
Apêndice A - Exemplos geométricos

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

INSTRUÇÕES DIRETAS COM E SEM AUXILIO DE PROCEDIMENTO

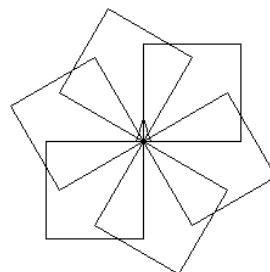
Parte das figuras apresentadas usarão o procedimento coringa chamado **FIGURAS** (figuras geométricas simples) que poderá desenhar imagens, de triângulos a circunferências.

```
TO FIGURAS :LADOS :TAMANHO  
  REPEAT :LADOS [FD :TAMANHO RT 360 / :LADOS]  
END
```

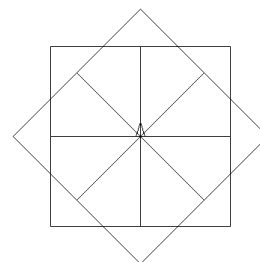
Desta forma, caso deseje a imagem de um quadrado execute "**FIGURAS 4 80**", caso deseje a imagem de um triângulo execute "**FIGURAS 3 80**", desejando uma circunferência execute "**FIGURAS 360 1**" e assim por diante.

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

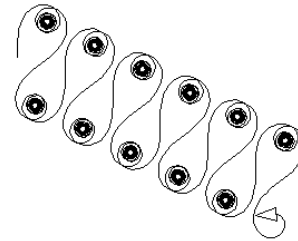
```
CS  
REPEAT 6 [RT 60 FIGURAS 4 80]
```



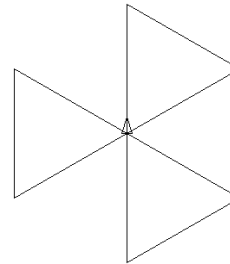
```
CS  
REPEAT 8 [FIGURAS 4 120 RT 45]
```



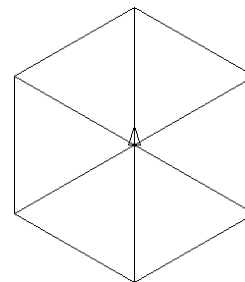
```
CS
FOR [I 0 2000] [FD 5 RT (90 * SIN :I)]
```



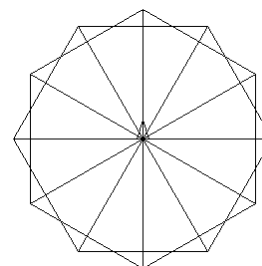
```
CS
REPEAT 3 [FIGURAS 3 150 RT 120]
```



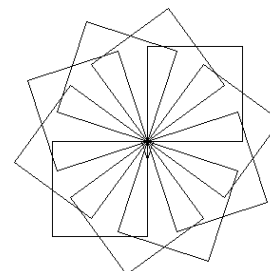
```
CS
REPEAT 6 [FIGURAS 3 140 RT 60]
```



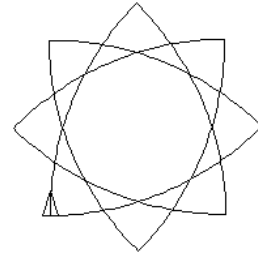
```
CS
REPEAT 360 / 30 [FIGURAS 3 130 RT 30]
```



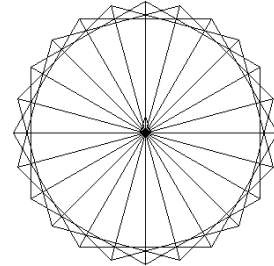
```
CS
REPEAT 10 [REPEAT 4 [FD 100 RT 90] RT 36]
```



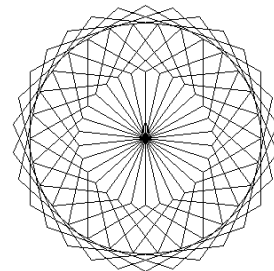
```
CS
REPEAT 8 [REPEAT 45 [FD 4 RT 1] RT 90]
```



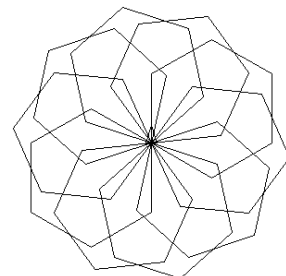
```
CS
REPEAT 24 [REPEAT 3 [FD 150 RT 120] RT 15]
```



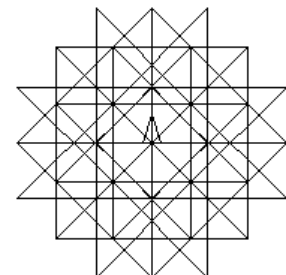
```
CS
REPEAT 30 [REPEAT 6 [FD 75 RT 60] RT 12]
```



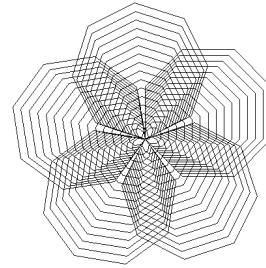
```
CS
REPEAT 10 [REPEAT 6 [FD 75 RT 60] RT 36]
```



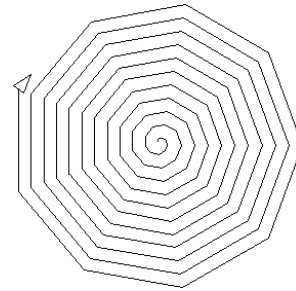
```
CS
REPEAT 8 [REPEAT 8 [LT 135 FD 90] LT 45]
```



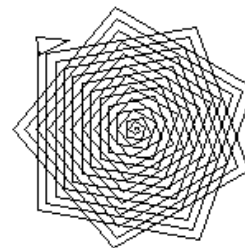
```
CS
FOR [I 10 80 5] [
  REPEAT 5 [
    REPEAT 8 [
      FD :I RT 45
    ]
    RT 72
  ]
]
```



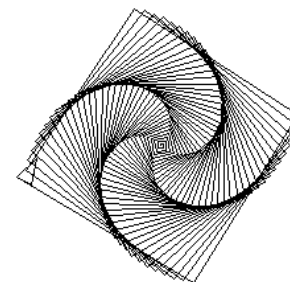
```
CS
REPEAT 100 [FD REPCOUNT RT 40]
```



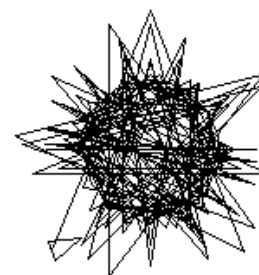
```
CS
REPEAT 100 [FD REPCOUNT RT 80]
```



