CAPÍTULO 4 - Ações especificas

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

4.1 - Entrada de dados

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

Logo permite a entrada de dados em tempo de execução. Para isso deve-se fazer uso de uma das primitivas: **READCHAR** (**RC**), **READLIST** (**RL**) e **READWORD** (**RW**) que possuem como estrutura sintática o estilo:

```
READCHAR <conteúdo>
READLIST <conteúdo>
READWORD <conteúdo>
```

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

O uso dos comandos **READCHAR**, **READLIST**, **READWORD** deve ser realizado em conjunto de outro comando como **PRINT** ou **MAKE**. Quando um desses comados é usado ocorre uma aparente paralização da execução do procedimento. Esta “parada” é um momento em que o interpretador espera a entrada do dado para a ação do procedimento.

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

```
TO IDADE
  PR [Entre sua idade: ]
  MAKE "VALOR READWORD
  IFELSE :VALOR >= 18 [PR [Maior de idade]] [PR [Menor de idade]]
END
```

Em seguida execute o procedimento "**IDADE**" algumas vezes e informe valores diferentes e veja os resultados apresentados. Note que ao ser executado o procedimento "**IDADE**" ocorre a apresentação da mensagem "**Entre sua idade:**" e ocorrer a espera da entrada do dado para a operação.

Anteriormente foi apresentado o comando **REPEAT** que tem por finalidade repetir um trecho de código demarcado dentro de colchetes um determinado número de vezes. Há um comando

que pode mostrar o momento da contagem em que o comando **REPEAT** se encontra. Conheça a primitiva **REPCOUNT**.

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

```
REPEAT 5 [PR (SENTENCE "Conta: REPCOUNT)]
```

Veja o resultado obtido junto a figura 4.1.

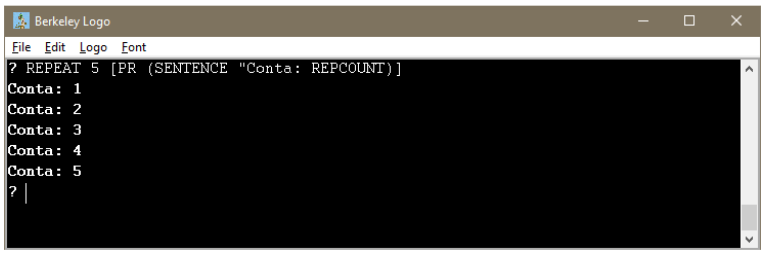


Figura 4.1 - Resultado da ação do comando "REPCOUNT" junto a "SENTENCE" com "PR".

Veja que a cada execução do comando **REPEAT** o comando **REPCOUNT** detectou um valor da contagem. A cada contagem o valor de **REPCOUNT** foi concatenado a palavra "**Conta:**" por meio do comando **SENTENCE**. O resultado da tabuada a ser apresentado deve ser delimitado entre parênteses estabelecendo assim uma estrutura de lista.

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

```
TO TABUADA
  PR [Entre um valor numerico para tabuada:]
  MAKE "NUMERO RW
  IFELSE (AND :NUMERO >= 2 :NUMERO <= 9) [
    REPEAT 10 [PR (
      SE :NUMERO "X REPCOUNT "= :NUMERO * REPCOUNT
    )]
  ][
    PR [Valor fora da faixa]
  ]
END
```

Em seguida execute separadamente a instrução:

```
TABUADA
```

Quando apresentada a caixa de diálogo para entrada, forneça um valor numérico entre "**2**" e "**9**". Por exemplo, entre o valor "**7**" e veja a tabuada apresentada como indicado na figura 4.2.

Figura 4.2 - Resultado de uma tabuada simples.

