



# LINGUAGEM LOGO

Introdução com FMSLogo

SISTEMA OPERACIONAL  
**Windows**



JOSÉ AUGUSTO N. G. MANZANO



Augusto N. G. Manzano



ORCID: 0000-0001-9248-7765



# Linguagem Logo

## Introdução com FMS Logo

São Paulo  
2021 - Propes Vivens



© 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, microfílmicos, fotográficos, reprográficos, 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 cibernético 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  
FMSLogo / José Augusto Navarro Garcia Manzano. -- 1.  
ed. -- São Paulo : Ed. do Autor, 2021.  
PDF  
  
ISBN 978-65-00-26071-7  
  
1. Ciência da computação 2. Computadores 3.  
Linguagem de programação (Computadores) 4.  
Processamento de dados I. Título.

21-71567

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 - 07/07/2021

1.01 - 09/07/2021

1.02 - 17/07/2021

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

Edição: Propes Vivens

# FERRAMENTA UTILIZADA

Produto: **FMSLogo [Versão 8.1.0]**

Sítio: <https://fmslogo.sourceforge.io/>

## REQUISITOS DE HARDWARE E SOFTWARE

- ◆ 512 MB de memória RAM;
- ◆ 20 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 *FMSLogo* 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\\_Logo-Intro-FMS-Logo](https://github.com/J-AugustoManzano/livro_Logo-Intro-FMS-Logo).

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 retorne 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 FMSLogo .....	12
1.3. O ambiente FMSLogo .....	15
2. AÇÕES BÁSICAS	
2.1. Primitivas iniciais .....	21
2.2. Outras interações .....	24
2.3. Primitivas complementares .....	27
3. AÇÕES ESPECIALIZADAS	
3.1. Repetições .....	31
3.2. Procedimentos .....	33
3.3. Sub-rotinas .....	37
3.4. Variável .....	40
3.5. Decisão .....	43
3.6. Recursão .....	49
3.7. Memorização e amnésia .....	51
3.8. Exercícios de fixação .....	53
4. AÇÕES ESPECÍFICAS	
4.1. Entrada de dados .....	55
4.2. Randomização .....	58
4.3. Coordenadas .....	60
4.4. Cores .....	64
4.5. Fractais .....	67
4.6. Outras repetições .....	69
4.7. Exercícios de fixação .....	73
5. AÇÕES COMPLEMENTARES	
5.1. Funções matemáticas .....	77
5.2. Funções lógicas .....	79
5.3. Algumas funções complementares .....	80
5.4. Alguns eventos de controle .....	84
5.5. Programação sem figuras .....	84
5.6. Manipulação básica de dados .....	88
5.7. Escopo e visibilidade de variáveis .....	92
5.8. Exercícios de fixação .....	95
APÊNDICES	
A. Algumas cores .....	99
B. Primitivas inglês/espanhol (algumas) .....	103
C. Exemplos geométricos .....	107
D. Efeitos sonoros .....	121
E. Espiral hexagonal (imagem da capa) .....	127
F. Gabarito .....	131
REFERÊNCIAS BIBLIOGRÁFICAS .....	147



---

## 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, tais 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 FMSLogo

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

Para obter o ambiente "**FMSLogo**" acesse o endereço "<https://fmslogo.sourceforge.io/>". A partir do menu lateral direito selecione a opção "**Download FMSLogo**" como mostra a figura 1.1.

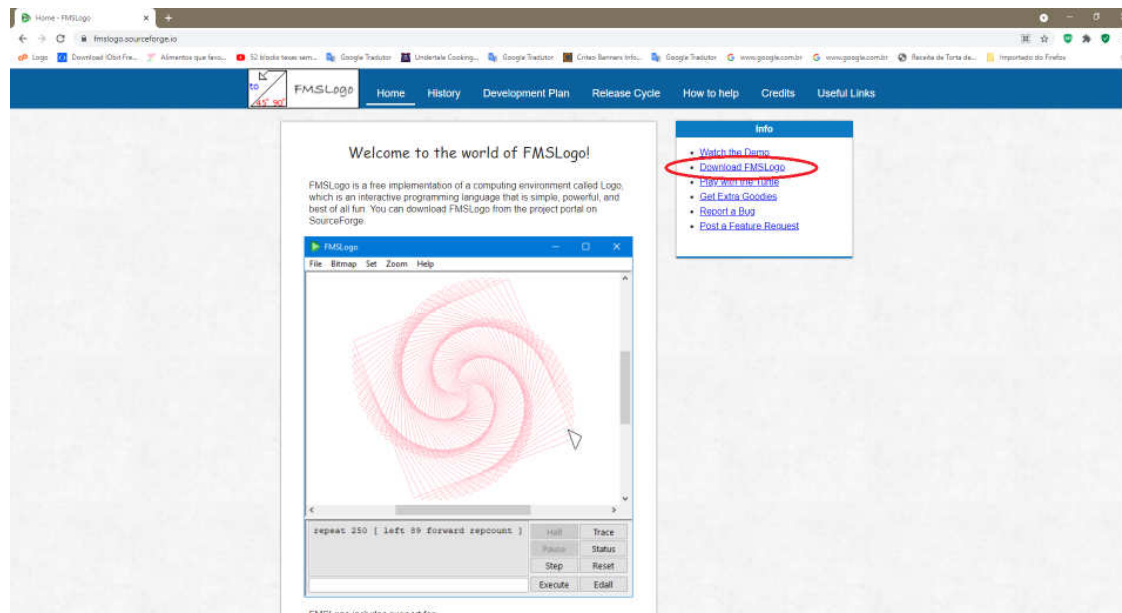


Figura 1.1 - Sítio oficial do projeto "FMSLogo".

Após selecionar a opção "**Download FMSLogo**" é apresentada a página da plataforma de repositório de projetos de código aberto "**SOURCEFORGE**" como indica a figura 1.2.

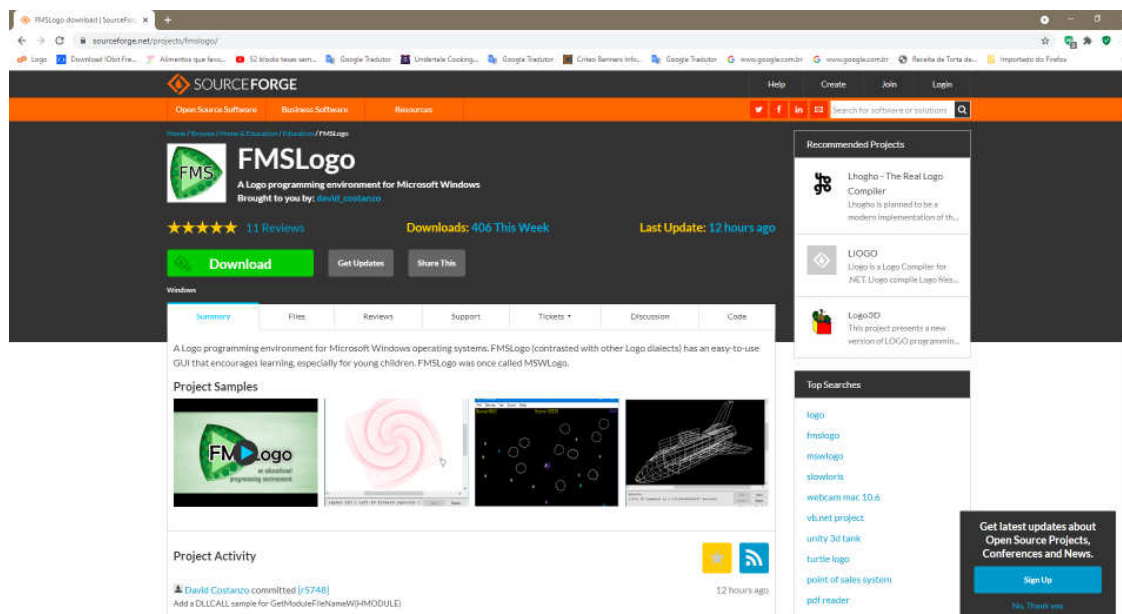


Figura 1.2 - Sítio da plataforma "SOURCEFORGE".

Na plataforma "**SOURCEFORGE**" acione o botão verde "**Download**" e aguarde o arquivo com o ambiente de programação da linguagem Logo ser copiado para seu computador.

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) o arquivo copiado "**fmslogo-8.1.0.exe**". Veja a figura 1.3.

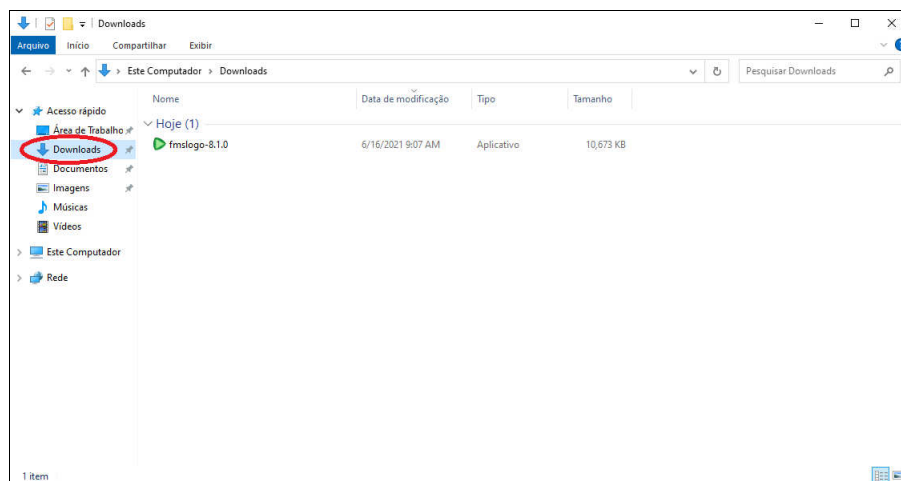


Figura 1.3 - Programa "Explorador de arquivos".

Assim que o processo de instalação é iniciado é solicitada a escolha do idioma usado no ambiente Logo. Esta etapa é muito importante, pois a escolha determina o idioma usado, o qual não poderá ser posteriormente alterado. Veja na figura 1.4 a caixa de seleção "**Installer Language**" com a lista de idiomas suportados.

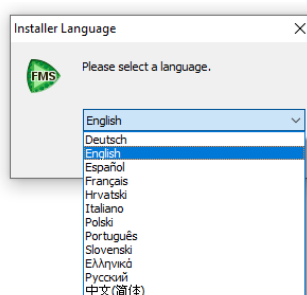


Figura 1.4 - Lista de idiomas suportados.

É ideal selecionar o idioma "**Português**", mas se desejar escolha outro de sua preferência. No entanto, os códigos e explanações deste livro levam em consideração o uso do idioma português. Após selecionar o idioma acione o botão "**OK**" da caixa de seleção "**Installer Language**".

Após selecionar o idioma é apresentado na caixa de diálogo "**Instalação de FMSLogo**", como mostra a figura 1.5, o "**Contrato de Licença**". Nesta fase é importante lê-lo e se concordar com seus termos acione o botão "**Aceito**" para que o processo de instalação seja iniciado.

Assim que o botão "**Aceito**" é acionado ocorre a apresentação na caixa de diálogo "**Instalação de FMSLogo**", como mostra a figura 1.6, a "**Escolha de Componentes**". Neste instante acione apenas o botão "**Seguinte**".

Na sequência é apresentada a caixa de diálogo "**Instalação de FMSLogo**", como mostra a figura 1.7, com a indicação "**Escolha do Local de Instalação**". Neste instante acione apenas o botão "**Seguinte**".

A próxima etapa do processo de instalação mostra a caixa de diálogo "**Instalação de FMSLogo**", como mostra a figura 1.8, com a indicação "**Escolha uma Pasta do Menu Iniciar**". Neste instante acione apenas o botão "**Seguinte**".

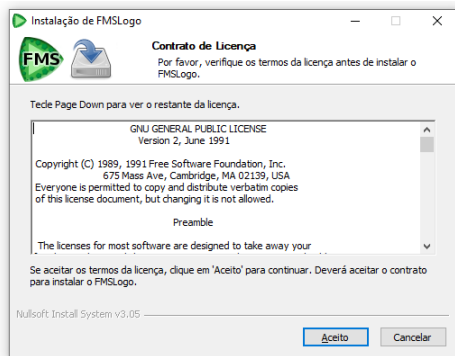


Figura 1.5 - Instalação "Contrato de Licença".

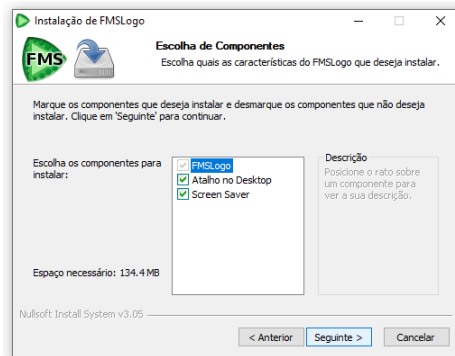


Figura 1.6 - Instalação "Escolha de Componentes".

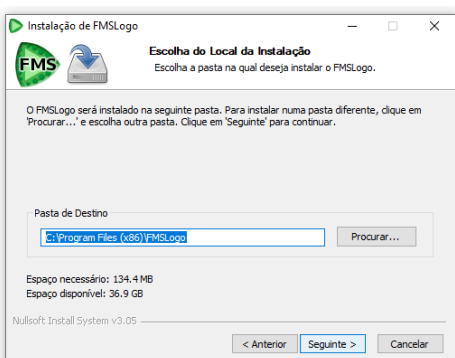


Figura 1.7 - Instalação "Local de Instalação".

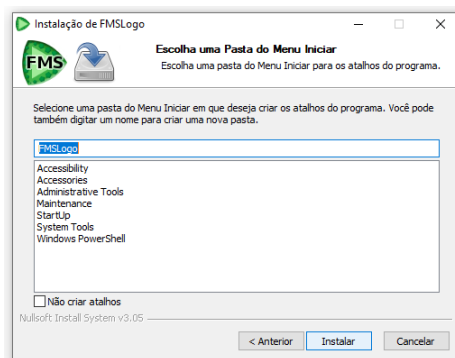


Figura 1.8 - Instalação "Pasta do Menu Iniciar".

Após a seleção do local de identificação no "**Menu Iniciar**" o processo de instalação é iniciado. Aguarde pacientemente a conclusão desta etapa que dependendo da configuração do computador poderá ser demorada (figura 1.9) e quando concluída acione o botão "**Terminar**" na caixa de conclusão da instalação indicada na figura 1.10.

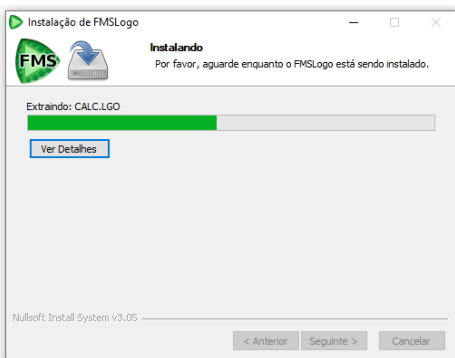


Figura 1.9 - Andamento da instalação.

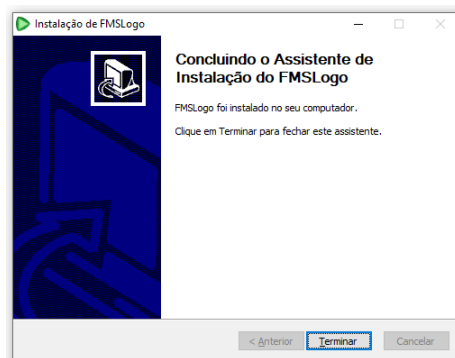


Figura 1.10 - Encerramento da instalação.

Concluída a etapa de instalação a linguagem Logo do ambiente "**FMSLogo**" já pode ser executada, havendo duas formas de fazê-lo: uma a partir da execução do ícone gerado na área de trabalho do sistema e outra a partir do "**Menu Iniciar**".

Para acessar por meio do "**Menu Iniciar**" acione o ícone "**Windows**" ao lado esquerdo da barra de ferramentas e a partir da letra "**F**" selecione a pasta "**FMSLogo**" e dentro desta pasta selecione o ícone "**FMSLogo**" como indicado pelas marcações na figura 1.11. Será apresentada a tela do ambiente de desenvolvimento como mostra a figura 1.12.



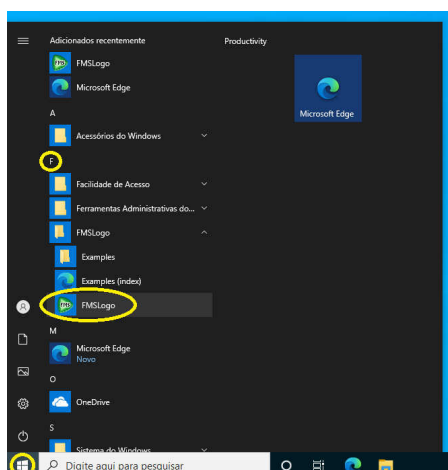


Figura 1.11 - Seleção do ambiente "FMSLogo".

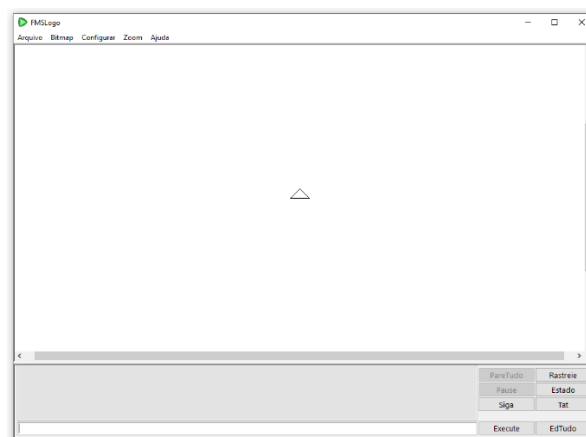


Figura 1.12 - Ambiente "FMSLogo".

A partir dessas instruções já é possível fazer uso do ambiente de programação **FMSLogo**.

### 1.3 - O ambiente FMSLogo

O "FMSLogo" é familiar aos programas "SuperLogo" e "BetaLogo" e de certa forma parecido com estes. A diferença principal está em algumas opções do menu, inexistentes no "FMSLogo" e na apresentação da interface gráfica. Nos programas "SuperLogo" e "BetaLogo" a área de trabalho é dividida em duas janelas independentes (mas integradas), sendo uma *Janela de Comandos* na parte inferior e a *Janela de Menu* na parte superior.

No programa "FMSLogo" ambas as janelas são dispostas em uma mesma tela a partir de uma: barra de título; barra de menu; área de ação para o modo gráfico; área de ação para o modo texto; barra de ferramentas lateral e um campo de entrada de comandos, dados e instruções que estabelecem a área de comandos do programa. Veja estes detalhes indicados na figura 1.13. 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.

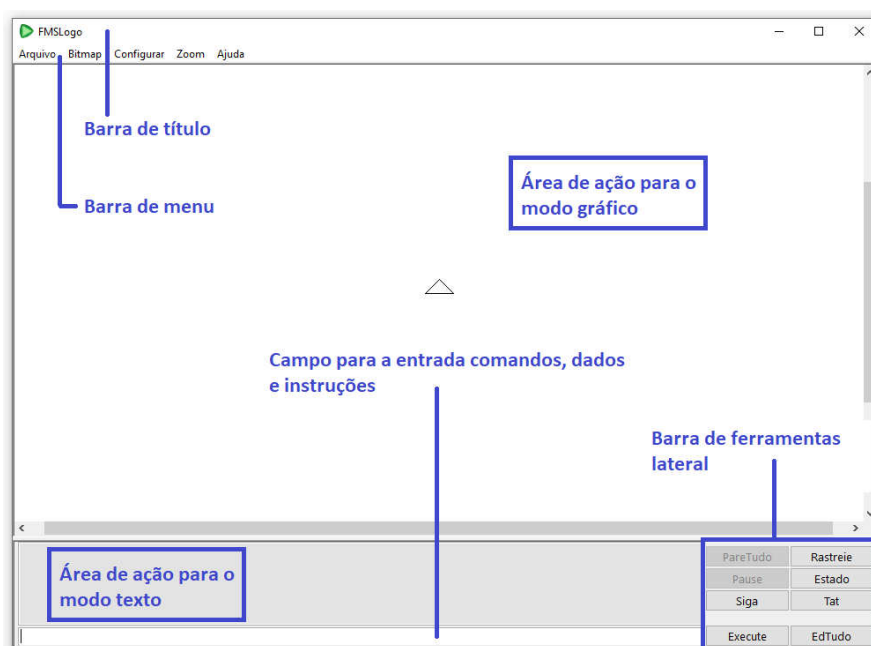


Figura 1.13 - Estrutura de organização do ambiente "FMSLogo".

A barra de menu é formada pelos comandos: **Arquivo** (*Novo, Carregar, Abrir, Guardar, Guardar como, Coloque como Protetor de Tela, Editar, Apagar e Sair*), **Bitmap** (*Novo, Carregar, Abrir, Guardar, Guardar como, imprimir, Configurar página e Área Ativa*), **Configurar** (*Tamanho do lápis, Tipo de letra para rótulo, Tipo de letras para comandos, Cor de lápis, Cor de preenchimento e Color de fundo*), **Zoom** (*Ampliar, Reduzir e Normal*) e **Ajuda** (*Início, Tutorial, Demo, Exemplos, Notas da versão, Sobre o FMSLogo e Sobre o FMS*). 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 "**Carregar...**" do menu "**Arquivo**"

Este comando efetua o carregamento de um arquivo de script Logo do disco para a memória anexando o arquivo carregado aos procedimentos existentes.

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**" do menu "**Arquivo**"

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

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", "Carregar..." ou "Abrir...".

Comando "**Editar...**" do menu "**Arquivo**"

Este comando abre a caixa de diálogo *Editar* que possibilita selecionar procedimentos definidos na memória para edição. O botão *OK* deve ser usado quando apenas um procedimento é editado e previamente selecionado e o botão *Tudo* pode ser usado quando há o desejo de editar mais de um procedimento existente na memória.

Comando "**Apagar...**" do menu "**Arquivo**"

Este comando abre a caixa de diálogo *Apagar* que possibilita selecionar procedimentos definidos na memória para remoção. O botão *OK* deve ser usado quando apenas um procedimento é apagado e previamente selecionado e o botão *Tudo* pode ser usado quando há o desejo de remover da memória todos os procedimentos existentes.

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

Este comando finaliza a execução do ambiente.

Comando "**Novo**" do menu "**Bitmap**"

Este comando efetua a limpeza da área de ação para o modo gráfico.

**Comando "Carregar..." do menu "Bitmap"**

Este comando permite carregar para a área de ação do modo gráfico de uma imagem do disco que esteja no padrão Bitmap do Windows (extensão .BMP ou .GIF).

**Comando "Guardar" do menu "Bitmap"**

Este comando permite fazer a guarda das imagens criadas no formato bitmap junto ao disco.

**Comando "Guardar como..." do menu "Bitmap"**

Este comando permite gravar com outro nome uma imagem carregada ou imagem previamente gravada.

**Comando "Configurar página..." do menu "Bitmap"**

Este comando permite configurar a formatação da página para impressão com a possibilidade de ajustes nas margens, no tamanho da folha e na orientação da impressão.

**Comando "Imprimir..." do menu "Bitmap"**

Este comando efetua a impressão da área de ação do modo gráfico.

**Comando "Área ativa..." do menu "Bitmap"**

Este comando seleciona a área de imagem a ser utilizada dentro pelo Logo.

**Comando "Tamanho do lápis..." do menu "Configurar"**

Este comando permite selecionar a espessura do lápis que a tartaruga usará ao se movimentar no ambiente.

**Comando "Tipo de letra para rótulo..." do menu "Configurar"**

Este comando permite selecionar o tipo de fonte de texto a ser usada, seu tamanho e estilo na área de ação para o modo gráfico.

**Comando "Tipo de letra para comandos..." do menu "Configurar"**

Este comando permite selecionar o tipo de fonte de texto a ser usada, seu tamanho e estilo na área de ação para o modo texto.

**Comando "Cor de lápis..." do menu "Configurar"**

Este comando permite selecionar a cor usada para se fazer os traços dos desenhos.

**Comando "Cor de preenchimento..." do menu "Configurar"**

Este comando permite selecionar a cor usada para realizar o preenchimento de figuras desenhadas.

**Comando "Color de fundo..." do menu "Configurar"**

Este comando permite selecionar a cor usada no estabelecimento do fundo da área de ação para o modo gráfico.

**Comando "Ampliar" do menu "Zoom"**

Este comando permite aumentar o zoom da área de ação para o modo gráfico.

Comando "**Reduzir**" do menu "**Zoom**"

Este comando permite diminuir o zoom da área de ação para o modo gráfico.

Comando "**Normal**" do menu "**Zoom**"

Este comando permite voltar o tamanho da área de ação para o modo gráfico ao tamanho padrão.

Comando "**Início**" do menu "**Ajuda**"

Este comando abre o modo de ajuda do ambiente a partir de seu início.

Comando "**Tutorial**" do menu "**Ajuda**"

Este comando abre o modo tutorial de aprendizagem onde seu usuário poderá avançar em outros estudos sobre o uso da linguagem Logo.

Comando "**Demo**" do menu "**Ajuda**"

Este comando coloca o ambiente em modo de demonstração apresentando efeitos que podem ser conseguidos.

Comando "**Exemplos**" do menu "**Ajuda**"

Este comando apresenta uma lista de exemplos do que se pode fazer com a linguagem Logo.

Comando "**Notas da versão**" do menu "**Ajuda**"

Este comando abre um arquivo texto no bloco de notas com informações sobre essa versão do FMSLogo em uso.

Comando "**Sobre o FMSLogo...**" do menu "**Ajuda**"

Este comando apresenta informações gerais sobre o programa FMSLogo.

Comando "**Sobre FMS...**" do menu "**Ajuda**"

Este comando apresenta o espaço para obtenção de maiores informações sobre Esclerose Múltipla (*Multiple Sclerosis*, em inglês). Pesquise sobre *Associação Brasileira de Esclerose Múltipla* e *Associação Nacional de Esclerose Múltipla*.

A barra de ferramenta inferior na lateral direita é formada pelos botões: **ParaTudo**, **Rastreie**, **Pause**, **Estado**, **Siga**, **Tat**, **Execute** e **EdTudo** e está ao lado da área de ação para o modo texto. Estes dois elementos formam o centro de controle onde se comanda o "FMSLogo". Nesta área a parte mais importante é o campo para a entrada de comandos, dados e instruções, além dos botões laterais. Observe o que faz cada um dos botões:

Botão "**PareTudo**"

Este botão permite interromper os procedimentos que estejam em execução e abre o modo de edição com acesso a todos os procedimentos para que sejam editados.

**Botão "Rastreie/Não Rastreia"**

Este botão permite colocar o modo de execução de procedimentos em modo de rastreamento para auxiliar ações de depuração. Este botão funciona independentemente do que estiver sendo acompanhado pela ação do botão **"Rastreie"**.

**Botão "Pause"**

Este botão interrompe a execução do procedimento em memória a partir da apresentação de uma caixa de diálogo que permite realizar alterações e acompanhamentos de diversas ocorrências. A caixa apresentada aceita a entrada de dados no campo *Entrada* e permite manter a ação de pausa com o uso do botão **"OK"** ou cancelar a operação por meio do botão **"Cancelar"**.

**Botão "Estado/SemEstado"**

Este botão permite abrir uma janela com a apresentação dos parâmetros em uso dentro do ambiente FMSLogo. Para fechar a janela de parâmetros acione o botão **"SemEstado"**.

**Botão "Siga/NãoSiga"**

Este botão coloca o ambiente em modo de depuração permitindo acompanhar a execução das linhas de código dos procedimentos. Para fechar o modo de depuração acione o botão **"NãoSiga"**.

**Botão "Tat"**

Este botão efetua a limpeza da área de ação do modo gráfico retornando o prompt da tartaruga para o centro da tela, coordenadas 0,0 com direção apontada para o norte, ou seja, valor 0.

**Botão "Execute"**

Este botão permite executar o que estiver escrito no campo de entrada de comandos, dados e instruções.

**Botão "EdTudo"**

Este botão permite abrir o modo de edição de texto para acesso a todos os procedimentos definidos em memória, dando a possibilidade de editá-los.

A barra de títulos do ambiente além do nome como título **"FMSLogo"** do lado esquerdo, possui do lado direito 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** ou **TAT**.

Para usar o ambiente localize seu ícone de acesso e selecione-o e para sair use **"Arquivo/Sair"** ou acione as teclas de atalho **"<Alt> + <F4>"** ou selecione o botão **"X"** na barra de título.

## ANOTAÇÕES

[illegible]

## 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 escreve-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 de "0" até "500" *pixels*.

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.