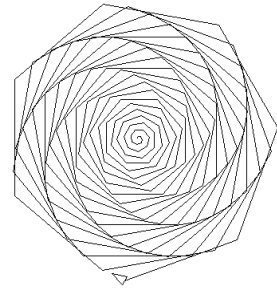
```
CS
REPEAT 150 [FD REPCOUNT RT 89]
```



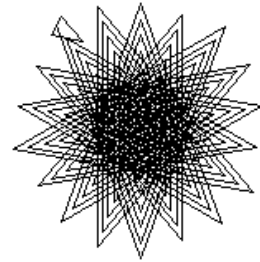
```
CS
REPEAT 150 [FD REPCOUNT RT REPCOUNT * 1.5]
```



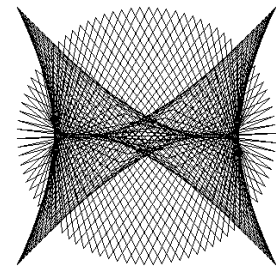
```
CS
REPEAT 150 [FD REPCOUNT RT 50]
```



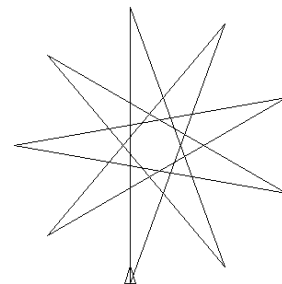
```
CS
REPEAT 150 [FD REPCOUNT RT 500]
```



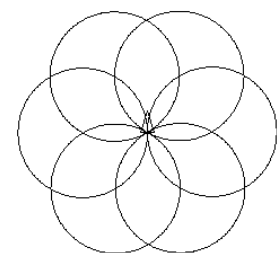
```
CS
REPEAT 360 [
  SETXY (SIN( 89 * REPCOUNT)) * 150
  (SIN(179 * REPCOUNT)) * 150
]
```



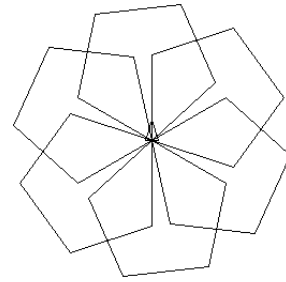
```
CS
REPEAT 9 [FD 300 RT 160]
```



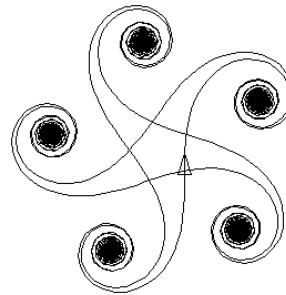
```
CS
REPEAT 6 [RT 60 FIGURAS 360 1]
```



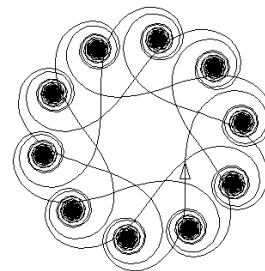
```
CS
REPEAT 6 [RT 60 FIGURAS 5 80]
```



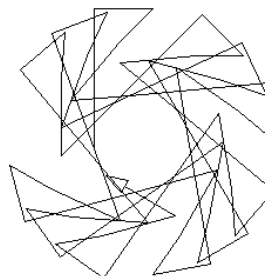
```
CS
REPEAT 1800 [FD 10 RT REPCOUNT + .1]
```



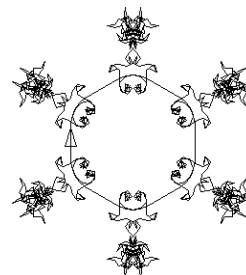
```
CS
REPEAT 3600 [FD 10 RT REPCOUNT + .2]
```



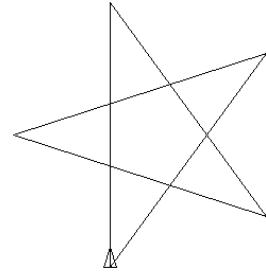
```
CS
REPEAT 12 [
  FD 120 RT 90 FD 50 RT 135 FD 40
  LT 185 BK 65 RT 45 FD 50
]
```



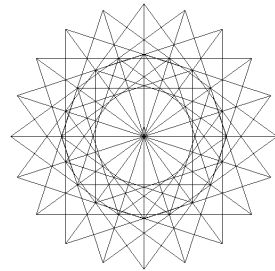
```
CS
FOR [I 0 1002] [
  FD 8 SETH (360 * (POWER :I 3) / 1002)
]
```



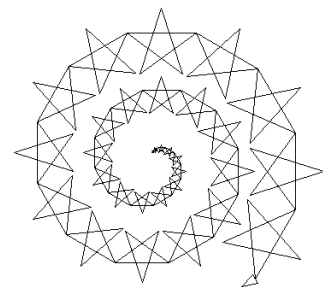
```
CS
REPEAT 5 [FD 250 RT 144]
```



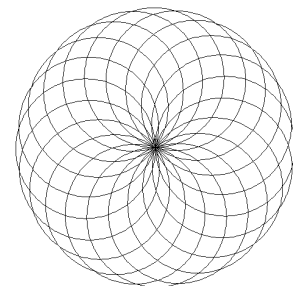
```
CS
REPEAT 20 [REPEAT 5 [FD 250 RT 144] RT 18]
```



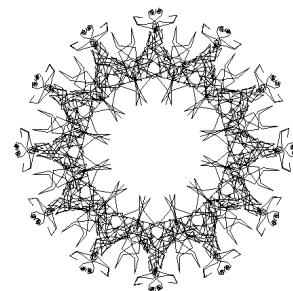
```
CS
FOR [I 0 95 3] [
  REPEAT 5 [
    FD :I RT 144
  ]
  FD :I RT 30
]
```



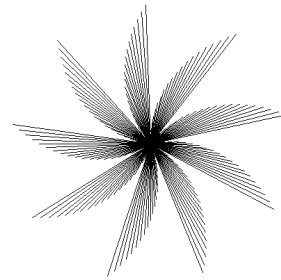
```
CS
REPEAT 20 [REPEAT 180 [FD 4 RT 2] RT 18]
```



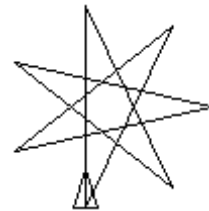
```
CS
FOR [I 0 2200] [
  FD (25 * SIN :I) RT (POWER :I 2)
]
```



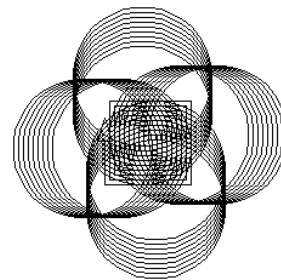
```
CS
REPEAT 9 [
  FOR [I 10 200 10] [
    FD :I BK :I RT 2
  ]
]
```



