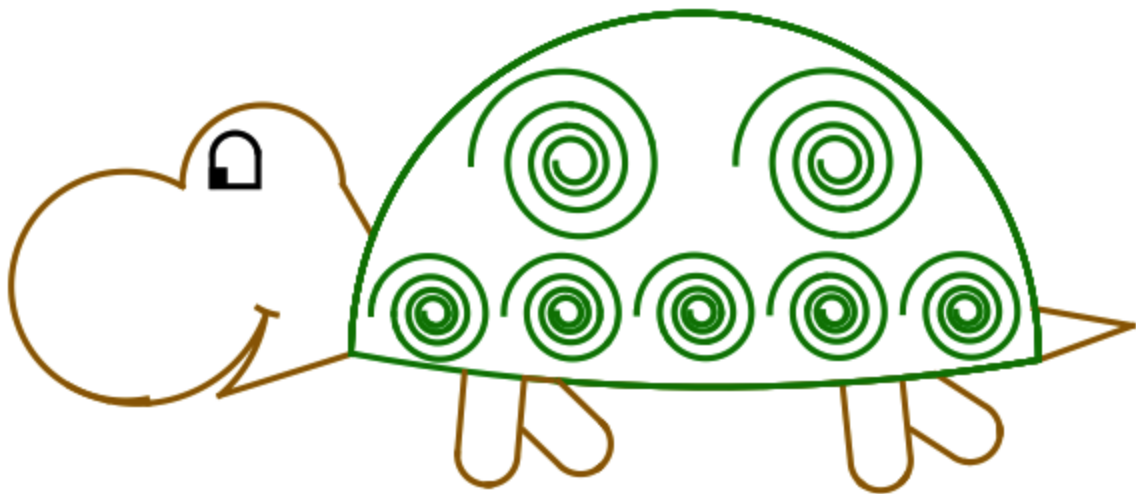


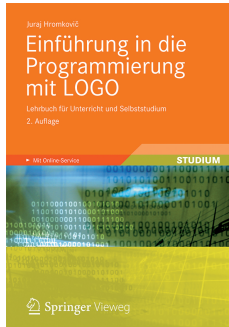
Heidi Gebauer Juraj Hromkovič Lucia Keller
Ivana Kosírová Giovanni Serafini Björn Steffen

Programar con LOGO



Programando con LOGO

Este documento es una versión abreviada de las lecciones 1–7 del libro de texto *Einführung in die Programmierung mit LOGO* (“Introducción a la programación con LOGO”). El libro de texto completo contiene ejercicios y explicaciones adicionales, además de indicaciones para el docente; y comprende en total 15 lecciones.



Juraj Hromkovič. *Einführung in die Programmierung mit LOGO: Lehrbuch für Unterricht und Selbststudium* (“Introducción a la programación con LOGO: Texto para la enseñanza y el auto-aprendizaje”), 3ra. Edición, Springer Vieweg 2014. ISBN: 978-3-658-04832-7.

Version 3.1, 6 de octubre de 2015, SVN-Rev: 17043

Traducción al español por: César Fuentes Montesinos, Angélica Herrera Loyo

Revisión de la versión en español: Angélica Herrera Loyo

Entorno de programación

Los ejemplos y ejercicios contenidos en el presente documento fueron desarrollados para el entorno de programación XLogo, el cual puede ser descargado gratuitamente desde el sitio web xlogo.tuxfamily.org.

Para que los programas aquí incluidos puedan ejecutarse correctamente, el idioma seleccionado dentro de XLogo debe ser el inglés.

Derechos de uso

El ABZ ofrece el presente material para su uso gratuito por parte de docentes e instituciones educativas, siempre y cuando el uso sea interno y con fines didácticos.

ABZ

El Centro de Capacitación y Asesoramiento para la Enseñanza de Informática o ABZ¹ de la Escuela Politécnica Federal de Zúrich (ETH Zürich) brinda apoyo a escuelas y docentes que desean iniciar o expandir actividades didácticas en el área de Informática. Los servicios que se ofrecen comprenden desde la enseñanza y asesoramiento individual en la misma escuela por parte de profesores del ETH y el equipo de ABZ, hasta cursos de capacitación y actualización para docentes, además de materiales de enseñanza.

www.abz.inf.ethz.ch

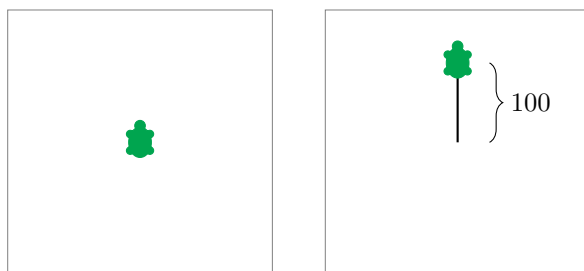
¹Ausbildungs- und Beratungszentrum für Informatikunterricht

1 Comandos fundamentales

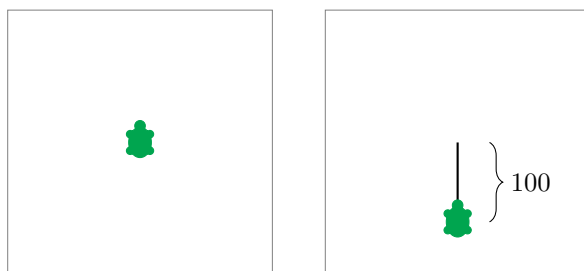
Un **comando** (también conocido como *primitiva*) es una instrucción que la computadora puede entender y ejecutar. En principio, la computadora sólo puede entender comandos muy sencillos, que luego pueden combinarse para formar instrucciones más complicadas. A la secuencia de instrucciones que resulta, la llamamos **programa**. No siempre es fácil escribir programas de computadora. Existen programas que consisten en millones de comandos. Para poder mantener todo bajo control, es muy importante usar un procedimiento metódico y bien organizado, y es justamente eso lo que aprenderemos en este curso de programación.

Dibujar líneas rectas

Con el comando **forward 100** o **fd 100** le ordenamos a la tortuga que se mueva 100 pasos hacia delante.



Con el comando **back 100** o **bk 100** la tortuga se mueve 100 pasos hacia atrás.



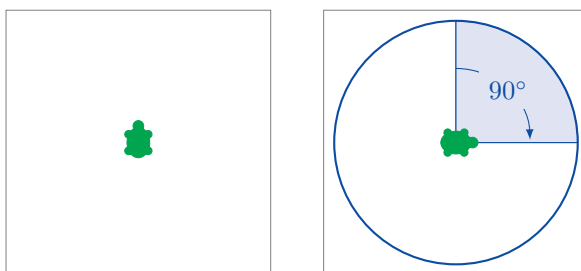
Limpiar y comenzar de nuevo

El comando **cs** borra de la pantalla todo lo que hemos dibujado y regresa a la tortuga a su posición inicial.

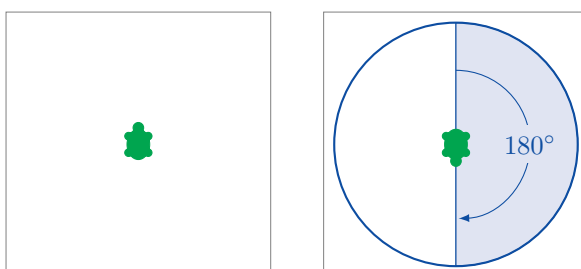
Giros

La tortuga siempre se mueve en la dirección hacia la que está mirando.

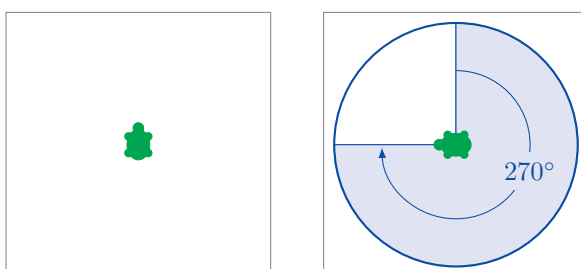
Con el comando **right 90** o **rt 90** le decimos a la tortuga que gire 90° hacia la derecha. Este giro corresponde a un cuarto de vuelta:



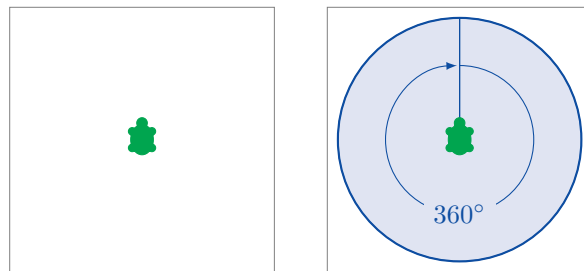
El comando **right 180** o **rt 180** hace girar a la tortuga 180° hacia la derecha, lo que corresponde a una media vuelta:



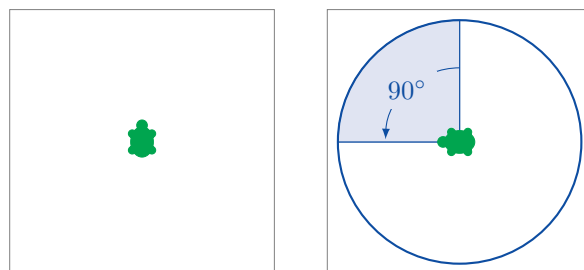
right 270 o **rt 270** hace girar a la tortuga 270° hacia la derecha:



