

# LINGUAGEM LOGO

Introdução com xLogo

JOSÉ AUGUSTO N. G. MANZANO





Augusto N. G. Manzano



ORCID: 0000-0001-9248-7765

# Lingagem Logo

## Introdução com xLogo

São Paulo  
2021 - Propes Vivens

ISBN 978-65-00-26491-3



9 786500 264913

**© Copyright 2021 by José Augusto N. G. Manzano / Propes Vivens.**

**Todos os direitos reservados.** Proibida a reprodução total ou parcial, por qualquer meio ou processo, especialmente por sistemas gráficos, microfilmicos, fotográficos, reprogramáticos, fonográficos, videográficos, internet, e-books. Vedada a memorização e/ou recuperação total ou parcial em qualquer sistema de processamento de dados e a inclusão de qualquer parte da obra em qualquer programa jus cibernetico existentes ou que a venham existir. Essas proibições aplicam-se também às características gráficas da obra e à sua editoração, exceto pelo exporto no próximo parágrafo. A violação dos direitos autorais é punível como crime (art. 184 e parágrafos, do Código Penal, conforme Lei nº 10.695, de 07.01.2003) com pena de reclusão, de dois a quatro anos, e multa, conjuntamente com busca e apreensão e indenizações diversas (artigos 102 e 103 parágrafo único, 104, 105, 106 e 107 itens 1, 2 e 3 da Lei nº 9.610, de 19.06.1998, Lei dos Direitos Autorais).

Esta obra é distribuída gratuitamente em formato digital (somente em PDF) apenas e tão somente no sítio do autor ([www.manzano.pro.br](http://www.manzano.pro.br)) e na forma impressa comercialmente e disponibilizada nas plataformas **Clube de Autores** e **Agbook**. Nenhum outro local da Internet ou fora dela está autorizado a distribuir, seja gratuitamente ou comercialmente este material. Não é permitido o compartilhamento deste material em qualquer lugar ou por qualquer meio exceto o exposto neste parágrafo, bem como outro formato digital. Os infratores estão sujeitos a processo judicial.

O Autor acredita que as informações aqui apresentadas estão corretas e podem ser utilizadas para qualquer fim legal. Entretanto, não existe qualquer garantia, explícita ou implícita, de que o uso de tais procedimentos conduzirá sempre ao resultado desejado. O autor e a editora não poderão ser responsabilizados civilmente ou criminalmente. Os nomes de sítios e empresas, mencionados, foram utilizados apenas como ilustração, não havendo nenhum vínculo com a obra, não garantindo a sua existência nem divulgação a posteriori.

**Dados Internacionais de Catalogação na Publicação (CIP)**  
**(Câmara Brasileira do Livro, SP, Brasil)**

Manzano, José Augusto Navarro Garcia  
Linguagem logo [livro eletrônico] : introdução com xLogo / José Augusto Navarro Garcia Manzano. -- 1. ed. -- São Paulo : Ed. do Autor, 2021.  
PDF  
ISBN 978-65-00-26491-3  
1. Ciência da Computação 2. Computadores 3. Linguagem de programação (Computadores) 4. Processamento de dados I. Título.

21-72336

CDD-004.07

**Índices para catálogo sistemático:**

1. Ciência da computação : Estudo e ensino 004.07

Aline Grazielle Benitez - Bibliotecária - CRB-1/3129

**ED.VER - DATA**  
1.00 - 16/07/2021

Produção e Editoração: José Augusto Navarro Garcia Manzano  
Capa: canva.com / Espiral hexagonal (veja apêndice)

Edição: Propes Vivens

# FERRAMENTA UTILIZADA

Produto: **xLogo [Versão 0.9.96]**  
Sítio: <https://xlogo.tuxfamily.org/pt/>

## REQUISITOS DE HARDWARE E SOFTWARE

- ◆ 512 MB de memória RAM;
- ◆ 5 MB de espaço em disco para instalação do ambiente;
- ◆ Sistema Operacional Windows XP de 32 bits ou superior;
- ◆ Monitor com 1024 x 768 pontos ou superior para melhor visualização;
- ◆ Mouse ou outro periférico de apontamento;
- ◆ Modem e acesso à Internet.



# CARTA AO ESTUDANTE

Olá, estudante de programação.

Espero que você esteja bem e com excelente animo para desenvolver habilidades na arte da programação de computadores a partir de uma linguagem de programação que segue o estilo declarativo de definição. Este livro vem de encontro a sua aprendizagem fornecendo diversos aspectos práticos no uso da linguagem Logo como suporte as aulas de lógica de programação, podendo ser usado por pessoas de todas as idades.

A linguagem Logo é uma ferramenta que proporciona a aprendizagem de detalhes lógicos de forma divertida e descontraída partindo-se de um ponto aparentemente inocente e permitindo ao estudante de programação desenvolver aplicações robustas e complexas. Neste livro se faz uma introdução a linguagem e seus principais recursos de operação desde o uso da geometria da tartaruga até o desenvolvimento de programas independentes da operação gráfica.

O livro encontra-se dividido em cinco capítulos, mais alguns apêndices complementares que abordam diversas características de uso da linguagem, onde alguns capítulos possuem como reforço um conjunto de exercícios de fixação. São fornecidas orientações desde a obtenção e instalação da fermenta de trabalho *xLogo* a apresentação de diversas ações que podem ser produzidas com a linguagem, como: funções; operadores aritméticos, lógicos e relacionais; ações de entrada e saída; decisões; recursividades; escopo e visibilidade de variáveis; uso de procedimentos, entre outros. São demonstradas ações que produzem formas geométricas a partir de diversas primitivas (comandos da linguagem) e outras operações.

Cabe destacar que este livro é distribuído gratuitamente em formato eletrônico PDF no site do autor ou disponibilizado comercialmente em sua forma impressa a partir das plataformas de publicação Clube de Autores e Agbook para quem desejar ter o material no formato livro impresso.

Este trabalho não passou por revisões de língua portuguesa e está em sua forma bruta, versão pré-alpha. Por isso, poderão ser encontrados erros de escrita, concordância ou outros que passaram “batidos”, exceto os códigos de programas apresentados os quais foram exaustivamente testados. Auxílios, neste sentido, são sempre bem vindos, desde que sejam realizados sem nenhum interesse financeiro ou comercial.

Para auxiliar parte do estudo você encontrará os arquivos dos códigos dos exercícios de aprendizagem disponíveis para aquisição em:

[https://github.com/J-AugustoManzano/livro\\_Lого-Intro-xLogo](https://github.com/J-AugustoManzano/livro_Lого-Intro-xLogo).

Espero que este conteúdo possa lhe ser bastante útil.

Com grande abraço.  
Augusto Manzano.



# **AGRADECIMENTOS**

À minha esposa Sandra e à minha filha Audrey, motivos de constante inspiração ao meu trabalho. Pela paciência nos momentos de ausência que tenho, quando estou absorto em escrever, dedicando-me ao ensino.

Há você que me lê neste livro e a todos os estudantes que passaram e passam por minhas mãos, que acreditaram e acreditam na minha pessoa e seguiram e seguem as orientações passadas; por me incentivarem continuamente quando me questionam sobre temas que ainda não conheço, por me levarem a um patamar maior, por exigirem assim que eu pesquise mais e retorno a você conhecimentos na forma de livros.

Vida longa e próspera.



# SUMÁRIO

1.	INTRODUÇÃO	
1.1.	Linguagem Logo.....	11
1.2.	Obtenção, instalação do ambiente xLogo .....	12
1.3.	O ambiente xLogo .....	14
1.4.	Modo externo de ajuda sobre xLogo .....	17
2.	AÇÕES BÁSICAS	
2.1.	Primitivas iniciais .....	19
2.2.	Outras interações .....	22
2.3.	Primitivas complementares .....	24
3.	AÇÕES ESPECIALIZADAS	
3.1.	Repetições .....	27
3.2.	Procedimentos .....	29
3.3.	Sub-rotinas .....	33
3.4.	Variável .....	35
3.5.	Decisão .....	38
3.6.	Recursão .....	43
3.7.	Memorização e amnésia .....	46
3.8.	Exercícios de fixação .....	47
4.	AÇÕES ESPECÍFICAS	
4.1.	Entrada de dados .....	49
4.2.	Randomização .....	51
4.3.	Coordenadas .....	52
4.4.	Cores .....	56
4.5.	Fractais .....	58
4.6.	Outras repetições .....	60
4.7.	Exercícios de fixação .....	64
5.	AÇÕES COMPLEMENTARES	
5.1.	Funções matemáticas .....	67
5.2.	Funções lógicas .....	69
5.3.	Algumas funções complementares .....	70
5.4.	Programação sem figuras .....	71
5.5.	Manipulação básica de dados .....	74
5.6.	Escopo e visibilidade de variáveis .....	76
5.7.	Exercícios de fixação .....	79
	APÊNDICES	
A.	Exemplos geométricos .....	83
B.	Música .....	97
C.	Espiral hexagonal (imagem da capa) .....	99
D.	Gabarito .....	103
	REFERÊNCIAS BIBLIOGRÁFICAS .....	117



---

## CAPÍTULO 1 - Introdução

A programação declarativa caracteriza-se por ser um paradigma de programação onde se diz a um computador o que deve ser feito e não como ser feito como ocorre no paradigma imperativo. Neste sentido, este capítulo apresenta a linguagem de programação Logo baseada na linguagem Lisp que é o primeiro exemplar de uma linguagem de programação declarativa.

### 1.1 - Linguagem Logo

A linguagem Logo foi desenvolvida durante a década de 1960, por uma equipe multidisciplinar dirigida pelo Filósofo, Matemático, Pesquisador e Professor Seymour Papert, com coautoria de Cynthia Solomon no *Massachusetts Institute of Technology* (MIT) com a participação direta do Professor Marvin Minsky e participação indireta de Daniel Bobrow e Wally Feurzeig.

Em 1960 o Professor Papert conhece na Universidade de Sorbonne (França) Jean Piaget e inicia com este um trabalho relacionado a teoria de aprendizagem. Deste momento histórico veio a influência para seus estudos nas áreas de inteligência artificial e robótica educacional, dando origem a linguagem Logo (LOGO FOUNDATION, 2015).

Em 1961 em uma conferência na Inglaterra o Professor Seymour Papert conhece o Professor Marvin Minsky do MIT um dos grandes expoentes da Inteligência Artificial (IA). Nessa ocasião eles apresentaram artigos muito semelhantes sobre o estudo de IA e o uso dessa tecnologia por crianças. Isso os aproxima e leva em 1964 o Professor Papert a integrar o Grupo de Inteligência Artificial do MIT. No MIT Papert conhece Daniel Bobrow, ex-aluno do Professor Minsky que foi trabalhar em uma empresa de pesquisa e desenvolvimento chamada *Bolt, Beranek and Newman* (BBN). Na BBN Daniel Bobrow conhece Wally Feurzeig e põe Feurzeig e Papert em contato. Nesta ocasião Bobrow, Papert e Feurzeig conversam sobre a linguagem que Papert deseja criar para crianças chamada *Mathland*, que posteriormente tornou-se *Logo* como um dialeto da linguagem Lisp (PALEOTRONIC, 2021).

Logo é uma linguagem declarativa com toques de programação imperativa. É fundamentada sobre os princípios da programação lógica e funcional tendo sido direcionada inicialmente a crianças. Quando a linguagem surgiu não existiam microcomputadores e o acesso a essa tecnologia era extremamente restrito. No entanto, após o ano de 1975 com o surgimento dos microcomputadores e o barateamento da tecnologia computacional o uso da linguagem se tornou mais popular. O ambiente de trabalho é baseado principalmente sobre uma interface plana com um ícone (cursor central) chamado tartaruga que tem por objetivo percorrer o plano desenhando imagens baseadas em figuras geométricas a partir de quatro comandos principais, suas primitivas de operação: **PARAFRENTE**, **PARATRÁS**, **PARADIREITA**, **PARAESQUERDA**. Além desses recursos existem outros que complementam a linguagem como: **REPITA**, **SE**, **APRENDA**, **USELÁPIS**, **USEBORRACHA**, entre outros.

Apesar da linguagem ser muito conhecida devido ao modo de operação chamado *geometria da tartaruga*, ela é mais completa do que isso possuindo outros recursos. O efeito do modo *geometria da tartaruga* é apenas uma parte do que a linguagem efetivamente é.

Ser conhecida como a primeira linguagem para crianças também trouxe um pequeno inconveniente. Muitas pessoas deixam de levar a linguagem a sério por pensarem, apenas, que é uma linguagem lúdica para ensinar apenas crianças, quando pessoas de outras idades podem também fazer grande uso desta linguagem.

## 1.2 - Obtenção, instalação do ambiente xLogo

Este livro foca para uso uma ferramenta chamada **xLogo** desenvolvida para ser executada em sistema operacional Windows. A versão aqui descrita funciona na edição Windows 10.

Para obter o ambiente **xLogo** acesse o endereço "<https://xlogo.tuxfamily.org/pt/>". A partir do menu lateral esquerdo selecione a opção "**Como conseguir?**", como indicado na figura 1.1.

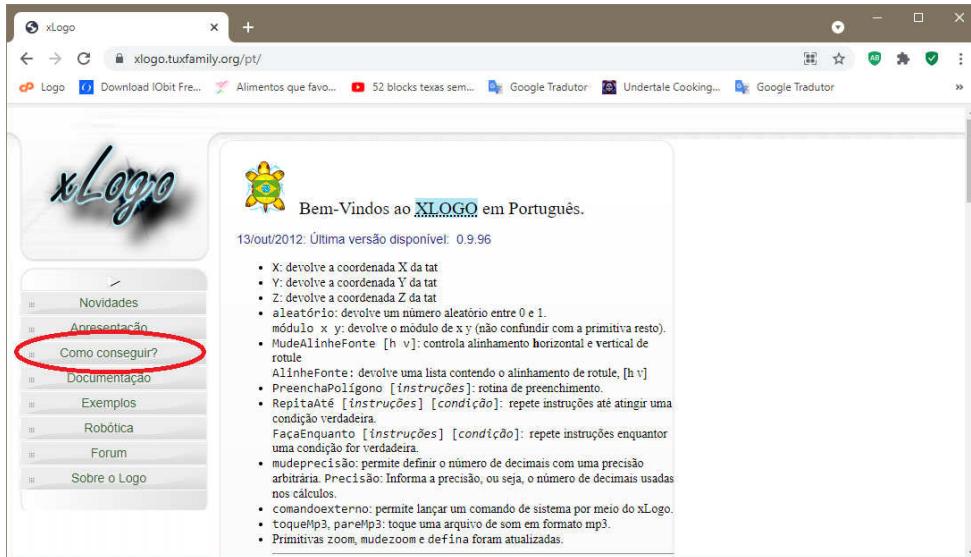


Figura 1.1 - Sítio oficial do projeto "XLogo".

Após selecionar a opção "**Como conseguir?**", é apresentada a página para obtenção da linguagem como indica a figura 1.2.



Figura 1.2 - Sítio da plataforma "SOURCEFORGE".

Posicione o ponteiro do *mouse* sobre o arquivo "**xlogo.jar**" na seção "**Arquivo XLogo**", selecione o botão de contexto (normalmente botão direito do *mouse*) e selecione no menu de contexto apresentado a opção "**Salvar link como...**" se estiver em uso navegador "**Chrome**" ou selecione "**Salvar destino como**" se estiver em uso o navegador "**Edge**". Será apresentada a caixa de diálogo "**Salvar como**" (figura 1.3) possivelmente apontando para a pasta "**Downloads**". Neste instante acione o botão "**Salvar**".

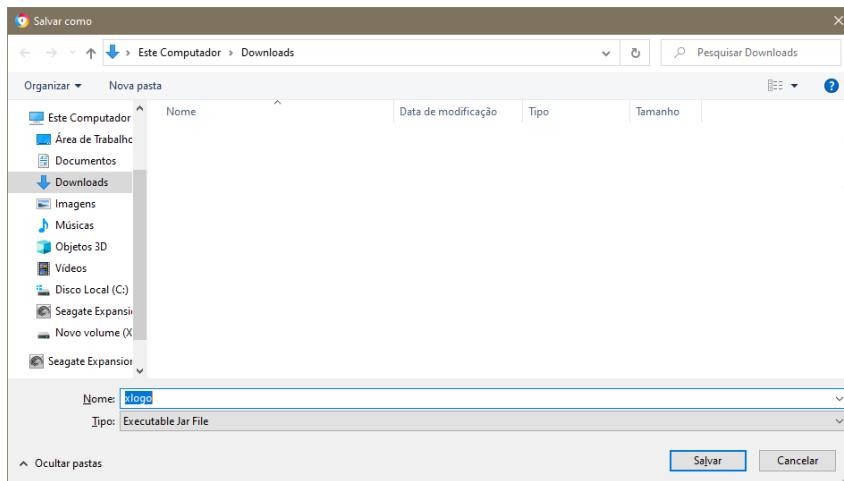


Figura 1.3 - Salvar "xlogo.jar" na pasta "Downloads".

Concluído o processo de cópia do programa feche as janelas que por ventura estejam abertas e vá ao local onde o arquivo foi copiado. Para tanto, abra o aplicativo **Explorador de Arquivos**, localize a pasta **Downloads** e selecione com um duplo clique do ponteiro do *mouse* por meio do botão de ação (geralmente o botão esquerdo).

Neste momento, poderá ocorrer uma de duas possibilidades: ou o programa "**xlogo.jar**" é executado (figura 1.5) ou o programa terá sua execução negada com a apresentação da mensagem de erro "**O Windows não pode abrir este tipo de arquivo (.jar)**" como mostra a figura 1.4.

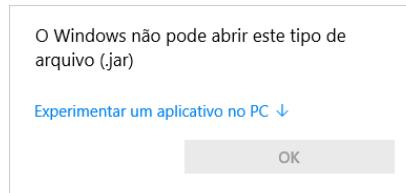


Figura 1.4 - Mensagem de erro, o programa não pode ser executado.

A apresentação do erro ocorrerá caso você não tenha instalado em seu computador o ambiente de execução "Java" conhecido como "**Java SE**". O ambiente JRE é gratuito e pode ser adquirido, se necessário, no endereço "<https://www.oracle.com/java/>". Na página apresentada localize a opção de obtenção do ambiente "**Java SE**", normalmente botão "**Download Java now**" e selecione-o. Na página apresentada selecione o link "**JDK Download**" e localize na lista apresentada de opções a versão indicada para seu sistema operacional Windows, selecione o arquivo desejado, leia a licença e após concordar com seus termos acione o botão "**Download**" apresentado e aguarde o arquivo ser copiado. Feche o navegador.

No programa **Explorador de Arquivos** execute o programa *Java* copiado e faça sua instalação. Após selecionar o programa basta confirmar a instalação comacionamentos sucessivos dos botões "**Next**" e quando apresentado o botão "**Close**", acione-o e pronto, a partir deste momento o programa "**xLogo.jar**" já pode ser executado.

Na primeira execução do programa "*xLogo*" é apresentada a tela de seleção dos idiomas suportados como indicado na figura 1.5. Neste momento, escolha com cuidado a bandeira desejada. Para o estudo deste livro selecione a bandeira do "**Brasil**" e em seguida acione o botão "**OK**".

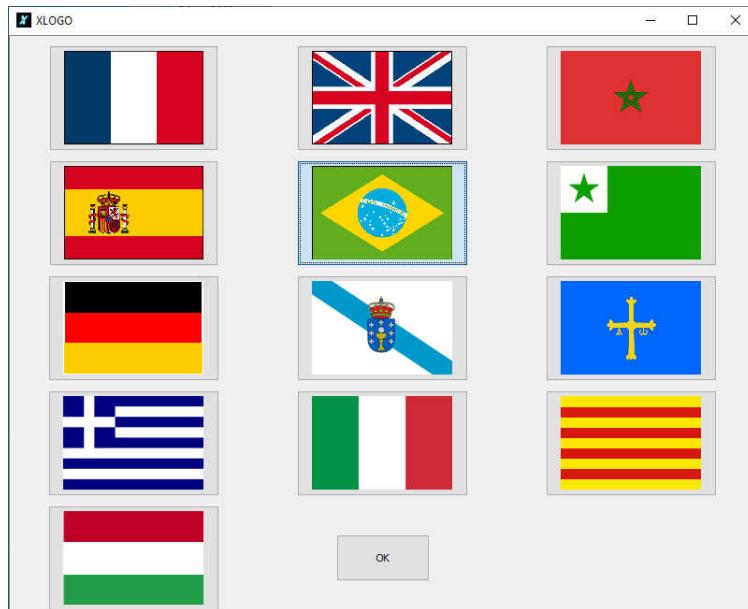


Figura 1.5 - Seleção do idioma da linguagem Logo no ambiente "xLogo".

A partir da seleção do idioma e da segunda execução do ambiente ocorrerá a apresentação do ambiente de trabalho da linguagem como apresenta a figura 1.6.

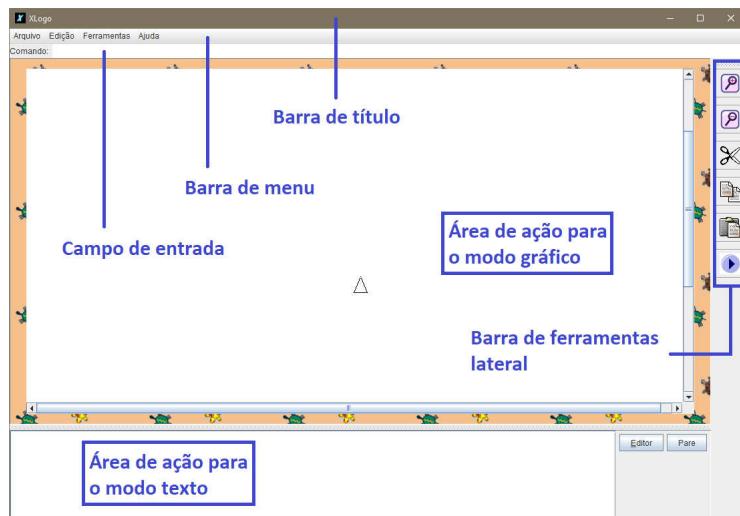


Figura 1.6 - Ambiente "xLogo".

### 1.3 - O ambiente xLogo

O ambiente "xLogo" possui definido algumas áreas de identificação, como: **barra de título**, **barra de menu**, **linha de comando**, **campo de entrada**, **área de ação para o modo gráfico**, **área de ação para o modo texto** com botões "**Editor**" e "**Pare**" e **barra de ferramentas** simplificada ao lado esquerdo da tela com as operações de **ampliar**, **reduzir**, **cortar**, **copiar**, **colar** e **executar**.

Veja os detalhes anteriormente comentados na figura 1.7. Nesta estrutura as partes mais importantes são a *barra de menu*, a *barra de ferramentas* e o *campo de entrada* por proporcionarem os mecanismos de entrada de operações, as *áreas de ação gráfica e texto* são usadas pelo programa para apresentar os resultados das ações operacionalizadas.



**Figura 1.7 - Estrutura de organização do ambiente "xLogo".**

A barra de menu é formada pelos comandos: **Arquivo** (*Novo, Abrir, Abrir, Guardar como, Guardar, Capturar imagem, Zona de texto e Sair*), **Edição** (*Recortar, Copiar, Colar e Selecione tudo*), **Ferramentas** (*Mudar cor do lápis, Mudar cor do fundo, Arquivo inicial, Traduzir procedimento, Apaga procedimentos e Preferências*) e **Ajuda** (*Manual, Licença, Tradução da licença, Traduz o xLogo e Sobre*). Observe o que faz cada uma das opções do menu:

#### Comando "**Novo**" do menu "**Arquivo**"

Este comando efetua a limpeza de todos os procedimentos que eventualmente estejam carregados na memória.

#### Comando "**Abrir...**" do menu "**Arquivo**"

Este comando efetua o carregamento de um arquivo de script Logo do disco para a memória removendo os procedimentos existentes.

#### Comando "**Guardar como.....**" do menu "**Arquivo**"

Este comando guarda ("salva") no disco todo o conteúdo da memória sob a definição de um novo nome, se anteriormente foi usado os comandos "Guardar" ou "Abrir...".

#### Comando "**Guardar**" do menu "**Arquivo**"

Este comando guarda ("salva") no disco todo o conteúdo da memória sob a definição de um nome.

#### Comando "**Capturar imagem**" do menu "**Arquivo**"

Este comando permite armazenar o desenho exibido na tela como uma imagem nos formatos "jpg" ou "png".

#### Comando "**Zona de texto**" do menu "**Arquivo**"

Este comando permite gravar o conteúdo da área de texto no formato "rtf".

#### Comando "**Sair**" do menu "**Arquivo**"

Este comando finaliza a execução do ambiente.

**Comando "Recortar" do menu "Edição"**

Este comando recorta o texto selecionado para o *clipboard*.

**Comando "Copiar" do menu "Edição"**

Este comando copia o texto selecionado para o *clipboard*.

**Comando "Colar" do menu "Edição"**

Este comando cola o conteúdo do *clipboard*.

**Comando "Selecionar tudo" do menu "Edição"**

Este comando seleciona um texto para alguma edição.

**Comando "Mudar a cor do lápis" do menu "Ferramentas"**

Este comando define uma cor para o lápis da tartaruga.

**Comando "Mudar a cor do fundo" do menu "Ferramentas"**

Este comando define uma cor para o fundo da área de ação do modo gráfico.

**Comando "Arquivo inicial..." do menu "Ferramentas"**

Este comando informa o caminho para uso dos arquivos de inicialização.

**Comando "Traduzir procedimentos" do menu "Ferramentas"**

Este comando aciona o modo de tradução automática de qualquer procedimento de um idioma para outro.

**Comando "Apaga procedimentos" do menu "Ferramentas"**

Este comando permite selecionar um procedimento da memória para remoção.

**Comando "Preferências" do menu "Ferramentas"**

Este comando abre uma caixa de diálogo que permite configurar diversos detalhes do ambiente.

**Comando "Manual" do menu "Ajuda"**

Este comando deveria abrir um manual de orientações de uso da linguagem. No entanto, na versão usada para este livro não surtiu nenhum efeito.

**Comando "Tradução da licença" do menu "Ajuda"**

Este comando apresenta uma versão não oficial da licença, neste caso, em português.

**Comando "Traduza o xLogo" do menu "Ajuda"**

Este comando permite ao usuário auxiliar a tradução do xLogo tendo que enviar o arquivo gerado a equipe de desenvolvimento.

**Comando "Sobre" do menu "Ajuda"**

Este comando apresenta informações de autoria do ambiente.

A barra de títulos do ambiente além do nome como título "xLogo" do lado esquerdo, possui do lado deito os botões: **Minimizar**, **Maximizar** e **Fechar**.

Por ser este um trabalho de introdução, focado nas ações da geometria da tartaruga, não são apresentados muitos comandos, apenas os que são essenciais aos objetivos propostos. No entanto, a partir do modo **Ajuda** da ferramenta é possível descobrir muitos recursos interessantes.

No meio da tela na área de ação do modo gráfico encontra-se o ícone de um triângulo que representa o *prompt* de operação do ambiente Logo. Este triângulo é conhecido, carinhosamente, pelo nome **TARTARUGA**, podendo ser substituído por outras imagens a partir da seleção da opção "**Preferências**" do menu "**Ferramentas**" a partir da guia "**Roupa da Tartaruga**".

Para sair use "**Arquivo/Sair**" ou acione as teclas de atalho "**<Alt> + <F4>**" ou selecione o botão "**X**" na barra de título.

#### 1.4 - Modo externo de ajuda spnre xLogo

O modo de ajuda do ambiente "xLogo" não é carregado internamente. Seu material de consulta encontra-se publicado externamente a partir de seu sítio oficial. Assim sendo, acesse o endereço Web: "<http://xlogo.tuxfamily.org/pt/html/index.html>" e você terá acesso ao conteúdo explicativo da ferramenta, como indicado na figura 1.8.

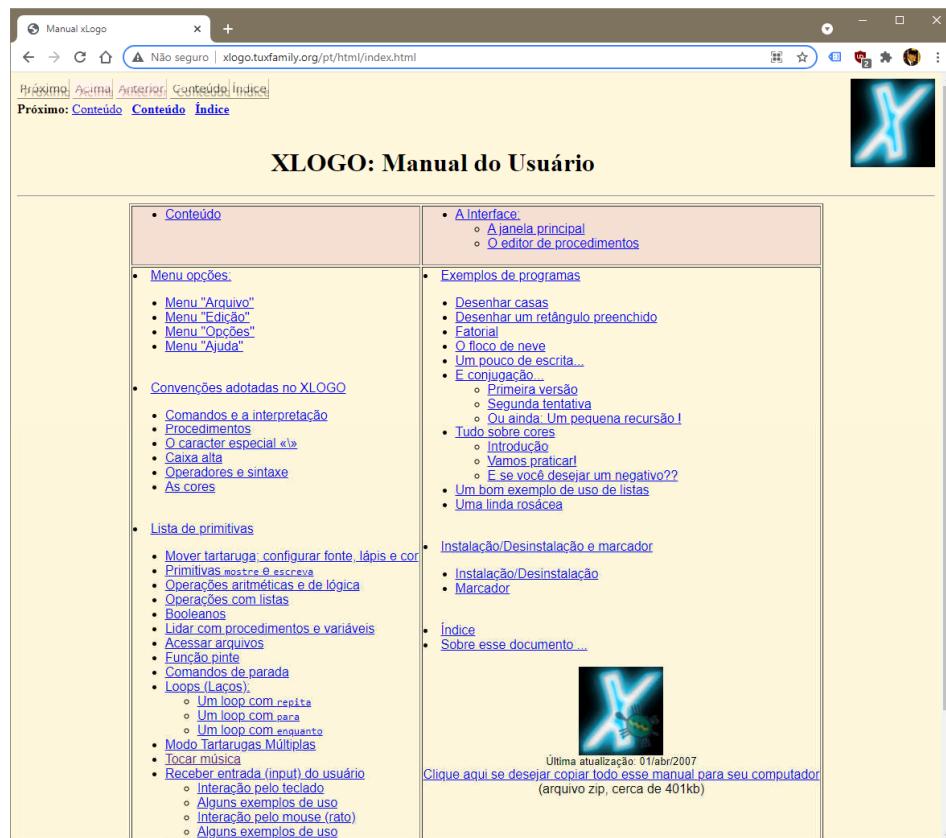


Figura 1.8 - Modo de deajuda "xLogo" on line.

# ANOTAÇÕES

## CAPÍTULO 2 - Ações básicas

As *primitivas* em Logo caracterizam-se por serem o conjunto de comandos e funções básicas da linguagem e de como esses elementos podem ser usados por estudantes para interagirem com o ambiente como um todo. Os comandos ou *primitivas* são palavras que determinam ações a serem executadas pela linguagem como **PARAFRENTE** e **PARADIREITA** entre outros. As funções por sua vez caracterizam-se por serem recursos operacionais que devolvem uma resposta a sua operação como **PI**, **SOMA**, **INTEIRO** e etc.

### 2.1 - Primitivas iniciais

O conjunto de comandos na linguagem Logo é extenso. No entanto, não é necessário conhecer todos as primitivas para poder usufruir da linguagem, pois a partir de um pequeno conjunto de ações já é possível realizar algumas ações divertidas.

Antes de começar é importante ter em mente que um comando é uma ação a ser realizada no computador pela linguagem e que este pode ser usado de forma isolada ou acompanhado de um parâmetro. Um comando escrito com ou sem parâmetro pode com o acionamento da tecla <Enter> passar ao computador uma instrução de ação a ser realizada.

Veja o que é preciso para se fazer o desenho de um **quadrado**.

