

Objetivos

Esta quincena aprenderá sobre:

- Conocer y comprender los conceptos relacionados con la comunicación y control de un sistema.
- Entender e implementar programas sencillos en un lenguaje de programación sencillo como MSWLOGO.
- Comprender el funcionamiento básico de una tarjeta controladora.

Antes de empezar

1.Comunicación con el ordenador.....	pág. 2
Lenguaje	
Tarjetas controladoras	
Entradas y salidas	
2.Primitivas de programación.....	pág. 7
Programación en lenguaje LOGO	
Entorno de MSWLOGO	
Primitivas básicas	
Usando colores	
Creando secuencias repetitivas	
Mostrar texto: rotula	
Crear ventanas y botones	
Procedimientos	
Control de tiempo: espera	
3.Variables de programación.....	pág. 17
Concepto	
Usando variables en texto	
Procedimientos con variables	
Variables de datos en ventanas	
Variables de azar	
4.Control del programa.....	pág. 21
Evaluando condiciones	
Bucles: Órdenes reiterativas	
Primitivas de control de la tarjeta	
5.Recuerta lo más importante.....	pág.26
6.Para practicar.....	pág. 27
7.Autoevaluación.....	pág. 38
8.Para saber más.....	pág. 48

Contenidos

1. Comunicación con el ordenador

Lenguaje

El **lenguaje de programación** o **código** es la forma que tenemos de comunicarnos con un ordenador; los circuitos electrónicos digitales de la máquina le permiten transformar nuestras instrucciones en señales eléctricas.

Recordemos que es el **código binario** el que nos permite interactuar con la máquina. A los **ceros** y **unos** se les llama con el nombre de **lenguaje máquina** porque son instrucciones que el sistema electrónico es capaz de comprender (pasa o no pasa corriente).

Pero sería muy laborioso traducir a ceros y unos las instrucciones que queremos que realice el ordenador. El código nos permite recurrir a un lenguaje mucho más comprensible para nosotros, llamado **lenguaje de programación de alto nivel**; bajo este código existe un subcódigo encargado de traducir nuestras instrucciones al lenguaje máquina, es decir, a ceros y unos, pero ya no necesitamos conocerlo.

Existen cientos de lenguajes de programación y son muy diversos; algunos de ellos son muy antiguos, mientras que otros son relativamente nuevos. Además, varían mucho según su complejidad, existiendo algunos más o menos simples. A finales de los años 60 se desarrolló un lenguaje sencillo para educación llamado **LOGO**. Este lenguaje es el que vamos a utilizar en esta unidad didáctica.

[Imagen por Scientus en en.wikipedia.org](https://en.wikipedia.org)



Contenidos

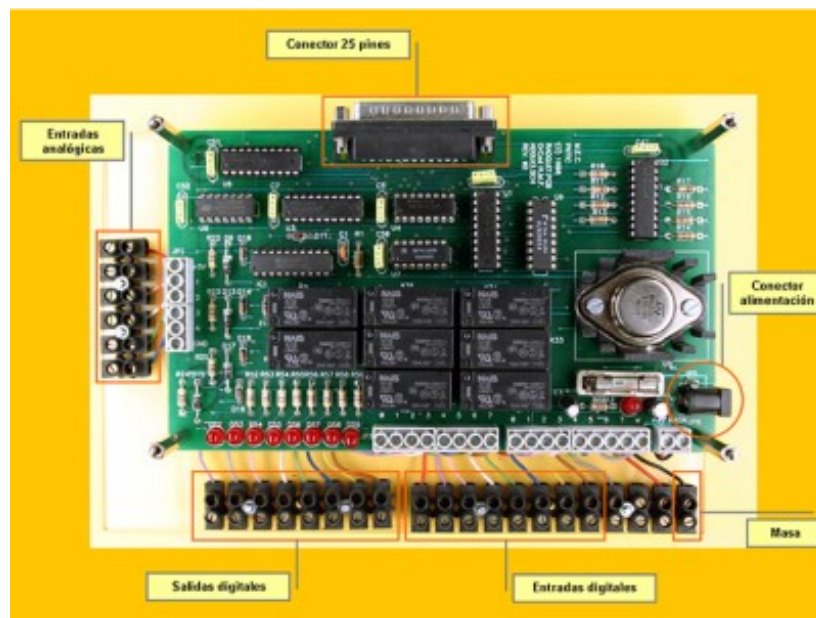
1. Comunicación con el ordenador

Tarjetas controladoras

Las tarjetas de control o controladoras sirven de enlace entre el ordenador y el sistema a controlar o un robot. Reciben las instrucciones del ordenador en forma digital y tienen que convertirlas en señales, normalmente analógicas, que sean comprensibles para el robot; y viceversa, también tienen que recibir las señales del sistema robótico y enviárselas al ordenador para su procesamiento. Es decir, estas tarjetas tienen una serie de entradas y una serie de salidas.

Existen diferentes tipos de controladoras pero su apariencia es la de cualquier circuito impreso. Las controladoras necesitan su propia fuente de alimentación.

Tarjeta controladora. Fuente ITE



Contenidos

1. Comunicación con el ordenador

Entradas y salidas

Los controladores disponen de varias entradas y salidas; éstas pueden ser analógicas o digitales:

- **Digitales** admiten solamente información del tipo pasa-no pasa; permitirán o no el paso de la corriente por el circuito; son adecuadas para conectar elementos del sistema robótico como lámparas, diodos LED o indicadores del funcionamiento de la máquina.
- **Analógicas**, en cambio, permiten regular la cantidad de corriente que pasa (recordemos que las variables analógicas admiten cualquier valor). Serán adecuadas si queremos regular la luz que emite una bombilla, la velocidad de giro de un motor .

Analógico	Digital
Luz (más o menos intensidad)	Luz (encendido / apagado)
Motor (más o menos velocidad)	Motor (encendido /apagado)

Contenidos

2. Primitivas de programación

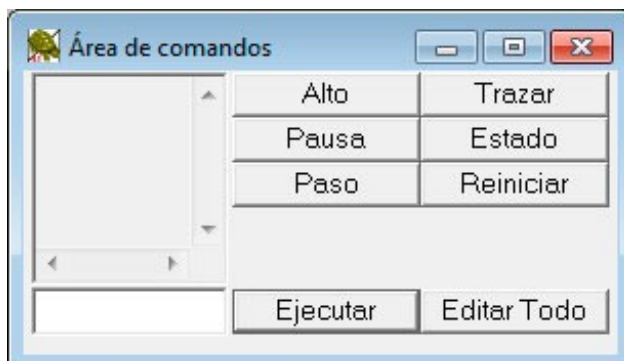
Programación en Lenguaje LOGO

Una vez que tenemos una idea de cómo se transmiten nuestras órdenes al robot, vamos a empezar a estudiar las órdenes, es decir, los lenguajes de programación; por muy variados que sean estos, su estructura es similar: siempre parte de unas órdenes básicas denominadas **primitivas**.

