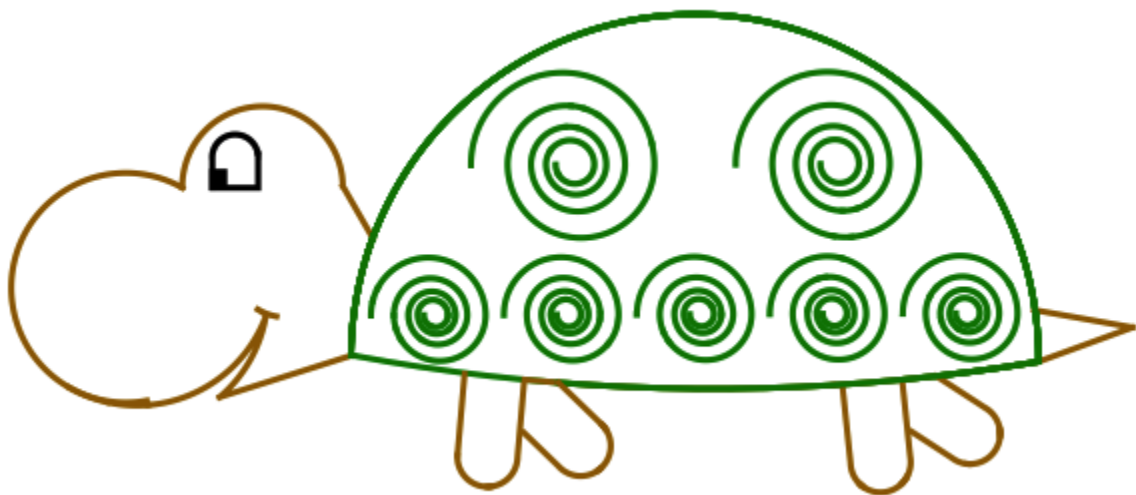


Heidi Gebauer  
Ivana Kosírová

Juraj Hromkovič  
Giovanni Serafini

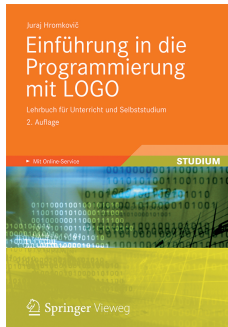
Lucia Keller  
Björn Steffen

# Programar com LOGO



## Programar com LOGO

Este caderno é uma versão abreviada das lições 1 a 7 do livro escolar *Einführung in die Programmierung mit LOGO*. O livro escolar contém exercícios e explicações adicionais. Além disso, está dotado com indicações para o professor. O livro escolar contém no total 15 lições.



Juraj Hromkovič. *Einführung in die Programmierung mit LOGO: Lehrbuch für Unterricht und Selbststudium*. 2. Aufl., Springer Vieweg 2012. ISBN: 978-3-8348-1852-2.

Versão 3.1, 8 de Abril de 2015, SVN-Rev: 16260

## Meio de programação

Os documentos de ensino presentes, foram desenvolvidos para o programa XLogo. XLogo está gratuitamente disponível no site [xlogo.tuxfamily.org](http://xlogo.tuxfamily.org).

Para que os programas do Logo, presentes nestes documentos, possam ser executados, XLogo tem de ser ajustado em inglês.

## Direito de uso

O ABZ põe este programa gratuitamente à disposição para o uso interno, para os professores e instituições interessadas, como apoio das aulas.

## ABZ

O Ausbildungs- und Beratungszentrum für Informatikunterricht da ETH Zúriquem suporta escolas e professores que querem criar ou desenvolver o seu ensino de informática com um sortimento variado. Vai da orientação e aulas individuais de um professor da ETH e da equipa do ABZ, diretamente no próprio local nas escolas, por cursos de instrução e formação complementar para professores, assim como material para as aulas.

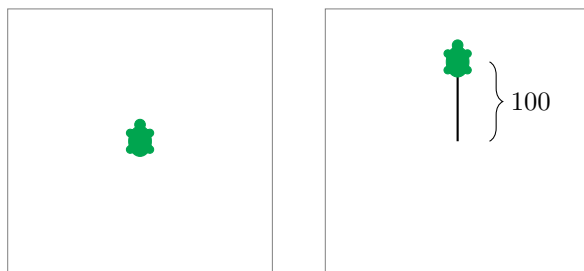
[www.abz.inf.ethz.ch](http://www.abz.inf.ethz.ch)

# 1 Comandos básicos

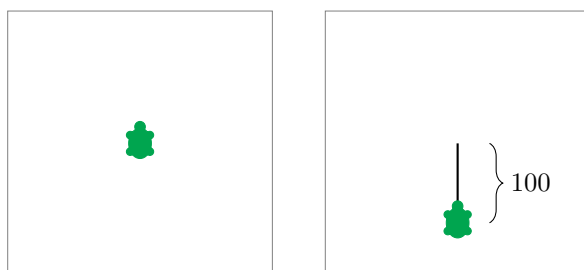
Um **comando de computador** é uma ordem que o computador é capaz de perceber e exercer. No fundo o computador só conhece poucos comandos e todas as atividades complicadas que nós queremos que o computador realize, temos de compor por comandos fáceis. Essa sequência de comandos tem o nome de **programa**. Há programas que são compostos por milhões de comandos. Nesse caso para não perder a visão geral, exige um procedimento calculado e claro que vamos aprender neste curso de programação.

## Desenhar linhas retas

Com o comando **forward 100** ou **fd 100**, comandas a tartaruga a avançar 100 passos em frente:



Com o comando **back 100** ou **bk 100**, a tartaruga recua 100 passos para trás:



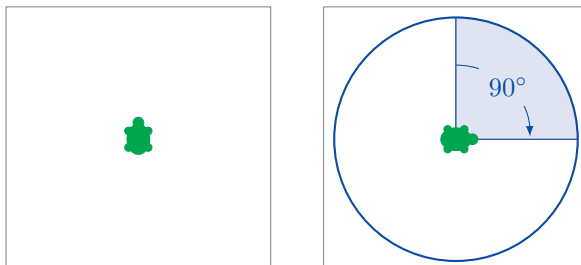
## Apagar e começar de novo

O comando **cs** apaga tudo o que foi desenhado e a tartaruga volta à posição inicial.

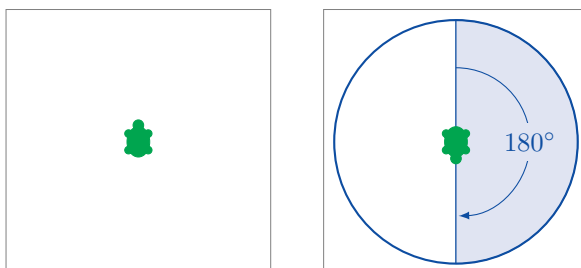
# Girar

A tartaruga sempre se move na direção em que está a olhar.

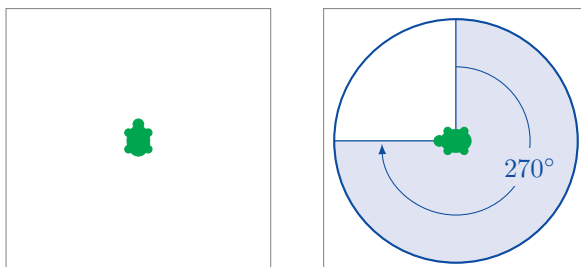
Com o comando **right 90** ou **rt 90**, a tartaruga gira-se  $90^\circ$  para a direita. Isto corresponde a um quarto de um círculo:



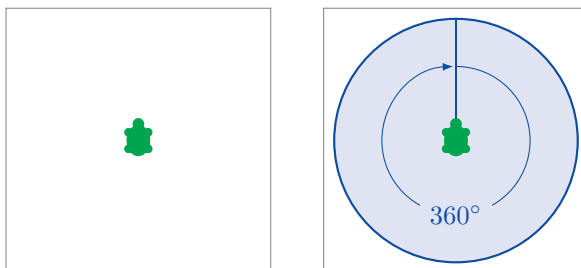
O comando **right 180** ou **rt 180** gira a tartaruga  $180^\circ$  para a direita. Isto corresponde a uma meia rotação:



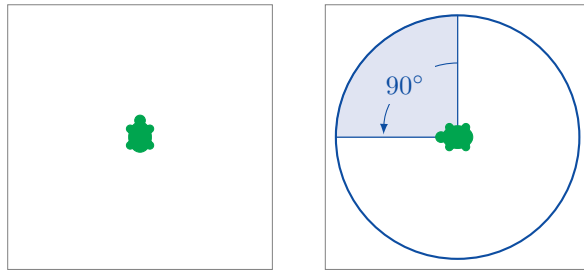
**right 270** ou **rt 270** gira a tartaruga  $270^\circ$  para a direita:



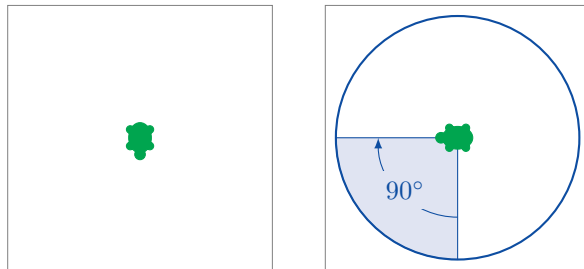
Os comandos **right 360** e **rt 360** giram a tartaruga  $360^\circ$  para a direita. Isto corresponde a uma rotação completa:



Com o comando **left 90** ou **lt 90**, a tartaruga gira-se  $90^\circ$  para a esquerda:



Repara que a rotação para a esquerda ou para a direita refere-se á visão da tartaruga, como se vê no seguinte exemplo com o comando **rt 90**:



## Programar

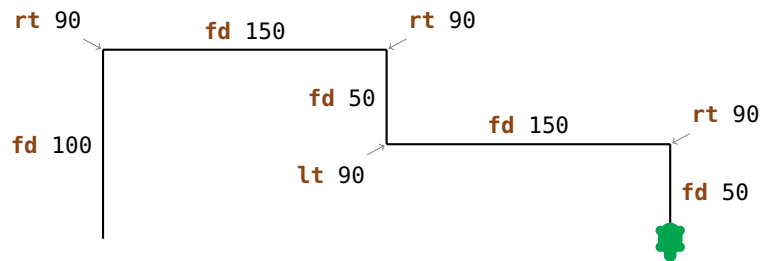
**Programar** significa escrever uma sequência de comandos um atrás do outro.