No campo para a entrada de comandos, dados e instruções escreva tanto em letras minúsculas quanto em letras maiúsculas a instrução seguinte e acione após escrever-la a tecla <Enter>:

**PARAFRENTE 80**

Após executar a instrução anterior formada pelo comando **PARAFRENTE** e pelo parâmetro "80" ocorre o desenho de uma linha de baixo para cima na posição central da tela com 80 pixels (*pixel* é o menor ponto luminoso imprimível na tela do monitor de vídeo - ecrã). A instrução após sua execução é apresentada dentro da área de ação do modo texto, como indicado na figura 2.1.

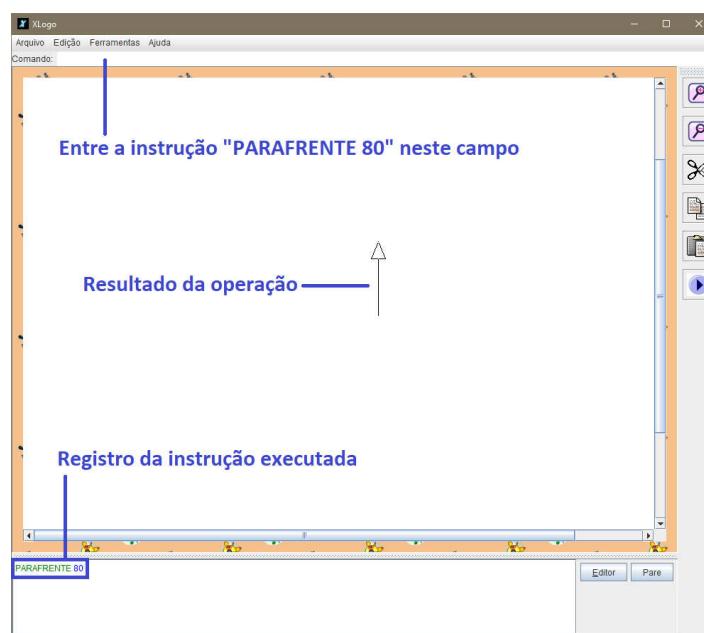


Figura 2.1 - Resultado da ação para a instrução "PARAFRENTE 80".

Além da primitiva **PARAFRENTE**, há sua inversa **PARATRÁS** que faz com que a TARTARUGA ande de cima para baixo. Os valores padrão de deslocamento para esses parâmetros são normalmente configurados para operar de "0" até "500" pixels. O "xLogo" em sua documentação na diz sobre o limite máximo. Para todos os efeitos esta obra considera como limite o valor "500".

O primeiro traço do que deverá ser um quadrado já está definido. Agora é necessário fazer com que o próximo traço seja desenhado em um sentido lateral. Observe que o desenho do primeiro traço ocorreu de baixo para cima, ou seja, ocorreu no sentido **NORTE**. Considere que se deseja fazer com que o próximo traço seja desenhado no sentido **LESTE** ou seja a direita do ponto em que a TARTARUGA se encontra. Para este caso, use o comando **PARADIREITA** com o parâmetro **90** a partir da instrução:

**PARADIREITA 90**

Veja que a TARTARUGA é apontada para a direção **LESTE** como mostra a figura 2.2.

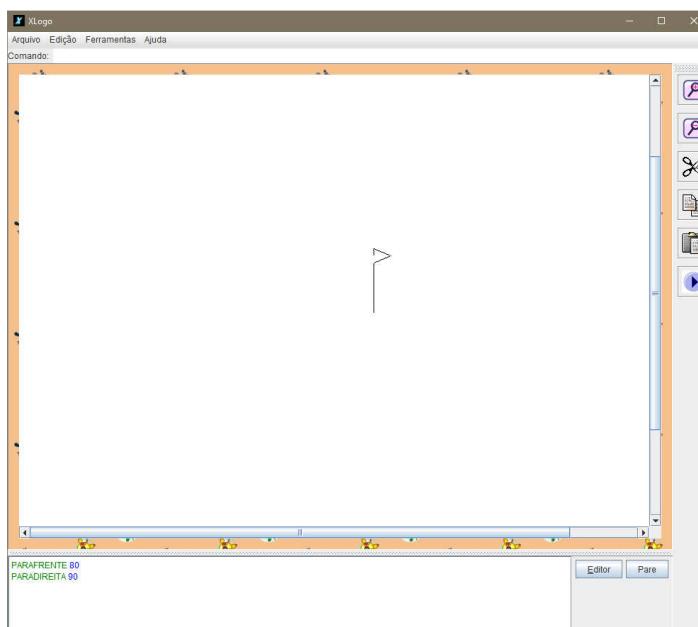


Figura 2.2 - Resultado da tartaruga no sentido LESTE após instrução "PARADIREITA 90".

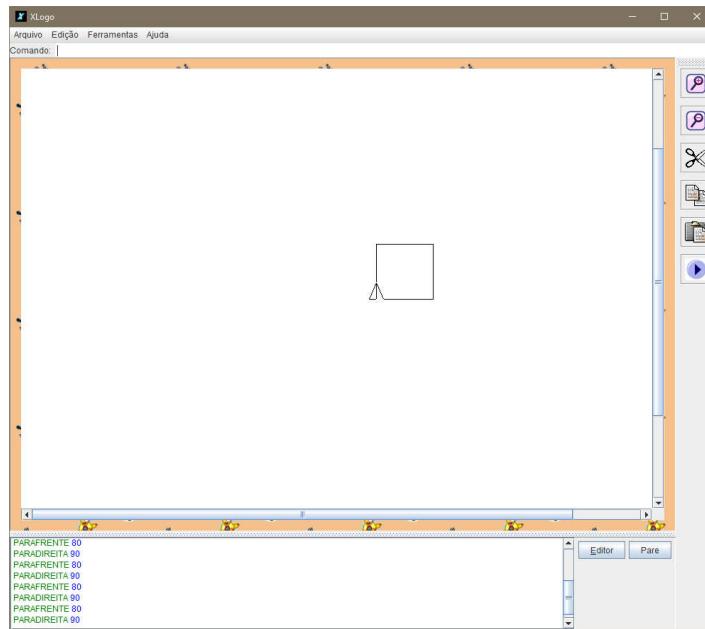
Além do comando **PARADIREITA** que altera o sentido de giro da TARTARUGA de **NORTE** para **LESTE**, há seu inverso **PARAESQUERDA** que altera o sentido de giro da TARTARUGA de **NORTE** para **OESTE**. Os valores padrão de parâmetros permitidos para esses comandos são de "0" a "360" graus.

A partir da definição da nova direção a ser seguida basta repetir por mais três vezes a execução das instruções:

**PARAFRENTE 80**

**PARADIREITA 90**

Veja junto a figura 2.3 a apresentação do conjunto de instruções e a imagem do quadrado desenhada. Note que para apresentar o conjunto de instruções o tamanho da janela foi ajustado. Este ajuste pode ser feito com o posicionamento do *mouse* sobre a linha que divide ás áreas de ação do modo gráfico e do modo texto.

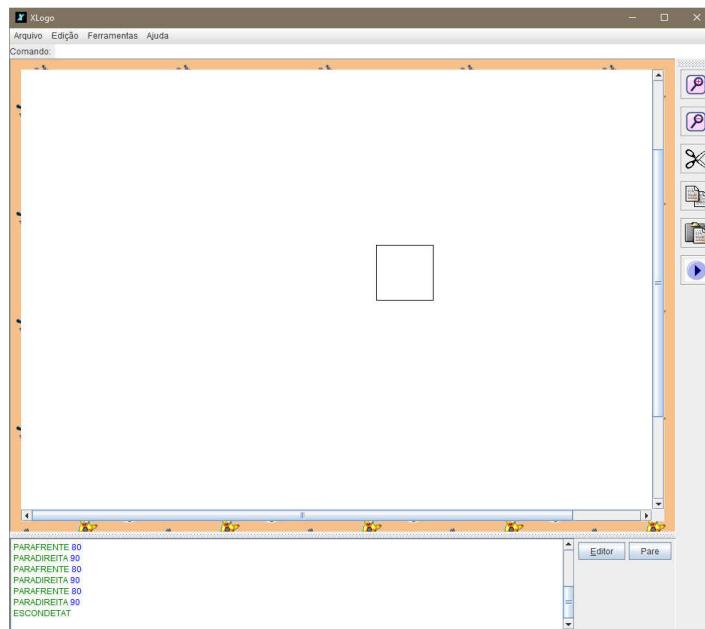


**Figura 2.3 - Apresentação de um quadrado.**

Note que a sobreposição da TARTARUGA sobre a figura desenhada pode atrapalhar um pouco sua visualização. Neste sentido, é possível pedir que a TARTARUGA seja ocultada a partir do uso da instrução:

#### ESCONDETAT

Observe junto a figura 2.4 a apresentação da imagem do quadrado sem a sobreposição da TARTARUGA.



**Figura 2.4 - Apresentação de um quadrado sem sobreposição da tartaruga.**

Para retornar a apresentação da TARTARUGA use a instrução:

#### MOSTRETAT

Uma das ações mais importantes é um comando especial que efetua a limpeza da área de ação do modo gráfico e coloca a TARTARUGA na posição central apontando para o norte. Neste momento, execute a instrução:

### LIMPEDESENHO

Além da limpeza da área de ação do modo gráfico é possível limpar a área de ação do modo texto. Para tanto, use a instrução:

### LIMPETEXTO

Veja que até este ponto você já aprendeu que uma instrução pode ser definida a partir do uso de um comando (**ESCONDETAT**) ou de um comando com parâmetro (**PARADIREITA 90**) e que a partir dessas diretivas básicas já é possível fazer diversos desenhos geométricos interessantes, ou seja, de um triângulo até uma circunferência. Mas antes de partir para outras criações é importante conhecer mais alguns detalhes da linguagem Logo.

## 2.2 - Outras interações

Logo é um ambiente interativo que além de desenhar pode realizar diversas operações, como cálculos aritméticos. A partir do uso das primitivas **ESCREVA** e **PALAVRA** veja algumas operações aritméticas simples interessantes a partir das seguintes instruções:

```
ESCREVA PALAVRA 5 + 2 "\n
ESCREVA PALAVRA 5 - 2 "\n
ESCREVA PALAVRA 5 * 2 "\n
ESCREVA PALAVRA 5 / 2 "\n
```

Veja que as instruções anteriores para surtirem o efeito esperado usam duas primitivas a primitiva **ESCREVA** faz a escrita do resultado da operação indicada e a primitiva **PALAVRA** pega o resultado da operação e concatena ao resultado o código "\n" que faz um salto de linha após a escrita do resultado. A figura 2.5 mostra os resultados das operações aritméticas estabelecidas dentro na área de ação do modo texto.

```
ESCREVA PALAVRA 5 + 2 "\n
7
ESCREVA PALAVRA 5 - 2 "\n
3
ESCREVA PALAVRA 5 * 2 "\n
10
ESCREVA PALAVRA 5 / 2 "\n
2.5
```

Figura 2.5 - Resultado de operações aritméticas.

O comando **ESCREVA** também pode ser usado para apresentar mensagens. Mas, neste caso é importante ter atenção no que se deseja apresentar. Se for uma mensagem simples, basta após o comando indicar como parâmetro a palavra precedida de aspa inglesa (""), mas se for uma frase é importante que está esteja definida entre colchetes ([ e ]). Observe os exemplos para apresentação da palavra "Logo" e da frase "Linguagem Logo".

```
ESCREVA "Logo\n
ESCREVA [Linguagem Logo\n]
```

Veja o resultado das duas apresentações na figura 2.6.

```
ESCREVA "Logo\n
Logo
ESCREVA [Linguagem Logo\n]
Linguagem Logo
```

Figura 2.6 - Resultado da apresentação de texto.

Além das operações aritméticas básicas é possível fazer a apresentação de outros resultados baseando-os no uso de *funções*. Veja a seguir alguns exemplos no uso de algumas funções integradas com o uso do comando **ESCREVA**. Acompanhe as seguintes instruções:

```
ESCREVA PALAVRA SOMA 5 2 "\n
ESCREVA PALAVRA DIFERENÇA 5 2 "\n
ESCREVA PALAVRA PRODUTO 5 2 "\n
ESCREVA PALAVRA QUOCIENTE 5 2 "\n
ESCREVA PALAVRA RESTO 5 2 "\n
ESCREVA PALAVRA INTEIRO QUOCIENTE 5 2 "\n
ESCREVA PALAVRA ABS -5 "\n
ESCREVA PALAVRA POTÊNCIA 5 2 "\n
ESCREVA PALAVRA RAIZQ 25 "\n
```

Veja os resultados do uso de funções na figura 2.7.

```
ESCREVA PALAVRA SOMA 5 2 "\n
7
ESCREVA PALAVRA DIFERENÇA 5 2 "\n
3
ESCREVA PALAVRA PRODUTO 5 2 "\n
10
ESCREVA PALAVRA QUOCIENTE 5 2 "\n
2
ESCREVA PALAVRA RESTO 5 2 "\n
1
ESCREVA PALAVRA INTEIRO QUOCIENTE 5 2 "\n
2
ESCREVA PALAVRA ABS -5 "\n
5
ESCREVA PALAVRA POTÊNCIA 5 2 "\n
25
ESCREVA PALAVRA RAIZQ 25 "\n
5
```

Figura 2.7 - Apresentação dos resultados no uso de funções.

**OBS:**

O comando **ESCREVA** possui como sinônimo o comando **MOSTRE** que gera o mesmo efeito de ação. Experimente.

Note que as funções apresentadas são operadas com um ou dois parâmetros. Tomando por base a função **SOMA** imagine o desejo de realizar a apresentação da soma de três argumentos. Veja o que acontece:

```
ESCREVA PALAVRA SOMA 1 2 7 "\n
```

O desejo desta ação é, de fato, obter o resultado "10" como resposta a soma de "1 + 2 + 7". No entanto, surpreendentemente ocorre a apresentação do valor "3" que é a soma dos valores "1" e "2" e a indicação da mensagem de erro "37Que devo fazer com" informando que o ambiente não sabe o que fazer com o valor a mais, neste caso "7". Isto ocorre devido ao fato da função **SOMA** (e de outras funções) fazer uso de dois parâmetros e não de três parâmetros como pretendido. Mas há uma maneira de fazer esta ocorrência funcionar, basta executar a instrução:

```
ESCREVA PALAVRA (SOMA 1 2 7) "\n
```

Veja que ao colocar a função **SOMA** e os parâmetros "1", "2" e "7" dentro de parênteses consegue-se obter o resultado da operação pretendida, ou seja, obter o valor "10".

Usar parênteses entre uma função e seus parâmetros é uma maneira de contornar a limitação no uso de parâmetros. Então, mantenha atenção sobre esse detalhe.

A apresentação de elementos em tela também pode ser produzida na área de ação do modo gráfico a partir do uso do comando **ROTULE**. No entanto, é importante considerar que o comando escreve na direção da tartaruga e não na direção da linha. Para ver o texto linearmente é ideal executar antes um giro para o **LESTE**. Observe as instruções seguintes:

**ROTULE [Linguagem Logo\n]**

A figura 2.8 mostra o resultado da apresentação da frase "Linguagem Logo" dentro da área de ação do modo gráfico.

Até este ponto todos as instruções estabelecidas foram executadas pelo Logo. Mas, o que acontece se o Logo não conseguir executar uma instrução. Para fazer um teste escreva no campo de entrada de comandos, dados e instruções a palavra **QUADRADO** e acione <Enter>. Veja a apresentação da mensagem de erro "**não sei como fazer QUADRADO**" em tom vermelho na área de ação do modo texto como indica a figura 2.9.

Observe que Logo disse-lhe "*não sei como fazer*". O que isso significa? Significa que Logo ainda não aprendeu a fazer o que lhe foi pedido, apesar de você ter desenhado um quadrado, Logo não sabe o que é, de fato, um quadrado.

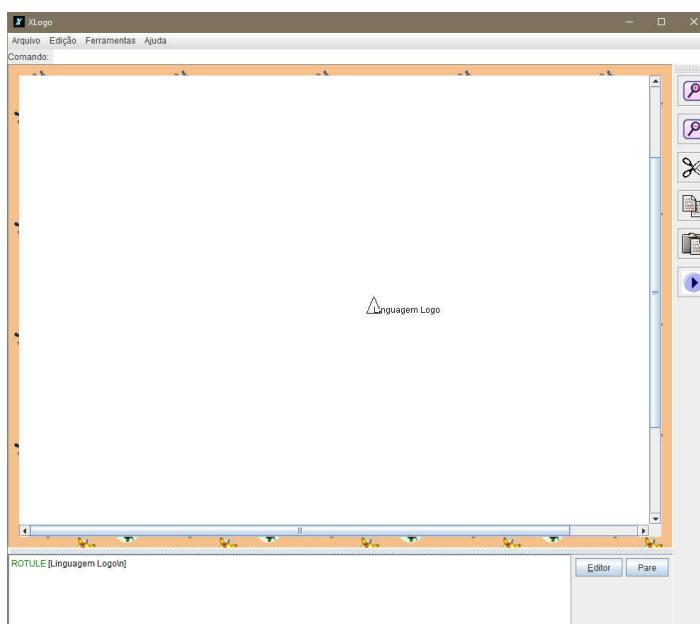


Figura 2.8 - Apresentação de mensagem na área gráfica.



Figura 2.9 - Apresentação de mensagem de erro.

Mas, nem tudo é perdido, pois ao dizer que ainda não sabe fazer, Logo abre espaço para que você ensine a tartaruga a fazer um quadrado. Este é um assunto que será visto no próximo capítulo.

### 2.3 - Primitivas complementares

Além do que foi apresentado, há outras ações básicas a serem conhecidas que ajudarão você a fazer diversas operações.

Quando a tartaruga anda, ela por padrão desenha, pois carrega com ela um lápis posicionado sobre o ambiente de trabalho. Mas nem sempre se deseja que a tartaruga ande desenhando. Por vezes é interessante que a tartaruga ande sem desenhar.

Para andar sem desenhar, ou seja, com o lápis erguido é necessário antes do movimento pedir a execução da instrução:

#### **USENADA**

Para voltar a desenhar basta usar a instrução:

#### **USELÁPIS**

A fim de demonstrar o uso dos comandos **USENADA (UN)** e **USELÁPIS (UL)** considere as seguintes instruções:

**LIMPETEXTO**

**LIMPEDESENHO**

**PARAFRENTE 40**

**PARAESQUERDA 90**

**USENADA**

**PARAFRENTE 40**

**PARADIREITA 90**

**USELÁPIS**

**PARAFRENTE 40**

Após executar as instruções anteriores ter-se-á na figura 2.10 a imagem de duas linhas verticais deslocadas.

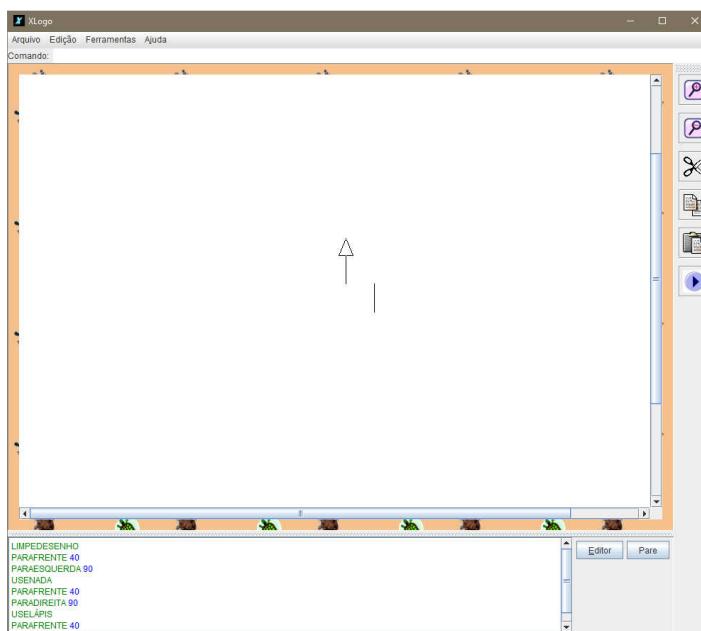


Figura 2.10 - Apresentação de linhas verticais deslocadas.

Caso queira apagar um traço desenhado, por acidente, você pode usar a instrução:

#### **USEBORRACHA**

A primitiva **USEBORRACHA** após seu uso precisa ser guardada. Para tanto, é necessário voltar ao modo de tracejamento. Neste caso, após **USEBORRACHA** use a primitiva **USELÁPIS (UL)** ou a primitiva **USERISCO**.

A partir das primitivas apresentadas você tem em mãos algumas ferramentas básicas para o desenvolvimento de diversas operações e a possibilidade de desenhar diversas formas. Mas antes de sair desenhando é importante ter noção da dimensão, do tamanho, do universo Logo.

Das diretivas apresentadas há alguns comandos que podem ser usados de forma simplificada a partir de siglas de identificação, seus mnemônicos.

É importante considerar que nem todos os comandos possuem este recurso siglas de simplificação ou identificação. Veja na tabela 2.1 a indicação de alguns dos comandos apresentados nesta obra que possuem siglas de identificação.

Mas tenha cuidado no uso de primitivas a partir das siglas de simplificação. Somente as use quando você tiver certeza absoluta de seu significado.

COMANDO	SIGLA
PARAFRENTE	PF
PARATRÁS	PT
PARADIREITA	PD
PARAESQUERDA	PE
ESCONDETAT	DT
MOSTRETAT	AT
LIMPEDESENHO	LD
LIMPETEXTO	LT
ESCREVA	ESC
USENADA	UN
USELÁPIS	UL
USEBORRACHA	UB

Tabela 2.1 - Primitivas simplificadas.

## CAPÍTULO 3 - Ações especializadas

As operações em Logo, no que tange, ao universo da *geometria da tartaruga* vão além das primitivas apresentadas. Neste contexto, desenhar um quadrado ou triângulo não é difícil apesar de maçante, mas desenhar figuras geométricas com maior número de lados poderá se tornar inviável. É neste sentido que entram outras primitivas de apoio especializadas em facilitar o uso de certos recursos da linguagem, as quais são apresentadas ao longo deste capítulo.

### 3.1 - Repetições

O desenho de um quadrado pode ser produzido com uso básico da combinação das primitivas **PARAFRENTE**, **PARATRÁS**, **PARADIREITA** ou **PARAESQUERDA** repetidos quatro vezes. A combinação de uso das primitivas é de cunho pessoal ou da necessidade do que se deseja efetivamente fazer.

Para facilitar repetições Logo possui uma primitiva chamada **REPITA** com a finalidade de repetir um grupo de instruções definidas entre colchetes um certo número de vezes. O comando (primitiva) **REPITA** para ser usado deve seguir a seguinte estrutura sintática:

**REPITA <n> [<instruções>]**

Onde, o indicativo "**n**" refere-se a quantidade de repetições e o indicativo "**instruções**" refere-se as ações que se deseja executar. Para desenhar um quadrado, por exemplo, use a instrução:

**REPITA 4 [PF 80 PD 90]**

A figura 3.1 mostra a apresentação de um quadrado desenhado a partir do uso da instrução "**REPITA 4 [PF 80 PD 90]**" formada pelas primitivas **REPITA**, **PF** e **PD**.

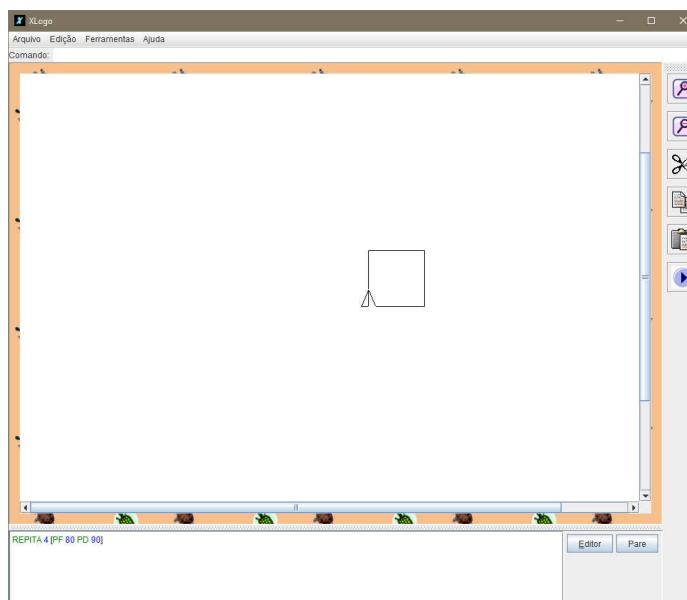


Figura 3.1 - Quadrado apresentado a partir do uso do comando "REPITA".

Veja, por exemplo, a apresentação de um triângulo equilátero com o uso do comando **REPITA**:

LD

REPITA 3 [PF 80 PD 120]

A figura 3.2 mostra a apresentação de um triângulo equilátero desenhado a partir do uso do comando REPITA.

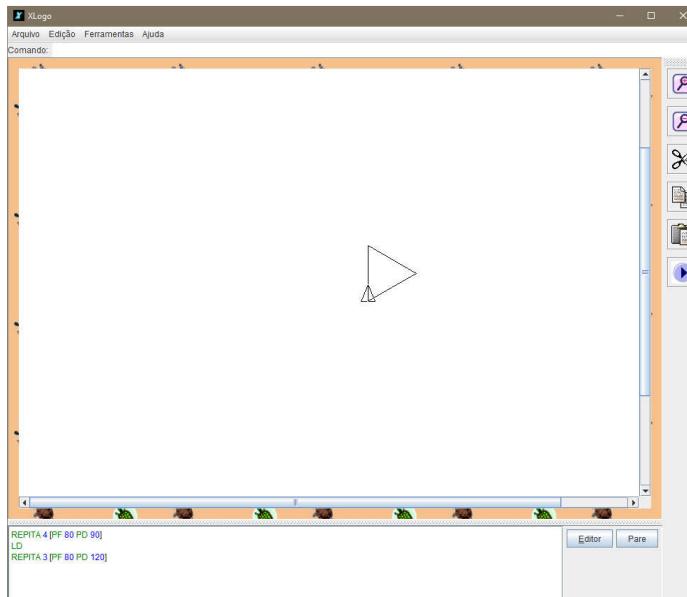


Figura 3.2 - Triângulo equilátero apresentado a partir do uso do comando "REPITA".

Note que para desenhar um triângulo equilátero usou-se a medida de graus como **120**. No ambiente Logo usa-se a medida externa da figura. Se fosse usado a medida interna o valor para um triângulo equilátero seria **60**.

Agora veja o desenho de um pentágono. Assim sendo, execute a instrução:

LD

REPITA 5 [PF 80 PD 72]

A figura 3.3 mostra o resultado obtido.

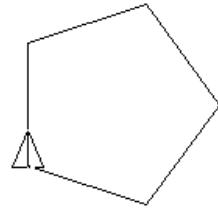


Figura 3.3 - Pentágono.

Agora pode estar em sua mente a pergunta que não quer calar. *Como saber exatamente o valor dos graus a ser usado para a definição de certa figura geométrica?* Recorde que os comandos PARADIREITA (PD) e PARAESQUERDA (PE) podem ser usados com valores entre **0** e **360**. No universo Logo a menor figura geométrica possui três lados, que é um triângulo, por sua vez a maior figura geométrica é a circunferência com trezentos e sessenta lados.

Veja a instrução:

LD

REPITA 360 [PF 1 PD 1]

A figura 3.3 mostra o resultado obtido.

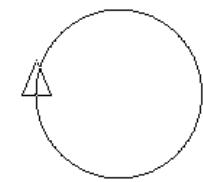


Figura 3.4 - Circunferência.

A resposta à pergunta é que basta você pegar o número de lados desejados e dividi-los por **360** para obter o valor exato de graus a ser usado para desenhar determinada figura geométrica.

A partir dessas orientações fica fácil desenhar qualquer figura geométrica plana compreendida entre **3** e **360** lados, ou seja, desenhar um polígono. Como ilustração veja a tabela 3.1 com os nomes de alguns polígonos, seus respectivos lados e o valor calculado dos ângulos de rotação.

POLÍGONO	LADOS	360 / LADOS	ÂNGULO
TRIÂNGULO	3	360 / 3	120.00
QUADRILÁTERO	4	360 / 4	90.00
PENTÁGONO	5	360 / 5	72.00
HEXÁGONO	6	360 / 6	60.00
HEPTÁGONO	7	360 / 7	51.43
OCTÓGONO	8	360 / 8	45.00
ENEÁGONO	9	360 / 9	40.00
DECÁGONO	10	360 / 10	36.00
UNDECÁGONO	11	360 / 11	32.73
DODECÁGONO	12	360 / 12	30.00
TRIDECA GONO	13	360 / 13	27.69
TETRADECÁGONO	14	360 / 14	25.71
PENTADECÁGONO	15	360 / 15	24.00
HEXADECÁGONO	16	360 / 16	22.50
HEPTADECÁGONO	17	360 / 17	21.18
OCTODECÁGONO	18	360 / 18	20.00
ENEADECÁGONO	19	360 / 19	18.95
ICOSÁGONO	20	360 / 20	18.00
TRIACONTÁGONO	30	360 / 30	12.00
TETRACONTÁGONO	40	360 / 40	9.00
PENTACONTÁGONO	50	360 / 50	7.20
HEXACONTÁGONO	60	360 / 60	6.00
HEPTACONTÁGONO	70	360 / 70	5.14
OCTOCONTÁGONO	80	360 / 80	4.50
ENEACONTÁGONO	90	360 / 90	4.00
HECTÁGONO	100	360 / 100	3.60
CIRCUNFERÊNCIA	360	360 / 360	1.00

Tabela 3.1 - Algumas medidas de ângulos.

Além da produção de figuras geométricas planas há outras possibilidades de geração de imagens na linguagem. Mas este é um assunto a ser visto um pouco mais adiante.

### 3.2 - Procedimentos

Já foi comentado que o fato de Logo apresentar uma figura não significa em absoluto que Logo saiba fazer a figura. Para que Logo aprenda a fazer uma figura é necessário que você ensine Logo a fazer.

A ideia de "ensinar" um computador a realizar certa tarefa, de modo que o computador aprenda e tenha esse "conhecimento" armazenado para uso futuro chamasse comumente de "inteligência artificial".

A "inteligência artificial" é a junção de duas palavras distintas com significados absolutos: *inteligência* é a faculdade de conhecer, compreender e aprender a resolver problemas e de adaptá-los a novas situações e *artificial* significa aquilo que não revela naturalidade, sendo produzido pelo ser humano e não pela natureza. Desta forma, pode-se entender que a "inteligência artificial" é a fabricação de conhecimentos específicos processados por computadores e controlados dentro das bases da Ciência da Computação.

Uma forma de "ensinar" a linguagem Logo a realizar tarefas de modo que se desenvolva a ideias de "inteligência artificial" é fazer uso de rotinas de scripts de programas chamadas de **procedimentos**.

Antes de proceder o desenvolvimento de procedimentos é importante definir uma configuração de uso de fontes de caracteres mono espaçadas. Desta forma, selecione a opção "**Preferências**" no menu "**Ferramentas**". Na caixa de diálogo apresentada selecione a guia "**Fonte**" e na lista de fontes selecione a fonte "**Consolas Bold**" definindo seu tamanho para "**18**". Acione o botão "**OK**" e observe que a aparência do ambiente mudou um pouco.

A criação de procedimentos em *Logo* é produzida com o uso do comando **APRENDA** a partir da seguinte sintaxe:

**APRENDA <nome>**

Onde, o indicativo "**nome**" refere-se a definição de um nome de identificação para o procedimento a ser usado no campo de comandos, dados e instruções como se fosse uma primitiva da linguagem. Note que se tem duas categorias de primitivas na linguagem Logo: as primitivas internas (pertencente a linguagem) e as primitivas externas (definidas por humanos) para adaptar a linguagem Logo a necessidades particulares.