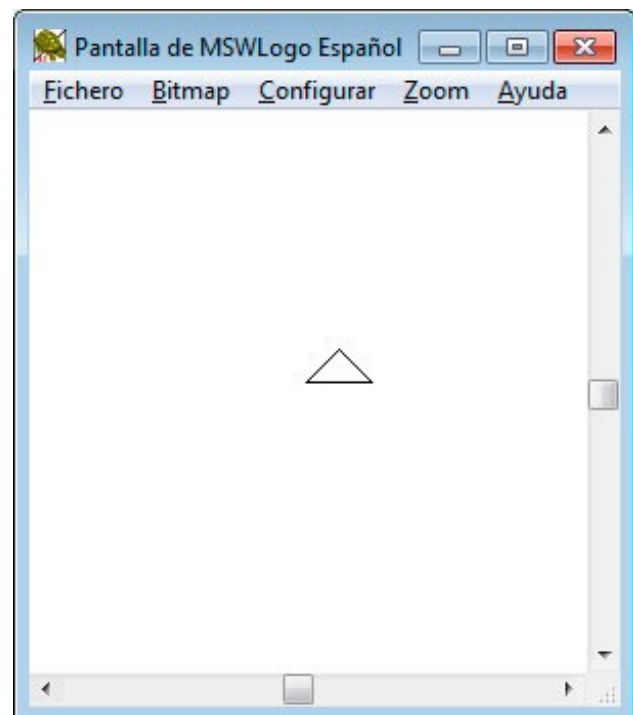
Vamos a estudiar las primitivas que usa el software **MSWLogo**, basado en el lenguaje **Logo**, pensado con fines didácticos para explicar los conceptos básicos de programación.

Entorno de MSWLOGO

La pantalla de Logo se divide en dos partes; Un **lienzo**, en la zona superior donde podemos ver la posición y dirección del **cursor** y el resultado de las órdenes que hayamos dado hasta entonces. En la zona inferior podemos ver la **consola de comandos** con el **código** que hayamos escrito.



Consola de comandos



Lienzo

Contenidos

2. Primitivas de programación

Primitivas básicas

Las órdenes o instrucciones básicas de MSWLogo se llaman **primitivas**. Las primitivas deben escribirse en el cuadro inferior de la ventana de trabajo. Al pulsar Enter o hacer clic en el botón Ejecutar, la primitiva se ejecuta. Si la primitiva es mal escrita o si le faltan datos, el intérprete contesta "no sé cómo..." Cada una de las órdenes queda anotada en la ventana de trabajo.

La primitiva puede escribirse completa o mediante abreviaturas y tanto en minúsculas como en mayúsculas.

- **AVANZA (AV)**

Sirve para desplazar el cursor una determinada longitud que debe de especificarse a continuación de AVANZA o de AV. Por ejemplo, AV 40 desplazará el cursor 40 unidades. Es importante separar con un espacio la primitiva (AV) del parámetro (40) o el programa no entenderá la orden.

- **GIRADERECHA (GD) Y GIRAIZQUIERDA (GI)**

Sirven para cambiar la orientación del cursor; el parámetro que acompaña a la primitiva será el número de grados de giro y tendrá que ir separado mediante un espacio de la primitiva. Al abrir el programa, el cursor está orientado en dirección vertical con sentido hacia arriba.

GD 90 desplazará el cursor 90 grados en el sentido de las agujas del reloj. GD 180 o GI 180 da la vuelta al cursor y lo pone en posición de retroceso.

- **RETROCEDE (RE)**

Hace retroceder el cursor el número de unidades que le indique el parámetro que venga a continuación. Es equivalente a GD 180 o a GI 180 seguido de AV.

- **BORRAPANTALLA (BP)**

Pone la pantalla en blanco y vuelve a llevar el cursor a su posición y su orientación inicial.

- **BORRATEXTO (BT)**

Borra todo el código escrito en el área de comandos.

- **SUBELAPIZ (SL)**

Sirve para avanzar sin que el rastro quede marcado.

- **BAJALAPIZ (BL)**

Al avanzar el rastro deja marca; es la opción que está seleccionada por defecto si no hemos utilizado SL.

- **CIRCLE**

Dibuja un círculo; debe ir acompañada de un parámetro que indique el valor del radio.

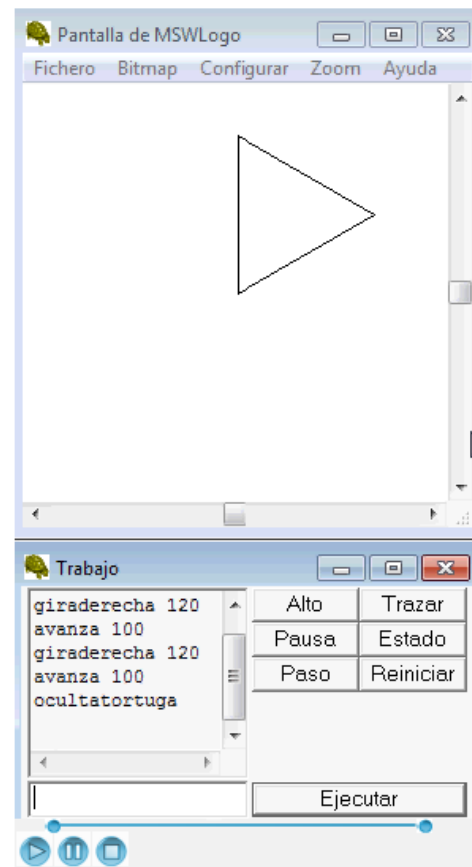
Contenidos

2. Primitivas de programación

Primitivas básicas

Código fuente de un programa para dibujar un triángulo equilátero de lado 100

Comando largo	Abreviado	Explicación
AVANZA 100	AV 100	Hacemos que la tortuga avance 100
GIRADERECHA 120	GD 120	La tortuga gira 120 grados hacia la derecha
AVANZA 100	AV 100	Hacemos que la tortuga avance 100
GIRADERECHA 120	GD 120	La tortuga gira 120 grados hacia la derecha
AVANZA 100	AV 100	Hacemos que la tortuga avance 100
GIRADERECHA 120	GD 120	La tortuga gira 120 grados hacia la derecha
OCULTATORTUGA	OT	Hace que la tortuga desaparezca de la pantalla