### Exercício 1

Escreve o seguinte programa e executa-o:

```
fd 100  
rt 90  
fd 150  
rt 90  
fd 50  
lt 90  
fd 150  
rt 90  
fd 50
```

Desenhaste a seguinte imagem?



### Exercício 2

Escreve o seguinte programa e executa-o:

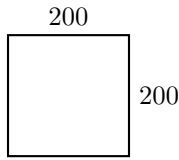
```
fd 100  
rt 90  
fd 200  
rt 90  
fd 80  
rt 90  
fd 100  
rt 90  
fd 50
```



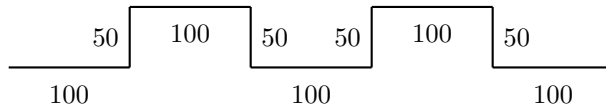
Desenha a imagem que surgiu ao lado do programa em cima e descreve (como no Exercício 1) o que foi que cada comando executou.

### Exercício 3

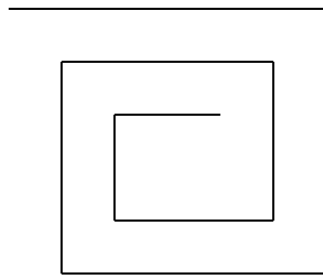
Escreve programas que desenham as seguintes imagens. Em todas as imagens podes escolher a posição do começo da tartaruga em relação à imagem própria que é para ser desenhada.



(a)

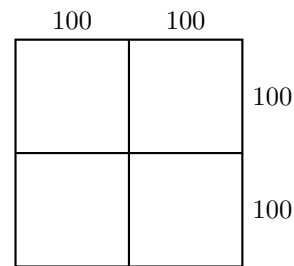


(b)



O tamanho podes escolher tu próprio.

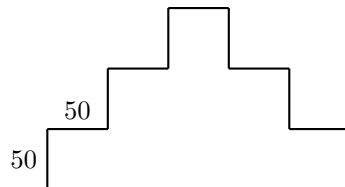
(c)



(d)

### Exercício 4

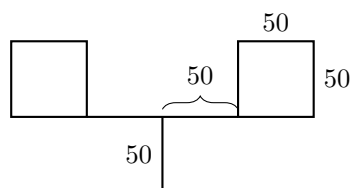
Escreve um programa que desenha a seguinte imagem:



Consegues alterar o teu programa para que tenhas de utilizar apenas os comandos **fd 50** e **rt 90**?

### Exercício 5

A Anna quer desenhar a seguinte imagem. Es capaz de a ajudar?





## 2 O comando **repeat**

Quando queremos desenhar um quadrado com um lado de comprimento 100,



podemos utilizar o seguinte programa:

```
fd 100  
rt 90  
fd 100  
rt 90  
fd 100  
rt 90  
fd 100  
rt 90
```

Reparamos que os dois comandos

```
fd 100  
rt 90
```

repetem-se quatro vezes. Não seria mais fácil dizer ao computador que repetisse esses dois comandos quatro vezes, em vez de escrever os comandos quatro vezes um depois do outro?

Precisamente isto podemos fazer de certa maneira:

<b>repeat</b>	4	[ <b>fd</b> 100 <b>rt</b> 90]
Comando para a repetição de um programa	Número de repetições	Série de comandos que são para ser repetidos

### Exercício 6

Escreve o seguinte programa e executa-o:

```
fd 75 lt 90  
fd 75 lt 90  
fd 75 lt 90  
fd 75 lt 90
```

O que desenha o programa? És capaz de utilizar o comando **repeat** para reduzir o tamanho do programa?

### Exercício 7

Escreve o seguinte programa para ver o que desenha:

```
fd 50 rt 60  
fd 50 rt 60  
fd 50 rt 60  
fd 50 rt 60  
fd 50 rt 60  
fd 50 rt 60
```

Escreve-o mais curto, aplicando o comando **repeat**.

### Exercício 8

Utiliza o comando **repeat**, para desenhar um quadrado com um lado de comprimento 200.

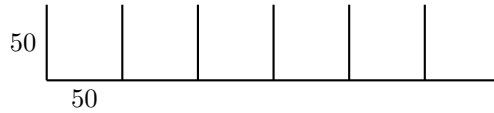
### Exercício 9

Escreve o seguinte programa e executa-o:

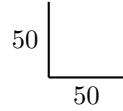
```
fd 100 rt 120  
fd 100 rt 120  
fd 100 rt 120
```

O que desenha o programa? És capaz de utilizar o comando **repeat** para reduzir o tamanho do programa?

Queremos agora desenhar a seguinte imagem com a ajuda do comando **repeat**:



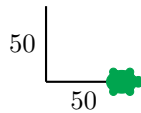
Antes de começar a desenhar, temos em primeiro lugar de pensar qual é o modelo que se repete. Por exemplo podemos utilizar a seguinte imagem como modelo que se repete:



Se começamos no canto inferior esquerdo, podemos desenhar o modelo com o seguinte programa:

```
fd 50 bk 50 rt 90 fd 50
```

Depois da execução deste programa, a tartaruga está situada como indicado na seguinte imagem e olha para a direita:



Isto é bom, porque já chegou à posição na qual podemos desenhar o modelo outra vez. A tartaruga já só precisa de olhar para cima, o que podemos atingir com o comando **lt 90**.

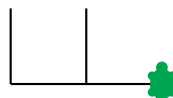
Executamos o programa para verificar se está correto:

```
fd 50 bk 50 rt 90 fd 50  
lt 90
```

Recebemos a situação desejada:



Se deixamos executar o mesmo programa de novo, recebemos:



Portanto, vemos que a nossa ideia funciona e podemos repetir o nosso programa 6 vezes:

```
repeat 6 [ fd 50 bk 50 rt 90 fd 50 lt 90 ]
```

Modelo                                  Ajustar

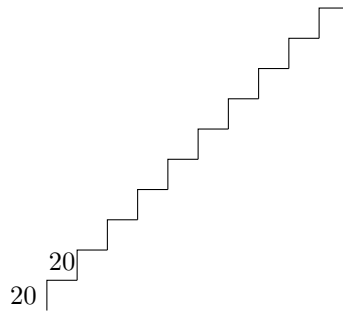
Muitos exercícios podemos resolver com este procedimento. Lembra-te que primeiro tens sempre de encontrar um modelo que se repete. Depois, tens de desenvolver por um lado um programa para o *modelo*, e por outro lado um programa para o *ajuste* da tartaruga para o próximo modelo. O teu programa vai ter a seguinte forma.

**repeat** *quantidade* [*modelo ajustar*]

### Exercício 10

Desenhar escadas.

(a) Desenha uma escada com 10 degraus do tamanho 20:



- Primeiro encontra um modelo que se repete e depois escreve um programa para esse caso.
- Inventa um programa para o ajuste da tartaruga, para que esta esteja bem situada para a próxima repetição do modelo.
- Depois junta os dois programas corretamente para resolver o exercício.

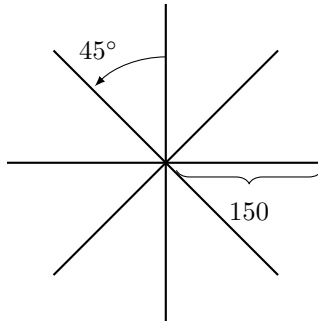
(b) Desenha uma escada com 5 degraus do tamanho 50.

(c) Desenha uma escada com 20 degraus do tamanho 10.

### Exercício 11

Agora queremos desenhar estrelas.

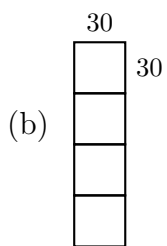
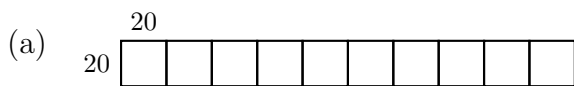
(a) Desenha a seguinte estrela:



(b) A estrela tem oito raios do tamanho 150. Também es capaz de desenhar uma estrela com 16 raios do tamanho 100?

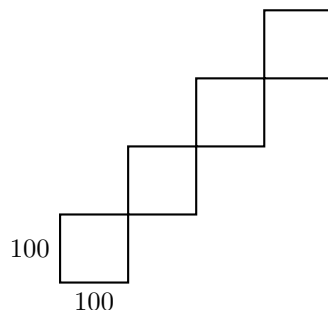
### Exercício 12

Desenha as seguintes imagens:



### Exercício 13

Desenha com um programa a seguinte imagem:



### Exercício 14

Faz uma cópia do seguinte programa e executa-o:

```
repeat 4 [fd 100 rt 90]
rt 90
repeat 4 [fd 100 rt 90]
rt 90
repeat 4 [fd 100 rt 90]
rt 90
repeat 4 [fd 100 rt 90]
rt 90
```

O que surge neste caso? És capaz de escrever este programa ainda mais curto?

## Modo caminhar

Normalmente a tartaruga está no **modo lápis**. Isto significa que tem um lápis na mão e sempre que se move também desenha.

No **modo caminhar** a tartaruga move-se no ecrã sem desenhá-lo. Ao modo caminhar chegas com o comando

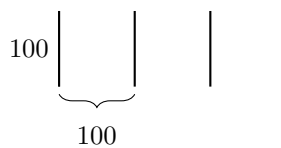
**penup** ou mais curto **pu**.