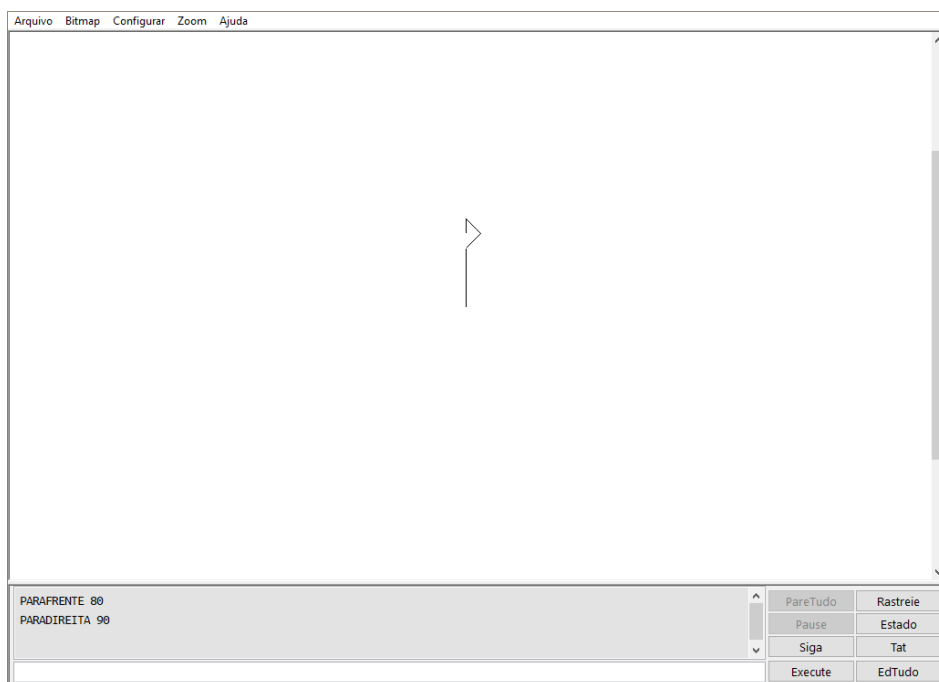


Figura 2.2 - Resultado da tartaruga no sentido **LESTE** após instrução "PARADIREITA 90".

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 ponteiro do *mouse* sobre a linha que divide as áreas de ação do modo gráfico e do modo texto.



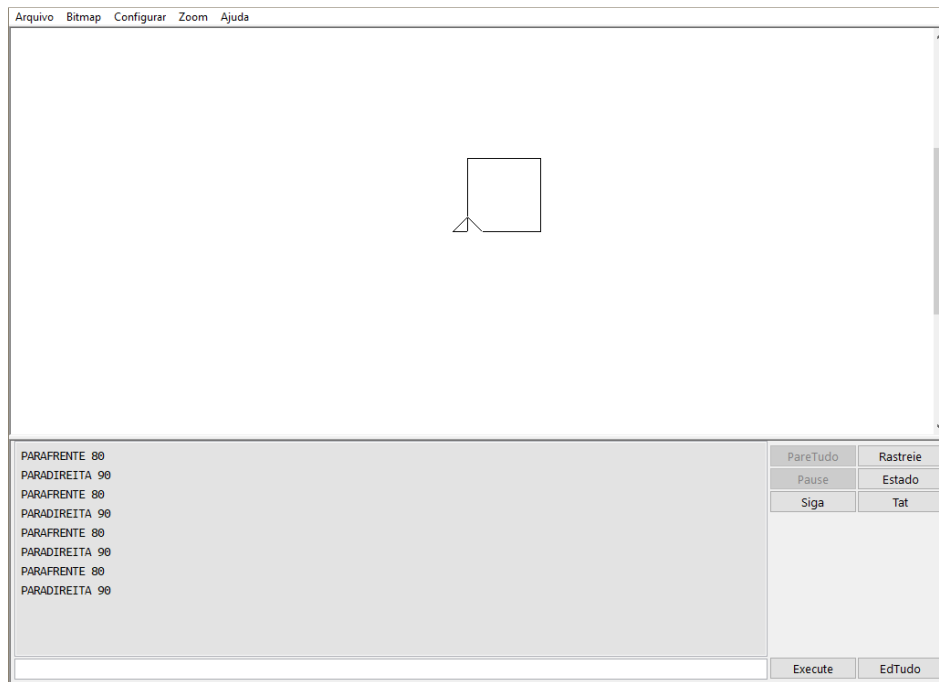


Figura 2.3 - Apresentação de um quadrado.

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:

#### OCULTETAT

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

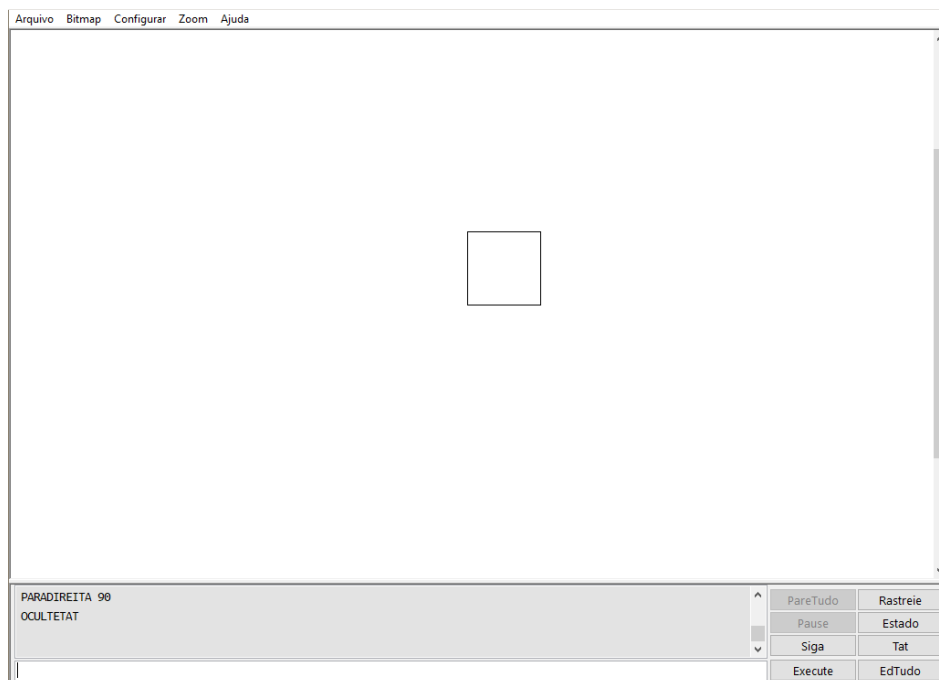


Figura 2.4 - Apresentação de um quadrado sem sobreposição da tartaruga.

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:

### TARTARUGA

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:

### LIMPEJANELACOMANDOS

Veja que até este ponto você já aprendeu que uma instrução pode ser definida a partir do uso de um comando (**OCULTETAT**) 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 da primitiva **ESCREVA** veja algumas operações aritméticas simples interessantes a partir das seguintes instruções:

ESCREVA 5 + 2

ESCREVA 5 - 2

ESCREVA 5 \* 2

ESCREVA 5 / 2

Veja que as instruções anteriores são formadas pelo uso de um comando com um parâmetro identificado pela definição de um cálculo aritmético. A figura 2.5 mostra os resultados das operações aritméticas estabelecidas dentro na área de ação do modo texto.



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

ESCREVA [Linguagem Logo]

Veja o resultado das duas apresentações na figura 2.6.



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 SOMA 5 2
ESCREVA DIFERENÇA 5 2
ESCREVA PRODUTO 5 2
ESCREVA QUOCIENTE 5 2
ESCREVA RESTO 5 2
ESCREVA INTEIRO QUOCIENTE 5 2
ESCREVA ABS -5
ESCREVA POTÊNCIA 5 2
ESCREVA RAIZQ 25
```

Veja os resultados do uso de funções na figura 2.7.



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 SOMA 1 2 7
```

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 "**Você não disse o que devo fazer com 7**" 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 (SOMA 1 2 7)
```

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:

**PARADIREITA 90**

**ROTULE [Linguagem Logo]**

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

Até este ponto todas 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.

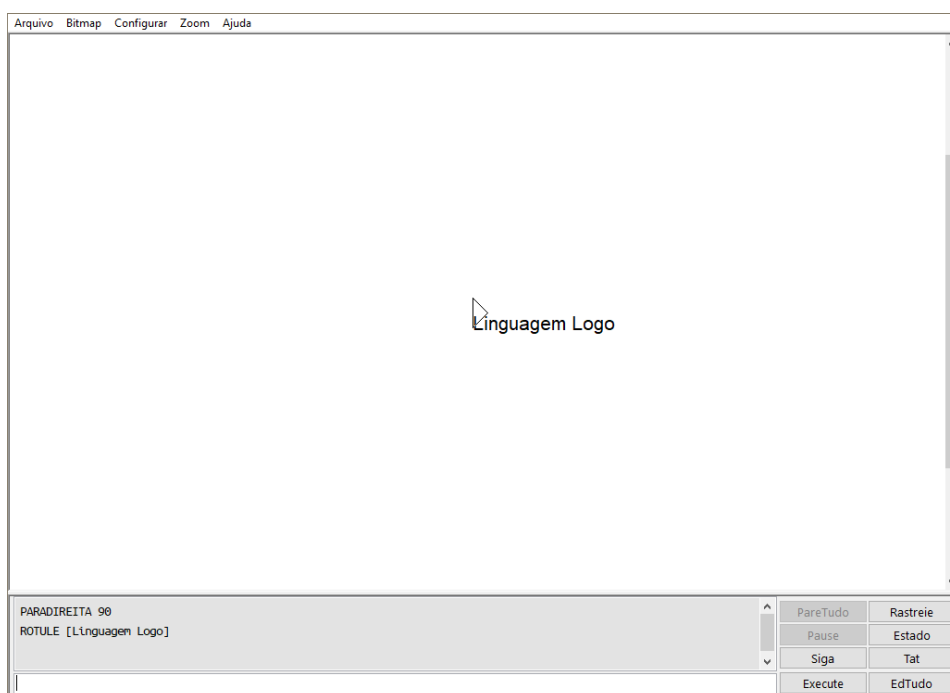


Figura 2.8 - Apresentação de mensagem na área gráfica.

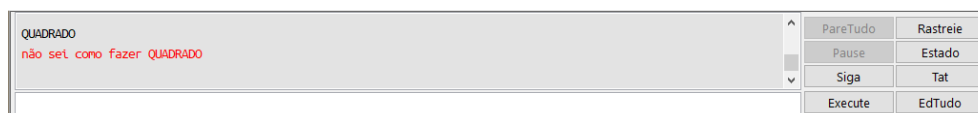


Figura 2.9 - Apresentação de mensagem de erro.

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 a 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:

**LIMPEJANELACOMANDOS**

**TARTARUGA**

**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.

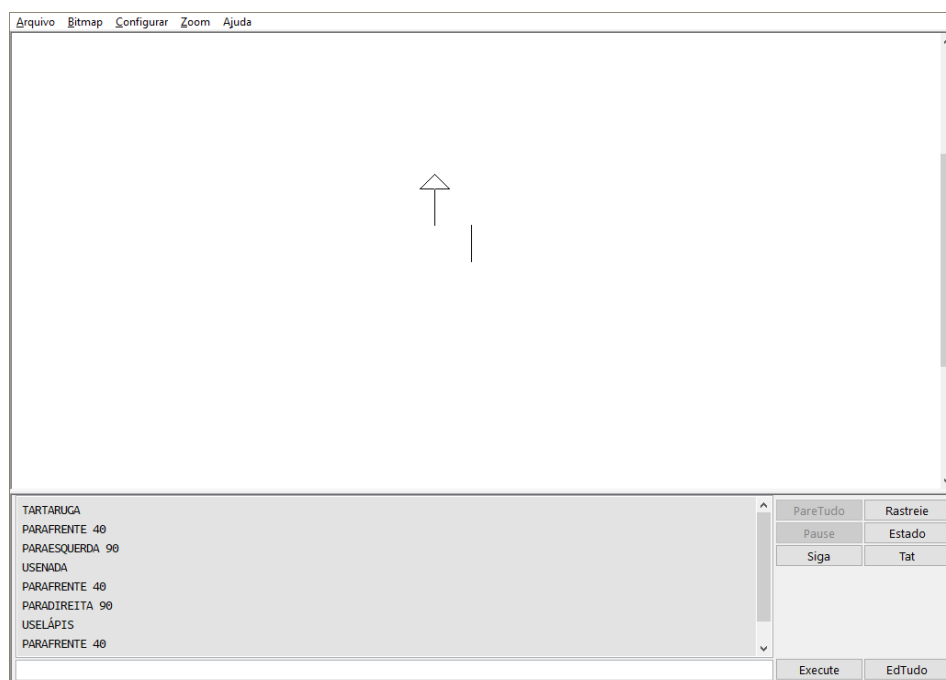


Figura 2.10 - Apresentação de linhas verticais deslocadas.

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.

Anteriormente foi comentado que o tamanho de *pixels* usado com os comandos **PARAFRENTE** e **PARATRÁS** deve ser entre "1" e "500". Mas, o que aconteceria se fosse definido um valor acima de 500? Assim sendo, após executar os comandos **TARTARUGA** e **LIMPEJANELACOMANDOS** execute a instrução seguinte:

### PARAFRENTE 750

Se você esperava um erro, ficou sem vê-lo, pois a instrução foi processada e executada, como mostra a figura 2.11.

O que aconteceu então? O universo Logo do ambiente **FMSLogo** é, na verdade, uma esfera. Isto posto, se executada uma instrução **PARAFRENTE 1050** ocorrerá a sobreposição do movimento em si mesmo.

Há um provérbio chinês que diz que "*se você não mudar a direção, terminará exatamente onde partiu*". Veja que isso ocorre exatamente em Logo.

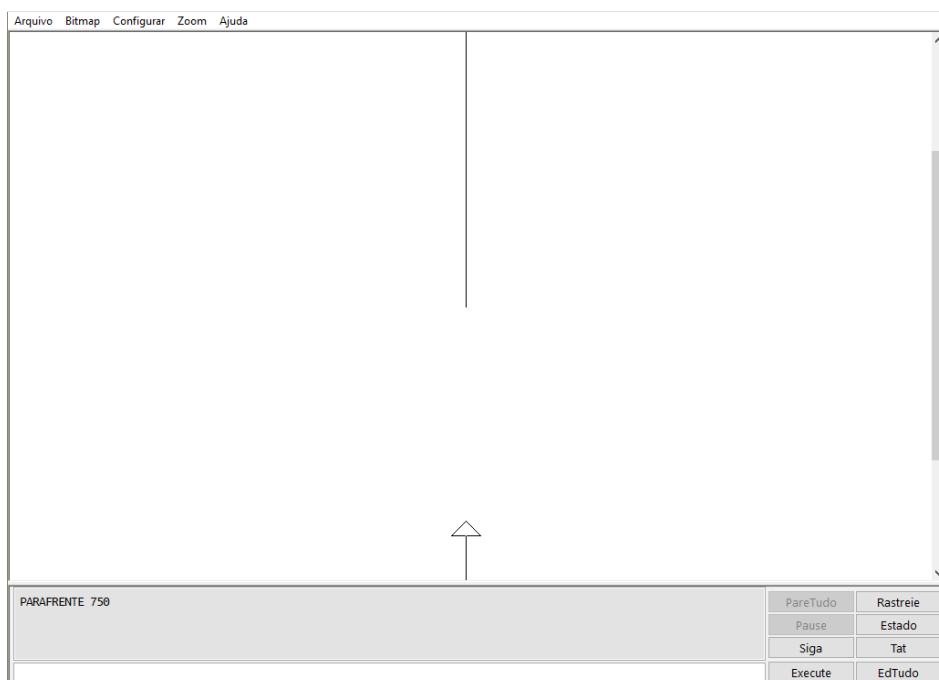


Figura 2.11 - Apresentação de traço além do limite entre 1 e 500.

O plano de ação que permite desenhar na área de ação do modo gráfico pode ser esquematizado segundo a estrutura indicada na figura 2.12 que demonstra de forma simplificada a dimensão padrão da tela de trabalho para deslocamento e graus de giro para locomoção da tartaruga.

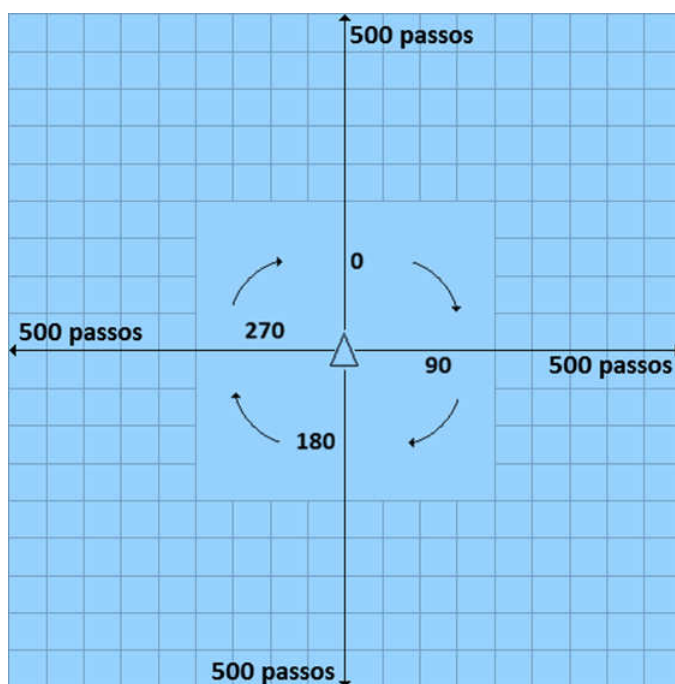


Figura 2.12 - Dimensão padrão do universo Logo no FMSLogo.

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
OCULTETAT	OT
MOSTRETAT	MT
TARTARUGA	TAT
LIMPEJANELACOMANDOS	LJC
ESCREVA	ESC
USENADA	UN
USELÁPIS	UL
USEBORRACHA	UB

Tabela 2.1 - Primitivas simplificadas.

## ANOTAÇÕES

[illegible]



## 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**.

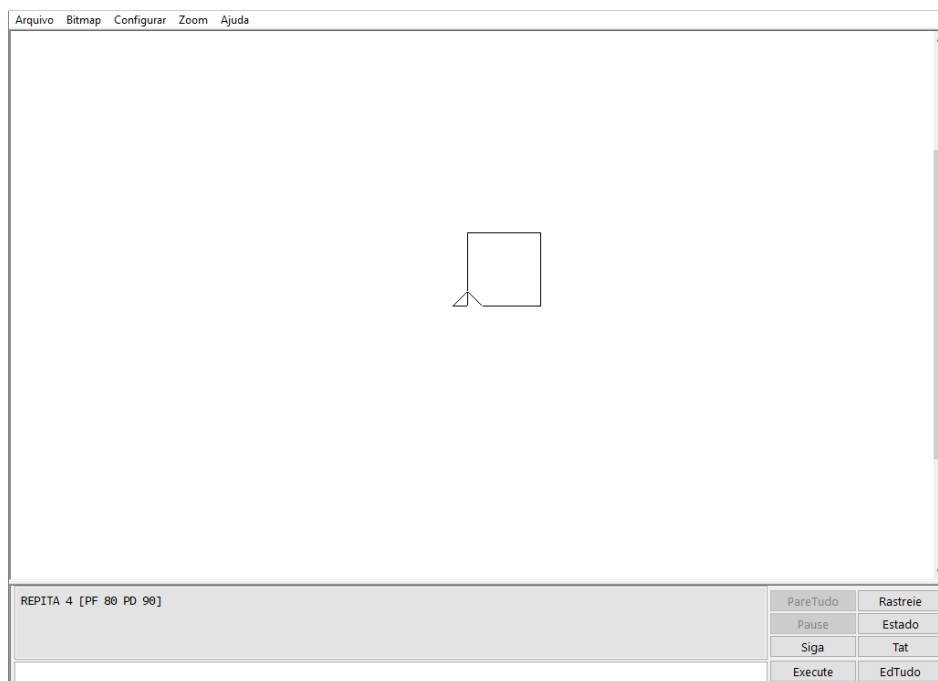


Figura 3.1 - Quadrado apresentado a partir do uso do comando "REPITA".

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

TAT

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**.

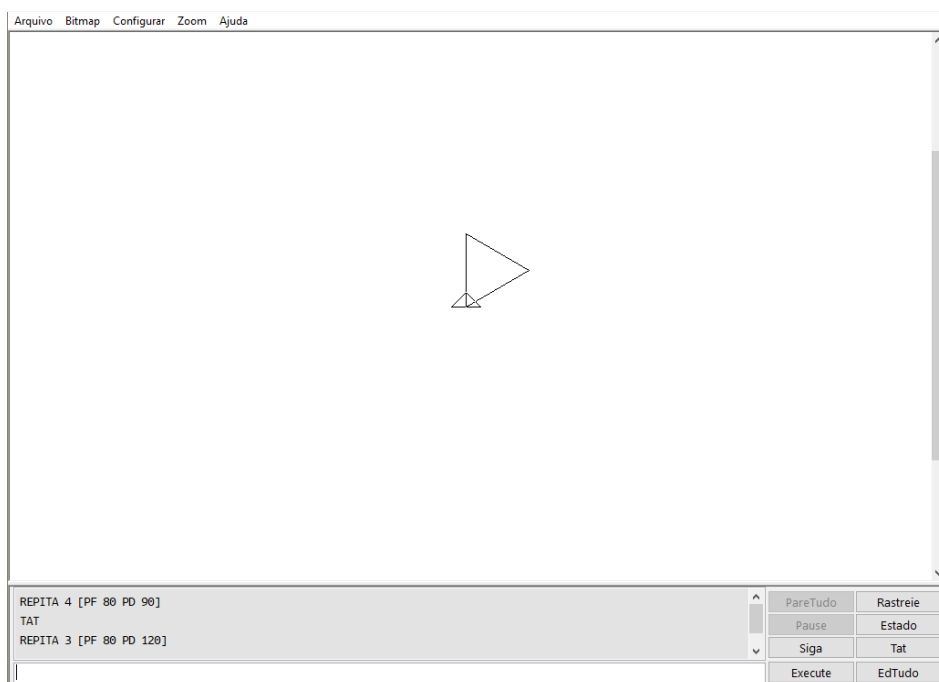


Figura 3.2 - Triângulo equilátero apresentado a partir do uso do comando "REPITA".

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:

TAT

REPITA 5 [PF 80 PD 72]

A figura 3.3 mostra o resultado obtido.

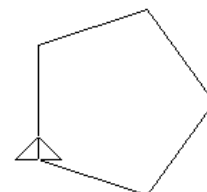


Figura 3.3 - Pentágono.

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:

TAT

REPITA 360 [PF 1 PD 1]

A figura 3.3 mostra o resultado obtido.

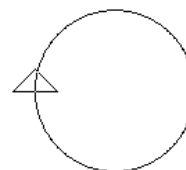


Figura 3.4 - Circunferência.

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.

<b>POLÍGONO</b>	<b>LADOS</b>	<b>360 / LADOS</b>	<b>ÂNGULO</b>
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
TRIDECÁGONO	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
ENEDECÁGONO	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**.

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 uma caixa de diálogo com o nome do procedimento em edição como mostra a figura 3.5, onde o código de operação desejado deve ser informado. Neste momento, escreva o que deve ser executado. Por exemplo, coloque a instrução **REPITA 4 [PF 80 PD 90]** como mostrado na figura 3.6. Para finalizar a edição acione o botão **Fim**. Se acionado o botão **Cancelar** a edição do procedimento não será registrada na memória.

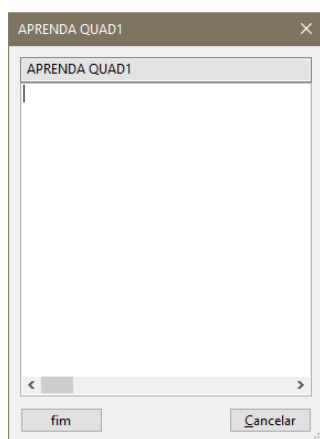


Figura 3.5 - Caixa de diálogo para definição de procedimento.

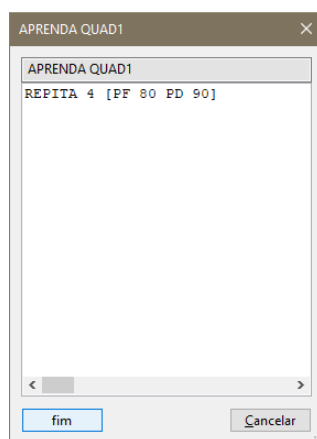


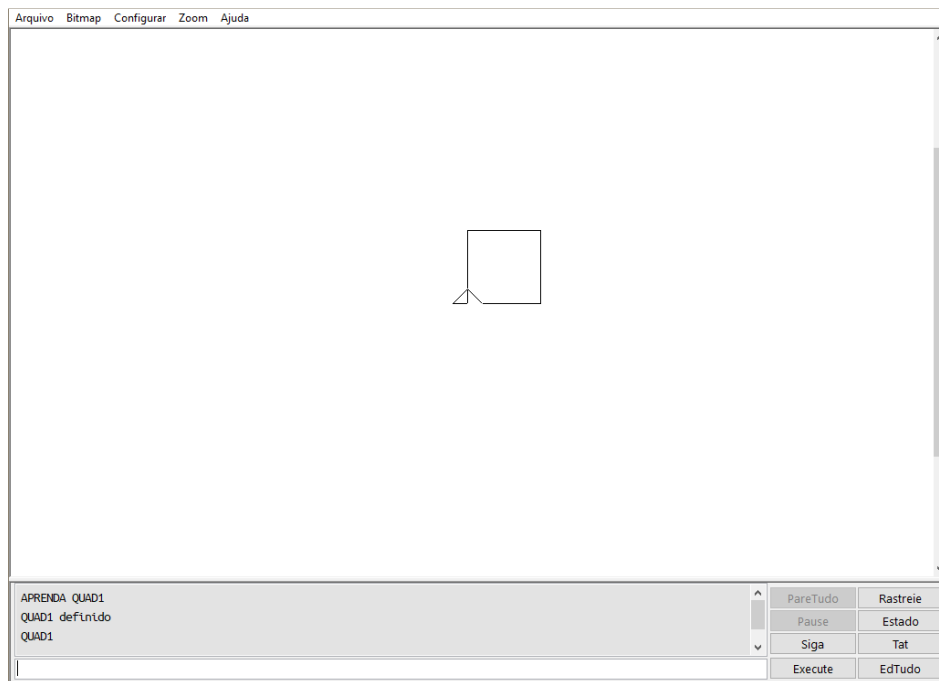
Figura 3.6 - Caixa de diálogo com procedimento definido.

Assim que o procedimento é devidamente registrado em memória ocorre a apresentação na área de ação do modo texto da mensagem "**QUAD1 definido**".

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.

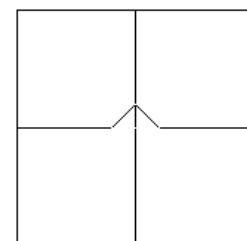


**Figura 3.7 - Quadrado desenhado a partir da definição de um conhecimento artificial.**

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:

**TAT**  
**LJC**  
**REPITA 4 [QUAD1 PE 90]**

A figura 3.8 mostra o resultado desta operação.



**Figura 3.8 - Quadrado repetido.**

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:

**TAT**  
**LJC**  
**REPITA 8 [QUAD1 PE 45]**

A figura 3.9 mostra o resultado desta operação.

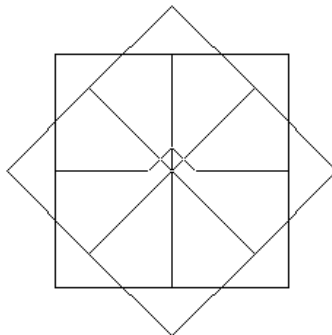


Figura 3.9 - Quadrados intercalados.

A forma apresentada de definição de procedimento é uma das maneiras possíveis de uso deste recurso. Outro jeito é a partir do uso da opção "**Editar...**" menu "**Arquivo**".

Após acionar "**Arquivo/Editar...**" ocorre a apresentação da caixa de diálogo "**Editar**" como apresentado na figura 3.10.

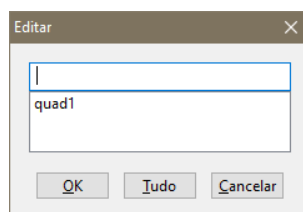


Figura 3.10 - Caixa de diálogo "Editar".

Na caixa apresentada é possível escrever o nome de um novo procedimento ou selecionar um procedimento existente na lista para edição. O botão "**OK**" entra em modo de edição para o registro de um novo procedimento ou para editar um procedimento selecionado, o botão "**Tudo**" coloca todos os procedimentos existentes em edição e o botão "**Cancelar**" encerra a ação de edição. O uso dos botões "**OK**" e "**Tudo**" abrem o programa de edição "**Editor**" semelhante ao programa **Bloco de notas** do *Windows*. Neste instante, defina como novo procedimento o nome "**PENTA1**" e acione o botão "**OK**". Veja na figura 3.11 a apresentação do modo **Editor** com a definição inicial do novo procedimento.

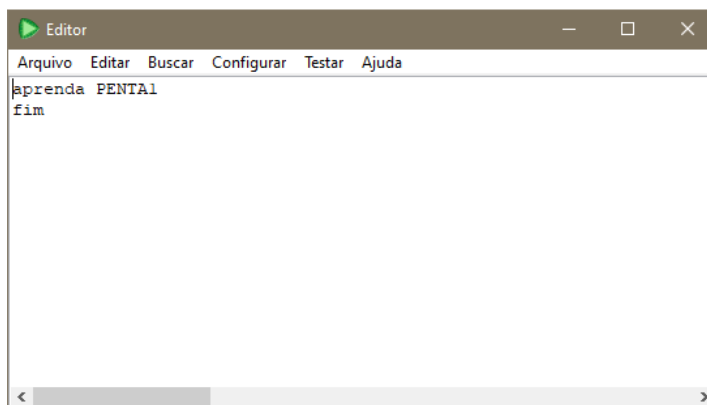


Figura 3.11 - Tela do programa "Editor".

Entre as instruções **aprenda PENTA1** e **fim** escreva com deslocamento de dois espaços a instrução **REPITA 5 [PF 50 PD 360 / 5]** como mostrado na figura 3.12 e acione em seguida a opção "**Guardar e Sair**" do menu "**Arquivo**" do programa **Editor**.

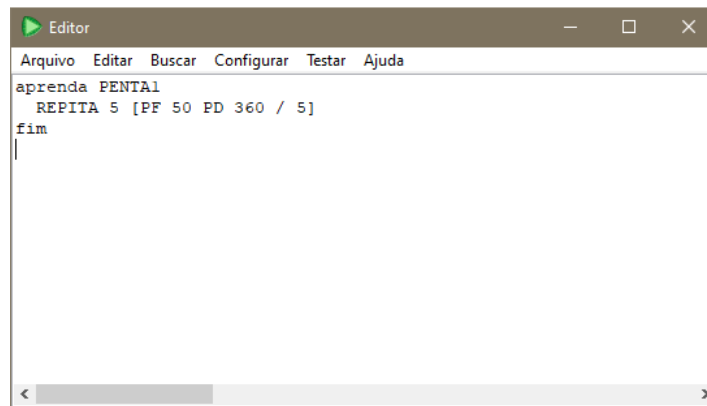


Figura 3.12 - Definição do procedimento "PENTA1".

Agora execute a instrução:

