1ª Edição – corrigida

Ficha técnica

Título:	**Imagina, Cria e Constrói com a Tartaruga** **Manual técnico**
Manual original:	A. Blaho, I. Kalas, L. Salanci, P.Tomscanyi, J. Pixton Logotron, Ltd
Tradução e adaptação:	Patrícia Correia, Secundino Correia, Teresa Pinto, Tiago Correia
ISBN:	972-8336-15-2
Composição, Edição e Execução Gráfica:	Cnotinfor, Lda. Urbanização Panorama, Lote 2, Loja 2 | Monte Formoso 3000 – 446 Coimbra Tel: (+351) 239 499 235 Fax: (+351) 239 499 239 info@cnotinfor.pt www.cnotinfor.pt

Introdução. A Linguagem Logo

A linguagem Logo é provavelmente o único software educativo usado em todo o mundo, para iniciar as crianças no mundo multifacetado dos computadores digitais. O seu objectivo e inspiração, nas palavras de Seymour Papert, é "permitir às crianças programar computadores em vez de ter crianças programadas pelos computadores". Tem sido, muitas vezes, descrita como o único ambiente computacional que também pressupõe uma filosofia de educação.

A Linguagem Logo nasceu nos laboratórios de inteligência artificial do Instituto de Tecnologia de Massachusetts (MIT), no início da década de 70. Seymour Papert, um dos seus principais arquitectos, é um matemático e psicólogo cognitivista, que estudou na linha de pensamento do psicólogo Jean Piaget. Qualquer leitor que deseje conhecer mais aprofundadamente as ideias de Papert, deverá ler: **Logo: Computadores e Educação**, São Paulo, Ed. Brasiliense [*Mindstorms: Children, Computers and Powerful Ideas*]; **A Máquina das Crianças**, Porto Alegre, Artes Médicas [*The children's Machine: rethinking School Education in the Age of the Computer*]; **A Família em Rede**, Lisboa, Relógio de Água Editores.

Papert e os seus colegas, Cynthia Solomon e Wally Feurzeig, entre outros, trabalharam tendo por base a ideia central de Piaget de que as crianças aprendem por descoberta e experiência e de que a função do professor é proporcionar um ambiente onde essa aprendizagem possa ocorrer. Papert irritava muitos professores, quando lhes dava a entender que eles, na sua sala de aula, não acompanhavam as crianças e os seus computadores.

De facto, agora sabemos que os professores são um factor fundamental para um grupo de crianças poder ter uma iniciação no Logo com sucesso. A interacção professor/aluno já não é a mesma do mundo do giz e do quadro, pelo que é necessário aprender novas formas de intervenção. Contudo, o papel do professor na condução e apoio das crianças continua a ser muito importante.

As palavras de Papert foram mal interpretadas, ao defender com ênfase que as crianças deverão ser os responsáveis pelo computador e que deverão ser elas a tomar a iniciativa. Há vários anos que este tem sido um tema central em discussão na Educação Básica Britânica.

Nos primeiros anos da introdução de computadores nas escolas, a linguagem Logo constituía uma alternativa um tanto heterodoxa à corrente que vigorava e que defendia o uso do computador como uma máquina de ensinar ou a "Ensino Assistido por Computador – EAC", como lhe chamavam. A ideia por trás do EAC era fornecer à criança informações sequenciais, de forma cuidadosa, interpoladas com questões de dificuldade

gradual. A criança progredia através do trabalho com uma série de módulos curriculares específicos. A filosofia educacional que está por trás do EAC tradicional deve mais ao Behaviorista de Harvard B. F. Skinner do que a Piaget.

Muitos estudiosos das políticas educacionais, especialmente nos Estados Unidos, estavam interessados em descobrir se as crianças aprendiam mais depressa com o EAC ou com os métodos tradicionais, baseados no professor e nos manuais.

Hoje, existem boas e diferentes perspectivas para o uso do computador na sala de aula, de forma criativa. Têm-se registado magníficos desenvolvimentos no uso de processadores de texto, da robótica, da música e da matemática. Muitas dessas aplicações foram inspiradas por professores, que começaram com o Logo e depois aplicaram a sua filosofia de modos diferentes.

O Logo em si próprio tem sofrido um processo de desenvolvimento e transformação, à medida que vão surgindo novos computadores, cada vez mais potentes.

Este manual foi concebido como uma introdução à Programação em Logo – a linguagem do Imagina. Não foi concebido para lhe apresentar todas as suas características principais. Para isso, o melhor será descobri-las através da experiência e exploração. O objectivo é familiarizá-lo com os pontos mais importantes da Linguagem, bem como com os seus princípios subjacentes. Em particular, focamo-nos nos tópicos que mais frequentemente constituem dificuldades na aprendizagem da Linguagem. Todos os exemplos apresentados podem ser experimentados e, para isso, apontamos algumas indicações. Quanto mais tempo dedicar aos exemplos, melhor será.

O texto deste manual foi concebido mais para os professores do que para as crianças[1], o que não invalida que algumas crianças o possam utilizar como um recurso para a sua própria programação. No entanto, o texto foi escrito para professores, de forma a dar cobertura a um vasto leque de materiais, de uma maneira relativamente condensada. Assim, será possível a um professor que trabalhe com crianças bastante novas, adaptar esse mesmo trabalho a um contexto que também desafie crianças do 2º ciclo do Ensino Básico, por exemplo, para novas formas de trabalhar.

Todo o manual pressupõe o acesso a um computador. Este manual não fará muito sentido, se não se trabalharem os exemplos. Para isso, apresentamos um amplo leque de materiais em aberto (sem um fim pré-definido), que requer a participação activa do utilizador. O Logo só se pode aprender com o computador e através da experiência, quer se trate de professores ou de crianças.

[1] Para as crianças e mais jovens, criámos um manual específico e animado com exemplos e linguagem mais adequados à sua realidade.

Professores e alunos poderão aprender o Logo ao mesmo tempo, desde o início. Mas se o objectivo do exercício é conhecer as suas reais potencialidades, o melhor será os professores dedicarem algum tempo a aprender o Logo como deve ser, por si próprios. Assim, os professores terão uma compreensão muito mais profunda do trabalho das crianças e estarão em melhor posição para poder ajudá-las nos seus projectos, quando necessário. Também saberão melhor quando devem deixar as crianças procurar as respostas sozinhas. Trata-se de um investimento de vários dias de estudo intensivo e muito compensador.

A pretensão que fazemos para o Logo é que é muito mais fácil aprendê-lo como um principiante e que os resultados interessantes aparecem mais depressa do que em qualquer outro ambiente de programação.

A maior parte deste livro trata da Geometria da Tartaruga, pois esta é a introdução mais simples para iniciar programação de computador em qualquer linguagem. A beleza da Geometria da Tartaruga é o facto de podermos sempre ver o que fizemos e o que está errado. O *feedback* do trabalho é visual e instantâneo. Os primeiros capítulos poderão parecer algo lentos, mas a nossa experiência diz-nos que é errado começar com ideias muito emocionantes, sem que o utilizador tenha primeiramente absorvido bem as bases. Se já tem alguma experiência nesta matéria, poderá trabalhar os primeiros capítulos numa só sessão.

Bom trabalho!

E já sabe, em caso de dúvida, não hesite em contactar-nos,

por e-mail info@cnotinfor.pt ou por telefone (+351) 239 792 825.

Capítulo 1. A Tartaruga

Algumas Convenções:

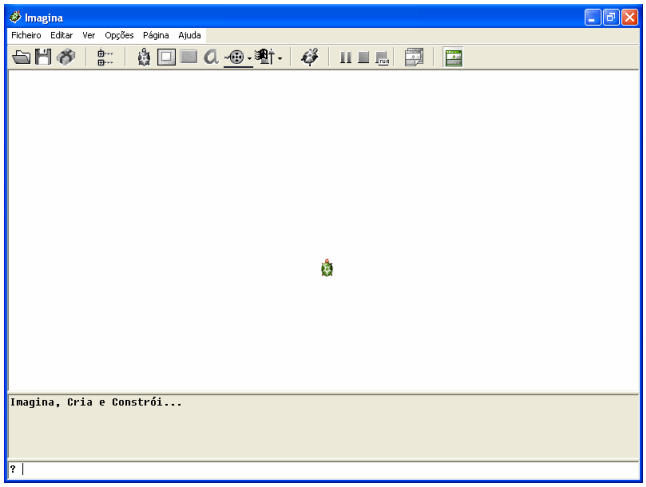
1. Ao longo de todo este manual, vamos usar diferentes tipos de letra para indicar interacções com o computador.
2. Toda a programação a efectuar pelo utilizador surge deste modo: **`avança 50`**.
3. As respostas e as mensagens dadas pelo Imagina possuem o mesmo tipo de letra, mas não estão a negrito: **`não sei como fazer...`**.
4. Se uma palavra estiver entre **`< >`**, indica o nome de uma tecla que deve ser pressionada: **`<Enter>`**.

> **Nota**
> No Imagina, pode escrever os comandos de diversas maneiras: com letra maiúscula ou minúscula ou uma mistura dos dois tipos de letra. O Imagina entende-o de qualquer das formas.

1. Começar

Abra o Imagina.

Aparece um cenário em branco com as palavras **`Imagina, Cria e Constrói...`** e com um ponto de interrogação (**`?`**) no fundo do ecrã à esquerda. O ponto de interrogação é um convite para que escreva uma instrução para o computador.



Escreva, por exemplo **et**, a abreviatura de **escondetartaruga** mesmo à direita do ponto de interrogação e carregue na tecla **<Enter>**. A tartaruga no ecrã desaparece.

No fundo do ecrã, aparece um ponto de interrogação, aguardando as suas instruções. (nunca tem de escrever o **?**, uma vez que ele aparece automaticamente).

Escreva:

```
? mt
? et
?
```

Lembre-se de carregar na tecla **<Enter>** depois de escrever **mt** e **et**.

mt indica **mostratartaruga**. Ao escrever **mt**, a tartaruga reaparece.

Pode querer saber porque é que chamamos **Tartaruga** ao objecto com o qual trabalhamos na Linguagem LOGO...

2. Tudo sobre as Tartarugas

Quando Seymour Papert veio da África do Sul para a Inglaterra, no início dos anos 60, um célebre neurologista chamado Gray Walter encontrava-se a realizar experiências com pequenos robots eléctricos aos quais chamou de Cágados. Quando Seymour estava a trabalhar no Instituto de Tecnologia de Massachusetts (MIT) à procura de maneiras de interessar as crianças pelo computador, lembrou-se dos Cágados de Gray Walter. Posteriormente, chamou Tartarugas aos seus robots, uma vez que as crianças americanas estavam mais familiarizadas com Tartarugas do que com Cágados.

A tartaruga original do MIT era um aparelho muito simples. Parecia um cesto de papéis de metal, sobre rodas. Podia andar para a frente (**avança**), para trás (**recua**), virar para a **direita** ou para a **esquerda**; um outro motor controlava um braço que segurava uma caneta; a caneta podia estar levantada (**levantacaneta**) ou baixada (**baixacaneta**).

A ideia era escrever programas simples que explorassem as principais características do robot. Estes robots ainda estão disponíveis. Chamamos-lhes **Robot Roamer** e recomendamos vivamente o seu uso com crianças pequenas, antes de trabalharem com o computador.

A tartaruga do seu ecrã comporta-se tal e qual como a tartaruga robot do MIT. Experimente, escrevendo:

```
? avança 150
```

e carregue na tecla Enter. A tartaruga movimenta-se em frente, desenhando uma linha à medida que avança.

Agora escreva:

```
? direita 45 <Enter>
? avança 75 <Enter>
?
```

Experimente por alguns minutos conduzir a tartaruga pelo ecrã, utilizando diferentes ângulos de direcção e diversos números de passos. É mais fácil se pensar na tartaruga como um robot que recebe e executa instruções. Não acontece nada sem que haja uma decisão e uma instrução do utilizador.

Pode usar qualquer um dos comandos **esquerda**, **direita**, **recua** ou **avança**. Também pode usar as abreviaturas correspondentes: **esq**, **dta**, **re**, **av**, respectivamente.

Aprender a Escrever

Quando utiliza a linguagem Logo pela primeira vez, provavelmente vai receber algumas mensagens do programa, como por exemplo:

```
? avança100
Não sei como fazer avança100
O procedimento não foi definido ou foi mal escrito.

? avnça 100
Não sei como fazer avnça 100
```

```
O procedimento não foi definido ou foi mal escrito.
```

No primeiro exemplo, esqueceu-se de colocar um espaço entre **avança** e **100**, e o computador leu a instrução como uma só palavra. Escreva de novo a instrução original correctamente e a tartaruga move-se.

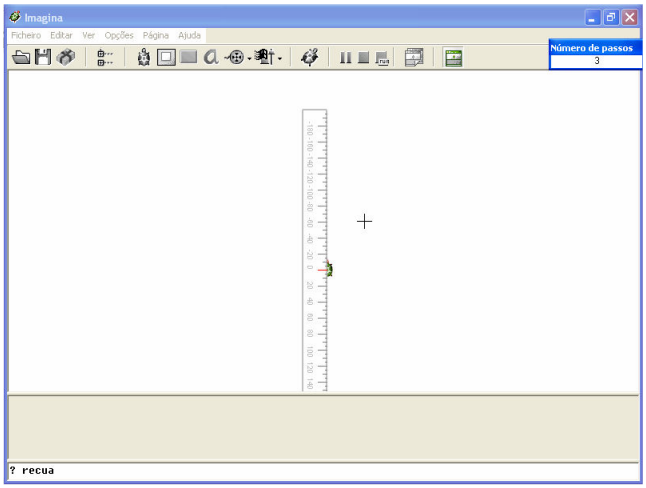
Pode parecer-lhe que, pelo facto de o computador aceitar **avança** e **av**, também pode aceitar **avnça**. No entanto, os computadores não são lá muito espertos e interpretam as instruções sempre à letra, por isso, tem sempre que escrever e soletrar cuidadosamente os comandos. Contudo, é fácil corrigir os seus erros:

Prima a tecla de direcção para cima e edite a sua instrução.

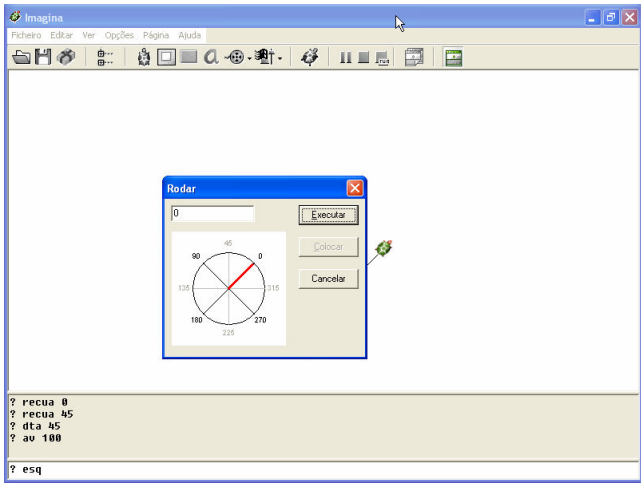
O Imagina também tenta ajudá-lo.

Escreva:

```
? recua
?
```



Aparece uma régua no ecrã. Isto acontece porque não indicou quantos passos quer que a tartaruga dê. Clique na régua o número de passos que deseja que a tartaruga recue. Existe uma pequena janela no canto superior direito do ecrã que permite visualizar os passos exactos da tartaruga.



Os comandos de direcção também necessitam de saber quantos graus devem virar.

Repare neste exemplo:

```
? dta 45
? av 100
? esq
?
```

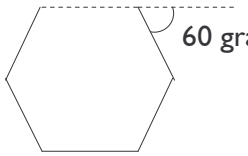
Surge no ecrã uma janela com uma circunferência marcada com diversos ângulos. Pode escrever na janela o n.º de ângulos que deseja ou clicar directamente na circunferência, e depois clicar no botão **Executar**.

No que diz respeito aos ângulos, não há nenhuma necessidade de ensinar este assunto às crianças, antes de elas experimentarem e trabalharem com a linguagem Logo. Depressa vão descobrir que os números maiores produzem um efeito muito diferente comparativamente com números pequenos. As crianças intuitivamente desenvolvem e compreendem os conceitos de "graus" e de "ângulo".

Pais e professores (não as crianças) que aprenderam a geometria convencional na escola, poderão ser confundidos pelo facto de a tartaruga virar através de um ângulo, por isso, a melhor maneira de o perceber é pensar em termos de ângulos externos e não internos.

Este é o ângulo de rotação da tartaruga (30 graus)

```
direita 30
```




```
repete 6 [av 50 dta 60]
```

3. Muitos Comandos

Numa só linha, é possível combinar mais do que um comando. Assim, pode escrever:

```
? avança 100 direita 30 avança 100 direita 30 recua 370 esquerda 30 avança 256
?
```

Assim que carregar na tecla **<Enter>**, a tartaruga executa a sequência de instruções indicada. Se a linha exceder a largura do ecrã, a tartaruga continua a escrever para além do ecrã. Se quiser limpar o ecrã e começar de novo, escreva

```
? le
?
```

Le é a abreviatura de **limpaecrã**. A tartaruga retoma sua posição inicial no ecrã.

Se escrever, pode ver que a tartaruga se comporta de um modo bem diferente.

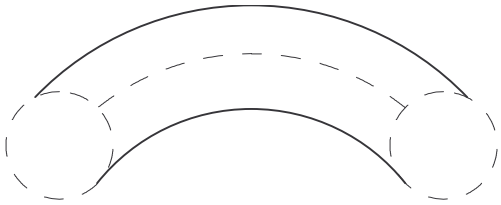
```
? esquerda 60 av 100 limpafundo
```

```
?
```

4. Sem Limites

Facilmente vai verificar que ao conduzir a tartaruga para fora dos limites do ecrã, ela reaparece no outro lado. No entanto, a imagem pode estar temporariamente escondida pela área de trabalho do ecrã. A altura da área de trabalho do ecrã pode ser alterada, arrastando a área de texto para cima ou para baixo. As crianças aceitam espontaneamente que o ecrã reaja deste modo, mas pode interrogar-se porque é que reaparece precisamente naquele sítio. Uma forma de explicar isto, é pegar numa folha A4, fazer um rolo com ela, dobrar as pontas de forma a juntá-las e formar um *donuts*, ou, mais tecnicamente, um toro.