Do modo caminhar ao modo lápis voltas com o comando

**pendown** ou mais curto **pd**.

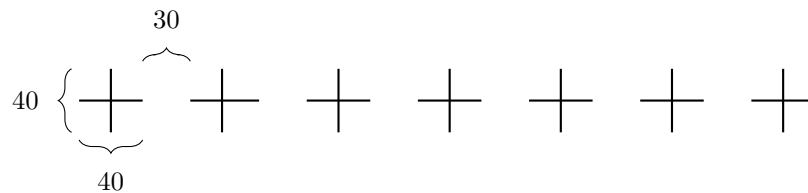
### Exercício 15

Desenha a seguinte imagem com um programa:



### Exercício 16

Escreve um programa para a seguinte imagem:



## 3 Nomear programas e chama-los

A todos os programas que escrevemos podemos dar um nome. Quando introduzimos o nome do programa na barra de comandos, a atividade do programa é efetuada.

O programa para desenhar um quadrado com um lado de comprimento 100 é:

```
repeat 4 [fd 100 rt 90]
```

Podemos dotar este programa com o seguinte nome **QUADRADO100**:

```
to QUADRADO100
repeat 4 [fd 100 rt 90]
end
```

Portanto escrevemos o mesmo programa duas vezes, uma vez com nome e outra vez sem ele.

Programas com nome, escrevemos no **editor**. Os tais programas estão assinalados neste caderno com uma caixa cinzenta. Assim que o programa esteja pronto, carrega no botão com a tartaruga para fechar o editor outra vez.

O nome pode ser escolhido individualmente, nós escolhemos **QUADRADO100**, porque queremos dar a entender que se trata de um quadrado com um comprimento de lado 100. A única condição para os nomes, é que têm de ser escritos com letras e números e numa só peça, sem intervalo.

No ecrã ainda não foi desenhado nada porque até agora ainda só demos um nome ao programa, mas ainda não o executamos. Se agora escrevemos o nome

**QUADRADO100**

na barra de comandos, o programa **repeat 4 [fd 100 rt 90]** será executado. No ecrã aparece:



Vamos olhar de novo para o Exercício 12(a). Podemos resolver este exercício mais fácil, se em primeiro lugar escrevemos um programa para o modelo repetitivo, ou seja, o quadrado com lado de comprimento 20, e lhe damos um nome:

```
to QUADRADO20
repeat 4 [fd 20 rt 90]
end
```

Depois de desenhar **QUADRADO20** a tartaruga está no canto inferior esquerdo do quadrado:



Para desenhar o próximo quadrado, ela tem de ir para o canto inferior direito. Vamos consegui-lo com o programa

```
rt 90 fd 20 lt 90
```

Vamos também nomear este programa:

```
to AJUSTAR20
rt 90 fd 20 lt 90
end
```

Com estes dois programas podemos escrever um programa para o Exercício 12(a) da seguinte forma:

```
repeat 10 [QUADRADO20 AJUSTAR20]
```

Também podemos nomear o nosso programa anterior. Por exemplo:

```
to FILA10
repeat 10 [QUADRADO20 AJUSTAR20]
end
```

Se o fizemos, chamamos os programas **QUADRADO20** e **AJUSTAR20** subprogramas do programa **FILA10**.

### Exercício 17

Escreve um programa para resolver o Exercício 12(b), que usa um programa que desenha quadrados com um lado de comprimento 30. O programa tem de parecer da seguinte forma:

```
repeat 4 [QUADRADO30 AJUSTAR30]
```

Portanto tens de escrever os subprogramas apropriados **QUADRADO30** e **AJUSTAR30**.

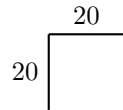


### Exercício 18

Aproveita o programa **QUADRADO100** como subprograma, para desenhar a imagem do Exercício 13.

### Exercício 19

Escreve um programa para desenhar um degrau de escada



e usa-o como subprograma para resolver o Exercício 10(a).

### Exercício 20

Resolve de novo o Exercício 11(a), utilizando o seguinte subprograma:

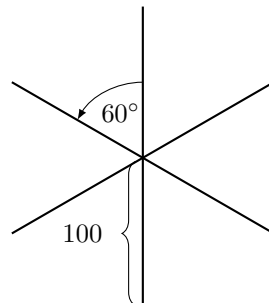
```
to LINHA
fd 150 bk 150
end
```

### Exercício 21

Escreve o seguinte programa **RAIO** e executa-o:

```
to RAI0
fd 100 bk 200 fd 100
end
```

Usa o programa **RAIO** como subprograma para o programa **ESTRELA6**, que vai desenhar a seguinte imagem:



### Exercício 22

Resolve outra vez o Exercício 15 e o Exercício 16, com a ajuda de subprogramas.

### Exercício 23

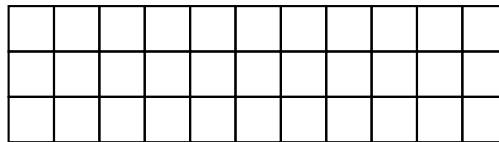
Nós criamos anteriormente um programa **FILA10**. O que faz o seguinte programa?

```
FILA10 fd 20 lt 90 fd 200 rt 90
```

Verifica a tua ideia no computador.

### Exercício 24

Escreve um programa que desenha a seguinte imagem:



### Exercício 25

Desenhar quadrados de tamanhos diferentes.

- Escreve um programa que desenha um quadrado com um lado de comprimento 50 e nomea-o **QUADRADO50**. Experimenta-o, para ver se faz o que deve fazer.
- Escreve um programa que desenha um quadrado com um lado de comprimento de 75.
- Executa o programa:

```
QUADRADO50  
QUADRADO75  
QUADRADO100
```

O que surge?

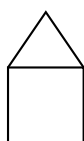
- Como é que mudavas o programa para desenhar mais três quadrados adicionais, com maior tamanho?

## Construir casas

Em seguida queremos ajudar um arquiteto na construção de uma urbanização. Para que a construção seja mais fácil, ele quer construir todas as casas iguais. Nós apresentamos-lhe a seguinte proposta:

```
to CASA
rt 90
repeat 4 [fd 50 rt 90]
lt 60 fd 50 rt 120 fd 50 lt 150
end
```

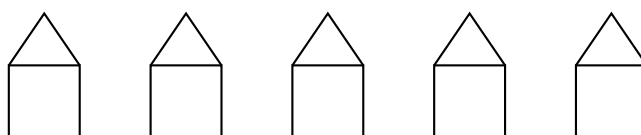
Este programa desenha a seguinte casa:



### Exercício 26

Onde arranca a tartaruga com o desenho da casa? Pensa no caminho que a tartaruga percorre enquanto desenha a casa atrás mencionada, com a ajuda do programa **CASA**. Onde está situada a tartaruga depois de ter desenhado a casa? Desenha a correspondente imagem e descreve como no Exercício 1 que comando teve a qual consequência.

O arquiteto construiu esta casa e ve agora que tudo funcionou. Por esse motivo ele usa este programa como modelo, para construir a primeira rua com casas. No fim a rua deve de ter a seguinte forma:



Já que desenha a casa sempre conforme o mesmo modelo, pode usar a componente **CASA** cinco vezes e não tem que voltar sempre a pensar, como deve construir a casa. No início deixa a tartaruga desenhando a primeira casa da esquerda e depois diz á tartaruga, que salte para o ponto de partida da segunda casa:



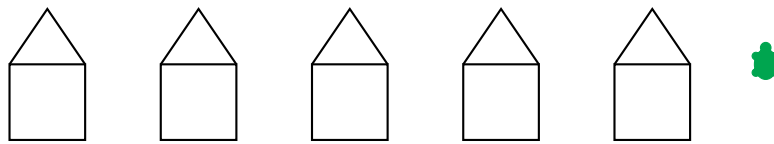
O arquiteto faz isto com o seguinte programa:

```
CASA rt 90 pu fd 50 lt 90 pd
```

Agora a tartaruga pode desenhar neste sítio, exatamente a mesma casa outra vez e saltar de novo para o ponto de partida da próxima casa. Ela faz isto até ter desenhado todas as 5 casas. Portanto temos de repetir a parte do programa em cima cinco vezes e assim recebemos uma nova fila com 5 casas iguais. O programa para isso chamamos **FILACASA**:

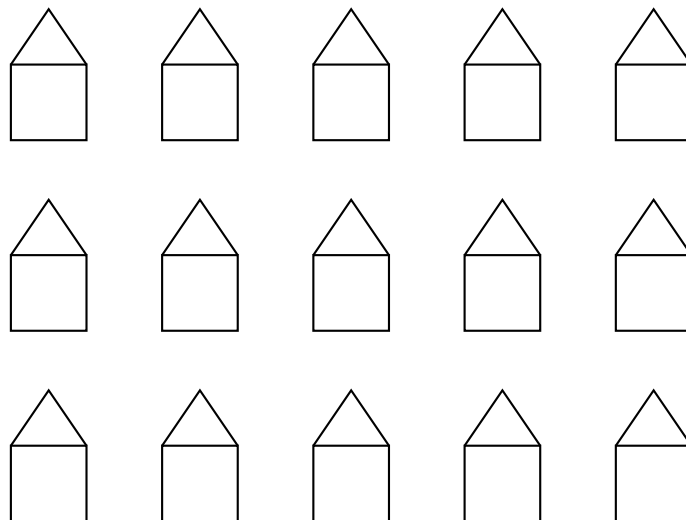
```
to FILACASA  
repeat 5 [CASA rt 90 pu fd 50 lt 90 pd]  
end
```

No fim a tartaruga está situada, onde a próxima casa seria desenhada:



### Exercício 27

Agora queremos aumentar a urbanização por algumas ruas. Usa o programa **FILACASA** como modelo, para desenhar a seguinte imagem:



*Aviso:* Depois de cada fila a tartaruga tem de saltar para o sítio correto, para desenhar a próxima fila.

## Linhas grossas e quadrados pretos

### Exercício 28

Desenhar linhas grossas com o programa **GROSSO**.