Cada procedimento criado é a definição de um grau de conhecimento que o ambiente Logo pode artificialmente obter dando-lhe um nível de inteligência, além do que foi projetado. Assim sendo, veja a definição de um procedimento que desenhe um quadrado chamado "**QUAD1**". Para tanto, execute a instrução:

**APRENDA QUAD1**

Ao ser executada a instrução anterior é apresentada o modo **Editor** com o nome do procedimento em edição como apresenta a figura 3.5, onde o código de operação desejado deve ser informado.

Neste momento, escreva o que deve ser executado dentro do modo **Editor**. Por exemplo, coloque entre as primitivas **APRENDA** e **FIM** a instrução **REPITA 4 [PF 80 PD 90]** a partir de dois espaços em branco como mostrado na figura 3.6.

Para finalizar a edição acione o botão "**Guardar e sair do editor**" (primeiro botão da barra de ferramentas do modo **Editor** da esquerda para à direita - ícone TARTARUGA). Se acionado o botão "**Sair do editor sem guardar**" (segundo botão da barra de ferramentas da esquerda para à direita - ícone de placa de trânsito inglesa de proibido seguir em frente) a edição do procedimento não será registrada na memória.

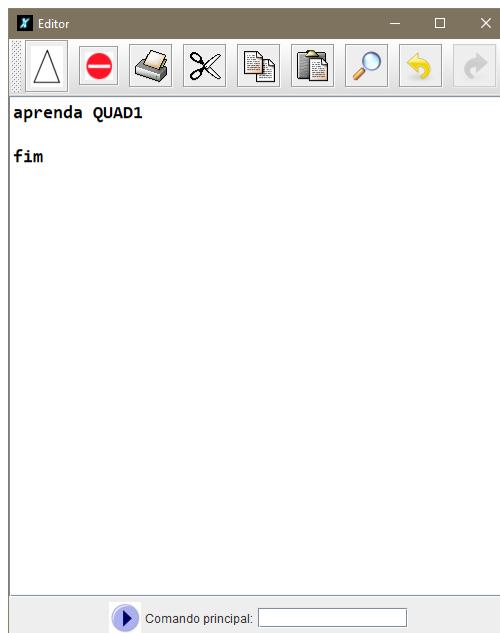


Figura 3.5 - Modo editor para definição de procedimento.

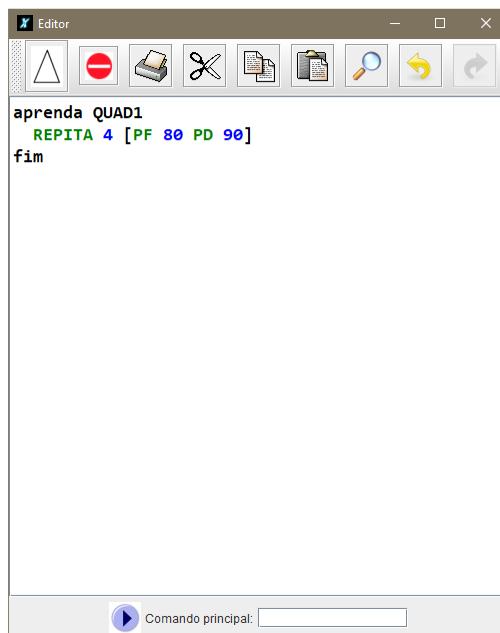


Figura 3.6 - Modo editor com procedimento definido.

Assim que o procedimento é devidamente registrado em memória ocorre a apresentação na área de ação do modo texto da mensagem "**Você definiu quad1.**".

A partir deste instante o Logo sabe desenhar um quadrado por meio da primitiva derivada chamada "**QUAD1**". Assim sendo, no capo de comandos, dados e instruções execute a instrução:

**QUAD1**

A figura 3.7 mostra o resultado obtido a partir da definição interna do conhecimento do que é um quadrado para a linguagem Logo.

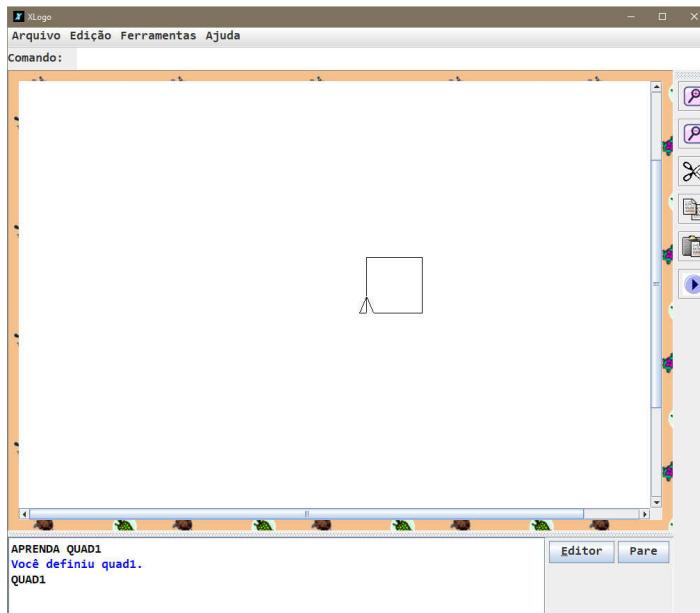


Figura 3.7 - Quadrado desenhado a partir da definição de um conhecimento artificial.

Assim como é possível repetir as primitivas internas também é possível repetir os procedimentos (que são as rotinas definidas a partir de ações derivadas). Por exemplo, imagine querer repetir o procedimento "QUAD1" por quatro vezes com ângulo de giro de "90" graus para a esquerda. Assim sendo, execute a instrução:

```
LD
LT
REPITA 4 [QUAD1 PE 90]
```

A figura 3.8 mostra o resultado desta operação.

Dependendo do valor de ângulo de giro é possível formar imagens geométricas muito diferentes como um conjunto de quadrados intercalados a partir da execução da instrução:

```
LD
LT
REPITA 8 [QUAD1 PE 45]
```

A figura 3.9 mostra o resultado desta operação.

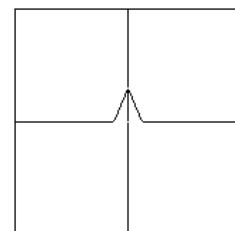


Figura 3.8 - Quadrado repetido.

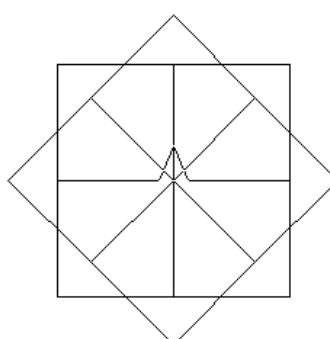


Figura 3.9 - Quadrados intercalados.

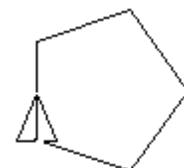
A forma apresentada de definição de procedimento é uma das maneiras possíveis de uso deste recurso. Outro jeito é fazer uso do botão "**Editor**".

Após acionar o botão "**Editor**" ocorre a abertura do modo editor com o código anteriormente definido. Neste instante acione uma vez a tecla "**<Enter>**" para pular uma linha em branco. Em seguida escreva o procedimento:

```
APRENDA PENTA1
REPITA 5 [PF 50 PD 360 / 5]
FIM
```

Para finalizar a edição acione o botão "**Guardar e sair do editor**" e em seguida execute as instruções:

```
LD
LT
PENTA1
```



**Figura 3.10 - Pentágono.**

A figura 3.10 mostra o resultado desta operação.

A partir da definição da imagem de um pentágono veja os dois exemplos de imagens geradas a partir do procedimento "**PENTA1**".

Execute primeiro a instrução:

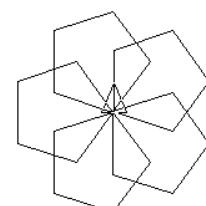
```
LD
LT
REPITA 5 [PENTA1 PD 72]
```

A figura 3.11 mostra o resultado desta operação.

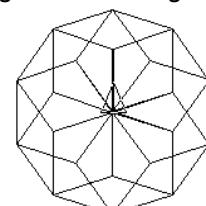
Depois execute a instrução:

```
LD
LT
REPITA 10 [PENTA1 PD 36]
```

A figura 3.12 mostra o resultado desta operação.



**Figura 3.11 - Pentágono 1.**



**Figura 3.12 - Pentágono 2.**

### 3.3 - Sub-rotinas

A definição de procedimentos trás para a linguagem Logo a possibilidade de se fazer a criação de comandos derivados. Todo comando derivado é baseado sobre o uso de alguma primitiva padrão da linguagem.

O fato que se definir procedimentos e dar-lhes a mesma capacidade que possuem os comandos traz muita mobilidade, pois um procedimento pode ser usado da mesma forma que um comando é usado no estabelecimento de instruções. Desta forma, torna-se naturalmente possível usar um procedimento dentro de outro procedimento, o que caracteriza a existência de sub-rotinas.

Para experimentar esta ideia considere realizar o desenho de uma flor com oito pétalas. Veja que uma flor, é grosso modo, um conjunto de pétalas. Uma pétala por sua vez pode ser a junção de duas meias circunferências.

Veja pelo que é descrito que o desenho de uma flor será realizado a partir de três procedimentos interligados. Assim sendo, considere os procedimentos "**MEIOCIRC**", "**PETALA**" e "**FLOR**". Para definir esses procedimentos acione o botão "**Editor**" e acrescente os procedimentos a seguir:

APRENDA MEIOCIRC

REPITA 90 [PF 1 PD 1]

FIM

APRENDA PETALA

MEIOCIRC

PD 90

MEIOCIRC

FIM

APRENDA FLOR

LD

REPITA 8 [PETALA PD 45]

FIM

A figura 3.13 mostra o conjunto de procedimentos definidos no modo **Editor**. Note que o procedimento "**FLOR**" faz uso do procedimento "**PETALA**" por oito vezes, o procedimento "**PETALA**" faz uso do procedimento "**MEIOCIRC**" por duas vezes.

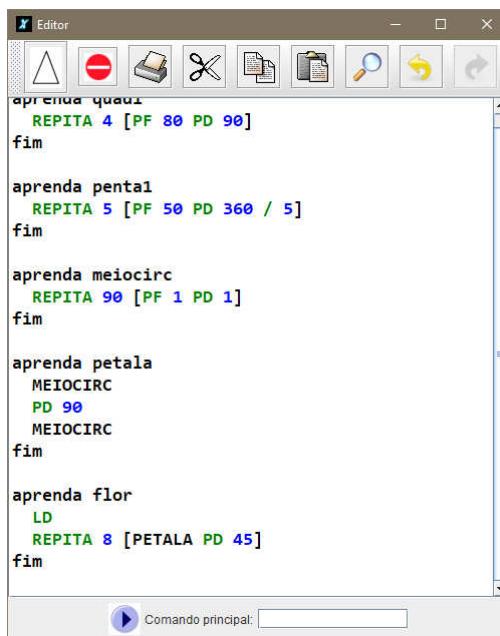


Figura 3.13 - Programa "Editor" com o conjunto de sub-rotinas.

Veja que o efeito em cascata para se fazer o desenho da flor é o que se chama de sub-rotinas, ou seja, "**MEIOCIRC**" é uma sub-rotina usada por "**PETALA**" que, por sua vez, é uma sub-rotina usada por "**FLOR**".

Para encerrar o modo **Editor** acione o botão "**Guardar e sair do editor**" e em seguida execute a instrução:

FLOR

A figura 3.14 mostra o resultado da execução do procedimento "**FLOR**".

Imagine desenhar uma flor mais encorpada com um número maior de pétalas, por exemplo vinte pétalas. Para tanto, execute a instrução:

LD

REPITA 20 [PETALA PD 36]

A Figura 3.15 mostra a imagem de uma flor com vinte pétalas.

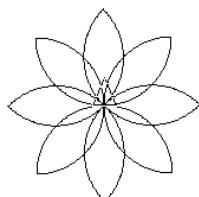


Figura 3.14 - Execução do procedimento "FLOR".

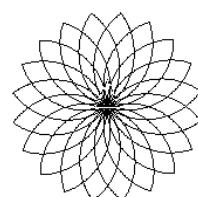


Figura 3.15 - Execução do procedimento "PETALA" para criar flor.

A partir da definição de sub-rotinas é possível diminuir muito a carga de trabalho, pois é possível criar comandos derivados e especializados em desenhar parte de uma imagem e formar, a partir daí, imagens com as combinações dos comandos definidos em conjunto com as primitivas e mesmo as funções da linguagem Logo.

### 3.4 - Variável

Você viu anteriormente o uso de instruções baseadas em primitivas simples e primitivas com o uso de parâmetros.

No tópico anterior você aprendeu a escrever seus próprios comandos a partir da definição de procedimentos. Os procedimentos, então criados, se assemelharam aos comandos simples da linguagem Logo. É chegada a hora de aprender a definir os próprios parâmetros para certo procedimento. É aqui que entra a ideia de uso de *variáveis*.

Uma *variável* do ponto de vista mais amplo para a computação é uma região de memória que armazena determinado valor por certo espaço de tempo. O valor armazenado em memória poderá ou não ser usado em ações de processamento.

Para criar variáveis em memória deve-se fazer uso do comando **ATRIBUA (ATR)** a partir da estrutura sintática:

**ATRIBUA <"variável> <valor>**

Onde, o indicativo "*variável*" refere-se ao nome de identificação da variável definido após o uso do símbolo de aspa inglesa ("") e "*valor*" a definição do valor associado ao nome da variável. Os caracteres maiúsculos e minúsculos interferem no nome de uma variável.

Para apresentar o conteúdo de uma variável é importante que seja usado antes do nome da variável o símbolo de dois pontos (:). Por exemplo, veja a definição e apresentação do conteúdo de uma variável chamada "**PAISES**" com o valor **Brasil, França e Argentina**.

```
ATRIBUA "PAISES [Brasil França Argentina\n]
ESCREVA :PAISES
```

A figura 3.16 mostra o resultado da ação anterior na área de ação do modo texto.

```
ATRIBUA "PAISES [Brasil França Argentina\n]
ESCREVA :PAISES
Brasil França Argentina
```

Figura 3.16 - Definição e apresentação de variável.

Veja outro exemplo de definição de variável e visualização do nome *Manzano*:

```
ATRIBUA "NOME "Manzano\n
ESCREVA :NOME
```

Perceba que para imprimir o conteúdo de uma variável usa-se o símbolo dois pontos. Cuidado em não fazer uso do símbolo aspa inglesa. Caso use o nome de uma variável com aspa inglesa a linguagem Logo não entenderá que se trata de uma variável e considerará como sendo apenas uma palavra a ser escrita usando o nome da variável.

Escreva a instrução:

```
ESCREVA :PAISES
```

E em seguida, escreva a instrução

```
ESCREVA "PAISES
```

Veja na figura 3.17 a diferença de resultado apresentado para cada uma das formas de acesso.

```
ATRIBUA "PAISES [Brasil França Argentina\n
ESCREVA :PAISES
Brasil França Argentina
ATRIBUA "NOME "Manzano\n
ESCREVA :NOME
Manzano
```

Figura 3.17 - Diferença no uso dos símbolos dois pontos e aspa inglesa.

No entanto, é possível fazer com que ocorra a apresentação do conteúdo de uma variável utilizando-se o símbolo aspa inglesa. Mas neste caso, é necessário fazer uso do comando **OBJETO** antes no nome da variável identificada com aspa inglesa. Observe a instrução seguinte:

```
ESCREVA OBJETO "PAISES
```

O efeito no uso do comando **OBJETO** antes no nome da variável com aspa inglesa, ou seja, a instrução **ESCREVA OBJETO "PAISES** é exatamente o mesmo resultado obtido a partir da execução da instrução **ESCREVA :PAISES**. Veja a figura 3.18.

```
ESCREVA OBJETO "PAISES
Brasil França Argentina
```

Figura 3.18 - Resultado da instrução "ESCREVA VALOR "PAISES".

A partir de uma visão geral da definição e uso de variáveis é possível criar um procedimento que, por exemplo, desenhe um quadrado em que o tamanho da figura será informado no uso e não dentro do procedimento. Assim sendo, acione o botão "**Editor**" e informe o procedimento "**QUAD2 :TAMANHO**".

```
APRENDA QUAD2 :TAMANHO
REPITA 4 [PF :TAMANHO PD 90]
FIM
```

Para encerrar o modo **Editor** acione o botão "**Guardar e sair do editor**" e em seguida execute as seguintes instruções:

```
LT LD
QUAD2 40
QUAD2 60
QUAD2 80
QUAD2 90
```

Perceba que para cada chamada do procedimento "QUAD2" foi usado em conjunto um parâmetro diferente proporcionando efeitos diferenciados como apresenta a figura 3.19.

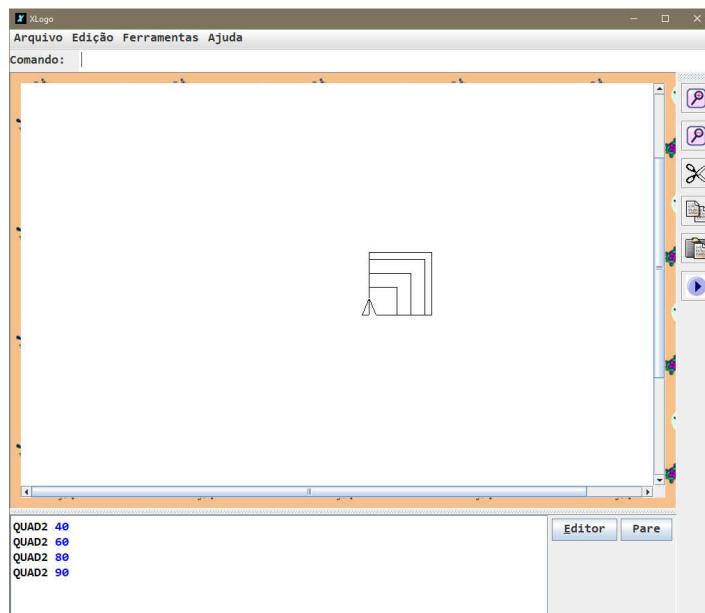


Figura 3.19 - Resultado a partir do uso de procedimento com parâmetro.

A partir desses recursos é possível criar um procedimento mais "inteligente" que a partir da informação do tamanho de traço e quantidade de lados desenhe qualquer figura entre "3" e "360" graus. Para tanto, acione o botão "Editor" e informe o seguinte código:

```
APRENDA POLIS :TAMANHO :LADO
REPITA :LADO [PF :TAMANHO PD 360 / :LADO]
FIM
```

Para encerrar o modo **Editor** acione o botão "**Guardar e sair do editor**" e em seguida execute as seguintes instruções:

A partir desta definição pode-se fazer uso, por exemplo, das seguintes instruções:

```
POLIS 80 3 - Desenha triângulo
POLIS 80 4 - Desenha quadrado
POLIS 80 5 - Desenha pentágono
POLIS 80 6 - Desenha hexágono
POLIS 1 360 - Desenha circunferência
```

A partir dessas orientações você já tem em mãos os recursos essenciais para a definição de parâmetros e de procedimentos. Assim sendo, já é possível criar comandos simples e comandos com parâmetros.

### 3.5 - Decisão

Logo é uma linguagem que possui internamente um nível de "inteligência" interessante. Entre as diversas possibilidades da linguagem há o comando **SE** que permite tomar decisões a partir do estabelecimento de duas formas de citação. Observe a seguinte estrutura sintática:

```
SE <condição> <[ação verdadeira]>
SE <condição> <[ação verdadeira]> <[ação falsa]>
```

Onde, o indicativo "**condição**" refere-se a definição de uma relação lógica que devolve como resposta valores **FALSO** ou **VERD**. Os indicativos "[**ação verdadeira**]" e "[**ação falsa**]" correspondem a definição de listas de ações que são disparadas dependendo do valor da condição. A primeira citação é usada para operar tomada de decisão simples e a segunda citação é usada para operar tomada de decisão composta. Para realizar a definição da condição usada com o comando **SE** é necessário levar em consideração o uso de operadores específicos para o estabelecimento das relações entre variáveis com variáveis ou de variáveis com constantes. A tabela 3.2 descreve o conjunto de *operadores relacionais* usados no estabelecimento de condições.

OPERADOR	SIGNIFICADO
=	IGUAL A
>	MAIOR QUE
<	MENOR QUE
>=	MAIOR OU IGUAL A
<=	MENOR OU IGUAL A
NÃO =	DIFERENTE DE

Tabela 3.2 - Operadores relacionais.

Para realizar um teste rápido sobre o uso de *operadores relacionais* execute as seguintes instruções observando a apresentação dos valores **FALSO** e **VERD**:

```
ESCREVA PALAVRA 1 = 1 "\n      (mostra: VERD)
ESCREVA PALAVRA NÃO 1 = 1 "\n (mostra: FALSO)
ESCREVA PALAVRA 1 < 2 "\n      (mostra: VERD)
ESCREVA PALAVRA 1 > 2 "\n      (mostra: FALSO)
ESCREVA PALAVRA 1 >= 1 "\n     (mostra: VERD)
ESCREVA PALAVRA 2 <= 2 "\n     (mostra: VERD)
```

Como exemplo no uso de tomada de decisão composta considere o desenvolvimento de um procedimento chamado "**MAXIMO**" que retorne o maior valor numérico a partir de dois valores fornecidos como parâmetro. Assim sendo, acione o botão "**Editor**" e informe o seguinte código:

```
APRENDA MAXIMO :A :B
  SE :A > :B [ESC :A] [ESC :B]
    ESC "\n
  FIM
```

Para encerrar o modo **Editor** acione o botão "**Guardar e sair do editor**" e em seguida execute as seguintes instruções:

```
MAXIMO 9 11
MAXIMO 15 8
```

Veja na figura 3.20 a apresentação do resultado das operações.

```
MAXIMO 9 11
11
MAXIMO 15 8
15
```

Figura 3.20 - Resultado de uma tomada decisão.

No sentido de demonstrar outro exemplo de tomada de decisão composta considere um procedimento chamado "**PAR**" que apresente a mensagem "**Ok**" se o valor numérico for par ou mostre a mensagem "**Erro**" caso o valor numérico não seja par. Para tanto, acione o botão "**Editor**" e informe o seguinte código:

```
APRENDA PAR :N
SE (RESTO :N 2) = 0 [ESCREVA [Ok]] [ESCREVA [Erro]]
ESC "\n"
FIM
```

Para encerrar o modo **Editor** acione o botão "**Guardar e sair do editor**" e em seguida execute as seguintes instruções:

```
PAR 2
PAR 3
```

Veja na figura 3.21 a apresentação do resultado das operações.

```
PAR 2
Ok
PAR 3
Erro
```

Figura 3.21 - Resultado do procedimento "PAR".

Observe a partir desses dois exemplos a definição da condição em vermelho. No procedimento "**MAXIMO**" tem-se a definição de uma condição simples de uma variável com outra variável, mas no procedimento "**PAR**" a condição é mais complexa, pois para saber se um valor é par é necessário validar o resto da divisão com a função **RESTO** e comparar este valor com a constante **0** (zero). Os trechos marcados em verde referem-se a ação se a condição for verdadeira e ocre se a condição for falsa.

Como exemplo de tomada de decisão simples considere o procedimento "**IMPAR**" que apresenta a mensagem "**Ok**" se o valor numérico for par e não mostre nada, caso contrário. Para tanto, acione o botão "**Editor**" e informe o seguinte código:

```
APRENDA IMPAR :N
SE NÃO (RESTO :N 2) = 0 [ESCREVA [Ok]]
ESC "\n"
FIM
```

Para encerrar o modo **Editor** acione o botão "**Guardar e sair do editor**" e em seguida execute as seguintes instruções:

```
IMPAR 3  
IMPAR 2
```

Veja na figura 3.222 a apresentação do resultado das operações.

```
IMPAR 3  
Ok  
IMPAR 2
```

Figura 3.22 - Resultado do procedimento "IMPAR".

A partir do conhecimento de tomada de decisão com o comando **SE** considere, como exemplo, um procedimento chamado "**FLORAL**" que aceite a entrada apenas de valores entre "1" e "7". Qualquer valor abaixo de "1" ou acima de "7" deve fazer com o procedimento mostre a mensagem de erro "**Use valores entre 1 e 7**". Assim sendo, acione o botão "**Editor**" e informe o seguinte código (não se preocupe com a primitiva **OU**, adiante a sua explicação):

```
APRENDA FLORAL :N  
SE OU :N < 1 :N > 7 [  
    ESCREVA [Use valores entre 1 e 7]  
][  
    REPITA 8 [  
        PD 45  
        REPITA :N [  
            REPITA 90 [  
                PF 2  
                PD 2  
            ]  
            PD 90  
        ]  
    ]  
FIM
```

O procedimento "**FLORAL**" é uma adaptação de exemplo Logo encontrado num material de aula do Professor Carlos Eduardo Aguiar do Centro de Educação Superior a Distância do Estado do Rio de Janeiro: <http://omnis.if.ufrj.br/~carlos/infoenci/notasdeaula/roteiros/aula10.pdf>.

Veja que no procedimento "**FLORAL**" a estrutura do código está sendo definida em outro estilo de escrita. Neste caso, baseando-a com o uso de endentações no sentido de indicar visivelmente quem está dentro de quem (observe as cores). Nada impede, por exemplo de escrever o procedimento "**FLORAL**" de outras maneiras, mas a forma apresenta é um estilo que deixa o emaranhado de instruções mais claras.

Para encerrar o modo **Editor** acione o botão "**Guardar e sair do editor**" e em seguida execute as seguintes instruções:

```
LD FLORAL 0
LD FLORAL 1
LD FLORAL 2
LD FLORAL 3
LD FLORAL 4
LD FLORAL 5
LD FLORAL 6
LD FLORAL 7
LD FLORAL 8
```

Veja na figura 3.23 a apresentação dos vários resultados obtidos a partir dos valores numéricos entre "1" e "7" válidas para o procedimento "FLORAL".

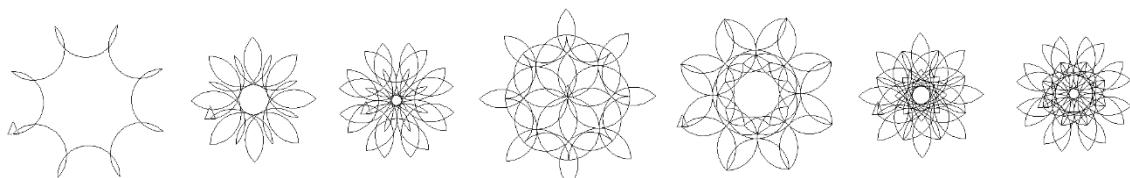


Figura 3.23 - Resultado do procedimento "FLORAL".

Além dos operadores relacionais são encontrados os operadores lógicos que auxiliam ações de tomada de decisão como é o caso do operador **OU** usado no procedimento "FLORAL". Além do operador **OU** existem em Logo os operadores lógicos **E** e **NÃO**. Atente para a tabela 3.3.

OPERADOR	SIGNIFICADO	RESULTADO
<b>E</b>	<b>CONJUNÇÃO</b>	<b>VERDADEIRO</b> quando todas as condições forem verdadeiras.
<b>OU</b>	<b>DISJUNÇÃO</b>	<b>VERDADEIRO</b> quando pelo menos uma das condições for verdadeira.
<b>NÃO</b>	<b>NEGAÇÃO</b>	<b>VERDADEIRO</b> quando condição for falsa e <b>FALSO</b> quando condição for verdadeira.

Tabela 3.3 - Operadores lógicos.

Os operadores lógicos **E** e **OU** permitem vincular em uma tomada de decisão duas ou mais condições. Já o operador lógico **NÃO** faz a inversão do resultado lógico da condição a sua frente. A ordem de prioridade entre os operadores lógicos é: **NÃO**, **E** e **OU**. Observe as tabelas verdadeas a seguir para cada um dos operadores lógicos (**E**, **OU** e **NÃO**) e confronte o que é apresentado com o resumo descrito na tabela 3.3.

#### *Operador de conjunção*

A conjunção é a relação lógica entre duas ou mais condições que gera resultado lógico verdadeiro quando todas as proposições forem verdadeiras. A tabela 3.4 indica os resultados lógicos que são obtidos a partir do uso do operador lógico de conjunção "**E**".

CONDIÇÃO 1	CONDIÇÃO 2	RESULTADO
VERDADEIRO	VERDADEIRO	VERDADEIRO
VERDADEIRO	FALSO	FALSO
FALSO	VERDADEIRO	FALSO
FALSO	FALSO	FALSO

Tabela 3.4 - Tabela verdade para operador "E".

Para demonstrar o uso do operador lógico de conjunção **E** considere um procedimento chamado "**TESTE\_E**" que receba um valor numérico e informe se este valor está ou não na faixa de "1" a "9". Desta forma, acione o botão "**Editor**" e informe o seguinte código:

```
APRENDA TESTE_E :N
SE E :N >= 1 :N <= 9 [
    ESC [Valor está na faixa de 1 a 9.\n]
][
    ESC [Valor não está na faixa de 1 a 9.\n]
]
FIM
```

Para encerrar o modo **Editor** acione o botão "**Guardar e sair do editor**" e em seguida execute as seguintes instruções:

```
TESTE_E 5
TESTE_E 0
TESTE_E 11
```

Procure fazer mais testes com outros valores.

#### *Operador de disjunção*

A disjunção é a relação lógica entre duas ou mais condições de tal modo que seu resultado lógico será verdadeiro quando pelo menos uma das proposições for verdadeira. A tabela 3.5 apresenta os resultados lógicos que são obtidos a partir do uso do operador lógico de conjunção "**OU**".

CONDIÇÃO 1	CONDIÇÃO 2	RESULTADO
VERDADEIRO	VERDADEIRO	VERDADEIRO
VERDADEIRO	FALSO	VERDADEIRO
FALSO	VERDADEIRO	VERDADEIRO
FALSO	FALSO	FALSO

Tabela 3.5 - Tabela verdade para operador "OU".

Para demonstrar o uso do operador lógico de disjunção **OU** considere um procedimento chamado "**TESTE\_OU**" que receba um valor textual "**FRIO**" ou "**QUENTE**" e informe respectivamente as mensagens "**Tempo ruim, proteja-se.**" ou "**Tempo bom, aproveite.**". Desta forma, acione o botão "**Editor**" e informe o seguinte código:

```
APRENDA TESTE_OU :TEMPO
SE OU :TEMPO = "FRIO :TEMPO = "CHUVOSO [
    ESC [Tempo ruim, proteja-se.\n]
][
    ESC [Tempo bom, aproveite.\n]
]
FIM
```

Para encerrar o modo **Editor** acione o botão "**Guardar e sair do editor**" e em seguida execute as seguintes instruções:

```
TESTE_OU "SOL  
TESTE_OU "FRIO  
TESTE_OU "QUENTE  
TESTE_OU "CHUVOSO
```

Os termos "**SOL**", "**FRIO**", "**QUENTE**" e "**CHUVOSO**" devem ser informados em formato maiúsculo.

#### *Operador de negação*

A negação é a rejeição ou a contradição do todo ou de parte de um todo. Pode ser a relação entre uma condição "**p**" e sua negação "**não-p**". Se "**p**" for verdadeira, "**não-p**" é falsa e se "**p**" for falsa, "**não-p**" é verdadeira. A tabela 3.6 mostra os resultados lógicos que são obtidos a partir do uso do operador lógico de negação "**NÃO**".

CONDIÇÃO	RESULTADO
VERDADEIRO	FALSO
FALSO	VERDADEIRO

Tabela 3.6 - Tabela verdade para operador "**NÃO**".