A tartaruga movimenta-se no ecrã tal como se se movimentasse na superfície desse objecto, que tem, portanto, uma superfície contínua, sem limites. Trata-se pois de uma espécie de *donuts* oco, cortado pelas linhas tracejadas, de modo a formar um ecrã rectangular. Imagine a tartaruga a circular pela superfície de um toro não cortado. As linhas tracejadas são os limites do ecrã.



Tente adivinhar os passos da tartaruga. Por exemplo, quantos passos vão desde o início do trajecto da tartaruga até ao topo?

A outra característica que esta tartaruga electrónica tem em comum com a velha tartaruga de solo é a caneta, que tanto pode estar para baixo e desenhar, como pode estar levantada. O Logo controla a caneta com dois comandos: **`bc`** que é a abreviatura de **`baixacaneta`** e **`lc`** que é a abreviatura de **`levantacaneta`**. Nenhum dos comandos requer qualquer dado numérico. A caneta possui um de dois estados: baixada ou levantada.

Experimente o seguinte conjunto de comandos:

```
? le lc avança 100 direita 90 bc avança 50
?
? le bc avança 20 lc avança 20 bc avança 20 lc avança 20 bc avança 20 lc avança 20 bc avança 20 lc avança 20
?
```

Assegure-se de que já se sente à vontade com os comandos descritos neste capítulo, antes de passar ao seguinte.

Verifique-os um por um e se estiver confuso em relação a algum deles, experimente-o outra vez. Uma pequena sugestão: se, nas suas experiências iniciais criar algumas formas que o agradem em especial, anote os comandos que utilizou para as criar, e tenha-os à mão quando iniciar o capítulo seguinte.

Capítulo 2. Quando Usar o Computador Faz Diferença

Tudo aquilo que fez no primeiro capítulo teria sido mais fácil de fazer com um lápis, papel, borracha e transferidor.

Repare outra vez no conjunto de instruções dadas no final do último capítulo.

```
? le bc avança 20 lc avança 20 bc avança 20 lc avança 20 bc avança 20 lc avança
20 bc avança 20 lc avança 20
?
```

Os computadores realmente apreciam tarefas repetitivas.

Este conjunto de comandos pode ser escrito novamente, de uma forma mais clara, com cinco linhas:

```
? le
? bc avança 20 lc avança 20
? bc avança 20 lc avança 20
? bc avança 20 lc avança 20
? bc avança 20 lc avança 20
?
```

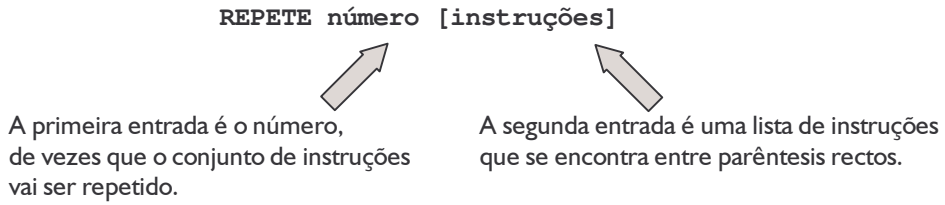
No entanto, é mais simples escrever:

```
? le
? repete 4 [bc avança 20 lc avança 20]
```

Se não resultar, é provável que o problema esteja na utilização de parêntesis rectos. Os parêntesis curvos **(** ou chavetas **{** não servem. O Logo apenas aceita parêntesis rectos **[**.

À medida que for evoluindo no Logo, vai usar este tipo de parêntesis com muita frequência. Eles permitem agrupar "listas", que podem ser listas de instruções, listas de palavras, listas de números. Neste caso, trata-se de uma lista de instruções. A linguagem Logo encontra **`repete`** seguido de um número que, por sua vez, está seguido de uma lista de instruções. Essa lista é executada tantas vezes quantas as que o referido número indicar.

O comando **`repete`** possui duas entradas:



Substitua a palavra "número" por um número e coloque entre os parêntesis quaisquer instruções que queira ver repetidas. Experimente o seguinte:

```
? le av 200 esq 90 av 200 esq 90 av 200 esq 90 av 200 esq 90 et
?
```

que é o mesmo que:

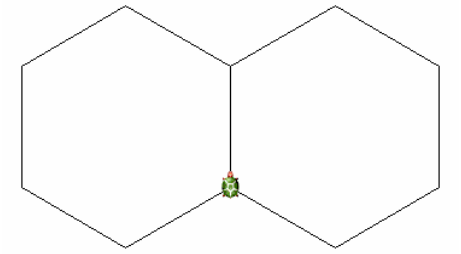
```
? le repete 4 [av 200 esq 90] et
?
```

Esta segunda versão não só requer menos palavras, como também permite entender mais facilmente o que está a acontecer. Comece cada grupo de instruções com **le**, pois torna-se mais fácil ver o que está a fazer com o ecrã limpo; e termine com **et**, porque se esconder a tartaruga, possui uma melhor visualização do desenho. No entanto, estes detalhes são completamente opcionais. Se tiver escondido a tartaruga e quiser que ele apareça novamente, escreva **mt** no início da sequência de instruções seguinte. A tartaruga continua a desenhar no ecrã, independentemente de estar visível ou escondida. Se ainda pensa que consegue desenhar mais facilmente com uma régua e um lápis, veja o seguinte:

```
? le repete 5 [av 150 dta 72]
?
```

Agora, desenhe um octógono utilizando a mesma abordagem; depois tente desenhar um par de hexágonos.

Rapidamente, vai descobrir que, utilizando o comando **repete**, pode produzir padrões invulgares e formidáveis, que seriam muito difíceis de duplicar, pelos métodos tradicionais.



Experimente o seguinte:

```
? le repete 36 [av 100 re 100 dta 10]
?
```

Para obter padrões ainda mais fantásticos, pode repetir o comando **repete**.

Escreva:

```
? le repete 12 [repete 4 [av 150 dta 90] dta 30] et
?
```

Agora, já está em condições de rodar os pentágonos e hexágonos que criou.

Um problema final para resolver: desenhe um rectângulo, que não seja quadrado, completando a seguinte linha de instruções:

```
? repete 2 [.........]
```

Substitua os pontos pelas suas instruções, de forma a desenhar um rectângulo.

1. Imprimir o seu trabalho

Provavelmente, deseja imprimir o trabalho que já fez até agora no Imagina. Para imprimir os desenhos do ecrã, proceda da seguinte forma:

Escolha a opção **Janela de Gráficos** do menu **Ver**.

Depois, escolha **Imprimir** do menu **Ficheiro** para obter uma cópia do seu desenho.

Capítulo 3. Procedimentos do Logo

Repare neste comando que cria um hexágono:

```
? le repete 6 [av 100 dta 60]
```

É possível ligar estas instruções à palavra **hexágono**, e depois escrever apenas:

```
? hexágono
```

Para isso, tem que definir, em primeiro lugar, o significado de **hexágono**. Esta operação pode ser feita de duas maneiras.

1. Método Um

Escreva o seguinte:

```
? para hexágono
> repete 6 [av 100 dta 60]
> fim
```

Assim que carregar na tecla **<Enter>**, depois de escrever **para hexágono**, aparece um novo símbolo na linha de comandos. Em vez do habitual **?**, surge **>**. Isto significa que o Logo não vai executar mais nenhuma instrução, até que complete a definição do **hexágono**.

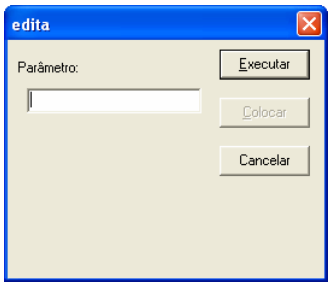
A definição encontra-se terminada para o Logo, assim que escrever a palavra **fim**. A palavra **fim** deve estar numa linha isolada, sem qualquer outro comando.

Depois de terminada a definição, verifique se funciona. Escreva:

```
? hexágono
? le repete 3 [hexágono dta 120]
```

Desta forma, pode criar qualquer nome para um procedimento, a partir de uma sequência de instruções.

2. Método Dois



Escreva:

```
? edita
```

e carregue na tecla **<Enter>**.

Acabou de entrar no **Editor**. Ao utilizar o Logo; pode estar na **Área de Trabalho** ou no **Editor**. Até agora, estivemos a trabalhar exclusivamente na Área de Trabalho, onde o Logo está vivo e à espera das suas instruções, com um **?** pronto a utilizar no fundo do ecrã.

Para cada linha escrita, o Logo tem respondido de três formas:

- Se era uma sequência de comandos Logo, ele executava-os;
- Se era uma nova definição, começando por **para**, o símbolo **?** passava a **>** e a definição era guardada;
- Se escrevesse qualquer outra coisa, o Logo respondia com uma mensagem a pedir orientação.

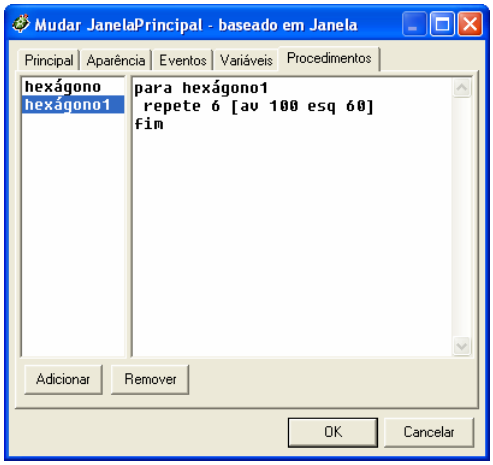
O **Editor** funciona de maneira diferente. É usado como uma área para escrever, onde pode criar e corrigir procedimentos.

Na zona de texto da caixa de diálogo que surge, escreva:

```
hexágono
```

e carregue no botão **Executar**.Pode editar o procedimento existente ou clicar no botão Adicionar para criar um novo procedimento no **Editor**. Este novo procedimento deve ser escrito exactamente como se estivesse na **Área de Trabalho**, mas não tem que se preocupar com os símbolos (**?** ou **>**):

```
para hexágono1
repete 6 [av 100 esq 60]
fim
```



O **hexágono1** é em tudo semelhante ao **hexágono**, com apenas uma pequena diferença. Para continuar, clique no botão **OK**.

Agora está de volta à **Área de Trabalho** e pode confirmá-lo escrevendo:

```
? le hexágono hexágono1
```

Existe um número de características importantes do **Editor**, que lhe vão facilitar a criação e a modificação de procedimentos.

Escreva:

```
? edita "hexágono
```

Neste caso, está a abrir um procedimento no **Editor**. Se não definiu este procedimento, recapitule a parte inicial deste capítulo, e defina-o agora. Sempre que se referir a um procedimento simples pelo nome, sem querer que o computador execute todos os comandos a ele associados, escreva um único símbolo de aspas no início da palavra. Por exemplo: **"hexágono** ou **"hexágono1**. Observe o seu procedimento **hexágono** no ecrã do **Editor**.

```
para hexágono
repete 6 [av 100 dta 60]
fim
```

Utilize o rato para entrar no procedimento. Pode usar as teclas de direcção, para se movimentar:

Seta para a esquerda	Move o cursor um carácter para a esquerda
Seta para a direita	Move o cursor um carácter para a direita
Seta para baixo	Move o cursor uma linha abaixo
Seta para cima	Move o cursor uma linha acima
Shift + Seta para a esquerda	Selecciona o texto à esquerda do cursor
Shift + Seta para a direita	Selecciona o texto à direita do cursor

Experimente executar esta sequência, utilizando o rato ou as teclas de direcção:

1. Posicione o cursor junto do **h** de **hexágono** . Apague o **h** e escreva **p**. Da mesma forma, substitua o **x** por **nt**.
2. Altere o número de vezes a repetir para **5** e o número de **dta** para **72**.

```
para pentágono
le repete 5 [av 100 dta 72]
fim
```

3. Clique no botão **OK**.

Escreva:

```
? le pentágono hexágono
```

Experimente criar um terceiro procedimento no **Editor**, que desenhe primeiro um hexágono e depois um pentágono. (Chame-lhe **hex.pent**). O nome de um procedimento deve consistir sempre numa só palavra. Por isso, é conveniente que use um ponto final para separar duas palavras num só nome. Se um procedimento não funcionar, regresse ao **Editor**, modifique-o e depois volte à Área de Trabalho. Escreva:

```
? edita "hex.pent
```

Está de novo no **Editor**, com os seus procedimentos tal como os deixou.

Experimente mais alguns procedimentos Logo. Se escrever **edita** seguido de " e do nome do procedimento, o procedimento que definiu é aberto no **Editor**. Os outros procedimentos que definiu, aparecem listados à esquerda. Para sair do **Editor**, basta carregar em **OK** ou em **Cancelar**. Experimente escrever **edita** seguido de um nome de um procedimento que ainda não tenha definido, por exemplo:

```
? edita "F00
```

Se desejar, pode criar um procedimento para **F00**. Uma vez no **Editor**, pode escolher qualquer procedimento da lista, seleccioná-lo e editá-lo.

Antes de terminar esta sessão, certifique-se de que guardou uma cópia do seu trabalho. Escreva:

```
? grava "polígonos
```

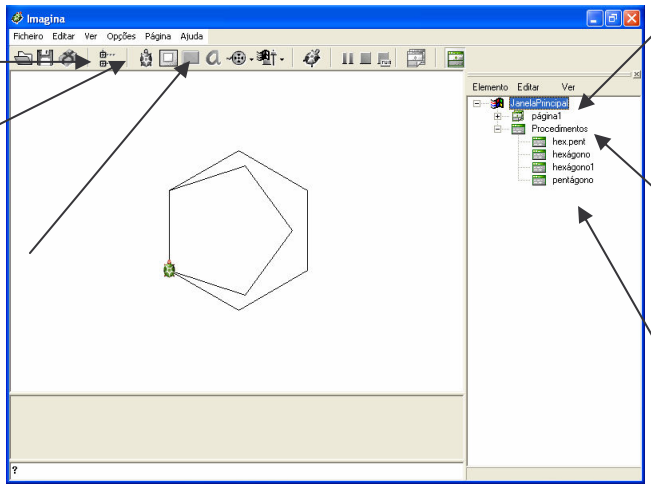
Dê uma vista de olhos final aos comandos usados até agora. Experimente também os comandos **edita**, **grava**, **abre** e **elimina**.

Em alternativa, repare no seguinte ecrã:

Clique aqui para abrir um projecto.

Clique aqui para gravar um projecto.

Clique aqui para abrir o **Explorador** do projecto.



Explorador

Aqui pode editar ou apagar os procedimentos do seu projecto.

Seleccione o nome de um procedimento com apenas um clique no rato e use a tecla **<Delete>** para o apagar.

Faça duplo clique sobre o nome de um procedimento para o editar.

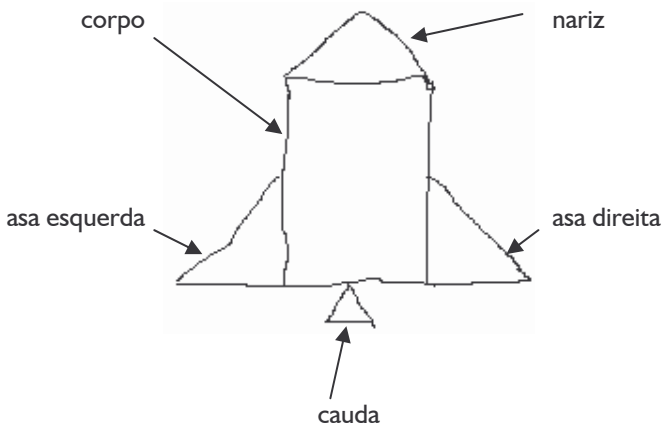
Capítulo 4. Desenvolver um Projecto

Quando se está a desenvolver um projecto, torna-se muito útil possuir uma metodologia de trabalho que ajude a identificar claramente a estrutura do problema que estamos a tentar resolver.

Para a solução de um mesmo problema, pessoas diferentes podem sentir-se mais à vontade com diferentes formas de abordagem. O exemplo que se segue ilustra uma abordagem que pode experimentar.

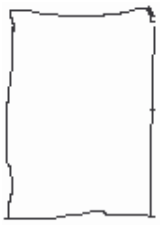
Primeiro, faça um esquema ou um rascunho daquilo que pretende que seja o seu projecto final. Este vai ajudá-lo a organizar ideias e o modo como vai trabalhar no seu projecto.

1. O meu Projecto Foguetão



Olhe atentamente para o projecto e tente atribuir um nome às várias partes.

Para simplificar, em vez de escrever um procedimento longo e extenso, que faça tudo, é melhor subdividir o seu projecto em partes mais simples. Pode definir um procedimento para cada parte do projecto, testá-los separadamente, e depois juntá-los para criar o desenho original.


corpo


nariz


asa esquerda

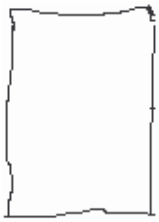

asa direita

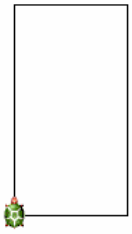

cauda

Aqui estão as várias partes do foguetão. Agora que já definimos como vamos dividir as partes do projecto **foguetão**, podemos começar.

Escreva um procedimento para o **corpo** do foguetão.

```
para corpo
avança 150
direita 90
avança 80
direita 90
avança 150
direita 90
avança 80
direita 90
fim
```

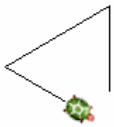




Experimente escrever **corpo**, e compare o que a tartaruga faz com o resultado esperado.

Repita o mesmo processo para cada uma das partes do foguetão.

```
para asa.esq
avança 60
esquerda 120
avança 85
esquerda 120
avança 60
fim
```

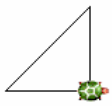


Quando testar a **asa esquerda** não vai ter o resultado esperado. A tartaruga não virou o suficiente. Assim sendo, edite o procedimento e modifique os ângulos de rotação para um número maior.

```
para asa.esq
avança 60
esquerda 150
avança 85
esquerda 150
avança 60
fim
```

Teste a nova versão. Agora o ângulo de curvatura é demasiado grande. Edite, novamente, o procedimento para a **asa esquerda**, experimentando um número maior do que 120 e menor do que 150.

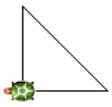
```
para asa.esq
avança 60
esquerda 135
avança 85
esquerda 135
avança 60
fim
```



Este procedimento já funciona.

Agora vamos à **asa direita**.

```
para asa.dta
avança 60
direita 135
avança 85
direita 135
avança 60
fim
```



Agora, vamos definir o **nariz**.