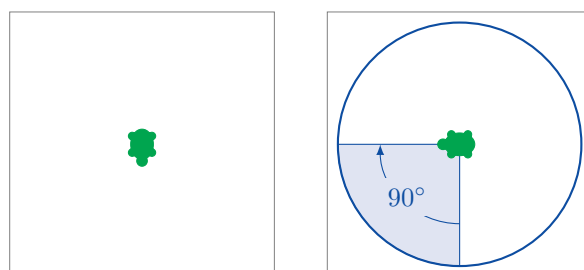
Los comandos **right 360** y **rt 360** hacen que la tortuga gire 360° hacia la derecha, lo que corresponde a una vuelta completa:



Con el comando **left 90** o **lt 90** hacemos girar a la tortuga 90° hacia la izquierda:



Observa que la dirección del giro hacia la izquierda o hacia la derecha depende del punto de vista de la tortuga, como se muestra en el siguiente ejemplo usando el comando **rt 90**:



Programando

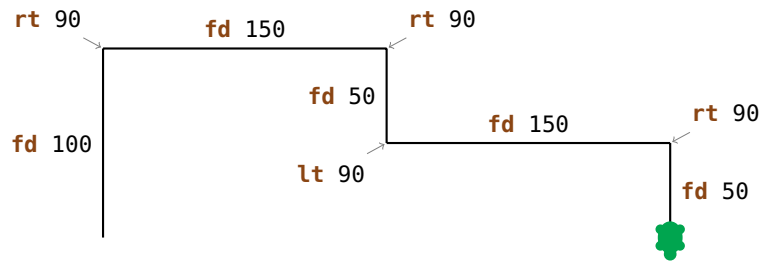
Programar significa escribir una serie de instrucciones para la computadora.

Ejercicio 1

Copia el siguiente programa y ejecútalo:

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

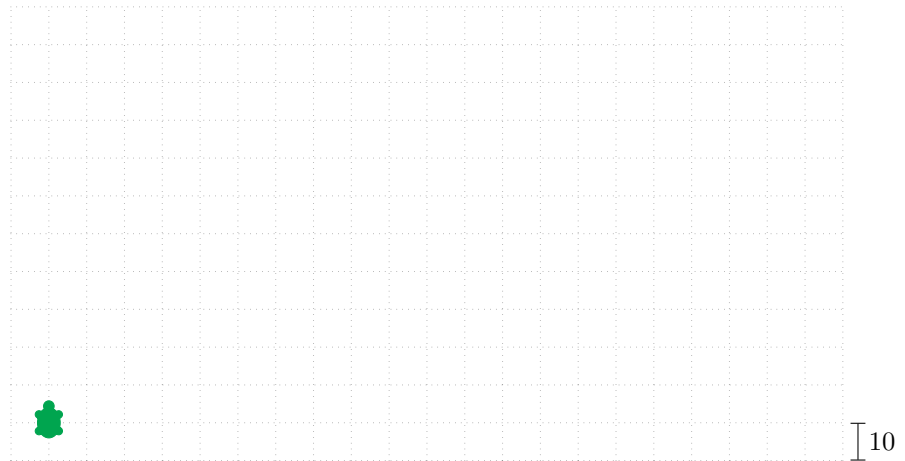
¿Has dibujado la siguiente figura?



Ejercicio 2

Escribe el siguiente programa y ejecútalo:

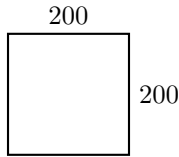
```
fd 100  
rt 90  
fd 200  
rt 90  
fd 80  
rt 90  
fd 100  
rt 90  
fd 50
```



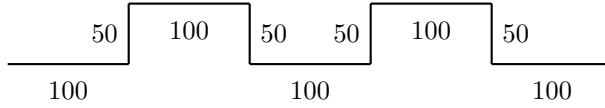
Dibuja la figura que aparece en la pantalla en el espacio cuadriculado de arriba, y escribe qué comando ha dibujado cada una de las partes (como en el Ejercicio 1).

Ejercicio 3

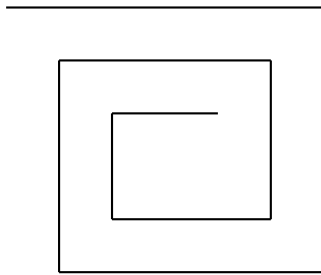
Escribe programas que dibujen las figuras mostradas a continuación. Para cada figura, puedes elegir por tu cuenta la posición inicial de la tortuga.



(a)

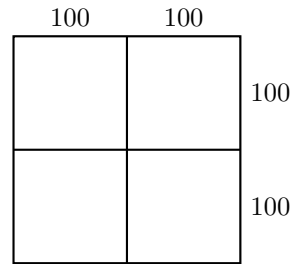


(b)



Puedes elegir el tamaño por tu cuenta.

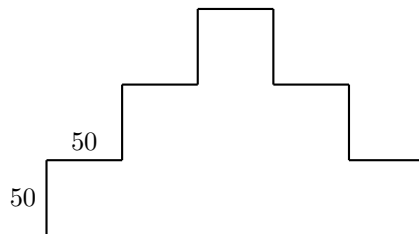
(c)



(d)

Ejercicio 4

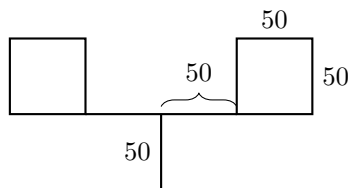
Escribe un programa que dibuje la siguiente figura:



¿Podrías reescribir el programa de tal forma que sólo utilice los comandos **fd 50** y **rt 90**?

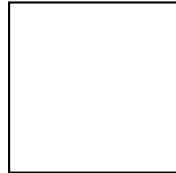
Ejercicio 5

Ana quiere dibujar la siguiente figura. ¿Puedes ayudarle?



2 El comando **repeat**

Si queremos dibujar un cuadrado con lado de longitud 100,



podemos hacerlo con el siguiente programa:

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

Nos podemos dar cuenta, que los comandos

```
fd 100  
rt 90
```

se repiten cuatro veces. ¿No sería más fácil si pudiéramos decirle a la computadora que repita esos dos comandos cuatro veces, en vez de tener que copiarlas de nuevo?

Pero podemos hacer exactamente eso de la siguiente manera:

repeat	4	[fd 100 rt 90]
Primitiva para repetir un programa	Número de repeticiones	Secuencia de comandos que queremos repetir

Ejercicio 6

Escribe el siguiente programa y ejecútalo:

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

¿Qué dibuja el programa? ¿Puedes usar el comando **repeat** para hacer el programa más corto?

Ejercicio 7

Copia el siguiente programa y observa lo que dibuja:

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

Haz el programa más corto usando el comando **repeat**.

Ejercicio 8

Usa el comando **repeat** para escribir un programa que dibuje un cuadrado con lado de longitud 200.

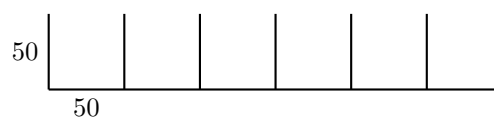
Ejercicio 9

Escribe el siguiente programa y ejecútalo:

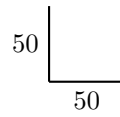
```
fd 100 rt 120
fd 100 rt 120
fd 100 rt 120
```

¿Qué dibuja el programa? ¿Puedes usar el comando **repeat** para hacer el programa más corto?

Ahora queremos dibujar la siguiente figura con la ayuda del comando **repeat**:



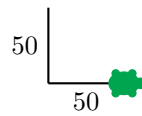
Antes de que comencemos a dibujar, debemos pensar cuál será el patrón que vamos a repetir. Por ejemplo, podemos elegir como patrón de repetición la siguiente figura:



Si queremos comenzar desde la esquina inferior izquierda, podemos dibujar el patrón con el siguiente programa:

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

Después de que ejecutamos este programa, la tortuga ha quedado en la posición que se muestra en la ilustración de abajo, y está mirando hacia la derecha:

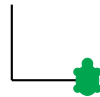


Eso está bien, porque la tortuga ya está en la posición que necesitamos para poder dibujar el patrón de nuevo. Solamente necesita mirar hacia arriba, lo que podemos lograr usando el comando **lt 90**.

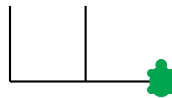
Ahora ejecutamos el programa para comprobar que funcione como queremos.