Para demonstrar o uso do operador lógico de negação **NÃO** leve em consideração um procedimento "**TESTE\_NAO**" que receba um valor numérico e apresente o quadrado deste valor caso este valor *não seja maior que "5"*. Assim sendo, acione o botão "**Editor**" e informe o seguinte código:

```
APRENDA TESTE_NAO :V  
SE NÃO :V > 5 [  
    ESC PALAVRA (POTÊNCIA :V 2) "\n"  
]  
FIM
```

Para encerrar o modo **Editor** acione o botão "**Guardar e sair do editor**" e em seguida execute as seguintes instruções:

```
TESTE_NAO 3  
TESTE_NAO 5  
TESTE_NAO 7
```

Ao ser executado o procedimento serão apresentados apenas resultados para entradas menores ou iguais a 5 (que são a forma de respeitar a condição *valores não maiores que 5*).

#### **3.6 - Recursão**

A ação de recursão é um recurso que na linguagem Logo permite a um procedimento chamar a si mesmo certo número de vezes. Veja que, um procedimento recursivo não pode chamar a si mesmo infinitamente, pois se assim o fizer entrará em um processo infinito de chamadas sucessivas podendo interromper a ação operacional do computador como um todo no momento em que os recursos de memória se tornarem esgotados. É importante que o procedimento recursivo tenha a definição de uma condição de encerramento (*aterramento*) controlada com o comando **SE** e com o auxílio do comando **PARE**.

A ação de aterramento deve ser definida antes da linha de código que efetua a recursão, pois ao contrário pode ocasionar o efeito colateral de execução infinita da recursividade inviabilizando seu uso (TOMIYAMA, 2016, p.2).

De modo prático um procedimento recursivo efetua ações de repetições semelhantemente as ações produzidas com o comando **REPITA**. A diferença é que o comando **REPITA** repete um trecho arbitrário de instruções e a recursão repete o próprio procedimento.

Para demonstrar um efeito de recursão considere um procedimento que produza a apresentação de imagens espiraladas. Vale ressaltar que uma espiral é o desenho de uma linha curva que se desenrola num plano de modo regular a partir de um ponto, dele gradualmente afastando-se. Desta forma, acione o botão "**Editor**" e informe o seguinte código:

```
APRENDA_ESPIRAL :TAMANHO :ANGULO :NUMERO
SE :NUMERO = 0 [
    PARE
]
PF :TAMANHO
PD :ANGULO
ESPIRAL (:TAMANHO + 8) :ANGULO (:NUMERO - 1)
FIM
```

Para encerrar o modo **Editor** acione o botão "**Guardar e sair do editor**" e em seguida execute separadamente as instruções:

```
LD ESPIRAL 4 122 60
LD ESPIRAL 4 92 50
```

Veja na figura 3.24 a apresentação do resultado das operações.

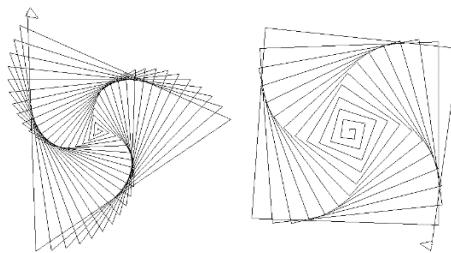


Figura 3.24 - Resultado do efeito recursivo sobre imagens espiraladas.

A partir do código do procedimento **ESPIRAL** note a definição em cor verde da decisão de aterrramento que detecta, neste caso, se o valor do parâmetro (variável) **:NUMERO** é igual a "0" (zero) e sendo faz a parada da execução do procedimento com o comando **PARE**. Note que se este trecho não estiver definido a função **ESPIRAL** chamará a si mesma sucessivamente.

Enquanto o valor da variável **:NUMERO** é diferente de zero ocorre a execução do procedimento que a cada chamada a si mesmo aumenta o valor de **:TAMANHO** em "8" (oito), mantém constante a distorção do valor de **:ANGULO** e diminui em "1" (um) o valor de **:NUMERO**. A cada chamada uma parte da imagem é desenhada com as instruções **PF :TAMANHO** e **PD :ANGULO** definidas na cor ocre. Veja também que a soma e subtração de valores sobre o valor inicial dos parâmetros tem que ser definida entre parênteses.

O efeito espiralado para a imagem de um triângulo é conseguido com o valor de giro de ângulo "**122**" para o parâmetro **:ANGULO** e a imagem do quadrado é conseguido com o valor "**92**".

Veja que esses valores são próximos ao valor real de giro para a apresentação das figuras geométricas planas.

O que aconteceria, por exemplo, se fossem usados os valores de ângulos "120" e "90" no procedimento "**ESPIRAL**"? Então, mãos à obra, execute separadamente as instruções:

```
LD ESPIRAL 4 120 60  
LD ESPIRAL 4 90 60
```

Veja os resultados na figura 3.25:

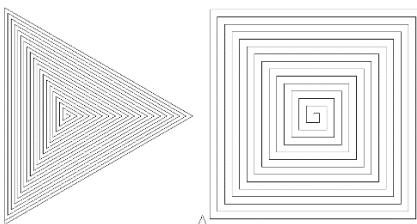


Figura 3.25 - Mais efeitos recursivos sobre imagens espiraladas.

No sentido de apresentar mais ações sobre recursões veja na próxima sequência de procedimentos os recursos para se desenhar uma treliça (unidade definida a partir de cinco triângulos) como base de sustentação para uma ponte (adaptado, MULLER, 1998 p. 345). Assim sendo, acione o botão "**Editor**" e informe o seguinte código:

```
APRENDA TRIANGULAR  
PF 40 PD 135  
PF 40 / RAIZQ 2 PD 90  
PF 40 / RAIZQ 2 PD 135  
FIM
```

```
APRENDA TRELICA :X  
SE :X = 0 [PARE]  
REPITA 4 [TRIANGULAR PF 40 PD 90]  
PD 90 PF 40 PE 90  
TRELICA :X - 1  
FIM
```

Para encerrar o modo **Editor** acione o botão "**Guardar e sair do editor**" e em seguida execute separadamente a instrução:

```
LD TRELICA 10
```

Veja na figura 3.26 a apresentação do resultado do efeito de recursividade.



Figura 3.26 - Resultado do efeito de recursividade do procedimento "TRELICA".

Perceba que foram apresentados até este momento duas formas de se fazer a repetições de trechos de códigos. Uma com o comando **REPITA** e a outra com o efeito de recursividade. Se a recursividade for implementada com o cuidado da definição da condição de aterramento será um mecanismo útil de repetição.

### 3.7 - Memorização e amnésia

Todo o desenvolvimento de procedimentos em prol da definição da estruturação da "inteligência artificial" da linguagem Logo para poder ter o efeito esperado precisa ser gravado na forma de arquivos. A gravação dos procedimentos criados permite estabelecer um recurso de "memorização", pois do contrário se nada for gravado quando o ambiente for executado posteriormente ao seu uso não se "lembra" de nada entrando em estado de "amnésia".

Um arquivo Logo gravado pode ser formado por um único procedimento ou a partir de um conjunto de procedimentos vinculados por meio de sub-rotinas ou não.

Durante os tópicos anteriores foram desenvolvidos alguns procedimentos, os quais devem ser gravados. Para tanto, selecione a opção "**Guardar como.....**" do menu "**Arquivo**". Será apresentada a caixa para gravação do arquivo de procedimento indicada na figura 3.27.

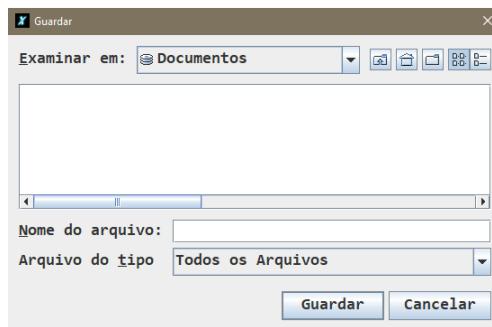


Figura 3.27 - Caixa para gravação de arquivos.

No campo "**Nome do arquivo**" informe um nome para a gravação do arquivo. Neste caso entre, por exemplo, o nome "**Cap03\_Aprendizagem**" e acione o botão **Guardar**.

Para realizar um teste, encerre o ambiente "xLogo" e em seguida carregue-o novamente para a memória. Assim que o ambiente for carregado tente executar o procedimento "**FLORAL 5**" e veja a apresentação da mensagem de erro "**não sei como fazer FLORAL**". Perceba que neste momento o ambiente Logo está em estado de "amnésia".

Para fazer *Logo* "lembra-se" do conhecimento que tem é necessário carregar para a memória o arquivo de procedimentos que se tenha anteriormente gravado. Para tanto, selecione a opção "**Carregar**" ou a opção "**Abrir**" do menu "**Arquivo**". Será apresentada a caixa para abertura do arquivo de procedimento que esteja gravado como mostra a figura 3.28.

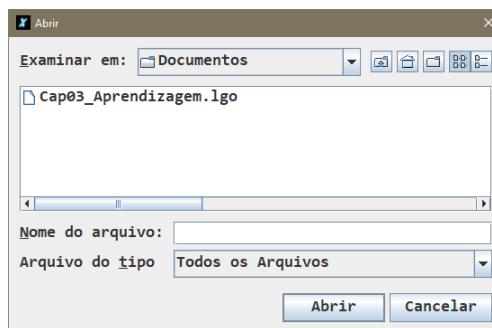


Figura 3.28 - Caixa para leitura de arquivos.

Selecione, com o ponteiro do mouse, o arquivo desejado, neste caso "**Cap03\_Aprendizagem**" e acione o botão **Abrir**. Será apresentada a tela do modo **Editor**. Neste momento, basta apenas acionar o botão "**Guardar e sair do editor**".

A partir deste instante, execute a chamada da instrução "**FLORAL 5**" e veja o resultado apresentado e observe que sempre será necessário gravar procedimentos que se deseja tê-los para uso futuro.

Neste momento, tanto o uso da opção "**Carregar**" ou da opção "**Abrir**" do menu "**Arquivo**" surte o mesmo efeito. Mas é importante saber a diferença entre essas duas formas.

A opção "**Abrir**" deve ser usada quando há a intenção de substituir um conhecimento de memória por outro conjunto de conhecimento, de modo que o conhecimento anterior seja totalmente esquecido, a menos que seja novamente carregado para a memória.

A opção "**Carregar**" deve ser usada quando há a intenção de implementar no conjunto de conhecimento existente o conhecimento definido em outro arquivo, de modo que se tenha em memória a soma dos conhecimentos vinculados.

No universo da "inteligência artificial" o conjunto de conhecimentos armazenados em um arquivo ou conjuntos de arquivos é também conhecido como *enciclopédia*.

### 3.8 - Exercícios de fixação

1. Quais são as figuras geométricas planas desenhadas a partir das seguintes instruções? Diga qual é a figura sem executar a instrução no ambiente de programação.

REPITA 4 [PARAFRENTE 100 PARADIREITA 90] \_\_\_\_\_

REPITA 5 [PF 100 PE 72] \_\_\_\_\_

REPITA 3 [PT 100 PD 120] \_\_\_\_\_

REPITA 36 [PF 20 PD 10] \_\_\_\_\_

2. Criar procedimento chamado **RETANGULO1** (sem acento) que desenhe um retângulo com lados de tamanhos **60** e **100** para frente com giro de gruas para à direita. O procedimento deve desenhar a imagem sem o uso do recurso **REPITA**.
3. Criar procedimento chamado **RETANGULO2** (sem acento) que desenhe um retângulo com lados de tamanhos **60** e **100** para trás com giro de gruas à esquerda. Usar a primitiva **REPITA**.
4. Criar, sem uso da primitiva **REPITA**, procedimento chamado **PENTAGONO** (sem acento) que construa um pentágono com tamanho **40**. Avance para frente com giro a direita.
5. Criar procedimento chamado **DECAGONO** (sem acento) que construa uma figura com dez lados a partir do uso da primitiva **REPITA** com giro de graus à esquerda com avanço a gente de **30**.

---

---

ANOTAÇÕES

---

---



## CAPÍTULO 4 - Ações específicas

Anteriormente foram apresentados alguns recursos da linguagem que proporcionam experiências interessantes, mas ainda limitadas. A linguagem Logo é mais poderosa do que realizar a apresentação de imagens geométricas planas. Veja neste capítulo alguns outros recursos da linguagem.

### 4.1 - Entrada de dados

Anteriormente foram apresentados recursos para a armazenamento básico de valores na forma de variáveis quando do uso de parâmetros e a apresentação escrita de resultados, principalmente na área de ação do modo texto a partir do campo de entrada de comandos, dados e instruções.

No entanto, *Logo* permite a entrada de dados em tempo de execução. Para isso deve-se fazer uso da primitiva **LEIA** que possui como estrutura sintática o estilo:

**LEIA <[mensagem]> <conteúdo> <"variável>**

Onde, o indicativo "<**mensagem**>" refere-se a uma mensagem de interação com o usuário e "**conteúdo**" refere-se ao dado informado na entrada. Para que o dado informado seja devidamente usado deve ser vinculado a uma "<"**variável**>".

O uso da primitiva **LEIA** deve seguir a instrução "**LEIA [Entre um valor:] "N"**" que quando executada apresenta uma caixa de diálogo para entrada de dados como indica a figura 4.1.



Figura 4.1 - Caixa de diálogo para entrada de dados.

Para fazer uma experiência no uso deste recurso considere o desenvolvimento de um procedimento chamado "**IDADE**" que solicite a entrada de uma idade, armazenando a idade na variável :**VALOR** e apresentando a mensagem "**Maior de idade**" se a idade for maior ou igual a "**18**" ou apresentando a mensagem "**Menor de idade**" caso a idade seja menor que "**18**". Desta forma, açãoe o botão "**Editor**" e informe o seguinte código:

**APRENDA IDADE**

```
LEIA [Informe sua idade:] "VALOR  
SE :VALOR >= 18 [ESC [Maior de idade]] [ESC [Menor de idade]]  
ESC "\n  
FIM
```

Para encerrar o modo **Editor** açãoe o botão "**Guardar e sair do editor**". Em seguida execute o procedimento "**IDADE**" algumas vezes e informe valores diferentes. Veja os resultados apresentados

Anteriormente foi apresentado o comando **REPITA** que tem por finalidade repetir um trecho de código demarcado dentro de colchetes um determinado número de vezes. Há um comando que pode mostrar o momento da contagem em que o comando **REPITA** se encontra. Conheça a primitiva **CONTEVEZES (CV)**.

Para mostrar o conteúdo do comando **CONTEVEZES** além do comando **ESCREVA** é necessário usar em conjunto o comando **SENTENÇA (SN)** que tem por finalidade criar uma lista de caracteres concatenados a ser apresentada pelo comando **ESCREVA**. Assim sendo, considere apresentar os valores gerados pela execução do comando **REPITA** durante a ação de **5** iterações. A cada contagem de **REPITA** o valor na memória deve ser resgatado e apresentado. Para tanto, execute a instrução:

```
REPITA 5 [ESC (SENTENÇA "Conta: CONTEVEZES "\n)]
```

Veja o resultado obtido junto a figura 4.2.

```
REPITA 5 [ESC (SENTENÇA "Conta: CONTEVEZES "\n)]
Conta: 1
Conta: 2
Conta: 3
Conta: 4
Conta: 5
```

Figura 4.2 - Resultado da ação do comando "CONTEVEZES" junto a "FRASE" com "ESC".

Veja que a cada execução do comando **REPITA** o comando **CONTEVEZES** detectou um valor da contagem. A cada contagem o valor de **CONTEVEZES** foi concatenado a palavra "**Conta:**" por meio do comando **SENTENÇA**. O resultado da tabuada a ser apresentado deve ser delimitado entre parênteses estabelecendo assim uma estrutura de lista. Veja que nesta versão ao invés de se usar a primitiva **ESCREVA** usa-se a primitiva **MOSTRE (MO)**. A diferença entre elas é que **MOSTRE** após apresentar o conteúdo salta automaticamente uma linha, não sendo necessário o uso do código "\n".

Agora, considere um procedimento chamado "**TABUADA**" que apresente a tabuada de um número qualquer entre "**2**" e "**9**". Para auxiliar esta operação entra em jogo a primitiva **CV** que tem por finalidade recuperar o momento de contagem do comando **REPITA**. Caso valores fora da faixa permitida sejam fornecidos o procedimento deve retornar a mensagem "**Valor fora da faixa**". Assim sendo, acione o botão "**Editor**" e informe o seguinte código:

#### APRENDA TABUADA

```
LEIA [Entre número para tabuada:] "NUMERO
SE E :NUMERO >= 2 :NUMERO <= 9 [
    REPITA 10 [MOSTRE (SN :NUMERO "X CV "= :NUMERO * CV)]
]
    ESC [Valor fora da faixa]
]
FIM
```

Para encerrar o modo **Editor** acione o botão "**Guardar e sair do editor**". Em seguida execute separadamente a instrução:

#### TABUADA

Quando apresentada a caixa de diálogo para entrada, forneça um valor numérico entre "**2**" e "**9**". Por exemplo, entre o valor "**7**" e veja a tabuada apresentada como indicado na figura 4.3.

```
TABUADA
Entre número para tabuada:
7
7 X 1 = 7
7 X 2 = 14
7 X 3 = 21
7 X 4 = 28
7 X 5 = 35
7 X 6 = 42
7 X 7 = 49
7 X 8 = 56
7 X 9 = 63
7 X 10 = 70
```

Figura 4.3 - Resultado de uma tabuada simples.

Perceba que Logo é uma linguagem que vai muito mais além do que simplesmente produzir lindas imagens e desenhos.

#### 4.2 - Randomização

Um recurso que pode ser explorado em computadores é a capacidade destes conseguirem sortear valores, principalmente numéricos, de forma aleatória. Valores aleatórios também são chamados de *valores randômicos*. Daí o termo *randomização*.

A linguagem Logo possui para auxiliar operações de sorteio o comando **SORTE** que é operado segundo a estrutura sintática:

**SORTEIE <número>**

Onde, o indicativo "**número**" refere-se ao valor máximo estabelecido para sorteio. Este valor poderá variar entre "0" (zero) e o valor estabelecido em "**número**".

Veja algumas instruções de sorteio entre os valores "0" e "100":

```
MOSTRE SORTEIE 100
```

Veja que a cada execução o valor sorteado retornado é diferente um do outro como comprova a figura 4.4.

```
MOSTRE SORTEIE 100
74
MOSTRE SORTEIE 100
35
MOSTRE SORTEIE 100
77
MOSTRE SORTEIE 100
75
MOSTRE SORTEIE 100
63
```

Figura 4.4 - Resultado do sorteio de valores entre "0" e "100".

A partir do recurso de randomização veja o procedimento "**MALUQUINHO**" que desenha figura geométrica desforme, pois avança para a direita aleatoriamente entre os valores "0" e "60" e gira para a esquerda aleatoriamente entre "0" e "360". Desta forma, acione o botão "**Editor**" e informe o seguinte código:

```
APRENDA MALUQUINHO  
REPITA 100 [  
    PF SORTEIE 60  
    PD SORTEIE 360  
]  
FIM
```

Para encerrar o modo **Editor** acione o botão "**Guardar e sair do editor**". Em seguida execute a instrução:

```
LD MALUQUINHO
```

Veja o resultado da operação na figura 4.5, ressaltando que em seu sistema a imagem tende a ser completamente diferente.

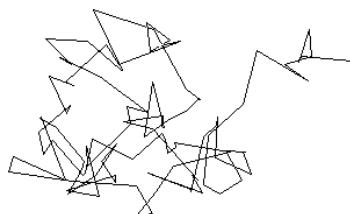


Figura 4.5 - Resultado de figura geométrica aleatória.

A primitiva **SORTEIE** sempre que é usado tende a gerar um valor aleatório diferente do anterior (pode ocorrer a repetição sequencial de dois valores, mas dificilmente três e quatro será virtualmente impossível), ou seja, se você carregar o ambiente "xLogo" na memória executar o comando **SORTEIE**, sair, carregar novamente o ambiente e executar a primitiva novamente terá valores sempre diferentes.

#### 4.3 - Coordenadas

Todos as figuras geométricas planas apresentadas até este ponto foram desenhadas a partir do ponto central da área de ação do modo gráfico, ou seja, das coordenadas "0" para X e "0" para Y. O ponto central pode-se ser chamado de *CASA* por ser o ponto de início das operações geométricas em Logo.

Para verificar a posição em que a tartaruga se encontra usa-se o comando **POS** em conjunto com o comando **MOSTRE**. No entanto, encontram-se outros comandos de operação para auxiliar mudanças da tartaruga, tais como: **MUDEX**, **MUDEY**, **MUDEXY** e **CENTRO**.

Quando mudanças de posição são feitas dentro da área de ação do modo gráfico o rastro da tartaruga tende a ser traçado. Isso faz com que seja necessário antes de proceder mudança de posição usar o comando **USENADA (UN)**. Caso deseje traçar na mudança de coordenada use o comando **USELÁPIS (UL)**.

Para realizar alguns testes de posicionamento e verificação de posicionamento acione primeiro as primitivas **LD LT** e em seguida execute as instruções:

```
UN  
MOSTRE POS  
MUDEX 50  
MUDEY -30  
MOSTRE POS
```

A figura 4.6 mostra os resultados apresentados. Veja os valores gerados após o uso da instrução **MOSTRE POS** na coordenada "0 0" com marcação em vermelho e na coordenada "50 -30" com marcação em azul.



Figura 4.6 - Identificando posições cartesianas.

Para os comandos **MUDEX** e **MUDEY** os valores usados devem estar limitados a dimensão da área de trabalho, ou seja, valores definidos entre "0" e "500". Veja que a partir da posição central (a *CASA*) é possível usar valores positivos para posicionar a tartaruga à direita ou acima da posição inicial ou usar valores negativos para posicionar a tartaruga à esquerda ou abaixo da posição inicial. A Figura 4.7 representa esquematicamente os movimentos de salto com o par de comandos **MUDEX 30** com **MUDEY 20** e **MUDEX -30** com **MUDEY -10**.

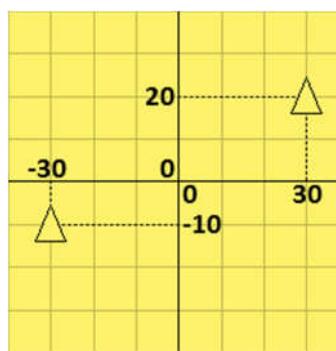


Figura 4.7 - Posições cartesianas da tartaruga.

O acesso as coordenadas **X** e **Y** podem ser realizadas a partir de uma única vez, por intermédio do comando **MUDEXY** e o retorno ao ponto central, a *CASA*, pode ser conseguido com o comando **CENTRO**. Teste as seguintes instruções:

```

LT
LD
UN
MOSTRE POS
MUDEXY MENOS 50 25
MOSTRE POS
CENTRO
MOSTRE POS
  
```

Na sequência execute o comando **UL**. A figura 4.8 apresenta os valores das posições cartesianas após o uso dos comandos **MUDEXY** e **CENTRO**.

Observe que a instrução "**MUDEXY MENOS 50 25**" faz uso da função **MENOS** para expressar um valor negativo. O mesmo poderia ser feito com a instrução "**MUDEXY -50 25**", mas se a instrução fosse "**MUDEXY 50 -25**" ocorreria a apresentação da mensagem de erro "**Faltam dados para MUDEXY**", pois o ambiente *xLogo* realizaria a subtração de "**50 -25**". Neste caso, quando houver mais de um parâmetro e a partir do segundo parâmetro haja a necessidade de se fazer uso de

valores numéricos negativos é indicado usar a função **MENOS**. Desta forma, a ação da instrução "**MUDEXY 50 -25**" deve ser escrita como "**MUDEXY 50 MENOS 25**".

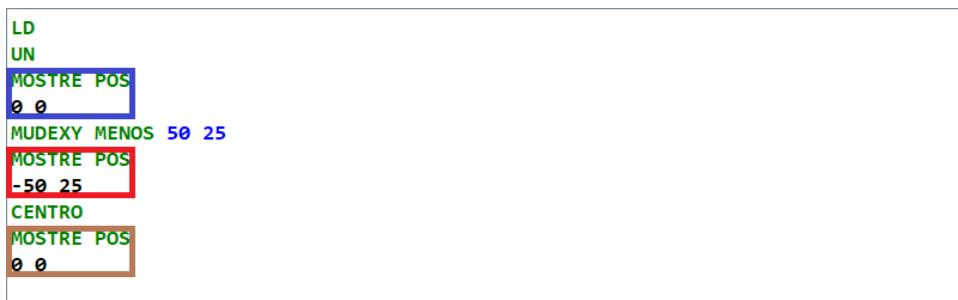


Figura 4.8 - Posições cartesianas após comandos "MUDEXY" e "CENTRO".

A partir desses "novos" recursos e de alguns recursos conhecidos é possível extrapolar e criar desenhos e imagens mais elaboradas, como por exemplo a figura da face quadrada indicada junto a figura 4.9, adaptada de Harvey (1997, p. 184).

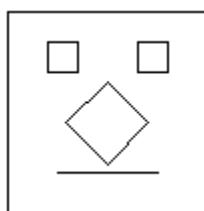


Figura 4.9 - Face quadrada.

Apesar de simples a imagem da figura 4.9 apresenta um grau de complexidade interessante, pois diversas ações são necessárias. Desta forma, acione o botão "**Editor**" e informe os seguintes códigos:

```

APRENDA CABECA
UL REPITA 4 [PF 100 PD 90] UN
FIM
  
```

```

APRENDA PREPARA.OLHO
PE 90 PF 65 PD 90 PF 20
FIM
  
```

```

APRENDA OLHO.DIREITO
UL OLHO
UN PT 15
FIM
  
```

```

APRENDA BOCA
UN
PF 20 PD 90 PF 25
UL
PF 50
UN
PT 75
FIM
  
```

```

APRENDA OLHO
REPITA 4 [PF 15 PD 90]
FIM
  
```

```

APRENDA OLHO.ESQUERDO
UL OLHO
UN PF 45
FIM
  
```

```

APRENDA NARIZ
PD 90 PF 20 PE 45
UL REPITA 4 [PF 29 PD 90]
UN CENTRO
PE 135
UL
FIM
  
```

```

APRENDA FACE
CABECA
BOCA
PREPARA.OLHO
OLHO.ESQUERDO
OLHO.DIREITO
NARIZ
FIM
  
```

Para encerrar o modo **Editor** acione o botão "**Guardar e sair do editor**". Em seguida execute separadamente as instruções após fazer uso do comando **LD**:

**DT**  
**FACE**

Após executar o procedimento e visualizar a imagem da face quadrada execute o comando **AT** para reapresentar a tartaruga na área de ação do modo gráfico. Lembre-se que os comandos **DT** e **AT** são respectivamente as siglas dos comandos **ESCONDETAT** e **MOSTRETAT**.

Às vezes é interessante fazer com que o ambiente Logo opere em velocidade menor do que trabalha, pois muitas vezes é interessante ver o desenho sendo vagarosamente formado. Para este tipo de necessidade há o comando **ESPERE** que possui a estrutura sintática:

**ESPERE <tempo>**

Onde, o indicativo "**tempo**" refere-se a um valor inteiro que determina valor em segundos. Por exemplo se quiser esperar *um segundo* use o valor "**60**", para *meio segundo* use "**30**" e assim sucessivamente. Veja então um exemplo mais sofisticado de criação de figura utilizando-se alguns dos comandos conhecidos. Atente para o código do procedimento **POLIGOM** operado respectivamente a partir dos parâmetros **:LADOS** e **:TAMANHO**. Atente para o uso do comando **ESPERE**. Desta forma, acione o botão "**Editor**" e informe o seguinte código:

**APRENDA POLIGOM :LADOS :TAMANHO**  
**DT**  
**SE :LADOS = 30 [PARE]**  
**USELÁPIS**  
**REPITA :LADOS [PF :TAMANHO PD 360 / :LADOS]**  
**USENADA**  
**CENTRO**  
**USELÁPIS**  
**REPITA :LADOS [PF :TAMANHO PE 360 / :LADOS]**  
**ESPERE 30**  
**POLIGOM (:LADOS + 1) :TAMANHO**  
**AT**  
**FIM**

Para encerrar o modo **Editor** acione o botão "**Guardar e sair do editor**". Em seguida execute separadamente as instruções após fazer uso do comando **LD**:

**POLIGOM 3 50**

Veja o resultado da operação na figura 4.10. Atente para o efeito espelho conseguido com o posicionamento ao centro após o desenho do lado direito para então desenhar o lado esquerdo. Atente para os movimentos dos efeitos dos comandos **USENADA** e **USELÁPIS** entre as imagens direita e esquerda.

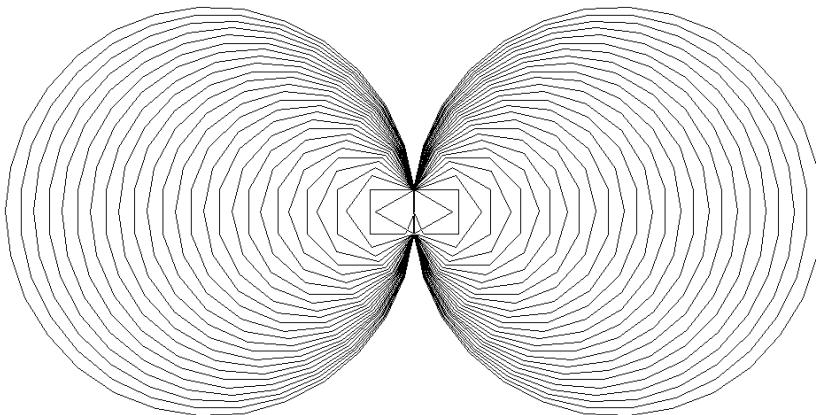


Figura 4.10 - Polígono com efeito de espelhamento.

#### 4.4 - Cores

Os desenhos produzidos até este momento foram riscados em cor preta em um fundo branco. A linguagem Logo permite manipular o conjunto de cores do risco do desenho e do fundo da tela.

O uso de cores é obtido a partir da combinação do padrão de cores básicas, denominado **RGB** (**Red**, **Green** e **Blue**), onde cada cor a ser usada é a combinação do espectro de cores formado a partir das cores básicas definidas por meio de valores entre "0" e "255".

O sistema de cores **RGB** pode não ser intuitivo, mas é um sistema que possibilita a geração de **16.777.216** tons de cores diferentes, considerando-se os níveis de brilho e saturação dessas cores, mesmo que não seja possível ao olho humano distinguir tal espectro de cores. A tabela 4.1 mostra dezessete tonalidades de cores básicas que podem ser usadas no *xLogo*.

ÍNDICE	COR	CÓDIGO RGB
0	PRETO	[000 000 000]
1	VERMELHO	[255 000 000]
2	VERDE	[000 255 000]
3	AMARELO	[255 255 000]
4	AZUL	[000 000 255]
5	MAGENTA	[255 000 255]
6	CIANO	[000 255 255]
7	BRANCO	[255 255 255]
8	CINZA	[128 128 128]
9	CINZA CLARO	[192 192 192]
10	VERMELHO ESCURO	[128 000 000]
11	VERDE ESCURO	[000 128 000]
12	AZUL ESCURO	[000 000 128]
13	LARANJA	[255 200 000]
14	ROSA	[255 175 175]
15	VIOLETA	[128 000 255]
16	MARROM	[153 102 000]

Tabela 4.1 - Índice de cores do ambiente *xLogo*.

Os valores das cores grafados entre colchetes na coluna "CÓDIGO RGB" da tabela 4.1 segue o formato posicional [R G B], em que cada componente é um valor numérico entre "0" e "255".