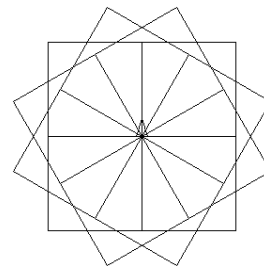
```
CS
REPEAT 7 [FD 100 RT 360 * 3 / 7]
```



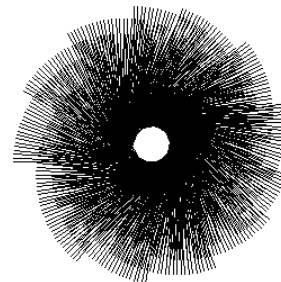
```
CS
REPEAT 36 [
  REPEAT 36 [
    FD 10 RT 10
  ]
  FD REPCOUNT RT 90 FD REPCOUNT
]
```



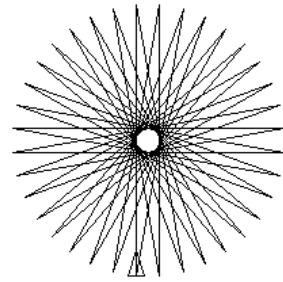
```
CS
REPEAT 12 [REPEAT 4 [FIGURAS 4 100] RT 30]
```



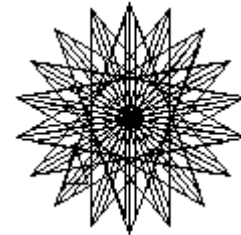
```
CS
REPEAT 12 [REPEAT 55 [FD 100 BK 100 RT 2] FD 45]
```



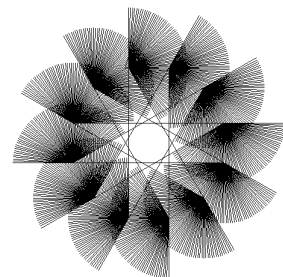

```
CS
REPEAT 54 [REPEAT 8 [FD 200 RT 170]]
```



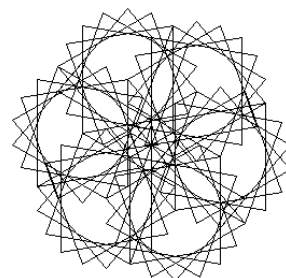
```
CS
REPEAT 18 [REPEAT 5 [RT 40 FD 100 RT 120] RT 20]
```



```
CS PU SETY 150 PD
REPEAT 12 [
  REPEAT 75 [
    FD 100 BK 100 RT 2
  ]
  FD 250
]
```



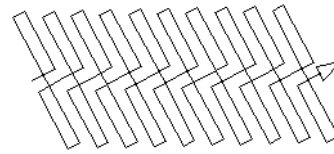
```
CS
REPEAT 12 [REPEAT 360 [FD 100 RT 100] RT 60]
```



INSTRUÇÕES A PARTIR DA DEFINIÇÃO DE PROCEDIMENTOS

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

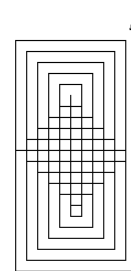
```
TO SEGMENTO
  FD 20 LT 90
  FD 50 LT 90
  FD 10 LT 90
  FD 55
  FD 55 RT 90
  FD 10 RT 90
  FD 50 RT 90
  FD 20
END
```



```
TO PADRAO
  RT 62
  REPEAT 10 [ SEGMENTO ]
END
```

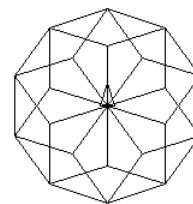
EXECUTE: CS PADRAO

```
TO CAMINHO
  REPEAT 22 [
    RT 90
    FD 110 - REPCOUNT * 10
    RT 90
    FD REPCOUNT * 10
  ]
END
```



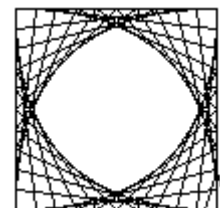
EXECUTE: CS CAMINHO

```
TO HEXAFLOR :PETALAS
  REPEAT :PETALAS [
    FIGURAS 5 50
    RT 360 / :PETALAS
  ]
END
```



EXECUTE: CS HEXAFLOR 10

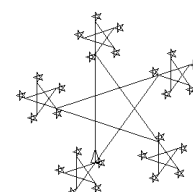
```
TO CICLO :INDICE :ULTIMO
  SETXY :INDICE 0
  SETXY :ULTIMO :INDICE
  SETXY (:ULTIMO - :INDICE) :ULTIMO
  SETXY 0 (:ULTIMO - :INDICE)
  SETXY :INDICE 0
END
```



```
TO QUADART
  REPEAT 10 [CICLO REPCOUNT * 10 100]
END
```

EXECUTE: CS QUADART

```
TO ESTRELANDO :TAMANHO :LIMITE
  IF :TAMANHO < :LIMITE [STOP]
  REPEAT 5 [FD :TAMANHO ESTRELANDO
    :TAMANHO * .3 :LIMITE RT 144]
END
```



EXECUTE: CS ESTRELANDO 150 10

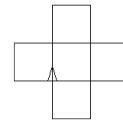
```
TO BANDEIRA
  FD 50
  FIGURAS 4 50
END
```

EXECUTE: CS BANDEIRA



```
TO CRUZ
  REPEAT 4 [BANDEIRA RT 90]
END
```

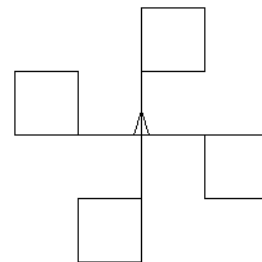
EXECUTE: CS CRUZ



```
TO VOLTABANDA
  BANDEIRA
  BK 50
END
```

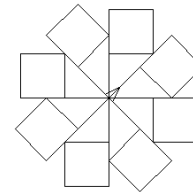
```
TO BANDEIRAS
  REPEAT 4 [VOLTABANDA RT 90]
END
```

EXECUTE: CS BANDEIRAS



```
TO MUITASBANDEIRAS
  BANDEIRAS
  RT 45
  BANDEIRAS
END
```

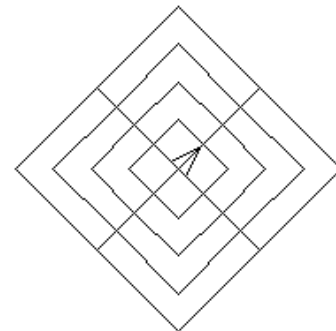
EXECUTE: CS MUITASBANDEIRAS



```
TO QUADRADOS
  FIGURAS 4 20
  FIGURAS 4 35
  FIGURAS 4 50
  FIGURAS 4 65
END
```

```
TO DIAMANTES
  RT 45
  REPEAT 4 [QUADRADOS RT 90]
END
```