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

Conseguimos el resultado esperado:



Si ejecutamos el mismo programa nuevamente, obtenemos:



De esta forma vemos que nuestra idea funciona, y podemos repetir nuestro programa 6 veces:

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

 Patrón Orientación

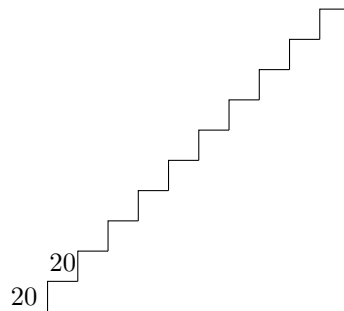
Muchas de las tareas en esta sección, y en particular los siguientes ejercicios, pueden ser solucionados utilizando este mismo procedimiento. Sólo recuerda que siempre debes primero encontrar el patrón que se va a repetir. Luego debes escribir un programa para dibujar el *patrón* y otro programa para *orientar* la tortuga, de tal forma que quede mirando en la dirección correcta para la siguiente repetición del patrón. Al final, la estructura de tu programa será como sigue:

repeat *Número-de-repeticiones* [*Patrón Orientación*]

Ejercicio 10

Dibujando escaleras.

(a) Dibuja una escalera con 10 pasos de tamaño 20 cada uno:



- Primero encuentra el patrón que debe repetirse y escribe un programa para dibujarlo.
- Piensa cómo escribir un programa para orientar la tortuga, de tal forma que quede mirando en la dirección correcta para la siguiente repetición del patrón.
- Finalmente combina ambas partes para poder resolver el ejercicio.

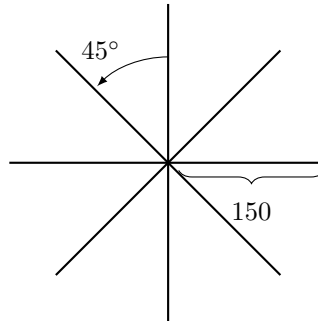
(b) Dibuja una escalera con 5 escalones de tamaño 50.

(c) Dibuja una escalera con 20 escalones de tamaño 10.

Ejercicio 11

Ahora vamos a dibujar estrellas.

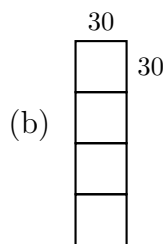
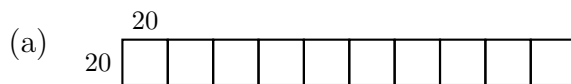
(a) Dibuja la estrella que se muestra a continuación:



(b) La estrella tiene ocho puntas de longitud 150. ¿Puedes dibujar también una estrella con 16 puntas de longitud 100?

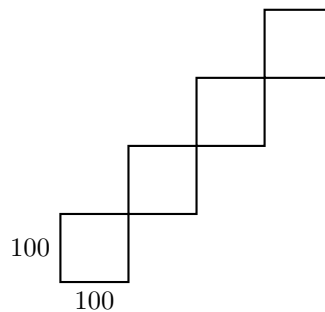
Ejercicio 12

Dibuja las siguientes figuras:



Ejercicio 13

Escribe un programa para dibujar la siguiente figura:



Ejercicio 14

Escribe el siguiente programa y ejecútalo:

```
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
```

¿Qué se dibuja en la pantalla? ¿Podrías hacer este programa más corto?

Modalidad de movimiento

Normalmente la tortuga está en **modalidad de dibujo**, lo que significa que lleva un lápiz consigo y dibuja una línea en la pantalla cada vez que se mueve.

En **modalidad de movimiento**, la tortuga se mueve por la pantalla sin dibujar. Puedes activar la modalidad de movimiento con el comando

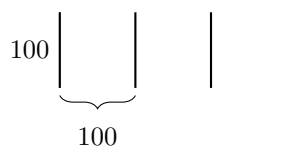
penup o abreviada como **pu**.

Para regresar a la modalidad de dibujo puedes usar el comando

pendown o abreviada como **pd**.

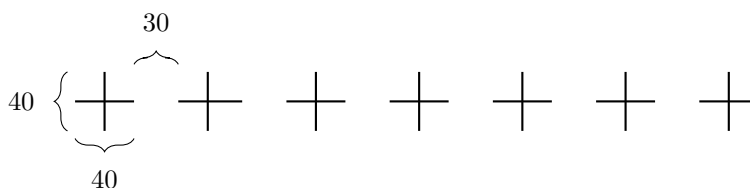
Ejercicio 15

Escribe un programa para dibujar la siguiente figura:



Ejercicio 16

Escribe un programa para dibujar la siguiente figura:



3 Nombrar y ejecutar programas

Podemos darle un nombre a cada programa que hemos escrito hasta el momento. Si después escribimos el nombre del programa en la línea de comandos, las instrucciones contenidas en éste son ejecutadas.

El programa para dibujar un cuadrado con lado de longitud 100 es:

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

Podemos nombrar o “etiquetar” este programa con el nombre **CUADRADO100** así:

```
to CUADRADO100
repeat 4 [fd 100 rt 90]
end
```

Hemos escrito el mismo programa dos veces, una vez sin darle nombre y otro con nombre.

Para escribir un programa con un nombre, tenemos que usar el **Editor**. En este manual, indicaremos cuáles son tales programas poniéndolos dentro de un cuadro gris. Tan pronto hayamos terminado de escribir el programa, podemos hacer clic en el botón con la figura de la tortuga para cerrar el editor.

Uno es libre de elegir cualquier nombre para el programa, en este caso hemos elegido **CUADRADO100** porque queremos indicar que el programa dibuja un cuadrado con lado de longitud 100. Las únicas restricciones para los nombres son que éstos deben comenzar con una letra y que deben tener una sola palabra (es decir, sin espacios en medio).

Después de haber escrito el programa, no se dibujará nada en el pantalla. Esto es porque sólo le hemos dado un nombre, pero no lo hemos ejecutado todavía. Si ahora escribimos

CUADRADO100

en la línea de comandos, entonces el programa **repeat 4 [fd 100 rt 90]** será ejecutado. Ahora se mostrará esto en la pantalla:



Revisemos nuevamente el Ejercicio 12(a). Podríamos hacer la solución más sencilla, si primero escribimos un programa para el patrón que se repite (el cuadrado con lados de longitud 20) y le damos un nombre apropiado:

```
to CUADRADO20
repeat 4 [fd 20 rt 90]
end
```

Después de dibujar el **CUADRADO20**, la tortuga se queda en la esquina inferior izquierda del cuadrado.



Para poder dibujar el siguiente cuadrado, tenemos que mover la tortuga a la esquina inferior derecha. Podemos lograr esto con el programa

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

También le damos un nombre a este programa:

```
to ALINEAR20
rt 90 fd 20 lt 90
end
```

Usando estos dos programas podemos escribir un nuevo programa para el Ejercicio 12(a) de la siguiente manera:

```
repeat 10 [CUADRADO20 ALINEAR20]
```

Naturalmente, también podemos darle un nombre al programa que escribimos anteriormente. Por ejemplo:

```
to FILA10
repeat 10 [CUADRADO20 ALINEAR20]
end
```

Si hacemos esto, llamamos a los programas **CUADRADO20** y **ALINEAR20** **subprogramas** o **subrutinas** del programa **FILA10**.

Ejercicio 17

Escribe un programa para solucionar el Ejercicio 12(b), pero que use otro programa para dibujar un cuadrado de lado 30. Tu programa final debe parecerse a éste:

```
repeat 4 [CUADRADO30 ALINEAR30]
```

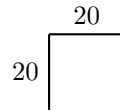
También debes escribir los programas **CUADRADO30** y **ALINEAR30**.

Ejercicio 18

Usa el programa **CUADRADO100** como una subrutina para dibujar la figura del Ejercicio 13.

Ejercicio 19

Escribe un programa para dibujar un escalón



y úsalo como subrutina para la solución del Ejercicio 10(a).

Ejercicio 20

Resuelve nuevamente el Ejercicio 11(a) usando el siguiente subprograma:

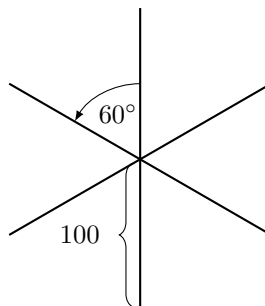
```
to LINEA  
fd 150 bk 150  
end
```

Ejercicio 21

Escribe el siguiente programa en el editor:

```
to RAYO  
fd 150 bk 300 fd 150  
end
```

Usa **RAYO** como un subprograma del programa **ESTRELLA6** para dibujar esta figura:



Ejercicio 22

Solucionar el Ejercicio 15 y el Ejercicio 16 de nuevo, pero ahora usando subrutinas.

Ejercicio 23

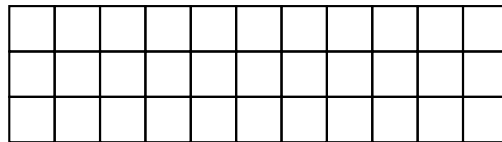
Ya hemos escrito el programa `FILA10`. ¿Qué hace el siguiente programa?

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

Comprueba tu idea usando la computadora.

Ejercicio 24

Escribe un programa que dibuje la siguiente figura:



Ejercicio 25

Dibujar cuadrados de diferentes tamaños.

- Escribe un programa que dibuje un cuadrado de lado 50 y llámalo `CUADRADO50`. Ejecútalo para comprobar que funciona como es debido.
- Escribe un programa que dibuje un cuadrado de lado 75.
- Ejecuta el programa

```
CUADRADO50  
CUADRADO75  
CUADRADO100
```

¿Cómo se ve la figura que resulta?