A mudança de cores de fundo e do traço a ser riscado se faz com os comandos **MUDECF** (mude cor do fundo) e **MUDECL** (mude cor do lápis) com a indicação do padrão **RGB** da cor desejado ou pelo índice da cor indicado na tabela 4.1.

Para formatar com código **RGB** o fundo na cor preta e o lápis na cor amarelo use as instruções:

```
MUDECF [000 000 000]
MUDECL [255 255 000]
```

Para ver o resultado execute a instrução:

```
LD FACE
```

A figura 4.11 mostra o resultado com fundo preto e imagem em tom amarelo.

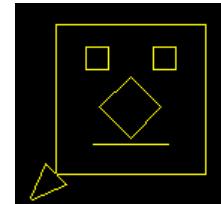


Figura 4.11 - Imagem no fundo preto em tom amarelo.

Para formatar com código **ÍNDICE** o fundo na cor azul e o lápis na cor azul celeste use as instruções:

```
MUDECF 4
MUDECL 6
```

Para ver o resultado execute a instrução:

```
LD FACE
```

A figura 4.12 mostra o resultado com fundo azul e imagem em tom azul celeste.

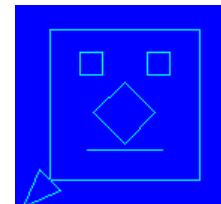


Figura 4.12 - Imagem no fundo azul em tom azul celeste.

**OBS:** O uso de cores baseadas no **ÍNDICE** fica limitado aos valores de "0" a "16" indicados na tabela 4.1. Usar o padrão de cores a partir do formato [R G B] proporciona um leque maior de opções.

Para um teste do uso de cores considere o conjunto de procedimentos a seguir usados para desenhar um sol estilizado com o traço do lápis na cor laranja "[255 200 000]" em fundo na cor azul escuro "[000 000 128]" (adaptado de Abelson, 1981, p. 22). A Figura 4.13 apresenta a imagem gerada pela execução da chamada do procedimento "**SOL 10**".

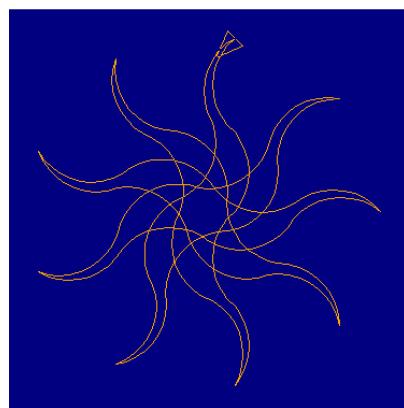


Figura 4.13 - Procedimento "SOL 10".

Desta forma, acione o botão "**Editor**" e informe o seguinte código:

```

APRENDA ARCO1 :TAMANHO :GRAU
REPITA :GRAU / 10 [
  PARAFRENTE :TAMANHO
  PARADIREITA 10
]
FIM

```

```

APRENDA RAIO :TAMANHO
REPITA 2 [
  ARCO2 :TAMANHO 90
  ARCO1 :TAMANHO 90
]
FIM

```

```

APRENDA ARCO2 :TAMANHO :GRAU
REPITA :GRAU / 10 [
  PARAFRENTE :TAMANHO
  PARAESQUERDA 10
]
FIM

```

```

APRENDA SOL :TAMANHO
MUDECF [0 0 128]
MUDECL [255 165 0]
REPITA 9 [
  RAIO :TAMANHO
  PARADIREITA 160
]
FIM

```

Para encerrar o modo **Editor** acione o botão "**Guardar e sair do editor**". Em seguida execute separadamente as instruções após fazer uso do comando **LD** e "**SOL 10**".

Experimente executar os procedimentos "**ARCO1**", "**ARCO2**" e "**RAIO**" separadamente para ver o que cada um deles faz para compor o desenho do procedimento "**SOL**".

Para voltar a tela ao modo padrão execute as instruções:

```

MUDECF 7
MUDECL 0

```

#### 4.5 - Fractais

Fractal é uma estrutura geométrica de forma complexa não regular que possui normalmente propriedades que se repetem em diversas escalas. A linguagem Logo se caracteriza em ser uma ferramenta adequada para a geração dessas formas geométricas.

Não é objetivo deste tópico explorar este tema com profundidade, o objetivo é apenas e tão somente demonstrar a utilização da linguagem nesta tarefa operacional. Para tanto, considere a execução do procedimento "**PONTA**" o qual mostrará a imagem indicada na figura 4.14 a partir da chamada "**PONTA 30**".

```

APRENDA PONTA :LADO
PF :LADO PE 60
PF :LADO PD 120
PF :LADO PE 60
PF :LADO
FIM

```



Figura 4.14 - Imagem do procedimento "PONTA".

A partir do procedimento "**PONTA**" considere o procedimento "**FRACTAL1**" a seguir que executa o procedimento ponta seis vezes, como mostra a figura 4.16 após a chamada "**FRACTAL1 30**".

```

APRENDA FRACTAL1 :LADO
REPITA 6 [
  PD 120
  PONTA :LADO
  PE 60
]
FIM

```

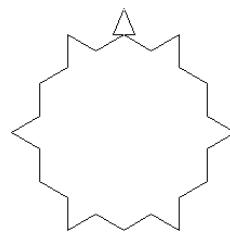


Figura 4.15 - Imagem do procedimento "FRACTAL1".

Note que com os procedimentos "**PONTA**" e "**FRACTAL1**" desenvolvidos foi efetuada uma imagem geométrica irregular (Figura 4.15) a partir de seis repetições da imagem gerada pelo procedimento "**PONTA**" (Figura 4.16).

A partir da imagem proposta pelo procedimento "**PONTA**" é possível formar outras imagens baseadas em fractais. Observe por exemplo o procedimento "**FRACTAL2**" em que se estabelecem mudanças na direção de deslocamento da tartaruga a partir dos comandos **PD** e **PE** dentro do laço gerenciado pelo comando **REPITA**, como indicado na figura 4.16. Use a chamada de instrução "**FRACTAL2 30**".

```

APRENDA FRACTAL2 :LADO
REPITA 6 [
  PE 120
  PONTA :LADO
  PD 60
]
FIM

```

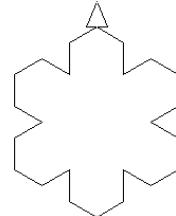


Figura 4.16 - Imagem do procedimento "FRACTAL2".

É possível a partir da execução do procedimento "**PONTA**" fazer a apresentação de formas fractais com formato mais complexo. Observe o procedimento "**FRACTAL3**" que mostra um trecho de imagem fractal bem tradicional, como indicado na figura 4.17. Use "**FRACTAL3 30**".

```

APRENDA FRACTAL3 :LADO
PD 90
REPITA 2 [
  PONTA :LADO
  PE 60
  PONTA :LADO
  PD 120
]
FIM

```

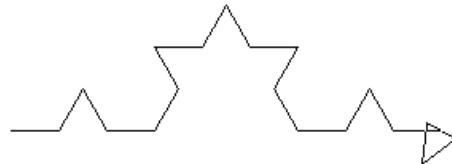


Figura 4.17 - Imagem do procedimento "FRACTAL3".

O efeito gerado pelo procedimento "**FRACTAL3**" faz uso do procedimento "**PONTA**" assim como também fizeram os procedimentos "**FRACTAL1**" e "**FRACTAL2**". A diferença está no posicionamento da tartaruga para a continuidade do desenho. Neste caso, faz-se o deslocamento de direção da tartaruga após desenhar "**PONTA**" pela primeira vez "**60**" graus para a esquerda, faz novo desenho de "**PONTA**" e gira "**120**" graus para a direita.

A partir do procedimento "**FRACTAL3**" pode-se extrapolar a criação de fractais com base no procedimento "**PONTA**". Observe o procedimento "**FRACTAL4**" que gera a imagem indicada na figura 4.18 a partir da execução de "**FRACTAL4 30**" que mostra o fractal de *von Kock* (floco de neve).

```
APRENDA FRACTAL4 :LADO
REPITA 4 [
    FRACTAL3 :LADO
    PE 120
    PD 30
]
FIM
```

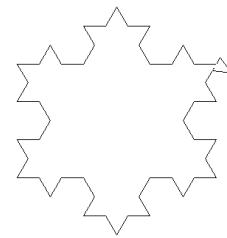


Figura 4.18 - Imagem do procedimento "FRACTAL4".

O procedimento "**SAMAMBAIA**" mostra a imagem de um fractal a partir da definição de determinado número de folhas. A Figura 4.19 mostra a imagem gerada a partir da execução da instrução "**SAMAMBAIA 300**" (adaptado, NASCIMENTO, 2000).

```
APRENDA SAMAMBAIA :FOLHAS
SE :FOLHAS > 4 [
    PF :FOLHAS / 25
    PE 80 SAMAMBAIA :FOLHAS * 0.3
    PD 82 PF :FOLHAS / 25
    PD 80 SAMAMBAIA :FOLHAS * 0.3
    PE 78 SAMAMBAIA :FOLHAS * 0.9
    PE 2 PT :FOLHAS / 25
    PE 2 PT :FOLHAS / 25
]
FIM
```

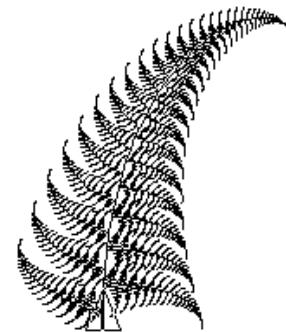


Figura 4.19 - Procedimento "SAMAMBAIA 300".

#### 4.6 - Outras repetições

Além das repetições produzidas com o comando **REPITA** e as operações de recursão Logo possui outras primitivas para a realização de operações baseadas em repetições de trechos de códigos, como: **ENQUANTO** e **PARA**.

O comando **ENQUANTO** (laço condicional pré-teste com fluxo verdadeiro) realiza repetições de trechos de código de forma condicional sem o uso de recursividade enquanto a condição de verificação permanece verdadeira. No momento em que a condição torna-se falsa o laço é automaticamente encerrado. Isso é útil em momentos que se necessite de laço interativo onde não se conhece antecipadamente o número de repetições.

Para verificar o efeito de funcionamento do comando **ENQUANTO** considere o procedimento "**CONTAGEM1**" que apresenta os valores de contagem entre "1" e "5" de "1" em "1". Desta forma, acione o botão "**Editor**" e informe o seguinte código:

```
APRENDA CONTAGEM1
ATRIBUA "I 1
ENQUANTO [:I <= 5] [
    MOSTRE :I
    ATRIBUA "I :I + 1
]
FIM
```

Para encerrar o modo **Editor** acione o botão "**Guardar e sair do editor**". Em seguida execute as instruções **LD**, **LT** e "**CONTAGEM1**" e veja a imagem apresentada semelhante à figura 4.20.

```
CONTAGEM1
1
2
3
4
5
```

Figura 4.20 - Apresentação da contagem de "1" a "5" com "ENQUANTO".

Veja que para o comando **ENQUANTO** funcionar a condição deve permanecer verdadeira por certo tempo, pois a repetição somente é encerrada quando a condição se torna falsa.

No procedimento "**CONTAGEM1**" a variável "I" é iniciada com o "**1**" e cada escrita é somada a ela mais "**1** ("I :I + 1") fazendo que a variável atinja um valor que fará a condição "[**:I <= 5**]" se tornar falsa encerrando as repetições. Atente para os pontos colorizados em vermelho e verde exemplificados anteriormente.

Veja o procedimento "**PENTAGRAMA**" que mostra uma estrela de cinco pontas gerada a partir das primitivas **ENQUANTO** e **MUDEDIREÇÃO (MUDEDÇ)**. A primitiva **MUDEDÇ** pode ser usada como substituta da diretiva **PD**. Assim sendo, acione o botão "**Editor**" e informe o seguinte código:

```
APRENDA PENTAGRAMA
MUDEDIREÇÃO 18
ATRIBUA "I 1
ENQUANTO [:I <= 5] [
  PF 100
  PD 144
  ATRIBUA "I :I + 1
]
FIM
```

Para encerrar o modo **Editor** acione o botão "**Guardar e sair do editor**". Em seguida execute as instruções **LD** e "**PENTAGRAMA**" e veja a imagem apresentada semelhante à figura 4.21.

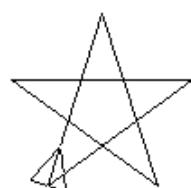


Figura 4.21 - Apresentação de imagem gerada com comando "ENQUANTO".

Os trechos marcados em vermelho mostram a estrutura a ser usada na definição de repetições iterativas que venham a substituir o uso do comando **REPITA**. O uso das atribuições de inicialização ("I 1) e incremento ("I :I + 1) são obrigatórias.

Para se fazer o desenho da estrela de cinco pontas é necessário usar um ângulo de "**144**" graus como apresentado. Talvez a pergunta em sua mente, seja: como saberei esse valor de grau? A resposta é basicamente simples. Divida o valor de graus máximos que é "**360**" por "**5**" e obter-se-á o valor "**72**" que é o ângulo interno de uma figura geométrica, multiplique o valor "**72**" por "**2**" e obter-se-á o valor "**144**" que é o valor do ângulo externo considerado pela linguagem Logo para a geração das imagens de figuras geométricas.

A primitiva (comando) **ENQUANTO** opera repetição de trechos de códigos utilizando-se de condição. Por esta razão podem ser usados para a execução de laços interativos e iterativos (como faz a primitiva **REPITA**)

A primitiva **PARA** é semelhante a **REPITA** por realizar ações iterativas. Para ver o funcionamento de **PARA** considere o procedimento "**CONTAGEM2**" que mostra os valores de contagem entre **1** e **5** de **1** em **1**. Para tanto, acione o botão "**Editor**" e informe o seguinte código:

```
APRENDA CONTAGEM2
  PARA [I 1 5 1] [
    MOSTRE :I
  ]
FIM
```

Para encerrar o modo **Editor** acione o botão "**Guardar e sair do editor**". Em seguida execute as instruções **LD** e "**CONTAGEM2**". Note que a primitiva **PARA** usa a definição de uma variável e a definição de três valores: o primeiro valor determina o início da contagem, o segundo valor determina o fim da contagem e o terceiro valor determina o incremento da contagem, que pode ser omitido quando o incremento é de "1" em "1".

Já que foi feita uma estrela de cinco pontas (pentagrama) com o comando **ENQUANTO** será feita uma estrela de quarenta pontas com o comando **PARA**. Para tanto, definida o procedimento "**ESTRELA40**". Assim sendo, acione o botão "**Editor**" e informe o seguinte código:

```
APRENDA ESTRELA40
  PARA [I 1 40] [
    PT 200
    PE 170
  ]
FIM
```

Para encerrar o modo **Editor** acione o botão "**Guardar e sair do editor**". Em seguida execute as instruções **TAT** e "**ESTRELA40**" e veja a imagem apresentada semelhante à figura 4.22.

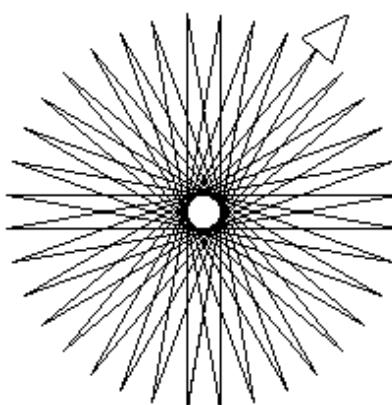


Figura 4.22 - Apresentação de imagem gerada com comando "PARA".

Os comandos: **REPITA**, **ENQUANTO** e **PARA** são recursos complementares e não necessariamente substitutos um dos outros. Esses comandos podem ser trabalhados juntos dentro de um mesmo projeto.

Os comandos podem ser usados sequencialmente ou encadeados. Sequencialmente quando um é definido após o outro e encadeado quando é definido um dentro do outro. Veja o procedimento seguinte chamado "**RODAFLOR**" que faz uso dos comandos **PARA**, **ENQUANTO** e **REPITA** no mesmo projeto na forma sequencial e encadeada. Assim sendo, acione o botão "**Editor**" e informe o seguinte código:

#### APRENDA RODAFLOR

```
PARA [I 1 6] [
    ATRIBUA "I 1
    ENQUANTO [:I <= 120] [
        PF 1 PD 1
        ATRIBUA "I :I + 1
    ]
    PD 60
    REPITA 180 [
        PF 1 PD 1
    ]
    PD 60
]
FIM
```

Para encerrar o modo **Editor** acione o botão "**Guardar e sair do editor**". Em seguida execute as instruções **TAT** e "**RODAFLOR**" e veja a imagem apresentada semelhante à figura 4.23.

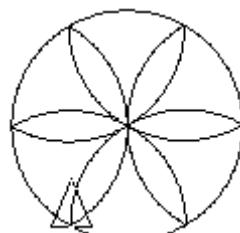


Figura 4.23 - Apresentação de imagem gerada com repetições sequenciais e encadeadas.

O comando **PARA** pode ser um substituto ao comando **REPITA** exatamente por operar na mesma categoria de repetições, uma vez que é usado para repetições iterativas. No entanto, possui duas características particulares que o diferencia do comando **REPITA**.

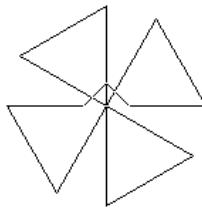
O comando **REPITA** usa para identificar o valor da contagem o apoio do comando **CONTEVEZES**. Com o comando **PARA** é necessário fazer a definição de uma variável para a contagem e definir seus valores de início, fim e incremento quando diferente de "1". No entanto, **PARA** apresenta maior mobilidade que **REPITA** pois permite definir valores de contagem variados enquanto que **REPITA** sempre faz a contagem de "1" em "1" começando em "1" até o valor limite a ele estabelecido.

#### OBS:

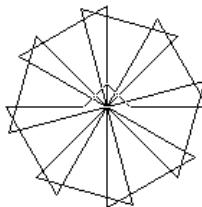
O ambiente "xLogo" executa os procedimentos desenvolvidos de maneira rápida. Apesar dessa rapidez ser importante no processamento dos dados elaborados por computadores, ela pode atrapalhar um pouco a percepção de como os desenhos são efetivamente feitos. Desta forma, pode-se pedir para que o ambiente opere em velocidade menor com o uso da primitiva **ESPERE** após riscar um traço. Por exemplo: "**REPITA 4 [PF 120 ESPERE 60 PD 90]**". O valor "60" corresponde a "1" segundo.

#### 4.7 - Exercícios de fixação

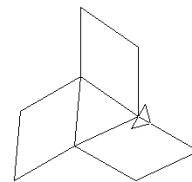
1. Criar procedimento chamado **RETAMETA** que desenhe um retângulo cujo lado menor (comprimento) seja a metade do lado maior (altura) onde o tamanho informado deverá ser fornecido como parâmetro. Movimente o desenho para frente com sentido a direita.
2. Crie um procedimento chamado **TRIANGEX**, que apresente um triângulo equilátero com lado de tamanho "70" para trás a partir de giros a esquerda.
3. Criar procedimento chamado **FLORTIG** desenhado a partir do procedimento **TRIANGEX** com giro a direita de modo que tenha a seguinte aparência.



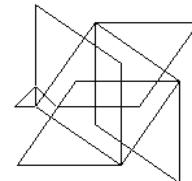
4. Criar procedimento chamado **VENTITRIG** desenhado a partir do procedimento **TRIANGEX** com giro a esquerda de modo que tenha a seguinte aparência.



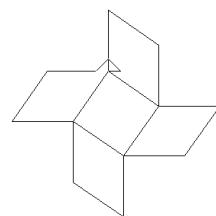
5. Criar procedimento chamado **LOSANGO** que desenhe imagem de mesmo nome com tamanho fornecido por parâmetro a partir de giro a direita.
6. A partir do procedimento **LOSANGO** crie procedimento chamado **FLORLOSA1** de modo que seja apresentada a figura a seguir com tamanho "80".



7. A partir do procedimento **LOSANGO** crie procedimento chamado **FLORLOSA2** de modo que seja apresentada a figura a seguir com tamanho "80".



8. A partir do procedimento **LOSANGO** crie procedimento chamado **FLORLOSA3** de modo que seja apresentada a figura a seguir com tamanho "**80**".



9. Sem executar no computador descrimine qual imagem é apresentada.

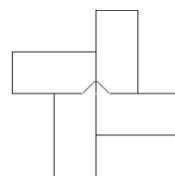
REPITA 12 [REPITA 3 [PF 50 PD 120] PD 30]

---

---

10. Criar procedimento chamado **RETANGULO3** com tamanhos "**100**" (altura) e "**50**" (largura). Movimente-se para frente com giro a direita.

11. Criar procedimento chamado **CATAVENTO1** com o formato da figura seguinte a partir do procedimento **RETANGULO3**.

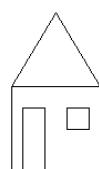


12. Criar procedimento chamado **TRIANGPR** que desenhe um triângulo equilátero com tamanho definido por parâmetro com comando **PARA** contando de "**0**" a "**2**" no sentido a frente com giro a direita.

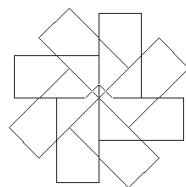
13. Criar procedimento chamado **QUADRADO1** que desenhe um quadrado com lado definido por parâmetro a partir do uso da primitiva **ENQUANTO**. Conte de "**1**" a "**4**".

14. Criar procedimento chamado **QUADRADO2** que desenhe um quadrado com lado definido por parâmetro a partir do uso da primitiva **PARA**. Conte de "**1**" a "**4**".

15. Criar procedimento chamado **CASINHA** que mostre a imagem da casa. O quadrado e o triângulo deverão possuir tamanho "**80**", a porta é um retângulo de "**60**" por "**20**" e a janela é um quadrado de tamanho "**20**".



16. Criar procedimento chamado **CATAVENTO2** com o formato da figura seguinte a partir do procedimento **RETANGULO3**.



17. Descubra sem o uso do computador qual é a imagem:

```
APRENDA QUADRO :TAMANHO
REPITA 4 [
    PF :TAMANHO
    PD 90
]
PD 45
PF :TAMANHO * 7 / 5
PT :TAMANHO * 7 / 5
PE 45
PF :TAMANHO
PD 135
PF :TAMANHO * 7 / 5
PT :TAMANHO * 7 / 5
FIM
```

---

---

## CAPÍTULO 5 - Ações complementares

A linguagem Logo vai além da possibilidade de apenas fazer belas figuras. Logo é uma linguagem como outra qualquer e possibilita muitos recursos para o desenvolvimento de programação lógica, funcional e estruturada. Este capítulo distancia-se do mundo gráfico da tartaruga e mostra a linguagem em uma outra dimensão.

### 5.1 - Funções matemáticas

Anteriormente foram comentadas e apresentadas algumas funções matemáticas da linguagem, como: **ABS**, **DIFERENÇA**, **INTEIRO**, **POTÊNCIA**, **PRODUTO**, **QUOCIENTE**, **RAIZQ**, **REPITASORTEIE**, **RESTO**, **SOMA** e **SORTEIENÚMERO**, sendo essas algumas das funções existentes. A linguagem Logo possuí as funções matemáticas: **ARCCOS**, **ARCCOSRAD**, **ARCSEN**, **ARCSENRAD**, **ARCTAN**, **ARCTANRAD**, **ARREDONDE**, **COS**, **COSRAD**, **EXP**, **LN**, **LOG10**, **PI**, **SEN**, **SENRAD**, **TAN** e **TANRAD**, indicadas na tabela 5.1.

FUNÇÃO	OPERAÇÃO
ACOS	Calcula o arco cosseno de número, em graus.
ASEN	Calcula o arco seno de número, em graus.
ATAN	Calcula o arco tangente em graus, de número.
COS	Calcula o cosseno de um ângulo medido em graus.
EXP	Calcula o exponencial de um número
LN	Calcula o logaritmo natural (base "e") de número.
LOG10	Calcula o logaritmo na base 10 de número.
PI	Retorna o valor de pi 3.14159265358979).
SEN	Calcula o seno do ângulo medido em graus.
TAN	Calcula a tangente do ângulo medido em graus.

Tabela 5.1 - Funções matemáticas e suas ações de ângulos.

Veja a seguir alguns exemplos de apresentação de resultados gerados a partir do uso das funções matemáticas apresentadas.

```
MO ACOS 0.5  
MO ASEN 0.5  
MO ATAN 0.5  
MO COS 0.5  
MO EXP 1  
MO LN (EXP 2)  
MO LOG10 100  
MO PI  
MO SEN 90  
MO TAN 90
```

Após essas execuções use a primitiva **MUDEPRECISÃO** para estabelecer a quantidade de casas decimais a ser apresentada. Pode-se fazer uso de valores numéricos inteiros entre "0" e "16". Assim sendo execute a instrução:

```
MUDEPRECISÃO 2
```

Veja mais alguns testes de execução com uso de funções matemáticas:

```
MO COS 0.5  
MO PI  
MO SEN 90  
MO TAN 90
```

Veja os resultados gerados a partir das funções comentadas na figura 5.1.

```
MO ACOS 0.5  
60.0000000000001  
MO ASEÑ 0.5  
30.0000000000004  
MO ATAN 0.5  
26.56505117707799  
MO COS 0.5  
0.9999619230641713  
MO EXP 1  
2.718281828459045  
MO LN (EXP 2)  
2  
MO LOG10 100  
2  
MO PI  
3.141592653589793  
MO SEN 90  
1  
MO TAN 90  
1.633123935319537E16  
MUDEPRECISÃO 2  
MO COS 0.5  
0.91  
MO PI  
3.14  
MO SEN 90  
1  
MO TAN 90  
16331239353195370
```

Figura 5.1 - Apresentação dos resultados no uso de funções matemáticas.

É sabido que um computador efetua três ações essenciais: entrada de dados, processamento de dados e saída de dados na forma de informação. As operações de entrada na linguagem Logo podem ser realizadas com o uso de parâmetros junto aos procedimentos ou a partir da primitiva **LEIA** e as operações de saída ocorrem nas áreas de ação dos modos gráfico e texto a partir das primitivas **ESCREVA**, **MOSTRE** ou **ROTULE**. Em relação ao processamento este evento pode ocorrer em duas dimensões que se misturam: uma dimensão matemática e outra dimensão lógica.

As funções matemáticas e os operadores aritméticos são responsáveis por atenderem a necessidade de execução de cálculos aritméticos na resolução de problemas matemáticos. As funções matemáticas podem ser utilizadas no desenvolvimento de figuras, especialmente as funções trigonométricas. Veja um exemplo de imagem com o uso da função **SEN**:

```
REPITA 360 [MUDEXY (SEN(2 * CV)) * 150 (SEN(3 * CV)) * 100]
```

A figura 5.2 mostra o resultado da execução da instrução com uso da função **SEN**.

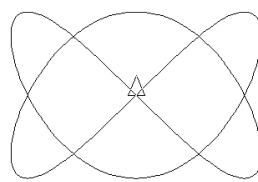


Figura 5.2 - Imagem obtida a partir do uso da função "SEN".

## 5.2 - Funções lógicas

Além das funções matemáticas a um conjunto de funções lógicas que auxiliam diversas ações de processamento. Funções do tipo lógicas são recursos que retornam como resposta de sua operação os valores **VERD** ou **FALSO** sendo ótimas no estabelecimento de condições para as tomadas de decisão. Veja algumas funções lógicas existentes em Logo: **ÉMEMBRO**, **ÉNÚMERO**, **ÉTECLA?** e **SÃOIGUAIS?**, indicadas na tabela 5.2.

FUNÇÃO	OPERAÇÃO
<b>ÉMEMBRO?</b>	<b>Retorna VERD se parâmetro 1 for membro de parâmetro 2.</b>
<b>ÉNÚMERO?</b>	<b>Retorna VERD se valor for um número.</b>
<b>SÃOIGUAIS?</b>	<b>Retorna VERD se parâmetros forem iguais.</b>

Tabela 5.2 - Funções lógicas (algumas).

Veja a seguir alguns exemplos de apresentação de resultados gerados a partir do uso das funções lógicas apresentadas e seus resultados na figura 5.3.

```
MO ÉMEMBRO? 1 [1 2 3 4 5]
MO ÉMEMBRO? 7 [1 2 3 4 5]
MO ÉMEMBRO? 7 "678
MO ÉMEMBRO? 7 [678]
MO ÉNÚMERO? 3
MO ÉNÚMERO? [1 2 3]
MO SÃOIGUAIS? 2 2
MO SÃOIGUAIS? 1 2
MO SÃOIGUAIS? [1 2 3] [1 2 3]
MO SÃOIGUAIS? [1 2 3] [3 2 1]
```

```
MO ÉMEMBRO? 1 [1 2 3 4 5]
verd
MO ÉMEMBRO? 7 [1 2 3 4 5]
falso
MO ÉMEMBRO? 7 "678
verd
MO ÉMEMBRO? 7 [678]
falso
MO ÉNÚMERO? 3
verd
MO ÉNÚMERO? [1 2 3]
falso
MO SÃOIGUAIS? 2 2
verd
MO SÃOIGUAIS? 1 2
falso
MO SÃOIGUAIS? [1 2 3] [1 2 3]
verd
MO SÃOIGUAIS? [1 2 3] [3 2 1]
falso
```

Figura 5.3 - Apresentação dos resultados no uso de funções matemáticas.

As funções lógicas são úteis no estabelecimento de controles em tomadas de decisão e execução de laços condicionais exatamente por retornarem como resposta a sua ação um valor **VERD** ou **FALSO**. Essas funções podem ser usadas para auxiliar diversas ações da linguagem seja no estabelecimento de operações matemáticas ou na elaboração de figuras.

### 5.3 - Algumas funções complementares

O conjunto de funções na linguagem Logo é extenso veja algumas funções complementares que não foram anteriormente qualificadas, sendo: **PALAVRA**, **PRIMEIRO**, **ÚLTIMO** e **OBJETO**. Observe a tabela 5.3.

FUNÇÃO	OPERAÇÃO
<b>OBJETO</b>	<b>Retorna o conteúdo que esteja associado a uma variável.</b>
<b>PALAVRA</b>	<b>Concatena palavras formando outra palavra.</b>
<b>PRIMEIRO (PRI)</b>	<b>Retorna o primeiro elemento de uma série de valores.</b>
<b>ÚLTIMO (ULT)</b>	<b>Retorna o último elemento de uma série de valores.</b>

Tabela 5.3 - Funções complementares (algumas).

As funções **PALAVRA**, **PRIMEIRO** e **ÚLTIMO** podem ser demonstradas em uma linha de instrução como foram demonstradas as funções matemáticas e lógicas. Veja a seguir alguns exemplos e atente ao uso das chaves com a função **PRIMEIRO** e colchetes com a função **ÚLTIMO** quando usadas na manipulação de conjuntos de elementos e os resultados na figura 5.4.

```
MO PALAVRA "Lo "go
MO PALAVRA "x "Logo
MO PRIMEIRO [1 2 3 4 5]
MO PRI [5 4 3 2 1]
MO PRI "Logo
MO PRIMEIRO [[L][o][g][o]]
MO ÚLTIMO [1 2 3 4 5]
MO ULT "Computador
MO ÚLTIMO [[L][o][g][o]]
MO ÚLTIMO [A B C D E]
```

```
MO PALAVRA "Lo "go
Logo
MO PALAVRA "x "Logo
xLogo
MO PRIMEIRO [1 2 3 4 5]
1
MO PRI [5 4 3 2 1]
5
MO PRI "Logo
L
MO PRIMEIRO [[L][o][g][o]]
L
MO ÚLTIMO [1 2 3 4 5]
5
MO ULT "Computador
r
MO ÚLTIMO [[L][o][g][o]]
o
MO ÚLTIMO [A B C D E]
E
```

Figura 5.4 - Apresentação dos resultados no uso de funções auxiliares.