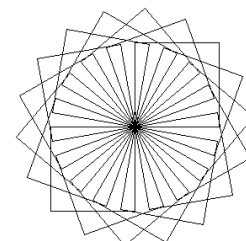
EXECUTE: CS DIAMANTES



```
TO QUADRADO
  REPEAT 4 [FIGURAS 4 100]
END
```

```
TO FLORQUADRADA
  REPEAT 18 [QUADRADO RT 20]
END
```

EXECUTE: CS FLORQUADRADA



```

TO BANDTRI
  FD 100 RT 120
  FD 30 RT 120
  FD 30 RT 120
  BK 70
END

TO FLORBANDTRI
  REPEAT 20 [BANDTRI WAIT 30 RT 18]
END

```

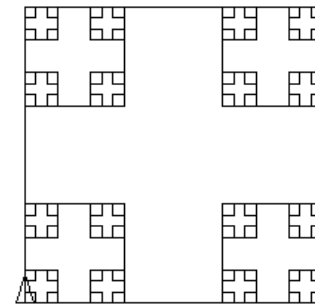


EXECUTE: CS FLORBANDTRI

```

TO QUADRICULADO :VALOR
  IF (:VALOR < 3) [
    STOP
  ]
  REPEAT 4 [
    QUADRICULADO :VALOR / 3
    FD :VALOR
    RT 90
  ]
END

```

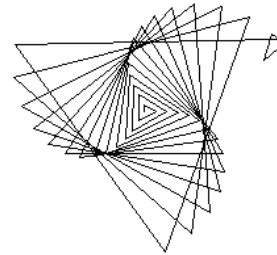


EXECUTE: CS QUADRICULADO 200

```

TO TUNELTRI :TAMANHO :ANGULO
  IF (:TAMANHO > 200) [
    STOP
  ]
  FD :TAMANHO
  RT :ANGULO
  TUNELTRI :TAMANHO + 5 :ANGULO + 0.12
END

```

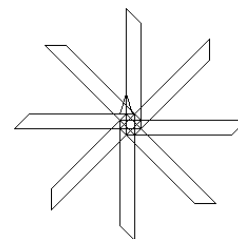


EXECUTE: CS TUNELTRI 5 120

```

TO PA
  REPEAT 2 [FD 100 RT 135 FD 20 RT 45]
END

```



```

TO HELICE
  REPEAT 8 [PA RT 135 FD 20]
END

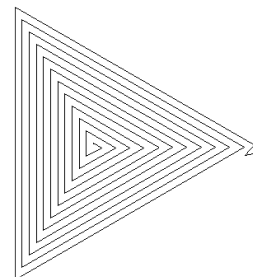
```

EXECUTE: CS HELICE

```

TO ESPIRALTRI :LADO
  IF (:LADO > 350) [
    STOP
  ]
  FD :LADO WAIT 20
  RT 120 WAIT 20
  ESPIRALTRI :LADO + 10 WAIT 30
END

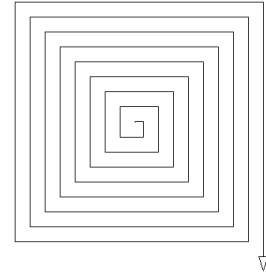
```



EXECUTE: CS ESPIRALTRI 1

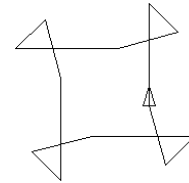
```
TO ESPIRALQUAD :LADO
  IF (:LADO > 350) [
    STOP
  ]
  FD :LADO WAIT 20
  RT 90 WAIT 20
  ESPIRALQUAD :LADO + 10 WAIT 30
  RT 90 WAIT 50
END
```

```
EXECUTE: CS ESPIRALQUAD 1
```



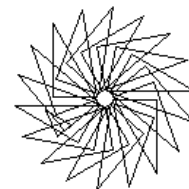
```
TO FORMA
  FD 100 RT 135
  FD 40 RT 120
  FD 60 RT 15
END
```

```
EXECUTE: CS REPEAT 4 [FORMA]
```



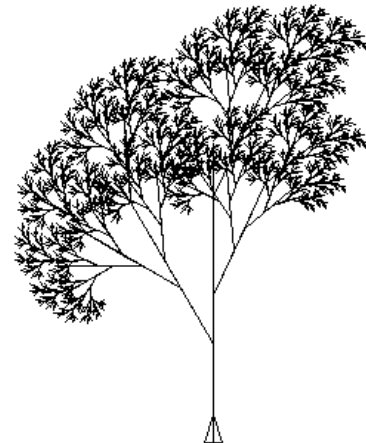
```
TO FORMATRI
  FD 50 RT 150
  FD 60 RT 100
  FD 30 RT 90
END
```

```
EXECUTE: CS REPEAT 20 [FORMATRI]
```



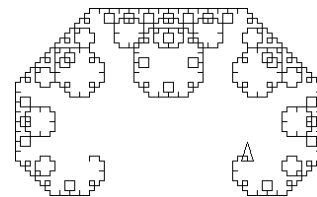
```
TO ARVORE :TAMANHO
  IF :TAMANHO < 5 [FD :TAMANHO BK :TAMANHO STOP]
  FD :TAMANHO / 3
  LT 30
  ARVORE :TAMANHO * 2 / 3
  RT 30
  FD :TAMANHO / 6
  RT 25
  ARVORE :TAMANHO / 2
  LT 25
  FD :TAMANHO / 3
  RT 25
  ARVORE :TAMANHO / 2
  LT 25
  FD :TAMANHO / 6
  BK :TAMANHO
END
```

```
EXECUTE: CS ARVORE 200
```



```
TO CURVAC :TAMANHO :NIVEL
  IF :NIVEL = 0 [FD :TAMANHO STOP]
  CURVAC :TAMANHO :NIVEL - 1 RT 90
  CURVAC :TAMANHO :NIVEL - 1 LT 90
END
```

```
EXECUTE: CS CURVAC 5 10
```



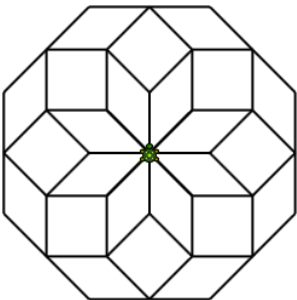
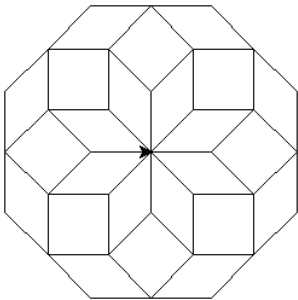
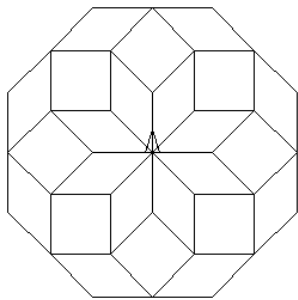
Grave este conteúdo com o nome "**Apendice_A**".