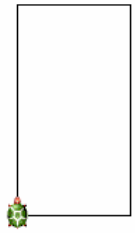
```
para nariz
esquerda 30
avança 80
esquerda 120
avança 80
fim
```

e finalmente a **cauda**.

```
para cauda
direita 30
avança 45
direita 120
avança 45
direita 120
avança 45
fim
```

Uma vez terminadas as várias partes do foguetão, pode começar a construi-lo.

```
para foguetão
le
corpo
fim
```

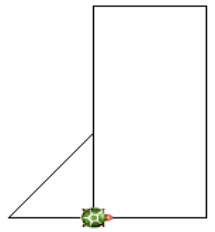


Este é o procedimento mais importante. Acrescente a parte seguinte do foguetão, um bocadinho de cada vez, teste-a e, se funcionar, passe à parte seguinte.

Se algo não funcionar em qualquer das fases, pode editar facilmente o seu problema, escrevendo:

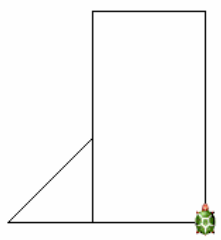
```
? edita "foguetão
```

Pode editar o procedimento **foguetão** e acrescentar a asa esquerda.



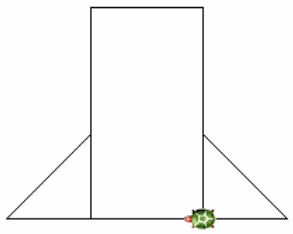
```
para foguetão
le
corpo
asa.esq
fim
? foguetão
```

Antes de acrescentar a asa seguinte, tem que deslocar a tartaruga para o outro lado do corpo do foguetão, adicionando uma instrução **avança**. Para que a tartaruga fique na posição correcta para desenhar a asa direita, é necessário que ela fique virada para cima, pelo que também é necessário acrescentar uma instrução **esquerda**.



```
para foguetão
le
corpo
asa.esq
avança 80
esquerda 90
fim
? foguetão
```

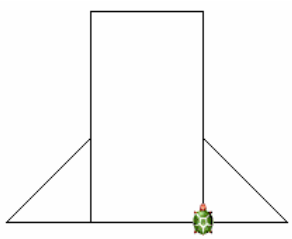
Agora, a tartaruga está virada para o lado direito para fazer a asa direita do foguetão. Edite o procedimento principal **foguetão** e acrescente a **asa direita**.



```
para foguetão
le
corpo
asa.esq
avança 80
asa.dta
fim
? foguetão
```

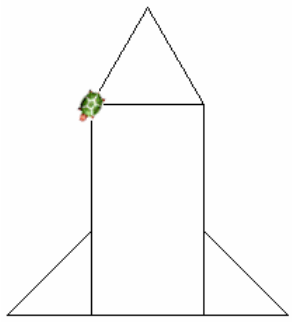
Antes de poder acrescentar o **nariz**, tem que deslocar a tartaruga para o topo do corpo do foguetão. Em primeiro lugar deve virá-la para cima, ou seja, deve virá-la para a **direita 90** graus. Experimente.

```
para foguetão
le
corpo
asa.esq
avança 80
asa.dta
direita 90
fim

? foguetão
```



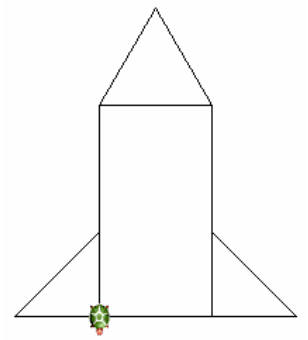
Uma instrução **avança 150** deve deixar a tartaruga na posição correcta para colocar o nariz. Edite novamente o foguetão e adicione estas instruções ao procedimento.

```
para foguetão
le
corpo
asa.esq
avança 80
asa.dta
direita 90
avança 150
nariz
fim
? foguetão
```



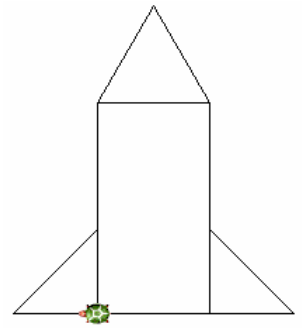
Seguidamente, é necessário deslocar-se para a base do corpo do foguetão, para poder acrescentar a **cauda**. Experimente virar a tartaruga para a **esquerda 30** graus e **avançar 150** passos, pois esse é o comprimento do corpo.

```
para foguetão
le
corpo
asa.esq
avança 80
asa.dta
direita 90
avança 150
nariz
esquerda 30
avança 150
fim
? foguetão
```



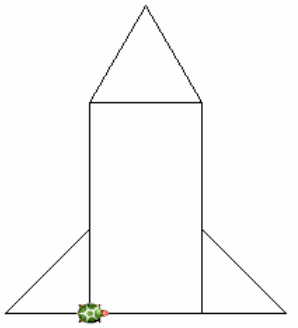
Agora, desloque a tartaruga para o meio do corpo do foguetão e vire a tartaruga. Primeiro, é necessário virá-la para o lado correcto, por isso, insira **direita 90**.

```
para foguetão
le
corpo
asa.esq
avança 80
asa.dta
direita 90
avança 150
nariz
esquerda 30
avança 150
direita 90
fim
? foguetão
```



A tartaruga está agora virada para a direcção contrária à desejada. Edite o procedimento novamente e modifique a instrução **direita 90** para **esquerda 90**.

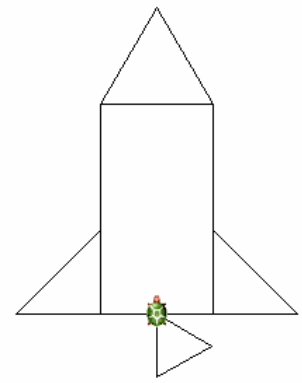
```
para foguetão
le
corpo
asa.esq
avança 80
asa.dta
direita 90
avança 150
nariz
esquerda 30
avança 150
esquerda 90
fim
? foguetão
```

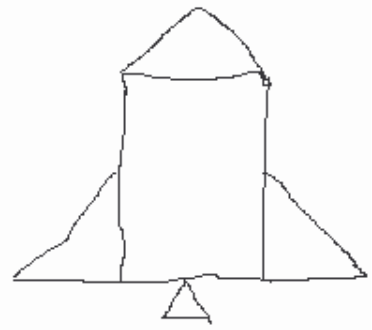


Agora desloque-se até meio do corpo do foguetão. Escreva **avança 40**, porque o corpo tem 80 no total. Depois, adicione a **cauda**.

Pode necessitar de usar a barra de deslocamento do editor para visualizar todo o procedimento.

```
para foguetão
le
corpo
asa.esq
avança 80
asa.dta
direita 90
avança 150
nariz
esquerda 30
avança 150
esquerda 90
avança 40
cauda
```

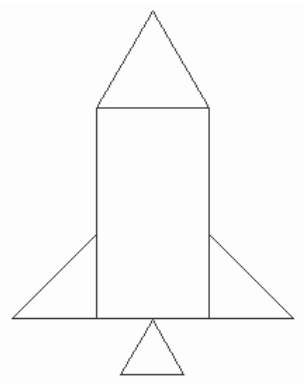


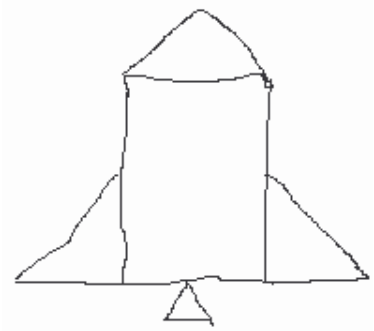


```
fim
? foguetão
```

A cauda está na posição errada. Assim, tem que colocar a tartaruga na posição correcta. Experimente.

```
para foguetão
le
corpo
asa.esq
avança 80
esquerda 90
asa.dta
direita 90
avança 150
nariz
esquerda 30
avança 150
esquerda 90
avança 17
esquerda 90
lc
recua 40
bc
cauda
et
fim
? foguetão
```





Está satisfeito com o resultado?

Compare-o com o desenho original.

Quer fazer algumas alterações?

Não se esqueça de guardar o seu trabalho!

```
? grava "foguetão
```

Antes de passar ao capítulo seguinte, tente desenvolver um projecto sozinho, de um modo semelhante ao que fizemos. Escolha algo simples, de preferência sem curvas, e centre-se na forma como desenvolve e organiza o seu projecto.

Ideias para Projectos
um castelo
uma casa
um carro
um aeroplano
uma cadeira
uma ponte
uma grua
um escadote
um prédio
um autocarro
um comboio
uma escola
uma fábrica
um camião

Depois de terminar, faça estas perguntas a si próprio:

- Comecei por fazer um desenho do que eu queria fazer?
- Dei nome às diferentes partes do meu desenho?
- Escrevi um procedimento diferente para cada parte?
- Construí a solução global, juntando as pequenas partes?
- Conseguir editar um procedimento para modificar a sua definição?
- Entrei em pânico quando um dos meus procedimentos não produziu o resultado esperado?
- Consegui localizar e resolver os erros dos meus procedimentos?
- Guardei o meu trabalho, de forma a poder utilizá-lo em sessões seguinte?

Capítulo 5. Mais Procedimentos Logo

Agora vamos definir outros procedimentos e também aprender melhor como guardá-los em ficheiros. Recomendamos que defina todos os procedimentos no **Editor**. Assim, pode corrigir os erros, à medida que vão surgindo. Se algum procedimento não funcionar, como é provável que aconteça à primeira tentativa, é fácil escrever **edita** e modificá-lo.

Iremos agora escrever procedimentos Logo para reproduzir o estilo de números usados nas calculadoras antigas, todos desenhados com ângulos rectos. Por exemplo:

```
? edita "dois
```

No **Editor**, vai encontrar:

```
para dois

fim
```

Posicione o cursor, de modo a poder escrever o procedimento.

```
para dois
mt esq 90 av 80
repete 2 [dta 90 av 80]
repete 2 [esq 90 av 80] et
fim
```

Clique em **OK** e escreva:

```
? dois
```

Se o **dois** não for suficientemente grande, pode alterar o número de passos da tartaruga para ajustar a imagem, regressando ao **Editor** e alterando os números.

Aqui estão mais dois números:

```
para um
mt av 160 et
fim

para quatro
mt av 160 re 80 esq 90
av 80 dta 90 av 80 et
fim
```

Cada procedimento começa com **mt** (**mostratartaruga**) e termina com **et** (**escondetartaruga**). Isto ajuda quando pretende desenhar um número depois do outro.

Escreva:

```
? um le dois le quatro
```

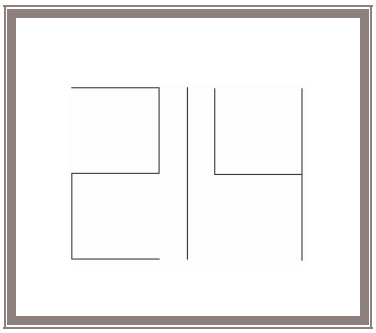
Isto acontece muito rapidamente e no ecrã aparece apenas o número final.

Se escrever:

```
? um
? dois
? quatro
```

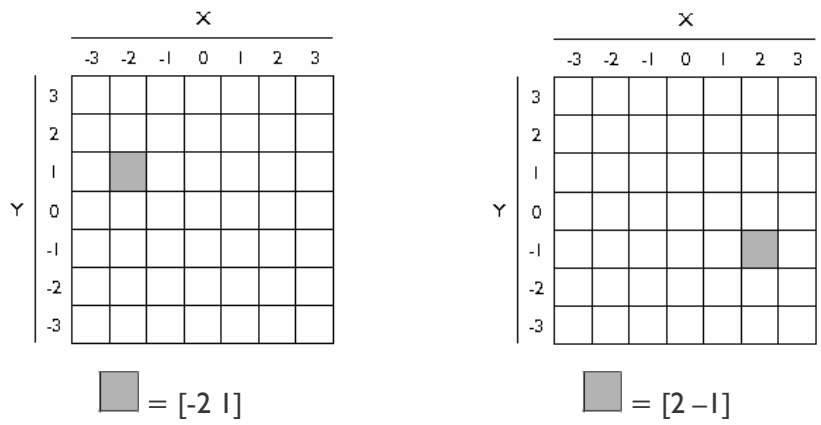
na expectativa de ver 214

ficará desapontado.

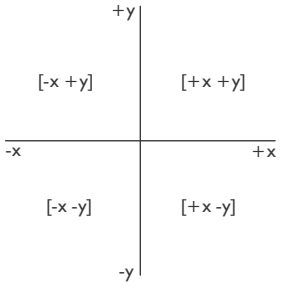


Para ultrapassar esta situação, é necessário que aprenda algo novo acerca do ecrã. Todas as posições no ecrã podem ser descritas como um par de números, semelhante a uma referência de mapa. O diagrama que se segue mostra, de uma forma simples, como qualquer pessoa pode descrever um quadrado numa matriz quadrada 7x7, 49, utilizando as coordenadas –3 para +3 nos eixos x e y. Por convenção e como acontece numa referência de mapa, a coordenada x é dada em primeiro lugar. É necessário então colocar as coordenadas entre parêntesis rectos, pois só dessa forma o Logo consegue juntá-las e lê-las como uma única referência de mapa.

Quando estamos a descrever localizações num ecrã de computador, devemos ser muito mais precisos do que o que é possível com uma matriz de 7x7. Uma semelhança é o facto do meio do ecrã possuir as coordenadas [0 0]. Neste ponto, é onde se encontra a tartaruga, depois de escrever **`centro`** ou **`le`**.



Se os sinais (+ ou -) nos diferentes cantos do ecrã lhe fazem confusão, o diagrama que se segue pode ajudá-lo.



Para posicionar a tartaruga em determinado ponto do ecrã, deve usar o comando **`fixapos`**, seguido de um par de coordenadas dentro de parêntesis rectos. Experimente:

```
? le fixapos [100 –50]
```

A linha desenhada pela tartaruga ao longo do ecrã pode interferir no efeito que tentava atingir. Levante a caneta da tartaruga antes de a movimentar, mas lembre-se de a baixar antes de lhe dar o comando seguinte.

```
? le lc fixapos [100 –50]
```

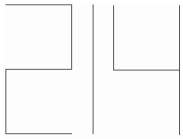
Experimente outras coordenadas. Altere ainda o ângulo da tartaruga.

Repare que a tartaruga não regressa à posição inicial, depois de um comando **`fixapos`**, mas continua voltada para o mesmo ângulo que estava antes de ser efectuado o movimento. Quando estiver a escrever procedimentos e a trabalhar na Área de Trabalho, precisa de ter isto em consideração.

Regressemos ao exemplo dos números da máquina de calcular. Escreva o seguinte:

```
? le um lc fixapos [-100 0] bc dois lc fixapos [200 0] dta 90 bc
quatro
```

O resultado que irá aparecer é:



Experimente com o tamanho do primeiro número entre parêntesis para conseguir o espaço certo entre os dígitos.

Continue a escrever procedimentos para todos os dígitos de 0 a 9.

Verifique se todos funcionam correctamente. No capítulo seguinte, vai utilizar estes procedimentos, portanto guarde-os para o futuro com o nome **números**.

Escreva

```
? grava "números
```

Capítulo 6. Palavras Mágicas do Logo

Algumas das ideias que já encontrou atrás são fundamentais em programação de computadores. A mais importante é, talvez, a noção de agrupar sequências de pequenas instruções em instruções maiores chamadas de procedimentos. Os comandos básicos do Logo, palavras como **avança**, **recua** e **repete**, são chamados de "primitivas", pois estas palavras são o ponto de partida de tudo o resto. Esta é a definição do dicionário; só recentemente é que o termo "primitiva" tem sido mais utilizado. As primitivas do Logo são colecções de instruções na linguagem do computador.

Cada procedimento é um programa de computador e já vimos atrás como é que eles podem ser armazenados e recuperados no início de cada sessão de trabalho. Até ao momento, temos visto maneiras de controlar a tartaruga, mas todas as ideias que já conheceu na primeira parte deste manual possuem aplicações muito mais vastas.

Vamos ver agora uma ideia que é fundamental para todas as linguagens de programação. Abra o ficheiro dos **números** na **Área de Trabalho**.

```
? abreprojecto "números
```

e depois escreva:

```
? edita "dois
```

```
para dois
mt esq 90 av 80
repete 2 [dta 90 av 80]
repete 2 [esq 90 av 80] et
fim
```

Se quiser que o dígito possua o dobro do tamanho, basta substituir cada **av 80** por **av 160**. No entanto, é ainda melhor substituir cada **80** por um símbolo, cujo valor pode ser variável. Para ver como isto pode funcionar, modifique o procedimento **dois** da seguinte maneira. Onde encontrar um número de passos, substitua o número por **:tamanho**, sem se esquecer de colocar os dois pontos **:** antes da palavra **tamanho**.

```
para dois :tamanho
mt esq 90 av :tamanho
repete 2 [dta 90 av :tamanho]
repete 2 [esq 90 av :tamanho] et
fim
```

Regresse à **Área de Trabalho** e experimente:

```
? le dois 60
? le dois 100
? le dois 150
```

em cada situação, está a dar ao **:tamanho** um valor diferente. Cada vez que o Logo encontra a palavra **:tamanho**, substitui-a pelo valor definido.

1. Dois Pontos e Aspas

O processo que acabámos de descrever não é difícil de compreender, mas talvez os dois pontos **:** tenham suscitado algumas dúvidas. Porquê **:tamanho**? Também já deve ter perguntado porquê as aspas **"** antes do nome dos ficheiros, por exemplo **"polígonos** ou **"números**. A linguagem Logo é mais livre do que a maioria das linguagens de programação, na medida em que permite utilizar palavras portuguesas. Por essa razão, os designers desta linguagem tiveram que encontrar uma boa forma de distinguir os diferentes tipos de palavras que o Logo poderia encontrar.

Cada palavra num programa Logo é percebida como o nome de uma primitiva ou o nome de um procedimento, **a menos que** tenha dois pontos **:** ou aspas **"** antes dela. Se o Logo encontrar uma palavra sem dois pontos ou aspas, que não reconhece nem como primitiva nem como procedimento, ele responde:

```
Não sei como fazer...
O procedimento não foi definido ou foi mal escrito
```

Se uma palavra for precedida de **"**, como em **"números**, o Logo sabe que está a lidar com um nome. Pode ser um nome de um procedimento ou o nome de um ficheiro. Nestes casos, não tem que procurar o significado do nome.