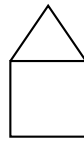
- ¿Cómo cambiarías el programa para que dibuje tres cuadrados adicionales más grandes que los anteriores?

Construyendo casas

Nuestra siguiente tarea es ayudar a un arquitecto que está diseñando una urbanización. Para que la construcción sea más sencilla, él está planeando construir todas las casas de la misma forma. Le hemos propuesto el siguiente diseño para una casa:

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

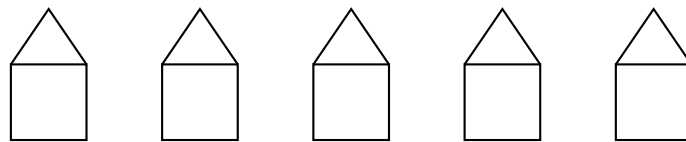
Este programa dibuja la casa que se muestra a continuación:



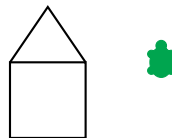
Ejercicio 26

¿Por dónde empieza la tortuga a dibujar la casa? Piensa sobre el camino que sigue la tortuga para dibujar la casa usando el programa **CASA**. ¿Dónde está la tortuga cuando termina de dibujar la casa? Dibuja la figura y describe el efecto de cada instrucción de la misma forma que lo hiciste en el Ejercicio 1.

El arquitecto manda a construir la casa de acuerdo a nuestro programa y queda satisfecho con el resultado. Ahora él quiere usar este programa como componente principal, para construir toda una calle de casas. Al final la calle debe verse como se muestra aquí:



Como todas las casas siguen el mismo patrón, el arquitecto puede usar el programa base **CASA** 5 veces, sin tener que pensar sobre el diseño de cada casa. Él deja que la tortuga dibuje la primera casa por la izquierda y luego le indica que se mueva al punto de inicio de la segunda casa:



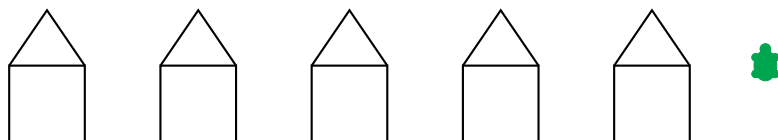
El arquitecto logra hacer esto usando el siguiente fragmento de programa:

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

Ahora que la tortuga está en la posición correcta, puede dibujar la siguiente casa de la misma forma que la anterior, y luego saltar nuevamente a la ubicación de inicio de la siguiente casa. La tortuga debe repetir este proceso hasta que las 5 casas sean dibujadas. Es decir, debemos repetir las instrucciones mostradas arriba cinco veces, para obtener una fila de 5 casas iguales. Llamaremos al programa que resulta **FILACASAS**:

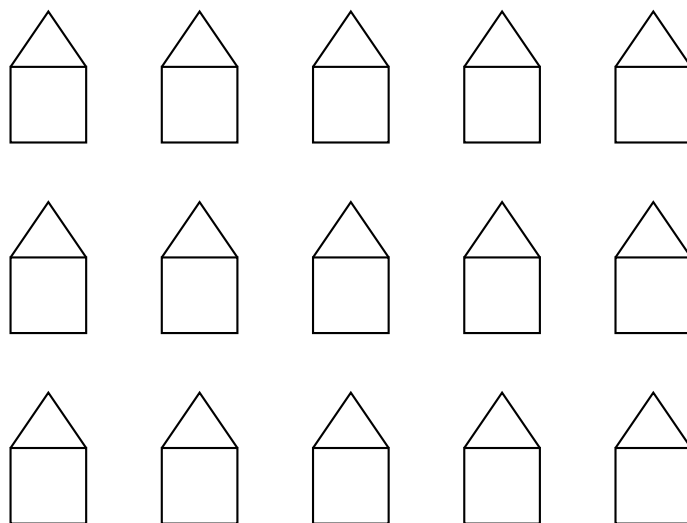
```
to FILACASAS
repeat 5 [CASA rt 90 pu fd 50 lt 90 pd]
end
```

Cuando el programa termina de ejecutarse, la tortuga se encuentra en la posición donde se comenzaría a dibujar la siguiente casa.



Ejercicio 27

Ahora queremos ampliar la urbanización con unas cuantas calles. Usa el programa **FILACASAS** como bloque base, para dibujar la figura que se muestra a continuación:



Una pista: Después de terminar una fila, la tortuga se tiene que mover a la posición correcta para poder comenzar a dibujar la siguiente.

Líneas gruesas y cuadrados negros

Ejercicio 28

Escribe el siguiente programa y guárdalo con el nombre **GRUESA**

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

y luego escribe en la línea de comandos

GRUESA

¿Qué dibuja la tortuga? Dibuja en una hoja de papel usando un lápiz, cómo es que se forma esta figura.

Ejercicio 29

Repite 100 veces el programa **GRUESA** con el programa

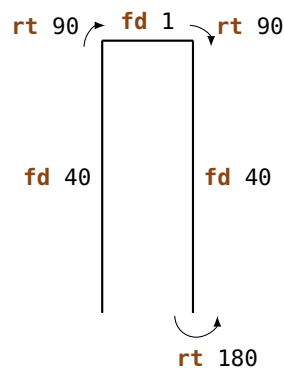
```
repeat 100 [GRUESA]
```

¿Qué aparece en la pantalla?

Ejercicio 30

En este ejercicio dibujaremos líneas gruesas y áreas oscuras. En el Ejercicio 28 hemos visto que una línea gruesa puede ser dibujada como sigue:

```
to GRUESA40  
fd 40  
rt 90  
fd 1  
rt 90  
fd 40  
rt 180  
end
```



Logramos hacer líneas gruesas si dibujamos dos líneas normales tan juntas que parecen ser una sola. Copia el programa **GRUESA40** y pruébalo.

Ejercicio 31

Una línea gruesa de largo 40 puede ser vista como un rectángulo de ancho 1 y largo 40. Después de ejecutar **GRUESA40**, la tortuga está en el extremo inferior de las dos líneas y mirando hacia arriba. Si el programa **GRUESA40** es ejecutado de nuevo, la tortuga repinta la segunda línea. Por lo tanto obtenemos un rectángulo de ancho 2 y largo 40. Con cada repetición añadimos una nueva línea. Si repetimos **GRUESA40** 40 veces, obtenemos un cuadrado “negro” con lado de longitud 40. Comprueba que esto es así, repitiendo **GRUESA40** 40 veces.

Escribe un programa con el nombre **NEGR040**, que dibuje un cuadrado negro con lado de longitud 40.

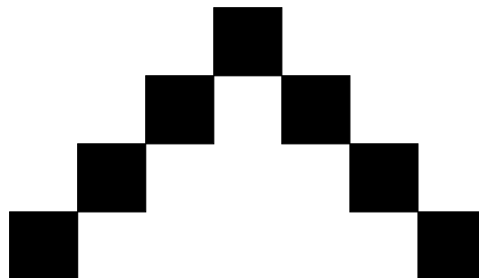
Ejercicio 32

Usa el programa **NEGR040** para dibujar la figura que se muestra:



Ejercicio 33

Utiliza el programa **NEGR040** para dibujar la siguiente figura:



Ejercicio 34

Dibuja la figura que se muestra a continuación:



Ejercicio 35

Escribe un programa que dibuje la siguiente figura:



Ejercicio 36

El arquitecto ha decidido encargar los techos de las casas a otro proveedor. Por ello, ahora tiene dos bloques de construcción: un bloque **TECHO** y un bloque **PARTEBAJA**. Escribe, para el arquitecto, dos programas que dibujen cada una de las partes y luego combínalos en un nuevo programa **CASA1** que dibuje la casa completa.

Ejercicio 37

Las casas en el Ejercicio 27 tienen un diseño bastante sencillo. Usa tu imaginación y crea un nuevo diseño para la casa, luego usa esta nueva casa para construir toda una nueva urbanización.

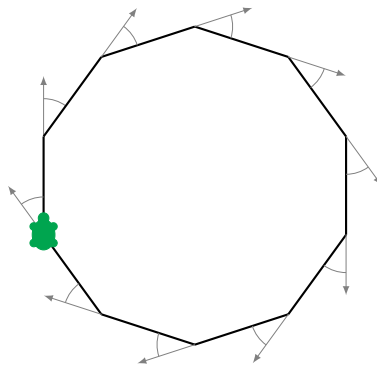
4 Polígonos regulares y círculos

Polígonos regulares

Un k -gono regular tiene k vértices (las “esquinas”) y k lados de igual longitud. Por ejemplo, para dibujar un decágono regular (polígono de 10 lados) con un lápiz debes trazar 10 líneas, de tal forma que después de cada línea cambias la dirección del trazo (giras) “sólo un poco”.

¿Pero cuánto necesitamos girar?

Cuando dibujamos un polígono regular giramos varias veces, pero al final debemos terminar en la misma posición donde empezamos y mirando en la misma dirección.