```

TAT
LJC
PENTA1

```

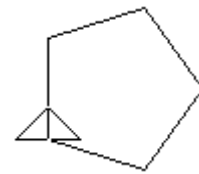


Figura 3.13 - Pentágono.

A figura 3.13 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:

```

TAT
LJC
REPITA 5 [PENTA1 PD 72]

```

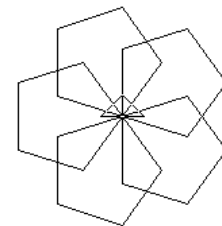


Figura 3.14 - Pentágono 1.

A figura 3.14 mostra o resultado desta operação.

Depois execute a instrução:

```

TAT
LJC
REPITA 10 [PENTA1 PD 36]

```

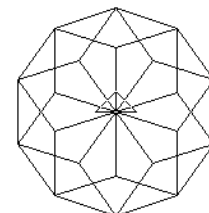


Figura 3.15 - Pentágono 2.

A figura 3.15 mostra o resultado desta operação.

### 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 selecione a opção "**Editar...**" menu "**Arquivo**" e será aberta a caixa de diálogo "**Editar**". Neste momento, acione o botão "**OK**" e será apresentado o programa **Editor** como indica a figura 3.16.



Figura 3.16 - Programa "Editor".

Antes de escrever os procedimentos necessários acione as teclas de atalho "<Ctrl> + <A>" e em seguida tecla "<Del>" e escreva, na sequência, as instruções:

```
APRENDAR MEIOCIRC
  REPITA 90 [PF 1 PD 1]
FIM
```

```
APRENDAR PETALA
  MEIOCIRC
  PD 90
  MEIOCIRC
FIM
```

```
APRENDAR FLOR
  TAT
  REPITA 8 [PETALA PD 45]
FIM
```

A figura 3.17 mostra o conjunto de procedimentos definidos no programa **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.

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**".



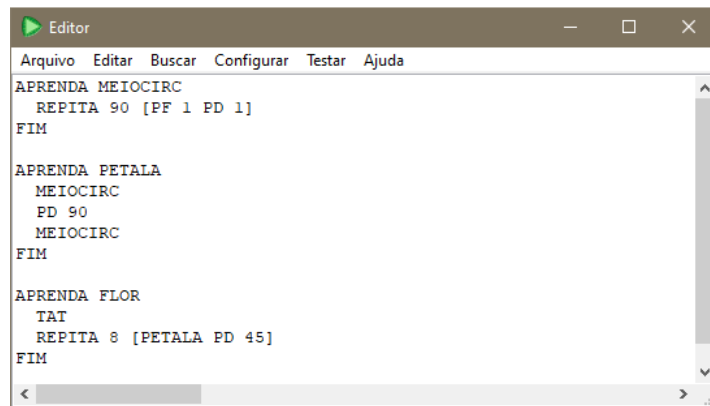


Figura 3.17 - Programa "Editor" com o conjunto de sub-rotinas.

Para encerrar o programa **Editor** selecione a opção "**Guardar e Sair**" menu "**Arquivo**" ou acione as teclas de atalho "**<Ctrl> + <D>**". Na sequência execute a instrução:

**FLOR**

A figura 3.18 mostra o resultado da execução do procedimento "**FLOR**".

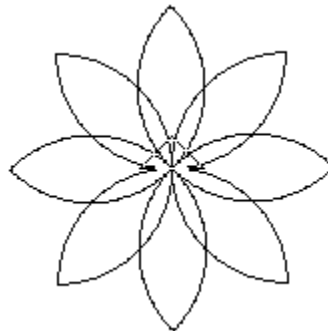


Figura 3.18 - Execução do procedimento "FLOR".

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

**TAT**

**REPITA 20 [PETALA PD 36]**

A Figura 3.19 mostra a imagem de uma flor com vinte pétalas.

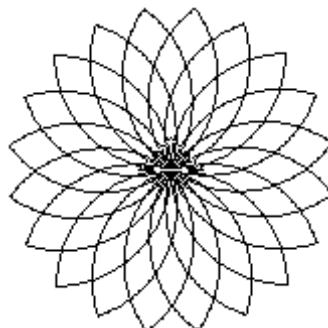


Figura 3.19 - Execução do procedimento "PETALA" para criar flor.

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]**  
**ESCREVA :PAISES**

A figura 3.20 mostra o resultado da ação anterior na área de ação do modo texto.



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

Figura 3.20 - Definição e apresentação de variável.

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

**ATRIBUA "NOME "Manzano**  
**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 3.21 a diferença de resultado apresentado para cada uma das formas de acesso.

```
ESCREVA :PAISES
Brasil França Argentina
ESCREVA "PAISES
PAISES
```

Figura 3.21 - 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 da primitiva **VALOR** antes no nome da variável identificada com aspa inglesa. Observe a instrução seguinte:

**ESCREVA VALOR "PAISES**

O efeito no uso da primitiva **VALOR** antes no nome da variável com aspa inglesa, ou seja, a instrução **ESCREVA VALOR "PAISES** é exatamente o mesmo resultado obtido a partir da execução da instrução **ESCREVA :PAISES**. Veja a figura 3.22.

```
ESCREVA VALOR "PAISES
Brasil França Argentina
```

Figura 3.22 - 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, selecione a opção "**Editar...**" menu "**Arquivo**" e no campo da caixa de diálogo "**Editar**" informe o nome "**QUAD2 :TAMANHO**" como apresenta a figura 3.23.

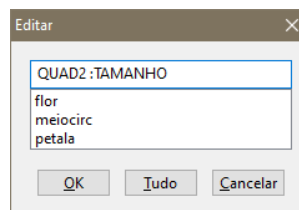


Figura 3.23 - Caixa de diálogo "Editar" com uso de variável.

Acione, na sequência, o botão "**OK**" e informe no programa **Editor** entre a apresentação das instruções "**aprenda QUAD2 :TAMANHO**" e "**fim**" instrução:

**REPITA 4 [PF :TAMANHO PD 90]**

Veja na figura 3.24 como deverá ficar a definição do procedimento.



Figura 3.24 - Programa "Editor" com uso de variável no procedimento.

Para fechar o **Editor** execute "**Arquivo/Guardar e Sair**" ou acione as teclas "<Ctrl> + <D>" e veja na sequência a execução sucessivamente das seguintes instruções:

```
LJC TAT
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 3.25.

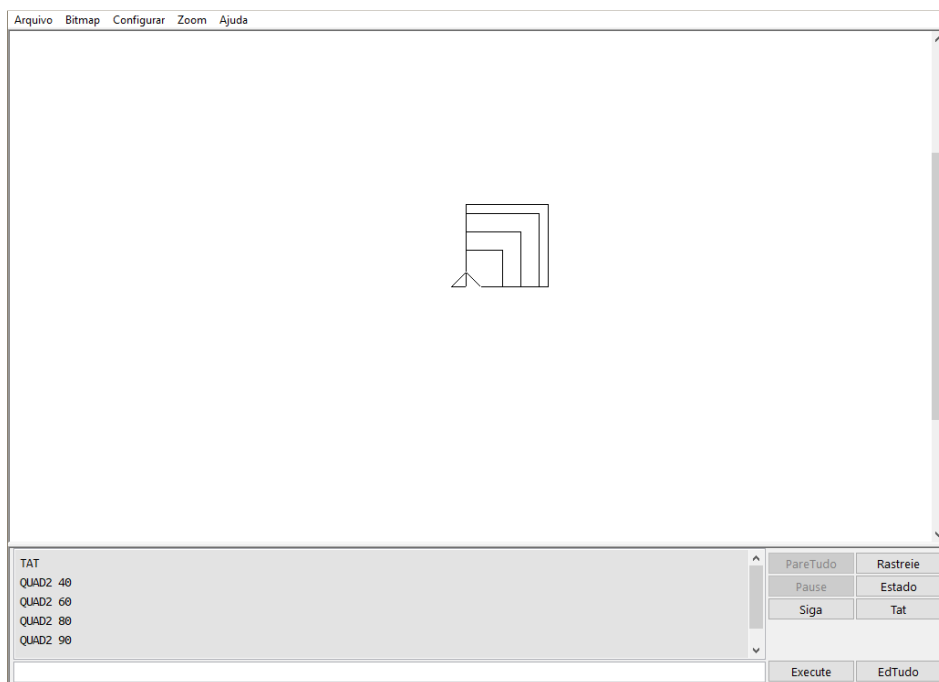


Figura 3.25 - Resultado a partir do uso de procedimento com parâmetro.

A partir desses recursos é possível criar um procedimento mais "inteligente" que a partir da informação do tamanho de traço e quantidade de lados desenha qualquer figura entre "3" e "360" graus. Para tanto, selecione a opção "**Editar...**" menu "**Arquivo**", acione o botão "**OK**" e as teclas de atalho "<Ctrl> + <A>", tecle "<Del>" e entre o seguinte código:

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

Ao término desta definição feche o programa **Editor** com "**Arquivo/Guardar e Sair**" ou acione as teclas "<Ctrl> + <D>".

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 as primitivas **SE/SESENÃO** que permitem tomar decisões a partir do estabelecimento de duas formas de citação. Observe a estrutura sintática:

**SE** <condição> <[ação verdadeira]>

**SESENÃO** <condição> <[ação verdadeira]> <[ação falsa]>

Onde, o indicativo "**condição**" (não importando **SE** ou **SESENÃO**) refere-se a definição de uma relação lógica que devolve como resposta um dos valores lógicos **FALSO** ou **VERD**.. O indicativo "[**ação verdadeira**]" corresponde a definição da ação a ser realizada dentro de uma lista caso a condição seja verdadeira, tanto para **SE** como para **SESENÃO**. O indicativo "[**ação falsa**]" é executado quando a condição para **SESENÃO** for falsa. A primitiva **SE** é usada na tomada de decisão simples e a primitiva **SESENÃO** é usada na 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
<>	DIFERENTE DE

Tabela 3.2 - Operadores relacionais.

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

ESCREVA 1 = 1 (mostra: VERD)

ESCREVA 1 <> 1 (mostra: FALSO)

ESCREVA 1 < 2 (mostra: VERD)

ESCREVA 1 > 2 (mostra: FALSO)

ESCREVA 1 >= 1 (mostra: VERD)

ESCREVA 2 <= 2 (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. Desta forma, selecione a opção "**Editar...**" menu "**Arquivo**", acione botão **OK**, acione as teclas de atalho "<Ctrl> + <A>", tecle "<Del>" e entre o seguinte código:

```
APRENDA MAXIMO :A :B
  SESENÃO :A > :B [ESC :A] [ESC :B]
FIM
```

Ao término desta definição feche o programa **Editor** com "**Arquivo/Guardar e Sair**" ou acione as teclas "**<Ctrl> + <D>**". Em seguida execute separadamente as instruções:

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

Veja na figura 3.26 a apresentação do resultado das operações.



Figura 3.26 - 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. Desta forma, selecione a opção "**Editar...**" menu "**Arquivo**", acione botão "**OK**", acione as teclas de atalho "**<Ctrl> + <A>**", tecle "**<Del>**" e entre o seguinte código:

```
APRENDA PAR :N
  SESENÃO (RESTO :N 2) = 0 [ESCREVA [Ok]] [ESCREVA [Erro]]
FIM
```

Ao término desta definição feche o programa **Editor** com "**Arquivo/Guardar e Sair**" ou acione as teclas "**<Ctrl> + <D>**". Em seguida execute separadamente as instruções:

```
PAR 2
PAR 3
```

Veja na figura 3.27 a apresentação do resultado das operações.



Figura 3.27 - 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. Desta forma, selecione a opção "**Editar...**" menu "**Arquivo**", acione botão "**OK**", acione as teclas de atalho "**<Ctrl> + <A>**", tecle "**<Del>**" e entre o seguinte código:

APRENDA IMPAR :N

SE (RESTO :N 2) <> 0 [ESCREVA [Ok]]

FIM

Ao término desta definição feche o programa **Editor** com "**Arquivo/Guardar e Sair**" ou acione as teclas "<Ctrl> + <D>". Em seguida execute separadamente as instruções:

IMPAR 3

IMPAR 2

Veja na figura 3.28 a apresentação do resultado das operações.

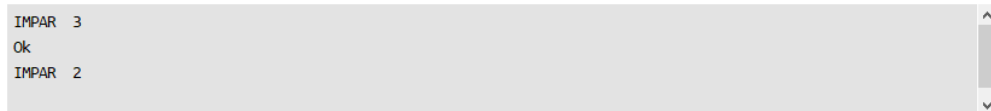


Figura 3.28 - Resultado do procedimento "IMPAR".

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**". Para tanto, selecione a opção "**Editar...**" menu "**Arquivo**", acione o botão "**OK**" e em seguida as teclas de atalho "<Ctrl> + <A>", tecle "<Del>" e entre o seguinte código (não se preocupe com a primitiva **OU**, adiante a sua explicação:

APRENDA FLORAL :N

SESENÃO (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.

Ao término desta definição feche o programa **Editor** com "**Arquivo/Guardar e Sair**" ou acione as teclas "<Ctrl> + <D>". Em seguida execute separadamente as instruções:

TAT FLORAL 0

TAT FLORAL 1

TAT FLORAL 2

TAT FLORAL 3

TAT FLORAL 4

TAT FLORAL 5

TAT FLORAL 6

TAT FLORAL 7

TAT FLORAL 8

Veja na figura 3.29 a apresentação dos vários resultados obtidos a partir dos valores numéricos entre 1 e 7 válidas para o procedimento "**FLORAL**".

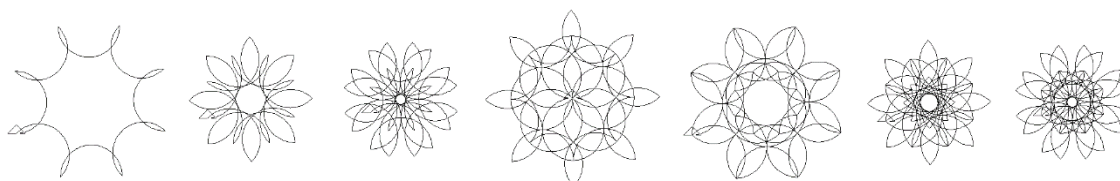


Figura 3.29 - Resultado do procedimento "FLORAL".

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 os operadores lógicos **E** e **NÃO**. Atente para a tabela 3.3.

OPERADOR	SIGNIFICADO	RESULTADO
E	CONJUNÇÃO	VERDADEIRO quando todas as condições forem verdadeiras.
OU	DISJUNÇÃO	VERDADEIRO quando pelo menos uma das condições for verdadeira.
NÃO	NEGAÇÃO	VERDADEIRO quando condição for falsa e FALSO 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 verdades 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, selecione a opção "Editar..." menu "Arquivo", acione o botão "OK" e as teclas de atalho "<Ctrl> + <A>", tecle "<Del>" e entre o código:

```
APRENDA TESTE_E :N
  SESENÃO (E :N >= 1 :N <= 9) [
    ESC [Valor está na faixa de 1 a 9.]
  ]
  ESC [Valor não está na faixa de 1 a 9.]
]
FIM
```

Ao término, feche o programa **Editor** com "Arquivo/Guardar e Sair" ou acione as teclas "<Ctrl> + <D>". Em seguida execute separadamente as instruções a seguir:

```
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, selecione a opção **"Editar..."** menu **"Arquivo"**, acione o botão **"OK"** e as teclas de atalho **"<Ctrl> + <A>"**, tecle **"<Del>"** e entre o código:

APRENDA TESTE\_OU :TEMPO

```
SESENÃO (OU MAIÚSCULAS :TEMPO = "FRIO MAIÚSCULAS :TEMPO = "CHUVOSO) [
  ESC [Tempo ruim, proteja-se.]
][
  ESC [Tempo bom, aproveite.]
]
FIM
```

Ao término, feche o programa **Editor** com **"Arquivo/Guardar e Sair"** ou acione as teclas **"<Ctrl> + <D>"**. Em seguida execute separadamente as instruções a seguir:

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

Procure fazer mais testes com outros valores. Note o uso da função **MAIÚSCULAS** para garantir que não importando o formato do texto informado este será transformado em texto maiúsculo antes de ser verificado na condição.

#### 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 mostra o quadrado deste valor caso este valor *não seja maior que "5"*. Desta forma, selecione a opção **"Editar..."** menu **"Arquivo"**, acione o botão **"OK"** e as teclas de atalho **"<Ctrl> + <A>"**, tecle **"<Del>"** e entre o código:

APRENDA TESTE\_NAO :V

```
SE (NÃO :V > 5) [
  ESC POTÊNCIA :V 2
]
FIM
```

Ao término, feche o programa **Editor** com "**Arquivo/Guardar e Sair**" ou acione as teclas "<Ctrl> + <D>". Em seguida execute separadamente as instruções a seguir:

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. Assim sendo, selecione a opção "**Editar...**" menu "**Arquivo**", acione botão "**OK**", acione as teclas de atalho "<Ctrl> + <A>", tecle "<Del>" e entre o seguinte código:

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

Ao término desta definição feche o programa **Editor** com "**Arquivo/Guardar e Sair**" ou acione as teclas "<Ctrl> + <D>". Em seguida execute separadamente as instruções:

TAT ESPIRAL 4 122 60

TAT ESPIRAL 4 92 50

Veja na figura 3.30 a apresentação do resultado das operações.

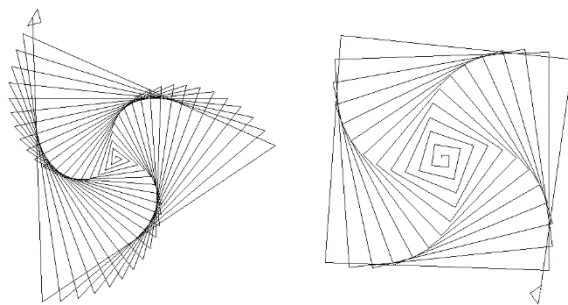


Figura 3.30 - Resultado do efeito recursivo sobre imagens espiraladas.

A partir do código do procedimento **"ESPIRAL"** note a definição em cor verde da decisão de aterramento 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:

```
TAT ESPIRAL 4 120 60
TAT ESPIRAL 4 90 60
```

Veja os resultados na figura 3.31:

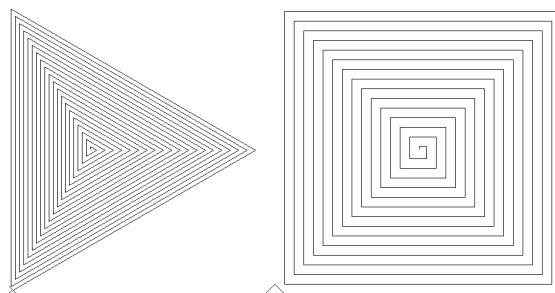


Figura 3.31 - Mais efeitos recursivos sobre imagens espiraladas.

No sentido de apresentar mais ações sobre recursões veja na próxima sequência de procedimentos os recursos para se desenharmos uma treliça (unidade definida a partir de cinco triângulos) como base de sustentação para uma ponte (adaptado, MULLER, 1998 p. 345). Assim sendo, selecione a opção **"Editar..."** menu **"Arquivo"**, acione botão **"OK"**, acione as teclas de atalho **"<Ctrl> + <A>"**, tecla **"<Del>"** e entre 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

Ao término desta definição feche o programa **Editor** com "**Arquivo/Guardar e Sair**" ou acione as teclas "**<Ctrl> + <D>**". Em seguida execute a instrução:

**TAT TRELICA 10**

Veja na figura 3.32 a apresentação do resultado do efeito de recursividade.



Figura 3.32 - 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 "lembrará" 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**" do menu "**Arquivo**". Será apresentada a caixa para gravação do arquivo de procedimento indicada na figura 3.33.

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

Para realizar um teste, encerre o ambiente "**FMSLogo**" 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".

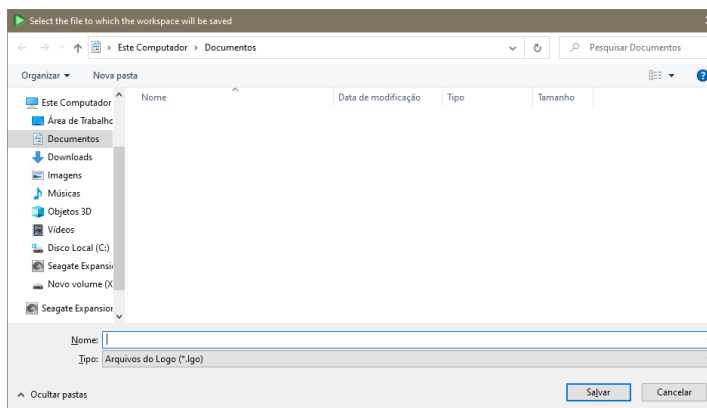


Figura 3.33 - Caixa para gravação de arquivos.

Para fazer Logo "lembrar-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 3.34.

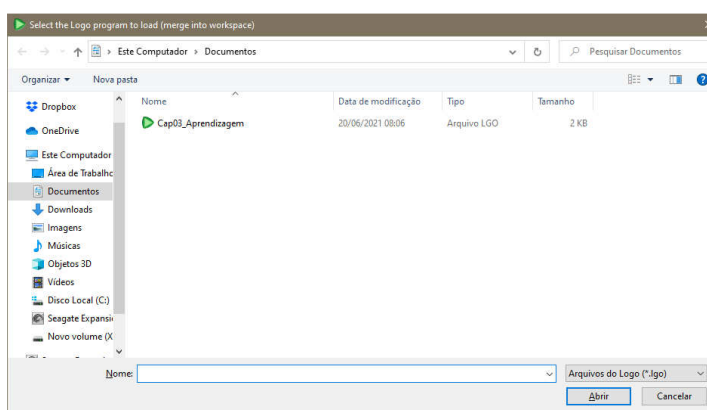


Figura 3.34 - Caixa para leitura de arquivos.

Selecione, com o ponteiro do mouse, o arquivo desejado, neste caso "**Cap03\_Aprendizagem**" e acione o botão "**Abrir**". 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 graus 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 graus 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 para frente de "**30**" passos.
6. Criar procedimento chamado **ICOSAGONO** (sem acento) que desenhe a figura de mesmo nome a partir do uso da primitiva **REPITA** com tamanho "**35**" movimentado para trás.

## ANOTAÇÕES

[illegible]



---

## 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.

*Logo* permite a entrada de dados em tempo de execução. Para isso deve-se fazer uso de uma das primitivas: **LEIACARACTERE** (**LEIACAR**), **LEIALISTA** (**LEIAL**) e **LEIAPALAVRA** (**LEIAP**) que possuem como estrutura sintática o estilo:

**LEIACARACTERE** <conteúdo>

**LEIALISTA** <conteúdo>

**LEIAPALAVRA** <conteúdo>

Onde, o indicativo "**conteúdo**" refere-se ao conteúdo informado na entrada que para ser usado deve ser vinculado a uma variável. A diferença entre os comandos está no fato de que **LEIACAR** efetua a leitura apenas de um caractere da entrada (não importando o tamanho da entrada), **LEIALISTA** efetua a leitura de um conjunto de caracteres na forma de lista e **LEIAPALAVRA** efetua a leitura de um conjunto de caracteres de forma livre (sendo a forma de entrada mais flexível).

O uso dos comandos **LEIACARACTERE**, **LEIALISTA**, **LEIAPALAVRA** deve ser realizado em conjunto de outro comando como **ESCREVA** ou **ATRIBUA**. Quando um desses comandos é usado é apresentada uma caixa de diálogo **Modo de Entrada** como indica a figura 4.1.

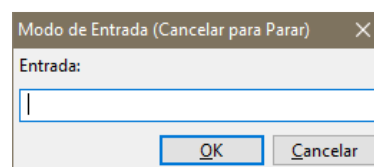


Figura 4.1 - Caixa de diálogo "Modo de Entrada".

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**". Assim sendo, selecione a opção "**Editar...**" menu "**Arquivo**", acione o botão "**OK**" e as teclas de atalho "**<Ctrl> + <A>**", tecle "**<Del>**" e entre o seguinte código:

**APRENDA IDADE**

**ATRIBUA "VALOR LEIAP**

**DEVOLVA (SE :VALOR >= 18 [ESC [Maior de idade]] [ESC [Menor de idade]])**

**FIM**

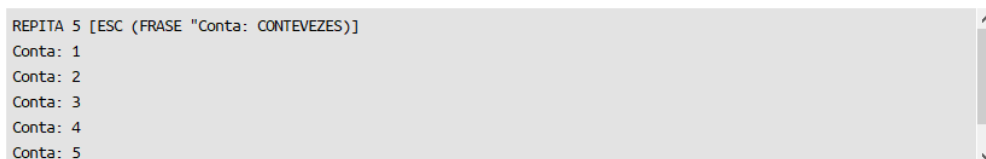
Ao término desta definição feche o programa **Editor** com "**Arquivo/Guardar e Sair**" ou acione as teclas "<Ctrl> + <D>". Em seguida execute o procedimento "**IDADE**" algumas vezes e informe valores diferentes. Veja os resultados apresentados.

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**.

Para mostrar o conteúdo da primitiva **CONTEVEZES** além da primitiva **ESCREVA** é necessário usar em conjunto o comando **FRASE (FR)** 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 (FRASE "Conta: CONTEVEZES)]**

Veja o resultado obtido junto a figura 4.2.



```

REPITA 5 [ESC (FRASE "Conta: CONTEVEZES)]
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 **FRASE**. O resultado da tabuada a ser apresentado deve ser delimitado entre parênteses estabelecendo assim uma estrutura de lista.

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 um novo comando chamado **CONTEVEZES** 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, selecione a opção "**Editar...**" menu "**Arquivo**", acione o botão "**OK**" as teclas de atalho "<Ctrl> + <A>", tecle "<Del>" e entre o seguinte código:

#### APRENDA TABUADA

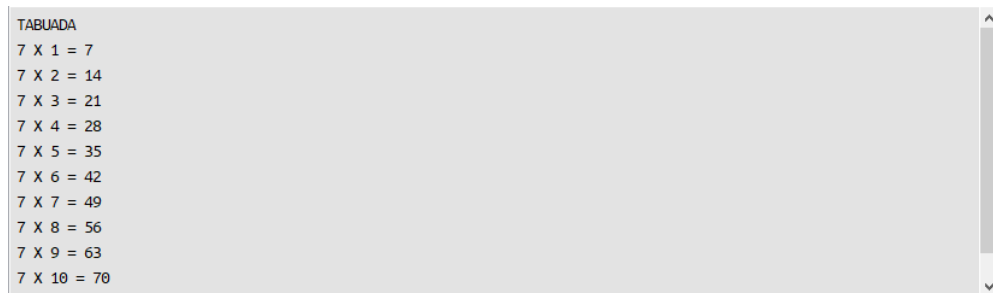
```

ATRIBUA "NUMERO LEIAP
SESENÃO (E :NUMERO >= 2 :NUMERO <= 9) [
  REPITA 10 [ESC (
    FR :NUMERO "X CONTEVEZES "= :NUMERO * CONTEVEZES
  )]
][
  ESC [Valor fora da faixa]
]
FIM
    
```

Ao término, feche o programa **Editor** com "**Arquivo/Guardar e Sair**" ou acione as teclas "<Ctrl> + <D>". 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
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.

O resultado apresentado pelo procedimento "TABUADA" pode ter seu visual melhorado com auxílio do comando **FORMATONÚMERO** que opera segundo a estrutura sintática:

**FORMATONÚMERO** <número> <expoente> <mantissa>

Onde, o indicativo "**número**" refere-se ao valor numérico apresentado, "**expoente**" é a largura total mínima incluindo o ponto decimal e "**mantissa**" é a quantidade de casas após o ponto decimal.