Nomea o seguinte programa **GROSSO**

```
fd 100  
rt 90  
fd 1  
rt 90  
fd 100  
rt 180
```

no editor e depois escreve

**GROSSO**

na barra de comandos. O que desenha a tartaruga? Numa folha desenha com um lápis, como surgiu a imagem.

### Exercício 29

Repete 100 vezes o programa **GROSSO** com o programa

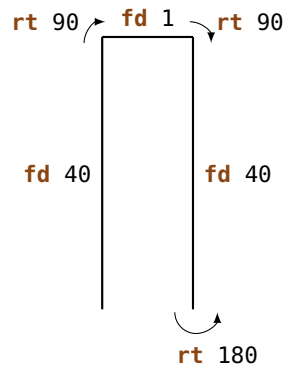
```
repeat 100 [GROSSO]
```

O que surge no ecrã?

### Exercício 30

Neste exercício vamos desenhar linhas grossas. No Exercício 28 nós vimos que uma linha grossa, pode ser desenhada da seguinte forma:

```
to GROSS040
fd 40
rt 90
fd 1
rt 90
fd 40
rt 180
end
```



A linha grossa surge quando se desenharam duas linhas tão compactas, uma ao lado da outra, as quais em conjunto parecem uma só linha grossa.

Escreve o programa **GROSS040** e experimenta-o.

### Exercício 31

Uma linha grossa com um comprimento de 40, pode ser considerado um retângulo com uma largura de 1 e um comprimento de 40. Depois de desenhar **GROSS040** a tartaruga está na segunda linha inferior e olha para cima. Quer dizer, se o programa **GROSS040** é repetido, a tartaruga pinta por cima da segunda linha. Portanto recebemos um retângulo com uma largura de 2 e um comprimento de 40. Com cada repetição é acrescentado apenas uma linha nova. Quando repetimos **GROSS040** 40 vezes, forma-se o quadrado preto com um lado de comprimento 40. Tenta-o, repetindo **GROSS040** 40 vezes.

Escreve um programa com o nome **PRET040**, que desenha um quadrado com um lado de comprimento 40.

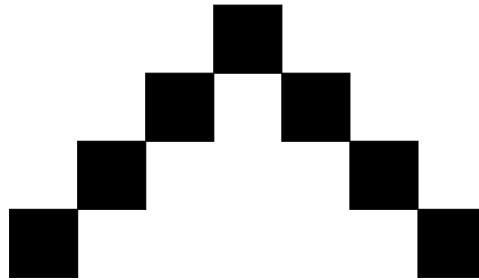
### Exercício 32

Usa o programa **PRET040**, para desenharmos a seguinte imagem:



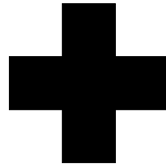
### Exercício 33

Usa o programa `PRET040`, para desenhar a seguinte imagem:



### Exercício 34

Desenha a seguinte imagem:



### Exercício 35

Escreve um programa para desenhar a seguinte imagem:



### Exercício 36

O arquiteto decide encomendar o telhado noutro fornecedor. Portanto ele recebe duas componentes: Uma componente **TELHADO** e uma componente **PARTEINFERIOR**. Escreve para o arquiteto dois programas, que desenhem essas duas componentes e depois junta-as num novo programa **CASA1** para uma casa.

### Exercício 37

As casas no Exercício 27 estão construídas bastante simples. Sê creativo e esboça uma nova casa e constrói com isto uma nova urbanização.



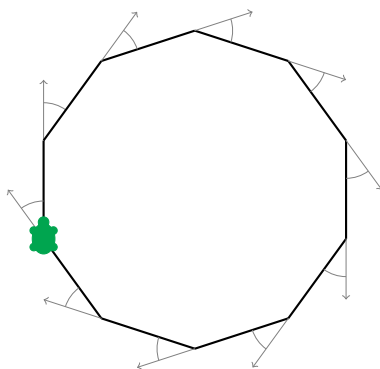
# 4 Polígonos regulares e círculos

## Polígonos regulares

Um polígono regular tem o mesmo número de cantos e lados com o mesmo comprimento. Quando queres desenhar um polígono, por exemplo um decágono (polígono com 10 cantos) com o lápis, tens de desenhar 10 linhas e depois de cada linha mudar (girar) "um pouco" a direção.

Quanto tens de girar?

Quando se desenha um polígono, gira-se várias vezes pelo caminho, mas no fim está-se no mesmo sítio e olha-se exatamente para a mesma direção como no início.



Isto significa que pelo caminho fizemos 360° completos. Portanto quando se desenha um decágono regular, girámos precisamente dez vezes e sempre por um ângulo do mesmo tamanho. Este ângulo é:

$$\frac{360^\circ}{10} = 36^\circ$$

Por isso temos de girar-nos sempre 36°: **rt 36**. Experimentamos, escrevendo o seguinte programa:

```
repeat 10 [ fd 50 rt 36 ]
```

Lado com o comprimento                      Rotação por 36°

### Exercício 38

Desenha os seguintes polígonos regulares:

- (a) um polígono com 5 cantos (pentágono) e um lado de comprimento 180,
- (b) um polígono com 12 cantos (dodecágono) e um lado de comprimento 50,
- (c) um polígono com 4 cantos (quadrilátero) e um lado de comprimento 200,
- (d) um polígono com 6 cantos (hexágono) e um lado de comprimento 100,
- (e) um polígono com 3 cantos (triângulo) e um lado de comprimento 200 e
- (f) um polígono com 8 cantos (octógono) e um lado de comprimento 20.

Quando se quer desenhar um heptágono (7 cantos), tem-se o problema que não é possível dividir 360 por 7 sem resto. Neste caso deixamos o computador calcular o resultado, escrevendo

```
360/7
```

("/"significa para o computador "divide"). O computador encontra o resultado exato. Portanto podemos desenhar um heptágono (7 cantos) com um lado de comprimento 100 da seguinte maneira:

```
repeat 7 [fd 100 rt 360/7]
```

Experimenta-o.

## Desenhar círculos

Com os comandos **fd** e **rt** não é possível desenhar círculos exatos. Como com certeza já reparaste, os polígonos com muitos cantos são muito parecidos aos círculos. Então se escolhermos muitos cantos e lados muito curtos, recebemos dessa maneira círculos.

### Exercício 39

Testa os seguintes programas:

```
repeat 360 [fd 1 rt 1]  
repeat 180 [fd 3 rt 2]  
repeat 360 [fd 2 rt 1]  
repeat 360 [fd 3.5 rt 1]
```

3.5 significa 3 passos e meio.

### Exercício 40

- (a) O que farias para desenhar um círculo muito pequeno? Escreve um programa para isso.
- (b) O que farias para desenhar um círculo grande? Escreve um programa para isso.

### Exercício 41

Tenta desenhar os seguintes semicírculos. O tamanho podes escolher tu próprio:



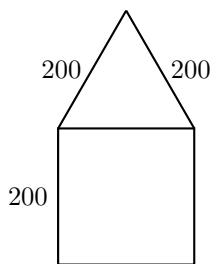
(a)



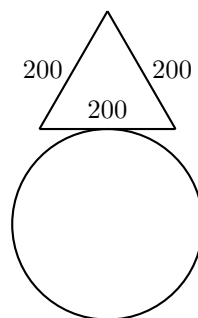
(b)

### Exercício 42

Aproveita as tuas novas experiências, para desenhar as seguintes imagens. O tamanho do círculo podes escolher tu próprio:



(a)



(b)

## Padrão de fantasia

Desenha um heptágono (7 cantos) com:

```
repeat 7 [fd 100 rt 360/7]
```

Depois gira a tartaruga 10° com

```
rt 10
```

Repete os dois algumas vezes e olha para a imagem. Depois de cada heptágono giramos sempre  $10^\circ$  com **rt 10**. Se queremos regressar á posição inicial, temos de repetir esta atividade

$$\frac{360^\circ}{10^\circ} = 36$$

vezes. Então olhamos para ver o que o seguinte programa desenha:

```
repeat 36 [repeat 7 [fd 100 rt 360/7] rt 10]
```

### Exercício 43

Desenha um dodecágono (12 cantos) regular com um lado de comprimento 70 e gira-o 18 vezes até chegares á posição inicial.

*Aviso:* Podes em primeiro lugar escrever um programa para um dodecágono (12 cantos) com um lado de comprimento 70 e por exemplo dar o nome **CANTOS12**. Depois já só tens de complementar o programa


















```
repeat 18 [CANTOS12 rt ... ]
```

### Exercício 44

Inventa um exercício parecido como no Exercício 43 e escreve um programa relacionado com isso.

## Cores

Já que desenhamos padrões de fantasia, também ficariam bem cores. A tartaruga não é capaz de desenhar apenas com preto, mas sim com uma cor qualquer. Cada cor está designada por um número. Uma visão geral de todas as cores encontra-se na seguinte tabela:

0		5		9		13	
1		6		10		14	
2		7		11		15	
3		8		12		16	
4							

Com o comando

<b>setpencolor</b>	<b>X</b>
Comando para mudar a cor	Número da cor desejada

a tartaruga muda da cor atual, para a cor com o número X. Podemos abreviar o comando por **setpc**.

Com isto podemos desenhar padrões engraçados, que surgem com o seguinte programa. Primeiro nomeamos dois programas para desenhar dois círculos com tamanhos diferentes:

```
to CIRCULO3
repeat 360 [fd 3 rt 1]
end

to CIRCULO1
repeat 360 [fd 1 rt 1]
end
```

Agora aproveitamos estes círculos para inventar padrões parecidos aos anteriores:

```
to PADR3
repeat 36 [CIRCULO3 rt 10]
end

to PADR1
repeat 18 [CIRCULO1 rt 20]
end
```

Agora experimentos com cores diferentes:

```
setpc 2
PADR3 rt 2
setpc 3
PADR3 rt 2
```

```
setpc 4
PADR3 rt 2
setpc 5
PADR3 rt 2
```

```
setpc 6
PADR1 rt 2
setpc 15
PADR1 rt 2
```

```
setpc 8
PADR1 rt 2
setpc 9
PADR1 rt 2
```

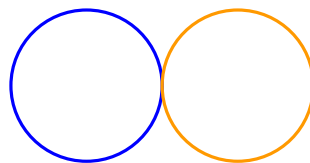
Podes continuar o trabalho à vontade e desenhar ainda mais. Ou desenha um padrão conforme a tua imaginação.