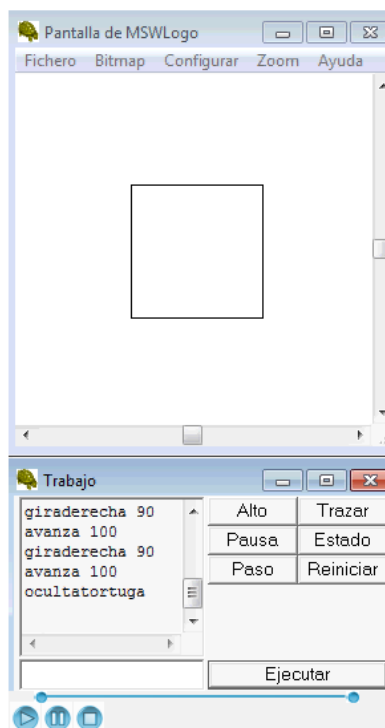
Contenidos

2. Primitivas de programación

Primitivas básicas

Código fuente de un programa para dibujar un cuadrado de lado 100

Comando largo	Abreviado	Explicación
MUESTRATORTUGA	MT	Hacemos que la tortuga aparezca en la pantalla si estaba oculta
SUBELAPIZ	SL	Hace que las siguientes ordenes no pinten
AVANZA 50	AV 50	Hacemos que la tortuga avance 50
GIRAIZQUIERDA 90	GI 90	La tortuga gira 90 grados hacia la izquierda
AVANZA 50	AV 50	Hacemos que la tortuga avance 50
GIRADERECHA 180	GD 180	La tortuga gira 180 grados hacia la derecha (da media vuelta)
BAJALAPIZ	BL	Hace que las siguientes ordenes pinten
AVANZA 100	AV 100	Hacemos que la tortuga avance 100
GIRADERECHA 90	GD 90	La tortuga gira 90 grados hacia la derecha
AVANZA 100	AV 100	Hacemos que la tortuga avance 100
GIRADERECHA 90	GD 90	La tortuga gira 90 grados hacia la derecha
AVANZA 100	AV 100	Hacemos que la tortuga avance 100
GIRADERECHA 90	GD 90	La tortuga gira 90 grados hacia la derecha
AVANZA 100	AV 100	Hacemos que la tortuga avance 100
OCULTATORTUGA	OT	Hace que la tortuga desaparezca de la pantalla





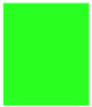



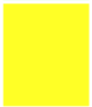








Contenidos

2. Primitivas de programación

Usando colores

- **PONCOLORLAPIZ (PONCL):**
Se usa para seleccionar el color de los trazos que irá produciendo el cursor al avanzar.
- **PONCOLORPAPEL (PONCP):**
Se usa para seleccionar el color de la pantalla.
- **PONCOLORRELLENO (POCCR):**
Se usa para seleccionar el color de relleno de las figuras que dibujemos. Para rellenar una figura que hayamos dibujado, tras definir el color de relleno con POCCR se emplea la primitiva RELLENA.

Estas tres primitivas necesitan un parámetro que indique el color, que es un número según la siguiente tabla:

0		1		2		3	
Negro		Azul oscuro		Verde claro		Azul claro	
4		5		6		7	
Rojo		Violeta		Amarillo		Blanco	
8		9		10		11	
Marrón		Dorado		Verde oscuro		Turquesa	
12		13		14		15	
Rosa claro		Morado		Naranja		Gris	

NOTA: esta tabla no tienes que sabértela; de ser necesaria en el examen, se te proporcionaría

Contenidos

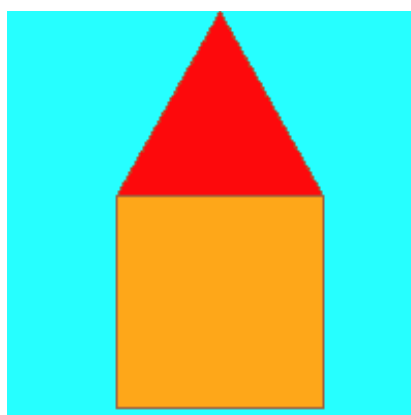
2. Primitivas de programación

Usando colores

Código fuente de ejemplo de uso de colores

Comando largo	Abreviado	Explicación
PONCOLORPAPEL 3	PONCP 3	Establece el color del papel como color número 3: azul claro
PONCOLORLAPIZ 8	PONCL 8	Establece el color del lápiz como color número 8: marrón
RETROCEDE 100	RE 100	Hacemos que la tortuga retroceda 100
GIRADERECHA 90	GD 90	La tortuga gira 90 grados hacia la derecha
AVANZA 50	AV 50	Hacemos que la tortuga avance 50
GIRADERECHA 90	GD 90	La tortuga gira 90 grados hacia la derecha
AVANZA 100	AV 100	Hacemos que la tortuga avance 100
GIRADERECHA 90	GD 90	La tortuga gira 90 grados hacia la derecha
AVANZA 100	AV 100	Hacemos que la tortuga avance 100
GIRADERECHA 90	GD 90	La tortuga gira 90 grados hacia la derecha
AVANZA 100	AV 100	Hacemos que la tortuga avance 100
GIRADERECHA 90	GD 90	La tortuga gira 90 grados hacia la derecha
AVANZA 100	AV 100	Hacemos que la tortuga avance 100
GIRADERECHA 90	GD 90	La tortuga gira 90 grados hacia la derecha
AVANZA 100	AV 100	Hacemos que la tortuga avance 100
GIRAIZQUIERDA 120	GI 120	La tortuga gira 120 grados hacia la izquierda
AVANZA 100	AV 100	Hacemos que la tortuga avance 100
GIRAIZQUIERDA 60	GI 60	La tortuga gira 60 grados hacia la izquierda (queda horizontal)

GIRAIZQUIERDA 60	GI 60	La tortuga gira 60 grados hacia la izquierda
AVANZA 100	AV 100	Hacemos que la tortuga avance 100
SUBELAPIZ	SL	Hace que las siguientes ordenes no pinten
CENTRO		Sitúa la tortuga en el centro de la pantalla
RETROCEDE 50	RE 50	Hacemos que la tortuga retroceda 50
PONCOLORRELLENO 14	PONCCR 14	Establece el color del relleno como color número 14: naranja
RELLENA		Rellena el interior de la figura con el color designado en PONCOLORRELLENO
AVANZA 75	AV 75	Hacemos que la tortuga avance 75
PONCOLORRELLENO 4	PONCCR 14	Establece el color del relleno como color número 4: rojo
RELLENA		Rellena el interior de la figura con el color designado en PONCOLORRELLENO
OCULTATORTUGA	OT	Hace que la tortuga desaparezca de la pantalla



Contenidos

2. Primitivas de programación

Creando secuencias repetitivas

Un programa mínimamente complejo lleva consigo la ejecución de un gran número de primitivas. El primer paso para ahorrarnos código es emplear la primitiva **REPITE**, que va acompañada de un parámetro que indica cuál es ese número de veces que se debe repetir, y de otra primitiva, que será la instrucción que tiene que repetirse un número de veces.