Se uma palavra é precedida de **:**, como em **:tamanho**, o Logo sabe que a palavra indica algo mais, e é-lhe pedido que olhe para o "algo mais". No Logo, chamamos a estas palavras (precedidas de dois pontos) de "variáveis". Elas são muito úteis e muito comuns. O seu significado ou valor pode facilmente ser alterado, como vimos no exemplo acima.

2. Utilizar as Variáveis

Pode agora voltar escrever todos os procedimentos do projecto **números**, substituindo os passos da tartaruga por uma "variável". No caso de **dois**, utilizámos **:tamanho**. Pode usar esta mesma palavra para todos os outros números, uma vez que o Logo não fica confuso. Para cada caso, poderia utilizar palavras diferentes, como **:comprimento** ou **:escala**. É mais sensato utilizar uma palavra que possua algum significado para si.

Faça as alterações nos procedimentos e depois guarde os novos procedimentos num novo projecto, por exemplo **números2**. Também poderia gravar por cima do ficheiro antigo; no entanto, é preferível manter sucessivas versões de um mesmo ficheiro. Deste modo, se achar que seguiu a direcção errada e quiser voltar a uma versão anterior do seu trabalho, pode fazê-lo.

2.1. Um Procedimento para Desenhar qualquer Polígono Regular

Abra novamente o projecto **"polígonos** e escreva:

```
? edita "hexágono
para hexágono
le repete 6 [av 100 dta 60]
fim
```

É necessário pensar no problema para escrevermos um procedimento que permita desenhar qualquer polígono. Se reparar num procedimento para desenhar um hexágono, é fácil perceber que é necessário variar o número de vezes que se repete **av 100** e o ângulo de rotação. Experimente:

```
para poli :lado :ângulo
repete :lado [av 100 dta :ângulo]
fim
```

Pode verificar se funciona ou não, experimentando:

```
? poli 5 72
```

3. Uma Primeira Abordagem à Aritmética do Logo

Um aspecto não satisfatório deste procedimento é o facto de ter que calcular o ângulo de rotação da tartaruga. É fácil se souber que a tartaruga tem que virar 72 graus para um pentágono e 45 graus para um octógono. Mas qual é o ângulo de rotação para uma figura com 10 lados. Os computadores devem ser usados para resolver este tipo de problemas...

Antes de deixar o computador fazer isto, deve descobrir como é que o LOGO faz aritmética simples. Se ainda é iniciante em computadores, necessita de saber que o asterisco (*) é utilizado como o símbolo da multiplicação e a barra (/) como o símbolo da divisão. Experimente o seguinte:

```
? escreve 3+4
7
? escreve 7*3
21
? escreve 6/2
3
? 7+5
Não sei o que fazer com 7
É necessário indicar o que fazer com o resultado
? escreve 7-2
5
```

O Logo pode ser utilizado da mesma forma que uma calculadora. A única vez que o Logo emitiu uma mensagem de erro foi quando omitimos a palavra **escreve**. Isto acontece porque o Logo espera que lhe digam o que fazer com o resultado de um cálculo. Ele pega numa expressão aritmética, avalia-a e passa o resultado para um comando Logo. Neste caso, utilizámos **escreve**, o que lhe indica simplesmente para mostrar o resultado no ecrã. O uso da palavra **escreve**, pode parecer estranho mas, para o computador, expor caracteres no ecrã é uma acção muito semelhante à acção de enviá-los para a impressora.

O Logo não utiliza os símbolos típicos utilizados nas salas de aula, o símbolo **x** para a multiplicação ou o símbolo ÷ para a divisão, porque quando foram introduzidas as máquinas de escrever, estas tinham um conjunto de caracteres muito restrito, que não incluía o símbolo ÷. O símbolo **x** não era utilizado para a multiplicação, pois poderia ser confundido com o x utilizado como uma variável em álgebra. É por esta razão que se utilizam os símbolos * e /.

Antes de regressarmos ao procedimento **poli**, experimente o seguinte:

```
? av 50*2 dta 90/2 av 50
? edita "hexágono
```

Encontra-se, de novo, no **Editor** com o procedimento **hexágono** no ecrã. Altere o **hexágono** para fazer o que se segue:

```
para hexágono
```

```
le repete 6 [av 100 dta 360/6]
fim
```

O ângulo de rotação para qualquer polígono regular é 360 dividido pelo seu número de lados. Isto pode dar-lhe uma ideia de como deve editar o polígono.

```
? edita "poli

para poli :num_lados
repete :num_lados [av 100 dta 360/:num_lados]
fim
```

Apenas precisa de saber o número de lados desejado. Experimente:

```
? poli 3
? poli 12
? poli 20
```

4. O_Carácter_Travessão

Pode suscitar-lhe alguma confusão o uso do carácter "travessão" _ entre a composição das palavras **:num_lados**. O nome de uma variável tem que ser uma só palavra. Se deixasse espaços, o Logo iria procurar novos significados para as restantes palavras. Ao mesmo tempo, é necessário que dê às suas variáveis nomes que reconheça e compreenda imediatamente. **:numlados** seria difícil de ler.

À medida que o número de lados aumenta, pode verificar que o respectivo polígono não cabe no ecrã. Regresse ao **Editor**. Vai necessitar de uma variável **:tamanho** para o lado do polígono. Assim:

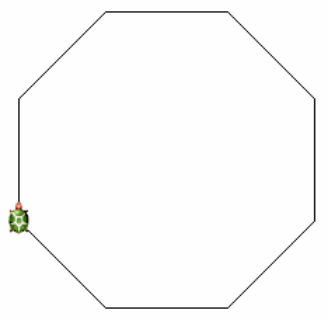
```
para poli :num_lados :tamanho
repete :num_lados [av :tamanho dta 360/:num_lados]
fim
```

Pode ainda fazer mais duas melhorias ao seu projecto. Por exemplo, reduzir gradualmente o **:tamanho** do lado, à medida que o número de lados aumenta.

Experimente substituir **av :tamanho** por **av :tamanho/:num_lados**.

Em seguida, experimente utilizar diferentes valores para **:tamanho**.

A segunda melhoria pode ser virar a tartaruga, de maneira a que o polígono tenha uma base horizontal, conforme a ilustração ao lado.



```
? poli 8 700
?
```

5. Círculos e Arcos

Quando esteve a desenhar o seu conjunto de dígitos, produziu números estilizados, uma vez que não tinha ainda trabalhado com arcos e círculos. Assumindo que passou algum tempo a tentar resolver o problema de modificar os lados do polígono. Experimente:

```
? poli 24 500
? poli 36 500
```

À medida que o número de lados aumenta, o polígono parece-se cada vez mais com um círculo. O que realmente deseja é um procedimento que defina um círculo pelo seu diâmetro **circ :diâmetro**.

Mesmo os que não são matemáticos lembram-se que a fórmula para calcular a circunferência de um círculo é Π (**Pi**) multiplicado pelo diâmetro do círculo. Para desenhar um círculo com determinado diâmetro, é necessário que a tartaruga dê 360 passos, cada um medindo **(pi * :diâmetro) / 360**, virando um grau à direita depois de cada passo.

Para evitar escrever o valor de Pi no seu procedimento, é necessário indicar o valor do Pi. No **Editor** escreva:

```
para pi
devolve 3.142
fim
```

Enquanto ainda está no **Editor** clique no botão **Adicionar** e escreva **circ**. Clique em **OK**.

Agora adicione o procedimento do círculo.

```
para circ :diâmetro
repete 360 [av pi * :diâmetro / 360 dta 1]
fim
```

Experimente:

```
? circ 100
? circ 200
```

Pode precisar de desenhar apenas uma parte de um círculo. Adicione os seguintes procedimentos **arco_direita** e **arco_esquerda**.

```
para arco_direita :raio :ângulo
repete :ângulo [av (2 * pi * :raio) / 360 dta 1]
fim
```

Experimente:

```
? arco_direita 100 50
? arco_direita 100 100
```

O **arco_esquerda** é idêntico ao anterior, com excepção da rotação que é **esq 1** em vez de **dta 1**. Experimente esta versão menos perfeita para o arco:

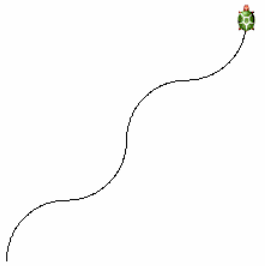
```
para arco_esquerda :raio :ângulo
repete :ângulo / 6 [av (2 * pi * :raio) / 360 esq 6]
fim
```

Guarde os procedimentos do **círculo** e dos **arcos** num projecto chamado **curvas**. Vai precisar deles mais tarde.

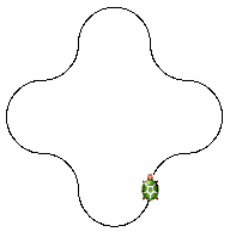
Agora, faça experiências com círculos e arcos. Por exemplo:

```
? lc fixapos [-300 0] bc repete 2
[arco_direita 50 180 arco_esquerda 50 180]
```

```
? lc fixapos [-300 0] bc repete 2 [arco_direita
50 90 arco_esquerda 50 90]
```



```
? lc fixapos [0 0] bc repete 4 [arco_direita 30
90 arco_esquerda 30 180]
```



6. Utilizar Procedimentos dentro de outros Procedimentos

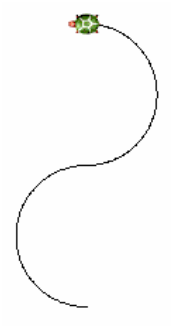
Depois da última secção, deve estar a perguntar se esperamos que crianças novas percebam o conceito de Π. Nós não contamos que as crianças aprendam a linguagem Logo a partir de um livro. Normalmente, as crianças trabalham com adultos que lhes são próximos, e se esse adulto não teve anteriormente nenhuma experiência de Logo, um livro como este pode ser uma ajuda. É fácil ocultar o Π do alcance das crianças, bem como os conceitos de "diâmetro" e "raio".

Abra o projecto **curvas**.

Vá ao **Editor** e crie novos procedimentos:

```
para meio_circ :tamanho
arco_direita :tamanho 180
fim

para curva :vezes :tamanho
esq 90
repete :vezes [arco_direita :tamanho 180 arco_esquerda :tamanho 180]
fim

para roda_grande
circ 100
fim

para roda_pequena
circ 25
fim

para rodas :tamanho
circ :tamanho
circ :tamanho*2
fim
```

Um procedimento Logo pode utilizar outros procedimentos, bem como comandos primitivos do Logo. Esta é única diferença entre os procedimentos que definimos acima e os que criámos anteriormente. Pode construir estes procedimentos, alterá-los e melhorá-los. Eles ilustram bem o ponto a partir do qual as crianças podem começar a trabalhar com círculos e arcos, sem precisarem de compreenderem o Π.



O procedimento **curva** contém **arco_direita** e **arco_esquerda** como "sub-procedimentos".

O procedimento **curva** é um "super-procedimento", que "chama" os dois "sub-procedimentos".

Repare no modo como a entrada **:tamanho** de **curva** passa o seu valor automaticamente para os sub-procedimentos **arco_direita** e **arco_esquerda**, que também possuem **:tamanho** como entrada.

Por exemplo, compare **curva 1 50** e **curva 5 10**.

7. CoorX, CoorY, Rumo e Posição

Todas as primitivas do Logo que utilizámos até agora são comandos. Tem estado a dizer ao computador para fazer coisas. Em geral, os comandos têm sido direccionados para a tartaruga, mas existem outros que se dirigem ao computador, como **escreve**, **le** e **edita**.

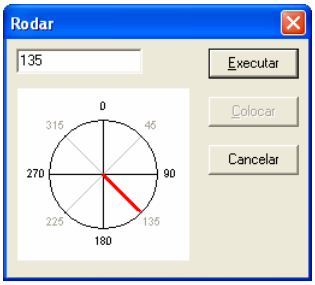
Por vezes, pode precisar de informação do computador. Analise esta informação, em primeiro lugar, em relação à tartaruga. Experimente:

```
? le esq 90 av 100
? escreve coorx
-100
? escreve coory
0
? escreve rumo
270
? dta 30
? escreve coory escreve rumo
0
300
? le
? escreve pos
0 0
```

Estivemos a trabalhar os significados de **coorx**, **coory**, **rumo** e **pos**.

Estes comandos fornecem-lhe a informação suficiente para identificar a posição exacta da tartaruga no ecrã e planear o passo seguinte.

8. FixaRumo



O comando **fixarumo** permite virar a tartaruga para determinada posição. Isto pode ser muito útil em diferentes circunstâncias.

A circunferência considera sempre o topo do ecrã como estando voltado para o **rumo 0** e o comando **fixarumo** permite dirigir a tartaruga para qualquer um dos seus pontos.

Experimente:

```
? le fixarumo 30 av 250 fixarumo 180 av 100
```

Capítulo 7. Cores e Efeitos

O seu computador é capaz de disponibilizar gráficos de cores altamente sofisticados. Tem ao seu dispor uma multidão de cores para escolher.

1. FixaCorCaneta, FixaFundo

Os comandos que se seguem são **`fixacorcaneta`** e **`fixafundo`**.

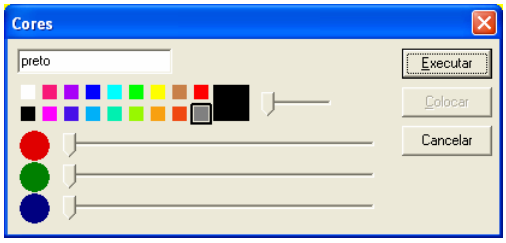
Veja os três métodos de abordagem que lhe propomos.

2. Método 1

Escreva

```
? fixacorcaneta
```

e carregue na tecla **`<Enter>`**.



Aparece no ecrã o selector das cores:

Seleccione uma cor e tente mover o cursor de deslocamento. A cor na caixa à direita altera-se e o nome da cor é modificado. Quando estiver satisfeito com a cor, clique no botão **Executar**. Agora, desloque a tartaruga e veja o resultado.

Modifique o fundo do mesmo modo:

```
? fixafundo
```

3. Método 2

Se escrever

`? fixacorcaneta` e a seguir pressionar a tecla **`<F9>`**

aparece o mesmo selector. No entanto, o botão **Colocar** deixa de aparecer a cinzento e pode ser pressionado. Permite inserir o nome da cor no seu texto. Este método tem de ser utilizado quando estiver a trabalhar no **Editor**.

4. Método 3

Pode definir as cores, utilizando o nome delas. Não se esqueça de que o nome tem que ser precedido de **"**, para que o Logo as reconheça.

```
para caixa
fixacorcaneta "vermelho
repete 4 [avança 200 direita 90]
direita 45
levantacaneta avança 100 baixacaneta
fixacorcaneta "laranja
pinta
fim
```

Consegue ver o que faz o comando **`pinta`**? Veja o que acontece se mover a tartaruga para fora do quadrado. Altere a cor da caneta e escreva **`pinta`**, novamente.

Não se esqueça de levantar a caneta, antes de mover a tartaruga, e de a baixar novamente, antes de utilizar o comando **`pinta`**.

- **`escreve corcaneta`** devolve a cor da caneta actual da tartaruga.
- **`escreve corfundo`** devolve a cor de fundo actual.

5. Um Projecto

Será que consegue criar um procedimento para dar cor ao ecrã? Tente criar um vaso azul, com um ramo de flores amarelas, vermelhas e laranjas sobre um fundo claro.

Planeie o seu projecto da mesma forma que planeou o **foguetão**, anteriormente.

Faça um rascunho de como deseja que o seu desenho seja no final. Dê um nome às diferentes partes.

Experimente e escreva um procedimento para cada parte. Experimente um de cada vez e depois utilize procedimentos para o construir.

Pode gravar o ecrã, em vez de guardar os procedimentos.

Escreva:

```
? gravacenário
```

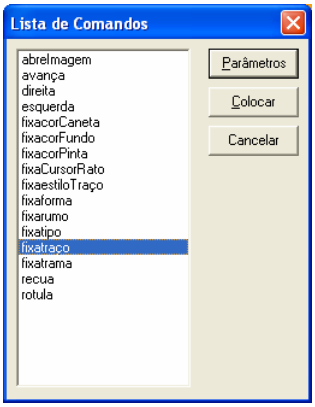
Dê ao seu desenho um nome adequado. Por exemplo, **flores**.

Pode abri-lo mais tarde, escrevendo:

```
? abrecenário "flores
```

5.1. Efeitos da Caneta

Tal como é possível alterar a cor da caneta da tartaruga e a cor do fundo, também podemos alterar a largura da ponta da caneta, os estilos de linha, a cor de preenchimento e o padrão de preenchimento.



Carregue na tecla de função **<F9>**.

Aparece o seguinte selector:

Escolha **fixatraço** e clique no botão **Parâmetros**. Aparece uma variedade de opções. Faça a sua selecção e depois clique no botão **Executar**.

Escreva:

```
? avança 100
```

Experimente várias grossuras do traço e, em seguida, seleccione **fixatrama**.

Se não se lembrar das definições dos comandos, pode perguntar:

? escreve traço

? escreve trama etc.

Quando já estiver à vontade com estes efeitos, pode experimentá-los com **fixatrama** e **pinta**. Depressa consegue perceber que a tartaruga, para conseguir produzir este efeito, tem de estar dentro de um contorno ou forma, sem tocar em qualquer linha, e que todas as linhas devem ser contínuas. De outro modo, o preenchimento sai para fora da área supostamente limitada.

Em alternativa, pode utilizar os comandos:

fixatraço	fixatraço número
fixaestilotraço	fixaestilotraço número
fixatrama	fixatrama número
fixacorpinta	fixacorpinta “nome

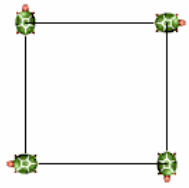
Capítulo 8. Formas e Animações

A pequena tartaruga verde que vê no ecrã é um objecto muito complexo e poderoso. Para além de possuir uma caneta que permite desenhar, contém ainda informação sobre a sua posição actual, rumo, direcção, cor da caneta, grossura e padrão do desenho. A sua forma actual é também uma das peças de informação que ela contém.