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

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

```
FORM <número> <tamanho> <precisão>
```

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

Para fazer uso do recurso **FORM** considere um procedimento chamado "**TABUADA2**" semelhante ao procedimento **TABUADA** que apresente os valores numéricos alinhados na direita para a esquerda. Desta forma, entre o seguinte código:

```
TO TABUADA2
  PR [Entre um valor numerico para tabuada:]
  MAKE "NUMERO READWORD
  IFELSE (AND :NUMERO >= 2 :NUMERO <= 9) [
    REPEAT 10 [PR (
      SE
        FORM :NUMERO 2 0
      "X
        FORM REPCOUNT 2 0
      "=
        FORM (:NUMERO * REPCOUNT) 3 0
    )]
  ][
    PR [Valor fora da faixa]
  ]
END
```

Em seguida execute separadamente as instruções após fazer uso do comando **CS**:

```
TABUADA2
```

Quando apresentada a caixa de diálogo entre o valor "**5**" e veja a tabuada apresentada como indicado na figura 4.3.

Figura 4.3 - Resultado de uma tabuada com valores formatados (alinhados).

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

4.2 - Randomização

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

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

```
RANDOM <número>
```

Onde, o indicativo "**número**" refere-se ao valor máximo estabelecido para sorteio. Este valor poderá variar entre "**0**" (zero) e o valor mais próximo ao valor anterior estabelecido no parâmetro "**número**".

Veja algumas instruções de sorteio entre os valores "**0**" e "**99**":

```
PRINT RANDOM 100
PRINT RANDOM 100
PRINT RANDOM 100
PRINT RANDOM 100
PRINT RANDOM 100
```

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

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

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

```
TO MALUQUINHO
  REPEAT 100 [
    FD (RANDOM 0 60)
    RT (RANDOM 0 360)
  ]
END
```

Além de operar valores randômicos entre "**0**" e "**N-1**" a primitiva **RANDOM** permite especificar faixas de valores predefinidos entre valores inicial e final, desde que a definição, neste caso, esteja definida entre parênteses. Por exemplo a forma "**(RANDOM 0 60)**" diz que são aceitos valores gerados entre "**0**" e "**60**", ou seja, "**(RANDOM 0 4)**" equivale a "**RANDOM 5**". A definição "**(RANDOM 3 7)**" equivale a "**(RANDOM 5)+3**".

Em seguida execute a instrução:

```
CS MALUQUINHO
```

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

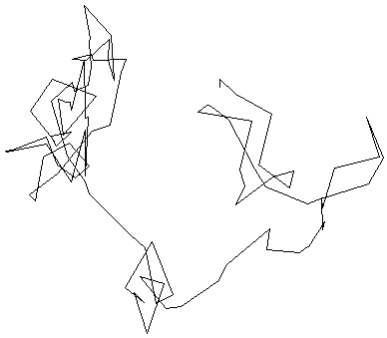


Figura 4.5 - Resultado de figura geométrica aleatória.

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

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

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

```
(RERANDOM <semente>)
```

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

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

```
(RERANDOM 1)
REPEAT 5 [PR (RANDOM 0 100)]
(RERANDOM 1)
REPEAT 5 [PR (RANDOM 0 100)]
```

Veja que a execução das duas séries de valor sorteados a partir da instrução **(RERANDOM 1)** é exatamente a mesma como mostra a figura 4.6.

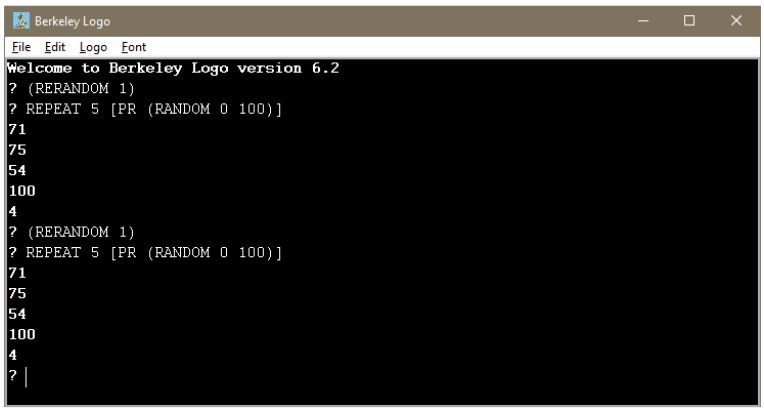


Figura 4.6 - Resultado repetidos de valores sorteados.