Uno de los ejemplos más típicos que explica la comodidad del uso de la primitiva **REPITE** es la construcción de un cuadrado. Para ello es necesario hacer avanzar el cursor la longitud del lado y girar su dirección 90°; habrá que repetir esta instrucción cuatro veces.

Utilizando **REPITE** podemos reducir cuatro líneas de código a una sola; las primitivas que deben repetirse se colocan entre corchetes:

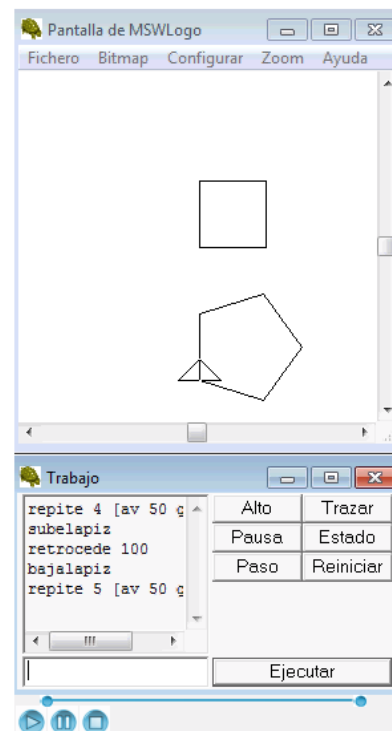
REPITE 4 [AV 50 GD 90]

Con esta orden el ordenador entiende que queremos repetir 4 veces la siguiente secuencia: avanza 50 y gira derecha 90 grados. El resultado final es el dibujo de un cuadrado.

Veamos este ejemplo con más detalle así como otro ejemplo de dibujo de un pentágono.

Código fuente de ejemplo de uso de REPITE

Comando largo	Abreviado	Explicación
REPITE 4 [AV 50 GD 90]		Repite 4 veces las siguientes órdenes: avanza 50, gira a la derecha 90 grados. Obtenemos un cuadrado
SUBELAPIZ	SL	Levantamos el lápiz para no dejar trazo
RETROCEDE 100	RE 100	Hacemos que la tortuga retroceda 100
BAJALAPIZ	BL	Bajamos el lápiz para escribir
REPITE 5 [AV 50 GD 72]		Repite 4 veces las siguientes órdenes: avanza 50, gira a la derecha 72 grados. Obtenemos un pentágono.



Contenidos

2. Primitivas de programación

Mostrar texto: ROTULA

ROTULA es una de las primitivas más interesantes porque nos permite escribir en la pantalla. Tenemos que tener cuidado porque el texto será escrito en la dirección del cursor (si el cursor está vertical el texto se escribirá en vertical).

Para introducir el texto podemos hacerlo con comillas al principio (no al final) si se trata de una palabra sola o entre corchetes si se trata de varias palabras.

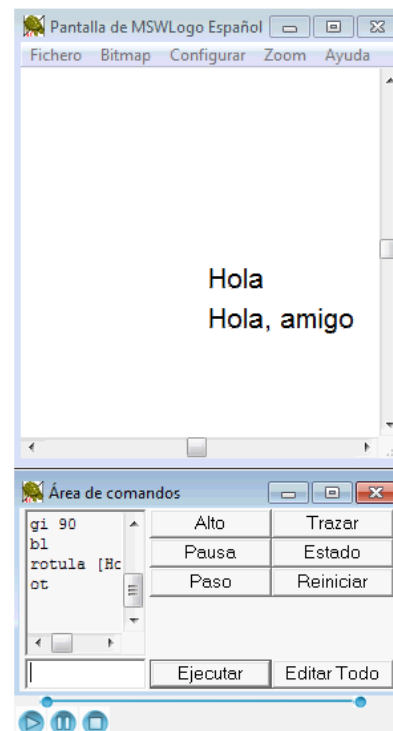
ROTULA "Hola"

ROTULA [Hola, amigo]

Para escribir varias líneas, tendremos que desplazar el cursor después de cada una o de lo contrario los textos se superpondrán. Antes de desplazar el cursor, habrá que subir el lápiz para no ir dejando huella y volver a bajarlo antes de escribir de nuevo.

Código fuente de ejemplo: ROTULA

Comando largo	Abreviado	Explicación
GIRADERECHA 90	GD 90	para que escriba el texto en horizontal
ROTULA "Hola"		Escribe el texto -Hola-
SUBELAPIZ	SL	Levantamos el lápiz para no dejar trazo
GIRADERECHA 90	GD 90	Giramos para avanzar hacia abajo
AVANZA 30	AV 30	Hacemos que la tortuga avance 30
GIRAIZQUIERDA 90	GI 90	La tortuga gira 90 grados hacia la izquierda para poder escribir otra línea
BAJALAPIZ	BL	Bajamos el lápiz para escribir
ROTULA [Hola, amigo]		Escribe el texto -Hola, amigo-
OCULTATORTUGA	OT	Hace que la tortuga desaparezca de la pantalla



Contenidos

2. Primitivas de programación

Crear ventanas y botones

En los programas suele ser muy útil disponer de botones que nos permitan controlar el cursor (que recordemos que lo que haga el cursor será lo que haga el sistema a controlar o el sistema robótico) y darle instrucciones desde la pantalla sin tener que entrar a modificar el código.

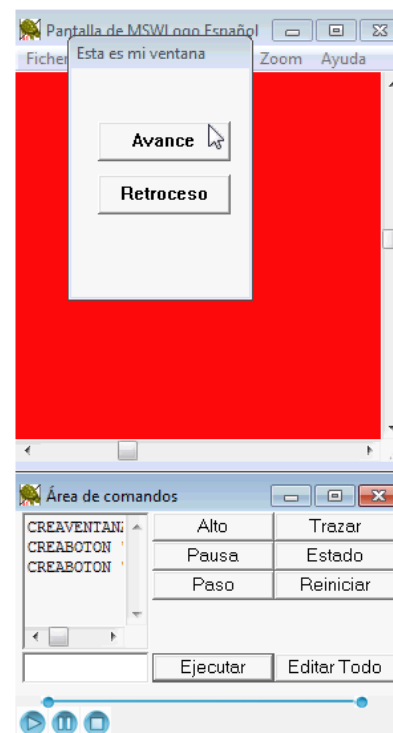
Existen varias Crear ventanas y botones, las más importantes son **CREAVENTANA** y **CREABOTÓN**, que nos permiten mostrar una ventana con botones dentro de nuestro programa. (Pulsa en ellas para obtener más información)

Si nos hemos equivocado al crear el botón o toda la ventana, los borramos con las primitivas **BORRABOTON** o **BORRAVENTANA**, seguidas del nombre del botón o la ventana precedidos de una comilla.