ANOTAÇÕES

[illegible]

Apêndice B - Espiral hexagonal (imagem da capa)

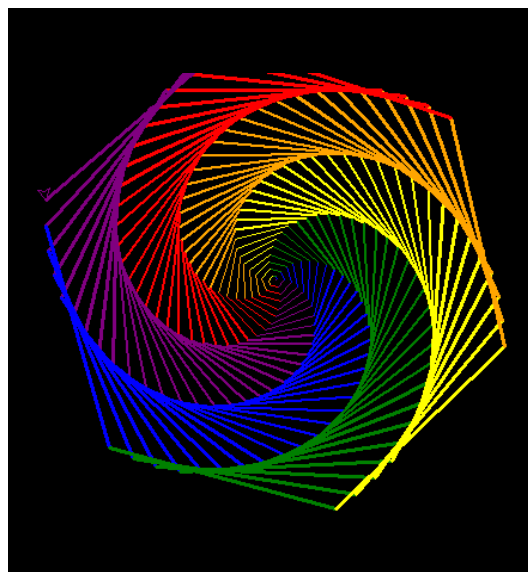
Muitos dos recursos de uso da linguagem Logo podem ser suportados em outras linguagens de programação. Por exemplo, a linguagem **"Small Basic"** desenvolvida pela empresa *Microsoft* (<https://smallbasic-publicwebsite.azurewebsites.net>) e **"Python"** produzida pela comunidade *Python* (<https://www.python.org>),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

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

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

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

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



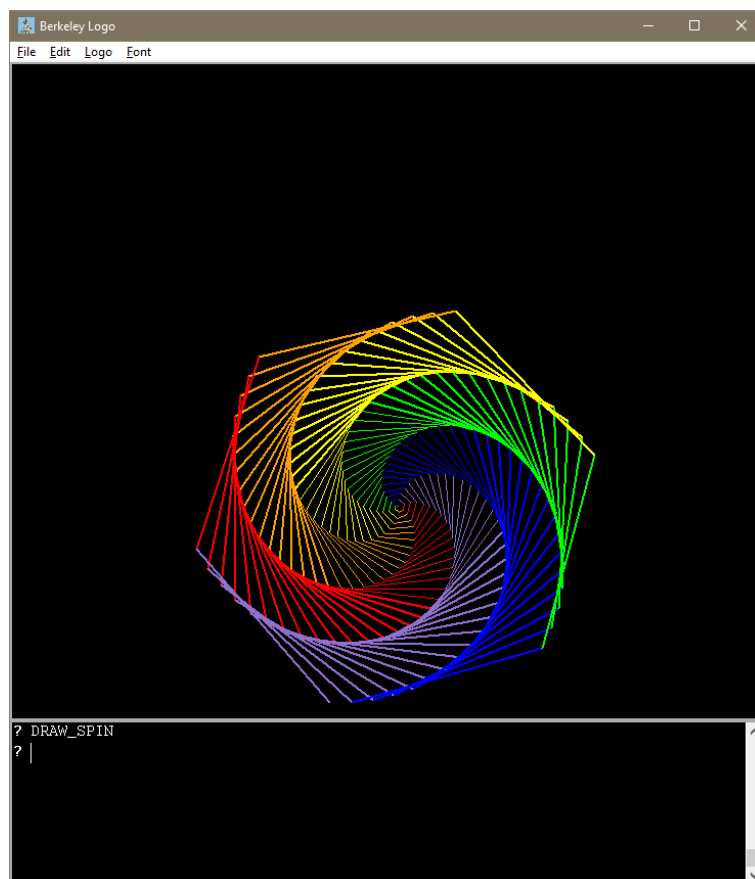
A partir do código **"Python"** foi realizada a transliteração e escrita de um código compatível em **"Logo"** que gerou a imagem usada na capa deste trabalho com o código:

```

TO DRAW_SPIN
  CS ST
  SETBG 0 ; PRETO
  FOR [X 0 199] [
    IF (MODULO :X 6) = 0 [SETPC 4] ; 0 - VERMELHO
    IF (MODULO :X 6) = 1 [SETPC 13] ; 1 - ROXO
    IF (MODULO :X 6) = 2 [SETPC 1] ; 2 - AZUL
    IF (MODULO :X 6) = 3 [SETPC 2] ; 3 - VERD
    IF (MODULO :X 6) = 4 [SETPC 6] ; 4 - AMARELO
    IF (MODULO :X 6) = 5 [SETPC 14] ; 5 - LARANJA
    SETPENSIZE INT (:X / 100 + 1)
    FD :X
    LT 59
  ]
  HT
  SETPENSIZE 0
END

```

O código do procedimento **"DRAW_SPIN"** produzido em **"Logo"** ao ser executado mostra imagem semelhante ao código original produzido em **"Python"**.



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

- **"TO"** e **"END"** são usadas para definir o escopo de escrita do programa no procedimento a partir do nome indicado a frente de **"TO"**, sendo isso compatível ao comando **"def"**;

- "**CS**" e "**ST**" efetuam respectivamente a limpeza da tela e a apresentação do ícone da tartaruga não tendo nenhuma relação direta com o código *Python* indicado;
- "**SETBG**" efetua a mudança da cor do fundo da área de trabalho para a cor preto de acordo com o código "**0**" similar a instrução "**turtle.bgcolor('black')**";
- "**FOR**" efetua a execução de duzentas passagens do grupo de instruções subordinadas contadas de "**0**" até "**199**" antes de encontrar a primitiva "**HT**" estando de acordo com a instrução "**for x in range(200):**" que efetivamente faz contagem de "**0**" até "**199**";
- Dentro do escopo de ação da primitiva "**FOR**" encontra-se definida uma sequência de instruções "**IF**" que detectam os valores do resto da divisão do conteúdo da variável "**X**" por "**6**" (que é a quantidade de cores em uso) que geram valores de resto entre "**0**" e "**5**" para a detecção e uso da cor definida para cada linha traçada do hexágono, estando este conjunto de instruções consoante a instrução "**turtle.pencolor(colores[x % 6])**". Logo não opera com o uso de variáveis compostas (matrizes) como *Python* "**colores[x % 6]**" por esta razão o uso das primitivas "**IF**" é necessário;
- "**SETPENSIZE INT (:X / 100 + 1)**" tem por finalidade a partir da primitiva "**MUDEESPL**" mudar a largura do traço do desenho em andamento estando esta instrução em consonância com a instrução "**turtle.width(x/100 + 1)**". O uso da função "**INT**" torna-se necessário pelo fato do valor a ser informado a primitiva "**SETPENSIZE**" deve ser expresso como número inteiro;
- "**PF :X**" faz a apresentação do traço com valores crescentes na variável "**X**" de "**0**" até "**199**" sendo compatível com a instrução "**turtle.forward(x)**";
- "**PE 59**" faz o giro em graus do traço para a formação de uma figura hexagonal, pois "**59**" é o valor numérico mais próximo de "**60**" que são os graus de formação de um hexágono estando de acordo com a instrução "**turtle.left(59)**";
- As demais instruções "**HT**" e "**SETPENSIZE 0**" efetuam respectivamente o ocultamento da tartaruga e o retorno do traço ao seu modo padrão, não tendo nenhuma relação com as demais instruções do código em *Python* que não vem ao caso.