Esto significa que al terminar de dibujar habremos dado un giro total de 360° . Volviendo al mismo ejemplo, si dibujamos un decágono regular entonces habremos girado en total diez veces, y en cada vez giramos el mismo ángulo. Dicho ángulo es entonces:

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

Por tanto, debemos girar 36° cada vez: **rt 36**. Comprobemos esto escribiendo el siguiente programa:

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

Longitud del lado Giro de 36°

Ejercicio 38

Dibuja los siguientes polígonos regulares:

- (a) un pentágono regular (5 lados) con lado de longitud 180,
- (b) un dodecágono regular (12 lados) con lado de longitud 50,
- (c) un tetragono o cuadrilátero regular (4 lados) con lado de longitud 200,
- (d) un hexágono regular (6 lados) con lado de longitud 100,
- (e) un triángulo regular (3 lados) con lado de longitud 200, y
- (f) un octodecágono regular (18 lados) con lado de longitud 20.

Cuando dibujamos un heptágono regular (polígono de 7 lados), nos encontramos con el problema que 360 no es divisible entre 7 sin que quede un resto. En esos casos dejamos que la computadora calcule el resultado escribiendo

```
360/7
```

(«/» es el símbolo que usa la computadora para “entre”). La computadora entonces encuentra el valor exacto. De esta forma podemos dibujar un heptágono regular de lado 100:

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

Compruébalo en la computadora.

Dibujando círculos

No es posible dibujar círculos exactos usando sólo los comandos **fd** y **rt**. Sin embargo, quizás te has dado cuenta que un polígono regular con muchos vértices se ve casi como un círculo. De esa forma podemos dibujar aproximadamente un círculo usando un polígono con muchos lados (muy cortos) y muchos vértices.

Ejercicio 39

Prueba los siguientes 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 tres y medio pasos.

Ejercicio 40

- (a) ¿Cómo harías para dibujar un círculo muy pequeño? Escribe un programa para ello.
- (b) ¿Cómo harías para dibujar un círculo grande? Escribe un programa para ello.

Ejercicio 41

Intenta dibujar los semicírculos que se muestran a continuación. Puedes elegir los tamaños por tu cuenta:



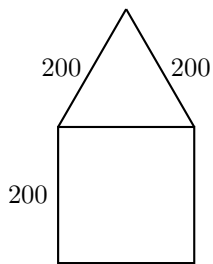
(a)



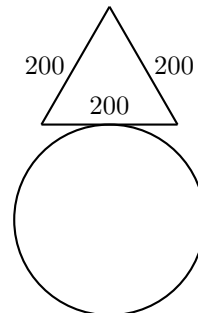
(b)

Ejercicio 42

Utiliza las ideas que acabas de aprender para dibujar las siguientes figuras. Puedes elegir el tamaño de los círculos por tu cuenta.



(a)



(b)

Dibujo libre

Dibuja un heptágono usando:

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

Luego haz que la tortuga gire 10° con

```
rt 10
```

Repite ambos pasos un par de veces, y mira la figura que resulta. Después de dibujar cada heptágono, giramos 10° con **rt 10**. Si queremos regresar a nuestra posición inicial, tenemos que repetir esta acción

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

veces. Ahora veamos lo que resulta al ejecutar el siguiente programa:

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

Ejercicio 43

Dibuja un polígono regular de 12 vértices con lado de longitud 70, y gíralo 18 veces hasta que hayas regresado a la posición inicial.

Pista: Puedes comenzar escribiendo un programa para dibujar un polígono de 12 lados (dodecágono) y luego llámalo por ejemplo **POLIGON012**. Después sólo necesitas completar el siguiente programa:


















```
repeat 18 [POLIGON012 rt ... ]
```

Ejercicio 44

Piensa en otra tarea similar al Ejercicio 43, y escribe un programa para resolverla.

Colores

Para poder dibujar figuras bonitas, necesitamos colores. La tortuga puede dibujar no sólo líneas negras, sino también de otros muchos colores. Cada color tiene asignado un número. La tabla que se muestra a continuación contiene una lista de todos los colores posibles:

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

Con el comando

setpencolor	X
Comando para cambiar el color	Número del color deseado

le indicamos a la tortuga que cambie el color actual por el color con el número **X**. También podemos usar la abreviatura del mismo comando: **setpc**.

Podemos hacer dibujos fantásticos utilizando colores, como por ejemplo el patrón que resulta de ejecutar el programa a continuación. Para empezar, creamos dos programas que dibujen círculos de distintos tamaños. Le damos un nombre a cada programa:

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

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

Luego usamos estos círculos para dibujar figuras similares a las que hemos visto anteriormente:

```
to PATRON3
repeat 36 [CIRCULO3 rt 10]
end

to PATRON1
repeat 18 [CIRCULO1 rt 20]
end
```

Ahora intentemos hacer lo mismo pero utilizando colores:

```
setpc 2
PATRON3 rt 2
setpc 3
PATRON3 rt 2

setpc 4
PATRON3 rt 2
setpc 5
PATRON3 rt 2
```

```
setpc 6
PATRON1 rt 2
setpc 15
PATRON1 rt 2
```

```
setpc 8
PATRON1 rt 2
setpc 9
PATRON1 rt 2
```

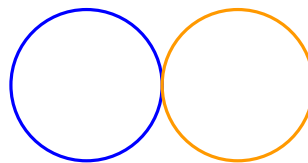
Siéntete libre de mejorar estos programas añadiéndoles más cosas, o si quieres puedes inventar patrones totalmente nuevos.

Ejercicio 45

Usa **PATRON3** para dibujar la figura correspondiente, pero de color naranja. Luego usa el comando **setpc 7**, para elegir el color blanco. ¿Qué pasa si ejecutas **PATRON3** de nuevo?

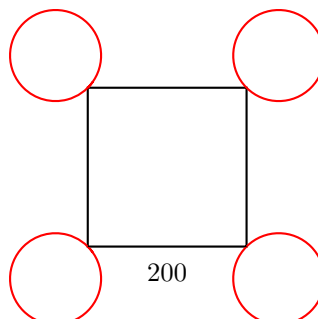
Ejercicio 46

Dibuja la figura que se muestra. La tortuga está al inicio en el punto común a ambos círculos (la intersección).



Ejercicio 47

Escribe un programa que dibuje la siguiente figura. Puedes elegir el tamaño de los círculos por tu cuenta.



5 Programas con parámetros

En la Lección 3 hemos aprendido cómo asignarle nombres a nuestros programas y luego usar esos nombres para llamarlos y dibujar las figuras correspondientes en la computadora. Luego, en la Lección 4 aprendimos a dibujar polígonos. Sin embargo, es muy tedioso tener que escribir un programa nuevo para cada polígono con diferente número de lados.

Ahora observemos por ejemplo los tres programas que se muestran:

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

Estos programas son muy similares y sólo difieren en los números resaltados en amarillo 7, 12 y 18. Estos números determinan el número de vértices (y lados) del polígono. Ahora queremos escribir un programa que pueda dibujar un polígono con cualquier número de vértices:

```
to POLIGONO :VERTICES
repeat :VERTICES [fd 50 rt 360/:VERTICES]
end
```

¿Pero qué hemos hecho aquí? En todos los sitios donde aparecía el número de vértices del polígono, escribimos en vez del número el nombre **:VERTICES**, para que la computadora sepa que luego queremos indicar el número de vértices libremente. No olvides que también debemos escribir **VERTICES** precedido por el símbolo **:** después del nombre del programa.

Si luego escribimos el comando **POLIGONO 12** en la línea de comandos, la computadora coloca en el programa **repeat** 12 **:VERTICES** **[fd 50 rt 360/**12**]** el número 12 en todos los lugares donde **:VERTICES** aparece y dibuja por tanto un dodecágono.

Comprobémoslo en la computadora:

```
POLIGONO 3
POLIGONO 4
POLIGONO 5
POLIGONO 6
```

A **:VERTICES** lo llamamos un **parámetro**. En el ejemplo de arriba los números 3, 4, 5 y 6 son los **valores del parámetro :VERTICES**. La computadora reconoce un parámetro por el símbolo **:**. Es por eso que en todos los sitios donde utilicemos un parámetro, el nombre de éste debe estar precedido por el símbolo **:**.

Ejercicio 48

Los siguientes programas dibujan cuadrados de diferentes tamaños:

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

Los números en amarillo 100, 50, 200 pueden ser vistos como valores de un parámetro que determina el tamaño del cuadrado. Escribe un programa con el parámetro **:LADO** para poder dibujar cuadrados de cualquier tamaño.

```
to CUADRADO :LADO
...
end
```

Ejercicio 49

Los siguientes programas dibujan círculos de diferentes tamaños:

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

Escribe un programa con un parámetro para dibujar círculos de cualquier tamaño, y pruébalo luego con los valores 1, 2, 3, 4 y 5. Puedes elegir el nombre del parámetro por tu cuenta pero no olvides que los dos puntos **:** deben estar siempre delante del parámetro.

Ejercicio 50

¿Todavía recuerdas cómo podíamos dibujar líneas gruesas (Ejercicio 28)? Escribe un programa con un parámetro, para poder dibujar líneas gruesas de cualquier longitud.