BORRABOTON "AVANCE BORRAVENTANA "MIVENTANA]

Código fuente de ejemplo: Pantalla con botones

Comando largo	Explicación
<code>creaventana "MIVENTANA [Esta es mi ventana] 120 60 70 100 [poncp 4]</code>	Crea una ventana con el texto "Esta es mi ventana", se sitúa a 120 unidades a la derecha y 60 debajo y tendrá 70 unidades de largo por 100 de ancho. Al crear la ventana la pantalla se volverá de color rojo.
<code>creaboton "MIVENTANA "AVANCE [Avance] 10 20 50 15 [av 50]</code>	Crea un botón situado a 10 unidades a la derecha y 20 debajo de la ventana MIVENTANA y tendrá 50 unidades de largo por 15 de ancho. Al pulsar el botón la toruga avanzará 50
<code>creaboton "MIVENTANA "RETROCESO [Retroceso] 10 40 50 15 [re 50]</code>	Crea un botón situado a 10 unidades a la derecha y 40 debajo de la ventana MIVENTANA y tendrá 50 unidades de largo por 15 de ancho. Al pulsar el botón la toruga retrocederá 50



Contenidos

2. Primitivas de programación

Procedimientos

A veces existe una **combinación de primitivas** que tenemos que usar con cierta frecuencia. Las primitivas son texto y por lo tanto se pueden copiar y pegar de un sitio a otro, pero eso supone añadir muchas líneas de código; intentar eliminar código superfluo es muy importante en un programa de cierta complejidad.

Para ello se crean los **procedimientos**; se asigna un nombre a ese conjunto de primitivas de forma que se ejecutan automáticamente cada vez que escribimos el nombre.

Como ejemplo, veamos un procedimiento que nos permite dibujar cuadrados: Ya habíamos reducido con anterioridad el código necesario para dibujar un cuadrado; gracias a la primitiva **REPITE**, habíamos convertido las cuatro primitivas necesarias para dibujar los cuatro lados en una sola:

REPITE 4 [AV 50 GD 90]

No obstante, si tenemos que dibujar un montón de cuadrados repetir constantemente esta primitiva complica el código del programa; creamos por lo tanto un procedimiento llamado cuadrado:

PARA cuadrado REPITE 4 [AV 50 GD 90] FIN

La primera línea de un procedimiento consta siempre de **PARA** seguido del nombre (no se admiten espacios ni tildes ni caracteres distintos de letras y números en el nombre). La última línea dirá únicamente **FIN** y las líneas intermedias, que pueden ser más de una, incluyen las primitivas que se van a ejecutar mediante el procedimiento.

Contenidos

2. Primitivas de programación

Procedimientos

La primera consiste en utilizar el editor del programa; para ello haremos clic en la esquina inferior derecha donde pone Editar todo. Se nos abrirá una ventana en la que podemos escribir el procedimiento:

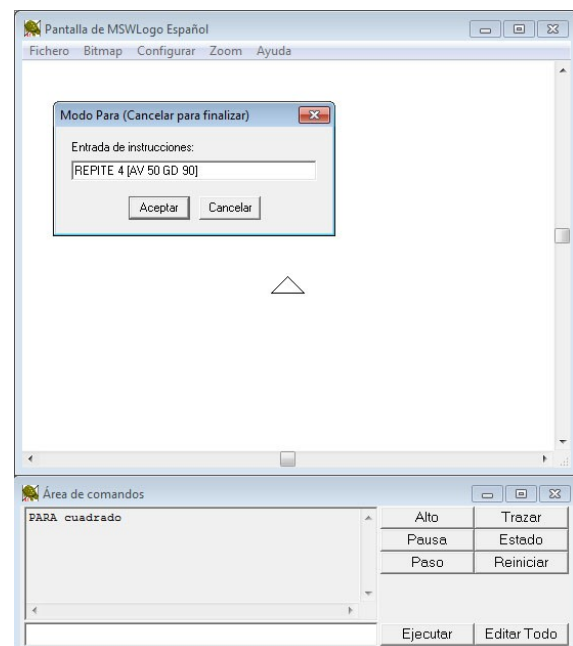
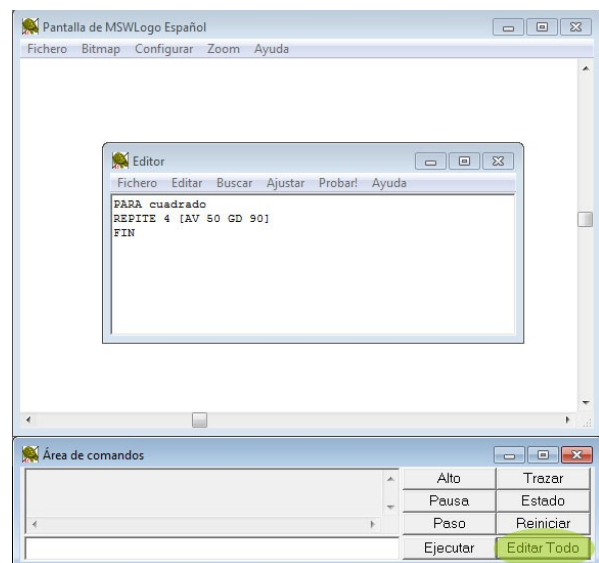
Tras escribir el procedimiento, seleccionamos dentro del menú Fichero la opción guardar y salir. El procedimiento quedará almacenado en la memoria y se accederá a él como si se tratase de una primitiva.

La segunda forma de escribir un procedimiento consiste en escribir la primera línea (PARA + nombre del procedimiento) y pulsar intro. Nos surgirá una ventana para que añadamos las primitivas con las instrucciones; después de cada primitiva pulsamos intro.

Cuando hayamos introducido todas las líneas intermedias del procedimiento (en el caso del cuadrado sólo es una) hacemos clic en Cancelar, para que el programa añada la línea FIN.

Cuando tenemos definido el procedimiento, basta con escribir su nombre (cuadrado) para que se ejecute, igual que si se tratara de una primitiva.

Para modificar el procedimiento, hacemos clic en Editar Todo y nos surgirá una ventana con el código de todos los procedimientos que hayamos definido. Podemos modificarlos y guardar cambios.



Contenidos

2. Primitivas de programación

Control de tiempo: ESPERA

Con **ESPERA** podemos generar un intervalo entre una primitiva y otra. Debe de ir seguido de un parámetro que indique el tiempo: cada segundo son 60 unidades de tiempo. De esta forma podemos definir a nuestro gusto el tiempo de ejecución de las distintas órdenes de un procedimiento.