Grave este conteúdo com o nome "**Apendice_B**".

[illegible]

Apêndice C - Gabarito

CAPÍTULO 3

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

REPEAT 4 [FORWARD 100 RIGHT 90]

Quadrado

REPEAT 5 [FD 100 LT 72]

Pentágono

REPEAT 3 [BK 100 RT 120]

Triângulo

REPEAT 36 [FD 20 RT 10]

Circunferência

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

TO RETANGULO1

FD 60

RT 90

FD 100

RT 90

FD 60

RT 90

FD 100

RT 90

END

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

TO RETANGULO2

REPEAT 2 [

BK 60

LT 90

BK 100

LT 90

]

END

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

```
TO PENTAGONO
  FD 40
  RT 72
  FD 40
  RT 72
  FD 40
  RT 72
  FD 40
  RT 72
  FD 40
  RT 72
END
```

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

```
TO DECAGONO
  REPEAT 10 [FD 30 LT 36]
END
```

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

```
TO ICOSAGONO
  REPEAT 20 [BK 35 RT 18]
END
```

ou

```
TO ICOSAGONO
  REPEAT 20 [BK 35 LT 18]
END
```

CAPÍTULO 4

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

```

TO RETAMETA :TAMANHO
  REPEAT 2 [
    FD :TAMANHO RT 90
    FD :TAMANHO / 2 RT 90
  ]
END

```

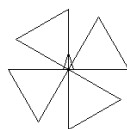
2. Crie um procedimento chamado **TRIANGEX**, que apresente um triângulo equilátero com lado de tamanho **70** para trás a partir de giros a esquerda.

```

TO TRIANGEX
  REPEAT 3 [
    BK 70 LT 120
  ]
END

```

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

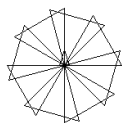


```

TO FLORTRIG
  REPEAT 4 [
    TRIANGEX RT 90
  ]
END

```

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



```

TO VENTITRIG
  REPEAT 8 [
    TRIANGEX LT 45
  ]
END

```

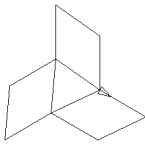
5. Criar procedimento chamado **LOSANGO** que desenhe imagem de mesmo nome com tamanho fornecido por parâmetro a partir de giro a direita.

```

TO LOSANGO :TAMANHO
  REPEAT 2 [
    FD :TAMANHO RT 125
    FD :TAMANHO RT 55
  ]
  LT 55 BK :TAMANHO RT 55
END

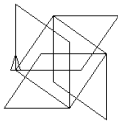
```

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



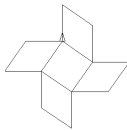
```
TO FLORLOSA1
  RT 120
  REPEAT 3 [
    LOSANGO 80 RT 120
  ]
END
```

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



```
TO FLORLOSA2
  REPEAT 4 [
    LOSANGO 80 LT 90
  ]
END
```

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



```
TO FLORLOSA3
  REPEAT 4 [
    LOSANGO 80 RT 90
  ]
END
```

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

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

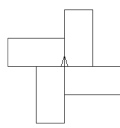
A imagem apresentada baseia-se em um quadrado repetido três vezes que após ser

Rotacionado a trinta graus é repetido mais doze vezes.

10. Criar procedimento chamado **RETANGPDO3** com tamanhos **100** (altura) e **50** (largura). Movimente-se para frente com giro a direita.

```
TO RETANGPDO3
  REPEAT 2 [
    FD 100
    RT 90
    FD 50
    RT 90
  ]
END
```

11. Criar procedimento chamado **CATAVENTO1** com o formato da figura seguinte a partir do procedimento **RETANGPDO3**.



```
TO CATAVENTO1
  REPEAT 4 [
    RETANGPDO3 RT 90
  ]
END
```

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.

```
TO TRIANGPR :TAMANHO
  FOR [0 0 2] [
    FD :TAMANHO RT 120
  ]
END
```

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**.

```
TO QUADRADO1 :TAMANHO
  MAKE "I 1
  WHILE [:I <= 4] [
    FD :TAMANHO RT 90
    MAKE "I :I + 1
  ]
END
```

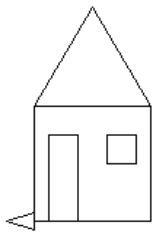
14. Criar procedimento chamado **QUADRADO2** que desenhe um quadrado com lado definido por parâmetro a partir do uso da primitiva **FOR**. Conte de **1** a **4**.

```
TO QUADRADO2 :TAMANHO
  FOR [I 1 4] [
    FD :TAMANHO RT 90
  ]
END
```

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

```
TO QUADRADO3 :TAMANHO
  MAKE "I 1
  DO.UNTIL[
    FD :TAMANHO RT 90
    MAKE "I :I + 1
  ] [:I > 4]
END
```

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**.

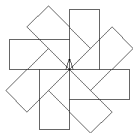


```

TO CASINHA
  PD
  FOR [I 1 4] [FD 80 RT 90]
  FD 80 RT 90
  FOR [I 1 2] [FD 80 LT 120]
  FD 80 SETPOS [0 0]
  RT 150 PU SETPOS [10 0]
  PD
  FD 60 RT 90
  FD 20 RT 90
  FD 60 RT 180
  PU
  SETPOS [50 60]
  PD
  RT 90
  FOR [I 1 4] [FD 20 RT 90]
  PU
  HOME
  LT 90
  PD
END

```

17. Criar procedimento chamado **CATAVENTO2** com o formato da figura seguinte a partir do procedimento **RETANGPDO3**.



```

TO CATAVENTO2
  REPEAT 8 [
    RETANGPDO3 RT 45
  ]
END

```

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

```

Mostra um quadrado com divisões perpendiculares dando um efeito de quadrado

criado a partir de quatro triângulos.

CAPÍTULO 5

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

```
TO CAP0501
  LOCAL "N
  PR [Entre valor para o calculo:]
  MAKE "N RW
  PR (SE "Resultado "= POWER INT :N 2)
END
```

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

```
TO CAP0502
  LOCAL "N
  LOCAL "R
  PR [Entre valor para o calculo:]
  MAKE "N RW
  MAKE "R POWER INT :N 3
  PR (SE "Resultado "= :R)
END
```

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