A função **OBJETO** opera sua ação sobre variáveis. Desta forma, é necessário que haja na memória definição de variáveis que possam ser utilizadas pela função. Assim sendo, considere o seguinte trecho de instruções (ATARI, 1983, p. 76) que definem as variáveis de memória:

```
ATR "VENCEDOR "COMPUTADOR
ATR "COMPUTADOR [100 PONTOS]
```

Observe um detalhe que pode passar desapercebido. Veja que na primeira instrução são definidas duas variáveis: **VENCEDOR** e **COMPUTADOR**. Veja que a variável **COMPUTADOR** ao ser definida após a variável **VENCEDOR** tornou-se conteúdo da variável **VENCEDOR**.

A partir dessa ocorrência, perceba, que o conteúdo da variável **COMPUTADOR** está vinculado a variável **VENCEDOR**. O acesso ao conteúdo vinculado entre as variáveis é realizado com o uso da função **OBJETO**. Veja as próximas instruções:

```
MO :VENCEDOR
MO OBJETO :VENCEDOR
MO OBJETO "VENCEDOR
```

Veja os resultados gerados com o uso da função **OBJETO** indicados na figura 5.5. Atente a diferença de uso dos símbolos [:] dois pontos e [""] aspa inglesa no acesso ao conteúdo da variável indicada.

```
ATR "VENCEDOR "COMPUTADOR
ATR "COMPUTADOR [100 PONTOS]
MO :VENCEDOR
COMPUTADOR
MO OBJETO :VENCEDOR
100 PONTOS
MO OBJETO "VENCEDOR
COMPUTADOR
```

Figura 5.5 - Apresentação de resultados com função "VALOR".

#### 5.4 - Programação sem figuras

Logo é uma linguagem que possui muito mais do que aparentemente mostra. É possível desenvolver várias categorias de aplicação com Logo, mesmo não sendo usada no universo comercial. As aplicações Logo são usadas mais intensamente no universo educacional e podem ser amplamente usadas industrialmente junto a controle de robôs em produção crítica que colocam a vida humana em risco.

A primeira versão da linguagem foi escrita em computadores que não são usados domesticamente, ou seja, computadores hoje chamados de grande porte. Sua interface de acesso para crianças pré-alfabetizadas era intermediada por um console de Terminal (equipamento com monitor de vídeo e teclado) e para crianças não alfabetizadas a partir de um console com botões conectados ao computador ligado por um cabo (cordão umbilical) a um robô mecânico, chamada tartaruga que percorria sobre uma grande folha de papel traçando imagens geométricas planas (figura 5.6).

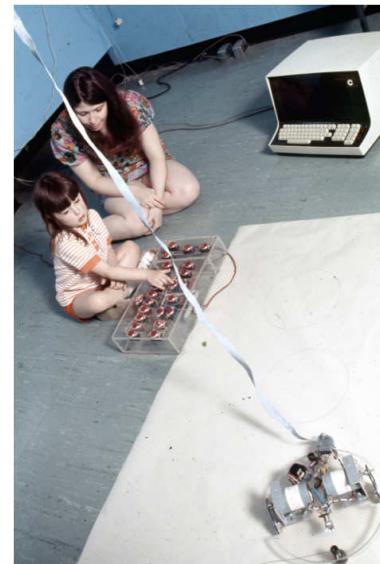


Figura 5.6 - Criança desenhando com Logo (SOLOMON, 2020, p. 39).



Figura 5.7 - Tartaruga remota  
(CATLIN, 2016).

Sem sombra de dúvidas pensar em Logo nas décadas de 1960 e 1970 era muito divertido, pois era possível ver um robô sobre uma folha de papel produzir desenhos incríveis (figura 5.7).

Depois veio a fase do controle remoto de periféricos onde o robô já não possuía mais um cordão umbilical e sim controlado por ondas de rádio, mais emocionante ainda.

Possivelmente por questões de custos o controle da tartaruga diretamente na tela de um computador se tornou bastante popular afastando fisicamente a linguagem da tartaruga.

A linguagem Logo vem sendo, há muito tempo, erroneamente vinculada a aprendizagem apenas de crianças. Isso pode até ter sido verdade, como as figuras anteriores mostram, há muito tempo, em meados de 1960, por ter sido inicialmente apresentada a esse grupo de pessoas. No entanto, a partir de 1975 com a introdução dos microcomputadores sua popularidade decolou e Logo se tornou bastante popular atraindo a atenção de outros públicos. Por controlar as ações de um robô, Logo se mostrou uma excelente linguagem para uso em ações de automação industrial e mecatrônica, indo além do que simplesmente fazer desenhos. Em certos momentos deste livro você deve ter notado exemplos de procedimentos que não elaboraram desenhos na área de ação do modo gráfico, como o procedimento "**TABUADA**" do capítulo "4" que mostra o resultado de sua operação na área de ação do modo texto. Esse é um tipo de ação que muitas vezes passa despercebido com o uso da linguagem Logo, pois é comum ver seu foco voltado a aplicações geométricas, até mesmo na quantidade de material produzido para apresentar a linguagem.

Para exemplificar a linguagem fora do eixo geométrico considere a seguir alguns exemplos aplicativos simples que efetuam algumas ações computacionais comuns a qualquer linguagem de programação.

Considere para o primeiro exemplo um procedimento chamado "**AREACIRC**" que receba a entrada de um valor real para a medida do raio de uma circunferência e apresente o resultado da área desta circunferência sem desenhá-la com três casa decimais. Desta forma, acione o botão "**Editor**" e informe o seguinte código:

```
APRENDA AREACIRC
LEIA [Informe a medida do raio de uma circunferência:] "RAIO"
ATR "AREA PI * (POTÊNCIA :RAIO 2)
ESC (SENTENÇA "Área " = :AREA)
FIM
```

Para encerrar o modo **Editor** acione o botão "**Guardar e sair do editor**". Execute o procedimento "**AREACIRC**" e informe para um primeiro teste o valor "**5**" e veja a apresentação do resultado da área com "**78.53**" (ou "**78.539816339744833**" caso a precisão decimal esteja em modo padrão) como mostra a figura 5.8.

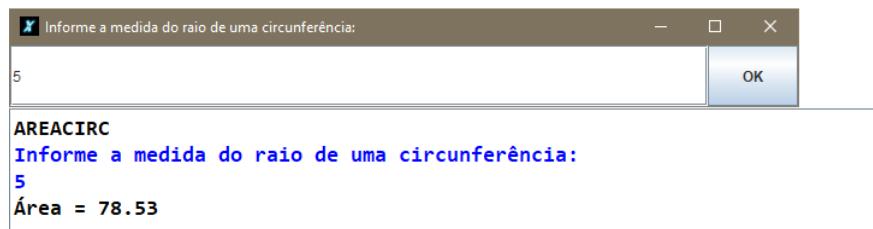


Figura 5.8 - Entrada e saída do procedimento "AREACIRC".

Observe cada detalhe de cada instrução do procedimento "**AREACIRC**". Note que são usados diversos elementos integrados na ideia de um todo com cada componente realizando uma ação que complementa outro componente.

O segundo exemplo, Procedimento chamado "**DECISAO**" recebe a entrada de dois valores numéricos representados pelas variáveis "**A**" e "**B**", soma-os armazenando seu resultado em "**X**" e apenas mostra o resultado caso este seja maior que "**10**". Assim sendo, acione o botão "**Editor**" e informe o seguinte código:

#### APRENDA DECISAO

```
LEIA [Entre um valor numérico para a variável <A>:] "A
LEIA [Entre um valor numérico para a variável <B>:] "B
ATR "X :A + :B
SE :X > 10 [
    MO (SN "Resultado" "=" :X)
]
FIM
```

Encerre o modo **Editor** com o botão "**Guardar e sair do editor**". Execute o procedimento "**DECISAO**", informe os valores "**5**" e "**6**" e veja o resultado "**11**", depois execute novamente o procedimento, informe os valores "**5**" e "**5**" e observe. As figuras 5.9 e 5.10 mostram os resultados desses testes.

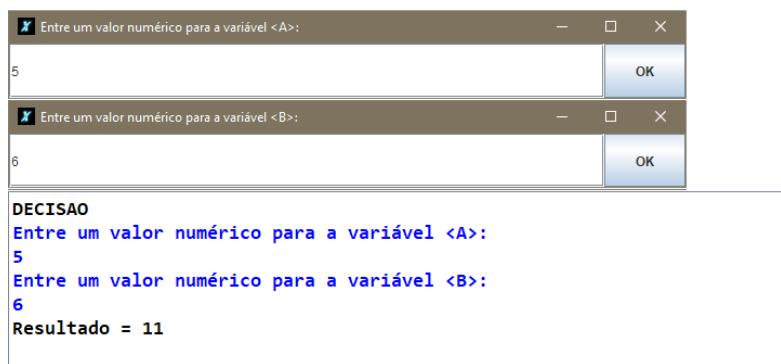


Figura 5.9 - Testes de execução do procedimento "DECISAO".

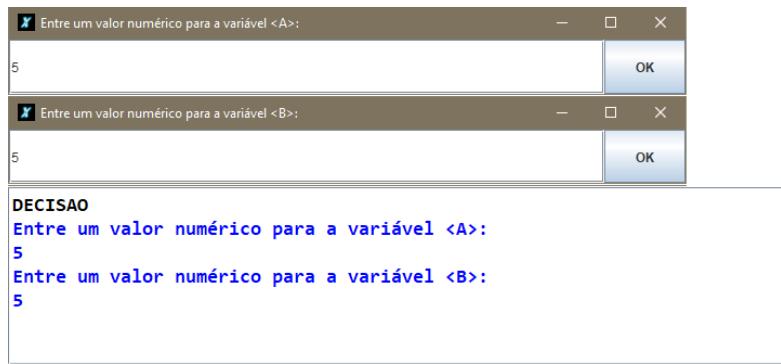


Figura 5.10 - Testes de execução do procedimento "DECISAO".

Observe que no procedimento "DECISAO" foi omitida na primitiva **SE** o segundo bloco da tomada de decisão. Quando isso é feito diz-se que está sendo usada uma estrutura de *tomada de decisão simples*. Anteriormente os exemplos demonstrados usam a estrutura de *tomada de decisão composta*.

### 5.5 - Manipulação básica de dados

Em alguns outros momentos deste estudo foram apresentados certos detalhes sobre o uso e apresentação de textos e o tratamento de dados. Cabe junto a este tópico rever alguns detalhes e apresentar outros que complementem este conhecimento. Observe a tabela 5.5, adaptado de Downey & Gay (2003, p. 31-32, 55, 57-58, 60 e 63).

FUNÇÃO	OPERAÇÃO
CAR	Retorna o caractere correspondente ao código ASCII.
ÉANTERIOR?	Retorna VERD se primeiro dado vem a frente do segundo.
SEMPRIMEIRO (SP)	Mostra todos os elementos, exceto o primeiro.
SEMÚLTIMO (SU)	Mostra todos os elementos, exceto o último.

Tabela 5.5 - Funções complementares (mais algumas).

Veja alguns detalhes de algumas funções apresentadas indicados por instruções em azul e seus efeitos como resultados gerados em vermelho.

MO PRIMEIRO "ABCDE"

A

MO PRIMEIRO 54321

5

MO ÚLTIMO "ABCDE"

E

MO ÚLTIMO 54321

1

MO CAR 65

A

MO ÉANTERIOR? "ABC" "ABD"

VERD

MO ÉANTERIOR? "ABD" "ABC"

FALSO

MO SEMPRIMEIRO 1.2345 # "SEMPRIMEIRO" pode ser escrito como "SP".

0.2345

MO SP 54321

4321

MO SEMÚLTIMO "ABCDE" # "SEMÚLTIMO" = "SU".

ABCD

MO SU 54321

5432

MO SENTENÇA "Linguagem" "Logo  
Linguagem Logo

(MO "Linguagem" "Logo)  
Linguagem Logo

MO (SENTENÇA "Estudo" "de" "Linguagem" "Logo)  
Estudo de Linguagem Logo

MO (SENTENÇA 1 2 3 4 5)  
1 2 3 4 5

MO (PRODUTO 2 3 4)  
24

MO (SOMA 1 2 3 4 5)  
15

Veja a seguir o procedimento "**TEXTOTRIANG1**" que a partir de uma palavra informada como parâmetro a apresente de forma triangular diminuindo-se a cada apresentação um caractere removido do lado esquerdo, adaptado de Muller (1998, p. 502). Desta forma, acione o botão "**Editor**" e informe o seguinte código:

```
APRENDA TEXTOTRIANG1 :CONTEUDO
  SE :CONTEUDO = " " [PARE] # Quando CONTEUDO for vazio PARE.
  MO :CONTEUDO
    TEXTOTRIANG1 SEMPRIMEIRO :CONTEUDO
  FIM
```

Encerre o modo **Editor** com o botão "**Guardar e sair do editor**". Na sequência execute a instrução **TEXTOTRIANG1** "**COMPUTADOR**" e veja o resultado apresentado na área de ação do modo texto.

COMPUTADOR  
COMPUTADO  
COMPUTAD  
COMPUTA  
COMPUT  
COMPU  
COMP  
COM  
CO  
C

Considere outro procedimento chamado "**TEXTOTRIANG1**" que apresente a palavra informada como parâmetro do último caractere para o primeiro, adaptado de Muller (1998, p. 504). Assim sendo, acione o botão "**Editor**" e informe o seguinte código:

```
APRENDA TEXTOTRIANG2 :CONTEUDO
  SE :CONTEUDO = " " [PARE] # Quando CONTEUDO for vazio PARE.
  TEXTOTRIANG2 SEMPRIMEIRO :CONTEUDO
  MO :CONTEUDO
  FIM
```

Encerre o modo **Editor** com o botão "**Guardar e sair do editor**". Na sequência execute a instrução **TEXTOTRIANG2 "COMPUTADOR"** e veja o resultado apresentado na área de ação do modo texto.

```
C  
CO  
COM  
COMP  
COMPU  
COMPUT  
COMPUTA  
COMPUTAD  
COMPUTADO  
COMPUTADOR
```

Veja que o efeito de apresentação entre os dois últimos procedimentos ocorre a partir da posição da instrução "**MO :CONTEUDO**" antes e após a instrução "**TEXTOTRIANG2 SEMPRIMEIRO :CONTEUDO**".

### 5.6 - Escopo e visibilidade de variáveis

O espaço de memória pode ser controlado com a definição do escopo de comportamento das variáveis, que podem ser globais e locais, dependendo apenas da primitiva de definição de variável em uso. As variáveis globais são definidas com a primitiva **ATRIBUA (ATR)** amplamente usada neste livro e as variáveis locais são definidas com a primitiva **LOCAL**. Quando uma variável global é definida dentro de um procedimento ela se torna visível fora do procedimento e para todos os sub procedimentos relacionados o que é diferente para uma variável definida como local, pois neste caso, esta variável é visível dentro do procedimento em que foi definida e também aos sub procedimento relacionados, mas não será visível fora do procedimento.

Observe o procedimento chamado "**VARGLOBA1**" com a definição de uma variável global e o acesso a esta variável de forma interna e externa ao procedimento. Para tanto, acione o botão "**Editor**" e informe o seguinte código:

```
APRENDA VARGLOBA1  
ATR "CONTADORG 1  
MO :CONTADORG  
ATR "CONTADORG :CONTADORG + 1  
MO :CONTADORG  
FIM
```

Encerre o modo **Editor** com o botão "**Guardar e sair do editor**". Em seguida execute a instrução de chamada do procedimento:

```
VARGLOBA1
```

Observe os resultados apresentados e na sequência na linha de comando do campo de entrada de comandos, dados e instruções informe a seguinte instrução:

```
MO SOMA :CONTADORG 1
```

Veja na figura 5.11 a apresentação dos resultados de uso de variável global. Note que a instrução "MO SOMA :CONTADORG 1" soma mais "1" ao valor da variável "CONTADORG" tornando seu resultado "3" por ser "CONTADORG" uma variável global.

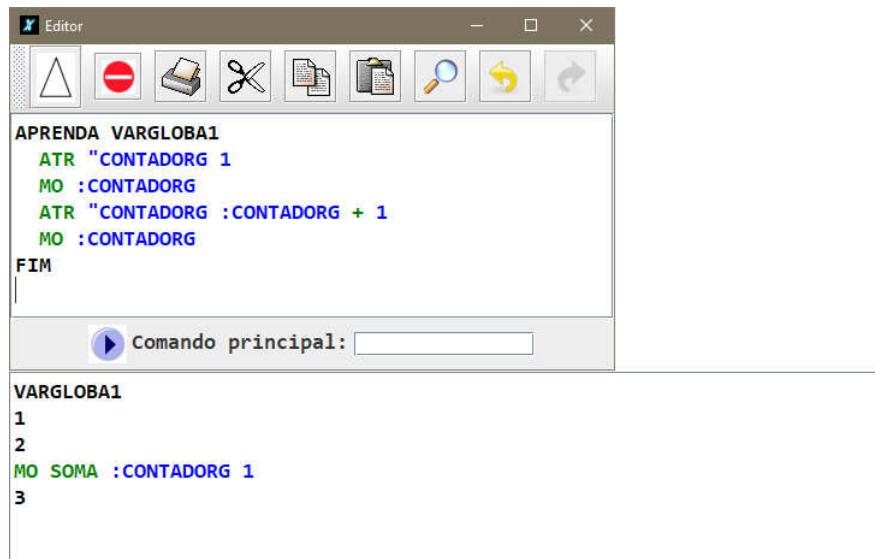


Figura 5.11 - Resultado no uso de variável global.

Observe o procedimento chamado "**VARLOCAL1**" com a definição de uma variável local com acesso apenas interno ao procedimento. Desta forma, acione o botão "**Editor**" e informe o seguinte código:

```

APRENDA VARLOCAL1
LOCAL "CONTADORL"
ATR "CONTADORL 1"
MO :CONTADORL
ATR "CONTADORL :CONTADORL + 1"
MO :CONTADORL
FIM

```

Encerre o modo **Editor** com o botão "**Guardar e sair do editor**". Em seguida execute a instrução de chamada do procedimento:

**VARLOCAL1**

Observe os resultados apresentados e na sequência na linha de comando do campo de entrada de comandos, dados e instruções informe a seguinte instrução:

**MO SOMA :CONTADORL 1**

Veja na figura 5.12 a apresentação dos resultados de uso de variável global. Note que a instrução "MO SOMA :CONTADORL 1" tenta somar mais "1" ao valor da variável "CONTADORL" que retorna como resposta a informação "**CONTADORL não possui um valor**" por ser "CONTADORL" uma variável local.

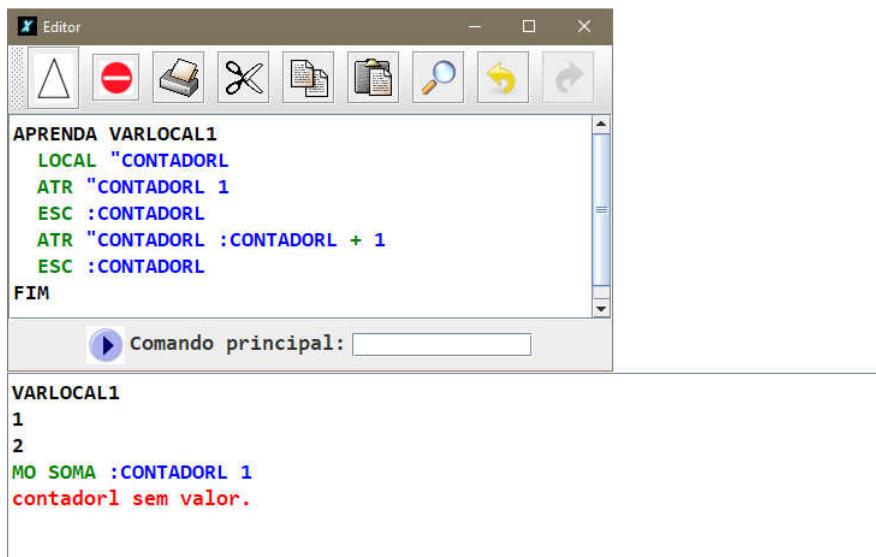


Figura 5.12 - Resultado no uso de variável local.

Observe que a primitiva **LOCAL** permite que seja definida uma variável com estado de visibilidade local, mas após está definição se faz uso da primitiva **ATRIBUA (ATR)** normalmente. Então, atente ao fato de que se uma variável definida com **ATRIBUA (ATR) sem o uso prévio de "LOCAL"** é global e *com o uso prévio de "LOCAL"* é local.

Veja em seguida o comportamento de variáveis globais e locais a partir do uso de sub procedimento (sub-rotinas).

Observe o procedimento chamado "**VARGLOBA2**" com a definição global de variável e uso de sub-rotina para a ação de contagem. Para tanto, acione o botão "**Editor**" e informe o seguinte código:

```

APRENDA VARGLOBA2
  ATR "CONTADORGX 1
  MO :CONTADORGX
  SUBVARGLOBA2
FIM
```

```

APRENDA SUBVARGLOBA2
  ATR "CONTADORGX :CONTADORGX + 1
  MO :CONTADORGX
FIM
```

Encerre o modo **Editor** com o botão "**Guardar e sair do editor**". Em seguida execute a instrução de chamada do procedimento:

**VARGLOBA2**

Observe os resultados apresentados e na sequência na linha de comando do campo de entrada de comandos, dados e instruções informe a seguinte instrução:

**MO SOMA :CONTADORGX 1**

Veja a apresentação dos resultados "**1**", "**2**" e "**3**" a partir da execução do procedimento principal com chamada de sub procedimento (sub-rotina) utilizando-se variável global, indicado na figura 5.13.

```
VARGLOBA2
1
2
MO SOMA :CONTADORGX 1
3
```

Figura 5.13 - Resultado no uso de variável global com procedimento e sub-rotina.

Na sequência veja o procedimento chamado "**VARLOCAL2**" com a definição local de variável e uso de sub-rotina para a ação de contagem. Desta forma, ação o botão "**Editor**" e informe o seguinte código:

```
APRENDA VARLOCAL2
LOCAL "CONTADORLX
ATR "CONTADORLX 1
MO :CONTADORLX
SUBVARLOCAL2
FIM
```

```
APRENDA SUBVARLOCAL2
ATR "CONTADORLX :CONTADORLX + 1
MO :CONTADORLX
FIM
```

Encerre o modo **Editor** com o botão "**Guardar e sair do editor**". Em seguida execute a instrução de chamada do procedimento:

```
VARLOCAL2
```

Veja a apresentação do resultado "1" e "**Em subvalorlocal2, linha :1 contadorlx sem valor**" a partir da execução do procedimento principal com chamada de sub procedimento (sub-rotina) utilizando-se variável local, indicado na figura 5.14. Veja que a sub-rotina retorna erro por não conseguir "ver" a variável na memória (por ser uma variável local).

```
VARLOCAL2
1
Em subvarlocal2, linha 1:
contadorlx sem valor.
```

Figura 5.14 - Resultado no uso de variável local com procedimento e sub-rotina.

## 5.7 - Exercícios de fixação

1. Criar procedimento chamado **CAP0501** que efetue a leitura de um valor numérico inteiro e apresente o resultado do valor lido elevado ao quadrado sem efetuar o armazenamento do resultado em memória. A variável que receberá a entrada do dado deve ser definida como local.
2. Criar procedimento chamado **CAP0502** que efetue a leitura de um valor numérico inteiro e apresente o resultado do valor lido elevado ao cubo com armazenamento do resultado calculado em memória. As variáveis devem ser definidas como local.

3. Criar procedimento chamado **CAP0503** que efetue a leitura de uma temperatura em graus Celsius e apresente essa temperatura em graus Fahrenheit, sua conversão. A fórmula de conversão é " $F \leftarrow C * 9 / 5 + 32$ ", sendo "F" a temperatura em Fahrenheit e "C" a temperatura em Celsius. Armazene em memória o resultado calculado. Use variáveis locais. Formate a saída numérica com duas casas decimais.
4. Criar procedimento chamado **CAP0504** que efetue a leitura de uma temperatura em graus Fahrenheit apresente essa temperatura em graus Celsius, sua conversão. A fórmula de conversão é " $C \leftarrow ((F - 32) * 5) / 9$ ", sendo "F" a temperatura em Fahrenheit e "C" a temperatura em Celsius. Armazene em memória o resultado calculado. Use variáveis locais. Formate a saída numérica com duas casas decimais.
5. Criar procedimento chamado **CAP0505** que efetue a leitura de dois valores numéricos inteiros (representados pelas variáveis locais "A" e "B") e mostre o resultado armazenado em memória do quadrado da diferença do primeiro valor (variável "A") em relação ao segundo valor (variável "B") junto a variável local "R".
6. Criar procedimento chamado **CAP0506** que efetue a leitura de um número inteiro qualquer em uma variável local e multiplique este número por "2" armazenando o resultado em memória. Apresentar o resultado da multiplicação somente se o resultado for maior que "30".
7. Criar procedimento chamado **CAP0507** que efetue a leitura de dois valores numéricos reais representados pelas variáveis locais "A" e "B" e apresente o resultado da diferença do maior valor pelo menor valor sem armazenar o resultado em memória.
8. Criar procedimento chamado **CAP0508** que efetue a leitura de dois valores numéricos reais representados pelas variáveis locais "A" e "B" e apresente o resultado da diferença do maior valor pelo menor valor com armazenamento do cálculo em memória.
9. Criar procedimento chamado **CAP0509** que efetue a leitura de três valores numéricos inteiros desconhecidos representados pelas variáveis locais "A", "B" e "C". O procedimento deve somar esses valores, armazenar o resultado em memória e apresentar este resultado somente se for "**maior ou igual**" a "100".
10. Criar procedimento chamado **CAP0510** que apresente a soma dos cem primeiros números naturais "**(1 + 2 + 3 + ... + 98 + 99 + 100)**" utilizando-se a primitiva "**REPITA**".
11. Criar procedimento chamado **CAP0511** que apresente a soma dos cem primeiros números naturais "**(1 + 2 + 3 + ... + 98 + 99 + 100)**" utilizando-se a primitiva "**ENQUANTO**".
12. Criar procedimento chamado **CAP0512** que apresente a soma dos cem primeiros números naturais "**(1 + 2 + 3 + ... + 98 + 99 + 100)**" utilizando-se a primitiva "**PARA**".
13. Criar procedimento chamado **CAP0513** que apresente o somatório dos *valores pares* existentes na faixa de "1" até "500". Use a primitiva "**REPITA**".
14. Criar procedimento chamado **CAP0514** que apresente o somatório dos *valores pares* existentes na faixa de "1" até "500". Use a primitiva "**ENQUANTO**".

15. Criar procedimento chamado **CAP0515** que apresente o somatório dos *valores pares* existentes na faixa de "1" até "500". Use a primitiva "**PARA**".
16. Criar procedimento chamado **CAP0516** que mostre todos os valores numéricos divisíveis por "4" *menores* que "20" iniciando a contagem em "1". Use a primitiva "**REPITA**".
17. Criar procedimento chamado **CAP0517** que mostre todos os valores numéricos divisíveis por "4" *menores* que "20" iniciando a contagem em "1". Use a primitiva "**ENQUANTO**".
18. Criar procedimento chamado **CAP0518** que mostre todos os valores numéricos divisíveis por "4" *menores* que "20" iniciando a contagem em "1". Use a primitiva "**PARA**".
19. Criar procedimento chamado **CAP0519** que apresente os valores numéricos inteiros compreendidos na faixa de "0" e "4". Use a primitiva "**REPITA**".
20. Criar procedimento chamado **CAP0520** que apresente os valores numéricos inteiros compreendidos na faixa de "0" e "4". Use a primitiva "**ENQUANTO**".
21. Criar procedimento chamado **CAP0521** que apresente os valores numéricos inteiros compreendidos na faixa de "0" e "4". Use a primitiva "**PARA**".
22. Criar procedimento chamado **CAP0522** que apresente os valores numéricos inteiros compreendidos na faixa de "0" até "15" de "3" em "3". Use a primitiva "**REPITA**".
23. Criar procedimento chamado **CAP0523** que apresente os valores numéricos inteiros compreendidos na faixa de "0" até "15" de "3" em "3". Use a primitiva "**ENQUANTO**".
24. Criar procedimento chamado **CAP0524** que apresente os valores numéricos inteiros compreendidos na faixa de "0" até "15" de "3" em "3". Use a primitiva "**PARA**".

# ANOTAÇÕES

---

## Apêndice A - Exemplos geométricos

Neste livro foram apresentados exemplos de criação de figuras geométricas. Nesta parte, são indicados outros exemplos de figuras geradas a partir do uso de instruções diretas ou de procedimentos isolados ou usados como apoio para instruções diretas. O objetivo deste material é ampliar seu conhecimento e fornecer subsídios para aumentar sua criatividade. Algumas das imagens apresentadas são reproduções de códigos de sítios ou manuais antigos da linguagem Logo: Petti (2021), Muller (1998), Harvey (1997), Corrales Mora (1996), Sparer (1984), Winter (1984), Abelson (1984), ATARI (1983), Kheriaty & Gerhold (1982), Bass (2002), Erfan's (2021) e Joys (2021).

### INSTRUÇÕES DIRETAS COM E SEM AUXILIO DE PROCEDIMENTO

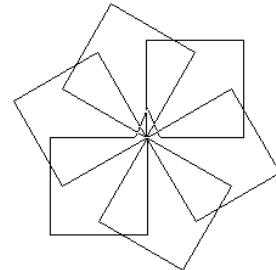
Parte das figuras apresentadas usarão o procedimento coringa chamado **FIGURAS** (figuras geométricas simples) que poderá desenhar imagens, de triângulos a circunferências.

```
APRENDA FIGURAS :LADOS :TAMANHO  
REPITA :LADOS [PF :TAMANHO PD 360 / :LADOS]  
FIM
```

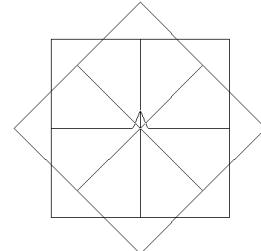
Desta forma, caso deseje a imagem de um quadrado execute "**FIGURAS 4 80**", caso deseje a imagem de um triângulo execute "**FIGURAS 3 80**", desejando uma circunferência execute "**FIGURAS 360 1**" e assim por diante.