Por ejemplo, vamos a ir introduciendo un texto por partes:

(NOTA: los comentarios dentro del código tras el ; no serán procesados por el ordenador)

```
PARA saludo  
GD 90  
ROTULA "Hola  
ESPERA 30 ;medio segundo  
BP ;de lo contrario, los textos se superpondrán  
GD90  
ROTULA [¿Qué tal estás?]  
ESPERA 30  
BP  
FIN
```

Si lo que queremos es que el programa se repita de forma indefinida, la última instrucción del programa debe ser el propio nombre del programa.

De esa forma el programa se llamará a sí mismo y volverá a empezar antes de acabar.

Contenidos

3. Variables de programación

Concepto

Las **variables** son **datos**, numéricos o de texto, que pueden tomar diferentes valores. A las variables se les asigna un **valor** mediante la primitiva **HAZ**. Para que el programa las identifique como variables, su nombre debe ir precedido de comillas.

```
HAZ "numero 80
```

Una vez que le hemos asignado un valor, para utilizarla escribiremos su nombre precedido de dos puntos:

```
AV :numero
```

Por lo tanto, se emplean las comillas para definir la variable y los dos puntos para leerla.

Si la variable es de texto, se define su valor introduciéndolo entre corchetes:

```
HAZ "nombre [Pablo]
```

De esta forma nos escribirá en la pantalla el texto de la variable, como aparecía en la imagen anterior:

```
ROTULA :nombre
```

Vamos a ver un ejemplo:

```
HAZ "numero 80  
GD 90  
AV :numero  
HAZ "tunombre [Pablo]  
ROTULA :tunombre
```

Explicación del código del ejemplo:

Almacenamos el valor 80 en la variable numero. Accedemos a este valor usando los dos puntos.

A continuación, almacenamos el valor de texto "Pablo" en la variable tunombre. Accedemos a este valor usando los dos puntos.

Contenidos

3. Variables de programación

Usando variables en texto

A veces nos interesa introducir variables en el medio de un texto. Para ello disponemos de la primitiva **FRASE**.

Tenemos que poner el texto entre corchetes y toda la frase, el texto y las variables, entre paréntesis para que el programa lo considere todo como una unidad.

Veámoslo con un ejemplo:

```
GD 90  
HAZ "nombre [Pablo]  
ROTULA (FRASE [Me llamo ] :nombre)
```

Aparecerá en la pantalla "Me llamo Pablo"

Procedimientos en variables

Los procedimientos permiten incluir variables en su nombre; de esa forma, al llamar al procedimiento le estaremos dando también el valor que le tiene que aplicar a la variable.

Veámoslo retomando el ejemplo del cuadrado: nos resulta muy útil convertir el lado del cuadrado en una variable y dar su valor en el mismo momento de llamar al procedimiento:

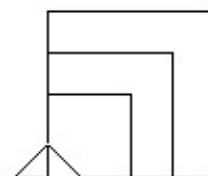
```
PARA cuadrado :lado  
REPITE 4 [AV :lado GD90]  
FIN
```

Definido así el procedimiento, al llamarlo podremos dar ya el valor del lado y dibujar muy fácilmente un gran número de cuadrados del lado que deseemos:

cuadrado 40

cuadrado 60

cuadrado 80



Contenidos

3. Variables de programación

Variables de datos en ventanas

Una utilidad muy habitual de las variables es la de permitir que el usuario introduzca datos. Para ello tenemos que conocer, en primer lugar, la primitiva PREGUNTABOX, que sirve para abrir una ventana en la que el programa nos pida información:

PREGUNTABOX [Nombre de la caja] [Pregunta]

Los parámetros de texto que acompañan a esta primitiva son el nombre que figurará en la ventana y el texto de la pregunta o la petición de información que la ventana va a llevar a cabo.

PREGUNTABOX se puede emplear como parámetro de **HAZ** de forma que quede grabada en forma de variable la respuesta que introducimos en la ventana.

El contenido de **PREGUNTABOX** se asimila como valor tipo texto; si se tratara de almacenar un número tenemos que usar la siguiente sintaxis:

PREGUNTABOX PRIMERO [Nombre de la caja] [Pregunta]

Ejemplo de variables de datos en ventanas:

```
GD90
HAZ "nombre PREGUNTABOX [Dime tu nombre] [¿Cómo te llamas?]
ROTULA (FRASE [¡Hola, ] :nombre [!])
SL
GD 90
AV 100
GI 90
HAZ "numero PRIMERO PREGUNTABOX [Dime tu edad] [¿Cuántos años
tienes?]
ROTULA (FRASE [Tienes ] :numero [años])
OT
```

Explicación: Introducimos en una pantalla llamada "Dime tu nombre" y con el texto "¿Cómo te llamas?" un valor y lo almacenamos en la variable de texto nombre.

Lo usamos en una frase y lo escribimos en pantalla como "¡Hola nombre!"

Nos colocamos debajo de este texto

Introducimos en una pantalla llamada "Dime tu edad" y con el texto "¿Cuántos años tienes?" un valor y lo almacenamos en la variable numérica (por eso usamos PRIMERO) numero.

Lo usamos en una frase y lo escribimos en pantalla como "Tienes numero años"

Ocultamos la tortuga.

Contenidos

3. Variables de programación

Variables de azar

La primitiva **AZAR** elige al azar un número entre el 0 y el inferior al que le pongamos como parámetro; por ejemplo, **AZAR 11** elegirá un número aleatorio entre el 0 y el 10.

Veamos un ejemplo: Dibujo de un cuadrado de lado aleatorio entre 0 y 100.

```
PARA cuadrado :lado  
REPITE 4 [AV :lado GD 90]  
fin
```

```
HAZ "numero AZAR 101
```

```
cuadrado: numero
```

En la primera parte del programa creamos un procedimiento llamado cuadrado que necesita de una variable que definirá el lado del cuadrado a dibujar.

En la orden **HAZ "lado AZAR 101**, el programa elige al azar un valor entre 0 y 100 y lo almacenará en la variable numero. Después dibujamos un cuadrado con la variable numero como lado.

Ahora bien, ¿qué pasaría si volviéramos a dar la última orden? El programa volvería a dibujar un cuadrado del mismo tamaño, ya que no hemos modificado el valor numero. Podríamos modificarlo este valor repitiendo la orden **HAZ "numero AZAR 101** almacenándose otro valor al azar en esa variable número. Si dibujáramos entonces otro cuadrado :numero seguramente tendrá otro valor diferente.

Contenidos

4. Control del programa

Evaluando condiciones

A veces nos interesa que una primitiva o una parte del programa se ejecute sólomente si se cumple una condición. Para ello existe la primitiva **SI**; la condición debe escribirse entre corchetes. Veamos un ejemplo:

```
SI :lado > 40 [ROTULA [Lado muy grande]
```

Si el valor de la variable lado es mayor que 40 entonces escribe en la pantalla "Lado muy grande". si fuera menor o igual no haría nada.