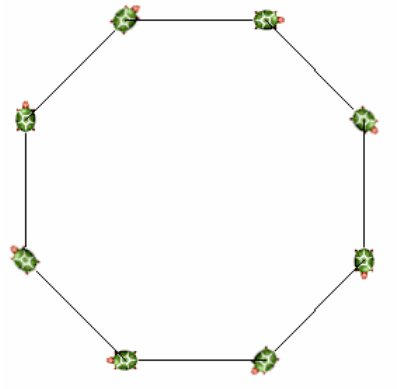
Podemos analisar a forma, rumo e rotação da tartaruga com a ajuda da primitiva **carimba**.

Edite ou redefina o procedimento **poli**:

```
para poli :num_lados
repete :num_lados [av 100 carimba dta 360/:num_lados]
fim
```



? poli 4 et



? le poli 8 et

Pode verificar que a forma da tartaruga roda de modo a mostrar sempre a sua direcção actual. Esta característica da tartaruga é muito útil, pois ajuda a reforçar o conceito de rodar segundo determinado ângulo.

1. Utilizar outras Formas para a Tartaruga

O Imagina oferece uma variedade de formas para a tartaruga. Para definir a forma da sua tartaruga escreva:

```
? le mt
? fixaforma
```

e carregue na tecla **<Enter>**.

Aparece uma caixa de diálogo, com diversas pastas. Escolha uma nova forma para a sua tartaruga.

Pesquise as várias opções e seleccione uma, clicando nela. Do lado direito da caixa de diálogo, pode pré-visualizar a sua selecção e, se estiver satisfeito, clique em no botão **OK**.

Escreva:

```
? le mt poli 5 et
```

Pode verificar que algumas formas não indicam a direcção, apenas mostram a sua imagem. Experimente outras formas para a tartaruga. Também pode utilizar tartarugas animadas.

Escreva:

```
? fixaforma
```

Aparece a mesma caixa de diálogo. Desta vez, seleccione uma forma animada.

```
? le mt poli 5 et
```

Nalguns casos, o resultado pode ser confuso. Contudo, consegue obter resultados bastante animados, dependendo da sua escolha. Escolha outra imagem.

```
? le lc poli 7
```

Pode construir ecrãs muito interessantes, usando as formas animadas e que podem ser guardadas utilizando `gravacenário` ou seleccionando a opção **Gravar Cenário** do menu **Página**.

2. Utilizar o AnimaLogo

O Imagina possui uma colecção de formas interessantes para experimentar. Algumas têm uma ou várias rotações, enquanto outras são animadas. Pode, de qualquer forma, criar as suas próprias formas no **AnimaLogo**, que é um editor de animação.

Ao instalar o Imagina, também é instalado o **AnimaLogo**, com o qual pode experimentar editar ou criar as suas próprias formas de tartaruga.

3. Criar Formas com a Linguagem Logo

Pode utilizar também a linguagem Logo para produzir uma nova forma de tartaruga. Experimente o seguinte procedimento:

```
para triângulo
esq 90
fixacorcaneta "vermelho fixatraço 3
repete 3 [av 15 dta 120 av 15]
dta 90
fim

? triângulo
```

Se quiser usar esta imagem como uma forma para a tartaruga, é necessário adicionar o comando **fixaforma** ao procedimento, e colocar todas as outras instruções entre parêntesis.

```
para triângulo
fixaforma
[esq 90
fixacorcaneta "vermelho fixatraço 3
repete 3 [av 15 dta 120 av 15]]
dta 90
fim

? fixacorcaneta "preto fixatraço 1
? triângulo bc poli 6 et
```

Como pode verificar, o triângulo vermelho é agora a nova forma da tartaruga e, tal como a tartaruga original, ele roda para mostrar a sua posição. Pode ainda adicionar um outro comando ao procedimento **triângulo**.

```
para triângulo
```

```
fixaforma
[esq 90
fixacorcaneta "vermelho fixatraço 3
repete 3 [av 15 dta 120 av 15]
dta 90
fixacorcaneta "azul marca 11]
fim

? triângulo
```

No triângulo, podemos ver a posição do bico da caneta. Uma outra forma que pode experimentar é a "seta".

```
para seta
fixaforma
[fixacorcaneta "azul
esq 90 av 15
dta 90 av 10
esq 90 av 7.5
repete 2 [dta 120 av 45]
dta 120 av 7.5
esq 90 av 10
dta 90 av 15]
fim

? seta
```

A forma **triângulo** da tartaruga é transparente. Se desejar formas preenchidas deve utilizar a primitiva polígono. Se acrescentar **fixacorpinta "nome_cor** aos comandos do **polígono**, pode obter a linha exterior de uma cor e o interior de outra cor.

```
para triângulo
fixaforma
[fixacorcaneta "amarelo fixatraço 2
```

```
polígono [esq 90 repete 3 [av 15 dta 120 av 15] dta 90]]
fim

? triângulo
```

Experimente outro

```
para hélice
fixaforma
[fixatraço 2 fixacorcaneta "castanho
repete 4 [av 55 re 44
polígono
[repete 2 [av 40 dta 90 av 10 dta 90]]
re 10 dta 90]]
fim
```

Não se esqueça de que, se desejar definir uma cor de preenchimento diferente da cor da caneta, deve usar **fixacorpinta** depois de especificar **fixacorcaneta**.

```
? hélice
? ciclo [dta 1 espera 10]
```

Para terminar o ciclo clique no botão da barra de ferramentas: **Parar Linha de Comandos**.

Capítulo 9. SeEntão e Se

Até aqui, estivemos a trabalhar com sequências lineares de comandos. Primeiro fazer isto, e depois aquilo; mas os computadores também podem lidar com possibilidades múltiplas. Fazer primeiro isto e depois, dependendo do resultado da primeira acção, fazer isto ou aquilo. É agora altura de explorar esta funcionalidade do Logo, relacionada com um caso em particular, que se repete para sempre, a menos que seja interrompido. Estes programas são mais comuns do que o que pode pensar. Experimente o seguinte:

```
para quadrado :lado
av :lado
dta 90
quadrado :lado
fim


? quadrado 100
```

Este procedimento é efectuado indefinidamente, a menos que carregue num dos botões **Parar todos** ou **Pausa todos/continuar** da barra de ferramentas.

O procedimento **quadrado** é um sub-procedimento ou um super-procedimento? É tanto um como outro. Antes de chegar ao fim de **quadrado**, ele já começou outra vez *ad infinitum*. Chamamos a isto "recursão". Mais à frente, vai reconhecer a "recursão" como uma das técnicas de programação mais poderosas da linguagem Logo.

Considere o seguinte problema: desenhar uma figura que consiste em quadrados encaixados, cada um com o lado 10 passos menor do que o seu predecessor.

Isto pode ser feito do seguinte modo:

```
para quadrado :lado
repete 4 [av :lado esq 90]
fim
```

```
para quadrados :lado
quadrado :lado
quadrado :lado – 10
quadrado :lado – 20
quadrado :lado – 30
fim
```

Pode ficar melhor se for escrito desta maneira:

```
para quadrados :lado
quadrado :lado
quadrados :lado – 10
fim
```

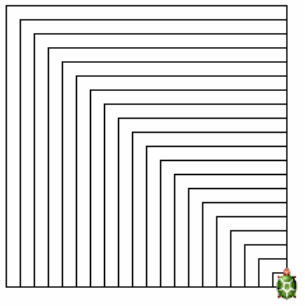
Se experimentar com:

```
? quadrados 50
```

Se calhar, não era exactamente o que tinha planeado. Então, clique rapidamente no botão **Parar todos** da barra de ferramentas. O problema é que **avança –5** é equivalente a **recua 5**. A questão que se coloca é como parar o programa, antes que o valor do **:lado** se torne negativo? Isto leva-nos ao conceito de **seentão**.

Regresse ao **Editor** e modifique o procedimento **quadrados**:

```
para quadrados :lado
seentão :lado < 10 [pára] [quadrado :lado]
quadrados :lado – 10
fim
```



Se tentar agora

```
? lc fixapos [–100 –100] bc quadrados 200
```

pode verifica que a tartaruga pára no fundo do canto esquerdo do quadrado original.

1. O Mecanismo de Paragem

É necessário que olhe para o mecanismo de paragem com mais pormenor. Tudo acontece na segunda linha do procedimento. Se o escreveu por extenso, pode lê-lo da seguinte maneira:

"*Se o lado do quadrado for menor que 10, executa as instruções do primeiro conjunto de parêntesis rectos; se não, executa as instruções do segundo conjunto de parêntesis rectos.*"

Assim que o comprimento do **:lado** for um número inferior a 10, o procedimento **pára**. Até que se atinja este ponto, o Logo desenha outro quadrado com o **:lado – 10**.

seentão ***condição*** **[isto] [aquilo]**

O comando **seentão** é seguido de uma expressão que o computador está apto a interpretar como **verdadeira** ou **falsa**. Se o computador decidir que a expressão é **verdadeira**, então ele segue as instruções que estão dentro do primeiro par de parêntesis; se decidir que é **falsa**, ele segue as instruções contidas no segundo conjunto de parêntesis. Pode perguntar-se que tipo de expressão é aceitável.

Experimente:

```
para quadrados :lado
seentão (2 + 2) = 5 [pára] [quadrado :lado]
quadrados :lado-10
fim
```

O computador aceita isto, porque **2+2 = 5** nunca pode ser **verdadeiro**.

Se, no entanto, tivesse escrito **2+2 = 4**, a tartaruga não iria mover-se, por ser sempre uma resposta verdadeira.

Pode ficar uma ideia mais clara de como este mecanismo funciona, se fizer o seguinte:

```
? escreve (4 * 3) = 13
falso
? escreve (9 + 8) > (2 * 8)
verdadeiro
? escreve (3 + 5 + 8) = (4 * 4)
verdadeiro
? escreve (12 – 4) > 7
verdadeiro
```

2. Aleatório e ParaSempre

Antes de realizar o seu projecto seguinte, precisa de conhecer mais duas primitivas: **aleatório** e **parasempre**.

Uma vez mais, é mais fácil explicar através de um exemplo:

```
? escreve aleatório 5
? repete 8 [escreve aleatório 10]
```

aleatório <n> gera um número, escolhido ao acaso, entre **0** e **<n>-1**.

aleatório 5 escolhe um número entre 0, 1, 2, 3, 4. **aleatório 5** nunca vai produzir um número maior do que 4, porque os computadores começam sempre a contar a partir do 0 e não do 1; assim, existem 5 possibilidades: 0 ou 1 ou 2 ou 3 ou 4.

Se quiser que o número mais pequeno seja o 1, tem de acrescentar sempre 1 ao resultado, como no exemplo que se segue:

```
para lança_dado
escreve 1 + aleatório 6
fim
```

Isto vai simular o lançamento de um dado. Pode perguntar-se: porquê **1 + aleatório 6**. Um dado não possui o número 0. Os números de um dado são de 1 a 6, e não de 0 a 5.

Se quiser qualquer número entre 50 e 100, escreva:

```
? escreve 50 + aleatório 51
```

Isto não produz o mesmo resultado que:

```
? escreve aleatório 51 + 50
```

O último exemplo pode gerar o número 23, uma vez que o comando **aleatório** possui o número 101 como limite superior, calculando 51+50=101 antes de procurar por **aleatório**.

3. Movimento Aleatório

Se pensar na tartaruga como uma criatura verdadeira, pode tentar modelar a forma como ela se movimenta como uma colecção de simples procedimentos. O tipo de movimento mais fácil de modelar talvez seja o **movimento aleatório**, no qual a tartaruga avança determinada distância ao acaso, roda num ângulo ao acaso.

Pode começar a modelar este comportamento seguindo o procedimento:

```
para corrida :distância :ângulo
parasempre [esquerda aleatório :ângulo avança aleatório :distância]
fim
```

Experimente:

```
? corrida 30 50
```

Se deixar continuar o procedimento, o ecrã fica todo preenchido; portanto, clique no botão **Parar todos** da barra de ferramentas.

4. Projecto Recursão

Atrás começámos a ver um "procedimento recursivo" que se referia a ele próprio na última linha, criando um ciclo interminável. Também vimos, no procedimento **quadrados**, como podíamos definir uma condição, que iria parar o ciclo. Repare novamente no procedimento **quadrados**:

```
para quadrados :lado
seentão (2 + 2) < 5 [pára] [quadrado :lado]
quadrados :lado – 5
fim
```

O que acontece se trocar a segunda e a terceira linha?

```
para quadrados :lado
quadrados :lado - 5
seentão (2 + 2) < 5 [pára] [quadrado :lado]
fim
```

Quando o testar, recebe uma mensagem de *Demasiados processos encadeados*. O procedimento nunca alcançou a terceira linha. Ele simplesmente anda às voltas, sem fazer absolutamente nada, até ter ocupado toda a memória do computador.

Escreva o procedimento mais uma vez. Desta vez, está a pedir unicamente ao computador para ler uma condição. Apenas necessita de escrever **se** e ter um par de **[ ]** na linha 2.

```
para quadrados :lado
se :lado < 5 [pára]
quadrados :lado - 5
quadrado :lado
fim
```

O quadrado vai crescer, em vez de encolher, porque até o **:lado** ser inferior a 5 passos de tartaruga, o Logo não alcança a última linha.

Pode encontrar outros procedimentos **conta_menos** e **conta_mais** mais fáceis de seguir. O princípio é idêntico.

```
para conta_menos :número
se :número = 0 [pára]
es :número
conta_menos :número - 1
fim
```

```
? conta_menos 5
5
4
3
2
1


para conta_mais :número
se :número = 0 [pára]
conta_mais :número – 1
es :número
fim
? conta_mais 5
1
2
3
4
5
```

No procedimento **conta_menos**, escrevemos um número depois do título. A segunda linha verifica se o número é um zero e, se for, pára o programa. Se não for, ele escreve o número indicado e reinicia o processo subtraindo 1 ao número indicado.

Isto acontece muito rapidamente. Para ser mais fácil seguir o processo, pode abrandar o ritmo, pedindo ao computador para esperar antes de escrever cada número.

```
para conta_menos :número
se :número = 0 [pára]
es :número espera 500
conta_menos :número – 1
fim


para conta_mais :número
se :número = 0 [pára]
```

```
conta_mais :número - 1
es :número espera 500
fim
```

Espera é um comando muito útil para abrandar qualquer procedimento. **Espera 1000** é igual a esperar 1 segundo, pelo que pode decidir quanto tempo deve esperar em cada ciclo.

O procedimento **conta_menos** também verifica se o número é diferente de zero. Mas, desta vez, a chamada recursiva vem antes da instrução para escrever o número. Assim, o Logo conta, de forma decrescente, da mesma maneira que o faz em **conta_mais**, mas não escreve os resultados à medida que são gerados.

Assim que o valor de **:número** alcançar o zero, o Logo termina os assuntos pendentes e escreve os números que tem estado a acumular. Se o último número a ser guardado tiver sido o 1, é o primeiro a ser escrito, o que indica que o programa parece contar por ordem crescente.

Algumas pessoas consideram este tipo de recursão difícil de conceber. Lembre-se que o computador trabalhou-a na ordem correcta, mas depois teve que os mostrar do último ao primeiro.

Um exemplo utilizado por Douglas Hofstadter para explicar a recursão (em *Godel, Escher, Bach: an eternal golden braid*) é imaginar um executivo muito ocupado a atender o telefone. Antes da chamada acabar, entra uma segunda, e coloca a outra em espera; antes da segunda estar completa, uma terceira irrompe, e a segunda fica em espera juntamente com a primeira.... e uma quarta.... e uma quinta chamada junta-se à fila. O executivo termina de falar com a sexta pessoa, e depois vai atravessando todo o conjunto de chamadas, fechando cada uma na sua vez, mas das mais recentes para as mais antigas (por ordem decrescente). A última chamada a ficar completa é a conversa com a primeira pessoa que ligou, e que pode muito bem ter-se aborrecido de esperar e desligou. Mas felizmente para nós, os computadores nunca se aborrecem. Cada chamada recursiva espera pacientemente pela sua vez na fila.

Um outro exemplo de um procedimento recursivo é **torre :tamanho** que envolve dois sub-procedimentos, **passo1** e **passo2.** Comporta-se como o procedimento **conta_mais,** tendo que esperar até terminar de executar, antes de executar o **passo2**. Dentro de si, armazena um **passo2** para cada vez que o programa o chama, e finalmente descarrega a reserva de mensagens para a tartaruga, desenhando o lado direito da torre.

```
para torre :tamanho
se :tamanho < 10 [pára]
```

```
passo1 :tamanho
torre :tamanho * 0.75
passo2 :tamanho
fim

para passo1 :tamanho
av :tamanho dta 90 av :tamanho esq 90
fim

para passo2 :tamanho
dta 90 av :tamanho esq 90 re :tamanho
fim
? torre 50
? repete 4 [torre 50]
```

A importância dos assuntos abordados neste capítulo não deve ser exagerada. Eles são a base para muitos conceitos que vão ser tratados a seguir. Grave os procedimentos **conta_mais**, **conta_menos**, **quadrados** e **torre** num projecto com o nome **recursão**. Pode precisar deles mais tarde.

Capítulo 10. Múltiplas Tartarugas

O Imagina permite trabalhar com muitas tartarugas no ecrã ao mesmo tempo. No exemplo que se segue, vai desenvolver um projecto simples, utilizando múltiplas tartarugas – neste caso, vão ser quatro cães que se perseguem uns aos outros.

Comece por escolher uma forma animada apropriada para a sua tartaruga:

Crie um procedimento que defina as propriedades da tartaruga. A tartaruga tem a forma de um cão e a sua principal característica é mover-se do lado esquerdo do ecrã para o direito e novamente ao contrário, isto é, ao atingir o limite do ecrã, a tartaruga volta para trás:

```
para prepara1
fixaforma "animais\\cão
fixarumo 90
levantacaneta
fixamodomundo "ricochete
fim
? prepara1
```

Este procedimento aplica a forma da tartaruga no gráfico escolhido, vira a tartaruga para Este, levanta a caneta e define o tipo de movimento, para que a tartaruga volte para trás, sempre que atinge a extremidade do ecrã.

Experimente a nova tartaruga:

```
? ciclo [avança aleatório 6 espera 50]
```

Esta operação aleatória faz com que o cão não se mova de forma previsível.

Para parar o movimento do cão, clique no botão **Parar Linha de Comandos** da barra de ferramentas principal.