4.3 - Coordenadas

Todos as figuras geométricas planas apresentadas até este ponto foram desenhadas a partir do ponto central da área de ação do modo gráfico, ou seja, das coordenadas "**0**" para **X** e "**0**" para **Y**. O ponto central pode-se ser chamado de *CASA* por ser o ponto de início das operações geométricas em Logo. Para verificar a posição em que a tartaruga se encontra usa-se a primitiva **POS** em conjunto com **PRINT**. No entanto, existem outras primitivas de operação para auxiliar mudanças do posicionamento da tartaruga, tais como: **SETX**, **SETY**, **SETXY** e **HOME**.

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

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

```
PENUP
PRINT POS
SETX  50
SETY -30
PRINT POS
```

A figura 4.7 mostra os resultados apresentados. Veja os valores gerados após o uso da instrução **PRINT POS** na coordenada "**0 0**" com marcação em amarelo e na coordenada "**50 -30**" com marcação em verde.

Figura 4.7 - Identificando posições cartesianas.

Para os comandos **SETX** e **SETY** os valores usados devem estar limitados a dimensão da área de trabalho. Veja que a partir da posição central (a *CASA*) é possível usar valores positivos para posicionar a tartaruga à direita ou acima da posição inicial ou usar valores negativos para posicionar a tartaruga à esquerda ou abaixo da posição inicial. A Figura 4.8 representa esquematicamente os movimentos de salto com o par de comandos **SETX 30** com **SETY 20** e **SETX -30** com **SETY -10**.

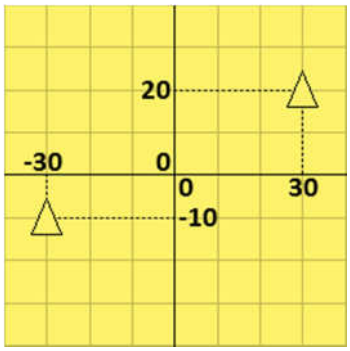


Figura 4.8 - Posições cartesianas da tartaruga.

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

```
CT CS
PRINT POS
SETXY -50 25
PRINT POS
HOME
PRINT POS
```

Na sequência execute o comando **PENDOWN**. A figura 4.9 mostra os valores das posições cartesianas após o uso dos comandos **SETXY** e **HOME**.

Figura 4.9 - Posições cartesianas após comandos "SETXY" e "HOME".

A partir desses "novos" recursos e de alguns recursos conhecidos é possível extrapolar e criar desenhos e imagens mais elaboradas, como por exemplo a figura da face quadrada indicada junto a figura 4.10, adaptada de Harvey (1997, p. 184).

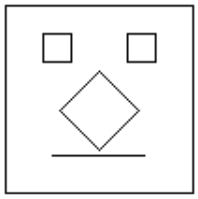


Figura 4.10 - Face quadrada.

Apesar de simples a imagem da figura 4.10 apresenta um grau de complexidade interessante, pois diversas ações são necessárias. Assim sendo, entre o seguinte código:

```
TO CABECA
  PENDOWN
  REPEAT 4 [FD 100 RT 90]
  PENUP
END

TO PREPARA.OLHO
  LT 90 FD 65 RT 90 FD 20
  END



TO OLHO.DIREITO
  PENDOWN
  OLHO
  PENUP
  BK 15
END



TO BOCA
  PENUP
  FD 20 RT 90 FD 25
  PENDOWN
  FD 50
  PENUP
  BK 75
END
```

```
TO OLHO
  REPEAT 4 [FD 15 RT 90]
END



TO OLHO.ESQUERDO
  PENDOWN
  OLHO
  PENUP
  FD 45
END

TO NARIZ
  RT 90 FD 20 LT 45
  PENDOWN
  REPEAT 4 [FD 29 RT 90]
  PENUP
  HOME
  LT 135
  PENDOWN
END

TO FACE
  CABECA
  BOCA
  PREPARA.OLHO
  OLHO.ESQUERDO
  OLHO.DIREITO
  NARIZ
END
```