Para fazer uso do recurso **FORMATONÚMERO** considere um procedimento chamado "**TABUADA2**" semelhante ao procedimento **TABUADA** que apresente os valores numéricos alinhados na direita para a esquerda. Selecione a opção "**Editar...**" menu "**Arquivo**", acione o botão "**OK**" e as teclas de atalho "<Ctrl> + <A>", tecle "<Del>" e entre o seguinte código:

### APRENDA TABUADA2

```

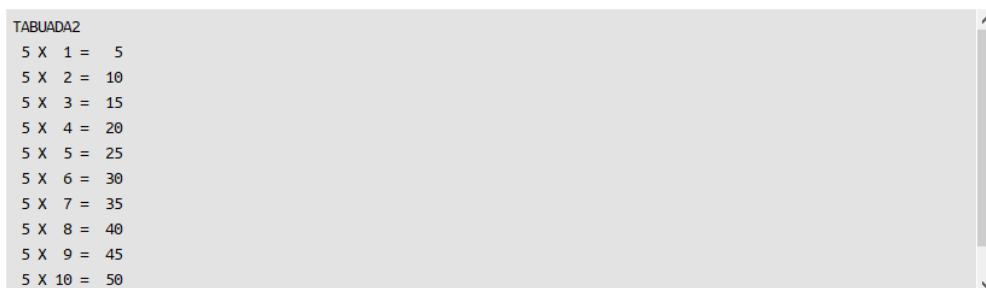
ATRIBUA "NUMERO LEIAP
SESENÃO (E :NUMERO >= 2 :NUMERO <= 9) [
  REPITA 10 [ESC (
    FR
      FORMATONÚMERO :NUMERO 2 0
    "X
      FORMATONÚMERO CONTEVEZES 2 0
    "=
      FORMATONÚMERO (:NUMERO * CONTEVEZES) 3 0
  )]
][
  ESC [Valor fora da faixa]
]
FIM

```

Ao término, feche o programa **Editor** com "**Arquivo/Guardar e Sair**" ou acione as teclas "<Ctrl> + <D>". Em seguida execute separadamente as instruções após fazer uso do comando **TAT**:

## TABUADA2

Quando apresentada a caixa de diálogo entre o valor "5" e veja a tabuada apresentada como indicado na figura 4.4.



5 X 1 =	5
5 X 2 =	10
5 X 3 =	15
5 X 4 =	20
5 X 5 =	25
5 X 6 =	30
5 X 7 =	35
5 X 8 =	40
5 X 9 =	45
5 X 10 =	50

Figura 4.4 - Resultado de uma tabuada com valores formatados (alinhados).

Perceba que a partir do que foi mostrado você tem em mãos alguns recursos que lhe permitem explorar diversos aspectos da "inteligência artificial" da linguagem Logo.

## 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 **SORTEIENÚMERO** que é operado segundo a estrutura sintática:

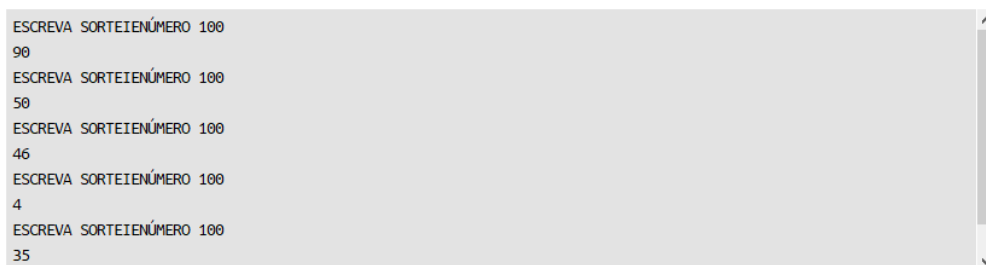
**SORTEIENÚMERO** <número>

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":

```
ESCREVA SORTEIENÚMERO 100
ESCREVA SORTEIENÚMERO 100
ESCREVA SORTEIENÚMERO 100
ESCREVA SORTEIENÚMERO 100
ESCREVA SORTEIENÚMERO 100
```

Veja que a cada execução o valor sorteado retornado é diferente um do outro como comprova a figura 4.5.



```
ESCREVA SORTEIENÚMERO 100
90
ESCREVA SORTEIENÚMERO 100
50
ESCREVA SORTEIENÚMERO 100
46
ESCREVA SORTEIENÚMERO 100
4
ESCREVA SORTEIENÚMERO 100
35
```

Figura 4.5 - 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". Selecione a opção "Editar..." menu "Arquivo", acione o botão "OK" e as teclas de atalho "<Ctrl> + <A>", tecle "<Del>" e entre o seguinte código:

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

Ao término, feche o programa **Editor** com "Arquivo/Guardar e Sair" ou acione as teclas "<Ctrl> + <D>". Em seguida execute a instrução:

**TAT MALUQUINHO**

Veja o resultado da operação na figura 4.6, ressaltando que em seu sistema a imagem tende a ser completamente diferente.

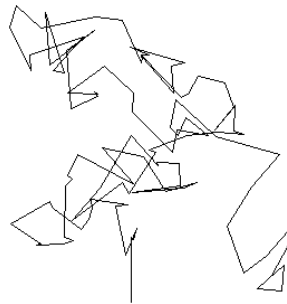


Figura 4.6 - Resultado de figura geométrica aleatória.

O comando **SORTEIENÚMERO** 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 "FMSLogo" na memória executar o comando **SORTEIENÚMERO**, sair, carregar novamente o ambiente e executar novamente o comando **SORTEIENÚMERO** terá valores sempre diferentes.

Mas e se houver a necessidade de em certo momento fazer o sorteio de uma série de valores que necessitem se repetir? Por exemplo, desejando-se obter inicialmente cinco valores realmente aleatórios, mas a partir daí é necessário quando se pedir para gerar o sorteio obter os mesmos cinco valores anterior?

Para este caso pode-se lançar mão do comando **REPITASORTEIE** que deverá ser utilizado, obviamente, sempre antes do comando **SORTEIENÚMERO**. O comando **REPITASORTEIE** é operado a partir da estrutura sintática:

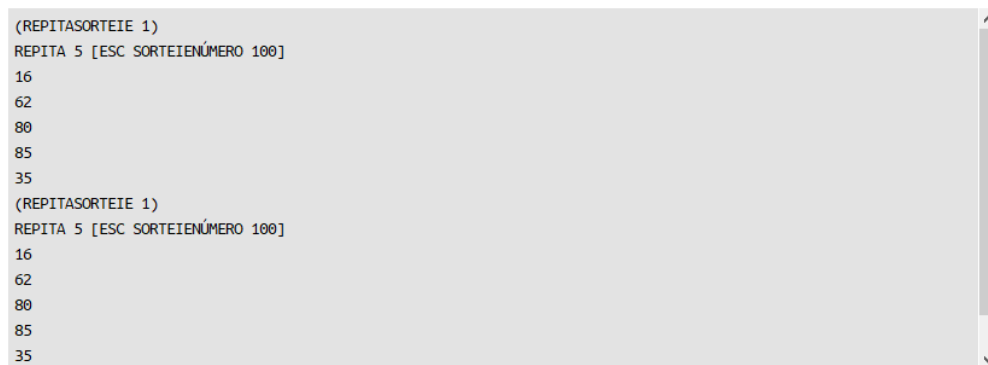
**(REPITASORTEIE <semente>)**

Onde, o indicativo "**semente**" refere-se ao estabelecimento de um valor que serve como ponto de partida, de inicialização ao comando **REPITASORTEIE** que deve ser definido entre parênteses. Desta forma utilizando-se a mesma semente obtém-se a mesma sequência de valores randômicos. O valor de semente pode ser qualquer número inteiro que determina a sequência fixa de sorteio.

Como exemplo, execute separadamente as instruções para sortear duas vezes os mesmos cinco valores entre "0" e "100":

```
(REPITASORTEIE 1)
REPITA 5 [ESC SORTEIENÚMERO 100]
(REPITASORTEIE 1)
REPITA 5 [ESC SORTEIENÚMERO 100]
```

Veja que a execução das duas séries de valor sorteados a partir da instrução **(REPITASORTEIE 1)** é exatamente a mesma como mostra a figura 4.7.



```
(REPITASORTEIE 1)
REPITA 5 [ESC SORTEIENÚMERO 100]
16
62
80
85
35
(REPITASORTEIE 1)
REPITA 5 [ESC SORTEIENÚMERO 100]
16
62
80
85
35
```

Figura 4.7 - Resultado repetidos de valores sorteados.

### 4.3 - Coordenadas

Todas 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 **ESCREVA**. 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**. Caso deseje traçar na mudança de coordenada use o comando **USELÁPIS**.

Para realizar alguns testes de posicionamento e verificação de posicionamento acione primeiro o comando **TAT** e em seguida execute as instruções:

```
USENADA
ESCREVA POS
MUDEX 50
MUDEY -30
ESCREVA POS
```

A figura 4.8 mostra os resultados apresentados. Veja os valores gerados após o uso da instrução **ESCREVA POS** na coordenada "0 0" com marcação em vermelho e na coordenada "50 -30" com marcação em azul.

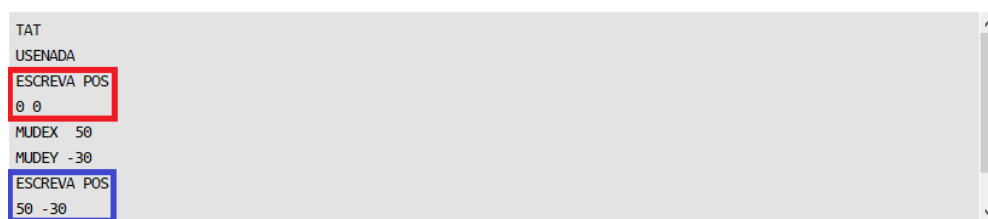


Figura 4.8 - Identificando posições cartesianas.

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 4.9 representa esquematicamente os movimentos de salto com o par de comandos **MUDEX 30** com **MUDEY 20** e **MUDEX -30** com **MUDEY -10**.

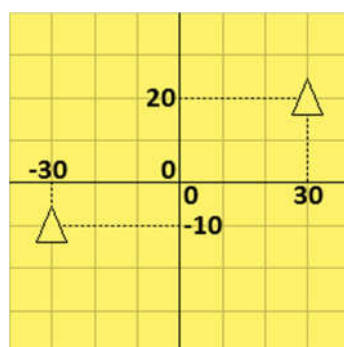


Figura 4.9 - Posições cartesianas da tartaruga.

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:

```
LJC
TAT
ESCREVA POS
MUDEXY -50 25
ESCREVA POS
CENTRO
ESCREVA POS
```

Na sequência execute o comando **USELÁPIS**. A figura 4.10 mostra os valores das posições cartesianas após o uso dos comandos **MUDEXY** e **CENTRO**.

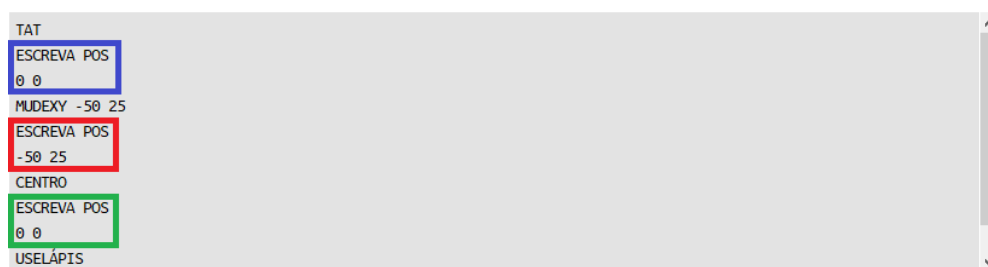


Figura 4.10 - Posições cartesianas após comandos "MUDEXY" e "CENTRO".

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 4.11, adaptada de Harvey (1997, p. 184).

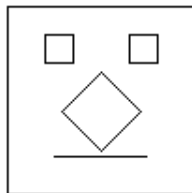


Figura 4.11 - Face quadrada.

Apesar de simples a imagem da figura 4.11 apresenta um grau de complexidade interessante, pois diversas ações são necessárias. Assim sendo, selecione a opção "**Editar...**" menu "**Arquivo**", acione o botão "**OK**" e as teclas de atalho "<Ctrl> + <A>", tecle "<Del>" e entre o seguinte código:

```
APRENDA CABECA
  USELÁPIS
  REPITA 4 [PF 100 PD 90]
  USENADA
FIM
```

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

```
APRENDA OLHO.DIREITO
  USELÁPIS
  OLHO
  USENADA
  PT 15
FIM
```

```
APRENDA BOCA
  USENADA
  PF 20 PD 90 PF 25
  USELÁPIS
  PF 50
  USENADA
  PT 75
FIM
```

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

```
APRENDA OLHO.ESQUERDO
  USELÁPIS
  OLHO
  USENADA
  PF 45
FIM
```

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

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

Ao término, feche o programa **Editor** com "**Arquivo/Guardar e Sair**" ou acione as teclas "<Ctrl> + <D>". Em seguida execute separadamente as instruções após fazer uso do comando **TAT**:



**OT**  
**FACE**

Após executar o procedimento e visualizar a imagem da face quadrada execute o comando **MT** para reapresentar a tartaruga na área de ação do modo gráfico. Lembre-se que os comandos **OT** e **MT** são respectivamente as siglas dos comandos **OCULTETAT** 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, selecione a opção "**Editar...**" menu "**Arquivo**", acione o botão "**OK**" e as teclas de atalho "**<Ctrl> + <A>**", tecle "**<Del>**" e entre o seguinte código:

```
APRENDA POLIGOM :LADOS :TAMANHO
OT
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
MT
FIM
```

Ao término, feche o programa **Editor** com "**Arquivo/Guardar e Sair**" ou acione as teclas "**<Ctrl> + <D>**". Em seguida execute separadamente as instruções após fazer uso do comando **TAT**:

**POLIGOM 3 50**

Veja o resultado da operação na figura 4.12. 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.

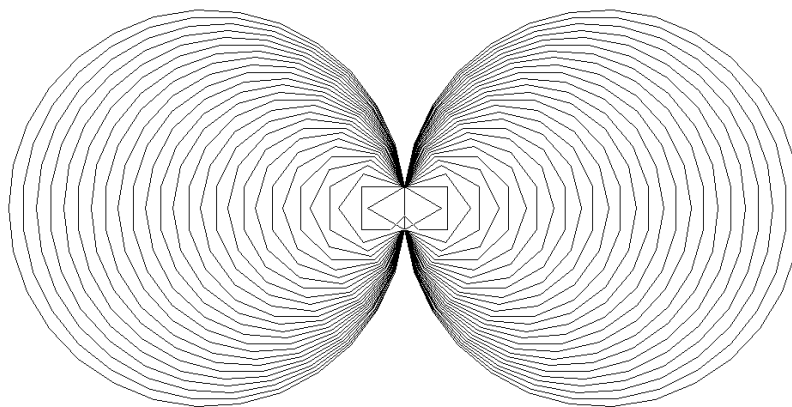


Figura 4.12 - Polígono com efeito de espelhamento.

#### 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** (**R**ed, **G**reen e **B**lue), 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 dezesseis tonalidades de cores básicas que podem ser usadas no "FMSLogo".

ÍNDICE	COR	CÓDIGO RGB
0	PRETO	[000 000 000]
1	AZUL	[000 000 255]
2	VERDE (LIMA)	[000 255 000]
3	CIANO (ÁGUA)	[000 255 255]
4	VERMELHO	[255 000 000]
5	MAGENTA (FÚCSIA)	[255 000 255]
6	AMARELO	[255 255 000]
7	BRANCO	[255 255 255]
8	MARROM	[155 096 059]
9	BRONZEADO	[197 136 018]
10	OLIVA	[100 162 064]
11	AZUL CELESTE	[120 187 187]
12	SALMÃO	[255 149 119]
13	ROXO (PÚRPURAMÉDIO)	[144 113 208]
14	LARANJA	[255 163 000]
15	CINZA CLARO	[183 183 183]

Tabela 4.1 - Índice de cores do ambiente FMSLogo.

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:

```
TAT FACE
```

A figura 4.13 mostra o resultado com fundo preto e imagem em tom amarelo.

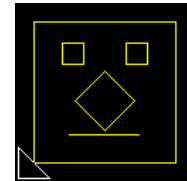


Figura 4.13 - Imagem no fundo preto em tom amarelo.

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

```
MUDECF 1
MUDECL 11
```

Para ver o resultado execute a instrução:

```
TAT FACE
```

A figura 4.14 mostra o resultado com fundo azul e imagem em tom azul celeste.

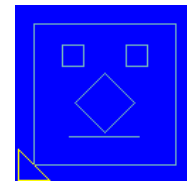


Figura 4.14 - Imagem no fundo azul em tom azul celeste.

**OBS:**

O uso de cores baseadas no **ÍNDICE** fica limitado aos valores de "0" a "15" indicados na tabela 4.1. Usar o padrão de cores a partir do formado [**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 "[255 165 000]" em fundo na cor azul marinho "[000 000 128]" (adaptado de Abelson, 1981, p. 22). A Figura 4.15 apresenta a imagem gerada pela execução da chamada do procedimento "**SOL 10**".

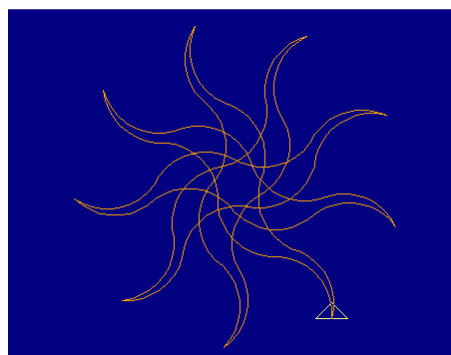


Figura 4.15 - Procedimento "SOL 10".

Selecione a opção "**Editar...**" menu "**Arquivo**", acione botão **OK**, acione as teclas de atalho "**<Ctrl> + <A>**", tecele "**<Del>**" e entre os seguintes códigos:

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

```
APRENDA RAO :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 [
    RAO :TAMANHO
    PARADIREITA 160
  ]
FIM
```

Ao término, feche o programa **Editor** com "**Arquivo/Guardar e Sair**" ou acione as teclas "**<Ctrl> + <D>**". Em seguida execute as instruções **TAT** 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
```

Além das mudanças na definição das cores do fundo da tela e do traço é possível fazer o preenchimento de certa cor dentro de uma figura. O preenchimento com certa cor é realizado com o uso do comando **MUDECP** (mude cor do preenchimento). Além de estabelecer a cor com o comando **MUDECP** é necessário fazer uso do comando **PINTE** para que a cor escolhida preencha a área da figura.

O procedimento "**LARANJA**" faz a apresentação de um quadrado nesta cor.

A fim de efetuar uma experimentação do preenchimento interno de uma imagem com cores o procedimento seguinte desenhará um quadrado laranja. Desta forma, selecione a opção "**Editar...**" menu "**Arquivo**", acione o botão "**OK**" e as teclas de atalho "**<Ctrl> + <A>**", tecle "**<Del>**" e entre o seguinte código:

```
APRENDA LARANJA
  REPITA 4 [PF 50 PD 90]
  USENADA
  MUDEPOS [30 30]
  MUDECP [255 163 0]
  PINTE
  MUDEPOS [0 0]
  MUDECL [0 0 0]
  USELÁPIS
FIM
```

Ao término, feche o programa **Editor** com "**Arquivo/Guardar e Sair**" ou acione as teclas "<Ctrl> + <D>".

Observe que no procedimento após a apresentação do quadrado se faz o recolhimento do lápis com o comando **USENADA**. Em seguida faz-se a movimentação da tartaruga para a coordenada "30" e "30", seleciona-se a cor "[255 163 0]" (*laranja*) com o comando **MUDECP** e efetua-se a pintura da imagem com o comando **PINTE**. Após a pintura da imagem faz-se com que a tartaruga volte para sua posição "CASA" com a instrução "**MUDEPOS [0 0]**", lembrando que poderá para esta ação ser usado o comando **CENTRO** e configura-se o lápis com tom de cor preto. Na sequência faz-se a ativação do lápis com o comando **USELÁPIS**.

Em seguida execute as instruções **TAT** e **LARANJA** e veja a imagem apresentada semelhante à figura 4.16.

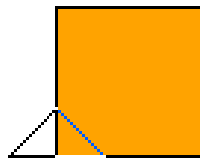


Figura 4.16 - Imagem com preenchimento.

#### 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 apropriada 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 4.17 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 4.17 - 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 4.18 após a chamada "**FRACTAL1 30**".

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

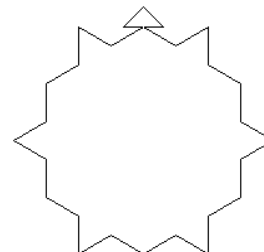


Figura 4.18 - Imagem do procedimento "FRACTAL1".

Note que com os procedimentos **"PONTA"** e **"FRACTAL1"** desenvolvidos foi efetuada uma imagem geométrica irregular (Figura 4.18) a partir de seis repetições da imagem gerada pelo procedimento **"PONTA"** (Figura 4.17).

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 4.19. Use a chamada de instrução **"FRACTAL2 30"**.

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

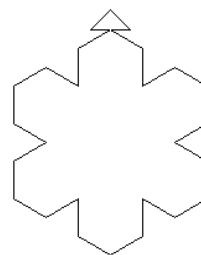


Figura 4.19 - Imagem do procedimento "FRACTAL2".

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 4.20. Use **"FRACTAL3 30"**.

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

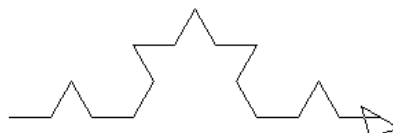


Figura 4.20 - Imagem do procedimento "FRACTAL3".

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 4.21 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
```

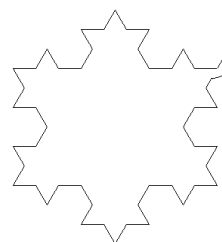


Figura 4.21 - Imagem do procedimento "FRACTAL4".

O procedimento "SAMAMBAIA" mostra a imagem de um fractal a partir da definição de determinado número de folhas. A Figura 4.22 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 4.22 - 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**, **FAÇA.ATÉ** e **FAÇAPARA**.

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 iterativo 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". Assim sendo, selecione a opção "Editar..." menu "Arquivo", acione o botão "OK" e as teclas de atalho "<Ctrl> + <A>", tecle "<Del>" e entre o seguinte código:

```

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

```

Ao término, feche o programa **Editor** com "Arquivo/Guardar e Sair" ou acione as teclas "<Ctrl> + <D>". Em seguida execute as instruções **TAT**, **LJC** e "CONTAGEM1" e veja a imagem apresentada semelhante à figura 4.23.

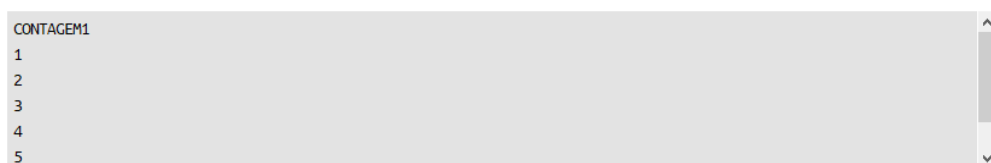


Figura 4.23 - Apresentação da contagem de "1" a "5" com "ENQUANTO".

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 (MUDEDC)**. A primitiva **MUDEDC** pode ser usada como substituta da diretiva **PD**. Para tanto, selecione **"Editar..."** no menu **"Arquivo"**, acione o botão **"OK"** e as teclas de atalho **"<Ctrl> + <A>"**, tecle **"<Del>"** e entre o seguinte código:

APRENDA PENTAGRAMA

MUDEDIREÇÃO 18

ATRIBUA "I 1

ENQUANTO [:I <= 5] [

PF 100

PD 144

ATRIBUA "I :I + 1

]

FIM

Após término, feche o programa **Editor** com **"Arquivo/Guardar e Sair"** ou acione as teclas **"<Ctrl> + <D>"**. Em seguida execute as instruções **TAT** e **"PENTAGRAMA"** e veja a imagem apresentada semelhante à figura 4.24.

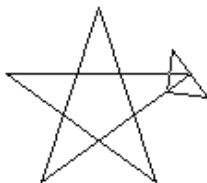


Figura 4.24 - Apresentação de imagem gerada com comando **"ENQUANTO"**.

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**. Usar as 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.

Além de um comando para a execução de laços pré-teste com fluxo condicional verdadeiro há a primitiva **FAÇA.ATÉ** que opera laços com fluxo falso, ou seja, executa um laço até que a condição se torne verdadeira. Este tipo de laço aceita a realização de um trecho de código, pelo menos uma vez, antes da verificação condicional.

Para ver o funcionamento do comando **FAÇA.ATÉ** considere o procedimento **"CONTAGEM2"** que apresenta os valores de contagem entre **"1"** e **"5"** de **"1"** em **"1"**. Assim sendo, selecione a



opção "**Editar...**" menu "**Arquivo**", acione o botão "**OK**" e as teclas de atalho "<Ctrl> + <A>", tecle "<Del>" e entre o seguinte código:

#### APRENDA CONTAGEM2

```
ATRIBUA "I 1
FAÇA.ATÉ [
  ESCREVA :I
  ATRIBUA "I :I + 1
]:I > 5]
```

FIM

Ao término, feche o programa **Editor** com "**Arquivo/Guardar e Sair**" ou acione as teclas "<Ctrl> + <D>". Veja que neste comando a escrita do valor e a atribuição de mais "1" ocorre uma vez antes da avaliação da condição "[I > 5]".

As primitivas (comandos) **ENQUANTO** e **FAÇA.ATÉ** são recursos que operam repetição de trechos de códigos utilizando-se condições. Por esta razão podem ser usados para a execução de laços interativos e iterativos (como faz a primitiva **REPITA**)

A primitiva **FAÇAPARA** é semelhante a **REPITA** por realizar ações iterativas. Para ver o funcionamento de **FAÇAPARA** considere o procedimento "**CONTAGEM3**" que mostra os valores de contagem entre "1" e "5" de "1" em "1". Assim sendo, selecione a opção "**Editar...**" menu "**Arquivo**", acione o botão "**OK**", acione as teclas de atalho "<Ctrl> + <A>", tecle "<Del>" e entre o seguinte código:

#### APRENDA CONTAGEM3

```
FAÇAPARA [I 1 5 1] [
  ESCREVA :I
]
```

FIM

Note que a primitiva **FAÇAPARA** 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".

Ao término, feche o programa **Editor** com "**Arquivo/Guardar e Sair**" ou acione as teclas "<Ctrl> + <D>". Em seguida execute as instruções **TAT**, **LIC** e "**CONTAGEM3**".

Já que foi feita uma estrela de cinco pontas (pentagrama) com o comando **ENQUANTO** será feita uma estrela de quarenta pontas com o comando **FAÇAPARA**. Para tanto, defina o procedimento "**ESTRELA40**". Assim sendo, selecione a opção "**Editar...**" menu "**Arquivo**", acione botão "**OK**", acione as teclas de atalho "<Ctrl> + <A>", tecle "<Del>" e entre o seguinte código:

#### APRENDA ESTRELA40

```
FAÇAPARA [I 1 40] [
  PT 200
  PE 170
]
```

FIM

Ao término, feche o programa **Editor** com "**Arquivo/Guardar e Sair**" ou acione as teclas "**<Ctrl> + <D>**". Em seguida execute as instruções **TAT** e "**ESTRELA40**" e veja a imagem apresentada semelhante à figura 4.25.

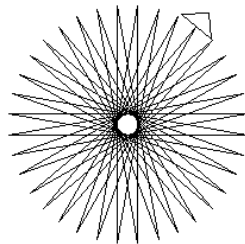


Figura 4.25 - Apresentação de imagem gerada com comando "PARA".