La primitiva **SISINO** perfecciona a la SI al permitimos dar dos instrucciones: una si la condición se cumple y otra si no se cumple. Las dos condiciones irán entre corchetes. Veamos un ejemplo sencillo:

```
SISINO :lado > 40 [ROTULA [Lado muy grande]] [REPITE 4 [AV :lado GD 90]]
```

Si el valor de la variable lado es mayor que 40 entonces escribe en la pantalla "Lado muy grande". Si fuera menor o igual realiza la orden que hay dentro del último corchete, dibuja un cuadrado cuyo lado está definido por la variable lado.

Ejemplo: Vamos a programar un juego por el que tengamos que adivinar un número al azar del 1 al 5. Para evitar el 0, utilizaremos la primitiva AZAR 5, que obtendrá un número entre 0 y 4, y le sumaremos 1:

```
PARA juego  
BP GD 90  
HAZ "numero 1 + AZAR 5  
HAZ "intento PRIMERO PREGUNTABOX [Adivina el número] [Escribe un  
número del 1 al 5:]  
SISINO :intento = :numero [ROTULA [¡Has acertado!]] [ROTULA [Has fallado]]  
FIN
```

Contenidos

4. Control del programa

Evaluando condiciones

Vamos a hacer una mejora en el juego; que nos dé una segunda oportunidad y una pista diciéndonos si el número buscado es mayor o menor que el que hemos puesto:

```
PARA juego  
BP GD 90  
HAZ "numero 1 + AZAR 5  
HAZ "intento PRIMERO PREGUNTABOX [Adivina el número] [Escribe un  
número del 1 al 5:]  
SI :intento > :numero [HAZ "intento2 PRIMERO PREGUNTABOX [Adivina el  
número] [iDemasiado grande! Escribe un nuevo número:]  
SI :intento < :numero [HAZ "intento2 PRIMERO PREGUNTABOX [Adivina el  
número] [iDemasiado pequeño! Escribe un nuevo número:]  
SISINO :intento2 = :numero [ROTULA [iHas acertado!]] [ROTULA [Has vuelto  
a fallar]]  
SI :intento = :numero [BP GD 90 ROTULA [iHas acertado!]];  
FIN
```

En la última línea le añadido BP y GD 90 para que, si he acertado a la primera, el programa borre el mensaje anterior y escriba recto (ya que, por no tener valor intento2, no coincidirá con el número y en la línea anterior escribirá que hemos vuelto a fallar)

Contenidos

4. Control del programa

Bucles: Órdenes reiterativas

Se trata de primitivas o procedimientos que se repiten un número indefinido de veces hasta que los detiene el usuario manualmente o hasta que se cumple alguna condición programada para que haga detenerse el procedimiento.

Para hacer una instrucción o un procedimiento reiterativo existen dos maneras: la primera es emplear la primitiva **SIEMPRE**, en la que la instrucción o instrucciones que se pongan a continuación entre paréntesis se repetirán una y otra vez de forma indefinida.

Por ejemplo, mediante esta instrucción tendremos un cuadrado intermitente; el programa dibujará un cuadrado, esperará, lo borrará, lo volverá a dibujar y así sucesivamente:

```
SIEMPRE [REPITE 4 [AV 40 GD 90] ESPERA 20 BP ESPERA 20]
```

Para detener la instrucción haremos clic en el botón Alto del área de comandos:

La otra forma es crear un procedimiento que se llame a sí mismo al finalizar (en la línea anterior a **FIN**); de esta forma se repetirá siempre:

```
PARA Cuadradointermitente  
REPITE 4 [AV 40 GD 90] ESPERA 20 BP ESPERA 20  
Cuadradointermitente  
FIN
```

Contenidos

4. Control del programa

Bucles: Órdenes reiterativas

A veces no nos interesa que la orden se repita siempre de forma igual, sino ir la modificando cada vez que se ejecute; por ejemplo, dibujar cuadrados de lado cada vez más grande. Para ello podemos crear una variable y programarla para que sufra alguna modificación en las sucesivas reiteraciones del procedimiento.

Por ejemplo, veamos como hacer cuadrados cuyo lado va aumentando en 10 unidades:

```
HAZ "lado :lado + 10
```

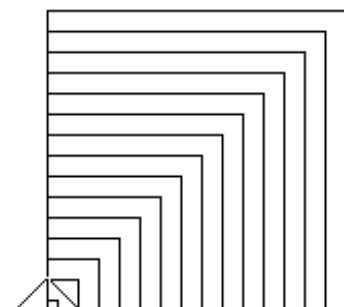
Es decir, súmale 10 al valor actual que tenga el lado.

```
PARA Cuadradoscrecientes :lado  
REPITE 4 [AV :lado GD 90] ESPERA 50  
HAZ "lado :lado + 10  
Cuadradoscrecientes :lado  
FIN
```

Al llamar al programa habrá que asignar un valor al lado del primer cuadrado, por ejemplo 5 unidades.

Para evitar que siga haciendo cuadrados hasta el infinito, podemos añadir la condición de que en el momento en que el lado supere un cierto valor el programa llame a la primitiva ALTO, es decir, se detenga. Naturalmente habrá que poner esta condición antes de dar la orden de repetir el programa.

```
PARA Cuadradoscrecientes :lado  
REPITE 4 [AV :lado GD 90] ESPERA 50  
HAZ "lado :lado + 10;  
SI :lado >150 [ALTO]  
Cuadradoscrecientes :lado  
FIN
```



Contenidos

4. Control del programa

Primitivas de control de la tarjeta

MSWLogo sirve como base para aprender la estructura de un lenguaje de programación; en tecnología tiene, no obstante, otro uso más específico del que ya hemos hablado, que es el de dar órdenes a un robot a través de una **tarjeta controladora**.

La tarjeta se conecta a un puerto del ordenador y en su driver de instalación debe tener un archivo que se copia y pega en la carpeta del disco duro donde está instalado el programa de control (en este caso MSWlogo).

A continuación se abre el programa y desde él se carga este archivo, que abre el puerto de comunicación entre el ordenador y la controladora y habilita las primitivas que permiten controlar las entradas y salidas de la tarjeta.

Algunas de estas primitivas son:

CONECTA 3; conecta la salida número 3 en este caso.

APAGA 3; desconecta la salida número 3.

ENTRADA 3; comprueba si la entrada número 3 está activada o no y devuelve una respuesta lógica, verdadero o falso.