#### Exercício 45

Usa `PADR3`, para desenhar a imagem correspondente em cor de laranja. Depois usa o comando `setpc 7`, para mudar para a cor branca. O que acontece agora se deixas executar `PADR3` de novo?

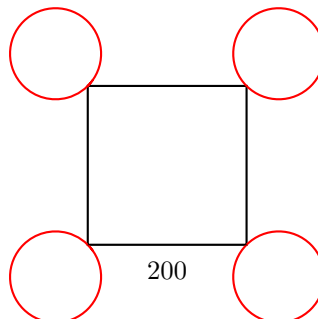
#### Exercício 46

Desenha a seguinte imagem. No início a tartaruga está no ponto comum (ponto de interseção) dos dois círculos.



#### Exercício 47

Escreve um programa para desenhar a seguinte imagem. O tamanho do círculo podes escolher tu próprio.



## 5 Programas com parâmetros

Na Lição 3 aprendemos a dar um nome aos programas e depois de os chamar com o nome, para que o computador possa desenhar a imagem desejada. Na Lição 4 aprendemos a desenhar polígonos. Torna-se muito laborioso de escrever um novo programa para cada polígono, com uma quantidade diferente de cantos.

Examinamos por exemplo os três programas seguintes:

```
repeat 7 [fd 50 rt 360/7]
repeat 12 [fd 50 rt 360/12]
repeat 18 [fd 50 rt 360/18]
```

Os programas são muito parecidos e só se distinguem pelos números amarelos **7**, **12** e **18**. Estes números determinam a quantidade dos cantos. Agora queremos escrever um programa, com o qual podemos desenhar todos os polígonos possíveis:

```
to POLIG :CANTO
repeat :CANTO [fd 50 rt 360/:CANTO]
end
```

O que fizemos? Em toda a parte onde está escrito o número de cantos no programa, escrevemos em lugar do número, um nome, neste caso **:CANTO**. Para que o computador saiba desde o princípio que queremos escolher livremente o número de cantos mais tarde, também tem de estar escrito **:CANTO** atrás do programa e antes um **:**.

Agora quando se escreve o comando **POLIG 12** na barra de comandos, o computador coloca no programa

```
repeat 12 :CANTO [fd 50 rt 360/12 :CANTO]
```

por toda a parte onde está escrito **:CANTO**, o número **12** e desenha assim um dodecágono (12 cantos). Experimenta esta nova forma:

```
POLIG 3
POLIG 4
POLIG 5
POLIG 6
```

Chamamos **:CANTO** um **parâmetro**. No exemplo em cima os números 3, 4, 5 e 6 são **valores do parâmetro :CANTO**. O computador reconhece o parâmetro devido ao **:**. Por isso, sempre onde aparece um parâmetro, tem de estar um **:** á frente do nome do parâmetro.

### Exercício 48

Os seguintes programas desenharam quadrados com lados de comprimento diferente:

```
repeat 4 [fd 100 rt 90]
repeat 4 [fd 50 rt 90]
repeat 4 [fd 200 rt 90]
```

Os números amarelos 100, 50, 200 podem ser considerados valores de um parâmetro que determinam o tamanho do quadrado. Escreve um programa com o parâmetro **:TA**, que é capaz de desenhar um quadrado de qualquer tamanho:

```
to QUADRADO :TA
...
end
```

### Exercício 49

Os seguintes programas desenharam círculos com tamanhos diferentes:

```
repeat 360 [fd 1 rt 1]
repeat 360 [fd 12 rt 1]
repeat 360 [fd 3 rt 1]
```

Escreve um programa com um parâmetro, com o qual se pode desenhar círculos de qualquer tamanho e experimenta-o para um parâmetro com o valor 1, 2, 3, 4 e 5. O nome do programa e o nome do parâmetro, podes escolher tu próprio. Só tens de prestar atenção, para que sempre estejam os dois pontos á frente do parâmetro.

### Exercício 50

Ainda te lembras como se pode desenhar linhas grossas (Exercício 28)? Escreve um programa com um parâmetro que é capaz de desenhar uma linha grossa com um comprimento qualquer.

*Aviso:* Em primeiro lugar podes escrever um programa para uma linha do comprimento 100 e um programa para uma linha do comprimento 50, para se aperceber onde o parâmetro pode ser colocado.



### Exercício 51

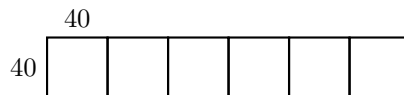
Escreve um programa com um parâmetro que desenha um triângulo equilátero com um tamanho qualquer. Depois desenha com este programa, um atrás do outro, triângulos com o tamanho

20, 40, 60, 80, 100, 120, 140, 160 e 180.

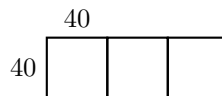
O que surge com isso?

### Exercício 52

Agora queremos desenhar quadriláteros (4 cantos), um ao lado do outro, com um lado de comprimento 40. Escreve um programa **QUAD** com um parâmetro **:NUM**. O parâmetro **:NUM** serve para determinar o número dos quadriláteros. Portanto quando se executa **QUAD 6**, a tartaruga é para desenhar a seguinte imagem:

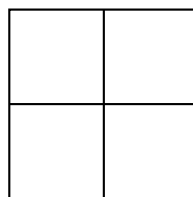


E assim parece quando se executa **QUAD 3**:



### Exercício 53

Escreve um programa que desenha a seguinte imagem, composta por 4 quadrados. O comprimento do lado é para ser determinado por um parâmetro.

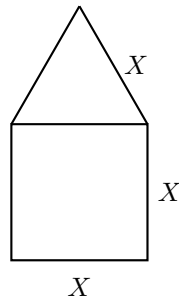


### Exercício 54

Escreve um programa com um parâmetro que desenha hexágonos (6 cantos), com um lado de qualquer comprimento. Experimenta o programa para desenhar hexágonos, com um lado de comprimento 40, 60 e 80.

### Exercício 55

Escreve um programa com um parâmetro `:X`, que desenhe casas de qualquer tamanho como na seguinte imagem.



## Programa com vários parâmetros

Um programa pode ter mais do que um parâmetro. Quando desenhamos polígonos, podemos determinar um parâmetro `:CANTO` para o número de cantos e um parâmetro `:TA` para o comprimento do lado.

Nos seguintes programas o parâmetro `:CANTO` está marcado com amarelo e o parâmetro `:TA` com verde:

```
repeat 13 [fd 100 rt 360/13]
repeat 3 [fd 300 rt 360/3]
repeat 17 [fd 10 rt 360/17]
repeat 60 [fd 3 rt 360/60]
```

Com isto agora podemos escrever um programa para polígonos diferentes:

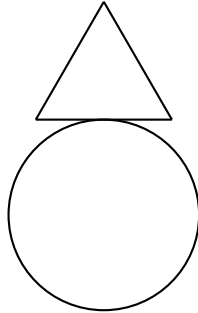
```
to POLIG :CANTO :TA
repeat :CANTO [fd :TA rt 360/:CANTO]
end
```

Testa o programa `POLIG` com os seguintes parâmetro:

```
POLIG 12 60
POLIG 12 45
POLIG 8 30
POLIG 9 30
POLIG 7 31
POLIG 11 50
```

### Exercício 56

Escreve um programa com dois parâmetros, que é capaz de desenhar a seguinte imagem. Neste caso o tamanho do círculo, como também o tamanho do triângulo, podem ser escolhidos livremente pelos parâmetros.



### Exercício 57

O programa

```
fd 100 rt 90 fd 200 rt 90 fd 100 rt 90 fd 200
```

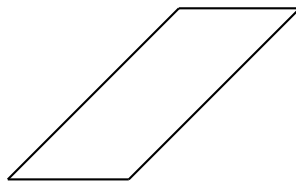
desenha um retângulo com uma largura de 100 e um comprimento de 200. Examina-o e escreve um programa com dois parâmetros, para que se possa desenhar retângulos com uma largura e um comprimento qualquer.

### Exercício 58

O seguinte programa

```
repeat 2 [rt 45 fd 200 rt 45 fd 100 rt 90]
```

desenha um paralelogramo:



Escreve um programa com dois parâmetro, que desenha tais paralelogramos com lados de comprimento á escolha.

### Exercício 59

Desenha uma flor, desenhando um círculo com

```
POLIG 360 2
```

depois girando a tartaruga um pouco com

```
rt 20
```

e a seguir desenhando de novo um círculo com

```
POLIG 360 2
```

e assim sucessivamente continuas com 

```
rt 20 POLIG 360 2 rt 20 POLIG 360 2 ...
```

Quando acabares de desenhar a flor, a tartaruga está situada na posição inicial. Portanto a tartaruga desenhou 18 círculos e no meio girou-se cada vez  $20^\circ$ , assim a tartaruga girou-se no total  $18 \times 20^\circ = 360^\circ$ .

Resumindo, isto resulta no seguinte programa:

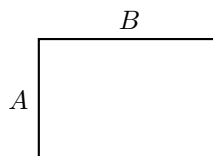
```
repeat 18 [POLIG 360 2 rt 20]
```

Experimenta-o.

- Tu também podes desenhar flores com 10 folhas (círculos) ou com 20 folhas (círculos). Como é que o farias? Escreve um programa para isto e experimenta-o.
- És capaz de escrever um programa com um parâmetro, com o qual se possa desenhar flores com qualquer número de folhas (círculos)?
- És capaz de escrever um programa, com o qual podes escolher os seguintes parâmetros livremente:
  - o número de folhas (círculos) e
  - o tamanho dos círculos?

