastingOuExpressao = '(' TypeOrExp
TypeOrExp = Type ')' Expressao | Expressao ')'



ExpString = Palavra OpString
OpString = '+' Expressao OpString | ε
Palavra = 'string' | 'char'

Id = 'id'
OperadoresLogicos = '<' | '>' | '<=' | '>=' | '!=' |
'=='
OperadorSize = '.' 'size' | ε

Array = '[' ValorArray ']'
ValorArray = Expressao ListaArray | ε
ListaArray = ',' Expressao ListaArray | ε
```

3. 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