Os comandos: **REPITA**, **ENQUANTO**, **FAÇA.ATÉ** e **FAÇAPARA** 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 **FAÇAPARA**, **ENQUANTO** e **REPITA** na forma sequencial e encadeada. Assim sendo, selecione a opção "**Editar...**" menu "**Arquivo**", acione botão "**OK**", acione as teclas de atalho "**<Ctrl> + <A>**", tecle "**<Del>**" e entre o seguinte código:

#### APRENDA RODAFLOR

```
FAÇAPARA [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

Ao término, feche o programa **Editor** com "**Arquivo/Guardar e Sair**" ou acione as teclas "**<Ctrl> + <D>**". Em seguida execute as instruções **TAT** e "**RODAFLOR**" e veja a imagem apresentada semelhante à figura 4.26.

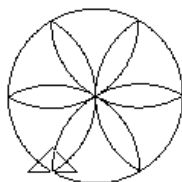


Figura 4.26 - Apresentação de imagem gerada com repetições sequenciais e encadeadas.

O comando **FAÇAPARA** 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**.

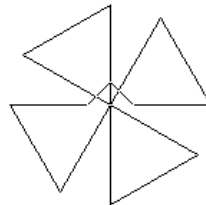
O comando **REPITA** usa para identificar o valor da contagem o apoio do comando **CONTEVEZES**. Com o comando **FAÇAPARA** é necessário fazer a definição de uma variável para a contagem e definir seus valores de início, fim e incremento quando diferente de "1". No entanto, o comando **FAÇAPARA** apresenta 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:**

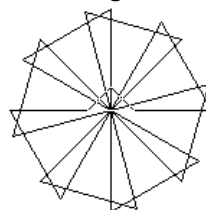
O ambiente "FMSLogo" executa os procedimentos desenvolvidos de maneira rápida. Apesar dessa rapidez ser importante no processamento dos dados elaborados por computadores, ela pode atrapalhar um pouco a percepção de como os desenhos são efetivamente feitos. Desta forma, pode-se pedir para que o ambiente 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 **FLORTRIG** desenhado a partir do procedimento **TRIANGEX** com giro a direita de modo que tenha a seguinte aparência.

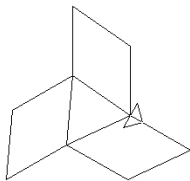


4. Criar procedimento chamado **VENTITRIG** desenhado a partir do procedimento **TRIANGEX** com giro a esquerda de modo que tenha a seguinte aparência.

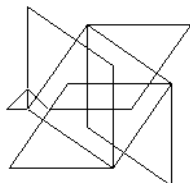


5. Criar procedimento chamado **LOSANGO** que desenhe imagem de mesmo nome com tamanho fornecido por parâmetro a partir de giro a direita.

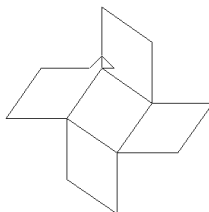
6. A partir do procedimento **LOSANGO** crie procedimento chamado **FLORLOSA1** de modo que seja apresentada a figura a seguir com tamanho "80".



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.

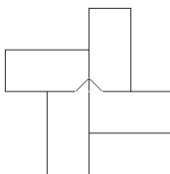
REPITA 12 [REPITA 3 [PF 50 PD 120] PD 30]

---

---

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**.

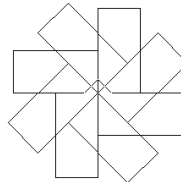


12. Criar procedimento chamado **TRIANGPR** que desenhe um triângulo equilátero com tamanho definido por parâmetro com comando **FAÇAPARA** contando de "0" a "2" no sentido a frente com giro a direita.

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 **FAÇAPARA**. Conte de "1" a "4".
15. Criar procedimento chamado **QUADRADO3** que desenhe um quadrado com lado definido por parâmetro a partir do uso da primitiva **FAÇA.ATÉ**. Conte de "1" a "4".
16. 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".



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



18. 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

```

---



---

## ANOTAÇÕES

[illegible]

## 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 possui 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
ARCCOS	Calcula o arco cosseno de número, em graus.
ARCCOSRAD	Calcula o arco cosseno de número, em radianos.
ARCSEN	Calcula o arco seno de número, em graus.
ARCSENRAD	Calcula o arco seno de número, em radianos.
ARCTAN	Calcula o arco tangente em graus, de número.
ARCTANRAD	Calcula o arco tangente em radianos.
ARREDONDE	Arredonda o número para o inteiro mais próximo.
COS	Calcula o cosseno de um ângulo medido em graus.
COSRAD	Calcula o cosseno de um ângulo medido em radianos.
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.
SENRAD	Calcula o seno do ângulo medido em radianos.
TAN	Calcula a tangente do ângulo medido em graus.
TANRAD	Calcula a tangente do ângulo medido em radianos.
EXP	Calcula o exponencial de um número

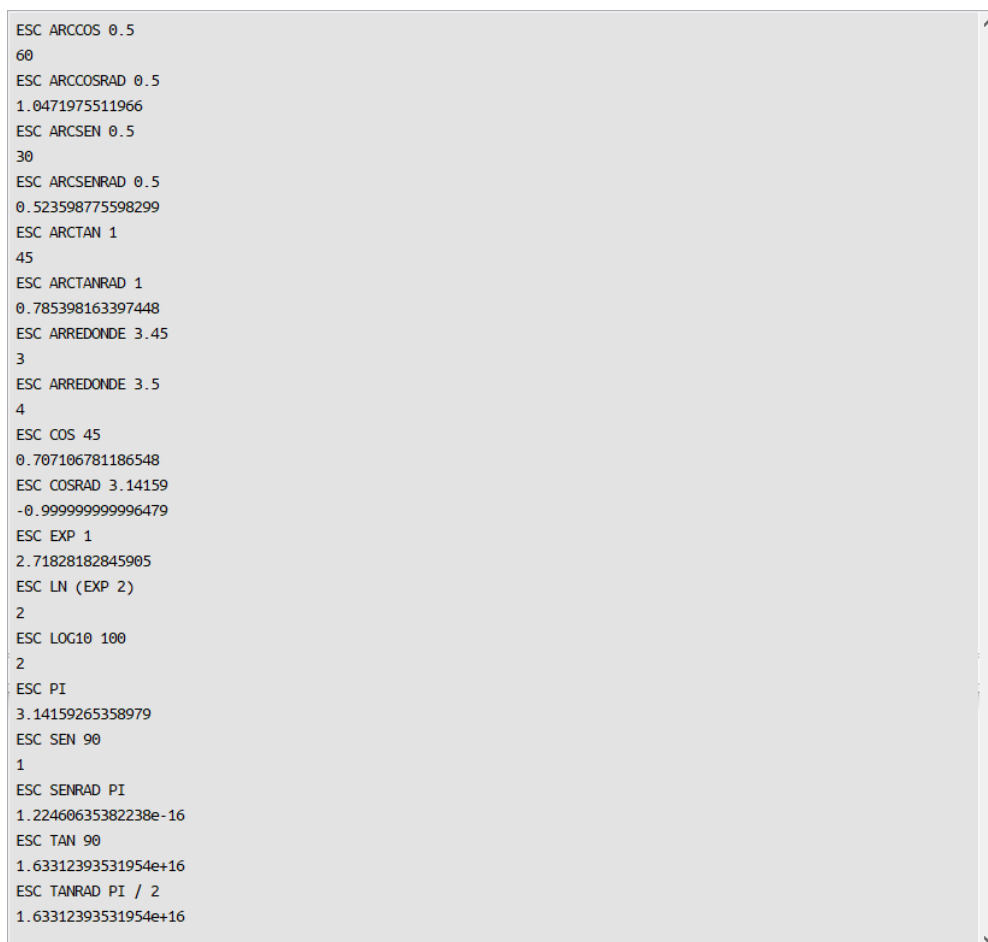
Tabela 5.1 - Funções matemáticas e suas ações de ângulos.

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

```
ESC ARCCOS 0.5
ESC ARCCOSRAD 0.5
ESC ARCSEN 0.5
ESC ARCSENRAD 0.5
ESC ARCTAN 1
ESC ARCTANRAD 1
ESC ARREDONDE 3.45
ESC ARREDONDE 3.5
```

```
ESC COS 45
ESC COSRAD 3.14159
ESC EXP 1
ESC LN (EXP 2)
ESC LOG10 100
ESC PI
ESC SEN 90
ESC SENRAD PI
ESC TAN 90
ESC TANRAD PI / 2
```

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



```
ESC ARCCOS 0.5
60
ESC ARCCOSRAD 0.5
1.0471975511966
ESC ARCSIN 0.5
30
ESC ARCSINRAD 0.5
0.523598775598299
ESC ARCTAN 1
45
ESC ARCTANRAD 1
0.785398163397448
ESC ARREDONDE 3.45
3
ESC ARREDONDE 3.5
4
ESC COS 45
0.707106781186548
ESC COSRAD 3.14159
-0.999999999996479
ESC EXP 1
2.71828182845905
ESC LN (EXP 2)
2
ESC LOG10 100
2
ESC PI
3.14159265358979
ESC SEN 90
1
ESC SENRAD PI
1.22460635382238e-16
ESC TAN 90
1.63312393531954e+16
ESC TANRAD PI / 2
1.63312393531954e+16
```

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 das primitivas **LEIACAR**, **LEIALISTA** ou **LEIAPALAVRA**; já 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 \* CONTEVEZES)) \* 150 (SEN(3 \* CONTEVEZES)) \* 100]

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

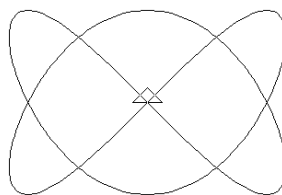


Figura 5.2 - Imagem obtida a partir do uso da função "SEN".

## 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 encontradas na linguagem Logo: **ÉANTERIOR?**, **ÉMAIOR?**, **ÉMEMBRO?**, **ÉMENOR?**, **ÉNÚMERO?** e **SÃOIGUAIS?**, indicadas na tabela 5.2.

FUNÇÃO	OPERAÇÃO
ÉANTERIOR?	Retorna VERD se parâmetro 1 vem à frente de parâmetro 2.
ÉMAIOR?	Retorna VERD se parâmetro 1 é maior que parâmetro 2.
ÉMEMBRO?	Retorna VERD se parâmetro 1 for membro de parâmetro 2.
ÉMENOR?	Retorna VERD se parâmetro 1 é menor que parâmetro 2.
ÉNÚMERO?	Retorna VERD se valor for um número.
SÃOIGUAIS?	Retorna VERD se parâmetros forem iguais.

Tabela 5.2 - Funções lógicas (algumas).

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

```

ESC ÉANTERIOR? "ABC "ABD
ESC ÉANTERIOR? "XYZ "ABC
ESC ÉMAIOR? 5 4
ESC ÉMAIOR? 4 5
ESC ÉMEMBRO? 1 [1 2 3 4 5]
ESC ÉMEMBRO? 7 [1 2 3 4 5]
ESC ÉMEMBRO? 7 "678
ESC ÉMEMBRO? 7 [678]
ESC ÉMENOR? 4 5
ESC ÉMENOR? 5 4
ESC ÉNÚMERO? 3
ESC ÉNÚMERO? [1 2 3]
ESC SÃOIGUAIS? 2 2
ESC SÃOIGUAIS? 1 2
ESC SÃOIGUAIS? [1 2 3] [1 2 3]
ESC SÃOIGUAIS? [1 2 3] [3 2 1]

```

Veja os resultados gerados a partir das funções comentadas na figura 5.3.

```

ESC ÉANTERIOR? "ABC "ABD
verd
ESC ÉANTERIOR? "XYZ "ABC
falso
ESC ÉMAIOR? 5 4
verd
ESC ÉMAIOR? 4 5
falso
ESC ÉMEMBRO? 1 [1 2 3 4 5]
verd
ESC ÉMEMBRO? 7 [1 2 3 4 5]
falso
ESC ÉMEMBRO? 7 "678
verd
ESC ÉMEMBRO? 7 [678]
falso
ESC ÉMENOR? 4 5
verd
ESC ÉMENOR? 5 4
falso
ESC ÉNÚMERO? 3
verd
ESC ÉNÚMERO? [1 2 3]
falso
ESC SÃOIGUAIS? 2 2
verd
ESC SÃOIGUAIS? 1 2
falso
ESC SÃOIGUAIS? [1 2 3] [1 2 3]
verd
ESC 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.

As operações de processamento lógico são efetivadas a partir do uso dos operadores relacionais, operadores lógicos e funções especializadas.

### 5.3 - Algumas funções complementares

O conjunto de funções na linguagem Logo é extenso e torna-se inadequado exemplificar todas elas em um livro, uma vez que o ambiente "FMSLogo" possui boa documentação em português de todos os recursos disponíveis na linguagem. Veja algumas funções complementares que não foram anteriormente qualificadas, sendo: **ÉNOME?** (função lógica) **PALAVRA**, **PRIMEIRO**, **TECLA?** (função lógica), **ÚLTIMO** e **VALOR**. Observe a tabela 5.3.

FUNÇÃO	OPERAÇÃO
ÉNOME?	Retorna VERD se conteúdo for o nome de uma variável.
PALAVRA	Concatena palavras formando outra palavra.
PRIMEIRO	Retorna o primeiro elemento de uma série de valores.
TECLA?	Retorna VERD se certa tecla esperada é acionada.
ÚLTIMO	Retorna o último elemento de uma série de valores.
VALOR	Retorna o conteúdo que esteja associado a uma variável.

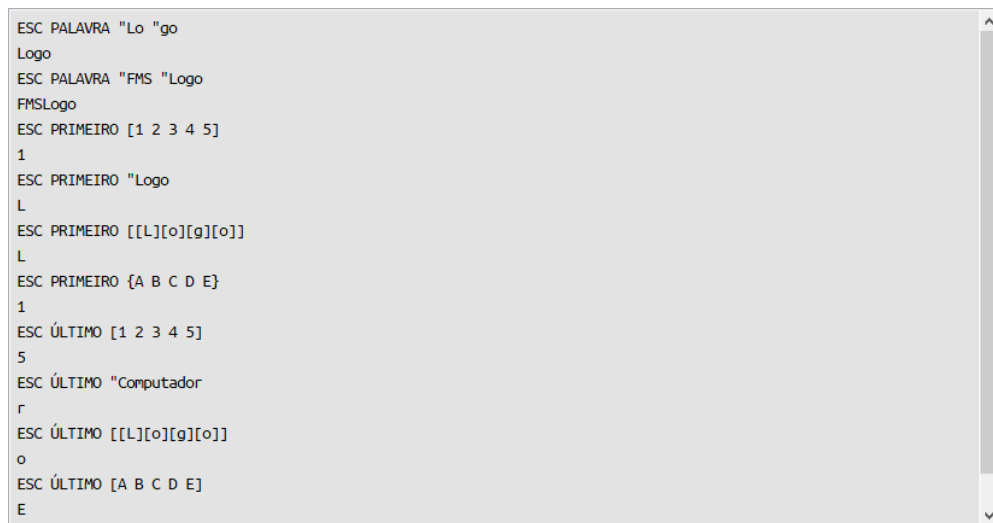
Tabela 5.3 - Funções complementares (algumas).

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.

```
ESC PALAVRA "Lo "go
ESC PALAVRA "FMS "Logo
ESC PRIMEIRO [1 2 3 4 5]
ESC PRIMEIRO "Logo
ESC PRIMEIRO [[L][o][g][o]]
ESC PRIMEIRO {A B C D E}
ESC ÚLTIMO [1 2 3 4 5]
ESC ÚLTIMO "Computador
ESC ÚLTIMO [[L][o][g][o]]
ESC ÚLTIMO [A B C D E]
```

Veja os resultados gerados a partir das funções comentadas na figura 5.4.



```
ESC PALAVRA "Lo "go
Logo
ESC PALAVRA "FMS "Logo
FMSLogo
ESC PRIMEIRO [1 2 3 4 5]
1
ESC PRIMEIRO "Logo
L
ESC PRIMEIRO [[L][o][g][o]]
L
ESC PRIMEIRO {A B C D E}
1
ESC ÚLTIMO [1 2 3 4 5]
5
ESC ÚLTIMO "Computador
r
ESC ÚLTIMO [[L][o][g][o]]
o
ESC ÚLTIMO [A B C D E]
E
```

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

As funções **ÉNOME?**, **TECLA?** e **VALOR** para serem demonstradas necessitam de mais passos que as demais funções. Por esta razão estão sendo demonstradas em separado.

A função **VALOR** 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:

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

Observe um detalhe que pode passar despercebido. 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 **VALOR**. Veja as próximas instruções:

```
ESC :VENCEDOR
ESC VALOR :VENCEDOR
ESC VALOR "VENCEDOR
```

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

```

ATRIBUA "VENCEDOR "COMPUTADOR
ATRIBUA "COMPUTADOR [100 PONTOS]
ESC :VENCEDOR
COMPUTADOR
ESC VALOR :VENCEDOR
100 PONTOS
ESC VALOR "VENCEDOR
COMPUTADOR

```

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

Observe na sequência o procedimento "INC" que acrescenta o valor "+ 1" a uma variável indicada e faz uso da função **ÉNOME?**. Assim, selecione a opção "Editar..." menu "Arquivo", acione botão "OK", acione as teclas de atalho "<Ctrl> + <A>", tecle "<Del>" e entre o seguinte código:

```

APRENDA INC :X
  SE NÃO ÉNOME? :X [PARE]
  SE ÉNÚMERO? VALOR :X [ATRIBUA :X 1 + VALOR :X]
FIM

```

Ao término, feche o programa **Editor** com "Arquivo/Guardar e Sair" ou acione as teclas "<Ctrl> + <D>". Execute na sequência as instruções a seguir e veja o resultado indicado na figura 5.6.

```

ATRIBUA "TOTAL 1
ESC :TOTAL
INC "TOTAL
ESC :TOTAL
INC "TOTAL
ESC :TOTAL

```

```

ATRIBUA "TOTAL 1
ESC :TOTAL
1
INC "TOTAL
ESC :TOTAL
2
INC "TOTAL
ESC :TOTAL
3

```

Figura 5.6 - Apresentação de resultados com função "VALOR" em procedimento.

Observe a evolução dos valores de "1" a "3" nos passos indicados. Note que o procedimento "INC" recebe um parâmetro (seu conteúdo) representado pela variável **X**.

Veja que a primeira instrução do procedimento "INC" é verificar se o conteúdo passado para o argumento **X** é de fato uma variável. Isso é detectado pela função **ÉNOME?** com auxílio do operador lógico de negação **NÃO**, ou seja, não sendo o nome fornecido uma variável **PARE**. A função **ÉNOME?** retorna o valor **VERD** se a condição for satisfeita ou **FALSO** caso contrário.

A segunda instrução do procedimento "INC" valida se o **VALOR** da variável **X** informado é de fato um número (**ÉNÚMERO?**), sendo efetue a atribuição (**ATRIBUA**) de "+ 1" sobre o conteúdo do **VALOR** armazenado na memória a partir da variável informada em **X**.

Uma função lógica interessante é **TECLA?** que pode ser usada na identificação do acionamento de teclas do teclado.

Observe a seguir o procedimento "**DIREÇÃO**" adaptado de ATARI (1983, p. 128) que risca a mão livre um desenho na área de ação do modo gráfico. Para tanto, selecione a opção "**Editar...**" menu "**Arquivo**", acione botão "**OK**", acione as teclas de atalho "**<Ctrl> + <A>**", tecle "**<Del>**" e entre o seguinte código:

APRENDA DIRECAO

SE TECLA? [REALIZE RC]

DIRECAO

FIM

APRENDA REALIZE :DIR

SE MAIÚSCULAS :DIR = "C [TAT] ; Limpa área de trabalho.

SE MAIÚSCULAS :DIR = "D [PD 10] ; Direção para direita.

SE MAIÚSCULAS :DIR = "E [PE 10] ; Direção para esquerda.

SE MAIÚSCULAS :DIR = "L [UL] ; Ativa o lápis.

SE MAIÚSCULAS :DIR = "N [UN] ; Desativa o lápis.

SE MAIÚSCULAS :DIR = "F [PF 2] ; Anda para frente.

SE MAIÚSCULAS :DIR = "T [PT 2] ; Anda para trás.

FIM

Ao término, feche o programa **Editor** com "**Arquivo/Guardar e Sair**" ou acione as teclas "**<Ctrl> + <D>**". Execute na sequência o comando **DIRECAO** e na caixa de diálogo apresentada informe os comandos "**C, D, E, L, N, F e T**" para gerenciar o desenvolvimento de um desenho a mão livre. Para encerrar a execução acione o botão **Cancelar**. Veja um exemplo de resultado na figura 5.7.

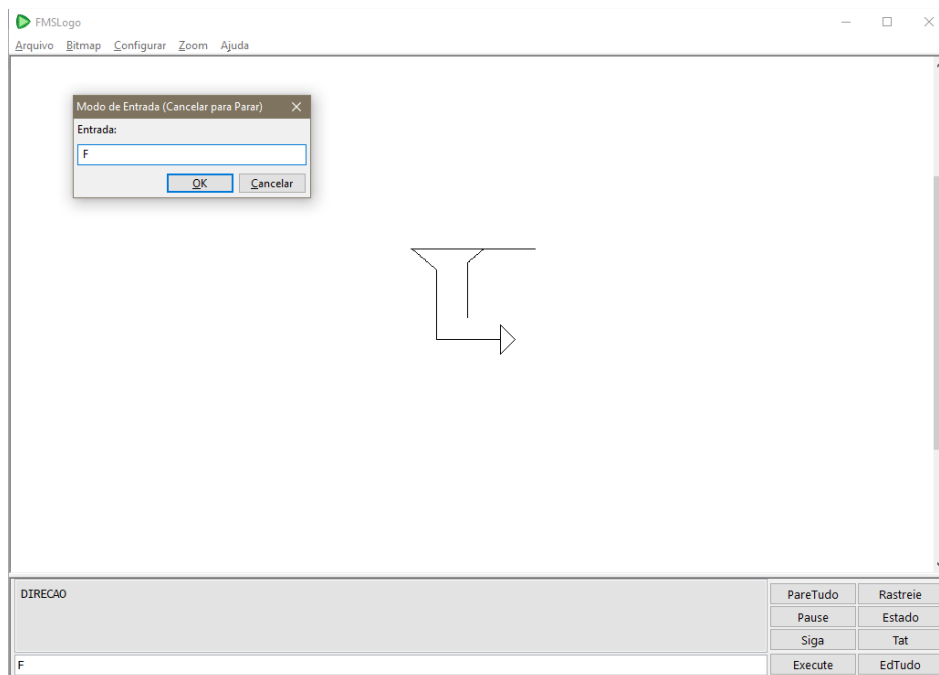


Figura 5.7 - Apresentação de desenho a mão livre.

Observe no procedimento "**REALIZE**" além da definição das teclas de acionamento e das ações a elas associadas o uso do símbolo ponto e vírgula com a descrição da ação de cada instrução do procedimento. O símbolo dois pontos é usado dentro do código quando há o desejo de declarar linhas de comentários dentro do código com o objetivo de definir uma linha de documentação interna do próprio código.

## 5.4 - Alguns eventos de controle

A linguagem Logo é operada dentro de um ambiente de desenvolvimento que fornece alguns recursos para auxiliar o gerenciamento do espaço de memória utilizado como por exemplo remover da memória uma variável definida ou retirar um procedimento que não se deseje mais. Para realizar essas e outras tarefas tem-se, por exemplo, as primitivas indicadas na tabela 5.4.

PRIMITIVA	OPERAÇÃO
ELTUDO	Limpa toda a memória da área de trabalho.
ELIMINE (EL)	Remove procedimento específico da memória.
ELN	Remove variável específica da memória.
ELNS	Remove todas as variáveis da memória.
ELPS	Remove todos os procedimentos da memória.

Tabela 5.4 - Primitivas de gerenciamento (algumas).

A primitiva **ELTUDO** não necessita de parâmetros e quando colocada em uso apresenta a caixa de diálogo **Warning** (figura 5.8) perguntando se há a certeza absoluta de execução da remoção dos componentes existentes na memória. O botão "Sim" confirma a remoção e os botões "Não" e "Cancelar" permitem interromper a ação.

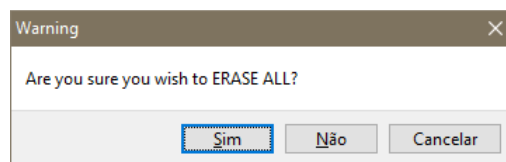


Figura 5.8 - Caixa de diálogo "Warning".

As primitivas: **ELIMINE**, **ELN**, **ELNS** e **ELPS** para serem usadas necessitam da definição de parâmetros que identifiquem os nomes das variáveis ou procedimentos. Lembre-se de antes de indicar o nome do componente a ser removido da memória usar o símbolo de aspa inglesa.

## 5.5 - 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 5.9).



Figura 5.9 - Criança desenhando com Logo (SOLOMON, 2020, p. 39).



Figura 5.10 - 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 5.10).

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. Para tanto, selecione a opção "**Editar...**" menu "**Arquivo**", acione botão "**OK**", acione as teclas de atalho "**<Ctrl> + <A>**", teclle "**<Del>**" e entre o seguinte código:

#### APRENDA AREACIRC

```
ESC [Informe a medida do raio de uma circunferência:]
```

```
ATR "RAIO LEIAP
```

```
ATR "AREA PI * (POTÊNCIA :RAIO 2)
```

```
ESC (FRASE "Área " = FORMATONÚMERO :AREA 0 2)
```

```
FIM
```

Ao término, feche o programa **Editor** com "**Arquivo/Guardar e Sair**" ou acione as teclas "**<Ctrl> + <D>**". Execute o procedimento "**AREACIRC**" e informe para um primeiro teste o valor "**5**" e veja a apresentação do resultado da área com "**78.54**" como mostra a figura 5.11.

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.

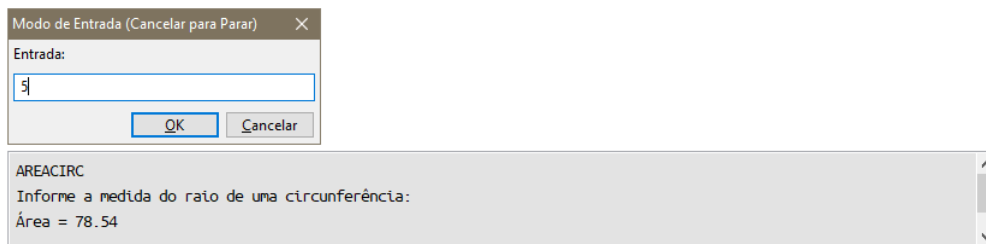


Figura 5.11 - Entrada e saída do procedimento "AREACIRC".

O segundo exemplo é fundamentado a partir de um procedimento chamado "**DECISAO**" que recebe a entrada separada de dois valores numéricos representados pelas variáveis "**A**" e "**B**", soma-os armazenando seu resultado na variável "**X**" e mostra o resultado da soma na variável "**X**" caso este seja maior que "**10**". Para somas obtidas menores que "**10**" não deve ser realizada nenhuma operação. Desta forma, selecione a opção "**Editar...**" menu "**Arquivo**", acione botão "**OK**", acione as teclas de atalho "<Ctrl> + <A>", tecle "<Del>" e entre o seguinte código:

#### APRENDA DECISAO

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

Ao término, feche o programa **Editor** com "**Arquivo/Guardar e Sair**" ou acione as teclas "<Ctrl> + <D>". Execute o procedimento "**DECISAO**" e informe separadamente os valores "**5**" e "**6**" e veja a apresentação do resultado "**11**", depois execute novamente o procedimento e informe os valores "**5**" e "**5**" e observe que nada é apresentado. As figuras 5.12 e 5.13 mostram os resultados desses testes.

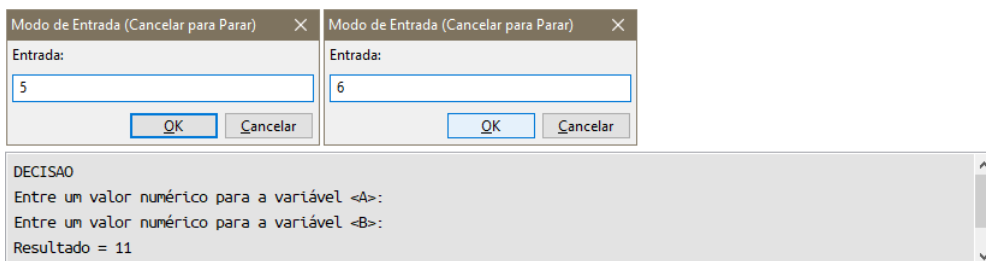


Figura 5.12 - Testes de execução do procedimento "DECISAO".

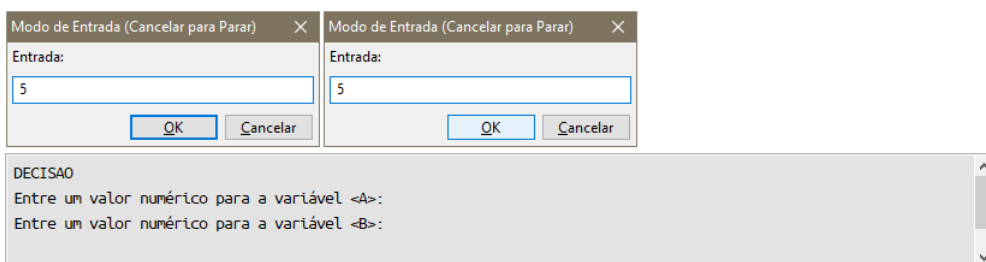


Figura 5.13 - Testes de execução do procedimento "DECISAO".

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*.

No terceiro exemplo considere um procedimento chamado "**CALCULAQUAD**" que apresente apenas o resultado do quadrado de um valor numérico inteiro e positivo (neste caso, maior ou igual a zero) lido pelo teclado sem armazená-lo em memória. Neste código considere validar a entrada recusando valores que não sejam numéricos e que não sejam positivos, além de considerar apenas a parte inteira do valor caso um valor real seja fornecido. Assim sendo, selecione a opção "**Editar...**" menu "**Arquivo**", acione o botão "**OK**", acione as teclas de atalho "<Ctrl> + <A>", tecle "<Del>" e entre o seguinte código:

#### APRENDA CALCULAQUAD