A seguir são apresentadas as instruções e as imagens geradas por essas instruções. Atente para cada detalhe pois isto poderá ajudar na resolução dos exercícios.

```
LD  
REPITA 6 [PD 60 FIGURAS 4 80]
```

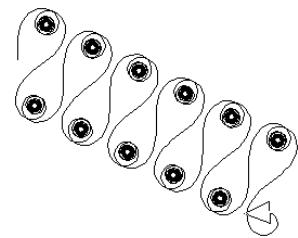


```
LD  
REPITA 8 [FIGURAS 4 120 PD 45]
```



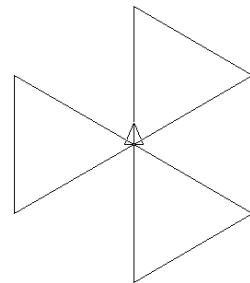
LD

PARA [I 0 2000] [PF 5 PD (90 \* SEN :I)]



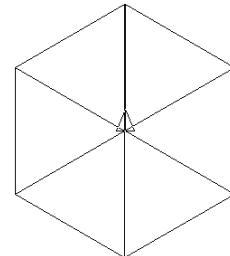
LD

REPITA 3 [FIGURAS 3 150 PD 120]



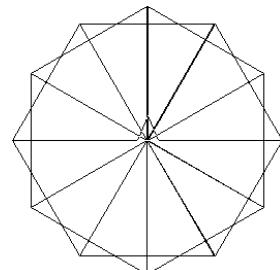
LD

REPITA 6 [FIGURAS 3 140 PD 60]



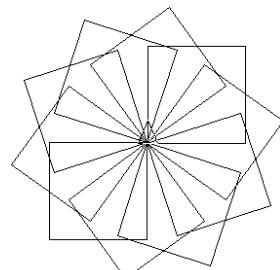
LD

REPITA 360 / 30 [FIGURAS 3 130 PD 30]

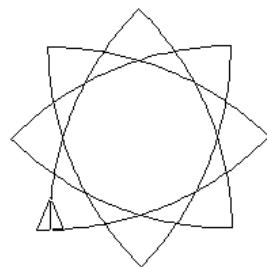


LD

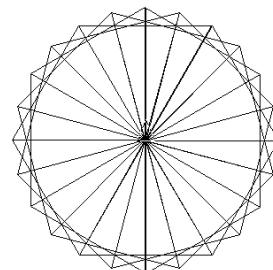
REPITA 10 [REPITA 4 [PF 100 PD 90] PD 36]



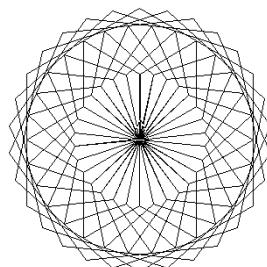
LD  
REPITA 8 [REPITA 45 [PF 4 PD 1] PD 90]



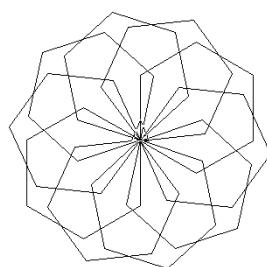
LD  
REPITA 24 [REPITA 3 [PF 150 PD 120] PD 15]



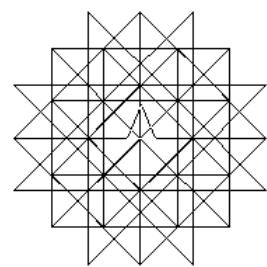
LD  
REPITA 30 [REPITA 6 [PF 75 PD 60] PD 12]



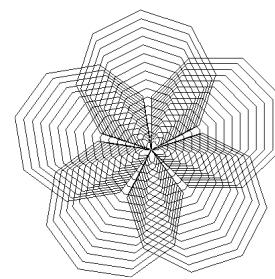
LD  
REPITA 10 [REPITA 6 [PF 75 PD 60] PD 36]



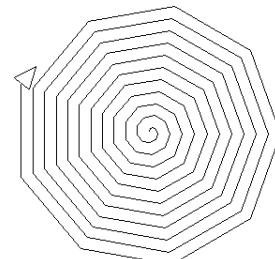
LD  
REPITA 8 [REPITA 8 [PE 135 PF 90] PE 45]



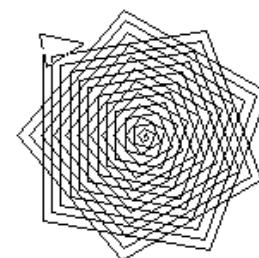
```
LD  
PARA [I 10 80 5] [  
    REPITA 5 [  
        REPITA 8 [  
            PF :I PD 45  
        ]  
        PD 72  
    ]  
]
```



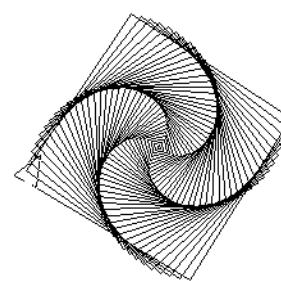
```
LD  
REPITA 100 [PF CV PD 40]
```



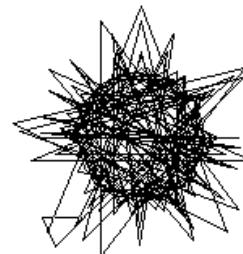
```
LD  
REPITA 100 [PF CV PD 80]
```



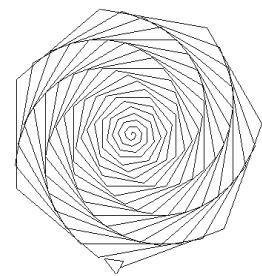
```
LD  
REPITA 150 [PF CV PD 89]
```



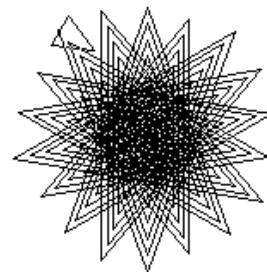
```
LD  
REPITA 150 [PF CV PD CV * 1.5]
```



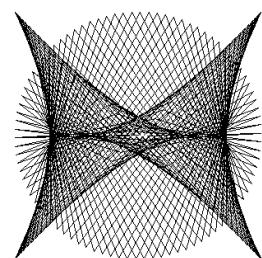
LD  
REPITA 150 [PF CV PD 50]



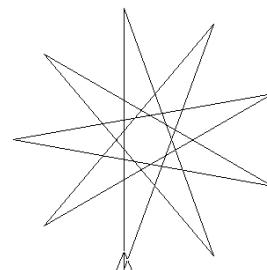
LD  
REPITA 150 [PF CV PD 500]



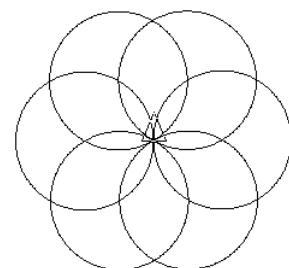
LD  
REPITA 360 [  
  MUDEXY (  
    SEN(89 \* CV)) \* 150 (SEN(179 \* CV)) \* 150  
]



LD  
REPITA 9 [PF 300 PD 160]

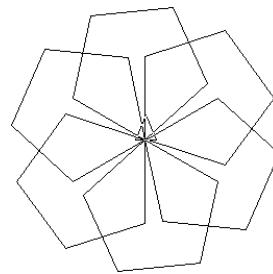


LD  
REPITA 6 [PD 60 FIGURAS 360 1]



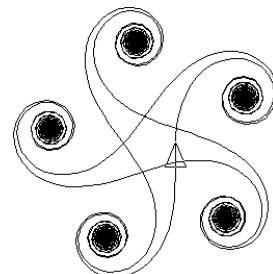
LD

REPITA 6 [PD 60 FIGURAS 5 80]



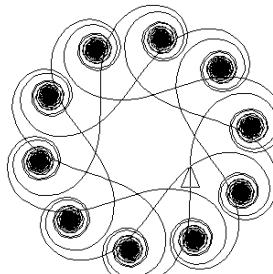
LD

REPITA 1800 [PF 10 PD CV + .1]



LD

REPITA 3600 [PF 10 PD CV + .2]

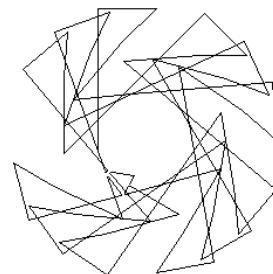


LD

REPITA 12 [

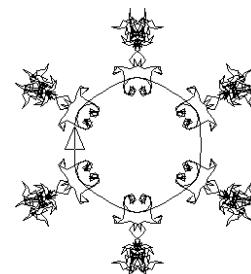
PF 120 PD 90 PF 50 PD 135 PF 40  
PE 185 PT 65 PD 45 PF 50

]

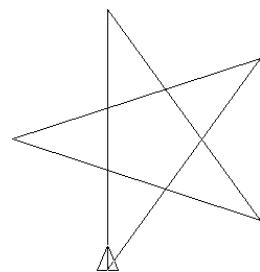


LD

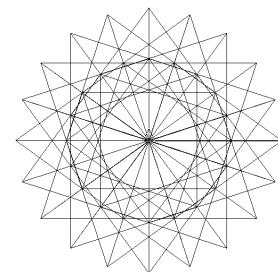
PARA [I 0 1002] [  
PF 8 MUDEDÇ (360 \* (POTÊNCIA :I 3) / 1002)  
]



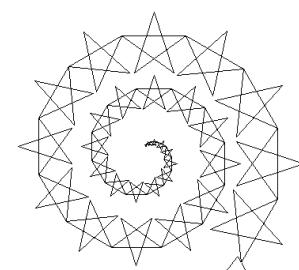
LD  
REPITA 5 [PF 250 PD 144]



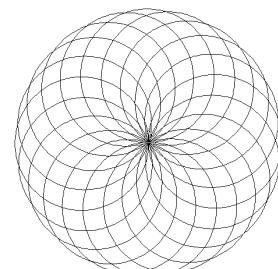
LD  
REPITA 20 [REPITA 5 [PF 250 PD 144] PD 18]



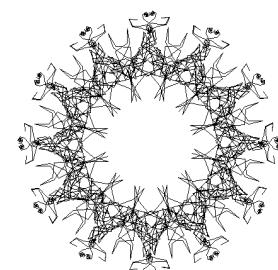
LD  
PARA [I 0 95 3] [  
  REPITA 5 [  
    PF :I PD 144  
  ]  
  PF :I PD 30  
]



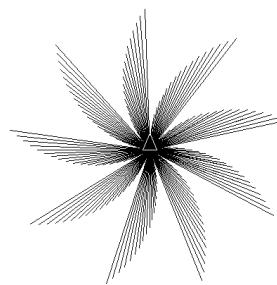
LD  
REPITA 20 [REPITA 180 [PF 4 PD 2] PD 18]



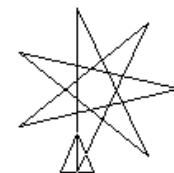
LD  
PARA [I 0 2200] [  
  PF (25 \* SEN :I) PD (POTÊNCIA :I 2)  
]



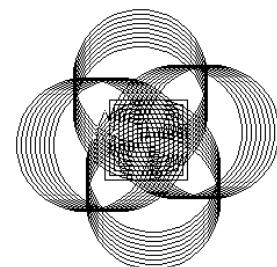
```
LD  
REPITA 9 [  
  PARA [I 10 200 10] [  
    PF :I PT :I PD 2  
  ]  
]
```



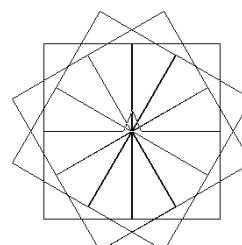
```
LD  
REPITA 7 [PF 100 PD 360 * 3 / 7]
```



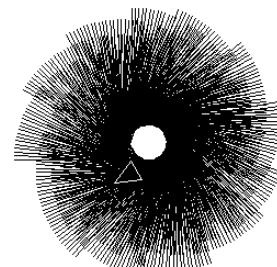
```
LD  
REPITA 36 [  
  REPITA 36 [  
    PF 10 PD 10  
  ]  
  PF CV PD 90 PF CV  
]
```



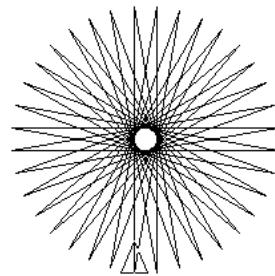
```
LD  
REPITA 12 [REPITA 4 [FIGURAS 4 100] PD 30]
```



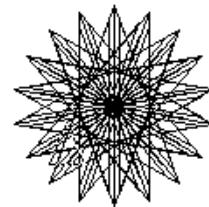
```
LD  
REPITA 12 [REPITA 55 [PF 100 PT 100 PD 2] PF 45]
```



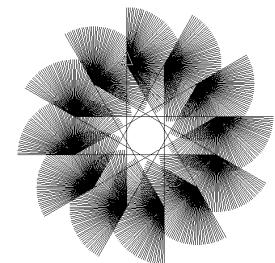
LD  
REPITA 54 [REPITA 8 [PF 200 PD 170]]



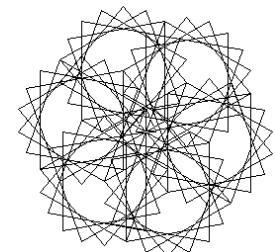
LD  
REPITA 18 [REPITA 5 [PD 40 PF 100 PD 120] PD 20]



LD  
REPITA 12 [  
REPITA 75 [  
PF 100 PT 100 PD 2  
]  
PF 250  
]



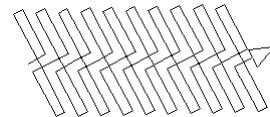
LD  
REPITA 12 [REPITA 360 [PF 100 PD 100] PD 60]



### INSTRUÇÕES A PARTIR DA DEFINIÇÃO DE PROCEDIMENTOS

Os scripts a seguir são baseados e adaptado a partir da bibliografia usada além de material instrucional nos sítios <https://fmslogo.sourceforge.io/workshop/> e <https://helloacm.com/logo-turtle-tutorial-how-to-draw-fractal-stars/>.

```
APRENDA SEGMENTO
PF 20 PE 90
PF 50 PE 90
PF 10 PE 90
PF 55
PF 55 PD 90
PF 10 PD 90
PF 50 PD 90
PF 20
FIM
```

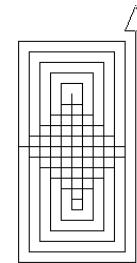


```
APRENDA PADRAO
PD 62
REPITA 10 [ SEGMENTO ]
FIM
```

EXECUTE: LD PADRAO

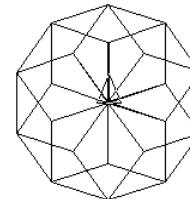
```
APRENDA CAMINHO
REPITA 22 [
  PD 90
  PF 110 - CV * 10
  PD 90
  PF CV * 10
]
FIM
```

EXECUTE: LD CAMINHO



```
APRENDA HEXAFLOR :PETALAS
REPITA :PETALAS [
  FIGURAS 5 50
  PD 360 / :PETALAS
]
FIM
```

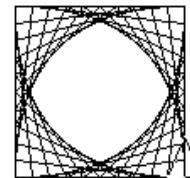
EXECUTE: LD HEXAFLOR 10



```
APRENDA CICLO :INDICE :ULTIMO
MUDEXY :INDICE      0
MUDEXY :ULTIMO      :INDICE
MUDEXY (:ULTIMO - :INDICE) :ULTIMO
MUDEXY 0             (:ULTIMO - :INDICE)
MUDEXY :INDICE      0
FIM
```

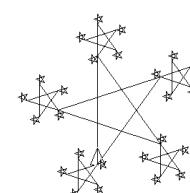
```
APRENDA QUADART
REPITA 10 [CICLO CV * 10 100]
FIM
```

EXECUTE: LD QUADART



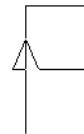
```
APRENDA ESTRELANDO :TAMANHO :LIMITE
SE :TAMANHO < :LIMITE [PARE]
REPITA 5 [PF :TAMANHO ESTRELANDO
          :TAMANHO * .3 :LIMITE PD 144]
FIM
```

EXECUTE: LD ESTRELANDO 150 10



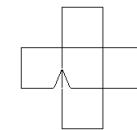
APRENDA BANDEIRA  
PF 50  
FIGURAS 4 50  
FIM

EXECUTE: LD BANDEIRA



APRENDA CRUZ  
REPITA 4 [BANDEIRA PD 90]  
FIM

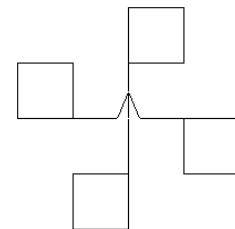
EXECUTE: LD CRUZ



APRENDA VOLTABANDA  
BANDEIRA  
PT 50  
FIM

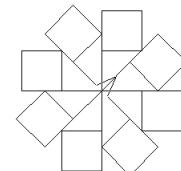
APRENDA BANDEIRAS  
REPITA 4 [VOLTABANDA PD 90]  
FIM

EXECUTE: LD BANDEIRAS



APRENDA MUITASBANDEIRAS  
BANDEIRAS  
PD 45  
BANDEIRAS  
FIM

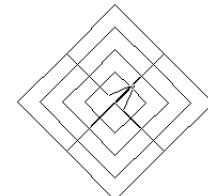
EXECUTE: LD MUITASBANDEIRAS



APRENDA QUADRADOS  
FIGURAS 4 20  
FIGURAS 4 35  
FIGURAS 4 50  
FIGURAS 4 65  
FIM

APRENDA DIAMANTES  
PD 45  
REPITA 4 [QUADRADOS PD 90]  
FIM

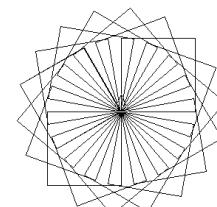
EXECUTE: LD DIAMANTES



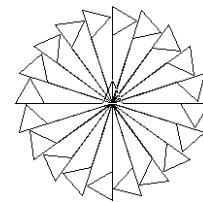
APRENDA QUADRADO  
REPITA 4 [FIGURAS 4 100]  
FIM

APRENDA FLORQUADRADA  
REPITA 18 [QUADRADO PD 20]  
FIM

EXECUTE: LD FLORQUADRADA



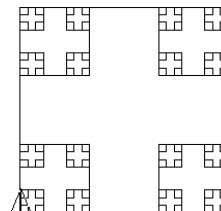
```
APRENDA BANDTRI
PF 100 PD 120
PF 30 PD 120
PF 30 PD 120
PT 70
FIM
```



```
APRENDA FLORBANDTRI
REPITA 20 [BANDTRI ESPERE 30 PD 18]
FIM
```

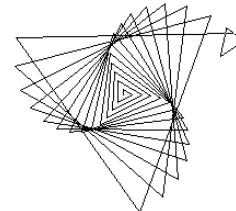
EXECUTE: LD FLORBANDTRI

```
APRENDA QUADRICULADO :VALOR
SE (:VALOR < 3) [
PARE
]
REPITA 4 [
QUADRICULADO :VALOR / 3
PF :VALOR
PD 90
]
FIM
```



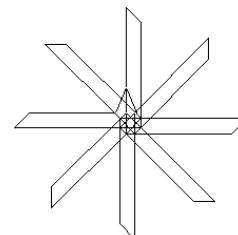
EXECUTE: LD QUADRICULADO 200

```
APRENDA TUNELTRI :TAMANHO :ANGULO
SE (:TAMANHO > 200) [
PARE
]
PF :TAMANHO
PD :ANGULO
TUNELTRI :TAMANHO + 5 :ANGULO + 0.12
FIM
```



EXECUTE: LD TUNELTRI 5 120

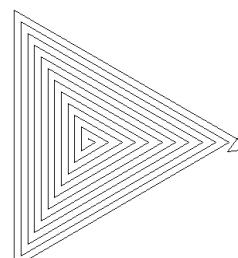
```
APRENDA PA
REPITA 2 [PF 100 PD 135 PF 20 PD 45]
FIM
```



```
APRENDA HELICE
REPITA 8 [PA PD 135 PF 20]
FIM
```

EXECUTE: LD HELICE

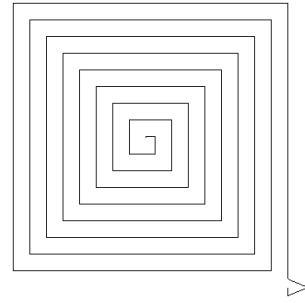
```
APRENDA ESPIRALTRI :LADO
SE (:LADO > 350) [
PARE
]
PF :LADO ESPERE 20
PD 120 ESPERE 20
ESPIRALTRI :LADO + 10 ESPERE 30
FIM
```



EXECUTE: LD ESPIRALTRI 1

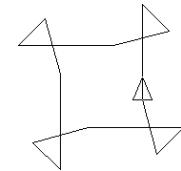
```
APRENDA ESPIRALQUAD :LADO
SE (:LADO > 350) [
    PARE
]
PF :LADO ESPERE 20
PD 90 ESPERE 20
ESPIRALQUAD :LADO + 10 ESPERE 30
PD 90 ESPERE 50
FIM
```

EXECUTE: LD ESPIRALQUAD 1



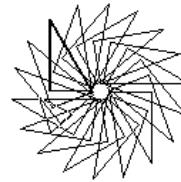
```
APRENDA FORMA
PF 100 PD 135
PF 40 PD 120
PF 60 PD 15
FIM
```

EXECUTE: LD REPITA 4 [FORMA]



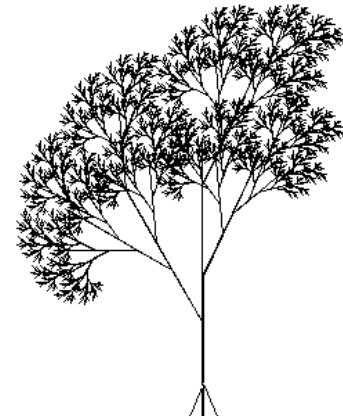
```
APRENDA FORMATRI
PF 50 PD 150
PF 60 PD 100
PF 30 PD 90
FIM
```

EXECUTE: LD REPITA 20 [FORMATRI]



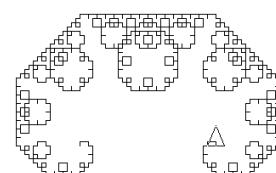
```
APRENDA ARVORE :TAMANHO
SE :TAMANHO < 5 [PF :TAMANHO PT :TAMANHO PARE]
PF :TAMANHO / 3
PE 30
ARVORE :TAMANHO * 2 / 3
PD 30
PF :TAMANHO / 6
PD 25
ARVORE :TAMANHO / 2
PE 25
PF :TAMANHO / 3
PD 25
ARVORE :TAMANHO / 2
PE 25
PF :TAMANHO / 6
PT :TAMANHO
FIM
```

EXECUTE: LD ARVORE 200



```
APRENDA CURVAC :TAMANHO :NIVEL
SE :NIVEL = 0 [PF :TAMANHO PARE]
CURVAC :TAMANHO :NIVEL - 1 PD 90
CURVAC :TAMANHO :NIVEL - 1 PE 90
FIM
```

EXECUTE: LD CURVAC 5 10



Grave este conteúdo com o nome "Apendice\_A".

## **ANOTAÇÕES**

## Apêndice B - Música

A linguagem Logo utilizada no ambiente "xLogo" possui a capacidade de tocar músicas a partir da definição de notas musicais. Para que esse efeito seja produzido é necessário usar duas primitivas: **SEQUÊNCIA (SEQ)** para armazenar as notas musicais em uma lista na memória e **TOQUE** para executar as notas armazenadas.

O ambiente opera com as notas da oitava na escala da clave de Sol, sendo: **DÓ, RÉ, MI, FÁ, SOL, LÁ e SI**. Para obter a nota **DÓ** mais aguda informe **DÓ+** ou mais grave informe **DÓ-** e assim para as demais notas. Para subir ou descer uma oitava, use respectivamente o símbolo ":" seguido dos símbolos "+" (mais) ou "-" (menos). Por exemplo, o indicativo "**:++**" diz que a sequência das notas a serem tocadas estão duas oitavas acima. Por padrão, as notas musicais são tocadas com duração de tempo "**1**". Caso deseje alterar a duração de uma nota é necessário a sua frente definir o valor de duração. Por exemplo, a instrução "**SEQUÊNCIA [DÓ RÉ MI FÁ 0.5 SOL LA SI]**" ser tocada com as notas **DÓ, RÉ, MI** e **FÁ** em tempo de duração "**1**" e as notas **SOL LA SI** em tempo de duração "**0.5**".

Para auxiliar o uso de sons musicais o ambiente "xLogo" oferece algumas primitivas de apoio além das primitivas **SEQUÊNCIA (SEQ)** e **TOQUE**, a saber:

PRIMITIVA	AÇÃO
<b>DELETESEQÜÊNCIA (DELSEQ)</b>	<b>Apaga sequência de notas memorizada.</b>
<b>INTRUMENTO (INSTR)</b>	<b>Mostra o número do instrumento em uso.</b>
<b>MUDEÍNDICESEQÜÊNCIA (MUDEINDSEQ)</b>	<b>Pula do início da sequência de notas a quantidade de notas indicadas.</b>
<b>MUDEINSTRUMENTO (MUDEINSTR)</b>	<b>Muda o instrumento em uso, sendo 235 opções. Veja a partir do menu: Ferramentas/Preferências/Sons.</b>

Como exemplo, considere um procedimento chamado **DOREMI** que toque a sequencia de notas "**DÓ-RÉ-MI-FÁ-FÁ-FÁ / DÓ-RÉ-DÓ-RÉ-RÉ-RÉ / DÓ-SOL-FÁ-MI-MI-MI / DÓ-RÉ-MI-FÁ-FÁ-FÁ**". Desta forma, acione o botão "**Editor**" e informe o seguinte código:

**APRENDA DOREMI**

**DELSEQ**

**SEQ [DÓ RÉ MI FÁ FÁ FÁ]**

**SEQ [DÓ RÉ DÓ RÉ RÉ RÉ]**

**SEQ [DÓ SOL FÁ MI MI MI]**

**SEQ [DÓ RÉ MI FÁ FÁ FÁ]**

**TOQUE**

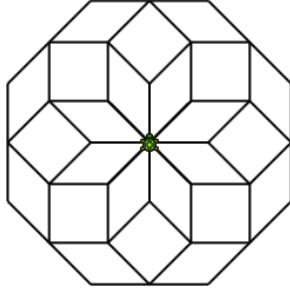
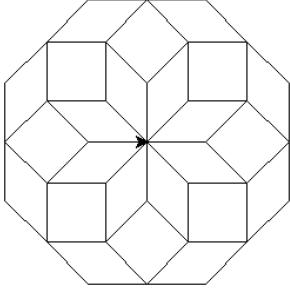
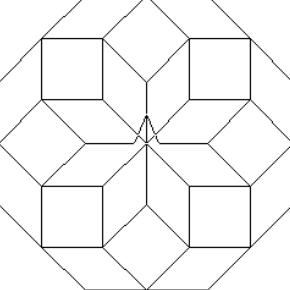
**FIM**

Para encerrar o modo **Editor** acione o botão "**Guardar e sair do editor**". Execute o procedimento **DOREMI** e ouça o som tocado. O uso da primitiva **ESPERE** é uma forma de criar um tempo de silêncio para a música sendo tocada. Grave este conteúdo com o nome "**Apendice\_B**".

Para mais detalhes consulte <http://xlogo.tuxfamily.org/pt/files/musica.htm>.

## Apêndice C - Espiral hexagonal (imagem da capa)

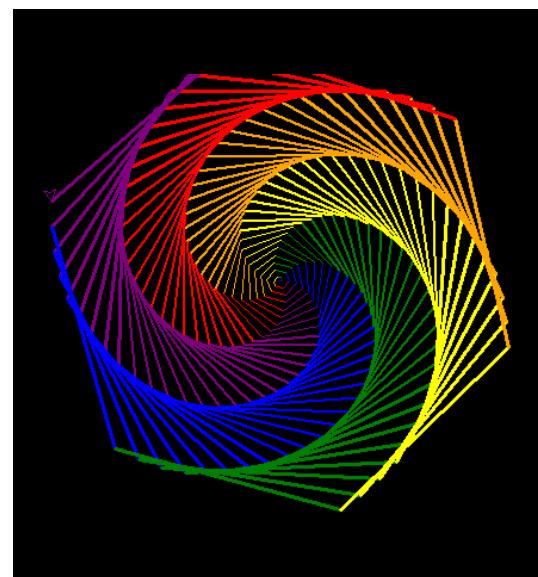
Muitos dos recursos de uso da linguagem Logo podem ser suportados em outras linguagens de programação. Por exemplo, a linguagem "Small Basic" desenvolvida pela empresa Microsoft (<https://smallbasic-publicwebsite.azurewebsites.net>) e "Python" produzida pela comunidade Python (<https://www.python.org>), além do processador de textos "Writer" do "LibreOffice". Observe os códigos escritos em "Small Basic", "Python" e "Logo" e a imagem gerada:

SMALL BASIC	PYTHON	LOGO (xLogo)
<pre>For I = 1 To 8   For J = 1 To 8     Turtle.Move(50)     Turtle.Turn(45)   EndFor   Turtle.Turn(45) EndFor</pre>	<pre>import turtle for I in range(8):   for J in range(8):     turtle.fd(50)     turtle.rt(45)   turtle.rt(45) input()</pre>	<pre>PARA [I 1 8] [   PARA [J 1 8] [     PF 50     PD 45   ]   PD 45 ]</pre>
		

Onde, o indicativo "freq" refere-se a uma frequência de som medida em hertz (vibração do som por segundo) e "duração" o tempo de execução do som em milissegundos, ambos indicados como valores numéricos inteiros. Quanto maior for a frequência mais agudo o som será.

A imagem da capa deste livro é originalmente apresentada no sítio "ProgrammerSough" a partir do endereço (<https://www.programmersought.com/article/92794745509/>), tendo sido produzida por meio da linguagem "Python" de acordo com o código seguinte adaptado:

```
#Drawing colorful spirals
import turtle
def draw_spin():
  colores = [
    'red', 'purple', 'blue',
    'green', 'yellow', 'orange'
  ]
  turtle.bgcolor('black')
  for x in range(200):
    turtle.pencolor(colores[x % 6])
    turtle.width(x / 100 + 1)
    turtle.forward(x)
    turtle.left(59)
draw_spin()
input()
```

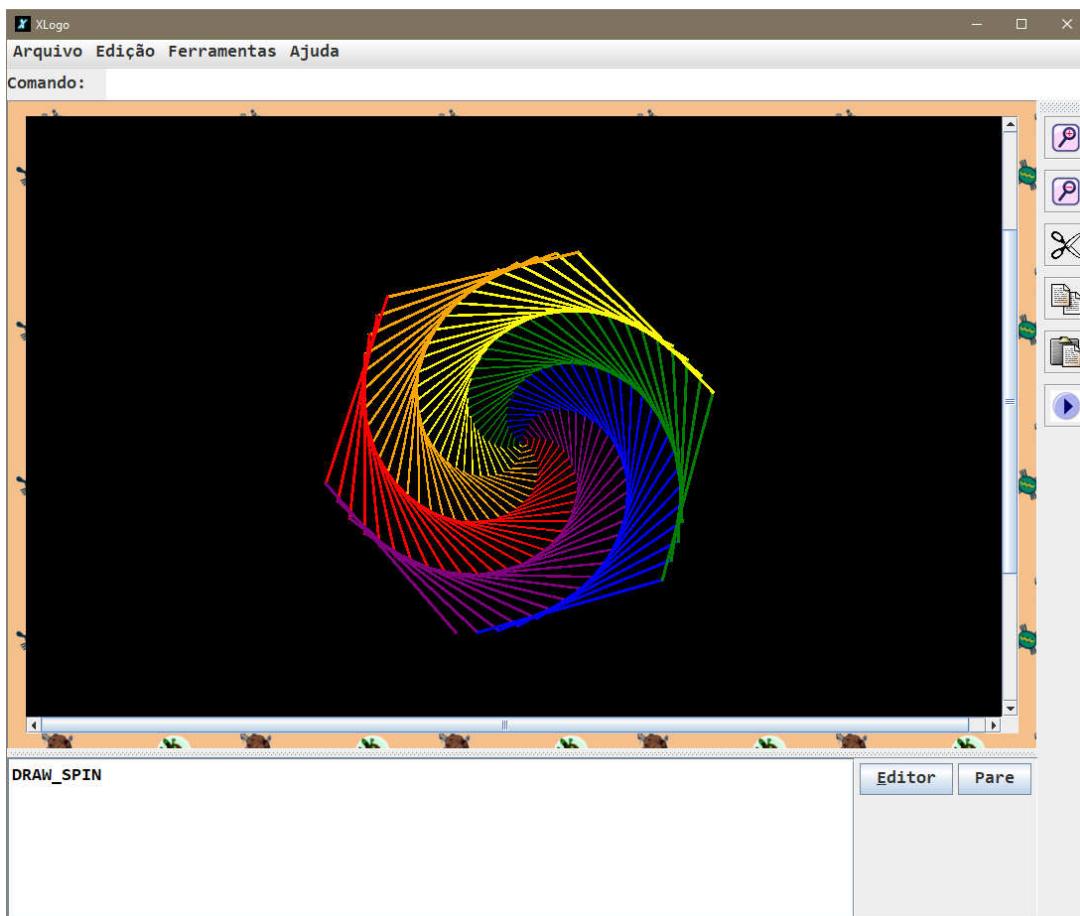


A partir do código "Python" foi realizada a transliteração e escrita de um código compatível em "Logo" que gerou a imagem usada na capa deste trabalho com o código:

## APRENDA DRAW\_SPIN

```
LD AT
MUDEC [000 000 000] # PRETO
PARA [X 0 199] [
  SE (RESTO :X 6) = 0 [MUDECL [255 000 000]] # 0 - VERMELHO
  SE (RESTO :X 6) = 1 [MUDECL [128 000 128]] # 1 - ROXO
  SE (RESTO :X 6) = 2 [MUDECL [000 000 255]] # 2 - AZUL
  SE (RESTO :X 6) = 3 [MUDECL [000 128 000]] # 3 - VERD
  SE (RESTO :X 6) = 4 [MUDECL [255 255 000]] # 4 - AMARELO
  SE (RESTO :X 6) = 5 [MUDECL [255 165 000]] # 5 - LARANJA
  MUDEESPRESSURADOLÁPIS (:X / 100) + 1
  PF :X
  PE 59
]
DT
MUDEEL 0
FIM
```

O código do procedimento "DRAW\_SPIN" produzido em "Logo" ao ser executado mostra imagem semelhante ao código original produzido em "Python".