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

```
HT
FACE
```

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

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

```
WAIT <tempo>
```

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

```
TO POLIGOM :LADOS :TAMANHO
  HT
  IF :LADOS = 30 [STOP]
  PENDOWN
  REPEAT :LADOS [FD :TAMANHO RT 360 / :LADOS]
  PENUP
  HOME
  PENDOWN
  REPEAT :LADOS [FD :TAMANHO LT 360 / :LADOS]
  WAIT 30
  POLIGOM (:LADOS + 1) :TAMANHO
  ST
END
```

Em seguida execute as instruções:

```
CS POLIGOM 3 50
```

Veja o resultado da operação na figura 4.11. Atente para o efeito espelho conseguido com o posicionamento ao HOME após o desenho do lado direito para então desenhar o lado esquerdo. Atente para os movimentos dos efeitos dos comandos **PENUP** e **PENDOWN** entre as imagens direita e esquerda.

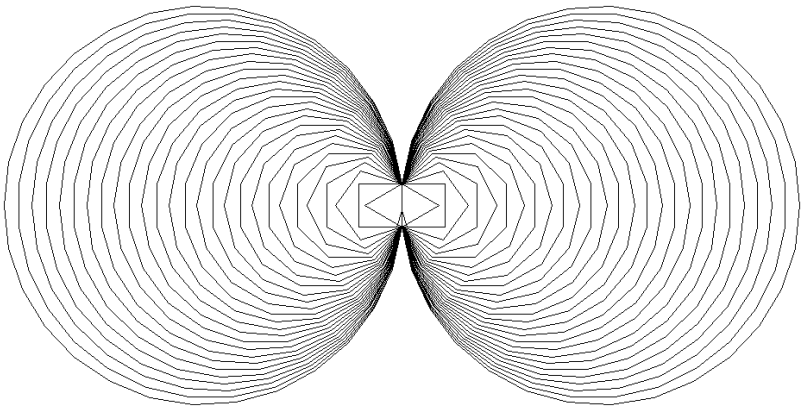


Figura 4.11 - Polígono com efeito de espelhamento.

4.4 - Cores

Os desenhos produzidos até este momento foram riscados em cor preta em um fundo branco. A linguagem Logo permite manipular o conjunto de cores do risco do desenho e do fundo da tela. A tabela 4.1 mostra dezesseis tonalidades de cores que podem ser usadas no "*UCBLogo*".

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

Tabela 4.1 - Índice de cores do ambiente UCBLogo.

A mudança de cores se faz com as primitivas **SETBG** (muda a cor do fundo - **SETBACKGROUND**) e **SETPC** (muda a cor do lápis - **SETPENCOLOR**) com a indicação do padrão **RGB** da cor desejada ou a partir do índice da cor indicado na tabela 4.1.

Para formatar o fundo em preto e o lápis em amarelo use as instruções:

```
SETBG 0
SETPC 6
```

Para ver o resultado execute a instrução:

```
CS FACE
```

Para formatar o fundo em azul e o lápis em azul celeste use as instruções:

```
SETBG  1
SETPC 11
```

Para ver o resultado execute a instrução:

```
CS FACE
```

As figuras 4.12 e 4.13 mostram os resultados com uso de cores.

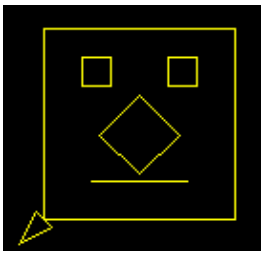


Figura 4.12 - Imagem no fundo preto em tom amarelo.

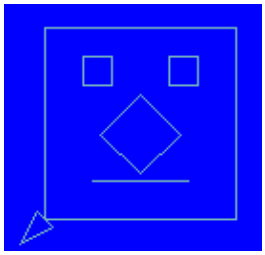


Figura 4.13 - Imagem no fundo azul em tom azul celeste.