```
TO CAP0503
  LOCAL "C
  LOCAL "F
  PR [Entre valor da temperatura em graus Celsius:]
  MAKE "C RW
  MAKE "F :C * 9 / 5 + 32
  PR (SE [Temperatura em Fahrenheit] "= FORM :F 0 2)
END
```

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

```
TO CAP0504
  LOCAL "F
  LOCAL "C
  PR [Entre valor da temperatura em graus Fahrenheit:]
  MAKE "F RW
  MAKE "C ((:F - 32) * 5) / 9
  PR (SE [Temperatura em Celsius] "= FORM :C 0 2)
END
```

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

```
TO CAP0505
  LOCAL "A
  LOCAL "B
  LOCAL "R
  PR [Entre valor o valor <A>:]
  MAKE "A RW
  PR [Entre valor o valor <B>:]
  MAKE "B RW
  MAKE "R POWER (INT :A - INT :B) 2
  PR (SE [Quadrado da diferenca] "= :R)
END
```

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

```
TO CAP0506
  LOCAL "N
  LOCAL "R
  PR [Entre valor numerico inteiro:]
  MAKE "N RW
  MAKE "R (INT :N) * 2
  IF :R > 30 [
    PR (SE "Resultado "= :R)
  ]
END
```

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

```
TO CAP0507
  LOCAL "A
  LOCAL "B
  PR [Entre valor numerico real <A>:]
  MAKE "A RW
  PR [Entre valor numerico real <B>:]
  MAKE "B RW
  IFELSE :A > :B [
    PR (SE "Resultado "= :A - :B)
  ][
    PR (SE "Resultado "= :B - :A)
  ]
END
```

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

```
TO CAP0508
  LOCAL "A
  LOCAL "B
  LOCAL "R
  PR [Entre valor numerico real <A>:]
  MAKE "A RW
  PR [Entre valor numerico real <B>:]
  MAKE "B RW
  IFELSE :A > :B [
    MAKE "R :A - :B
  ][
    MAKE "R :B - :A
  ]
  PR (SE "Resultado "= :R)
END
```

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

```
TO CAP0509
  LOCAL "A
  LOCAL "B
  LOCAL "C
  LOCAL "R
  PR [Entre valor numerico real <A>:]
  MAKE "A RW
  PR [Entre valor numerico real <B>:]
  MAKE "B RW
  PR [Entre valor numerico real <C>:]
  MAKE "C RW
  MAKE "R INT :A + INT :B + INT :C
  IF :R >= 100 [
    PR (SE "Resultado "= :R)
  ]
END
```

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

```
TO CAP0510
  LOCAL "S
  MAKE "S 0
  REPEAT 100 [
    MAKE "S :S + REPCOUNT
  ]
  PR (SE "Somatorio: "= :S)
END
```

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

```
TO CAP0511
  LOCAL "S
  MAKE "S 0
  LOCAL "I
  MAKE "I 1
  WHILE [:I <= 100] [
    MAKE "S :S + :I
    MAKE "I :I + 1
  ]
  PR (SE "Somatorio: "= :S)
END
```

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

```

TO CAP0512
  LOCAL "S
  MAKE "S 0
  LOCAL "I
  MAKE "I 1
  DO.UNTIL[
    MAKE "S :S + :I
    MAKE "I :I + 1
  ] [:I > 100]
  PR (SE "Somatorio: "= :S)
END

```

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

```

TO CAP0513
  LOCAL "S
  MAKE "S 0
  FOR [I 1 100] [
    MAKE "S :S + :I
  ]
  PR (SE "Somatorio: "= :S)
END

```

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

```

TO CAP0514
  LOCAL "S
  MAKE "S 0
  REPEAT 500 [
    IF (MODULO REPCOUNT 2) = 0 [
      MAKE "S :S + REPCOUNT
    ]
  ]
  PR (SE "Somatorio: "= :S)
END

```

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

```
TO CAP0515
  LOCAL "S
  MAKE "S 0
  LOCAL "I
  MAKE "I 1
  WHILE [:I <= 500] [
    IF (MODULO :I 2) = 0 [
      MAKE "S :S + :I
    ]
    MAKE "I :I + 1
  ]
  PR (SE "Somatorio: "= :S)
END
```

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

```
TO CAP0516
  LOCAL "S
  MAKE "S 0
  LOCAL "I
  MAKE "I 1
  DO.UNTIL[
    IF (MODULO :I 2) = 0 [
      MAKE "S :S + :I
    ]
    MAKE "I :I + 1
  ] [:I > 500]
  PR (SE "Somatorio: "= :S)
END
```

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

```
TO CAP0517
  LOCAL "S
  MAKE "S 0
  FOR [I 1 500] [
    IF (MODULO :I 2) = 0 [
      MAKE "S :S + :I
    ]
  ]
  PR (SE "Somatorio: "= :S)
END
```

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

```
TO CAP0518
  REPEAT 19 [
    IF (MODULO REPCOUNT 4) = 0 [
      PR REPCOUNT
    ]
  ]
END
```

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

```
TO CAP0519
  LOCAL "I
  MAKE "I 1
  WHILE [:I <= 19] [
    IF (MODULO :I 4) = 0 [
      PR :I
    ]
    MAKE "I :I + 1
  ]
END
```

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

```
TO CAP0520
  LOCAL "I
  MAKE "I 1
  DO.UNTIL[
    IF (MODULO :I 4) = 0 [
      PR :I
    ]
    MAKE "I :I + 1
  ] [:I > 19]
END
```

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

```
TO CAP0521
  FOR [I 1 19] [
    IF (MODULO :I 4) = 0 [
      PR :I
    ]
  ]
END
```

22. Criar procedimento chamado **CAP0522** que apresente os valores numéricos inteiros compreendidos na faixa de "0" e "4". Use a primitiva "REPEAT".

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

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

```
TO CAP0522
  LOCAL "I
  MAKE "I 0
  REPEAT 5 [
    PR :I
    MAKE "I :I + 1
  ]
END
```

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

dando um estilo de "WHILE" ou "DO.PUTIL". Então é melhor usar as primiti-

vas "WHILE" ou "DO.PUTIL" e resolver o problema de forma mais adequada.

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

```
TO CAP0523
  LOCAL "I
  MAKE "I 0
  WHILE [:I <= 4] [
    PR :I
    MAKE "I :I + 1
  ]
END
```

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

```
TO CAP0524
  LOCAL "I
  MAKE "I 0
  DO.UNTIL[
    PR :I
    MAKE "I :I + 1
  ] [:I > 4]
END
```


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

```
TO CAP0525
  FOR [I 0 4] [
    PR :I
  ]
END
```

26. Criar procedimento chamado **CAP0526** que apresente os valores numéricos inteiros compreendidos na faixa de "0" até "15" de "3" em "3". Use a primitiva "REPEAT".

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

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

"1" em "1".

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