Veja o que realiza cada instrução do procedimento "DRAW\_SPIN" a partir das primitivas *Logo* e sua relação com os comandos *Python*:

- "APRENDA" e "FIM" são usadas para definir o escopo de escrita do programa no procedimento a partir do nome indicado a frente de "APRENDA", sendo isso compatível ao comando "**def**";
- "LD" e "AT" efetuam respectivamente a limpeza da tela e a apresentação do ícone da tartaruga não tendo nenhuma relação direta com o código *Python* indicado;
- "MUDECF" efetua a mudança da cor do fundo da área de trabalho para a cor preto de acordo com o código "**RGB [000 000 000]**" similar a instrução "`turtle.bgcolor('black')`";
- "PARA" efetua a execução de duzentas passagens do grupo de instruções subordinadas contadas de "0" até "199" antes de encontrar a primitiva "OT" estando de acordo com a instrução "`for x in range(200):`" que efetivamente faz contagem de "0" até "199";
- Dentro do escopo de ação da primitiva "PARA" encontra-se definida uma sequência de instruções "SE" que detectam os valores do resto da divisão do conteúdo da variável "X" por "6" (que é a quantidade de cores em uso) que geram valores de resto entre "0" e "5" para a detecção e uso da cor definida para cada linha traçada do hexágono, estando este conjunto de instruções consoantes a instrução "`turtle.pencolor(colores[x % 6])`". Logo não opera com o uso de variáveis compostas (matrizes) como *Python* "`colores[x % 6]`" por esta razão o uso das primitivas "SE" é necessário;
- "MUDEESPRESSURADOLÁPIS INTEIRO (:X / 100 + 1)" tem por finalidade a partir da primitiva "MUDEESPRESSURADOLÁPIS" ou "MUDEEL" mudar a largura do traço do desenho em andamento estando está instrução em consonância com a instrução "`turtle.width(x/100 + 1)`";
- "PF :X" faz a apresentação do traço com valores crescentes na variável "X" de "0" até "199" sendo compatível com a instrução "`turtle.forward(x)`";
- "PE 59" faz o giro em graus do traço para a formação de uma figura hexagonal, pois "59" é o valor numérico mais próximo de "60" que são os graus de formação de um hexágono estando de acordo com a instrução "`turtle.left(59)`";
- As demais instruções "DT" e " MUDEEL 0" efetuam respectivamente o ocultamento da tartaruga e o retorno do traço ao seu modo padrão, não tendo nenhuma relação com as demais instruções do código em *Python* que não vem ao caso.

Grave este conteúdo com o nome "**Apendice\_E**".

# ANOTAÇÕES

---

## Apêndice D - Gabarito

### CAPÍTULO 3

---

- Quais são as figuras geométricas planas desenhadas a partir das seguintes instruções? Diga qual é a figura sem executar a instrução no ambiente de programação.

REPITA 4 [PARAFRENTE 100 PARADIREITA 90] Quadrado

REPITA 5 [PF 100 PE 72] Pentágono

REPITA 3 [PT 100 PD 120] Triângulo

REPITA 36 [PF 20 PD 10] Circunferência

- Criar procedimento chamado **RETANGULO1** (sem acento) que desenhe um retângulo com lados de tamanhos **60** e **100** para frente com giro de gruas para à direita. O procedimento deve desenhar a imagem sem o uso do recurso **REPITA**.

**APRENDA RETANGULO1**

PF 60

PD 90

PF 100

PD 90

PF 60

PD 90

PF 100

PD 90

FIM

- Criar procedimento chamado **RETANGULO2** (sem acento) que desenhe um retângulo com lados de tamanhos **60** e **100** para trás com giro de gruas à esquerda. Usar a primitiva **REPITA**.

**APRENDA RETANGULO2**

REPITA 2 [

PT 60

PE 90

PT 100

PE 90

]

FIM

4. Criar, sem uso da primitiva **REPITA**, procedimento chamado **PENTAGONO** (sem acento) que construa um pentágono com tamanho **40**. Avance para frente com giro a direita.

**APRENDA PENTAGONO**

```
PF 40  
PD 72  
PF 40  
PD 72
```

**FIM**

5. Criar procedimento chamado **DECAGONO** (sem acento) que construa uma figura com dez lados a partir do uso da primitiva **REPITA** com giro de graus à esquerda com avanço a gente de **30**.

**APRENDA DECAGONO**

```
REPITA 10 [PF 30 PE 36]
```

**FIM**

---

CAPÍTULO 4

---

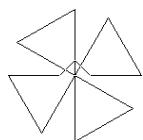
- Criar procedimento chamado **RETAMETA** que desenhe um retângulo cujo lado menor (comprimento) seja a metade do lado maior (altura) onde o tamanho informado deverá ser fornecido como parâmetro. Movimente o desenho para frente com sentido a direita.

**APRENDA RETAMETA :TAMANHO**  
**REPITA 2 [**  
**PF :TAMANHO PD 90**  
**PF :TAMANHO / 2 PD 90**  
**]**  
**FIM**

- Crie um procedimento chamado **TRIANGEX**, que apresente um triângulo equilátero com lado de tamanho **70** para trás a partir de giros a esquerda.

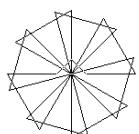
**APRENDA TRIANGEX**  
**REPITA 3 [**  
**PT 70 PE 120**  
**]**  
**FIM**

- Criar procedimento chamado **FLORTRIG** desenhado a partir do procedimento **TRIANGEX** com giro a direita de modo que tenha a seguinte aparência.



**APRENDA FLORTRIG**  
**REPITA 4 [**  
**TRIANGEX PD 90**  
**]**  
**FIM**

- Criar procedimento chamado **VENTITRIG** desenhado a partir do procedimento **TRIANGEX** com giro a esquerda de modo que tenha a seguinte aparência.

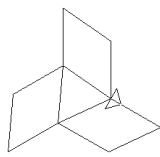


**APRENDA VENTITRIG**  
**REPITA 8 [**  
**TRIANGEX PE 45**  
**]**  
**FIM**

- Criar procedimento chamado **LOSANGO** que desenhe imagem de mesmo nome com tamanho fornecido por parâmetro a partir de giro a direita.

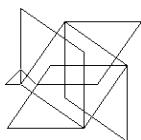
**APRENDA LOSANGO :TAMANHO**  
**REPITA 2 [**  
**PF :TAMANHO PD 125**  
**PF :TAMANHO PD 55**  
**]**  
**PE 55 PT :TAMANHO PD 55**  
**FIM**

6. A partir do procedimento **LOSANGO** crie procedimento chamado **FLORLOSA1** de modo que seja apresentada a figura a seguir com tamanho **80**.



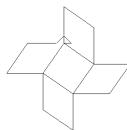
APRENDA FLORLOSA1  
PD 120  
REPITA 3 [  
LOSANGO 80 PD 120  
]  
FIM

7. A partir do procedimento **LOSANGO** crie procedimento chamado **FLORLOSA2** de modo que seja apresentada a figura a seguir com tamanho **80**.



APRENDA FLORLOSA2  
REPITA 4 [  
LOSANGO 80 PE 90  
]  
FIM

8. A partir do procedimento **LOSANGO** crie procedimento chamado **FLORLOSA3** de modo que seja apresentada a figura a seguir com tamanho **80**.



APRENDA FLORLOSA3  
REPITA 4 [  
LOSANGO 80 PD 90  
]  
FIM

9. Sem executar no computador descrimine qual imagem é apresentada.

REPITA 12 [REPITA 3 [PF 50 PD 120] PD 30]

A imagem apresentada baseia-se em um quadrado repetido três vezes que após ser

---

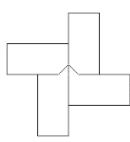
Rotacionado a trinta graus é repetido mais doze vezes.

---

10. Criar procedimento chamado **RETANGULO3** com tamanhos **100** (altura) e **50** (largura). Movimente-se para frente com giro a direita.

APRENDA RETANGULO3  
REPITA 2 [  
PF 100  
PD 90  
PF 50  
PD 90  
]  
FIM

11. Criar procedimento chamado **CATAVENTO1** com o formato da figura seguinte a partir do procedimento **RETANGULO3**.



APRENDA CATAVENTO1  
REPITA 4 [  
RETANGULO3 PD 90  
]  
FIM

12. Criar procedimento chamado **TRIANGPR** que desenhe um triângulo equilátero com tamanho definido por parâmetro com comando **PARA** contando de **0** a **2** no sentido a frente com giro a direita.

APRENDA TRIANGPR :TAMANHO  
PARA [I 0 2] [  
PF :TAMANHO PD 120  
]  
FIM

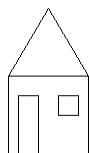
13. Criar procedimento chamado **QUADRADO1** que desenhe um quadrado com lado definido por parâmetro a partir do uso da primitiva **ENQUANTO**. Conte de **1** a **4**.

APRENDA QUADRADO1 :TAMANHO  
ATR "I 1  
ENQUANTO [:I <= 4] [  
PF :TAMANHO PD 90  
ATR "I :I + 1  
]  
FIM

14. Criar procedimento chamado **QUADRADO2** que desenhe um quadrado com lado definido por parâmetro a partir do uso da primitiva **PARA**. Conte de **1** a **4**.

APRENDA QUADRADO2 :TAMANHO  
PARA [I 1 4] [  
PF :TAMANHO PD 90  
]  
FIM

15. Criar procedimento chamado **CASINHA** que mostre a imagem da casa. O quadrado e o triângulo deverão possuir tamanho **80**, a porta é um retângulo de **60** por **20** e a janela é um quadrado de tamanho **20**.

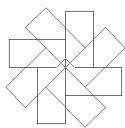


```

APRENDA CASINHA
UL
PARA [I 1 4] [PF 80 PD 90]
PF 80 PD 90
PARA [I 1 2] [PF 80 PE 120]
PF 80 MUDEPOS [0 0]
PD 150 UN MUDEPOS [10 0]
UL
PF 60 PD 90
PF 20 PD 90
PF 60 PD 180
UN
MUDEPOS [50 60]
UL
PD 90
PARA [I 1 4] [PF 20 PD 90]
UN
MUDEPOS [0 0]
PE 90
UL
FIM

```

16. Criar procedimento chamado **CATAVENTO2** com o formato da figura seguinte a partir do procedimento **RETANGULO3**.



```

APRENDA CATAVENTO2
REPITA 8 [
    RETANGULO3 PD 45
]
FIM

```

17. Descubra sem o uso do computador qual é a imagem:

```

APRENDA QUADRO :TAMANHO
REPITA 4 [
    PF :TAMANHO
    PD 90
]
PD 45
PF :TAMANHO * 7 / 5
PT :TAMANHO * 7 / 5
PE 45
PF :TAMANHO
PD 135
PF :TAMANHO * 7 / 5
PT :TAMANHO * 7 / 5
FIM

```

*Mostra um quadrado com divisões perpendiculares dando um efeito de quadradinhos.*

---

*criado a partir de quatro triângulos.*

---

---

CAPÍTULO 5

---

- Criar procedimento chamado **CAP0501** que efetue a leitura de um valor numérico inteiro e apresente o resultado do valor lido elevado ao quadrado sem efetuar o armazenamento do resultado em memória. A variável que receberá a entrada do dado deve ser definida como local.

```
APRENDA CAP0501
LOCAL "N
LEIA [Entre valor para o cálculo:] "N
MO (SN "Resultado "= POTÊNCIA INTEIRO :N 2)
FIM
```

- Criar procedimento chamado **CAP0502** que efetue a leitura de um valor numérico inteiro e apresente o resultado do valor lido elevado ao cubo com armazenamento do resultado calculado em memória. As variáveis devem ser definidas como local.

```
APRENDA CAP0502
LOCAL "N
LOCAL "R
LEIA [Entre valor para o cálculo:] "N
ATR "R POTÊNCIA INTEIRO :N 3
MO (SN "Resultado "= :R)
FIM
```

- Criar procedimento chamado **CAP0503** que efetue a leitura de uma temperatura em graus Celsius e apresente essa temperatura em graus Fahrenheit, sua conversão. A fórmula de conversão é " $F \leftarrow C * 9 / 5 + 32$ ", sendo "F" a temperatura em Fahrenheit e "C" a temperatura em Celsius. Armazene em memória o resultado calculado. Use variáveis locais. Formate a saída numérica com duas casas decimais.

```
APRENDA CAP0503
LOCAL "C
LOCAL "F
LEIA [Entre valor da temperatura em graus Celsius:] "C
ATR "F :C * 9 / 5 + 32
MO (SN [Temperatura em Fahrenheit] "= :F)
FIM
```

- Criar procedimento chamado **CAP0504** que efetue a leitura de uma temperatura em graus Fahrenheit apresente essa temperatura em graus Celsius, sua conversão. A fórmula de conversão é " $C \leftarrow ((F - 32) * 5) / 9$ ", sendo "F" a temperatura em Fahrenheit e "C" a temperatura em Celsius. Armazene em memória o resultado calculado. Use variáveis locais. Formate a saída numérica com duas casas decimais.

```
APRENDA CAP0504
LOCAL "F
LOCAL "C
LEIA [Entre valor da temperatura em graus Fahrenheit:] "F
ATR "C ((:F - 32) * 5) / 9
MO (SN [Temperatura em Celsius] "= :C)
FIM
```

5. Criar procedimento chamado **CAP0505** que efetue a leitura de dois valores numéricos inteiros (representados pelas variáveis locais "A" e "B") e mostre o resultado armazenado em memória do quadrado da diferença do primeiro valor (variável "A") em relação ao segundo valor (variável "B") junto a variável local "R".

```
APRENDA CAP0505
  LOCAL "A
  LOCAL "B
  LOCAL "R
  LEIA [Entre valor o valor <A>:] "A
  LEIA [Entre valor o valor <B>:] "B
  ATR "R POTÊNCIA (INTEIRO :A - INTEIRO :B) 2
  MO (SN [Quadrado da diferença] "= :R)
FIM
```

6. Criar procedimento chamado **CAP0506** que efetue a leitura de um número inteiro qualquer em uma variável local e multiplique este número por "2" armazenando o resultado em memória. Apresentar o resultado da multiplicação somente se o resultado for maior que "30".

```
APRENDA CAP0506
  LOCAL "N
  LOCAL "R
  LEIA [Entre valor numérico inteiro:] "N
  ATR "R (INTEIRO :N) * 2
  SE :R > 30 [
    MO (SN "Resultado" "= :R)
  ]
FIM
```

7. Criar procedimento chamado **CAP0507** que efetue a leitura de dois valores numéricos reais representados pelas variáveis locais "A" e "B" e apresente o resultado da diferença do maior valor pelo menor valor sem armazenar o resultado em memória.

```
APRENDA CAP0507
  LOCAL "A
  LOCAL "B
  LEIA [Entre valor numérico real <A>:] "A
  LEIA [Entre valor numérico real <B>:] "B
  SE :A > :B [
    MO (SN "Resultado" "= :A - :B)
  ][
    MO (SN "Resultado" "= :B - :A)
  ]
FIM
```

8. Criar procedimento chamado **CAP0508** que efetue a leitura de dois valores numéricos reais representados pelas variáveis locais "**A**" e "**B**" e apresente o resultado da diferença do maior valor pelo menor valor com armazenamento do cálculo em memória.

```
APRENDA CAP0508
LOCAL "A
LOCAL "B
LOCAL "R
LEIA [Entre valor numérico real <A>:] "A
LEIA [Entre valor numérico real <B>:] "B
SE :A > :B [
    ATR "R :A - :B
]
    ATR "R :B - :A
]
MO (SN "Resultado " = :R)
FIM
```

9. Criar procedimento chamado **CAP0509** que efetue a leitura de três valores numéricos inteiros desconhecidos representados pelas variáveis locais "**A**", "**B**" e "**C**". O procedimento deve somar esses valores, armazenar o resultado em memória e apresentar este resultado somente se for "**maior ou igual**" a "**100**".

```
APRENDA CAP0509
LOCAL "A
LOCAL "B
LOCAL "C
LOCAL "R
LEIA [Entre valor numérico real <A>:] "A
LEIA [Entre valor numérico real <B>:] "B
LEIA [Entre valor numérico real <C>:] "C
ATR "R INTEIRO :A + INTEIRO :B + INTEIRO :C
SE :R >= 100 [
    MO (SN "Resultado " = :R)
]
FIM
```

10. Criar procedimento chamado **CAP0510** que apresente a soma dos cem primeiros números naturais "(**1 + 2 + 3 + ... + 98 + 99 + 100**)" utilizando-se a primitiva "**REPITA**".

```
APRENDA CAP0510
LOCAL "S
ATR "S 0
REPITA 100 [
    ATR "S :S + CV
]
MO (SN "Somatório: " = :S)
FIM
```

11. Criar procedimento chamado **CAP0511** que apresente a soma dos cem primeiros números naturais "(1 + 2 + 3 + ... + 98 + 99 + 100)" utilizando-se a primitiva "**ENQUANTO**".

```
APRENDA CAP0511
  LOCAL "S
  ATR "S 0
  LOCAL "I
  ATR "I 1
  ENQUANTO [:I <= 100] [
    ATR "S :S + :I
    ATR "I :I + 1
  ]
  MO (SN "Somatório: "= :S)
FIM
```

12. Criar procedimento chamado **CAP0512** que apresente a soma dos cem primeiros números naturais "(1 + 2 + 3 + ... + 98 + 99 + 100)" utilizando-se a primitiva "**PARA**".

```
APRENDA CAP0512
  LOCAL "S
  ATR "S 0
  PARA [I 1 100] [
    ATR "S :S + :I
  ]
  MO (SN "Somatório: "= :S)
FIM
```

13. Criar procedimento chamado **CAP0513** que apresente o somatório dos *valores pares* existentes na faixa de "1" até "500". Use a primitiva "**REPITA**".

```
APRENDA CAP0513
  LOCAL "S
  ATR "S 0
  REPITA 500 [
    SE (RESTO CV 2) = 0 [
      ATR "S :S + CV
    ]
  ]
  MO (SN "Somatório: "= :S)
FIM
```

14. Criar procedimento chamado **CAP0514** que apresente o somatório dos *valores pares* existentes na faixa de "1" até "500". Use a primitiva "**ENQUANTO**".

```
APRENDA CAP0514
  LOCAL "S ATR "S 0
  LOCAL "I ATR "I 1
  ENQUANTO [:I <= 500] [
    SE (RESTO :I 2) = 0 [
      ATR "S :S + :I
    ]
    ATR "I :I + 1
  ]
  MO (SN "Somatório: "= :S)
FIM
```

15. Criar procedimento chamado **CAP0515** que apresente o somatório dos *valores pares* existentes na faixa de "1" até "500". Use a primitiva "**PARA**".

```
APRENDA CAP0515
LOCAL "S ATR "S 0
PARA [I 1 500] [
    SE (RESTO :I 2) = 0 [
        ATR "S :S + :I
    ]
]
MO (SN "Somatório: " = :S)
FIM
```

16. Criar procedimento chamado **CAP0516** que mostre todos os valores numéricos divisíveis por "4" *menores* que "20" iniciando a contagem em "1". Use a primitiva "**REPITA**".

```
APRENDA CAP0516
REPITA 19 [
    SE (RESTO CV 4) = 0 [
        MO CV
    ]
]
FIM
```

17. Criar procedimento chamado **CAP0517** que mostre todos os valores numéricos divisíveis por "4" *menores* que "20" iniciando a contagem em "1". Use a primitiva "**ENQUANTO**".

```
APRENDA CAP0517
LOCAL "I ATR "I 1
ENQUANTO [:I <= 19] [
    SE (RESTO :I 4) = 0 [
        MO :I
    ]
    ATR "I :I + 1
]
FIM
```

18. Criar procedimento chamado **CAP0518** que mostre todos os valores numéricos divisíveis por "4" *menores* que "20" iniciando a contagem em "1". Use a primitiva "**PARA**".

```
APRENDA CAP0518
PARA [I 1 19] [
    SE (RESTO :I 4) = 0 [
        MO :I
    ]
]
FIM
```

19. Criar procedimento chamado **CAP0519** que apresente os valores numéricos inteiros compreendidos na faixa de "0" e "4". Use a primitiva "**REPITA**".

Contagem de "0" a "4" com REPITA não é possível, pois a primitiva inicia a sua

---

ação sempre em "1", a menos que efetue o seguinte ajuste (não muito bom):

---

```
APRENDA CAP0519
  LOCAL "I
  ATR "I 0
  REPITA 5 [
    MO :I
    ATR "I :I + 1
  ]
FIM
```

Veja que a solução não é muito boa pois necessitou de uma variável auxiliar "I"

---

dando um estilo de "ENQUANTO". Então é melhor usar a primitiva "ENQUANTO"

---

e resolver o problema de forma mais adequada.

---

20. Criar procedimento chamado **CAP0520** que apresente os valores numéricos inteiros compreendidos na faixa de "0" e "4". Use a primitiva "**ENQUANTO**".

```
APRENDA CAP0520
  LOCAL "I
  ATR "I 0
  ENQUANTO [:I <= 4] [
    MO :I
    ATR "I :I + 1
  ]
FIM
```

21. Criar procedimento chamado **CAP0521** que apresente os valores numéricos inteiros compreendidos na faixa de "0" e "4". Use a primitiva "**PARA**".

```
APRENDA CAP0521
  PARA [I 0 4] [
    MO :I
  ]
FIM
```

22. Criar procedimento chamado **CAP0522** que apresente os valores numéricos inteiros compreendidos na faixa de "0" até "15" de "3" em "3". Use a primitiva "REPITA".

*Situação semelhante ao exercício 22. Só que aqui não vale nem a pena tentar fazer*

---

*a solução. O "REPITA" além de iniciar em "1" executa o salto de contagem sempre de*

---

*"1" em "1".*

---

23. Criar procedimento chamado **CAP0523** que apresente os valores numéricos inteiros compreendidos na faixa de "0" até "15" de "3" em "3". Use a primitiva "ENQUANTO".

```
APRENDA CAP0523
  LOCAL "I"
  ATR "I 0"
  ENQUANTO [:I <= 15] [
    MO :I
    ATR "I :I + 3
  ]
FIM
```

24. Criar procedimento chamado **CAP0524** que apresente os valores numéricos inteiros compreendidos na faixa de "0" até "15" de "3" em "3". Use a primitiva "PARA".

```
APRENDA CAP0524
  PARA [I 0 15 3] [
    MO :I
  ]
FIM
```

# ANOTAÇÕES

---

## Bibliografia

- ABELSON, H. **TI Logo**. New York: McGraw-Hill, 1984.
- \_\_\_\_\_. **TI Logo Education**. Lubbock: Texas Instruments & McGraw-Hill, 1981.
- ATARI. **ATARI Logo: Reference Manual**. Quebec: Logo Computer System, Inc., 1983.
- BASS, J. H. **Logo Programming**. Logo Spoken Here. 2002. Disponível em: <<http://pages.intnet.mu/jhbpage/main.htm>>. Acesso 28 jun. 2021.
- CATLIN, D. **My Personal Tribute to Seymour Papert**. GO Magazine. August, 2016. Disponível em: <<http://go.roamer-educational-robot.com/2016/08/12/my-personal-tribute-to-seymour-papert/>>. Acesso 26 jun. 2021.
- CORRALES MORA, M. **Lenguage Logo I: Descubriendo un mundo nuevo**. San José: EUNED, 1996.
- DOWNEY, A. B. & GAY, G. **How to think like a Computer Scientist: Logo version**. GNU Edition, 2003. Disponível em: <<http://openbookproject.net/thinkcs/archive/logo/english/thinklgo.pdf>>. Acesso 29 jun. 2021.
- ERFAN'S. **Welcome to Erfan's Zone of Logo**. Nestead. 2021. Disponível em: <<http://erfan96.50webs.com/>>. Acesso 28 jun. 2021.
- HARVEY, B. **Computer science Logo style: Symbolic computing**. 2nd. Massachusetts: MIT Press. 1998, v. 1.
- JOYS, D. **Hs-logo**. Logo Turtle Graphics Interpreter. 2021. Disponível em: <<https://deepakjois.github.io/hs-logo/>>. Acesso 28 jun. 2021.
- KHERIATY, I. & GERHOLD, G. **Radio Shack color Logo**. Massachusetts: Micropi, 1982.
- LOGO FOUNDATION. **What Is Logo?**. New York: Logo Foundation, 2021. Disponível em: <https://el.media.mit.edu/logo-foundation/>. Acesso em: 16 jun. 2021.
- MULLER, J. **The Great Logo Adventure: Discovering Logo on and Off the Computer**. Madison, AL, United States: Doone Pubns, 1998.
- PALEOTRONIC. **Past and Future Turtles - The Evolution of the Logo Programming Language: Part I**. Australia: Paleotronic Magazine, 2021. Disponível em: <https://paleotronic.com/2021/05/22/past-and-future-turtles-the-evolution-of-the-logo-programming-language-part-1/>. Acesso em: 16 jun. 2021.
- PETTI, W. A. **Math Cats**. Disponível em: <<http://www.mathcats.com/>>. Acesso em 23 jun. 2021.
- SOLOMON, C. J. (Et al). **History of Logo**. Proc. ACM Program. Lang. 4, HOPL, Article 79. June, 2020. Disponível em: <<https://dl.acm.org/doi/pdf/10.1145/3386329>>. Acesso em 28 jun. 2021.
- SPARER, E. **Sinclair Logo 1 Turtle Graphics**. Cambridge: Sinclair Research Ltd., 1984.
- TOMIYAMA, M. N. **Recursão**. Minas Gerais: Universidade Federal de Uberlândia - Faculdade de Computação, 2016. Disponível em: <<http://www.facom.ufu.br/~madriana/PF/recursao.pdf>>. Acesso em 22 jun. 2021.
- WINTER, M. J. **The Commodore 64 Logo workbook**. Chatsworth: DATAMOST, 1984.

## Referências bibliográficas

APPLE. **Apple Logo II: Reference manual.** California: Apple Computer, Inc. and Quebec: Logo Computer System. Inc. 1984.

ATARI. **ATARI Logo: Introduction programming through turtle graphics.** Quebec: Logo Computer System. Inc. 1983.

GOLDBERG, K. P. **Learning Commodore 64 Logo together.** Washington: Microsoft Press, 1984.

MANZANO, J. A. N. G. **Linguagem Logo: Programação de computadores - princípios de inteligência artificial.** São Paulo: All Print. 2012.

SASSENRATH, C. **Amiga LOGO: Tutorial and reference.** Commodore-Amiga, Inc. and Carl Sassenrath, 1989.

SOLOMON, C. J. **Apple Logo: Introduction programming through turtle graphics.** Quebec: Logo Computer System. Inc. 1982.

ANOTAÇÕES

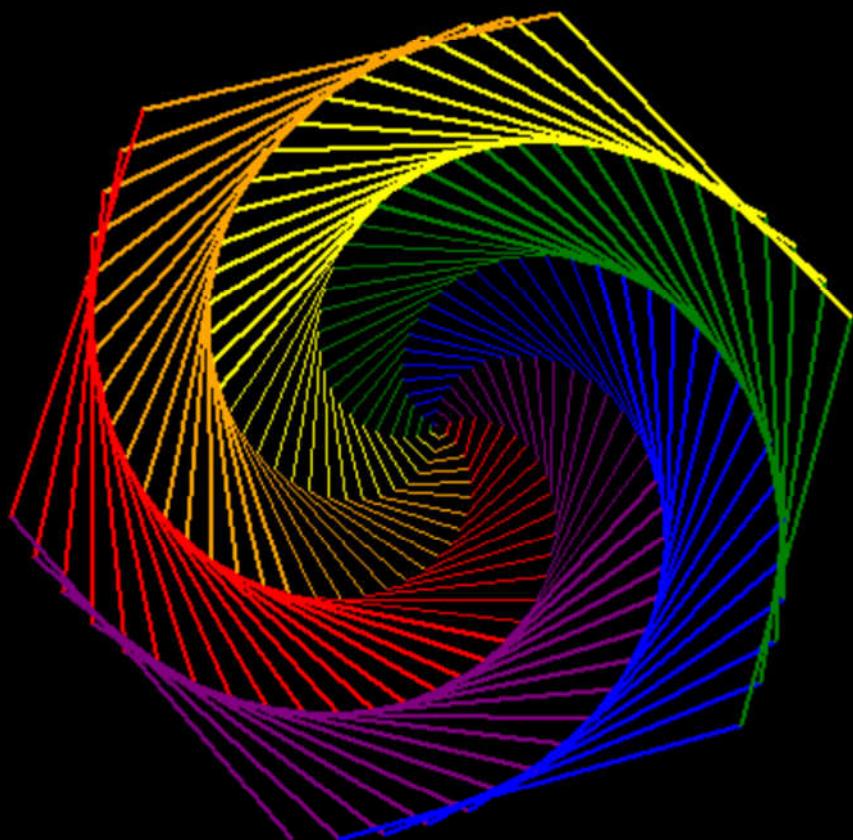






# LINGUAGEM LOGO

Introdução com xLogo



ESTE LIVRO MOSTRA A LINGUAGEM "LOGO" DE MANEIRA DINÂMICA E PRÁTICA A PARTIR DE EXEMPLOS DE APLICAÇÃO FOCADOS NOS PRINCÍPIOS DE LÓGICA DE PROGRAMAÇÃO.

A LINGUAGEM "LOGO" FOI CRIADA NA DÉCADA DE 1960 POR UMA EQUIPE MULTIDISCIPLINAR DE ESPECIALISTAS EM EDUCAÇÃO E INTELIGÊNCIA ARTIFICIAL DIRIGIDA PELO PROFESSOR SEYMOUR PAPERT.

NESTE TRABALHO A LINGUAGEM É APRESENTADA A PARTIR DO PONTO DE VISTA DAS AÇÕES DA GEOMETRIA DA TARTARUGA E DE AÇÕES BASEADAS NO DESENVOLVIMENTO TRADICIONAL DE PROGRAMAS COMO FAZEM OUTRAS LINGUAGENS DE PROGRAMAÇÃO.



ISBN 978-65-00-26491-3



9 786500 264913