Para um teste do uso de cores considere o conjunto de procedimentos a seguir usados para desenhar um sol estilizado com o traço do lápis na cor laranja "**14**" em fundo na cor azul "**1**" (adaptado de Abelson, 1981, p. 22). Assim sendo, entre o seguinte código:

```
TO ARCO1 :TAMANHO :GRAU
  REPEAT :GRAU / 10 [
    FORWARD :TAMANHO
    RIGHT 10
  ]
END

TO RAIO :TAMANHO
  REPEAT 2 [
    ARCO2 :TAMANHO 90
    ARCO1 :TAMANHO 90
  ]
END
```

```
TO ARCO2 :TAMANHO :GRAU
  REPEAT :GRAU / 10 [
    FORWARD :TAMANHO
    LEFT 10
  ]
END

TO SOL :TAMANHO
  SETBG  1
  SETPC 14
  REPEAT 9 [
    RAIO :TAMANHO
    RIGHT 160
  ]
END
```

Em seguida execute a instrução:

```
CS SOL 10
```

A Figura 4.14 apresenta a imagem gerada pela execução da chamada do procedimento "**SOL 10**".

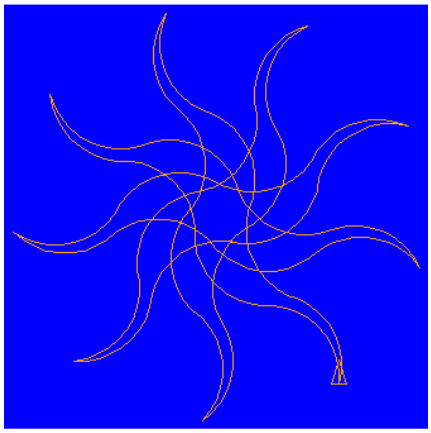


Figura 4.14 - Procedimento "SOL 10".

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

Para voltar a tela ao modo padrão execute as instruções:

```
SETBG 0
SETPC 7
```

Além das mudanças na definição das cores do fundo da tela e do traço é possível fazer o preenchimento de certa cor dentro de uma figura. O preenchimento com certa cor é realizado com o uso da primitiva **SETPC** com auxilio da prmitiva **FILL** aplicado dentro da área a ser colorizada.

O procedimento "**LARANJA**" a seguir faz a apresentação de um quadrado nesta cor. Desta forma, escreva o seguinte código:

```
TO LARANJA
  REPEAT 4 [FD 50 RT 90]
  PENUP
  SETPOS [30 30]
  SETPC  14
  FILL
  SETPOS [0 0]
  SETPC  0
  PENDOWN
END
```

Observe que no procedimento após a apresentação do quadrado se faz o recolhimento do lápis com o comando **PENUP**. Em seguida faz-se a movimentação da tartaruga para a coordenada "**30**" e "**30**", seleciona-se a cor "**14**" (*laranja*) com o comando **SETPC** e efetua-se a pintura da imagem com o comando **FILL**. Após a pintura da imagem faz-se com que a tartaruga volte para sua posição "*CASA*" com a instrução "**SETPOS [0 0]**", lembrando que poderá para esta ação ser usado o comando **HOME** e configura-se o lápis com tom de cor preto. Na sequência faz-se a ativação do lápis com o comando **PENDOWN**.

Em seguida execute as instruções **CS** e **LARANJA** e veja a imagem apresentada semelhante à figura 4.15.

Figura 4.15 - Imagem com preenchimento.

4.5 - Fractais

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

Não é objetivo deste tópico, explorar este tema com profundidade, o objetivo é apenas e tão somente demonstrar a utilização da linguagem nesta tarefa operacional. Para tanto, considere a execução do procedimento "**PONTA**" o qual mostrará a imagem indicada na figura 4.16 a partir da chamada "**PONTA 30**".

```
TO PONTA :LADO
  FD :LADO LT  60
  FD :LADO RT 120
  FD :LADO LT  60
  FD :LADO
END
```

Figura 4.16 - Imagem do procedimento "PONTA".

A partir do procedimento "**PONTA**" considere o procedimento "**FRACTAL1**" a seguir que executa o procedimento ponta seis vezes, como mostra a figura 4.17 após a chamada "**FRACTAL1 30**".

```
TO FRACTAL1 :LADO
  REPEAT 6 [
    RT 120
    PONTA :LADO
    LT  60
  ]
END
```

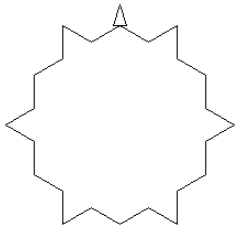


Figura 4.17 - Imagem do procedimento "FRACTAL1".