```
TO CAP0527
  LOCAL "I
  MAKE "I 0
  WHILE [:I <= 15] [
    PR :I
    MAKE "I :I + 3
  ]
END
```

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

```
TO CAP0528
  LOCAL "I
  MAKE "I 0
  DO.UNTIL[
    PR :I
    MAKE "I :I + 3
  ] [:I > 15]
END
```

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

```
TO CAP0529
  FOR [I 0 15 3] [
    PR :I
  ]
END
```

ANOTAÇÕES

[illegible]

Bibliografia

ABELSON, H. **TI Logo**. New York: McGraw-Hill, 1984.

_____. **TI Logo Education**. Lubbock: Texas Instruments & McGraw-Hill, 1981.

ATARI. **ATARI Logo: Reference Manual**. Quebec: Logo Computer System, Inc., 1983.

BASS, J. H. **Logo Programming**. Logo Spoken Here. 2002. Disponível em: <<http://pages.intnet.mu/jhbpge/main.htm>>. Acesso 28 jun. 2021.

CATLIN, D. **My Personal Tribute to Seymour Papert**. GO Magazine. August, 2016. Disponível em: <<http://go.roamer-educational-robot.com/2016/08/12/my-personal-tribute-to-seymour-papert/>>. Acesso 26 jun. 2021.

CORRALES MORA, M. **Language Logo I: Descubriendo un mundo nuevo**. San José: EUNED, 1996.

DOWNEY, A. B. & GAY, G. **How to think like a Computer Scientist: Logo version**. GNU Edition, 2003. Disponível em: <<http://openbookproject.net/thinkcs/archive/logo/english/thinklgo.pdf>>. Acesso 29 jun. 2021.

ERFAN'S. **Welcome to Erfan's Zone of Logo**. Nestead. 2021. Disponível em: <<http://erfan96.50webs.com/>>. Acesso 28 jun. 2021.

HARVEY, B. **Computer science Logo style: Symbolic computing**. 2nd. Massachusetts: MIT Press. 1998, v. 1.

JOYS, D. **Hs-logo**. Logo Turtle Graphics Interpreter. 2021. Disponível em: <<https://deepakjois.github.io/hs-logo/>>. Acesso 28 jun. 2021.

KHERIATY, I. & GERHOLD, G. **Radio Shack color Logo**. Massachusetts: Micropi, 1982.

LOGO FOUNDATION. **What Is Logo?**. New York: Logo Foundation, 2021. Disponível em: <https://el.media.mit.edu/logo-foundation/>. Acesso em: 16 jun. 2021.

MULLER, J. **The Great Logo Adventure: Discovering Logo on and Off the Computer**. Madison, AL, United States: Doone Pubns, 1998.

PALEOTRONIC. **Past and Future Turtles - The Evolution of the Logo Programming Language: Part I**. Australia: Paleotronic Magazine, 2021. Disponível em: <https://paleotronic.com/2021/05/22/past-and-future-turtles-the-evolution-of-the-logo-programming-language-part-1/>. Acesso em: 16 jun. 2021.

PETTI, W. A. **Math Cats**. Disponível em: <<http://www.mathcats.com/>>. Acesso em 23 jun. 2021.

SOLOMON, C. J. (Et al). **History of Logo**. Proc. ACM Program. Lang. 4, HOPL, Article 79. June, 2020. Disponível em: <<https://dl.acm.org/doi/pdf/10.1145/3386329>>. Acesso em 28 jun. 2021.

SPARER, E. **Sinclair Logo 1 Turtle Graphics**. Cambridge: Sinclair Research Ltd., 1984.

TOMIYAMA, M. N. **Recursão**. Minas Gerais: Universidade Federal de Uberlândia - Faculdade de Computação, 2016. Disponível em: <<http://www.facom.ufu.br/~madriana/PF/recursao.pdf>>. Acesso em 22 jun. 2021.

WINTER, M. J. **The Commodore 64 Logo workbook**. Chatsworth: DATAMOST, 1984.

Referências bibliográficas

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

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

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

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

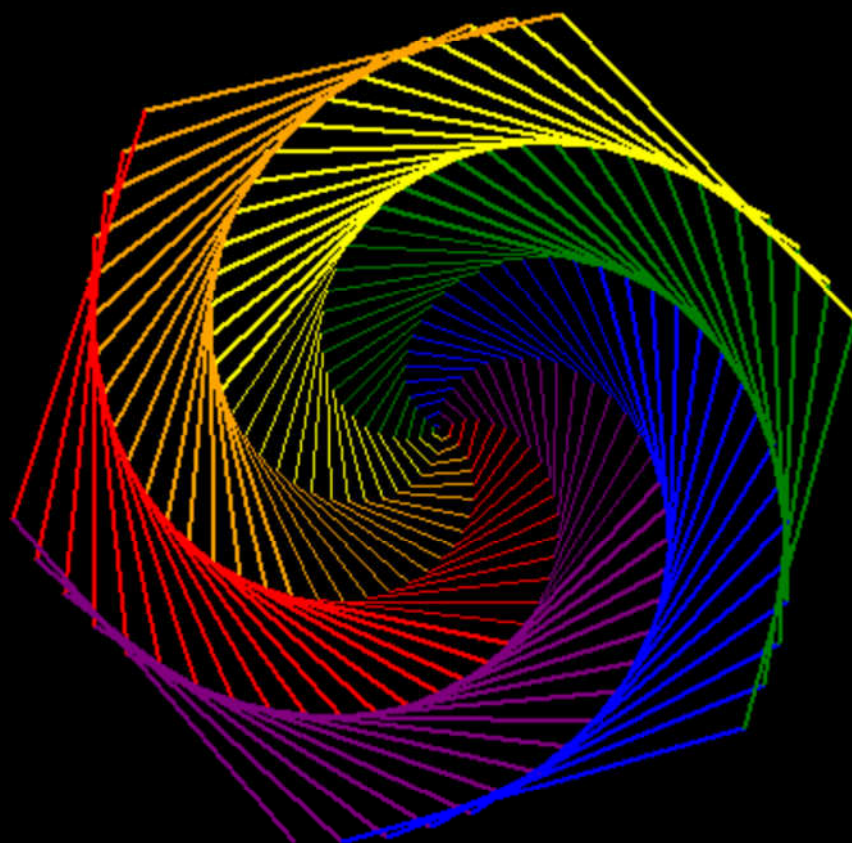
SASSENATH, C. **Amiga LOGO: Tutorial and reference**. Commodore-Amiga, Inc. and Carl Sassenrath, 1989.

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

ANOTAÇÕES

LINGUAGEM LOGO

Introdução com UCBLogo



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

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

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