```
FAÇA.ATÉ [
  ESC [Entre um valor numérico:]
  ATR "VLR INTEIRO LEIAP
  ][:VLR >= 0]
  ESC (FR "Resultado "= POTÊNCIA :VLR 2)
FIM
```

Ao término, feche o programa **Editor** com "**Arquivo/Guardar e Sair**" ou acione as teclas "<Ctrl> + <D>". Execute o procedimento e teste as entradas com valores positivos, negativos e reais. Note o controle de negação da entrada de valores negativos com o uso da primitiva **FAÇA.ATÉ**.

Como quinto e último exemplo de programação sem figuras considere um procedimento chamado "**FATORIAL**" que apresente o resultado da fatorial de um número. Este procedimento deve apresentar ao final para seu usuário a opção de fazer ou não novos cálculos, mas esta opção somente pode aceitar as respostas **SIM** ou **NÃO** em formato maiúsculo (qualquer outra forma de entrada deve ser recusada). Para tanto, selecione a opção "**Editar...**" menu "**Arquivo**", acione botão "**OK**", acione as teclas de atalho "<Ctrl> + <A>", tecle "<Del>" e entre o seguinte código:

#### APRENDA FATORIAL

```
FAÇA.ATÉ [
  ATR "FAT 1
  FAÇA.ATÉ [
    ESC [Entre um valor numérico:]
    ATR "N INTEIRO LEIAP
    ][:N >= 0]
    FAÇAPARA [I 1 :N] [
      ATR "FAT :FAT * :I
    ]
    ESC (FR PALAVRA :N "! "= :FAT)
    FAÇA.ATÉ [
      ESC [Deseja continuar? Responda: SIM ou NÃO]
      ATR "RESP LEIAP
      ][OU MAIÚSCULAS :RESP = "SIM MAIÚSCULAS :RESP = "NÃO]
      ][MAIÚSCULAS :RESP <> "SIM]
FIM
```

Ao término, feche o programa **Editor** com "**Arquivo/Guardar e Sair**" ou acione as teclas "<Ctrl> + <D>". Execute o procedimento e faça seus testes.

## 5.6 - 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).

<b>FUNÇÃO</b>	<b>OPERAÇÃO</b>
ASCII	Retorna código ASCII do caractere informado.
CAR	Retorna o caractere correspondente ao código ASCII.
CONTEITEM	Conta caracteres de uma palavra ou palavras de uma frase.
ÉANTERIOR?	Retorna VERD se primeiro conteúdo vem a frente do segundo.
ÉSUBSEQÜÊNCIA?	Retorna VERD se primeiro conteúdo faz parte do segundo.
INVERTE	Inverte uma sequência de caracteres.
ITEM	Mostra um elemento de uma coleção a partir da posição.
MINÚSCULAS	Mostra caracteres em formato minúsculo.
SEMPRIMEIRO	Mostra todos os elementos, exceto o primeiro.
SEMPRIMEIROS	Mostra os elementos de listas, exceto os primeiros.
SEMÚLTIMO	Mostra todos os elementos, exceto o último.
SEQINT	Gera lista de números inteiros customizada de 1 em 1.
SEQRAC	Gera lista de números racionais quantificados.

Tabela 5.5 - Funções complementares (mais algumas).

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

ESC PRIMEIRO "ABCDE

A

ESC PRIMEIRO 54321

5

ESC ÚLTIMO "ABCDE

E

ESC ÚLTIMO 54321

1

ESC ASCII "A

65

ESC CAR 65

A

ESC ÉANTERIOR? "ABC "ABD

VERD

ESC ÉANTERIOR? "ABD "ABC

FALSO

ESC SEMPRIMEIRO 1.2345 ; "SEMPRIMEIRO" pode ser escrito como "SP".  
.2345

ESC SP 54321  
4321

ESC SEMPRIMEIROS [[5 4 3 2 1] [A B C B D E]] ; "SEMPRIMEIROS" = "SPS".

[4 3 2 1] [B C B D E]

ESC ÉSUBSEQÜÊNCIA? "AB "ABC  
VERD

ESC ÉSUBSEQÜÊNCIA? "AC "ABC  
FALSO

ESC INVERTE "ABCDE  
EDCBA

ESC MINÚSCULAS "ABCDE  
abcde

ESC SEMÚLTIMO "ABCDE ; "SEMÚLTIMO" = "SU".  
ABCD

ESC SU 54321  
5432

ESC ITEM 2 "ABCDE  
B

ESC ITEM 4 54321  
2

ESC PALAVRA "Linguagem "Logo  
LinguagemLogo

ESC FRASE "Linguagem "Logo  
Linguagem Logo

(ESC "Linguagem "Logo)  
Linguagem Logo

ESC (FRASE "Estudo "de "Linguagem "Logo)  
Estudo de Linguagem Logo

ESC (FRASE 1 2 3 4 5)  
1 2 3 4 5

ESC (PRODUTO 2 3 4)  
24

ESC (SOMA 1 2 3 4 5)  
15

ESC SEQINT 2 9  
2 3 4 5 6 7 8 9

ESC SEQINT 9 2  
9 8 7 6 5 4 3 2

ESC SEQRAC 2 9 8  
2 3 4 5 6 7 8 9

ESC SEQRAC 9 2 8  
9 8 7 6 5 4 3 2

ESC CONTEITEM [Estudo de Linguagem Logo]  
4

ESC CONTEITEM "ABCDE  
5

ESC CONTEITEM (FRASE "Estudo "de "Linguagem "Logo)  
4

ESC CONTEITEM (PALAVRA "Estudo "de "Linguagem "Logo)  
21

ESC CONTEITEM PALAVRA PRIMEIRO "ABCD SEMPRIMEIRO 1234567890  
10

Um detalhe que deve ter sido percebido, ao longo do que foi até aqui apresentado, é a capacidade da linguagem operar com tipos de dados diferenciados, como números e textos, além do processamento aritmético e lógico. Um importante detalhe relacionados a manipulação de dados é o agrupamento de elementos em conjuntos contextualizados na forma de listas (elementos entre parênteses) e vetores (elementos entre chaves). Os dados de um tipo de agrupamento podem ser convertidos aos dados de outro tipo de agrupamento com o uso específicos das funções **VETORPARALISTA** e **LISTAPARAVETOR**. Observe exemplos de conversão.

MOSTRE VETORPARALISTA {1 2 3 4 5}  
[1 2 3 4 5]

```
ESC LISTAPARAVETOR [1 2 3 4 5]
{1 2 3 4 5}
```

No ambiente "FMSLogo" os dados de uma lista são apresentados sem a indicação dos parêntese quando em uso a primitiva **ESCREVA (ESC)**.

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). Para tanto, selecione a opção "**Editar...**" menu "**Arquivo**", acione botão "**OK**", acione as teclas de atalho "<Ctrl> + <A>", teclle "<Del>" e entre o seguinte código:

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

Ao término, feche o programa **Editor** com "**Arquivo/Guardar e Sair**" ou acione as teclas "<Ctrl> + <D>". 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, selecione a opção "**Editar...**" menu "**Arquivo**", acione botão "**OK**", acione as teclas de atalho "<Ctrl> + <A>", teclle "<Del>" e entre o seguinte código:

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

Ao término, feche o programa **Editor** com "**Arquivo/Guardar e Sair**" ou acione as teclas "<Ctrl> + <D>". 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 "**ESC :CONTEUDO**" antes e após a instrução "**TEXTOTRIANG2 SEMPRIMEIRO :CONTEUDO**".

## 5.7 - 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 subprocedimentos 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 subprocedimento 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, selecione a opção "**Editar...**" menu "**Arquivo**", acione botão "**OK**", acione as teclas de atalho "**<Ctrl> + <A>**", tecle "**<Del>**" e entre o seguinte código:

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

Ao término, feche o programa **Editor** com "**Arquivo/Guardar e Sair**" ou acione as teclas "**<Ctrl> + <D>**". 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:

```
ESC SOMA :CONTADORG 1
```

Veja na figura 5.14 a apresentação dos resultados de uso de variável global. Note que a instrução "**ESC SOMA :CONTADORG 1**" soma mais "**1**" ao valor da variável "**CONTADORG**" tornando seu resultado "**3**" por ser "**CONTADORG**" uma variável global.

Observe o procedimento chamado "**VARLOCAL1**" com a definição de uma variável local com acesso apenas interno ao procedimento. Desta forma, selecione a opção "**Editar...**" menu "**Arquivo**", acione botão "**OK**", acione as teclas de atalho "**<Ctrl> + <A>**", tecle "**<Del>**" e entre o seguinte código:

```
APRENDA VARLOCAL1
  LOCAL "CONTADORL
  ATR "CONTADORL 1
  ESC :CONTADORL
  ATR "CONTADORL :CONTADORL + 1
  ESC :CONTADORL
FIM
```

Ao término, feche o programa **Editor** com "**Arquivo/Guardar e Sair**" ou acione as teclas "**<Ctrl> + <D>**". 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:

**ESC SOMA :CONTADORL 1**

Veja na figura 5.15 a apresentação dos resultados de uso de variável global. Note que a instrução **"ESC 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.

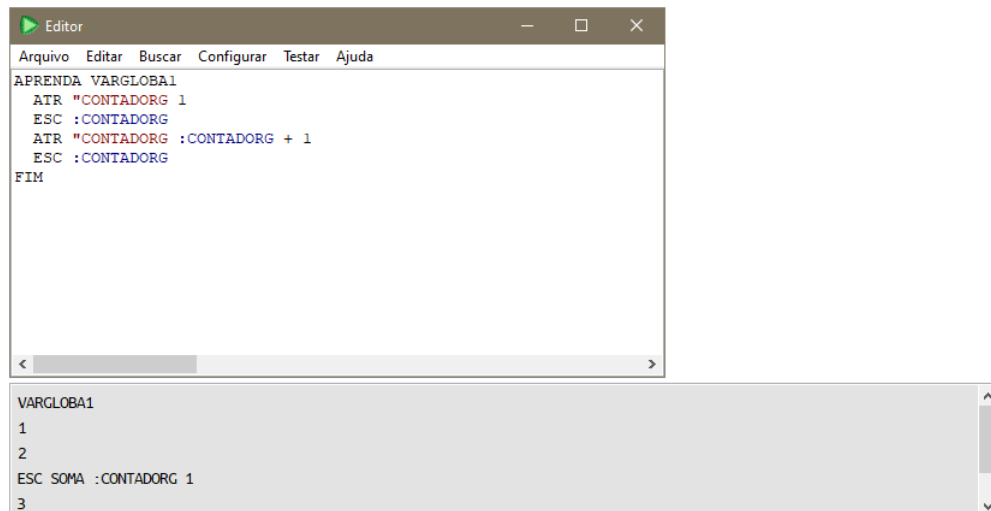


Figura 5.14 - Resultado no uso de variável global.

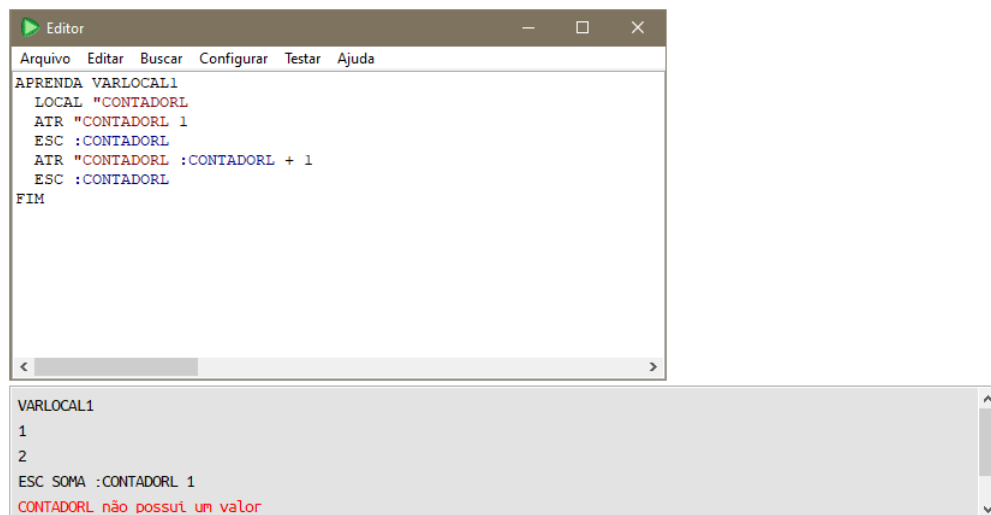


Figura 5.15 - Resultado no uso de variável local.

Observe que a primitiva **LOCAL** permite que seja definida uma variável com estado de visibilidade local, mas após esta 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 subprocedimento (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, selecione a opção **"Editar..."** menu **"Arquivo"**,

acione botão "OK", acione as teclas de atalho "<Ctrl> + <A>", tecle "<Del>" e entre o seguinte código:

```
APRENDA VARGLOBA2
```

```
  ATR "CONTADORGX 1
```

```
  ESC :CONTADORGX
```

```
  SUBVARGLOBA2
```

```
FIM
```

```
APRENDA SUBVARGLOBA2
```

```
  ATR "CONTADORGX :CONTADORGX + 1
```

```
  ESC :CONTADORGX
```

```
FIM
```

Ao término, feche o programa **Editor** com "Arquivo/Guardar e Sair" ou acione as teclas "<Ctrl> + <D>". 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:

```
ESC SOMA :CONTADORGX 1
```

Veja a apresentação dos resultados "1", "2" e "3" a partir da execução do procedimento principal com chamada de subprocedimento (sub-rotina) utilizando-se variável global, indicado na figura 5.16.

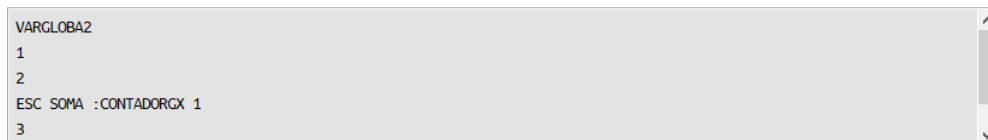


Figura 5.16 - Resultado no uso de variável global com procedimento e sub-rotina.

Na sequência veja o procedimento chamado "VARLOCAL2" com a definição local de variável e uso de sub-rotina para a ação de contagem. Assim sendo, selecione a opção "Editar..." menu "Arquivo", acione botão "OK", acione as teclas de atalho "<Ctrl> + <A>", tecle "<Del>" e entre o seguinte código:

```
APRENDA VARLOCAL2
```

```
  LOCAL "CONTADORLX
```

```
  ATR "CONTADORLX 1
```

```
  ESC :CONTADORLX
```

```
  SUBVARLOCAL2
```

```
FIM
```

```
APRENDA SUBVARLOCAL2
```

```
  ATR "CONTADORLX :CONTADORLX + 1
```

```
  ESC :CONTADORLX
```

```
FIM
```

Ao término, feche o programa **Editor** com "Arquivo/Guardar e Sair" ou acione as teclas "<Ctrl> + <D>". Em seguida execute a instrução de chamada do procedimento:

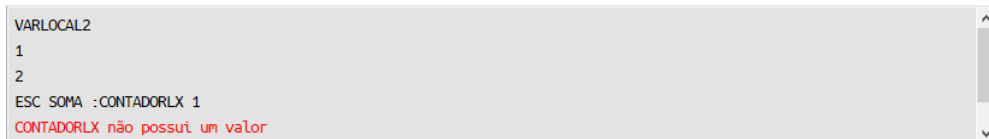


## VARLOCAL2

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

ESC SOMA :CONTADORLX 1

Veja a apresentação dos resultados "1", "2" e "CONTADORLX não possui um valor" a partir da execução do procedimento principal com chamada de sub procedimento (sub-rotina) utilizando-se variável local, indicado na figura 5.17.



```
VARLOCAL2
1
2
ESC SOMA :CONTADORLX 1
CONTADORLX não possui um valor
```

Figura 5.16 - Resultado no uso de variável local com procedimento e sub-rotina.

### 5.8 - 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 "**FAÇA.ATÉ**".
13. Criar procedimento chamado **CAP0513** que apresente a soma dos cem primeiros números naturais "**(1 + 2 + 3 + ... + 98 + 99 + 100)**" utilizando-se a primitiva "**FAÇAPARA**".
14. Criar procedimento chamado **CAP0514** que apresente o somatório dos *valores pares* existentes na faixa de "1" até "500". Use a primitiva "**REPITA**".
15. Criar procedimento chamado **CAP0515** que apresente o somatório dos *valores pares* existentes na faixa de "1" até "500". Use a primitiva "**ENQUANTO**".
16. Criar procedimento chamado **CAP0516** que apresente o somatório dos *valores pares* existentes na faixa de "1" até "500". Use a primitiva "**FAÇA.ATÉ**".
17. Criar procedimento chamado **CAP0517** que apresente o somatório dos *valores pares* existentes na faixa de "1" até "500". Use a primitiva "**FAÇAPARA**".
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 "**REPITA**".
19. Criar procedimento chamado **CAP0519** que mostre todos os valores numéricos divisíveis por "4" *menores* que "20" iniciando a contagem em "1". Use a primitiva "**ENQUANTO**".
20. Criar procedimento chamado **CAP0520** que mostre todos os valores numéricos divisíveis por "4" *menores* que "20" iniciando a contagem em "1". Use a primitiva "**FAÇA.ATÉ**".
21. Criar procedimento chamado **CAP0521** que mostre todos os valores numéricos divisíveis por "4" *menores* que "20" iniciando a contagem em "1". Use a primitiva "**FAÇAPARA**".

22. Criar procedimento chamado **CAP0522** que apresente os valores numéricos inteiros compreendidos na faixa de "0" e "4". Use a primitiva **"REPITA"**.
23. Criar procedimento chamado **CAP0523** que apresente os valores numéricos inteiros compreendidos na faixa de "0" e "4". Use a primitiva **"ENQUANTO"**.
24. Criar procedimento chamado **CAP0524** que apresente os valores numéricos inteiros compreendidos na faixa de "0" e "4". Use a primitiva **"FAÇA.ATÉ"**.
25. Criar procedimento chamado **CAP0525** que apresente os valores numéricos inteiros compreendidos na faixa de "0" e "4". Use a primitiva **"FAÇAPARA"**.
26. Criar procedimento chamado **CAP0526** que apresente os valores numéricos inteiros compreendidos na faixa de "0" até "15" de "3" em "3". Use a primitiva **"REPITA"**.
27. Criar procedimento chamado **CAP0527** que apresente os valores numéricos inteiros compreendidos na faixa de "0" até "15" de "3" em "3". Use a primitiva **"ENQUANTO"**.
28. Criar procedimento chamado **CAP0528** que apresente os valores numéricos inteiros compreendidos na faixa de "0" até "15" de "3" em "3". Use a primitiva **"FAÇA.ATÉ"**.
29. Criar procedimento chamado **CAP0529** que apresente os valores numéricos inteiros compreendidos na faixa de "0" até "15" de "3" em "3". Use a primitiva **"FAÇAPARA"**.

## ANOTAÇÕES

[illegible]

## Apêndice A - Algumas cores

Do grande conjunto de cores a ser gerada dentro do ambiente "FMSLogo" pode-se destacar o conjunto básico indicado em sua documentação.

COR (EM PORTUGUÊS)	CÓDIGO RGB	COR (EM INGLÊS)
ÁGUAMARINHA	[127 255 212]	AQUAMARINE
AGUAMARINHAMÉDIO	[102 205 170]	MEDIUMAQUAMARINE
ALARANJADO	[255 69 0]	ORANGERED
AMARELO	[255 255 0]	YELLOW
AMARELOCLARO	[255 255 224]	LIGHTYELLOW
AMARELODOURADOCLARO	[250 250 210]	LIGHTGOLDENRODYELLOW
AMARELOESVERDEADO	[173 255 47]	GREENYELLOW
AMEIXA	[221 160 221]	PLUM
AMÊNDOA	[255 235 205]	BLANCHEDALMOND
ANIL	[75 0 130]	INDIGO
AZUL	[0 0 255]	BLUE
AZULAÇO	[70 130 180]	STEELBLUE
AZULAÇOCLARO	[176 196 222]	LIGHTSTEELBLUE
AZULALICE	[240 248 255]	ALICEBLUE
AZULCADETE	[95 158 160]	CADETBLUE
AZULCELESTE	[240 255 255]	AZURE
AZULCELESTECLARO	[135 206 250]	LIGHTSKYBLUE
AZULCELESTEPROFUNDO	[0 191 255]	DEEPSKYBLUE
AZULCÉU	[135 206 235]	SKYBLUE
AZULCLARO	[173 216 230]	LIGHTBLUE
AZULESCURO	[0 0 139]	DARKBLUE
AZULFLORDEMILHO	[100 149 237]	CORNFLOWERBLUE
AZULFURTIVO	[30 144 255]	DODGERBLUE
AZULMÉDIO	[0 0 205]	MEDIUMBLUE
AZULMEIANOITE	[25 25 112]	MIDNIGHTBLUE
AZULPARDO	[106 90 205]	SLATEBLUE
AZULPARDOESCURO	[72 61 139]	DARKSLATEBLUE
AZULPARDOMÉDIO	[123 104 238]	MEDIUMSLATEBLUE
AZULPÓLVORA	[176 224 230]	POWDERBLUE
AZULREAL	[65 105 225]	ROYALBLUE
AZULVIOLETA	[138 43 226]	BLUEVIOLET
BEGE	[245 245 220]	BEIGE
BRANCO	[255 255 255]	WHITE

COR (EM PORTUGUÊS)	CÓDIGO RGB	COR (EM INGLÊS)
BRANCOANTIGO	[250 235 215]	ANTIQUEWHITE
BRANCOFLORAL	[255 250 240]	FLORALWHITE
BRANCOLIGEIRO	[248 248 255]	GHOSTWHITE
BRANCONAVAJO	[255 222 173]	NAVAJOWHITE
BRONZEADO	[210 180 140]	TAN
CAQUI	[240 230 140]	KHAKI
CAQUIESCURO	[189 183 107]	DARKKHAKI
CARDO	[216 191 216]	THISTLE
CARMIM	[220 20 60]	CRIMSON
CHOCOLATE	[210 105 30]	CHOCOLATE
CIANO (OU ÁGUA)	[0 255 255]	CYAN (OU AQUA)
CIANOCLARO	[224 255 255]	LIGHTCYAN
CIANOESCURO	[0 139 139]	DARKCYAN
CINZA	[128 128 128]	GREY
CINZACLARO	[211 211 211]	LIGHTGREY
CINZACLARO	[211 211 211]	LIGHTGRAY
CINZAESCURO	[169 169 169]	DARKGRAY (OU DARKGREY)
CINZAFOSCO	[105 105 105]	DIMGREY (OU DIMGREY)
CINZAMÉDIO	[220 220 220]	GAINSBORO
CINZAPARDO	[112 128 144]	SLATEGRAY (OU SLATEGREY)
CINZAPARDOCLARO	[119 136 153]	LIGHTSLATEGRAY (OU LIGHTSLATEGREY)
CINZAPARDOESCURO	[47 79 79]	DARKSLATEGRAY
CINZASUAVE	[105 105 105]	DIMGREY
CONCHA	[255 245 238]	SEASHELL
CORAL	[255 127 80]	CORAL
CORALCLARO	[240 128 128]	LIGHTCORAL
CREMEDEMARISCO	[255 228 196]	BISQUE
CREMEDEMENTA	[245 255 250]	MINTCREAM
DOURADO	[218 165 32]	GOLDENROD
DOURADOPÁLIDO	[238 232 170]	PALEGOLDENROD
DOURADOESCURO	[184 134 11]	DARKGOLDENROD
FUMAÇABRANCA	[245 245 245]	WHITESMOKE
LARANJA	[255 165 0]	ORANGE
LARANJAESCURO	[255 140 0]	DARKORANGE
LAVANDA	[230 230 250]	LAVENDER
LIMA	[0 255 0]	LIME

COR (EM PORTUGUÊS)	CÓDIGO RGB	COR (EM INGLÊS)
LINHO	[250 240 230]	LINEN
MADEIRA	[222 184 135]	BURLYWOOD
MAGENTA (OU FÚCSIA)	[255 0 255]	MAGENTA (OU FUCHSIA)
MAGENTAESCURO	[139 0 139]	DARKMAGENTA
MANÁ	[240 255 240]	HONEYDEW
MARFIM	[255 255 240]	IVORY
MARINHO	[0 0 128]	NAVY
MARROM	[128 0 0]	MAROON
MARROMAREIA	[244 164 96]	SANDYBROWN
MARROMCLARO	[165 42 42]	BROWN
MARROMROSADO	[188 143 143]	ROSYBROWN
MARROMTELHA	[139 69 19]	SADDLEBROWN
MILHO	[255 248 220]	CORNSILK
MOCASSIM	[255 228 181]	MOCCASIN
NEVE	[255 250 250]	SNOW
OLIVA	[128 128 0]	OLIVE
OLIVAESCURO	[85 107 47]	DARKOLIVEGREEN
OLIVAPARDO	[107 142 35]	OLIVEDRAB
ORQUÍDEA	[218 112 214]	ORCHID
ORQUÍDEAESCURO	[153 50 204]	DARKORCHID
ORQUÍDEAMÉDIO	[186 85 211]	MEDIUMORCHID
OURO	[255 215 0]	GOLD
PAPAIA	[255 239 213]	PAPAYAWHIP
PERU	[205 133 63]	PERU
PÊSSEGO	[255 218 185]	PEACHPUFF
PRATA	[128 128 128]	GRAY
PRATA	[192 192 192]	SILVER
PRETO	[0 0 0]	BLACK
PÚRPURA	[128 0 128]	PURPLE
PÚRPURAMÉDIO	[147 112 216]	MEDIUMPURPLE
RENDAANTIGA	[253 245 230]	OLDLACE
ROSA	[255 192 203]	PINK
ROSABRUMOSO	[255 228 225]	MISTYROSE
ROSACLARO	[255 182 193]	LIGHTPINK
ROSALAVANDA	[255 240 245]	LAVENDERBLUSH
ROSAPROFUNDO	[255 20 147]	DEEPPINK
ROSAQUENTE	[255 105 180]	HOTPINK

COR (EM PORTUGUÊS)	CÓDIGO RGB	COR (EM INGLÊS)
SALMÃO	[250 128 114]	SALMON
SALMÃOCLARO	[255 160 122]	LIGHTSALMON
SALMÃOESCURO	[233 150 122]	DARKSALMON
SEDA	[255 250 205]	LEMONCHIFFON
SIENA	[160 82 45]	SIENNA
TOMATE	[255 99 71]	TOMATO
TRIGO	[245 222 179]	WHEAT
TURQUESA	[64 224 208]	TURQUOISE
TURQUESAPÁLIDO	[175 238 238]	PALETURQUOISE
TURQUESAESCURO	[0 206 209]	DARKTURQUOISE
TURQUESAMÉDIO	[72 209 204]	MEDIUMTURQUOISE
VERDE	[0 128 0]	GREEN
VERDEAMARELADO	[154 205 50]	YELLOWGREEN
VERDEAZULADO	[0 128 128]	TEAL
VERDECLARO	[144 238 144]	LIGHTGREEN
VERDEESCURO	[0 100 0]	DARKGREEN
VERDEFLORESTA	[34 139 34]	FORESTGREEN
VERDEGRAMA	[124 252 0]	LAWNGREEN
VERDELIMA	[50 205 50]	LIMEGREEN
VERDEMARESCURO	[143 188 143]	DARKSEAGREEN
VERDEMARINHO	[46 139 87]	SEAGREEN
VERDEMARINHOCLARO	[32 178 170]	LIGHTSEAGREEN
VERDEMARMÉDIO	[60 179 113]	MEDIUMSEAGREEN
VERDEPÁLIDO	[152 251 152]	PALEGREEN
VERDEPARIS	[127 255 0]	CHARTREUSE
VERDEPRIMAVERA	[0 255 127]	SPRINGGREEN
VERDEPRIMAVERAMÉDIO	[0 250 154]	MEDIUMSPRINGGREEN
VERMELHO	[255 0 0]	RED
VERMELHOESCURO	[139 0 0]	DARKRED
VERMELHOINDIANO	[205 92 92]	INDIANRED
VERMELHOTIJOLO	[178 34 34]	FIREBRICK
VIOLETA	[238 130 238]	VIOLET
VIOLETAPÁLIDO	[216 112 147]	PALEVIOLETRED
VIOLETAESCURO	[148 0 211]	DARKVIOLET
VIOLETAMÉDIO	[199 21 133]	MEDIUMVIOLETRED



## Apêndice B - Primitivas inglês/espanhol (algumas)

O ambiente **FMSLogo** instalado em português também aceita comandos e procedimento escritos em inglês, o que é interessante para fixação de palavras do idioma inglês. Neste sentido, apresenta-se neste apêndice apenas as primitivas e funções que foram usadas no livro e sua equivalência em inglês (apresenta-se também os termos em espanhol). Para maiores detalhes acesse a opção "Início" no menu "Ajuda" e escolha o tópico "**Comandos: Inglês para Português**" ou "**Comandos: Português para Inglês**". As abreviaturas das primitivas estão grafadas em vermelho e quando houver mais uma abreviatura a segunda estará grafada em azul.

PORTUGUÊS	INGLES	ESPAÑHOL
ABS	ABS	ABS
APRENDA	TO	PARA
ARCCOS	ARCCOS	ARCCOS
ARCCOSRAD	RADARCCOS	RADARCCOS
ARCSIN	ARCSIN	ARCSIN
ARCTAN	ARCTAN	ARCTAN
ARCTANRAD	RADARCTAN	RADARCTAN
ARREDONDE	ROUND	REDONDEA
ASCII	ASCII	ASCII
ATRIBUA	MAKE	HAZ
CAR	CHAR	CAR
CENTRO	HOME	CENTRO
CONTEITEM	COUNT	CUENTA
CONTEVEZES	REPCOUNT	CUENTAREPITE
COS	COS	COS
COSRAD	RADCOS	RADCOS
DIFERENÇA	DIFFERENCE	DIFERENCIA
E	AND	Y
ÉANTERIOR?	BEFOREP	ANTERIOR?
ELIMINE ( <b>EL</b> )	ERASE ( <b>ER</b> )	BORRA ( <b>BO</b> )
ELN	ERN	BOVAR
ELNS	ERNS	BNOMBRES
ELPS	ERPS	BPROCS
ELTUDO	ERALL	BTODD
ÉMAIOR?	GREATER	MAYOROIGUAL?
ÉMEMBRO?	MEMBERP	MIEMBRO?
ÉMENOR?	LESSP	MENOR?
ÉNOME?	NAMEP	VAR?
ENQUANTO	WHILE	MIENTRAS
ÉNÚMERO?	NUMBERP	NUMERO?
ESCREVA ( <b>ESC</b> )	PRINT ( <b>PR</b> )	ESCRIBE ( <b>ES</b> )

PORTUGUÊS	INGLES	ESPAÑHOL
ESPERE	WAIT	ESPERA
ÉSUBSEQUÊNCIA?	SUBSTRINGP	CONTIENE?
FAÇA.ATÉ	DO.UNTIL	HAZ.HASTA
FAÇAPARA	FOR	DESDE
FALSO	FALSE	FALSO
FORMATONÚMERO	FORM	FORMATONUMERO
FRASE (FR)	SENTENCE (SE)	FRASE (FR)
INTEIRO	INT	ENTERO
INVERTE	REVERSE	INVIERTE
ITEM	ITEM	ELEMENTO
LEIACARACTERE (LEIACAR)	READCHAR (RC)	LEECAR (LC)
LEIALISTA (LEIAL)	READLIST (RL)	LEELISTA (LL)
LEIAPALAVRA (LEIAP)	READWORD (RW)	LEEPALABRA (LP)
LIMPEJANELACOMANDOS (LJC)	CLEARTEXT (CT)	BORRATEXTO (BT)
LISTAPARAVETOR	LISTTOARRAY	LISTAAVECTOR
LN	LN	LN
LOCAL	LOCAL	LOCAL
LOG10	LOG10	LOG10
MAIÚSCULAS	UPPERCASE	MAYUSCULAS
MINÚSCULAS	LOWERCASE	MINUSCULAS
MOSTRE	SHOW	MUESTRA
MOSTRETAT (MT)	SHOWTURTLE (ST)	MUESTRATORTUGA (ST)
MUDECF	SETSC	PONCF
MUDECL	SETPC	PONCL
MUDECP	SETFC	PONCR
MUDEDIREÇÃO (MUDEDC)	SETHEADING (SETH)	PONRUMBO (PONR)
MUDEESPL	SETPENSIZE	PONGROSOR
MUDEPOS	SETPOS	PONPOS
MUDEX	SETX	PONX
MUDEXY	SETXY	PONXY
MUDEY	SETY	PONY
NÃO	NOT	NO
OCULTETAT (OT)	HIDETURTLE (HT)	OCULTATORTUGA (OT)
OU	OR	O
PADAESQUERDA (PE)	LEFT (LT)	GIRAIZQUIERDA (GI)
PALAVRA	WORD	PALABRA
PARADIREITA (PD)	RIGHT (RT)	GIRADERECHA (GD)
PARAFRENTE (PF)	FORWARD (FD)	AVANZA (AV)

PORTUGUÊS	INGLES	ESPAÑHOL
PARATRÁS (PT)	BACK (BK)	RETROCEDE (RE)
PARE	STOP	ALTO
PI	PI	PI
PINTE	FILL	RELLENA
POS	POS	POS
POTÊNCIA	POWER	POTENCIA
PRIMEIRO	FIRST	PRIMERO
PRODUTO	PRODUCT	PRODUCTO
QUOCIENTE	QUOTIENT	COCIENTE
RAIZQ	SQRT	RAIZCUADRADA (RC)
REPITA	REPEAT	REPITE
REPITASORTEIE	RERANDOM	REAZAR
RESTO	REMAINDER	RESTO
ROTULE	LABEL	ROTULO
SÃOIGUAIS?	EQUALP	IGUALES?
SE	IF	SI
SEMPRIMEIRO (SP)	BUTFIRST (BF)	MENOSPRIMERO (MPR/MP)
SEMPRIMEIROS (SPS)	BUTFIRSTS (BFS)	MENOSPRIMEROS (MPS)
SEMÚLTIMO (SU)	BUTLAST (BL)	MENOSULTIMO (MU/MUL)
SEN	SIN	SEN
SENRAD	RADSIN	RADSEN
SEQINT	ISEQ	SECENT
SEQRAC	RSEQ	SECRAC
SOM	SOUND	SUENA
SOMA	SUM	ADICION
SORTEIENÚMERO	RANDOM	AZAR
TAN	TAN	TAN
TANRAD	RADTAN	RADTAN
TARTARUGA (TAT)	CLEARSCREEN (CS)	BORRAPANTALLA (BP)
TECLA?	KEYP	TECLA?
ÚLTIMO	LAST	ULTIMO (UL)
USEBORRACHA (UB)	PENERASE (PE)	GOMA
USELÁPIS (UL)	PENDOWN (PD)	BAJALAPIZ (BL)
USENADA (UN)	PENUP (PU)	SUBELAPIZ (SL)
USERISCO (PPT)	PENPAINT (PPT)	LAPIZNORMAL (LNORMAL)
VALOR	THING	VALOR
VERD	TRUE	CIERTO
VETORPARALISTA	ARRAYTOLIST	VECTORALISTA

## ANOTAÇÕES

[illegible]

---

## Apêndice C - 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]

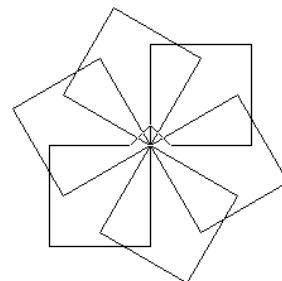
FIM

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.

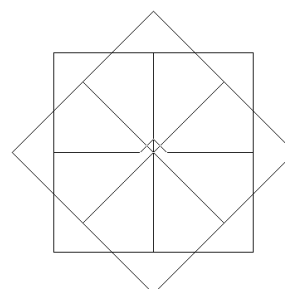
TAT

REPITA 6 [PD 60 FIGURAS 4 80]

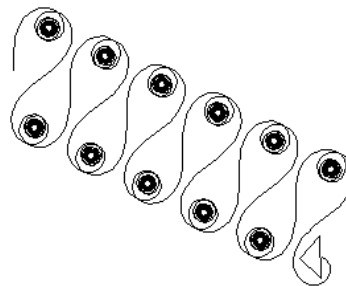


TAT

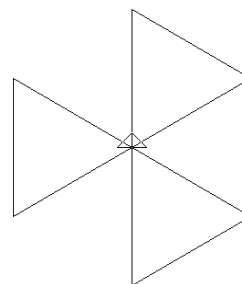
REPITA 8 [FIGURAS 4 120 PD 45]



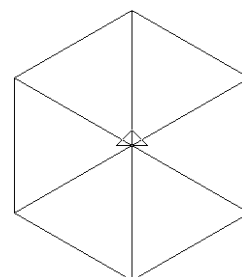
TAT  
FAÇAPARA [I 0 2000] [PF 5 PD (90 \* SEN :I)]



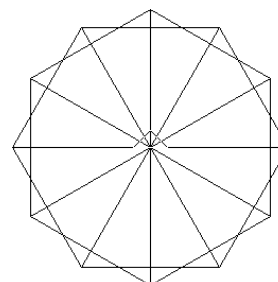
TAT  
REPITA 3 [FIGURAS 3 150 PD 120]



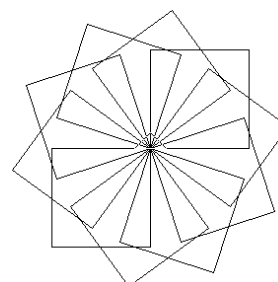
TAT  
REPITA 6 [FIGURAS 3 140 PD 60]



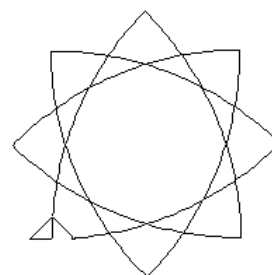
TAT  
REPITA 360 / 30 [FIGURAS 3 130 PD 30]



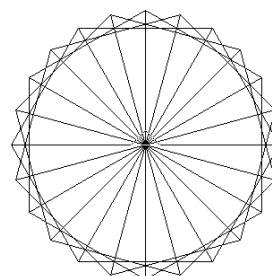
TAT  
REPITA 10 [REPITA 4 [PF 100 PD 90] PD 36]



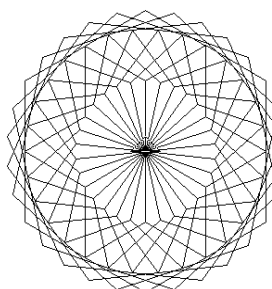
TAT  
 REPITA 8 [REPITA 45 [PF 4 PD 1] PD 90]



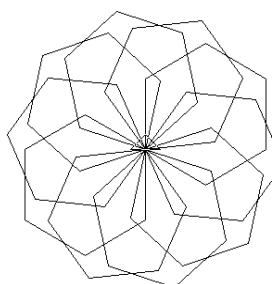
TAT  
 REPITA 24 [REPITA 3 [PF 150 PD 120] PD 15]



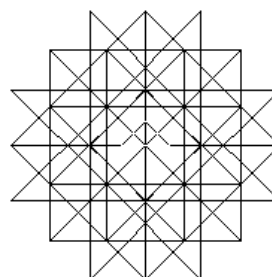
TAT  
 REPITA 30 [REPITA 6 [PF 75 PD 60] PD 12]



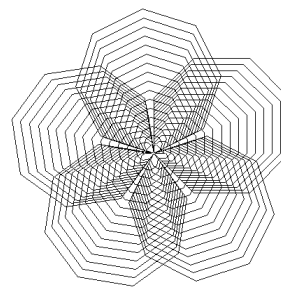
TAT  
 REPITA 10 [REPITA 6 [PF 75 PD 60] PD 36]



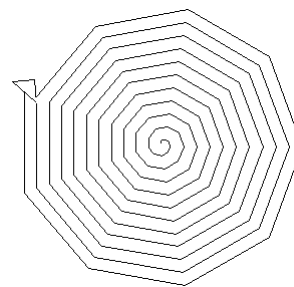
TAT  
 REPITA 8 [REPITA 8 [PE 135 PF 90] PE 45]



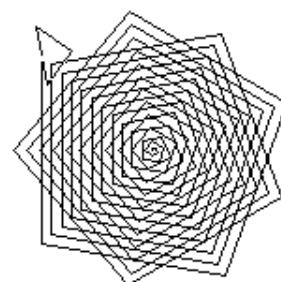
```
TAT
FAÇAPARA [I 10 80 5] [
  REPITA 5 [
    REPITA 8 [
      PF :I PD 45
    ]
    PD 72
  ]
]
```



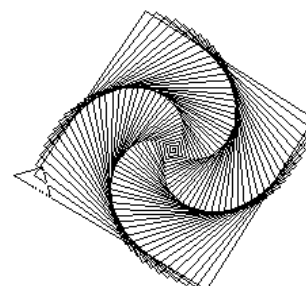
```
TAT
REPITA 100 [PF CONTEVEZES PD 40]
```



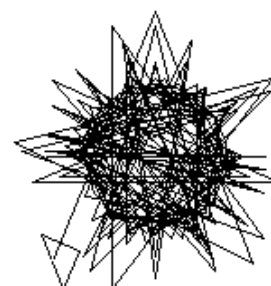
```
TAT
REPITA 100 [PF CONTEVEZES PD 80]
```