### Exercício 60

Escreve um programa para desenhar um retângulo qualquer, com uma cor qualquer:



Isto significa que os lados de comprimento  $A$  e  $B$ , bem como a cor, podem ser escolhidas livremente.

## 6 Desenhar flores e entregar parâmetros aos subprogramas

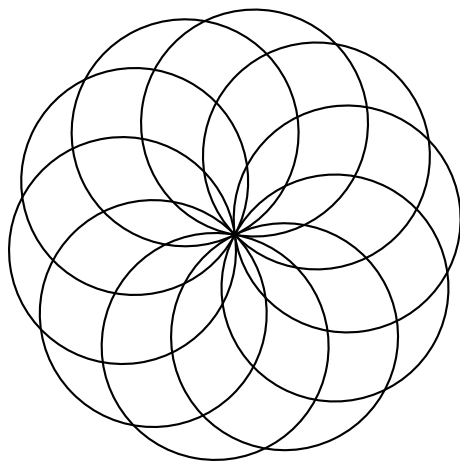
Nesta lição aprendemos a desenhar flores. Vamos escolher a sua forma e cor com a ajuda de parâmetros, para que a nossa tartaruga possa desenhar padrões bonitos, coloridos, fantasiosos.

Examinamos então o programa:

```
to CIRCULO :TA
repeat 360[fd :TA rt 1]
end
```

Este programa já o temos no editor. Agora podemos desenhar uma flor com 10 folhas com o seguinte programa:

```
repeat 10 [CIRCULO 1 rt 36]
```



### Exercício 61

Alguém quer desenhar uma flor com 24 folhas. Como é que temos de mudar o programa em cima?

### Exercício 62

Desenha uma flor com 12 folhas e com folhas que tenham o dobro do tamanho, comparando com a anterior.

Agora queremos escrever um programa para flores no editor, em qual o tamanho das folhas pode ser escolhido. Isto significa, que queremos utilizar o subprograma `CIRCULO :TA` e assim ter escolha livre para `:TA`. Isto só funciona se o programa para a flor também contém o parâmetro para a escolha do tamanho das folhas.

Escreve no editor

```
to FLOR :TA
repeat 10 [CIRCULO :TA rt 36]
end
```

Executa `FLOR 1`, `FLOR 2` e `FLOR 3` e repara a imagem. O que aconteceu? Quando executamos `FLOR 1`, o 1 em `:TA` foi colocada como valor. Assim o subprograma `CIRCULO :TA` é executado como `CIRCULO 1`.

### Exercício 63

Explica o que acontece na execução de `FLOR 2`.

### Exercício 64

Reflete sobre o que o seguinte programa faz e depois verifica.

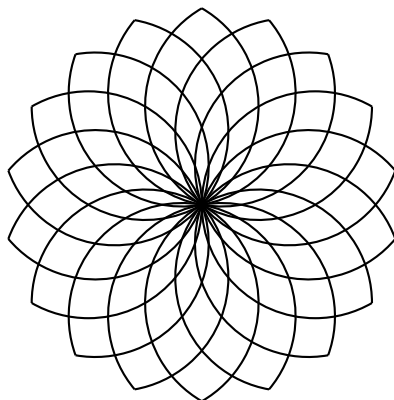
```
to FLORES :TA1 :TA2
setpc 3 FLOR :TA1
setpc 4 FLOR :TA2
end
```

### Exercício 65

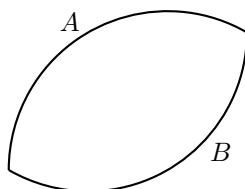
Queremos desenvolver o programa `FLOR` para `FLOR1`, para que não só o tamanho das folhas, mas sim também o número das folhas, possa ser escolhido livremente. Como é que o fazes?

## Uma flor com folhas bicudas

Queres aprender a desenhar uma flor com folhas bicudas? Que tal por exemplo esta flor?



Para desenhar a tal flor, em primeiro lugar temos de pensar como podemos desenhar uma única folha. Podemos considerar uma folha



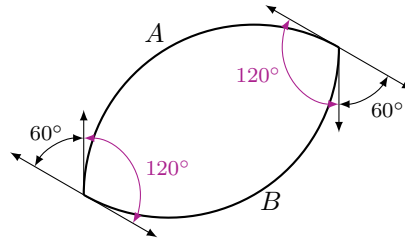
como duas partes de um círculo *A* e *B* coladas uma à outra. Uma parte de um círculo, podes por exemplo desenhar com o seguinte programa:

```
repeat 120 [fd 2 rt 1]
```

Experimenta-o.

Reparamos que este programa é muito parecido ao programa para círculos. Em vez de 360 movimentos pequenos com  $1^\circ$  de rotação, fazemos só 120 movimentos pequenos [fd 2 rt 1] e desta maneira, só desenhamos um terço de um círculo ( $120^\circ$ ).

Agora a pergunta é por quanto é que temos de girar a tartaruga, antes de desenhar a parte do círculo *B*, para o lado inferior da folha. Vamos olhar esse caso na seguinte imagem:



Se no fim queremos chegar à posição inicial, temos de girar a tartaruga na totalidade, como sempre, por  $360^\circ$ . Na parte *A* giramo-la por  $120^\circ$  e na parte *B* também por  $120^\circ$ . Portanto ainda sobram

$$360^\circ - 120^\circ - 120^\circ = 120^\circ$$

que temos de dividir regularmente pelas duas rotações na ponta da folha:

$$\frac{120^\circ}{2} = 60^\circ.$$

Com isto recebemos o seguinte programa:

```
repeat 120 [fd 2 rt 1]
rt 60
repeat 120 [fd 2 rt 1]
rt 60
```

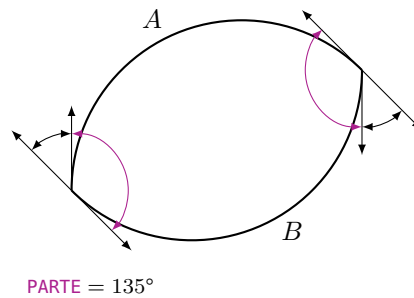
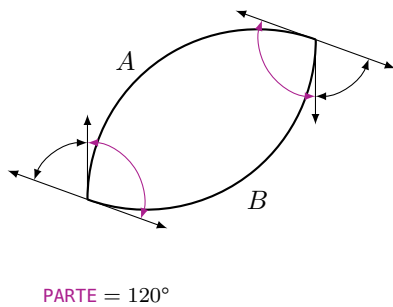
ou ainda mais simples:

```
repeat 2 [repeat 120 [fd 2 rt 1] rt 60]
```

Experimenta-o.



Agora podíamos desenhar folhas mais estreitas (as partes *A* e *B* são mais curtas) ou folhas mais largas (as partes *A* e *B* são mais compridas).



Em relação a isso, podemos utilizar novamente um parâmetro. Chamamos o parâmetro por exemplo **:PARTE**. Depois calculamos a rotação na ponta da folha da seguinte forma:

Antes de desenhar a parte *B* da folha, metade da rotação completa, quer dizer  $\frac{360^\circ}{2} = 180^\circ$ , tem de estar feita. Portanto a rotação na ponta da folha é

$$180^\circ - \text{:PARTE.}$$

Com isto podemos escrever o seguinte programa no editor:

```
to FOLHA :PARTE
repeat 2 [repeat :PARTE [fd 2 rt 1] rt 180-:PARTE]
end
```

Depois experimenta o programa, escrevendo os seguintes programas na barra de comandos:

- FOLHA 20
- FOLHA 40
- FOLHA 60
- FOLHA 80
- FOLHA 100

O que acontece?

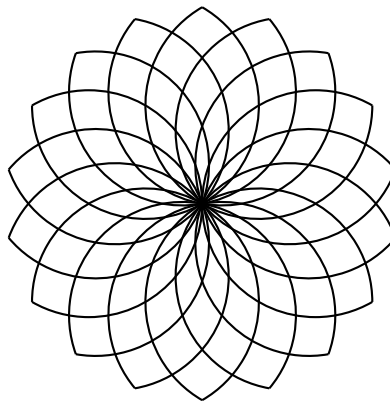
## Uma flor tem muitas folhas bicudas

Agora queremos usar **FOLHA** como subprograma para desenhar flores com folhas bicudas.

### Exercício 66

Primeiro desenha uma flor com o seguinte programa:

```
FOLHA 100  
rt 20  
FOLHA 100  
rt 20  
FOLHA 100  
....
```



Quantas vezes tens de repetir os comandos **FOLHA** e **rt 20**, para desenhar esta flor por completo?

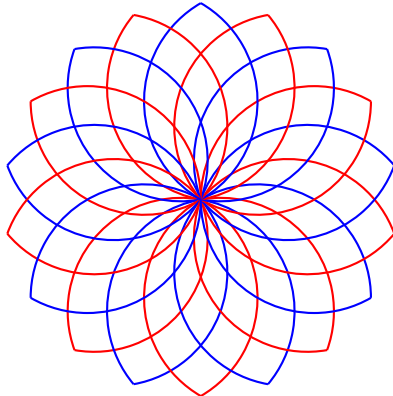
Escreve o programa para a flor numa só linha com um comando- **repeat** apropriado. (Lembra-te que no total todas as rotações **rt** entre as folhas individuais, tem de resultar em 360°.)

### Exercício 67

Introduz o programa do Exercício 66 no editor. Chama o programa **FLOR3**. O programa deve de ter o parâmetro **:PARTE**. O que acontece se tu executas **FLOR3 60**, **FLOR3 80** und **FLOR3 100**?

### Exercício 68

- (a) Escreve um programa com um parâmetro, que desenha a flor do Exercício 66 numa cor qualquer. Chama o teu programa **FLOR4**.
- (b) Agora muda o teu programa para **FLOR5**, assim que o número de folhas que são para ser desenhadas, seja definido por um parâmetro **:NUM** novo. Lembra-te que em conjunto, todas as rotações **rt** entre as folhas, tem de resultar em  $360^\circ$ .
- (c) Muda o teu programa **FLOR5**, assim que a flor seja desenhada em duas cores quaisquer. Chama o novo programa **FLOR6**.