Note que com os procedimentos "**PONTA**" e "**FRACTAL1**" desenvolvidos foi efetuada uma imagem geométrica irregular (Figura 4.17) a partir de seis repetições da imagem gerada pelo procedimento "**PONTA**" (Figura 4.16).

A partir da imagem proposta pelo procedimento "**PONTA**" é possível formar outras imagens baseadas em fractais. Observe por exemplo o procedimento "**FRACTAL2**" em que se estabelecem mudanças na direção de deslocamento da tartaruga a partir dos comandos **RT** e **LT** dentro do laço gerenciado pelo comando **REPEAT**, como indicado na figura 4.18. Use a chamada de instrução "**FRACTAL2 30**".

```
TO FRACTAL2 :LADO
  REPEAT 6 [
    LT 120
    PONTA :LADO
    RT  60
  ]
END
```

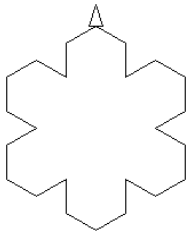


Figura 4.18 - Imagem do procedimento "FRACTAL2".

É possível a partir da execução do procedimento "**PONTA**" fazer a apresentação de formas fractais com formato mais complexo. Observe o procedimento"**FRACTAL3**" que mostra um trecho de imagem fractal bem tradicional, como indicado na figura 4.19. Use "**FRACTAL3 30**".

```
TO FRACTAL3 :LADO
  RT 90
  REPEAT 2 [
    PONTA :LADO
    LT  60
    PONTA :LADO
    RT 120
  ]
END
```

Figura 4.19 - Imagem do procedimento "FRACTAL3".

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

A partir do procedimento "**FRACTAL3**" pode-se extrapolar a criação de fractais com base no procedimento "**PONTA**". Observe o procedimento "**FRACTAL4**" que gera a imagem indicada na figura 4.20 a partir da execução de "**FRACTAL4 30**" que mostra o fractal de *von Kock* (floco de neve).

```
TO FRACTAL4 :LADO
  REPEAT 4 [
    FRACTAL3 :LADO
    LT 120
    RT  30
  ]
END
```

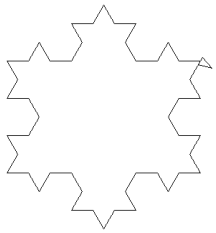


Figura 4.20 - Imagem do procedimento "FRACTAL4".

O procedimento "**SAMAMBAIA**" mostra a imagem de um fractal a partir da definição de determinado número de folhas. A Figura 4.21 mostra a imagem gerada a partir da execução da instrução "**SAMAMBAIA 300**" (adaptado, NASCIMENTO, 2000).

```
TO SAMAMBAIA :FOLHAS
  IF :FOLHAS > 4 [
    FD :FOLHAS / 25
    LT 80 SAMAMBAIA :FOLHAS * 0.3
    RT 82  FD :FOLHAS / 25
    RT 80 SAMAMBAIA :FOLHAS * 0.3
    LT 78 SAMAMBAIA :FOLHAS * 0.9
    LT  2 BK  :FOLHAS / 25
    LT  2 BK  :FOLHAS / 25
  ]
END
```

Figura 4.21 - Procedimento "SAMAMBAIA 300".

4.6 - Outras repetições

Além das repetições produzidas com o comando **REPEAT** e as operações de recursão Logo possui outras primitivas para a realização de operações baseadas em repetições de trechos de códigos, como: **WHILE**, **DO.UNTIL** e **FOR**.

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

Para verificar o funcionamento do comando **WHILE** considere o procedimento "**CONTAGEM1**" que apresenta os valores de contagem entre "**1**" e "**5**" de "**1**" em "**1**". Assim sendo, entre o seguinte código:

```
TO CONTAGEM1
  MAKE "I 1
  WHILE [:I <= 5] [
    PRINT :I
    MAKE "I :I + 1
  ]
END
```

Em seguida execute as instruções **CS**, **CT** e "**CONTAGEM1**" e veja a imagem apresentada semelhante à figura 4.22.