Crie a segunda tartaruga:

```
? novo "tartaruga [pos [0 100]]
```

Isto cria um novo objecto do tipo tartaruga na posição [0 100] no ecrã. Imagine que quer que esta tartaruga tenha também a forma de cão e que caminhe de forma aleatória, tal como a primeira tartaruga. O procedimento **prepara1** (que foi criado para a tartaruga 1) inclui, portanto, todas as instruções que queremos para a nova tartaruga (t2). Agora é necessário que diga à t2 para se comportar da mesma maneira:

```
? pede "t2 [prepara1]
```

Agora tem duas tartarugas. Para fazer com que as duas se mexam, escreva:

```
? ciclo [avança aleatório 6 espera 50]
```

Algo não está bem; só o primeiro cão parece mexer-se. É preciso dizer à tartaruga t2 para se mover também. Experimente:

```
? ciclo [pede [t1 t2] [av aleatório 6] espera 50]
```

Assim está melhor. Repare que teve que listar os nomes da tartaruga entre parêntesis rectos **[ ]**.

Se tivesse um projecto com muitas tartarugas e com muitos nomes, podia usar a operação **todas** para obter uma lista com os nomes de todas as tartarugas do projecto.

```
? apresenta todas
[t1 t2]
```

Temos apenas duas tartarugas, mas já pode ver como seria mais fácil se todas as tartarugas pudessem ser activadas ao mesmo tempo.

Escreva o seguinte:

```
? ciclo [pede todas [av aleatório 6] espera 50]
```

Este comando faz com que todas as tartarugas activas avancem, no máximo, seis passos de cada vez. Passado algum tempo, pode verificar que ambas (as tartarugas) se movimentam exactamente à mesma velocidade. Para conseguir ver melhor, coloque ambos os cães (tartarugas) na mesma posição inicial no ecrã.

```
? pede todas [fixacoorx 0 fixarumo 90]
? ciclo [pede todas [av aleatório 6] espera 50]
```

As tartarugas movimentam-se à mesma velocidade, porque **av aleatório 6** é calculado e utilizado com uma entrada para ambas (**todas**) tartarugas. Para este exemplo, é necessário separar as declarações para cada tartaruga, de forma a que cada uma possua uma única entrada aleatória.

Experimente:

```
? ciclo [pede "t1 [av aleatório 6] pede "t2 [av aleatório 6] espera 50]
```

Assim está melhor. Os cães agora mexem-se a velocidades diferentes.

Este comando pode ser simplificado da seguinte forma, com o uso da apóstrofe em cada comando:

```
? ciclo [t1'av aleatório 6 t2'av aleatório 6 espera 50]
```

No entanto, apenas pode usar a apóstrofe num comando com o nome de uma tartaruga. Não pode usar **todas'avança 100** por exemplo, uma vez que **todas** não é o nome de uma tartaruga.

Agora, vamos acrescentar duas novas tartarugas (cães), que se movem aleatoriamente no ecrã e completar esta actividade, de forma a que todas as tartarugas pareçam perseguir-se umas às outras.

Para isso, vai precisar do comando **cada** para dar um movimento sequencial a todas as tartarugas.

```
? ciclo [pede todas [cada [av aleatório 6]] espera 50]
```

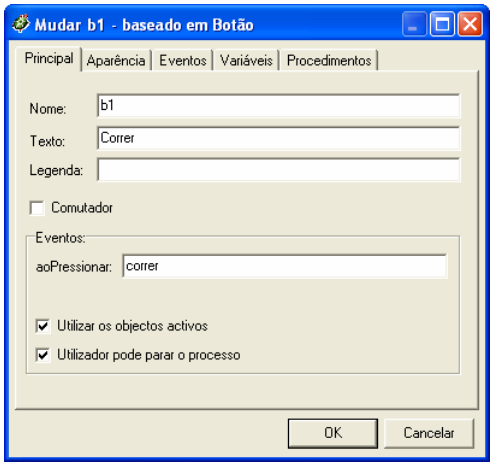
Agora, já não faz diferença se temos muitas tartarugas no ecrã ou não. Pode-se pedir a **cada** uma que se comporte de forma aleatória.

Podemos fazer o seguinte procedimento:

```
para correr
ciclo [pede todas [cada [av aleatório 6]] espera 50]
fim
```

Podemos acrescentar um botão para esta actividade, no ecrã, ao qual podíamos atribuir a acção:

- quando se carrega no botão, os cães (tartarugas) começam a mexer-se;
- quando se solta o botão, os cães (tartarugas) param.



Clique na ferramenta **Novo Botão** da barra de ferramentas e posicione-o no canto inferior direito do ecrã.

Clique no botão (**b1**) com o botão direito do rato e escolha a opção **Mudar b1**:

- Dê um título ao botão. Neste caso, o texto pode ser **`correr`**;
- Dê uma acção ao botão. Neste caso, **`aoPressionar`** pode ser correr (o procedimento definido anteriormente).

Finalmente, vamos acrescentar mais tartarugas (cães).

Em seguida, vamos desenvolver o procedimento **`prepara`**, que vai apagar todas as tartarugas e depois criar mais quatro com características de "perseguição", isto é, vai chamar o procedimento **`prepara1`**.

```
para prepara
eliminaobjecto todas
novo "tartaruga [pos [0 100]]
novo "tartaruga [pos [0 0]]
novo "tartaruga [pos [0 -100]]
novo "tartaruga [pos [0 -200]]
pede todas [prepara1]
fim
```

Elimina todos os objectos, isto é, todas as tartarugas.

Quatro novas tartarugas criadas em diferentes posições.

Todas as tartarugas são transformadas em cães de perseguição.

Para sintetizar o que acabámos de aprender:

- Com o comando **`novo "tartaruga [pos [x y]]`**, pode produzir um número de tartarugas arbitrário, cada uma delas numa nova posição;
- Com o comando **`pede "tn [comandos]`**, pode dar um conjunto de comandos a cada tartaruga;
- Pode fornecer um comando a múltiplas tartarugas, utilizando **`pede [t1 t2 t3] [comandos]`** ou **`pede todas [comandos]`**;
- Para criar comandos aleatórios ou individuais, pode usar **`pede todas [cada [comandos]]`**;
- O comando **`eliminaobjecto todas`** remove todas as tartarugas, enquanto que **`eliminaobjecto "t2`** apenas remove a tartaruga mencionada (t2).

Vamos agora introduzir dois novos comandos:

- O comando **`activa`** informa qual ou quais as tartarugas que vão obedecer aos comandos para todas as tartarugas. O Imagina começa sempre com **`activa "t1`**.
- A operação **`quais`** indica as tartarugas que estão activas em determinado momento. Por exemplo, **`apresenta quais`** mostra o nome da tartaruga activa ou a lista de todas as tartarugas activas.

1. Tarefas Avançadas para Múltiplas Tartarugas

No exemplo que se segue vamos desenvolver um projecto mais avançado, utilizando múltiplas tartarugas – neste caso várias tartarugas (t1, t2, t3, etc.) que correm em direcção ao seu sucessor (t1 em direcção a t2, t2 em direcção a t3, etc.). Cada tartaruga vai desenhar uma linha com uma cor aleatória, à medida que se vai aproximando do seu sucessor a 1/20 de distância entre si próprio e o seu sucessor. Todas as tartarugas vão mover-se, até que estejam tão perto que já não se possam movimentar mais.

Esta tarefa levanta alguns problemas interessantes:

- Como criar uma tartaruga em qualquer posição da página
- Como definir uma cor aleatória para a caneta de cada tartaruga
- Como encontrar o sucessor de cada tartaruga e como virar uma tartaruga para ele
- Como medir a distância até outra tartaruga, isto é, como calcular a distância entre quaisquer pontos na página

Em primeiro lugar, crie a tartaruga:

```
? novo "tartaruga [posição qualquer corcaneta qualquer]
```

Isto cria uma nova tartaruga em **qualquer** posição no ecrã e com **qualquer** cor da caneta (**corcaneta**).

Escreva:

```
? apresenta t2'corcaneta
```

para descobrir a cor da caneta (**corcaneta**) seleccionada para a tartaruga.

Uma tartaruga não é suficiente. Utilize o comando **aleatório** para gerar mais tartarugas:

```
? repete 4 + aleatório 6 [novo "tartaruga [pos qualquer corcaneta qualquer]]
```

Isto vai gerar entre quatro a nove tartarugas no ecrã, cada uma numa posição **aleatória** e com uma cor da caneta (**corcaneta**) **aleatória**.

Agora, vamos ao movimento. Queremos criar movimento com o seguinte comando:

```
? ciclo [pede todas [cada [passo]]
```

Isto vai fazer com que as tartarugas repitam a mesma actividade por ordem (e não de uma só vez) numa espiral infinita. Utilizando o procedimento **passo** (ainda não definido), as tartarugas vão desenhar linhas para os seus sucessores e mover-se 1/20 da distância.

Primeiro, é necessário definir o procedimento **passo**. Considere sempre os problemas a resolver um por um:

Como procurar o nome da tartaruga seguinte?

- O nome da tartaruga seguinte vai começar com a letra t e é seguido de um número com um valor a mais do que o próprio número da tartaruga.
- A tartaruga pode saber o seu próprio nome, chamando **meunome**.
- Para encontrar o número de uma tartaruga, pode usar o comando **semprimeiro** (**sp** abreviado). Isso dá-nos o número da tartaruga seguinte.
- Pode anexar a letra t em frente do número (utilizando a operação **palavra**) para obter o nome completo para a tartaruga seguinte.

Escreva o seguinte:

```
? para passo
> fazlocal "seguinte palavra "t (1 + semprimeiro meunome)
```

Agora vai ter o nome do sucessor na variável local **seguinte**. A última tartaruga, no entanto, possui um pequeno problema – não vai ter nenhuma tartaruga para perseguir. Deste modo, tem de procurar a última tartaruga e fazê-la perseguir a primeira tartaruga **t1**.

Escreva o seguinte:

```
> se não objecto? :seguinte [faz "seguinte "t1]
```

A operação **direcção** vai ajudar-nos a saber como virar-nos para a tartaruga seguinte:

```
> fixarumo direcção :seguinte
```

Deste modo a tartaruga fixa o rumo em direcção à tartaruga seguinte.

O passo seguinte consiste em desenhar uma linha que una a nossa tartaruga com a sua sucessora e depois regresse. Primeiro, é necessário criar duas variáveis locais: a posição da tartaruga e a posição do seu sucessor:

```
> fazlocal "minhapos pos
> fazlocal "posseguinte pede :seguinte [pos]
> fixapos :posseguinte
> fixapos :minhapos
```

Por fim, a tartaruga deve deslocar-se em frente naquela direcção, exactamente 1/20 da distância:

```
> avança (abs :minhapos - :posseguinte) / 20
> fim
```

Agora, experimente.

```
? ciclo [pede todas [cada [passo]]]
```

O procedimento **passo** está agora completo:

```
para passo
fazlocal "seguinte palavra "t (1 + semprimeiro meunome)
se não objecto? :seguinte [faz "seguinte "t1]
fixarumo direcção :seguinte
fazlocal "minhapos pos
fazlocal "posseguinte pede :seguinte [pos]
fixapos :posseguinte
fixapos :minhapos
avança (abs :minhapos - :posseguinte) / 20
fim
? ciclo [pede todas [cada [passo]]]
```

O seguinte procedimento **criaTartarugas** vai gerar tartarugas ocasionalmente e executar o procedimento **passo** numa espiral infinita:

```
para criaTartarugas
eliminaobjecto todas
limpafundo
repete 4 + aleatório 6 [novo "tartaruga [pos qualquer corcaneta qualquer]]
activa todas
ciclo [cada [passo]]
fim
```

Experimente e escreva:

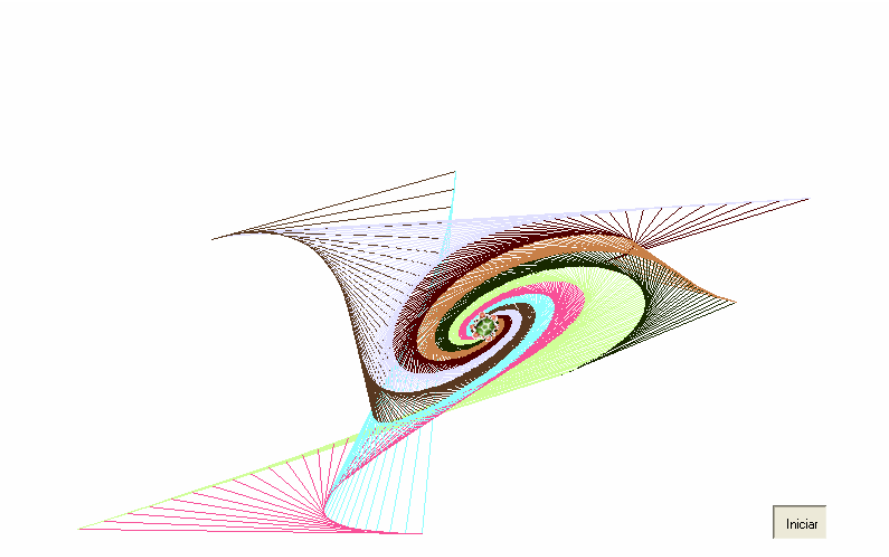
```
? criaTartarugas
```

No exemplo anterior, com os cães a correr, criámos um botão no ecrã para iniciar e parar a actividade. Crie um botão semelhante para esta actividade:

- Quando se carrega no botão, as tartarugas começam a movimentar-se em direcção ao seu sucessor.
- Quando se solta o botão, as tartarugas param de se mexer e de desenhar.

Dê um título ao botão; escreva no texto, por exemplo, **Iniciar**.

Dê uma acção ao botão; neste caso, aoPressionar deve ter **criaTartarugas**.



O que se espera é que, cada vez que clicar no botão **`Iniciar`**, obtenha um padrão diferente.

Capítulo 11. Tramas e Padrões

Antes de começar a secção seguinte sobre geometria, é necessário que aprenda uma outra característica da Linguagem Logo.

1. Variáveis Globais

Anteriormente, no capítulo acerca dos Pronomes do Logo, aprendemos a diferença entre **"números** e **:números**. Até agora, criámos "variáveis" como entradas para procedimentos. Por exemplo, **para torre :passo1 :passo2**. Estas variáveis não possuem qualquer valor fora dos procedimentos para as quais foram criadas. Para compreendermos melhor, abra o projecto **recursão**, que deve ter criado antes, e depois escreva:

```
? conta_menos 3
3
2
1
? escreve :número
A variável número não tem valor
Utilizar o comando FAZ ou NOMEIA para definir um valor para a variável
?
```

Talvez estivesse à espera de que o valor de **:número** fosse zero. No entanto, o Logo, assim que o procedimento termina, esquece-se do valor de **:número**, até o procedimento **conta_menos** ser chamado novamente, altura em que lhe é dada uma nova entrada e, por isso, **:número** possui um novo valor.

Contudo, vai sentir necessidade de criar uma variável que guarde o seu valor até que decida alterá-lo. A isto chamamos "declaração".

`? faz "número 7`	associa o valor 7 à variável número
`? escreve :número`	escreve o valor do número
`7`	
`? escreve "número`	escreve a palavra número

```
número
? faz "número :número + 3        adiciona 3 ao valor de número
? escreve :número                escreve o valor de número
10
```

Um erro comum que pode surgir é confundir `"número` com `:número` num programa.

Por exemplo:

```
? faz "número 7
? seentão "número = 7 [escreve "certo] [escreve "errado]
errado
```

A palavra `"número` não é igual a 7. Por isso:

```
? seentão :número = 7 [escreve "certo] [escreve "errado]
certo
```

Disposta desta forma, a questão torna-se perfeitamente clara, mas pode perder horas a perguntar-se onde é que se enganou. Vale sempre a pena verificar se utilizou dois pontos (`:`) onde deveria ter utilizado aspas (`"`) ou vice-versa.

Este capítulo debruça-se sobre as formas como os gráficos da tartaruga podem facilmente transformar-se em geometria da tartaruga ou em matemática da tartaruga.

Comece com este procedimento simples para desenhar um hexágono:

```
para hex :lado
repete 6 [av :lado dta 60]
fim
```

Consegue descobrir rapidamente que

```
? repete 6 [hex 50 av 50 esq 60]
```

produz um anel de hexágonos.

Contudo, se estivesse a fazer papel de parede com um procedimento hexagonal repetitivo, como faria para o conseguir?

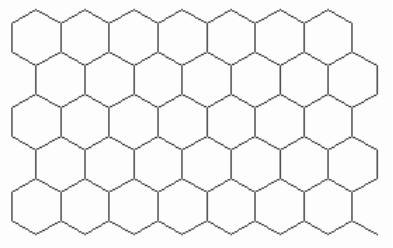
Comece com uma linha de hexágonos a percorrer o ecrã:

```
para linhahex :lado :tamanho
repete :tamanho [hex :lado dta 120 repete 2 [av :lado esq 60]]
fim
```

A próxima etapa é juntar duas linhas posicionando a tartaruga de forma a que ela recue ao longo da linha, desenhando a camada seguinte. Os movimentos a executar são diferentes consoante a tartaruga tenha terminado a linha da esquerda para a direita ou da direita para a esquerda.

Escrever um procedimento chamado **troca** pode fazer o que desejamos. Cada vez que se usa o **troca**, ele repõe a variável global chamada **interruptor**. O valor de **interruptor** é 0 ou 1. Se for 1, a tartaruga movimenta-se numa direcção e repõe o valor para 0; se for 0, a tartaruga movimenta-se na outra direcção e repõe o valor para 1. Uma vez que tenha compreendido como funciona o procedimento **troca**, o procedimento **padrãohex** é fácil de ser realizado.

```
para padrãohex :lado :tamanho :linhas
et faz "interruptor 0
repete :linhas [linhahex :lado :tamanho troca :lado]
fim
```



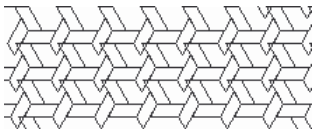
```
para troca :lado
seentão :interruptor = 0 [dta 120 av :lado dta 60 faz "interruptor 1]
[av :lado dta 60 av :lado esq 60 av :lado dta 180 faz "interruptor 0]
fim
```

Para proporcionar algum espaço para o procedimento **padrãohex**, é necessário mover a tartaruga para um ponto de partida apropriado. Escreva um procedimento, utilizando o comando **fixapos**.