### Exercício 69

No programa **FOLHA**, o comando **fd 2** determina o tamanho do círculo, do qual nós cortamos a parte do círculo do ângulo **:PARTE**. Este valor 2, também podemos substituir por um parâmetro com o nome **:TA** (tamanho). Escreve um programa

```
FOLHAS :PARTE :TA
```

com os parâmetros **:PARTE** e **:TA**, com os quais podemos regular a parte do círculo e o tamanho. Depois experimenta-o com as seguintes chamadas dos programas:

```
FOLHAS 100 1
```

```
FOLHAS 100 1.5
```

```
rt 100
```

```
FOLHAS 80 2
```

```
FOLHAS 80 2.5
```

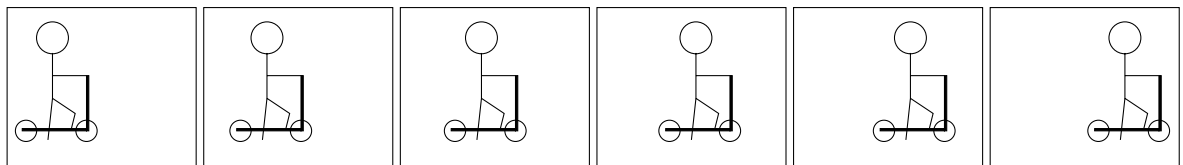
Gira a tartaruga  $80^\circ$  para a direita e repete o programa acima.

### Exercício 70

Inventa outras imagens de fantasia.

## 7 Programação de animações

Sabes como se produzem filmes de desenhos animados? Funciona exatamente igual como com um folioscópio. Em primeiro lugar desenham-se algumas imagens, que raramente se distinguem umas das outras. Na seguinte imagem por exemplo, de figura para figura, o rapaz com a trotinete só se move um pouco:

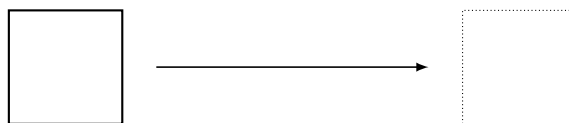


Colocam-se as imagens umas por cima das outras e folheia-se rapidamente com o polegar, ficando com a impressão que o rapaz anda com a sua trotinete da esquerda para a direita. Imagens movimentadas chamam-se **animações**.

Nesta lição aprendemos como se pode programar uma animação, com a ajuda da tararuga.

### Um quadrado que deixa rastros

Na nossa primeira animação escolhemos uma figura que não seja muito difícil e que já conhecemos há muito tempo: vamos deixar caminhar um quadrado da esquerda para a direita.



O programa para o quadrado já o conhecemos:

```
to QUAD100
repeat 4 [fd 100 rt 90]
end
```

Depois do quadrado ser desenhado uma vez, movemos a tartaruga um pouca para a direita e de novo desenhamos o quadrado. Removemos a tartaruga outra vez para a direita e novamente desenhamos um quadrado. Isto é repetido várias vezes.

No seguinte programa desenhamos 120 desses quadrados:

```
to QUADMOVER  
repeat 120 [QUAD100 rt 90 fd 4 lt 90]  
end
```

### Exercício 71

Escreve os programas `QUAD100` e `QUADMOVER` no editor e executa `QUADMOVER`. O que é desenhado?

Tu vês que o rasto de *todos* os quadrados são desenhados. Mas numa animação só queremos ver o último quadrado e limpar o rasto.

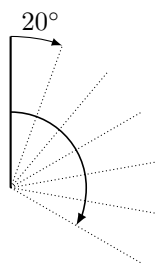


### Exercício 72

Deixa o quadrado caminhar de baixo para cima, em vez da esquerda para a direita.

### Exercício 73

Escreve um programa para uma linha com um comprimento de 20. Usa esse programa, para girar uma linha no ponto mais inferior no sentido horário:



## Desenhar um quadrado e apagar outra vez

Para limpar um rasto, temos de aprender a apagar uma figura que acabamos de desenhar. Para isso a tartaruga tem de utilizar uma borracha em vez de um lápis. Com o comando novo `penerase` ou, mais curto, `pe` a tartaruga muda de um lápis para uma borracha.

### Exercício 74

Reflete o que o programa `QUAD100 pe QUAD100` faz, sem o executar no computador.

Se a tartaruga é para desenhar novamente, temos de lhe comunicar claramente. Para isto também há um comando novo: `penpaint` ou, mais fácil, `ppt`. Utilizamos o novo comando já agora no programa do Exercício 74.

O programa vai parecer da seguinte maneira:

```
QUAD100 pe QUAD100 ppt
```

### Exercício 75

Executa o programa acima. O que acontece? És capaz de o explicar?

## O quadrado tem de esperar um pouco

Certamente reparaste quando resolveste o Exercício 75, depois de o quadrado ser desenhado, é apagado rapidamente. Nós nem reparamos que foi desenhado um quadrado. Antes de apagar um quadrado, temos de deixar o computador esperar um pouco.

Isto podemos fazer da seguinte maneira:

<code>wait</code>	4
Comando para esperar	Tempo de espera

### Exercício 76

Experimenta o seguinte programa:

```
QUAD100 wait 4 pe QUAD100 ppt
```

## Um quadrado que se move da esquerda para a direita

Agora estamos prontos a introduzir o apagar e o esperar do quadrado, no nosso programa `QUADMOVE`:

```
to QUADMOVE
repeat 120 [QUAD100 wait 4 pe QUAD100 rt 90 fd 4 lt 90 ppt]
end
```

Experimenta-o. Se a tartaruga te incomoda enquanto está a desenhar, então começa o programa com o comando **hideturtle** (ou mais curto: **ht**), que deixa desaparecer a tartaruga. Até vais reparar, que a animação vai ficar mais rápida. Termina o programa com o comando **showturtle** (ou mais curto: **st**) diretamente antes do **end**. Desta maneira a tartaruga torna-se visível de novo.

### Exercício 77

Mova um quadrado com o tamanho  $50 \times 50$  para cima.

### Exercício 78

Muda o programa **QUADMOVE**, para que o quadrado se mova a uma velocidade dupla para a direita.

### Exercício 79

Também es capaz de mudar o programa **QUADMOVE** de certa maneira, para que o quadrado se mova com metade da velocidade para a direita?

### Exercício 80

Muda o programa **QUADMOVE**, para que o quadrado se mova da direita para a esquerda, em vez da esquerda para a direita.

### Exercício 81

Primeiro pensa o que o seguinte programa vai fazer, e depois verifica a tua suposição, executando o programa:

```
to QUADMOVE1
ht
repeat 50 [QUAD100 wait 5 pe QUAD100 fd 3 rt 90 fd 3 lt 90 ppt]
QUAD100
st
end
```

### Exercício 82

Primeiro pensa no que o seguinte programa vai fazer, e depois executando o programa, verifica a tua suposição:

```
to GIRAR
ht
repeat 360 [QUAD100 wait 4 pe QUAD100 fd 5 rt 1 ppt]
QUAD100
st
end
```

### Exercício 83

Modifica o programa **GIRAR**, para que o quadrado se mova quatro vezes mais rapido.

### Exercício 84

O que faz o seguinte programa?

```
repeat 6 [GIRAR]
```

### Exercício 85

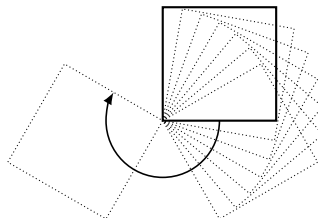
Pega no seguinte programa

```
to TERRA
repeat 45 [fd 16 rt 8]
end
```

e utiliza-o, para desenhar uma animação, na qual a Terra se move em círculo à volta do Sol. Podes desenhar o sol da maneira que tu queiras.

### Exercício 86

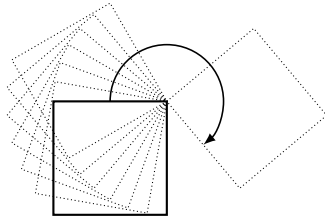
Gira um quadrado no sentido horário pelo seu canto esquerdo inferior. Tu podes escolher o comprimento do lado:





### Exercício 87

Agora gira o quadrado no sentido horário, pelo seu canto direito superior:



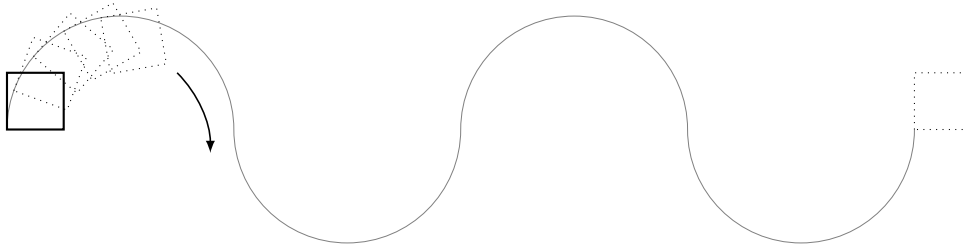
Se já conheces parâmetros, podes resolver os seguintes exercícios.

### Exercício 88

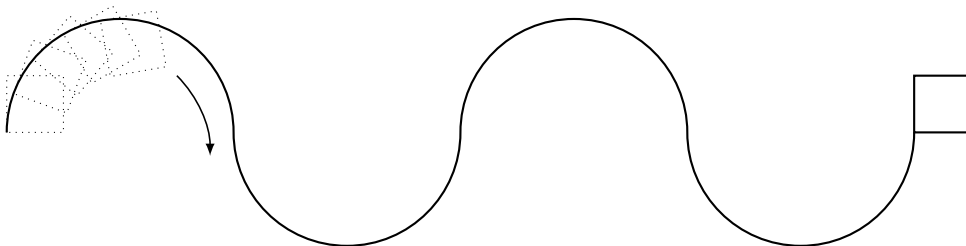
Escreve um programa com *dois parâmetros*, para deixar caminhar um quadrado da esquerda para a direita. Um parâmetro é para determinar o comprimento do lado, o outro parâmetro é para determinar a velocidade do movimento do quadrado.