Pista: Puedes empezar escribiendo un programa para una línea de longitud 100 y para otra de longitud 50, y observar qué valor puede ser el parámetro de tu nuevo programa.

Ejercicio 51

Escribe un programa con un parámetro, para dibujar triángulos de cualquier tamaño. Usa tu programa para dibujar triángulos de tamaño

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

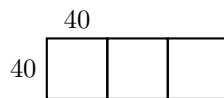
uno después del otro. ¿Qué es lo que obtienes?

Ejercicio 52

Ahora queremos dibujar cuadrados de lado 40, uno al lado del otro. Escribe un programa **CUADRADOS** con un parámetro **:NC**. El parámetro **:NC** indica cuántos cuadrados deben ser dibujados. Por ejemplo, cuando ejecutamos **CUADRADOS 6**, la tortuga dibuja la figura siguiente:

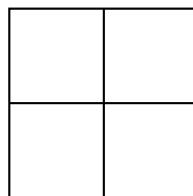


Esta otra figura será dibujada después de ejecutar **CUADRADOS 3**:



Ejercicio 53

Escribe un programa que dibuje la figura mostrada, que está compuesta de 4 cuadrados. El tamaño del cuadrado está determinado por un parámetro.

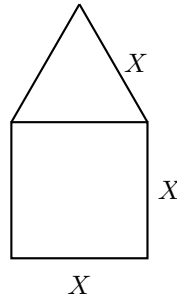


Ejercicio 54

Escribe un programa con un parámetro, para dibujar hexágonos de cualquier tamaño. Prueba tu programa dibujando hexágonos con longitud de lado 40, 60 y 80.

Ejercicio 55

Escribe un programa con un parámetro `:X`, que pueda dibujar casas de cualquier tamaño de acuerdo al siguiente modelo:



Programas con múltiples parámetros

Un programa puede tener más de un parámetro. Si estamos dibujando polígonos, por ejemplo, el programa puede tener un parámetro `:VERTICES` para el número de vértices y otro parámetro `:LADO` para indicar la longitud del lado.

En los siguientes programas el parámetro `:VERTICES` está resaltado en amarillo y el parámetro `:LADO` está resaltado en 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]
```

De esta forma podemos escribir un programa con dos parámetros que pueda dibujar cualquier polígono regular:

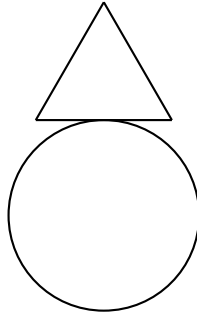
```
to POLI :VERTICES :LADO
repeat :VERTICES [fd :LADO rt 360/:VERTICES]
end
```

Prueba el programa `POLI` usando las siguientes llamadas:

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


Ejercicio 56

Escribe un programa con dos parámetros, que pueda dibujar la siguiente figura. El tamaño del círculo y del triángulo deben poder elegirse ambos libremente.



Ejercicio 57

El programa

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

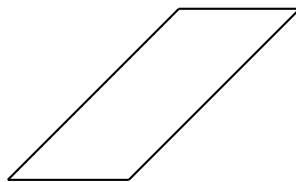
dibuja un rectángulo de ancho 100 y largo 200. Copia el programa para verificar que realmente funciona. Luego escribe otro programa con dos parámetros, que pueda dibujar rectángulos con el ancho y largo que sea.

Ejercicio 58

El siguiente programa

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

dibuja un paralelogramo:



Escribe un programa con dos parámetros, que pueda dibujar paralelogramos de cualquier tamaño.

Ejercicio 59

Dibuja una flor, de la siguiente manera: Comienza dibujando un círculo usando

```
POLI 360 2
```

luego haz girar a la tortuga ligeramente hacia la derecha

```
rt 20
```

y dibuja otro círculo con

```
POLI 360 2
```

Repite la misma operación varias veces con

```
rt 20 POLI 360 2 rt 20 POLI 360 2
```

...

Cuando la flor esté terminada, la tortuga debe encontrarse de nuevo en su posición inicial. Para entonces la tortuga habrá dibujado 18 círculos, habiendo girado 20° entre cada uno de ellos. Así que en total la tortuga habrá girado $18 \times 20^\circ = 360^\circ$.

Podemos escribir el programa completo como sigue:

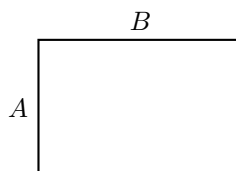
```
repeat 18 [POLI 360 2 rt 20]
```

Compruébalo en la computadora.

- También puedes dibujar una flor con 10 pétalos (círculos) o aún con 20 pétalos. ¿Cómo harías esto? Escribe un programa y Pruébalo.
- ¿Puedes escribir un programa con un parámetro para dibujar una flor con cualquier número de pétalos (círculos)?
- ¿Puedes escribir un programa que use los siguientes valores como parámetros:
 - el número de pétalos (círculos) y
 - el tamaño de los círculos?

Ejercicio 60

Escribe un programa para dibujar un rectángulo de cualquier color:



Esto significa que no solamente los lados *A* y *B* sino también el color deben poder elegirse libremente.

6 Dibujar flores y pasar parámetros a subprogramas

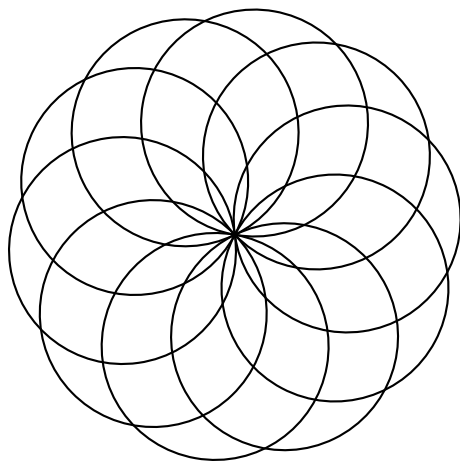
En esta lección vamos a aprender a dibujar flores. Vamos a seleccionar su forma y color con la ayuda de parámetros de tal manera que nuestra tortuga pueda dibujar bonitos, coloridos y fantásticos patrones.

Observemos el siguiente programa:

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

Este programa ya lo tenemos en el editor. Ahora podemos dibujar una flor con 10 pétalos con el programa:

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



Ejercicio 61

Alguien quiere dibujar una flor con 12 pétalos. ¿Cómo podríamos modificar el programa de arriba?

Ejercicio 62

Dibuja una flor con 12 pétalos y lo doble de grandes que antes.

Ahora queremos escribir en el editor un programa para hacer flores, en el cual el tamaño de los pétalos se puede elegir al ejecutar. Eso significa, que queremos utilizar el subprograma **CIRCULO :TAM** y de esta manera tener la opción de elegir para **:TAM**. Esto se puede sólo si el programa para la flor también contiene el parámetro para elegir el tamaño de los pétalos.

Escribe en el editor

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

llama **FLOR 1**, **FLOR 2** y **FLOR 3** y observa los dibujos. ¿Qué sucedió? Si nosotros hemos llamado **FLOR 1**, EL 1 será sustituido como valor de **:TAM**. De esta manera será llamado el subprograma **CIRCULO :TAM** como: **CIRCULO 1**.

Ejercicio 63

Describe, que sucede si ejecutas **FLOR 2**.

Ejercicio 64

Reflexiona, lo que hace el siguiente programa y después compruébalo.

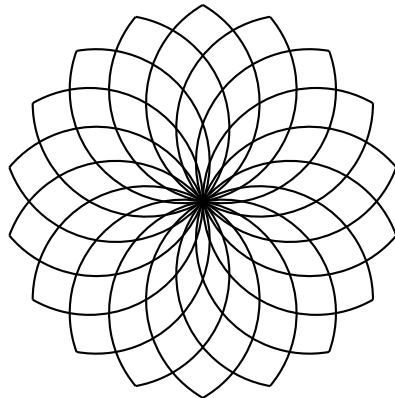
```
to FLORES :TAM1 :TAM2
setpc 3 FLOR :TAM1
setpc 4 FLOR :TAM2
end
```

Ejercicio 65

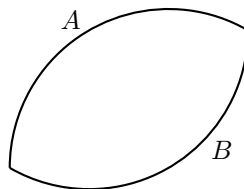
Queremos reelaborar el programa **FLOR** y darle el nombre **FLOR1**. De tal manera que ahora no sólo se puede elegir el tamaño de los pétalos sino también el número de pétalos. ¿Sabes cómo hacer esto?

Una flor con pétalos puntiagudos

¿Quieres aprender a dibujar flores con pétalos puntiagudos? ¿Te gusta la siguiente flor?



Para dibujar una flor así, tenemos que reflexionar primero, como dibujar cada pétalo. Un pétalo



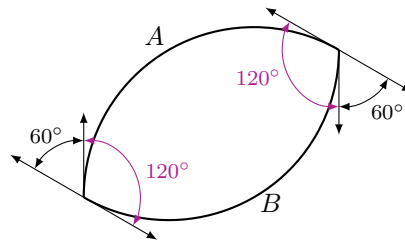
puede ser visto como dos arcos (o “trozos”) de circunferencia A y B unidos por los extremos. Un arco de circunferencia puede ser dibujado, por ejemplo, utilizando el siguiente programa:

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

Pruébalo en la computadora.

Podemos observar que este programa es muy parecido al que usamos para dibujar un círculo. En vez de dar 360 pequeños pasos con giros de 1° , ahora damos 120 pequeños pasos [**fd** 2 **rt** 1], y por tanto sólo dibujamos un tercio del círculo ($\frac{360^\circ}{3} = 120^\circ$).

La pregunta que queda es, ¿cuánto debe girar la tortuga antes de empezar a dibujar el arco B , que formá la parte de abajo del pétalo?. Veamos la siguiente ilustración:



Si queremos regresar a nuestra posición inicial después de haber terminado de dibujar el pétalo, debemos hacer girar a la tortuga 360° en total. Al dibujar el arco A , la tortuga gira 120° y para dibujar el arco B gira otros 120° . Entonces el ángulo que resta por girar es

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

Este giro restante debe ser repartido por igual en ambas puntas del pétalo:

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

Al final obtenemos el siguiente programa:

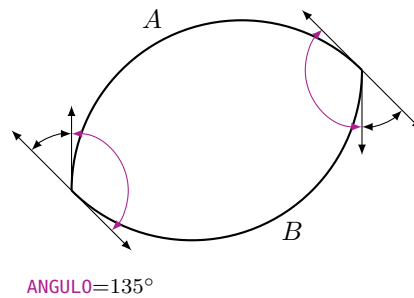
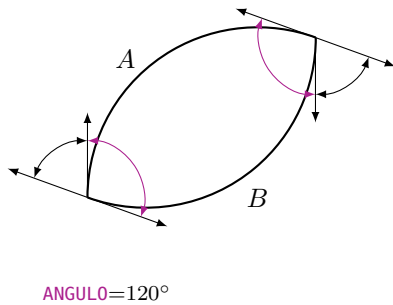
```
repeat 120 [fd 2 rt 1]
rt 60
repeat 120 [fd 2 rt 1]
rt 60
```

o si lo hacemos aún más sencillo:

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

Pruébalo en la computadora.

Ahora quisiéramos también poder dibujar pétalos más delgados (en los cuales los arcos A y B son más cortos) o más anchos (si los arcos A y B son más largos).



Para ello podemos agregar un nuevo parámetro. Llamemos al parámetro, por ejemplo, **:PARTE**. Luego podemos calcular el ángulo de giro en las puntas del pétalo como sigue:

Antes de comenzar a dibujar la parte B del pétalo, debemos haber completado la mitad del giro total, es decir $\frac{360^\circ}{2} = 180^\circ$. Entonces el ángulo que debemos girar en la punta del pétalo es dado por

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

De esa forma podemos escribir nuestro programa en el editor:

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

Prueba el programa que acabamos de escribir llamándolo varias veces desde la línea de comandos como se muestra a continuación:

```
PETALO 20
PETALO 40
PETALO 60
PETALO 80
PETALO 100
```

¿Qué es lo que obtienes?

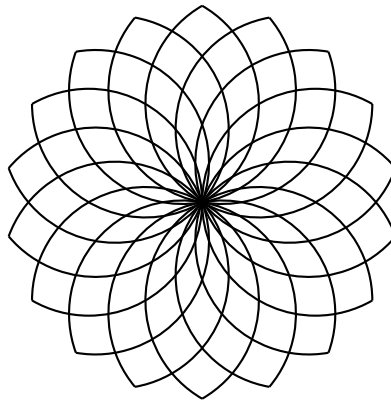
Una flor tiene muchos pétalos puntiagudos

Ahora queremos usar **PETALO** como subprograma, para dibujar flores con pétalos puntiagudos.

Ejercicio 66

Dibuja primero una flor con el siguiente programa:

```
PETALO 100  
rt 20  
PETALO 100  
rt 20  
PETALO 100  
....
```



¿Qué tan frecuente tienes que repetir los comandos **PETALO** y **rt 20**, para dibujar toda la flor?

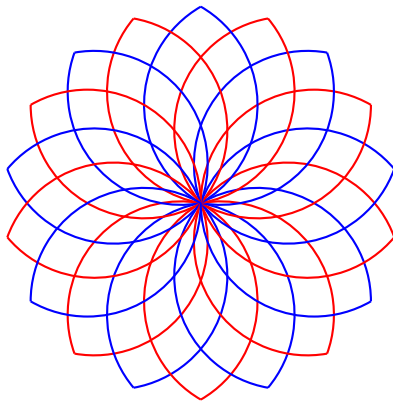
Escribe el programa para la flor en un sólo renglón con un apropiado comando **repeat**". (Pero recuerda que todos los giros **rt** entre cada uno de los pétalos tienen que dar en total 360°)

Ejercicio 67

Teclea el programa del Ejercicio 66 en el editor. Nombra el programa **FLOR3**. El programa debe tener el parámetro **:PARTE**. ¿Qué sucede si tecleas **FLOR3 60**, **FLOR3 80** und **FLOR3 100**?

Ejercicio 68

- Escribe un programa con un parámetro, que dibuje la flor del Ejercicio 66 de un color que pueda elegirse al ejecutar. Nombra tu programa **FLOR4**.
- Modifica tu programa, dale el nombre **FLOR5**, de tal manera que el número de pétalos que se dibujarán sean definidos por un nuevo parámetro **:NUMD**. Pero recuerda, que todos los giros **rt** entre cada uno de los pétalos debe dar como resultado 360° .
- Modifica el programa **FLOR5** de tal manera que la flor sea dibujada en dos colores que podamos elegir Nombra el nuevo programa **FLOR6**.



Ejercicio 69

En el programa **PETALO** el comando **fd 2** define el tamaño del círculo, del cual cortamos un trozo del ángulo **:PARTE** Este valor 2 también lo podemos sustituir por medio de un parámetro **:TAM** (Tamaño). Escribe un programa

```
PETALOS :PARTE :TAM
```

con el parámetro **:PARTE** y **:TAM** podemos definir el trozo del círculo y el tamaño. mit denen wir den Teilkreis und die Größe einstellen können. Pruébalo con las siguientes ejecuciones

```
PETALOS 100 1
PETALOS 100 1.5
rt 100
PETALOS 80 2
PETALOS 80 2.5
```

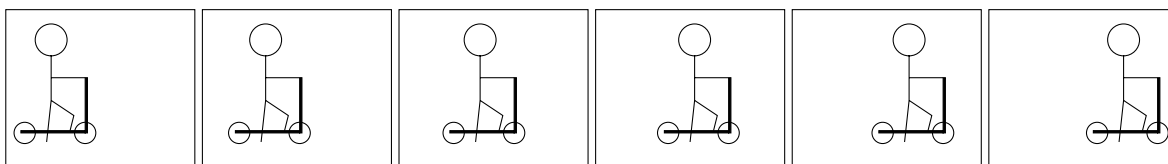
Después gira la tortuga hacia la derecha por 80° y repite el programa de arriba.

Ejercicio 70

Diseña o imagina tus propias figuras.

7 Dibujos animados

¿Sabes cómo se hace una película de dibujos animados? La idea es la misma que con un folioscopio (uno de esos libros con muchas figuras que cambian ligeramente de una página a la siguiente). Para hacer un dibujo animado, primero dibujamos dos figuras del mismo objeto pero con sólo una ligera variación en la posición. Por ejemplo, en los recuadros mostrados abajo tenemos dibujos de un niño en una patineta. En cada recuadro, el niño se ha movido un poco más hacia la derecha:

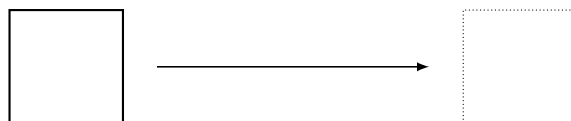


Regresando al ejemplo del “folioscopio” (nuestro libro animado), si colocamos todas las figuras una encima de otra y pasamos las páginas rápidamente con el dedo pulgar, da la impresión que el niño con su patineta se está moviendo de izquierda a derecha. A estas figuras con movimiento las llamamos **animaciones**.

En esta lección aprenderemos cómo programar animaciones con ayuda de la tortuga.

Un cuadrado que deja un rastro

Para nuestra primera animación elegiremos una figura que no es difícil de hacer y que ya conocemos muy bien: Haremos que un cuadrado se mueva de izquierda a derecha.



El programa para dibujar un cuadrado ya lo conocemos desde antes:

```
to CUAD100
repeat 4 [fd 100 rt 90]
end
```

Después de dibujar una vez el cuadrado, desplazamos la tortuga un poco hacia la derecha y dibujamos nuevamente el cuadrado. Luego, desplazamos la tortuga otra vez a la derecha y dibujamos otro cuadrado. Repetimos esta operación unas cuantas veces.

En el programa siguiente, dibujamos 120 cuadrados de esa manera:

```
to MOVERCUADRADO
repeat 120 [CUAD100 rt 90 fd 4 lt 90]
end
```

Ejercicio 71

Escribe los programas **CUAD100** y **MOVERCUADRADO** en el editor y ejecuta **MOVERCUADRADO**. ¿Qué se dibuja en la pantalla?

Puedes ver que la huella de *todos* los cuadrados se queda dibujada en la pantalla. Sin embargo, para nuestra animación queremos ver solamente la última posición del cuadrado y borrar las huellas anteriores.

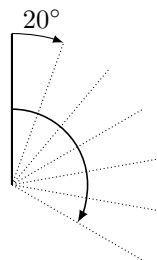


Ejercicio 72

Haz que el cuadrado se mueva de abajo hacia arriba en vez de izquierda a derecha.

Ejercicio 73

Escribe un programa para dibujar una línea de longitud 20. Utiliza ese programa para hacer que una línea gire sobre su extremo inferior en el sentido de las manecillas del reloj:



Dibujar un cuadrado y borrarlo cada vez

Para poder deshacernos de los rastros, tenemos que aprender cómo borrar figuras que acabamos de dibujar. Para ello la tortuga debe usar un borrador en lugar del lápiz. Con el nuevo comando **penerase**, abreviado como **pe**, le decimos a la tortuga que cambie su lápiz por un borrador.

Ejercicio 74

Trata de imaginar qué hace el programa `CUAD100 pe CUAD100` sin ejecutarlo en la computadora.

En caso que la tortuga tenga que empezar a dibujar nuevamente, debemos indicárselo claramente. Para ello hay también un nuevo comando: **penpaint**, abreviado como **ppt**. Podemos usar el nuevo comando directamente en el programa del Ejercicio 74.

Ahora el programa se ve de la siguiente manera:

```
CUAD100 pe CUAD100 ppt
```

Ejercicio 75

Ejecuta el programa mostrado arriba. ¿Qué ocurre? ¿Puedes explicar por qué?

El cuadrado debe esperar un poco

Seguramente habrás notado al resolver el Ejercicio 75, que el cuadrado se borra muy rápidamente después de ser mostrado. Ni siquiera nos damos cuenta que un cuadrado ha sido dibujado, así de rápida es la operación. Para solucionar esto, debemos decirle a la computadora que espere un poco antes de borrar el cuadrado.

Podemos hacerlo como sigue:

<code>wait</code>	4
Primitiva para esperar	Tiempo de espera

Ejercicio 76

Prueba el programa

```
CUAD100 wait 4 pe CUAD100 ppt
```

Un cuadrado que se mueve de izquierda a derecha

Ahora estamos listos para incluir en nuestro programa **MOVERCUADRADO** los comandos que borran el cuadrado e indican el tiempo de espera:

```
to MOVERCUADRADO
repeat 120 [CUAD100 wait 4 pe CUAD100 rt 90 fd 4 lt 90 ppt]
end
```

Pruébalo. En caso que la tortuga interfiera con la animación al aparecer dibujando los cuadrados, podemos comenzar el programa con el comando **hideturtle** (o abreviado: **ht**), que hace invisible a la tortuga. Te podrás dar cuenta de inmediato, que la animación se vuelve más rápida. Termina el programa con el comando **showturtle** (o abreviado: **st**) directamente antes de **end**. Así hacemos que la tortuga vuelva a hacerse visible.

Ejercicio 77

Haz que un cuadrado de tamaño 50×50 se mueva hacia arriba.

Ejercicio 78

Modifica el programa **MOVERCUADRADO** para que el cuadrado se desplace hacia la derecha el doble de rápido.

Ejercicio 79

¿Puedes también cambiar el programa **MOVERCUADRADO** para que el cuadrado se mueva hacia la derecha pero ahora la mitad de rápido?

Ejercicio 80

Modifica el programa **MOVERCUADRADO** para que el cuadrado se mueva ahora de derecha a izquierda en vez de izquierda a derecha.

Ejercicio 81

Primero piensa de antemano qué es lo que hace el siguiente programa, y luego comprueba tu suposición ejecutándolo:

```
to MOVERCUADRADO1
ht
repeat 50 [CUAD100 wait 5 pe CUAD100 fd 3 rt 90 fd 3 lt 90 ppt]
CUAD100
st
end
```

Ejercicio 82

Piensa primero qué es lo que hace el siguiente programa, y luego compruébalo ejecutándolo en la computadora:

```
to ROTACION
ht
repeat 360 [CUAD100 wait 4 pe CUAD100 fd 5 rt 1 ppt]
CUAD100
st
end
```

Ejercicio 83

Modifica el programa **ROTACION** para que el cuadrado se mueva cuatro veces más rápidamente.

Ejercicio 84

¿Qué hace el siguiente programa?

```
repeat 6 [ROTACION]
```

Ejercicio 85

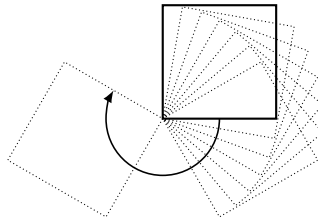
Toma el siguiente programa

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

y úsalo para hacer una animación en la cual la Tierra se mueva en una órbita circular alrededor del Sol. Usa tu imaginación para decidir la forma exacta en la que se dibujará el Sol.

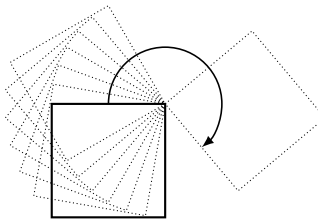
Ejercicio 86

Haz que un cuadrado gire en el sentido de las manecillas del reloj sobre su esquina inferior izquierda. Puedes decidir por tu cuenta la longitud del lado del cuadrado:



Ejercicio 87

Ahora haz que el cuadrado gire en el sentido de las manecillas del reloj pero sobre su esquina superior derecha:



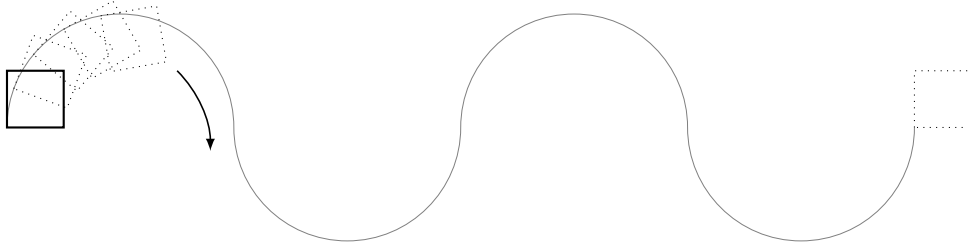
Si ya sabes cómo trabajar con parámetros, puedes desarrollar los ejercicios que siguen.

Ejercicio 88

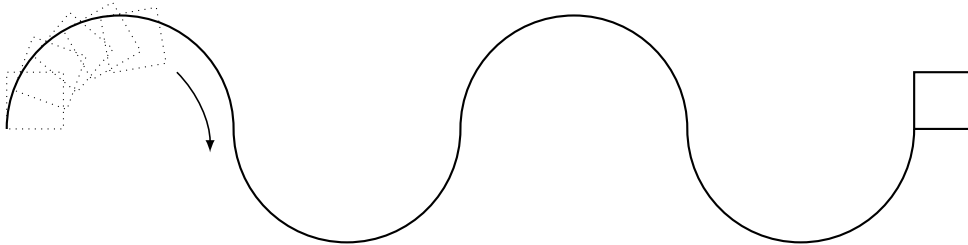
Escribe un programa con *dos parámetros* que haga que un cuadrado se mueva de izquierda a derecha. Un parámetro debe indicar la longitud del lado, y el otro debe indicar qué tan rápido se mueve el cuadrado.

Ejercicio 89

- (a) Haz que un cuadrado se mueva siguiendo el camino dibujado abajo, que está conformado por 4 medios círculos. La longitud del lado del cuadrado debe ser indicada con un parámetro.



- (b) Ahora haz que el camino se vaya dibujando como un rastro mientras el cuadrado se va moviendo.



- (c) ¿Puedes ampliar el programa de la parte (b), para que el número de medios círculos en el camino se pueda indicar también con un parámetro?

Resumen de comandos

- fd 100** dar 100 pasos hacia adelante
- bk 50** dar 50 pasos hacia atrás
- cs** limpiar la pantalla y comenzar de nuevo
- rt 90** girar 90 grados hacia la derecha
- lt 90** girar 90 grados hacia la izquierda
- repeat 4 [...]** el programa en [...] se repite 4 veces
 - pu** la tortuga entra en modalidad de movimiento
 - pd** la tortuga regresa a modalidad de dibujo
- setpc 3** cambiar el color del lápiz al color 3
- to NOMBRE** crear un programa con un nombre
- to NOMBRE :PARAMETRO** crear un programa con un nombre y un parámetro
 - end** todos los programas con nombre terminan con este comando
 - pe** la tortuga entra en modalidad de borrado
 - ppt** la tortuga regresa a modalidad de dibujo desde la de borrado
- wait 5** hacer que la tortuga espere 5 unidades de tiempo



Programar con LOGO

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

www.ite.ethz.ch
www.abz.inf.ethz.ch