

# Linguagem **MINI-TRIANGLE**

(Páginas 8 e 9 de PLPJ)

- Gramática livre de contexto
- Dependências de contexto
- Semântica

Revisado em 03/06/2022

# Sintaxe Livre de Contexto

# Observações:

- A gramática é livre de contexto, portanto, as dependências de contexto não estão descritas nela
- Raiz é Program
- Símbolos terminais estão em negrito
- O símbolo  $\rightarrow$  é substituído por  $::=$
- Alguns símbolos não-terminais estão descritos informalmente
- Esta gramática não contempla símbolos inúteis, apesar de descrever a estrutura dos comentários

Program ::=

single-Command

single-Command ::=

V-name := Expression

| Identifier ( Expression )

| **if** Expression **then** single-Command

**else** single-Command

| **while** Expression **do** single-Command

| **let** Declaration **in** single-Command

| **begin** Command **end**

Command ::=

single-Command

| Command ; single-Command

Expression ::=

primary-Expression

| Expression Operator primary-Expression

primary-Expression ::=

Integer-Literal

| V-name

| Operator primary-Expression

| ( Expression )

V-name ::=

Identifier

Identifier ::=

Letter

| Identifier Letter

| Identifier Digit

Letter ::=

**a | b | c | ... | z**

Digit ::=

**0 | 1 | 2 | ... | 9**

Integer-Literal ::=

Digit

| Integer-Literal Digit

Operator ::=

+ | - | \* | / | < | > | =

Declaration ::=

single-Declaration

| Declaration ; single-Declaration

single-Declaration ::=

**const** Identifier ~ Expression

| **var** Identifier : Type-denoter

Type-denoter ::=

Identifier

Comment ::=

! CommentLine EOL

CommentLine ::=

Graphic CommentLine

| Graphic

EOL ::=

*end-of-line character*

Graphic ::=

*any printable character or space*

# **Dependências de Contexto**

single-Command ::=

V-name := Expression

- **A variável deve ter sido previamente declarada;**
- **O tipo da expressão avaliada deve ser compatível com o tipo da variável sendo atribuída.**

| Identifier ( Expression )

- **O procedimento chamado deve fazer parte da biblioteca da linguagem.**
- **O tipo do argumento deve ser compatível com o tipo do parâmetro.**

| **if** Expression **then** single-Command  
**else** single-Command

- **A expressão deve resultar num valor do tipo lógico (booleano).**

| **while** Expression **do** single-Command

- **A expressão deve resultar num valor do tipo lógico (booleano).**



Expression ::=

| Expression Operator primary-Expression

- **O operador deve ser compatível com os tipos dos operandos.**

primary-Expression ::=

| V-name

- **A variável deve ter sido previamente declarada e ser visível no escopo da referência.**

| Operator primary-Expression

- **O operador deve ser compatível com o tipo do operando.**

single-Declaration ::=

**const** Identifier ~ Expression

- **O nome não pode ter sido declarado previamente no mesmo bloco.**

| **var** Identifier : Type-denoter

- **O nome não pode ter sido declarado previamente no mesmo bloco.**

Type-denoter ::=

Identifier

- **O nome deve se referir a um tipo de dados válido.**

# Semântica

# Observações:

- Usual, semelhante à outras linguagens;
- Subprogramas tem um único parâmetro;
- Não existe comando condicional sem a cláusula else;
- O comando let ... in representa um bloco (com declarações e comandos);
- O comando begin ... end representa um comando composto (com vários comandos);
- O valor de uma constante (const) é determinado em tempo de execução.