### Exercício 89

- (a) Deixa andar um quadrado em cima do caminho desenhado em baixo, que é composto por 4 semicírculos. O comprimento do lado, é para ser determinado por um parâmetro.

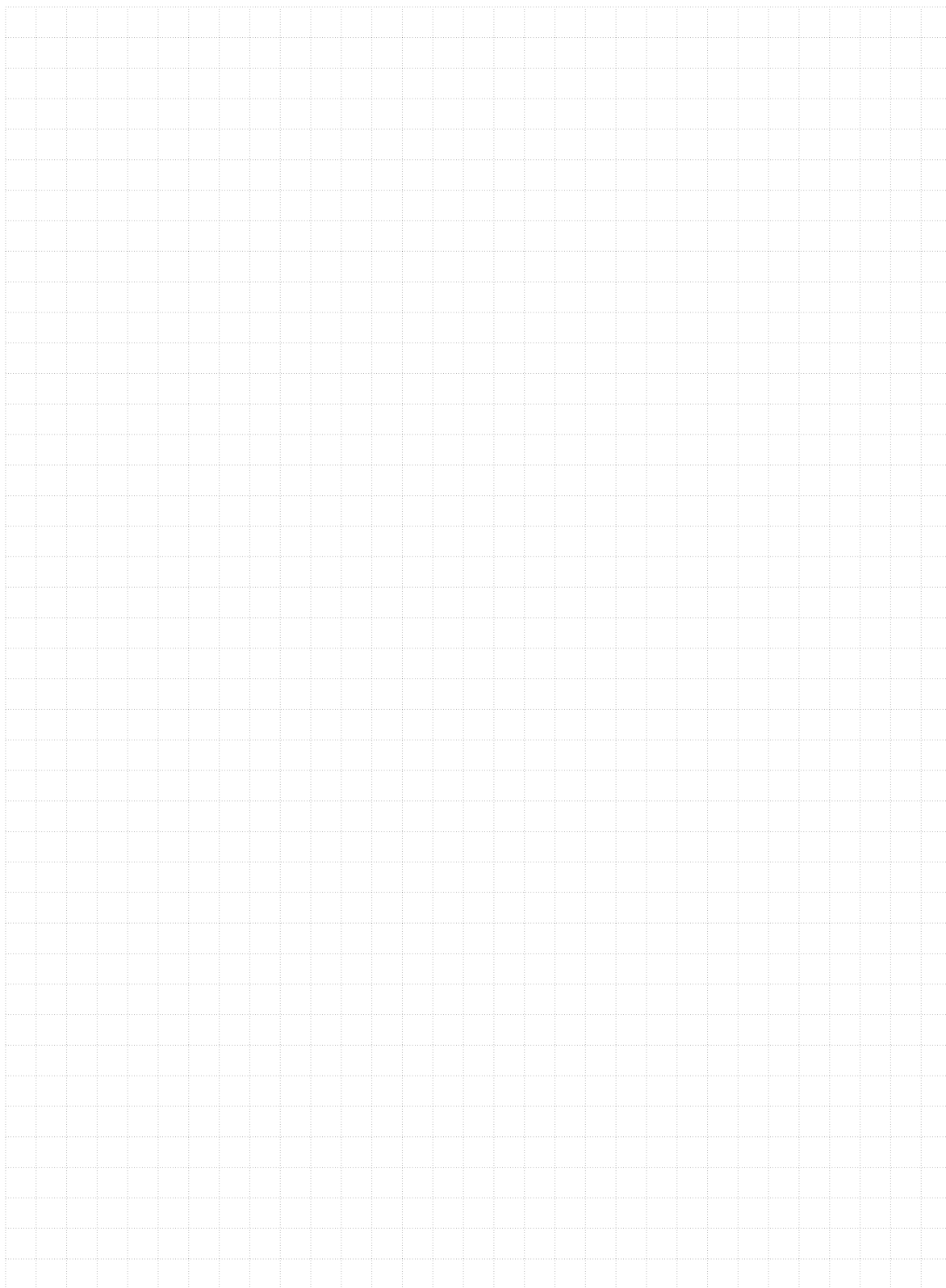


- (b) Agora o caminho é para ser desenhado como rasto.

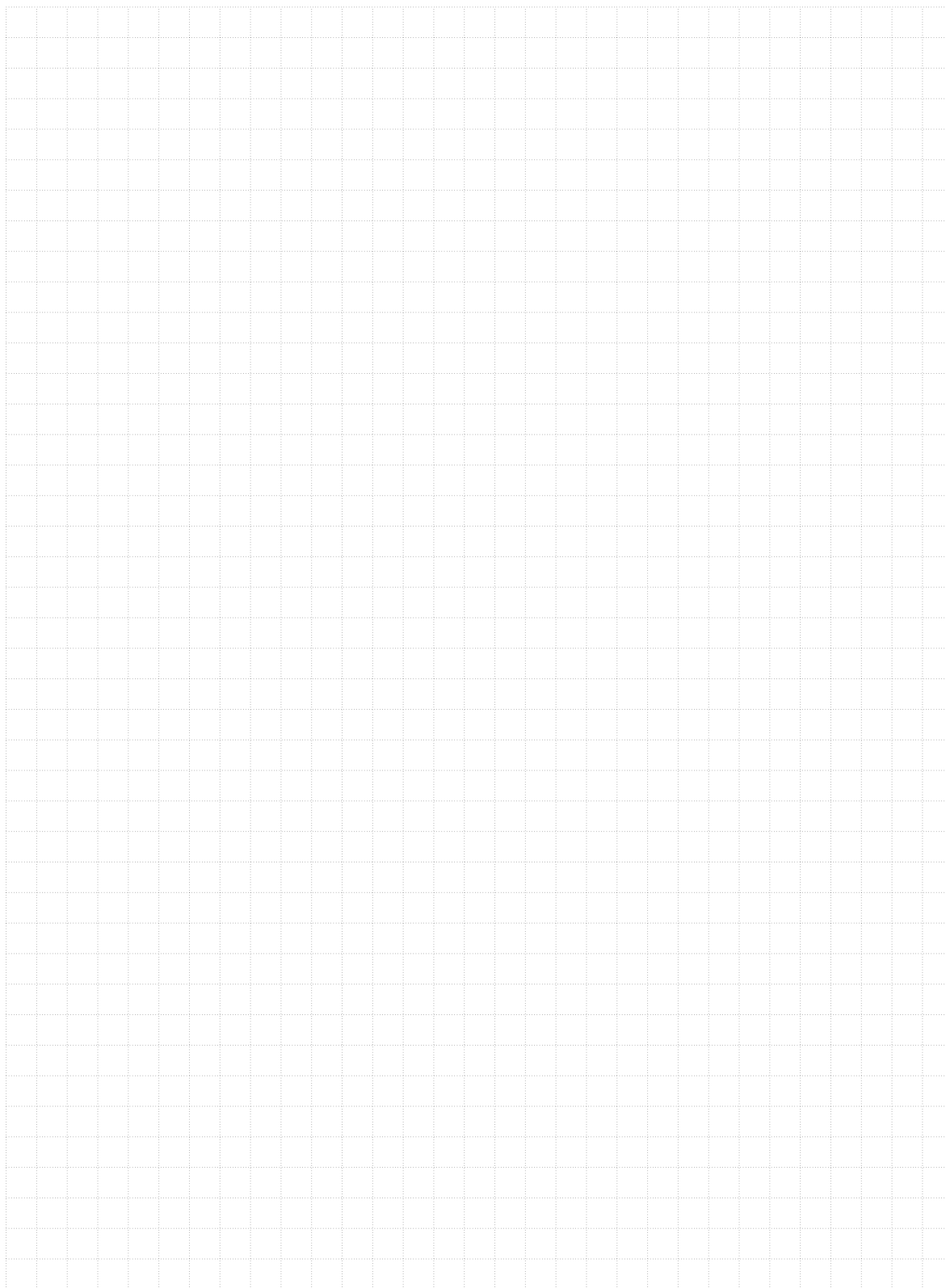


- (c) És capaz de aumentar o programa do (b) para que o número dos semicírculos, seja determinado por um parâmetro?

## Os meus apontamentos



## Os meus apontamentos



# Vista geral de comandos

<b>fd</b>	<b>100</b>	100 passos para a frente
<b>bk</b>	<b>50</b>	50 passos para trás
<b>cs</b>		apagar tudo e começar de novo
<b>rt</b>	<b>90</b>	girar 90 graus para a direita
<b>lt</b>	<b>90</b>	girar 90 graus para a esquerda
<b>repeat</b>	<b>4</b>	[...] o programa em [...] vai repetir-se quatro vezes
<b>pu</b>		a tartaruga muda para o modo caminhar
<b>pd</b>		a tartaruga volta para o modo lápis
<b>setpc</b>	<b>3</b>	muda a cor do lápis para a cor 3
<b>to</b>	<b>NOME</b>	cria um programa com um nome
<b>to</b>	<b>NOME :PARÂMETRO</b>	cria um programa com um nome e um parâmetro
<b>end</b>		todos os programas terminam com este comando
<b>pe</b>		a tartaruga muda para o modo borracha
<b>ppt</b>		a tartaruga volta para o modo lápis
<b>wait</b>	<b>5</b>	a tartaruga espera 5 unidades de tempo



# Programmar mit LOGO

Informationstechnologie und Ausbildung  
ETH Zürich, CAB F 15.1  
Universitätstrasse 6  
CH-8092 Zürich

[www.ite.ethz.ch](http://www.ite.ethz.ch)  
[www.abz.inf.ethz.ch](http://www.abz.inf.ethz.ch)