```
TAT
REPITA 150 [PF CONTEVEZES PD 89]
```

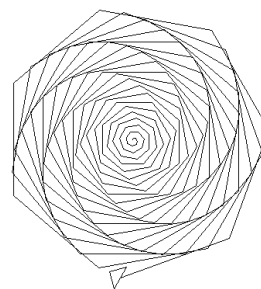


```
TAT
REPITA 150 [PF CONTEVEZES PD CONTEVEZES * 1.5]
```

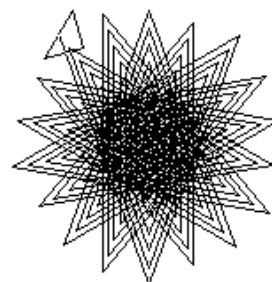




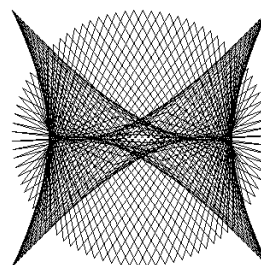
TAT  
 REPITA 150 [PF CONTEVEZES PD 50]



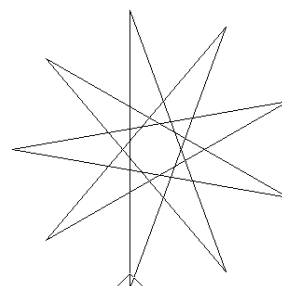
TAT  
 REPITA 150 [PF CONTEVEZES PD 500]



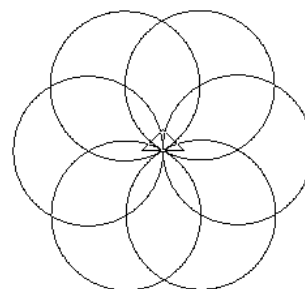
TAT  
 REPITA 360 [  
   MUDEXY (SEN( 89 \* CONTEVEZES)) \* 150  
   (SEN(179 \* CONTEVEZES)) \* 150  
 ]



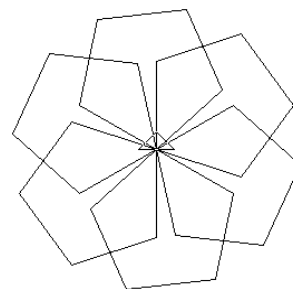
TAT  
 REPITA 9 [PF 300 PD 160]



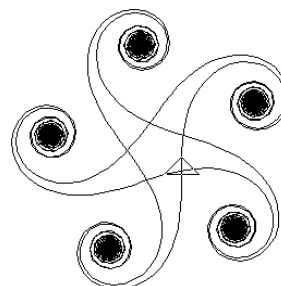
TAT  
 REPITA 6 [PD 60 FIGURAS 360 1]



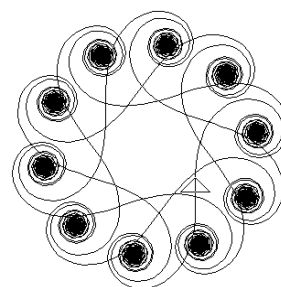
```
TAT
REPITA 6 [PD 60 FIGURAS 5 80]
```



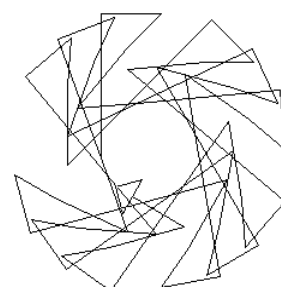
```
TAT
REPITA 1800 [PF 10 PD CONTEVEZES + .1]
```



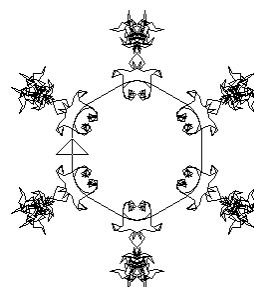
```
TAT
REPITA 3600 [PF 10 PD CONTEVEZES + .2]
```



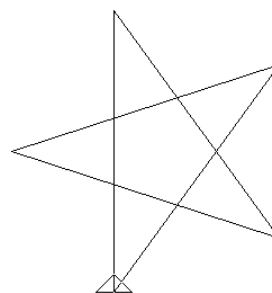
```
TAT
REPITA 12 [
  PF 120 PD 90 PF 50 PD 135 PF 40
  PE 185 PT 65 PD 45 PF 50
]
```



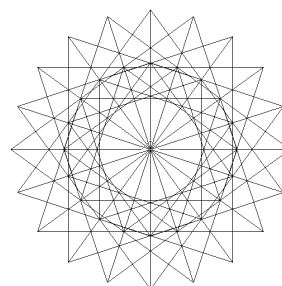
```
TAT
FAÇAPARA [I 0 1002] [
  PF 8 MUDEDC (360 * (POTÊNCIA :I 3) / 1002)
]
```



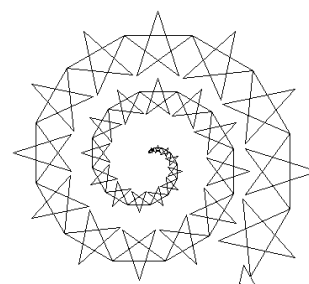
TAT  
REPITA 5 [PF 250 PD 144]



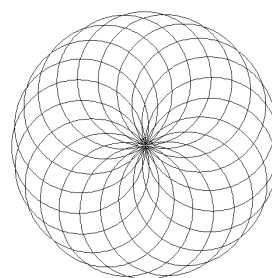
TAT  
REPITA 20 [REPITA 5 [PF 250 PD 144] PD 18]



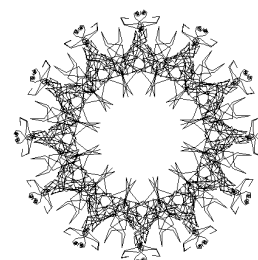
TAT  
FAÇAPARA [I 0 95 3] [  
  REPITA 5 [  
    PF :I PD 144  
  ]  
  PF :I PD 30  
]



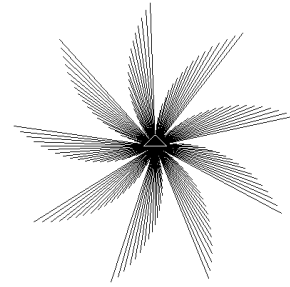
TAT  
REPITA 20 [REPITA 180 [PF 4 PD 2] PD 18]



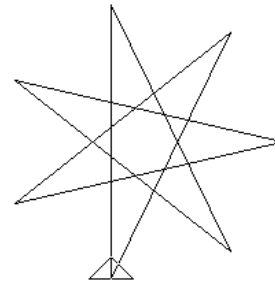
TAT  
FAÇAPARA [I 0 2200] [  
  PF (25 \* SEN :I) PD (POTÊNCIA :I 2)  
]



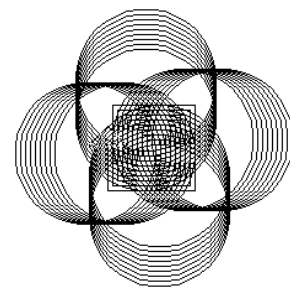
```
TAT
REPITA 9 [
  FAÇAPARA [I 10 200 10] [
    PF :I PT :I PD 2
  ]
]
```



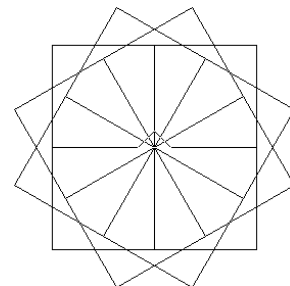
```
TAT
REPITA 7 [PF 100 PD 360 * 3 / 7]
```



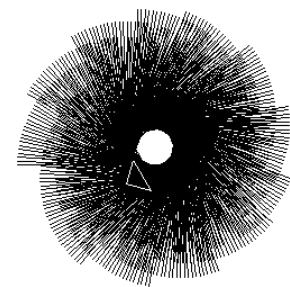
```
TAT
REPITA 36 [
  REPITA 36 [
    PF 10 PD 10
  ]
  PF CONTEVEZES PD 90 PF CONTEVEZES
]
```



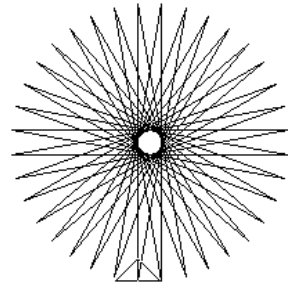
```
TAT
REPITA 12 [REPITA 4 [FIGURAS 4 100] PD 30]
```



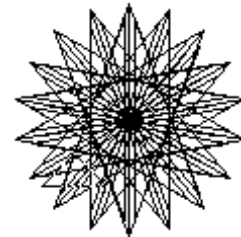
```
TAT
REPITA 12 [REPITA 55 [PF 100 PT 100 PD 2] PF 45]
```



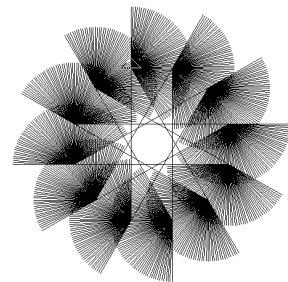
```
TAT
REPITA 54 [REPITA 8 [PF 200 PD 170]]
```



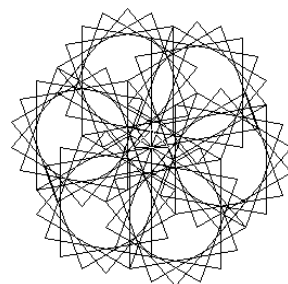
```
TAT
REPITA 18 [REPITA 5 [PD 40 PF 100 PD 120] PD 20]
```



```
TAT
REPITA 12 [
  REPITA 75 [
    PF 100 PT 100 PD 2
  ]
  PF 250
]
```



```
TAT
REPITA 12 [REPITA 360 [PF 100 PD 100] PD 60]
```



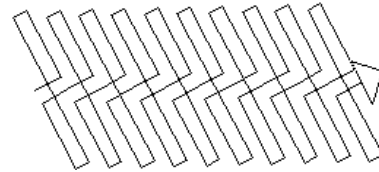
### INSTRUÇÕES A PARTIR DA DEFINIÇÃO DE PROCEDIMENTOS

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 ]
```

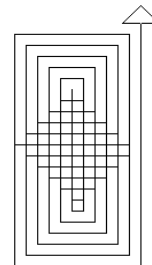
FIM

EXECUTE: TAT PADRAO

APRENDA CAMINHO

```
REPITA 22 [
  PD 90
  PF 110 - CONTEVEZES * 10
  PD 90
  PF CONTEVEZES * 10
]
```

FIM

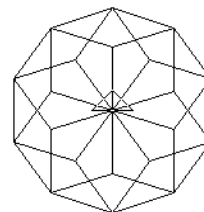


EXECUTE: TAT CAMINHO

APRENDA HEXAFLOR :PETALAS

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

FIM



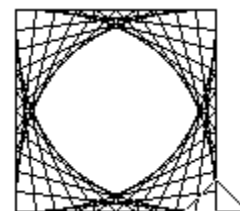
EXECUTE: TAT HEXAFLOR 10

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 CONTEVEZES * 10 100]
```

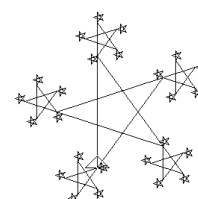
FIM

EXECUTE: TAT QUADART

APRENDA ESTRELANDO :TAMANHO :LIMITE

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

FIM

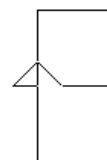


EXECUTE: TAT ESTRELANDO 150 10

APRENDA BANDEIRA

PF 50  
FIGURAS 4 50  
FIM

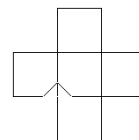
EXECUTE: TAT BANDEIRA



APRENDA CRUZ

REPITA 4 [BANDEIRA PD 90]  
FIM

EXECUTE: TAT CRUZ



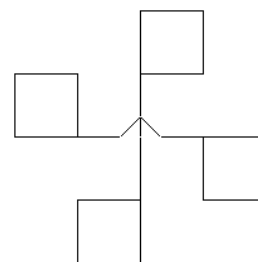
APRENDA VOLTABANDA

BANDEIRA  
PT 50  
FIM

APRENDA BANDEIRAS

REPITA 4 [VOLTABANDA PD 90]  
FIM

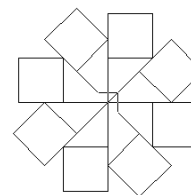
EXECUTE: TAT BANDEIRAS



APRENDA MUITASBANDEIRAS

BANDEIRAS  
PD 45  
BANDEIRAS  
FIM

EXECUTE: TAT MUITASBANDEIRAS



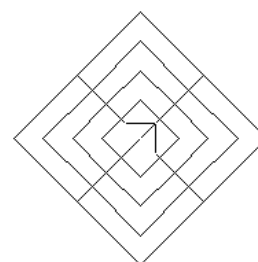
APRENDA QUADRADOS

FIGURAS 4 20  
FIGURAS 4 35  
FIGURAS 4 50  
FIGURAS 4 65  
FIM

APRENDA DIAMANTES

PD 45  
REPITA 4 [QUADRADOS PD 90]  
FIM

EXECUTE: TAT DIAMANTES



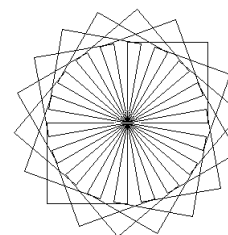
APRENDA QUADRADO

REPITA 4 [FIGURAS 4 100]  
FIM

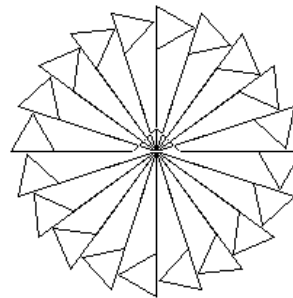
APRENDA FLORQUADRADA

REPITA 18 [QUADRADO PD 20]  
FIM

EXECUTE: TAT FLORQUADRADA



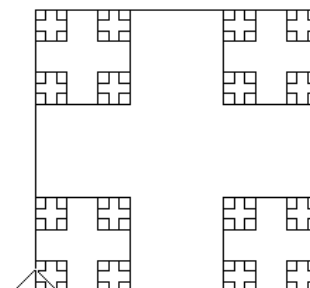
```
APRENDA BANDTRI
  PF 100 PD 120
  PF 30 PD 120
  PF 30 PD 120
  PT 70
FIM
```



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

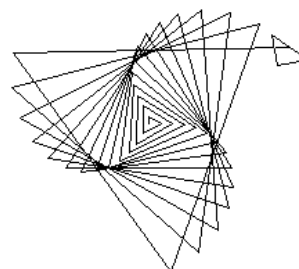
EXECUTE: TAT FLORBANDTRI

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



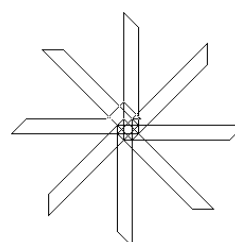
EXECUTE: TAT QUADRICULADO 200

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



EXECUTE: TAT TUNELTRI 5 120

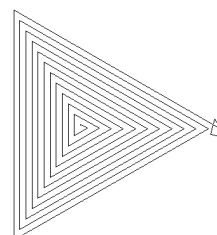
```
APRENDA PA
  REPITA 2 [PF 100 PD 135 PF 20 PD 45]
FIM
```



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

EXECUTE: HELICE

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



EXECUTE: TAT ESPIRALTRI 1

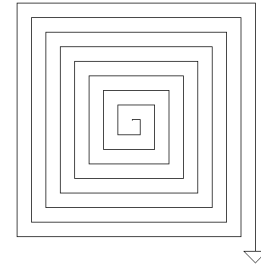


```

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: TAT ESPIRALQUAD 1

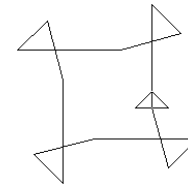


```

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

```

EXECUTE: TAT REPITA 4 [FORMA]

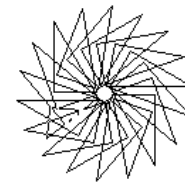


```

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

```

EXECUTE: TAT REPITA 20 [FORMATRI]

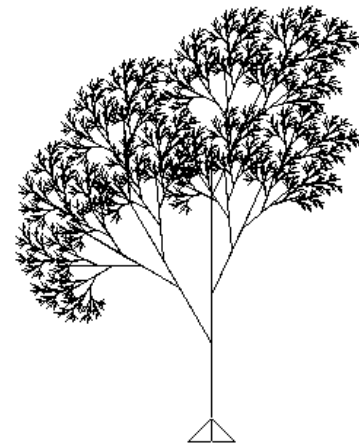


```

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: TAT ARVORE 200

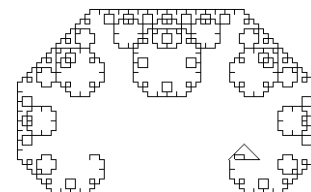