Este tipo de padrões de tecelagem é muito apreciado pelas crianças. Nas escolas do 1º ciclo, o Logo pode ser utilizado para investigar a Arte Islâmica que, por motivos religiosos, nunca representa a forma humana. As formas características da Arte Islâmica, em azulejos e tapetes, incluem frequentemente tecelagens elaboradas, que não se copiam facilmente utilizando os instrumentos convencionais da Arte Ocidental.

Para os computadores, estes padrões constituem um trabalho bastante simples. Por exemplo, o padrão de **tecido** usa apenas sete procedimentos simples. Em vez do procedimento **troca** utilizado em **padrãohex**, este programa utiliza os procedimentos **direito** e **regressa** para lidar com o problema de posicionar a tartaruga correctamente no final de uma linha.

Vejamos então:

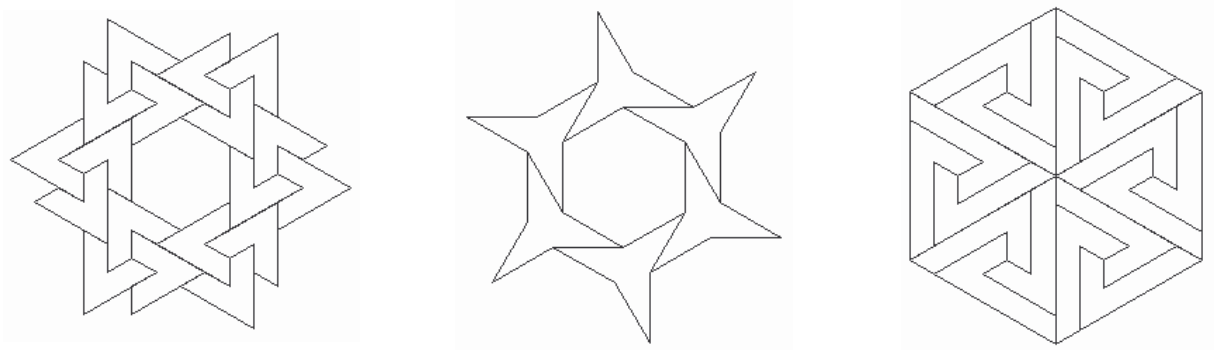


```
para tecido :lado :tamanho :profundidade
le et lc fixapos [-110 -100] esq 30 bc
repete :profundidade [desenhapadrão :lado :tamanho]
fim

para desenhapadrão :lado :tamanho
direito :lado :tamanho
salta :lado regressa :lado :tamanho - 1
salta :lado
fim

para direito :lado :tamanho
repete :tamanho [tri :lado dta 120 salta :lado esq 120]
tri :lado
fim

para regressa :lado :tamanho
repete :tamanho [tri :lado esq 60 salta :lado dta 60]
tri :lado
fim
```

```
para tri :lado
repete 3 [paralelo :lado dta 120]
fim

para paralelo :lado
repete 2 [av :lado dta 120 av :lado / 2 dta 60]
fim

para salta :lado
lc av :lado * 1.5 bc
fim
```

Agora procure outro padrão de tecelagem e escreva um conjunto de procedimentos Logo para o reproduzir.

Aqui estão algumas sugestões:



Estes padrões são regulares, sem rotações ou imagens espelhadas da unidade de base. Pode ainda divertir-se imenso com padrões irregulares, descobertos por um matemático de Oxford, Roger Penrose. Isto envolve uma dissecação de um losango para a forma de uma espécie de estrela e para a forma de uma seta. Todos este padrões são interligados por uma "Secção Dourada", uma relação muito apreciada pelos Gregos, que dissecaram a linha de forma a que:

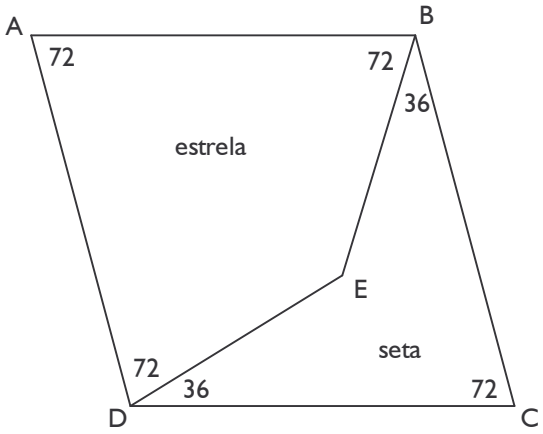
AB : PB :: PB : AB

Esta relação acaba por ser 1.618:1 e ocorre em todos os tipos de locais inesperados, incluindo o padrão de crescimento de certos tipos de caracóis do mar (búzios). Os arquitectos gregos acreditavam que um rectângulo cujos lados eram em proporção 1.618:1 fornecia as dimensões perfeitas para um edifício. Nos azulejos de Penrose, esta "Relação Dourada" aparece de variadas maneiras. A mais óbvia é a relação dos lados longos para os lados curtos, mas facilmente vai descobrir que a área da **estrela** é 1.618:1 vezes maior do que a área da **seta**.

Menos nitidamente, se criar grandes padrões de Penrose, consegue descobrir que usou 1.618:1 vezes o mesmo número de setas e estrelas. Se conseguir colocar uma turma à volta deste assunto, vai valer imenso a pena construir um conjunto de azulejos Penrose, sem recurso ao computador.

Aqui está um conjunto de procedimentos para começar:

```
para seta_dta :lado
av :lado * 1.618
dta 108 av :lado * 1.618
dta 144 av :lado
esq 36 av :lado
dta 144
fim
```

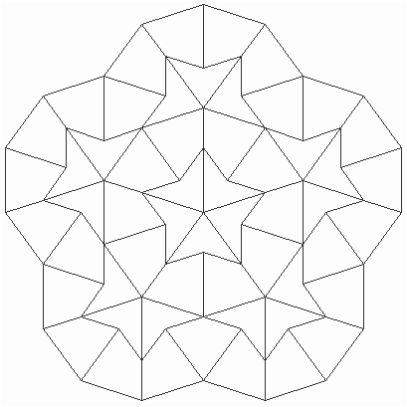



```
AB = BC = 1.618
AD = DC = 1.618
DE = EB = 1.00
```

```
para estrela_esq :lado
av :lado * 1.618 esq 108
av :lado * 1.618 esq 108
av :lado esq 36
av :lado esq 108
fim
para seta_esq :lado
av :lado * 1.618 esq 108
```

```
av :lado * 1.618 esq 144
av :lado dta 36
av :lado esq 144
fim

para estrela_dta :lado
av :lado * 1.618 dta 108
av :lado * 1.618 dta 108
av :lado dta 36
av :lado dta 108
fim

para estrela_grande :tamanho
av :tamanho * 1.618
dta 144 av :tamanho esq 108
estrela_esq :tamanho
estrela_dta :tamanho
dta 72 av :tamanho dta 144
seta_dta :tamanho
av :tamanho * 1.618
dta 108
fim

para estrela :tamanho :segmentos
repete :segmentos [estrela_grande :tamanho dta 72]
fim
```

```
para expandir :tamanho
estrela :tamanho 5
av :tamanho + :tamanho * 1.618 esq 108
repete 5 [estrela :tamanho 3 moveestrela :tamanho]
fim

para moveestrela :tamanho
av :tamanho * 1.618
dta 36
av :tamanho * 1.618
dta 180
fim

? expandir 100
```



Este é apenas um dos padrões que pode produzir, com a utilização dos azulejos Penrose.

Passemos agora para um tipo de padrão repetitivo completamente diferente, usando as técnicas recursivas que vimos anteriormente. Primeiro, crie o seguinte procedimento. Chama-se **lado1**, porque vamos compará-lo com o **lado2**.

```
para lado1 :lado
av :lado / 3
esq 60
av :lado / 3
dta 120
av :lado / 3
esq 60
av :lado / 3
fim
```

Siga estas instruções com um novo procedimento **estrela** e certifique-se de que desenha uma estrela com seis pontas, do tamanho que desejar.

```
para estrela :lado
repete 3 [lado1 :lado dta 120]
fim
```

Pode conceber o **lado1** como um tipo de **avança** diferente. A tartaruga é deslocada para frente uma determinada distância, e no final do seu caminho, a tartaruga ainda está virada para a mesma direcção como quando começou.

A grande diferença é que existe um bico na sua trajectória. Repare no que acontece se escrever **lado1** em substituição de **avança** com uma chamada recursiva para **lado2**. O procedimento nunca vai parar, se não possuir o mecanismo de controlo escrito na segunda linha do procedimento.

```
para lado2 :lado :nível
se :nível = 0 [av :lado pára]
lado2 :lado / 3 :nível - 1
esq 60
lado2 :lado / 3 :nível - 1
dta 120
lado2 :lado / 3 :nível - 1
esq 60
lado2 :lado / 3 :nível - 1
fim
```

Facilmente pode verificar que **lado1 50** e **lado2 50 1** produzem os mesmos resultados. Os benefícios de **lado2** só se tornam evidentes quando experimentamos:

```
? lado2 100 2
```

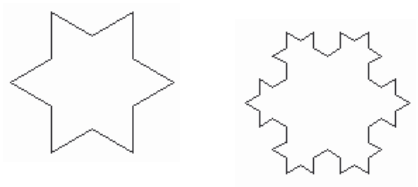
Cada segmento de linha adquiriu o seu próprio bico.

Repare nos diferentes níveis na ilustração. O único factor limitativo é a resolução do ecrã, que pode impedi-lo de distinguir os bicos, à medida que vai descendo para níveis mais profundos.

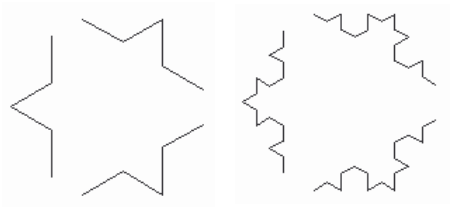
lado2 150 1 lado2 150 2 lado2 150 3 lado2 150 4

Esta ferramenta constituiu uma forma muito útil de desenhar flocos de neve.

```
para floconeve :lado :nível
repete 3 [lado2 :lado :nível dta 120]
fim
```

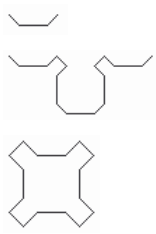


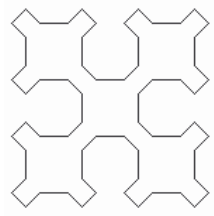
No caso de ainda estar confuso relativamente à forma como os lados se juntam para formar um floco de neve, estas imagens dos mesmos flocos de neve com os lados separados podem ajudá-lo.



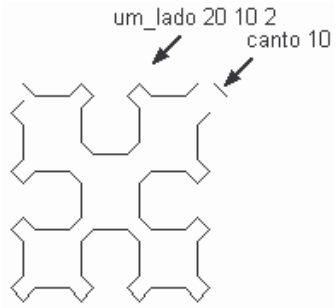
Esta é uma técnica muito poderosa, que pode ser usada para criar padrões ainda mais elaborados. Um matemático polaco de nome Sierpinski produziu o exemplo que se segue. O padrão é oriundo daqueles que encontramos nos azulejos orientais:

```
um_lado 20 10 1
um_lado 20 10 2

   azulejo 20 1
```



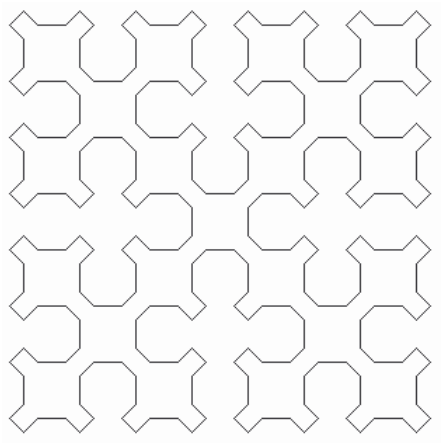


azulejo 20 2



É necessário experimentar os procedimentos abaixo indicados para poder saber o que está a acontecer. De qualquer modo, o princípio é exactamente o mesmo do procedimento **floconeve**, só que aqui está a lidar com uma figura de quatro lados, que em vez de fazer as curvas para fora, faz para dentro.

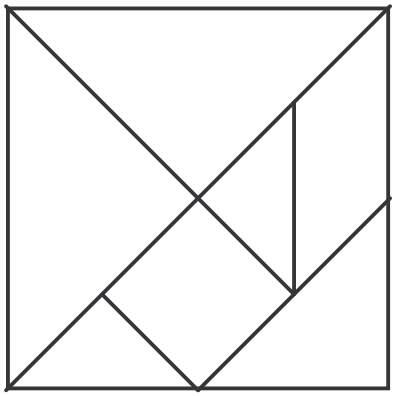
```
para azulejo :lado :nível
fazlocal "diag :lado / rqua 2
repete 4 [um_lado :lado :diag :nível canto :diag]
fim

para canto :lado
dta 45
av :lado
dta 45
fim
para um_lado :lado :diag :nível
se :nível = 0 [pára]
um_lado :lado :diag :nível – 1
dta 45 av :diag dta 45
um_lado :lado :diag :nível – 1
esq 90 av :lado esq 90
um_lado :lado :diag :nível – 1
dta 45 av :diag dta 45
um_lado :lado :diag :nível – 1
fim
```



O procedimento **um_lado** faz o mesmo que **lado2** em **floconeve**. A diferença principal é que a linha faz as curvas para dentro e não para fora, e assim estamos a trabalhar com um quadrilátero e não com uma figura triangular. Aqui está um exemplo de uma figura parcialmente explodida, para poder visualizar melhor o que está a acontecer.

Diversos livros sobre programação em Logo discutem curvas recursivas deste tipo mas, sem dúvida, que a melhor discussão é-nos fornecida por Harold Abelson e Andréa diSessa no livro *Turtle Geometry* (MIT Press 1980).

O antigo jogo chinês do **tangram** pode ser uma boa forma de testar as suas capacidades. Crie um conjunto de procedimentos Logo para implementar as peças de tangram e depois junte-as para fazer vários desenhos tangram.

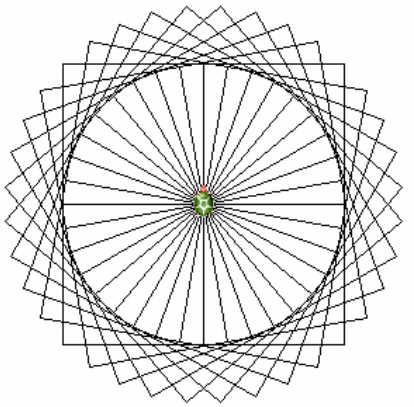


Capítulo 12. Processos e Eventos

Quando inicia o Imagina, existe apenas um processo – o processo de linha de comandos. Ele aceita comandos do utilizador e executa-os ao pressionar a tecla **<Enter>**. A linha de comandos não aceita nenhum novo comando até que o actual tenha terminado de ser executado.

Comece com um procedimento de desenho simples:

```
para desenhar :número
repete 36
[repete 4 [av :número dta 90 espera 10] dta 10]
fim
```



Experimente-o escrevendo:

```
? desenhar 100
```

Repare que o ponto de interrogação (**?**) não aparece na linha de comandos, enquanto o desenho não estiver completo.

1. Parar o Processo da Linha de Comandos

Para fazer parar o processo da linha de comandos e impedir que o continue a executar, pode clicar no botão **Parar Linha de Comandos** na barra de ferramentas, e forçá-lo a mostrar de novo o ponto de interrogação (**?**) e pedir um outro comando.

2. Começar um Novo Processo – o Comando Iniciar

A forma mais fácil de começar outro processo é utilizar o comando **iniciar**:

```
? iniciar [desenhar 100]
```

Este comando instrui o processo da linha de comandos no sentido de iniciar outro processo, que vai fazer o desenho, enquanto o processo de comando não tem mais nada para fazer. O ponto de interrogação (**?**) vai aparecer e vai ficar a aguardar um novo comando.

Se inserirmos alguns comandos para escrever, é mais fácil vermos o que está a acontecer:

```
? iniciar [escreve "início desenhar 100 escreve "fim] escreve [após iniciar]
```

Crie uma nova tartaruga t2:

```
? novo "tartaruga [pos qualquer]
```

e escreva um comando que inclua ambas as tartarugas:

```
? t1'desenhar 100 t2'desenhar 50
```

Agora compare-o com o resultado de:

```
? le t1'iniciar [desenhar 100] t2'iniciar [desenhar 50]
```

A imagem final no ecrã é a mesma, a maneira como foi desenhada é que foi visivelmente diferente. No primeiro caso, o processo da linha de comandos conclui o primeiro desenho com **t1**, depois o segundo desenho com **t2** e, por fim, termina e pede um novo comando. No segundo caso, o processo da linha de comandos inicia um processo para **t1**, outro para **t2** e pede um novo comando. Ambos os processos trabalham em paralelo.

3. O Comando ParaSempre

Neste exemplo, a tartaruga **t1** vai caminhar aleatoriamente e se atingir o alvo (a tartaruga **t2**), pára.

Para começar, deixe **t1** caminhar ao acaso:

```
? parasempre [av 1 dta aleatório 360]
```

O comando **parasempre** inicia um processo, que vai executar alguns comandos de forma repetitiva. É o mesmo que:

```
? iniciar [ciclo [av 1 dta aleatório 360]]
```

Se tivermos duas tartarugas, podemos começar mais processos com:

```
? t1'parasempre [av 1 dta aleatório 360] t2'parasempre [av 1 dta aleatório 360]
```

Em alternativa, podemos definir um procedimento para caminhar ao acaso desta forma:

```
para caminhar
av 1 dta aleatório 360
fim
```

Experimente:

```
? t1'parasempre [caminhar] t2'parasempre [caminhar]
```

4. Parar todos os Processos em Execução

Por vezes, pode acontecer que queira parar todos os processos que estão a ser executados.

Para fazê-lo, pode optar por uma das seguintes opções:

- Clicar no botão **Parar Todos** da barra de ferramentas principal;
- Pressionar a tecla **<F12>**;
- Pressionar em simultâneo as teclas **<Ctrl> + <Break>**;
- Escrever **páratudo** na linha de comandos.

5. Parar um Processo Sozinho

Dê um novo nome a **t2**, por exemplo **alvo**:

```
? t2'fixaestado [nome alvo]
```

Para que um processo possa parar sozinho, é necessário utilizar o comando **páre**:

```
para caminhar_e_experimentar
av 1 dta aleatório 360 se emcontacto? "alvo [pára]
fim
```

Experimente:

```
? t1'parasempre [caminhar_e_experimentar]
```

A tartaruga caminha sempre ao acaso e o processo pára quando a forma da tartaruga se sobrepõe à forma da outra tartaruga denominada de **alvo**.