Figura 4.22 - Apresentação da contagem de "1" a "5" com "WHILE".

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

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

Veja o procedimento "**PENTAGRAMA**" que mostra uma estrela de cinco pontas gerada a partir das primitivas **WHILE** e **SETHEADING** (**SETH**). A primitiva **SETH** pode ser usada como substituta da diretiva **RT**. Para tanto, entre o seguinte código:

```
TO PENTAGRAMA
  SETHEADING 18
  MAKE "I 1
  WHILE [:I <= 5] [
    FD 100
    RT 144
    MAKE "I :I + 1
  ]
END
```

Em seguida execute as instruções **CS** e "**PENTAGRAMA**" e veja a imagem apresentada semelhante à figura 4.23.

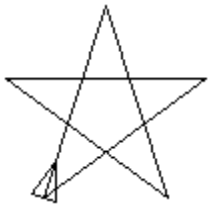


Figura 4.23 - Apresentação de imagem gerada com comando "WHILE".

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

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

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

Para ver o funcionamento do comando **DO.UNTIL** considere o procedimento "**CONTAGEM2**" que apresenta os valores de contagem entre "**1**" e "**5**" de "**1**" em "**1**". Assim sendo, entre o seguinte código:

```
TO CONTAGEM2
  MAKE "I 1
  DO.UNTIL [
    PRINT :I
    MAKE "I :I + 1
  ][:I > 5]
END
```

Veja que neste comando a escrita do valor e a atribuição de mais "**1**" ocorre uma vez antes da avaliação da condição "**[:I > 5]**".

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

A primitiva **FOR** é semelhante a **REPEAT** por realizar ações iterativas. Para ver o funcionamento de **FOR** considere o procedimento "**CONTAGEM3**" que mostra os valores de contagem entre "**1**" e "**5**" de "**1**" em "**1**". Assim sendo, entre o seguinte código:

```
TO CONTAGEM3
  FOR [I 1 5 1] [
    PRINT :I
  ]
END
```

Note que a primitiva **FOR** usa a definição de uma variável e a definição de três valores: o primeiro valor determina o início da contagem, o segundo valor determina o fim da contagem e o terceiro valor determina o incremento da contagem, que pode ser omitido quando o incremento é de "**1**" em "**1**".

Em seguida execute as instruções **CS**, **CT** e "**CONTAGEM3**".

Já que foi feita uma estrela de cinco pontas (pentagrama) com o comando **WHILE** será feita uma estrela de 40 pontas com o comando **FOR**. Para tanto, definida o procedimento "**ESTRELA40**". Assim sendo, entre o seguinte código:

```
TO ESTRELA40
  FOR [I 1 40] [
    BK 200
    LT 170
  ]
END
```

Em seguida execute as instruções **CS** e "**ESTRELA40**" e veja a imagem apresentada semelhante à figura 4.24.

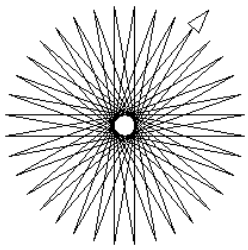


Figura 4.24 - Apresentação de imagem gerada com comando "FOR".

Os comandos: **REPEAT**, **WHILE**, **DO.UNTIL** e **FOR** são recursos complementares e não necessariamente substitutos um dos outros. Esses comandos podem ser trabalhados juntos dentro de um mesmo projeto sequencialmente ou encadeados. Sequencialmente quando um é definido após o outro e encadeado quando definido um dentro do outro. Veja o procedimento seguinte chamado "**RODAFLOR**" que faz uso dos comandos **FOR**, **WHILE** e **REPEAT** na forma sequencial e encadeado. Assim sendo, entre o seguinte código:

```
TO RODAFLOR
  FOR [I 1 6] [
    MAKE "I 1
    WHILE [:I <= 120] [
      FD 1 RT 1
      MAKE "I :I + 1
    ]
    RT 60
    REPEAT 180 [
      FD 1 RT 1
    ]
    RT 60
  ]
END
```

Em seguida execute as instruções **CS** e "**RODAFLOR**" e veja a imagem apresentada semelhante à figura 4.25.