```

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

```

EXECUTE: TAT CURVAC 5 10



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

## ANOTAÇÕES

[illegible]

## Apêndice D - Efeitos sonoros

A linguagem Logo utilizada no ambiente "FMSLogo" possui muitos recursos e entre eles há a possibilidade de se fazer a criação de sons como efeitos sonoros e música e para esta finalidade há a primitiva **SOM** que pode ser utilizada a partir da definição de um par de valores com a instrução:

**SOM** [<freq> <duração>]

**SOM** [<freq1> <duração1> <freq2> <duração2> ... <freqN> <duraçãoN>]

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.

Para um rápido teste execute a instrução seguinte, observado em tom vermelho a frequência do som e em tom verde a duração do som:

**REPITA 6** [**SOM** [100 200 150 200 200 200 250 200]]

Para melhor aproveitar, criar e tocar músicas com a linguagem Logo é necessário que você tenha conhecimento sobre teoria musical e tenha certa intimidade com os sons das notas musicais. Neste livro faz-se apenas uma pequena introdução despretensiosa ao tema.

Considere que para escrever uma música é necessário combinar notas musicais. Isto significa, grosso modo, que pode-se considerar música um conjunto de sons e silêncios organizados de forma melódica e harmoniosa a partir da marcação de certo ritmo. Melodia é o que pode ser cantado, é a voz da música. Já a harmonia é a sobreposição das notas de modo que se tenha uma base para a melodia e o ritmo é a marcação do tempo de uma música. Veja que a linguagem Logo por meio da primitiva **SOM** aproxima-se da definição da harmonia com o uso da frequência e do ritmo com o uso da duração. Quanto a melodia? Esta fica por sua conta.

Uma nota musical é o elemento mínimo que compõe uma música podendo ser representada por letras ou por partitura. Veja o padrão de representação de notas musicais por letras

LETRA	NOTA MUSICAL
C	DÓ
D	RÉ
E	MI
F	FÁ
G	SOL
A	LÁ
B (ou H em alemão)	SI

Veja que são "7" as notas musicais. Essas notas são chamadas de *notas naturais*. No entanto, o conjunto de notas é composto por outras notas chamadas *notas acidentadas* que são notas que possuem a alteração na altura de seu tom em meio tom acima ou meio tom abaixo, formando um conjunto de mais "5" notas, perfazendo um total de "12" notas. As notas "**MI (E)**" e "**SI (B)**" não possuem acidentes. Veja essa ocorrência:

MEIO TOM ABAIXO ( <i>b</i> )	NOTA MUSICAL	MEIO TOM ACIMA (#)
-	C	C <sup>#</sup>
D <sup>b</sup>	D	D <sup>#</sup>
E <sup>b</sup>	E	-
-	F	F <sup>#</sup>
G <sup>b</sup>	G	G <sup>#</sup>
A <sup>b</sup>	A	A <sup>#</sup>
B <sup>b</sup>	B	-

Note que a nota *sustenido* de um tom é igual a nota *bemol* do tom de uma nota anterior e vice versa, sendo, de fato, são as mesmas notas: C<sup>#</sup> = D<sup>b</sup>, D<sup>#</sup> = E<sup>b</sup>, F<sup>#</sup> = G<sup>b</sup>, G<sup>#</sup> = A<sup>b</sup> e A<sup>#</sup> = B<sup>b</sup> com notações diferentes.

Considerando a notação de sustenidos veja as frequências das notas graves, médias e agudas obtidas a partir de um piano (<https://www.intmath.com/trigonometric-graphs/music.php>).

NOTAS	FREQUÊNCIA		
	AGUDO	MÉDIO	GRAVE
C	130.82	261.63	523.25
C <sup>#</sup>	138.59	277.18	554.37
D	146.83	293.66	587.33
D <sup>#</sup>	155.56	311.13	622.25
E	164.81	329.63	659.26
F	174.61	349.23	698.46
F <sup>#</sup>	185.00	369.99	739.99
G	196.00	392.00	783.99
G <sup>#</sup>	207.65	415.30	830.61
A	220.00	440.00	880.00
A <sup>#</sup>	233.08	466.16	932.33
B	246.94	493.88	987.77

A forma mais simples de fazer uso de notas musicais no ambiente **FMSLogo** é criar um procedimento para cada nota e em seguida usá-los dentro de outros procedimentos ou junto ao campo de entrada de comandos, dados e instruções. Assim sendo, considere os procedimentos seguintes para a geração das notas musicais a partir do uso dos valores inteiros das frequências apresentadas com duração de seiscentos milissegundos.

APRENDA CA

SOM [130 600] ; DÓ agudo  
FIM

APRENDA CM

SOM [261 600] ; DÓ médio  
FIM

## APRENDA CG

SOM [523 600] ; DÓ grave  
FIM

## APRENDA C#A

SOM [138 600] ; DÓ su#enido agudo  
FIM

## APRENDA C#

SOM [277 600] ; DÓ su#enido médio  
FIM

## APRENDA C#G

SOM [554 600] ; DÓ su#enido grave  
FIM

## APRENDA DA

SOM [146 600] ; RÉ agudo  
FIM

## APRENDA DM

SOM [293 600] ; RÉ médio  
FIM

## APRENDA DG

SOM [587 600] ; RÉ grave  
FIM

## APRENDA D#A

SOM [155 600] ; RÉ su#enido agudo  
FIM

## APRENDA D#

SOM [311 600] ; RÉ su#enido médio  
FIM

## APRENDA D#G

SOM [622 600] ; RÉ su#enido grave  
FIM

## APRENDA EA

SOM [164 600] ; MI agudo  
FIM

## APRENDA EM

SOM [329 600] ; MI médio  
FIM

APRENDA EG

SOM [659 600] ; MI grave

FIM

APRENDA FA

SOM [174 600] ; FA agudo

FIM

APRENDA FM

SOM [349 600] ; FA médio

FIM

APRENDA FG

SOM [698 600]

FIM

APRENDA F#A

SOM [185 600] ; FA su#enido agudo

FIM

APRENDA F#

SOM [369 600] ; FA su#enido médio

FIM

APRENDA F#G

SOM [739 600] ; FA su#enido grave

FIM

APRENDA GA

SOM [196 600] ; SOL agudo

FIM

APRENDA GM

SOM [392 600] ; SOL médio

FIM

APRENDA GG

SOM [783 600] ; SOL grave

FIM

APRENDA G#A

SOM [207 600] ; SOL su#enido agudo

FIM

APRENDA G#

SOM [415 600] ; SOL su#enido médio

FIM

## APRENDA G#G

SOM [830 600] ; SOL su#enido grave  
FIM

## APRENDA AA

SOM [220 600] ; LÁ agudo  
FIM

## APRENDA AM

SOM [440 600] ; LÁ médio  
FIM

## APRENDA AG

SOM [880 600] ; LÁ grave  
FIM

## APRENDA A#A

SOM [233 600] ; LÁ su#enido agudo  
FIM

## APRENDA A#

SOM [466 600] ; LÁ su#enido médio  
FIM

## APRENDA A#G

SOM [932 600] ; LÁ su#enido grave  
FIM

## APRENDA BA

SOM [246 600] ; SI agudo  
FIM

## APRENDA BM

SOM [493 600] ; SI médio  
FIM

## APRENDA BG

SOM [987 600] ; SI grave  
FIM

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". Assim sendo, selecione a opção "Editar..." menu "Arquivo", acione botão **OK**, acione as teclas de atalho "<Ctrl> + <A>", tecle "<Del>" e entre o seguinte código:

## APRENDA DOREMI

CM DM EM FM FM FM ESPERE 10  
DM EM DM EM EM EM ESPERE 10  
CM GM FM EM EM EM ESPERE 10  
CM DM EM FM FM FM  
FIM

Ao término, feche o programa **Editor** com "**Arquivo/Guardar e Sair**" ou acione as teclas "**<Ctrl> + <D>**". 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\_E**".

A qualidade sonora não se compara a uma orquestra, sendo muito básica, mas é suficiente para permitir o uso de efeitos sonoros em diversos procedimentos.

Fica como desafio a proposta de escrever um procedimento chamados **MARCHA** que toque a música de autoria desconhecida do folclore brasileiro "Marcha Soldado". Veja sua partitura básica.

The musical score is written in 2/4 time and consists of three staves. Each note is color-coded to match the lyrics below it.

**Staff 1:** Notes G, G, E, C, C, E, G, G, G, E, D, E, F, F, F, D, G, G, A. Lyrics: Mar - cha sol - da - do ca - be - ça de pa - pel, quem não mar-char di - rei - to vai

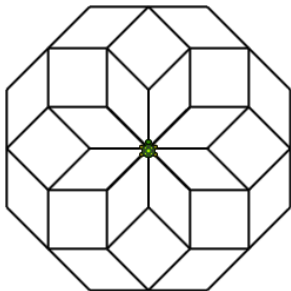
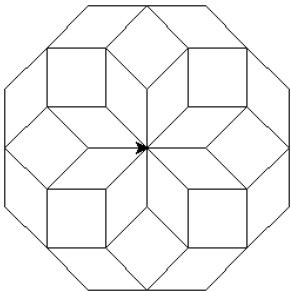
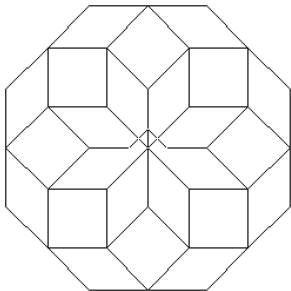
**Staff 2:** Notes G, F, E, D, C, C, E, G, G, E, C, C, E, G, G, G, E. Lyrics: pre - so no quar - tel, o quar - tel pe - gou fo - go, a po - li - cia deu si -

**Staff 3:** Notes D, E, F, F, F, D, G, G, A, G, F, E, D, C. Lyrics: nal: a - co - de! A-co-de! A - co - de a ban - dei - ra na - cio - nal.



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

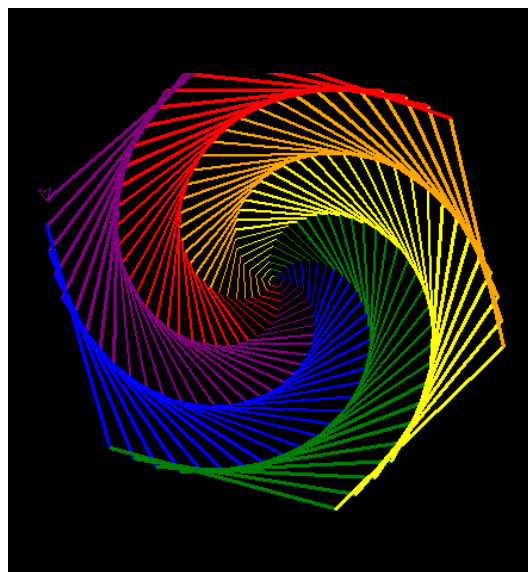
Muitos dos recursos de uso da linguagem Logo podem ser suportados em outras linguagens de programação. Por exemplo, a linguagem **"Small Basic"** desenvolvida pela empresa *Microsoft* (<https://smallbasic-publicwebsite.azurewebsites.net>) e **"Python"** produzida pela comunidade *Python* (<https://www.python.org>), 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 (FMSLogo)
<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>FAÇAPARA [I 1 8] [   FAÇAPARA [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():
    cores = [
        'red', 'purple', 'blue',
        'green', 'yellow', 'orange'
    ]
    turtle.bgcolor('black')
    for x in range(200):
        turtle.pencolor(cores[x % 6])
        turtle.width(x / 100 + 1)
        turtle.forward(x)
        turtle.left(59)
draw_spin()
input()
```



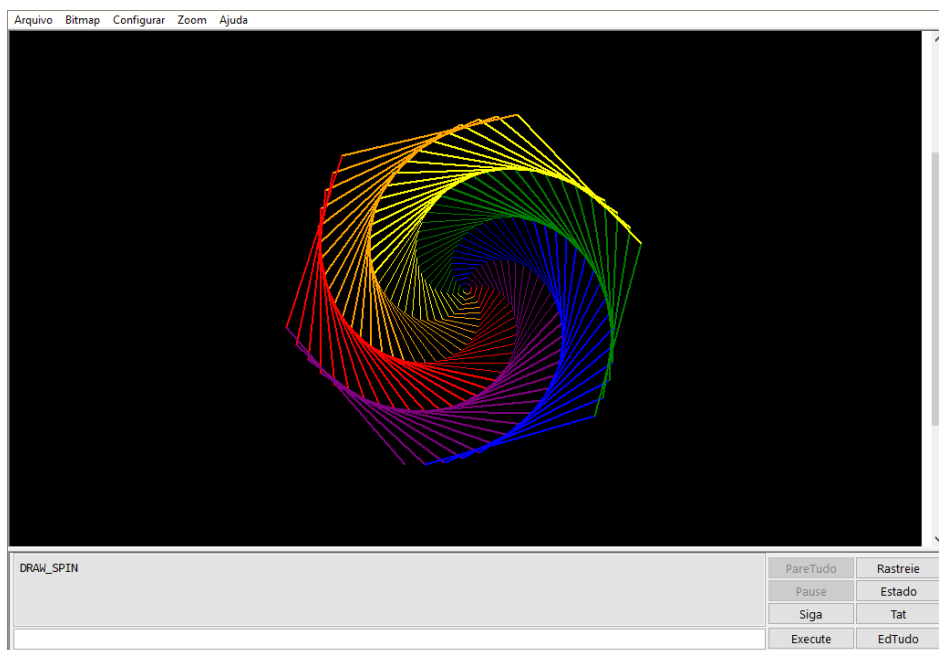
A partir do código **"Python"** foi realizada a transliteração e escrita de um código compatível em **"Logo"** que gerou a imagem usada na capa deste trabalho com o código:

## APRENDA DRAW\_SPIN

```

TAT MT
MUDECF [000 000 000] ; PRETO
FAÇAPARA [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
  MUDEESPL INTEIRO (:X / 100 + 1)
  PF :X
  PE 59
]
OT
MUDEESPL 0
FIM
  
```

O código do procedimento "DRAW\_SPIN" produzido em "Logo" ao ser executado mostra imagem semelhante ao código original prduzido 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";
- "TAT" e "MT" 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')";

- **"FAÇAPARA"** 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 **"FAÇAPARA"** 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 consoante 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;
- **"MUDEESPL INTEIRO (:X / 100 + 1)"** tem por finalidade a partir da primitiva **"MUDEESPL"** mudar a largura do traço do desenho em andamento estando está instrução em consonância com a instrução **"turtle.width(x/100 + 1)"**. O uso da função **"INTEIRO"** torna-se necessário pelo fato do valor a ser informado a primitiva **"MUDEESPL"** deve ser expresso como número inteiro;
- **"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 **"OT"** e **"MUDEESPL 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

[illegible]

---

## Apêndice F - Gabarito

---

### CAPÍTULO 3

---

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] Quadrado

REPITA 5 [PF 100 PE 72] Pentágono

REPITA 3 [PT 100 PD 120] Triângulo

REPITA 36 [PF 20 PD 10] Circunferência

2. Criar procedimento chamado **RETANGULO1** (sem acento) que desenhe um retângulo com lados de tamanhos **60** e **100** para frente com giro de graus 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
```

3. Criar procedimento chamado **RETANGULO2** (sem acento) que desenhe um retângulo com lados de tamanhos **60** e **100** para trás com giro de graus 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

PF 40

PD 72

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 para frente de **30** passos.

APRENDA DECAGONO

REPITA 10 [PF 30 PE 36]

FIM

6. Criar procedimento chamado **ICOSAGONO** (sem acento) que desenhe a figura de mesmo nome a partir do uso da primitiva **REPITA** com tamanho **35** movimentado para trás.

APRENDA ICOSAGONO

REPITA 20 [PT 35 PD 18]

FIM

ou

APRENDA ICOSAGONO

REPITA 20 [PT 35 PE 18]

FIM

## CAPÍTULO 4

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.

```

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

```

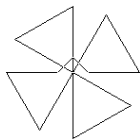
2. Crie um procedimento chamado **TRIANGEX**, que apresente um triângulo equilátero com lado de tamanho **70** para trás a partir de giros a esquerda.

```

APRENDA TRIANGEX
  REPITA 3 [
    PT 70 PE 120
  ]
FIM

```

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

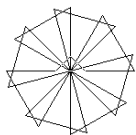


```

APRENDA FLORTRIG
  REPITA 4 [
    TRIANGEX PD 90
  ]
FIM

```

4. 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

```

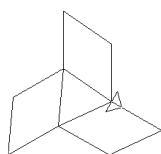
5. Criar procedimento chamado **LOSANGO** que desenhe imagem de mesmo nome com tamanho fornecido por parâmetro a partir de giro a direita.

```

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

```

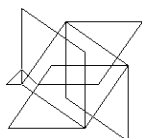
6. A partir do procedimento **LOSANGO** crie procedimento chamado **FLORLOSA1** de modo que seja apresentada a figura a seguir com tamanho **80**.



```

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

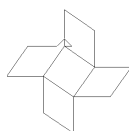
7. A partir do procedimento **LOSANGO** crie procedimento chamado **FLORLOSA2** de modo que seja apresentada a figura a seguir com tamanho **80**.



```

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.*

---

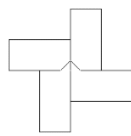
10. Criar procedimento chamado **RETANGULO3** com tamanhos **100** (altura) e **50** (largura). Movimente-se para frente com giro a direita.

```

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 **FAÇAPARA** contando de **0** a **2** no sentido a frente com giro a direita.

```

APRENDA TRIANGPR :TAMANHO
  FAÇAPARA [0 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 **FAÇAPARA**. Conte de **1** a **4**.

```

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

```

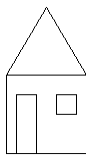
15. Criar procedimento chamado **QUADRADO3** que desenhe um quadrado com lado definido por parâmetro a partir do uso da primitiva **FAÇA.ATÉ**. Conte de **1** a **4**.

```

APRENDA QUADRADO3 :TAMANHO
  ATR "I 1
  FAÇA.ATÉ [
    PF :TAMANHO PD 90
    ATR "I :I + 1
  ] [:I > 4]
FIM

```

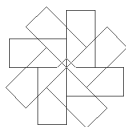
16. Criar procedimento chamado **CASINHA** que mostre a imagem da casa. O quadrado e o triângulo deverão possuir tamanho **80**, a porta é um retângulo de **60** por **20** e a janela é um quadrado de tamanho **20**.



```

APRENDA CASINHA
UL
FAÇAPARA [I 1 4] [PF 80 PD 90]
PF 80 PD 90
FAÇAPARA [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
FAÇAPARA [I 1 4] [PF 20 PD 90]
UN
MUDEPOS [0 0]
PE 90
UL
FIM
    
```

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



```

APRENDA CATAVENTO2
REPITA 8 [
    RETANGULO3 PD 45
]
FIM
    
```

18. 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 quadrado*

---

*criado a partir de quatro triângulos.*

---

## CAPÍTULO 5

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.

## APRENDA CAP0501

```
LOCAL "N
ESC [Entre valor para o cálculo:]
ATR "N LEIAP
ESC (FR "Resultado "= POTÊNCIA INTEIRO :N 2)
FIM
```

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.

## APRENDA CAP0502

```
LOCAL "N
LOCAL "R
ESC [Entre valor para o cálculo:]
ATR "N LEIAP
ATR "R POTÊNCIA INTEIRO :N 3
ESC (FR "Resultado "= :R)
FIM
```

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.

## APRENDA CAP0503

```
LOCAL "C
LOCAL "F
ESC [Entre valor da temperatura em graus Celsius:]
ATR "C LEIAP
ATR "F :C * 9 / 5 + 32
ESC (FR [Temperatura em Fahrenheit] "= FORMATONÚMERO :F 0 2)
FIM
```

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.

APRENDA CAP0504

```
LOCAL "F
LOCAL "C
ESC [Entre valor da temperatura em graus Fahrenheit:]
ATR "F LEIAP
ATR "C ((:F - 32) * 5) / 9
ESC (FR [Temperatura em Celsius] "= FORMATONÚMERO :C 0 2)
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
ESC [Entre valor o valor <A>:]
ATR "A LEIAP
ESC [Entre valor o valor <B>:]
ATR "B LEIAP
ATR "R POTÊNCIA (INTEIRO :A - INTEIRO :B) 2
ESC (FR [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
ESC [Entre valor numérico inteiro:]
ATR "N LEIAP
ATR "R (INTEIRO :N) * 2
SE :R > 30 [
  ESC (FR "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
ESC [Entre valor numérico real <A>:]
ATR "A LEIAP
ESC [Entre valor numérico real <B>:]
ATR "B LEIAP
SESENÃO :A > :B [
    ESC (FR "Resultado "= :A - :B)
][
    ESC (FR "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
ESC [Entre valor numérico real <A>:]
ATR "A LEIAP
ESC [Entre valor numérico real <B>:]
ATR "B LEIAP
SESENÃO :A > :B [
    ATR "R :A - :B
][
    ATR "R :B - :A
]
ESC (FR "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
  ESC [Entre valor numérico real <A>:]
  ATR "A LEIAP
  ESC [Entre valor numérico real <B>:]
  ATR "B LEIAP
  ESC [Entre valor numérico real <C>:]
  ATR "C LEIAP
  ATR "R INTEIRO :A + INTEIRO :B + INTEIRO :C
  SE :R >= 100 [
    ESC (FR "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 + CONTEVEZES
  ]
  ESC (FR "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
  ]
  ESC (FR "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 "FAÇA.ATÉ".

```
APRENDA CAP0512
  LOCAL "S
  ATR "S 0
  LOCAL "I
  ATR "I 1
  FAÇA.ATÉ [
    ATR "S :S + :I
    ATR "I :I + 1
  ] [:I > 100]
  ESC (FR "Somatório: "= :S)
FIM
```

13. Criar procedimento chamado **CAP0513** que apresente a soma dos cem primeiros números naturais "(1 + 2 + 3 + ... + 98 + 99 + 100)" utilizando-se a primitiva "FAÇAPARA".

```
APRENDA CAP0513
  LOCAL "S
  ATR "S 0
  FAÇAPARA [I 1 100] [
    ATR "S :S + :I
  ]
  ESC (FR "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 "REPITA".

```
APRENDA CAP0514
  LOCAL "S
  ATR "S 0
  REPITA 500 [
    SE (RESTO CONTEVEZES 2) = 0 [
      ATR "S :S + CONTEVEZES
    ]
  ]
  ESC (FR "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 "ENQUANTO".

```
APRENDA CAP0515
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
]
ESC (FR "Somatório: "= :S)
FIM
```

16. Criar procedimento chamado **CAP0516** que apresente o somatório dos *valores pares* existentes na faixa de "1" até "500". Use a primitiva "FAÇA.ATÉ".

```
APRENDA CAP0516
LOCAL "S
ATR "S 0
LOCAL "I
ATR "I 1
FAÇA.ATÉ [
  SE (RESTO :I 2) = 0 [
    ATR "S :S + :I
  ]
  ATR "I :I + 1
] [:I > 500]
ESC (FR "Somatório: "= :S)
FIM
```

17. Criar procedimento chamado **CAP0517** que apresente o somatório dos *valores pares* existentes na faixa de "1" até "500". Use a primitiva "FAÇAPARA".

```
APRENDA CAP0517
LOCAL "S
ATR "S 0
FAÇAPARA [I 1 500] [
  SE (RESTO :I 2) = 0 [
    ATR "S :S + :I
  ]
]
ESC (FR "Somatório: "= :S)
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 "REPITA".

```
APRENDA CAP0518
  REPITA 19 [
    SE (RESTO CONTEVEZES 4) = 0 [
      ESC CONTEVEZES
    ]
  ]
FIM
```

19. Criar procedimento chamado **CAP0519** que mostre todos os valores numéricos divisíveis por "4" *menores* que "20" iniciando a contagem em "1". Use a primitiva "ENQUANTO".

```
APRENDA CAP0519
  LOCAL "I
  ATR "I 1
  ENQUANTO [:I <= 19] [
    SE (RESTO :I 4) = 0 [
      ESC :I
    ]
    ATR "I :I + 1
  ]
FIM
```

20. Criar procedimento chamado **CAP0520** que mostre todos os valores numéricos divisíveis por "4" *menores* que "20" iniciando a contagem em "1". Use a primitiva "FAÇA.ATÉ".

```
APRENDA CAP0520
  LOCAL "I
  ATR "I 1
  FAÇA.ATÉ [
    SE (RESTO :I 4) = 0 [
      ESC :I
    ]
    ATR "I :I + 1
  ] [:I > 19]
FIM
```

21. Criar procedimento chamado **CAP0521** que mostre todos os valores numéricos divisíveis por "4" *menores* que "20" iniciando a contagem em "1". Use a primitiva "FAÇAPARA".

```
APRENDA CAP0521
  FAÇAPARA [I 1 19] [
    SE (RESTO :I 4) = 0 [
      ESC :I
    ]
  ]
FIM
```

22. Criar procedimento chamado **CAP0522** 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 CAP0522
  LOCAL "I
  ATR "I 0
  REPITA 5 [
    ESC :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" ou "FAÇA.ATÉ". Então é melhor usar as primiti-*

---

*vas "ENQUANTO" ou "FAÇA.ATÉ" e resolver o problema de forma mais adequada.*

---

23. Criar procedimento chamado **CAP0523** que apresente os valores numéricos inteiros compreendidos na faixa de "0" e "4". Use a primitiva "ENQUANTO".

```
APRENDA CAP0523
  LOCAL "I
  ATR "I 0
  ENQUANTO [:I <= 4] [
    ESC :I
    ATR "I :I + 1
  ]
FIM
```

24. Criar procedimento chamado **CAP0524** que apresente os valores numéricos inteiros compreendidos na faixa de "0" e "4". Use a primitiva "FAÇA.ATÉ".

```
APRENDA CAP0524
  LOCAL "I
  ATR "I 0
  FAÇA.ATÉ [
    ESC :I
    ATR "I :I + 1
  ] [:I > 4]
FIM
```

25. Criar procedimento chamado **CAP0525** que apresente os valores numéricos inteiros compreendidos na faixa de "0" e "4". Use a primitiva "FAÇAPARA".

```
APRENDA CAP0525
  FAÇAPARA [I 0 4] [
    ESC :I
  ]
FIM
```

26. Criar procedimento chamado **CAP0526** 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".*

27. Criar procedimento chamado **CAP0527** que apresente os valores numéricos inteiros compreendidos na faixa de "0" até "15" de "3" em "3". Use a primitiva "ENQUANTO".

```
APRENDA CAP0527
  LOCAL "I
  ATR "I 0
  ENQUANTO [:I <= 15] [
    ESC :I
    ATR "I :I + 3
  ]
FIM
```

28. Criar procedimento chamado **CAP0528** que apresente os valores numéricos inteiros compreendidos na faixa de "0" até "15" de "3" em "3". Use a primitiva "FAÇA.ATÉ".

```
APRENDA CAP0528
  LOCAL "I
  ATR "I 0
  FAÇA.ATÉ [
    ESC :I
    ATR "I :I + 3
  ] [:I > 15]
FIM
```

29. Criar procedimento chamado **CAP0529** que apresente os valores numéricos inteiros compreendidos na faixa de "0" até "15" de "3" em "3". Use a primitiva "FAÇAPARA".

```
APRENDA CAP0529
  FAÇAPARA [I 0 15 3] [
    ESC :I
  ]
FIM
```

## ANOTAÇÕES

[illegible]

---

## 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/jhbpge/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. **Language 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**. Califonia: 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.

SASSENATH, 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

---

---



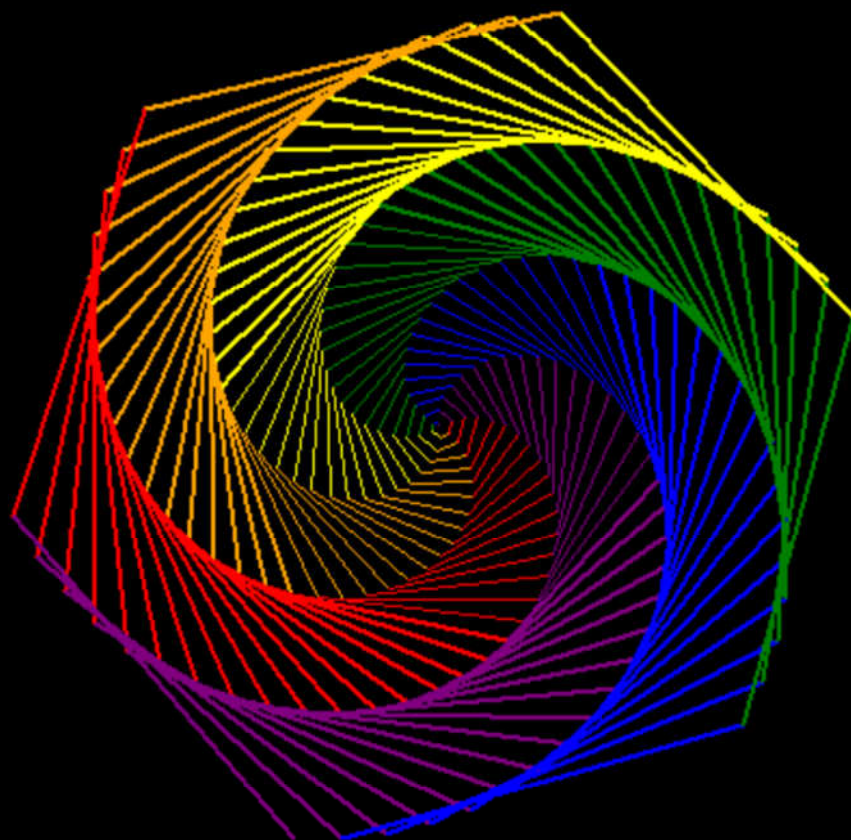






# LINGUAGEM LOGO

Introdução com FMSLogo



**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.**