Como pode verificar, caminhando ao acaso a tartaruga vai demorar bastante até atingir o alvo. E se desviarmos a tartaruga em determinada direcção?

Uma solução possível é fazer correr um outro processo, que direccione a tartaruga para o alvo. É como se a tartaruga tivesse um compasso, mas só o usasse de tempos a tempos para se virar para a direcção correcta ou, de outro modo, apenas caminha ao acaso.

Escreva o seguinte:

```
? t1'emcada 10 [fixarumo direcção "alvo]
```

Agora **t1** encontra o alvo muito mais rapidamente e pára junto dele. Experimente alterar o número 10 para um número maior ou menor. Não se esqueça de parar todos os processos, antes de começar uma nova experiência.

6. Parar um Processo com outro Processo

Existe ainda um problema. O processo que movimenta a tartaruga pára assim que ela atinge o alvo, mas o processo de compasso não pára.

Pode verificar isto, se olhar para a lista de todos os processos:

```
? apresenta todososprocessos
[@commander [fixarumo direcção "alvo]]
```

O primeiro processo é o processo de comando e o segundo é o processo de "compasso", e que ainda está em curso.

Podemos ainda ver que os processos começados pelo seu programa (utilizando **iniciar**, **parasempre**, **emcada** ou **após**) são denominados com base nas instruções que foram dadas.

Portanto, quando o processo **caminhar_e_experimentar** pára sozinho, deveria também parar o outro processo:

```
para caminhar_e_experimentar
av 1 dta aleatório 360
se emcontacto? "alvo [cancelar [fixarumo direcção "alvo] pára]
fim

? t1'parasempre [caminhar_e_experimentar]
```

7. Utilizar Nomes mais Curtos para os Processos

No exemplo acima descrito, pode tornar-se inconveniente ter uma lista tão longa de nomes de processos. Além disso, se modificarmos o processo de "compasso",

por exemplo, experimente:

```
? t1'emcada 100 [fixarumo (direcção "alvo + (aleatório 10) - 5)]
```

para introduzir um compasso não tão fiável. Então, teríamos também que alterar a entrada para cancelar o comando no procedimento **caminhar_e_experimentar**.

A solução é dar a cada processo um nome adequado e mais curto:

```
? (t1'emcada 100 [fixarumo direcção "alvo] "compasso)

para caminhar_e_experimentar
```

```
av 1 dta aleatório 360
se emcontacto? "alvo [cancelar "compasso pára]
fim

? (t1'parasempre [caminhar_e_experimentar] "caminhante)

? apresenta todososprocessos
[@commander compasso caminhante]
```

8. Lançar um Processo depois de um Tempo de Espera

Para terminar este exemplo, vamos acrescentar uma pequena recompensa ao processo. Quando a tartaruga atingir o alvo, irá escrever a palavra SUCESSO e se não o atingir em menos de 5 segundos, deve parar e aparecer a palavra FALHA.

Em vez de estar a incorporar o teste de tempo em cada passo do procedimento **caminhar_e_experimentar**, é muito mais fácil executar um outro processo, que apenas vai ser executado 5 segundos depois e que vai parar os dois processos:

```
? (após 5000 [escreve "FALHA cancelar "caminhante cancelar "compasso] "verificafim)
```

O comando após é apenas um modo mais simples de dizer:

```
? iniciar [espera 5000 escreve "FALHA cancelar "caminhante cancelar "compasso]
```

O nosso procedimento também necessita de ser ajustado, para anunciar o seu sucesso:

```
para caminhar_e_experimentar
av 1 dta aleatório 360
se emcontacto? "alvo [escreve "SUCESSO cancelar "compasso cancelar "verificafim párame]
fim
```

Finalmente, podemos criar um procedimento **vai** para incorporar todos os comandos necessários para iniciar uma experiência:

```
para vai
le
(após 5000 [escreve "FALHA cancelar "caminhante cancelar "compasso] "verificafim)
(t1'parasempre [caminhar_e_experimentar] "caminhante)
(t1'emcada 10 [fixarumo direcção "alvo] "compasso)
fim
```

Repare que, neste exemplo simples, não necessitamos de parar o processo acima descrito, porque o comando **cancelar** não indica nenhuma mensagem de erro, se o processo já tiver terminado. No entanto, nalguns casos pode ser um problema deixar um processo ser executado e, por essa razão, incluímos o comando **cancelar "verificafim**.

Se quisermos simplificar os nossos processos e se pressupusermos que não existem processos em curso no nosso projecto, então, em vez de cancelar cada processo individualmente, podemos utilizar o comando **páratodos** para parar todos os processos:

```
para caminhar_e_experimentar
av 1 dta aleatório 360
se emcontacto? "alvo [escreve "SUCESSO páratudo]
fim
```

```
para vai
le
(após 5000 [escreve "FALHA páratudo] "verificafim)
(t1'parasempre [caminhar_e_experimentar] "caminhante)
(t1'emcada 10 [fixarumo direcção "alvo] "compasso)
fim
```

9. Reagir a Eventos

Uma outra maneira de começar um processo é relacionar algumas instruções a um **evento**. Cada classe base tem um conjunto fixo de eventos aos quais pode reagir. Se relacionar uma lista de instruções a um evento de um objecto, então a lista de instruções é executada como um processo separado cada vez que o evento ocorrer.

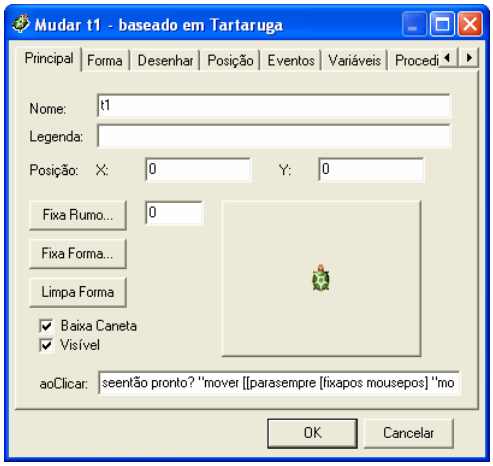
10. Eventos Básicos relacionados com o Rato em Tartarugas

No exemplo que se segue, vamos ter várias tartarugas animadas no ecrã. As tartarugas tornam-se animadas, quando o ponteiro do rato estiver por cima da tartaruga, e deixam de estar, quando o ponteiro do rato não estiver sobre a tartaruga. Vai poder, ainda, apontar e clicar sobre uma tartaruga e arrastá-la para uma nova posição no ecrã.

Comece um novo projecto e tente programar a tartaruga **t1**.

No primeiro passo, vamos tornar a tartaruga movível de uma forma simples: fazer com que a tartaruga esteja presa ao ponteiro do rato quando se clica pela primeira vez, e seja libertada quando for clicada pela segunda vez.

Para tal, temos que definir um evento **aoclicar** para a tartaruga, que vai executar um processo dizendo **fixapos posrato**, quando o utilizador clica na tartaruga pela primeira vez e pára o processo quando o utilizador clica nela pela segunda vez.



Podemos definir um evento na linha de comandos, utilizando o comando **fixaevento**:

```
? t1'fixaevento "aoclicar [seentão pronto? "mover [(parasempre [fixapos posrato] "mover)] [cancelar "mover]]
```

Uma forma mais fácil de fazer a mesma coisa é clicar com o botão direito do rato sobre a tartaruga, escolher a opção mudar t1 e escrever o corpo do evento (sem parêntesis) no local **aoclicar**, na página inicial da caixa de diálogo:

Certamente, não vai querer desenhar com a forma da sua tartaruga, portanto utilize o comando **levantacaneta** para levantar a caneta:

```
? t1'levantacaneta
```

Defina uma forma para a sua tartaruga. Utilize, por exemplo, um porco animado a andar de bicicleta. Experimente um dos vários animais que são fornecidos na biblioteca de imagens do Imagina.

```
? fixaforma "Animais\\porco.lgf
```

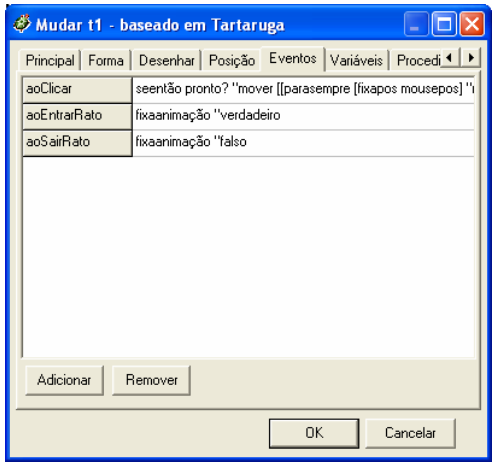
Por padrão, este animal é animado automaticamente. Para fazer parar a sua animação escreva:

```
? fixaanimação "falso
```

Agora defina dois eventos **aoentrarrato** e **aosairrato**, para ligar e desligar a animação respectivamente.

```
? t1'fixaevento "aoentrarrato [fixaanimação "verdadeiro]
? t1'fixaevento "aosairrato [fixaanimação "falso]
```

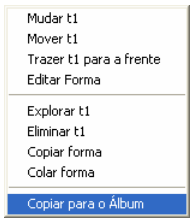
Experimente-os na sua nova tartaruga.



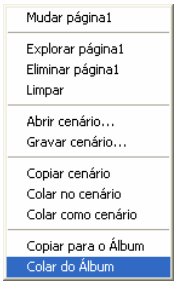
Uma vez mais, seria mais fácil definir estes eventos na caixa de diálogo **Mudar**. Para definir eventos para além do **aoclicar**, deve clicar na secção **Eventos**:

Já nesta caixa de diálogo e se desejar adicionar um novo evento, clique no botão **Adicionar** depois escolha o nome do evento e clique OK.

Para tornar o ecrã mais interessante, coloque mais tartarugas na página.



A forma mais fácil é copiar **t1** para o **álbum**. Clique com o botão direito do rato em **t1** e escolha a opção **Copiar para o Álbum**.



Clique com o lado direito do rato num outro espaço livre na página e escolha **Colar do Álbum**:

Agora pode mudar a forma desta tartaruga:

```
? t2'fixaforma "|Monstros\bicho 1|
```

Em alternativa, poderia escrever o seguinte na linha de comandos:

```
? repete 4 [clone "t1 [pos qualquer forma (palavra "|Monstros\bicho | contador)]]
```

Agora já possui uma pequena actividade de apontar e clicar.

11. Uma Introdução às Classes

Imagine um ecrã onde estão algumas tartarugas (objectos) no lado esquerdo, por exemplo casas, árvores e vedações. Ao clicar em qualquer objecto, pode obter uma cópia idêntica ao objecto (tartaruga) e que pode ser deslocada para qualquer local do ecrã. Para remover o objecto, basta arrastá-lo para o caixote do lixo. O caixote do lixo é apenas uma outra tartaruga no ecrã.

Para este jogo, vai precisar de três tipos de tartarugas:

Tartaruga Pilha — Quando clica com o botão esquerdo do rato sobre esta tartaruga, aparece uma tartaruga amovível com a mesma forma que se cola ao ponteiro do rato. A tartaruga pode ser arrastada e colocada em qualquer ponto do ecrã.

Tartaruga Móvel — Possui o atributo de **`autoarrastar`** accionado e a caneta levantada. Quando se clica sobre ela com o botão esquerdo do rato (por exemplo, quando começa uma operação de arrastamento), ela passa para a frente de todas as tartarugas. Se soltar o botão esquerdo do rato (por exemplo, quando termina a operação de arrastamento) e se for sobreposta pelo caixote do lixo, ela é apagada.

Caixote do Lixo — É apenas uma marca algures na página com o nome de lixo, não necessita de nenhum comportamento especial.

A maneira mais fácil de desenvolver um projecto com tartarugas de vários tipos é utilizar classes de tartarugas. Uma classe é uma amostra predefinida que indica como é que se devem comportar todas as tartarugas do mesmo tipo. Mas, ao contrário da tartaruga, essa amostra não vive em lado nenhum, não se pode ver nem indicar-lhe **`avança`**, **`direita`** ou **`esquerda`**. Esta última figura é necessária especialmente para as tartarugas móveis, porque inicialmente não está criado nenhum tipo dessas tartarugas.

```
? novaclasse "tartaruga "pilha [evento'aoclicaresquerdo [novo "móvel [pos (pos)
forma (forma)] pede últimonome [iniciaarrastar]]]
```

Este comando vai criar uma classe de tartarugas denominada **pilha**. Também define uma reacção ao evento **aoclicaresquerdo**. Quando clicada, é criada uma nova tartaruga da classe **móvel** e a sua forma é definida para o mesmo valor da forma da tartaruga que foi clicada.

O evento **aoclicaresquerdo** cria um novo objecto da classe **móvel**, define a sua posição e forma conforme as configurações de **pos** e **forma** do objecto que está a ser clicado. Então, é pedido ao objecto recém-criado (**últimonome** simboliza o nome desse objecto) que comece a arrastar. Isto é, a nova tartaruga é arrastada pelo rato até ser libertada, quando o botão esquerdo for solto.

```
? novaclasse "tartaruga "móvel [evento'aoclicaresquerdo [trazfrente]
evento'aolargaresquerdo [se emcontacto? "lixo [eliminaobjecto meunome]]
autoarrastar verdadeiro caneta lc]
```

Este comando define a classe das tartarugas móveis.

Dê um novo nome a **t1** para ser a tartaruga Caixote do Lixo:

```
? t1'fixaestado [nome lixo]
```

Desloque-a para o canto inferior esquerdo (utilize **<shift>** + **botão direito do rato**) e descubra uma forma diferente para ela na biblioteca de imagens do Imagina (pode utilizar, por exemplo, a imagem **lixo.lgf** que se encontra na pasta **Edifícios**).

Crie a primeira tartaruga pilha:

```
? novo "pilha []
```

Escolha, por exemplo, uma imagem de uma casa para a tartaruga:

Depois de definir a forma, podemos verificar que é muito grande para esta actividade – queremos criar uma pequena cidade de casa na página. Para tornar a forma da tartaruga mais pequena, necessitamos de modificar a escala da sua forma.

Clique com o botão direito do rato sobre a casa e escolha a opção **Mudar t2** do menu de contexto que surge.

Clique na secção **Forma** e altere a medida da **escalaX**. Experimente `0.3`.

Estamos a fazer esta operação apenas neste caso e não para toda a classe, pois as outras formas podem necessitar de outras escalas diferentes.

Também poderíamos ter feito isto directamente na linha de comandos:

```
? t2'fixaescalaforma 0.3
```

Para modificar o evento **aoclicaresquerdo** para a classe **pilha** e para definir a escala da forma de cada nova tartaruga móvel para a mesma que a original, faça o seguinte

```
? pilha'fixaevento "aoclicaresquerdo [novo "móvel [pos (pos) forma (forma) escalaforma (escalaforma)] pede últimonome [iniciaarrastar]]
```

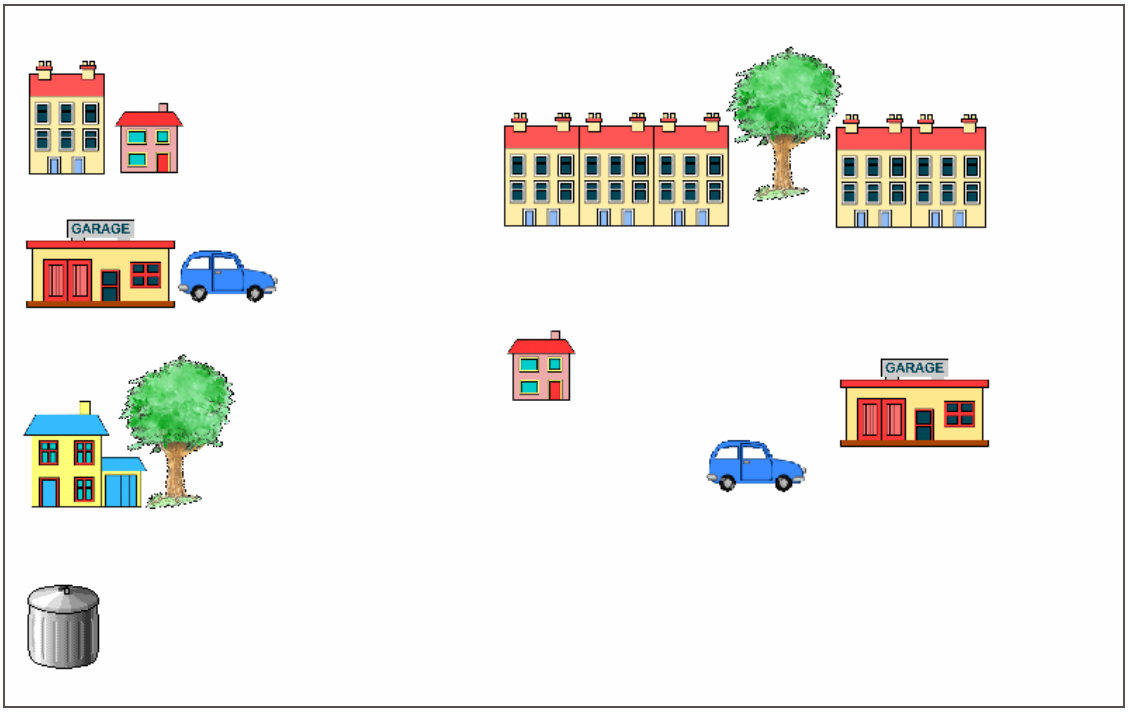
Experimente a nova actividade. Crie várias casas novas clicando na tartaruga pilha, arraste-as pelo ecrã e apague algumas, arrastando-as para o caixote do lixo.

Agora vamos criar outra tartaruga pilha.

A forma mais fácil de o fazer é clicar com o botão direito do rato sobre a que existe, escolher a opção **Copiar para o Álbum**, e depois clicar novamente com o botão direito na página e escolher **Colar do Álbum**. Por fim, modifique a sua forma para uma casa ou um edifício diferente.

Pode repetir esta operação diversas vezes para adicionar casas, garagens, árvores, vedações ou até carros à sua página. Por vezes, também pode ter necessidade de ajustar a escala da forma (tal como vimos anteriormente).

Agora pode experimentar a sua nova actividade e construir uma pequena cidade:



Índice