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

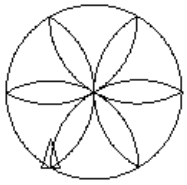


Figura 4.25 - imagem gerada com repetições sequenciais e encadeadas.

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

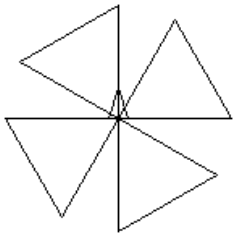
OBS:	O ambiente "*UCBLogo*" executa os procedimentos desenvolvidos de maneira rápida. Apesar dessa rapidez ser importante no processamento dos dados elaborados por computadores, ela pode atrapalhar um pouco a percepção de como os desenhos são efetivamente feitos. Desta forma, pode-se pedir para que o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

4.7 - Exercícios de fixação

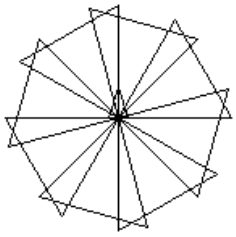
1. Criar procedimento chamado **RETAMETA** que desenhe um retângulo cujo lado menor (comprimento) seja a metade do lado maior (altura) onde o tamanho informado deverá ser fornecido como parâmetro. Movimente o desenho para frente com sentido a direita.

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

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

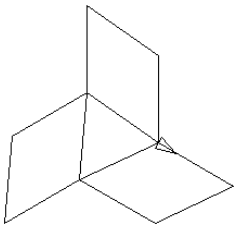


4. Criar procedimento chamado **VENTITRIG** desenhado a partir do procedimento **TRIANGEX** com giro a esquerda de modo que tenha a seguinte aparência.

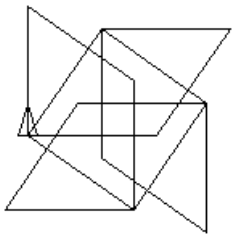


5. Criar procedimento chamado **LOSANGO** que desenhe imagem de mesmo nome com tamanho fornecido por parâmetro a partir de giro a direita.

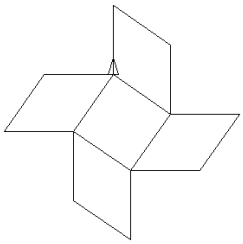
6. A partir do procedimento **LOSANGO** crie procedimento chamado **FLORLOSA1** de modo que seja apresentada a figura a seguir com tamanho "**80**".



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



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



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

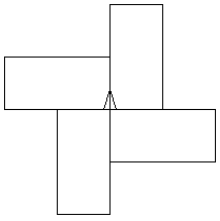
```
REPEAT 12 [REPEAT 3 [FD 50 RT 120] RT 30]
```

_____

_____

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

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



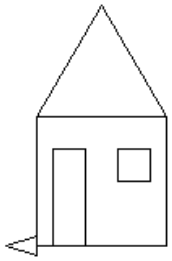
12. Criar procedimento chamado **TRIANGPR** que desenhe um triângulo equilátero com tamanho definido por parâmetro com comando **FOR** contando de "**0**" a "**2**" no sentido a frente com giro a direita.

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

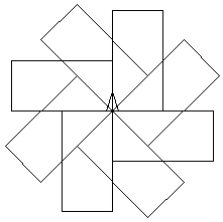
14. Criar procedimento chamado **QUADRADO2** que desenhe um quadrado com lado definido por parâmetro a partir do uso da primitiva **FOR**. Conte de "**1**" a "**4**".

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

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



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



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

```
TO QUADRO :TAMANHO
  REPEAT 4 [
    FD :TAMANHO
    RT 90
  ]
  RT 45
  FD :TAMANHO * 7 / 5
  BK :TAMANHO * 7 / 5
  LT 45
  FD :TAMANHO
  RT 135
  FD :TAMANHO * 7 / 5
  BK :TAMANHO * 7 / 5
END
```

_____

_____