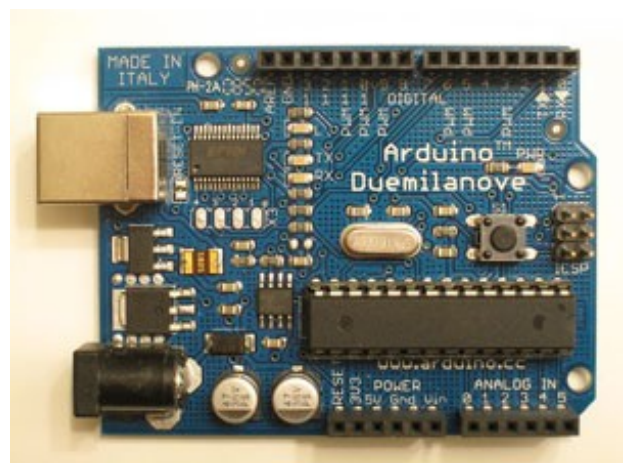
VOLTAJE 3 3; fija en la salida analógica 3 una tensión de 3 voltios.

NOTA: Puesto que al estudiar a distancia no disponéis de una tarjeta controladora, no tenéis por qué conocerlas pero el objetivo es que tengáis una idea del tipo de ordenes mediante las cuales se puede controlar la tarjeta de forma elemental desde el ordenador.

Controladora Arduino:

En la parte superior de la tarjeta podemos distinguir unas entradas / salidas digitales, en la inferior unas entradas analógicas y otras para alimentación de energía. Cada una de estas entradas y salidas tienen un número para identificarlas.

A la izquierda, de perfil, hay un elemento metálico que se trata de un puerto USB para conectarla al ordenador.





Recuerda lo más importante

El **lenguaje de programación** o código es la forma que tenemos de comunicarnos con un ordenador.

Existen cientos de **lenguajes de programación** y son muy diversos. **LOGO** es un lenguaje sencillo para educación.

Las **tarjetas de control** o controladoras sirven de enlace entre el ordenador y el sistema a controlar o un robot. constan de una serie de entradas y salidas que pueden ser **digitales y analógicas**.

En un lenguaje de programación consta de órdenes básicas denominadas **primitivas** cuya ejecución ordenada se conoce como **programa**.

PRIMITIVAS BÁSICAS

AV y RE: para desplazar el cursor	PONCL: color de los trazos	REPITE: crea secuencias repetitivas
GD y GI: cambiar la orientación	PONCP: color de la pantalla	ROTULA: para escribir en pantalla
SL y BL: subir y bajar el lápiz	POCCR: color del relleno	CREAVENTANA: crear una ventana

El código de un programa se denomina **código fuente**. En un programa podemos llamar a otros ya escritos, llamados **procedimientos**.

Un procedimiento siempre comienza con **para** seguido del nombre del procedimiento y termina con **fin**.

Para que se ejecute basta con escribir su nombre.

Los **datos** pueden ser almacenados en la memoria en las llamadas **variables** para su uso posterior.

A las variables se les asigna un valor mediante la primitiva HAZ, con su nombre precedido de comillas: haz "numero 80.

La primitiva **PREGUNTABOX** permite que el usuario introduzca datos de las variables.

Otra variable usada en programación es el resultado de usar la primitiva **AZAR**.

Para **controlar** la ejecución del programa se dispone de las primitivas condicionales como **SISINO**, que dependiendo del valor de una variable el programa ejecuta una u otra acción.

Además se pueden realizar **bucles**, que se repiten indefinidamente hasta que el usuario lo detenga.



Para practicar

EJERCICIO 1

Crea una ventana con cuatro botones, uno que dibuje círculos de radio 40, otro que dibuje cuadrados de lado 50, un tercero que cambie el color de fondo y lo ponga azul y el último que limpie la ventana y la vuelva a poner blanca. La ventana debe llamarse ejercicio, estar situada a unas coordenadas (60,40) del origen y medir 100 de alto x 100 de ancho. Los botones deben tener unas dimensiones de 35 x 35.

EJERCICIO 2

Define el procedimiento ESCALERA de forma que al ejecutarlo dibuje una escalera de 5 peldaños, cada peldaño de 20 por 20. Recuerda que este código no se escribe en el área de comandos, sino en el editor.

EJERCICIO 3

Crea el procedimiento CABAÑA, dibujando una cabaña como la del dibujo, donde el rectángulo tenga 80 de ancho por 50 de largo y el triángulo sea equilátero. Utiliza REPITE para la construcción del rectángulo y del triángulo.



Usando CABAÑA, construye un pueblo de nueve cabañas situadas en tres filas de tres cabañas cada una. La separación entre dos cabañas (medida entre el mismo punto en una y en la otra) será de 130 unidades, tanto en vertical como en horizontal.

EJERCICIO 4

Define un procedimiento por el que un círculo de radio 40 cambie de color cada 40 unidades de tiempo. El programa debe repetirse indefinidamente.

EJERCICIO 5

Diseña un semáforo con un disco de radio 40 metido dentro de un cuadrado de lado 100 (el centro del disco debe ser el centro del cuadrado). El semáforo debe cambiar del rojo al amarillo y luego al verde según la velocidad que le indiquemos al llamar al procedimiento (lo que indicaremos será el número de unidades de tiempo que permanecerá en cada color).



Para practicar

EJERCICIO 6

Mejora el semáforo anterior mediante una ventana con dos botones, uno de acelerador y otro de freno. Al ejecutar el programa aparecerá la ventana con los dos botones y se llamará al programa del semáforo, que cambiará la primera vez cada 32 unidades de tiempo. Al pulsar el acelerador se volverá a ejecutar el programa semáforo pero cambiando el tiempo a la mitad de su valor anterior, mientras que al pulsar el freno el semáforo cambiará en el doble de tiempo

EJERCICIO 7

Diseña un programa al que le proporciones el nombre y el número atómico de un elemento y te escriba en pantalla la frase "El nombre del elemento es un elemento químico de número atómico número atómico"; antes de escribirla, debes desplazar el cursor 200 unidades a la izquierda para que la frase te quepa en el área de trabajo.

Prueba el programa haciendo que escriba la frase:

El carbono es un elemento químico de número atómico

EJERCICIO 8

Mejora el programa del ejercicio 7 para que te pregunte el nombre del elemento químico y el número atómico y que escriba la frase con esos datos. Haz la prueba del nuevo programa con el oxígeno (número atómico 8) y con el litio (número atómico 3).

EJERCICIO 9

Diseña un programa que te permita jugar a la lotería primitiva al escribirte en la pantalla seis números entre el 1 y el 49.

EJERCICIO 10

En el semáforo del ejercicio 6 si se pulsa demasiadas veces el botón de aceleración llegará un momento en que el tiempo de cambio será menor que 1 y el programa dará error. Mejora el programa de forma que si el tiempo de respuesta es menor que 1 de un mensaje de "Demasiado rápido. Pulse frenar para continuar".

EJERCICIO 11

Haz una última modificación en el semáforo del ejercicio anterior haciendo que el semáforo no cambie sólo una vez sino que al cambiar a verde vuelva a rojo de forma automática.



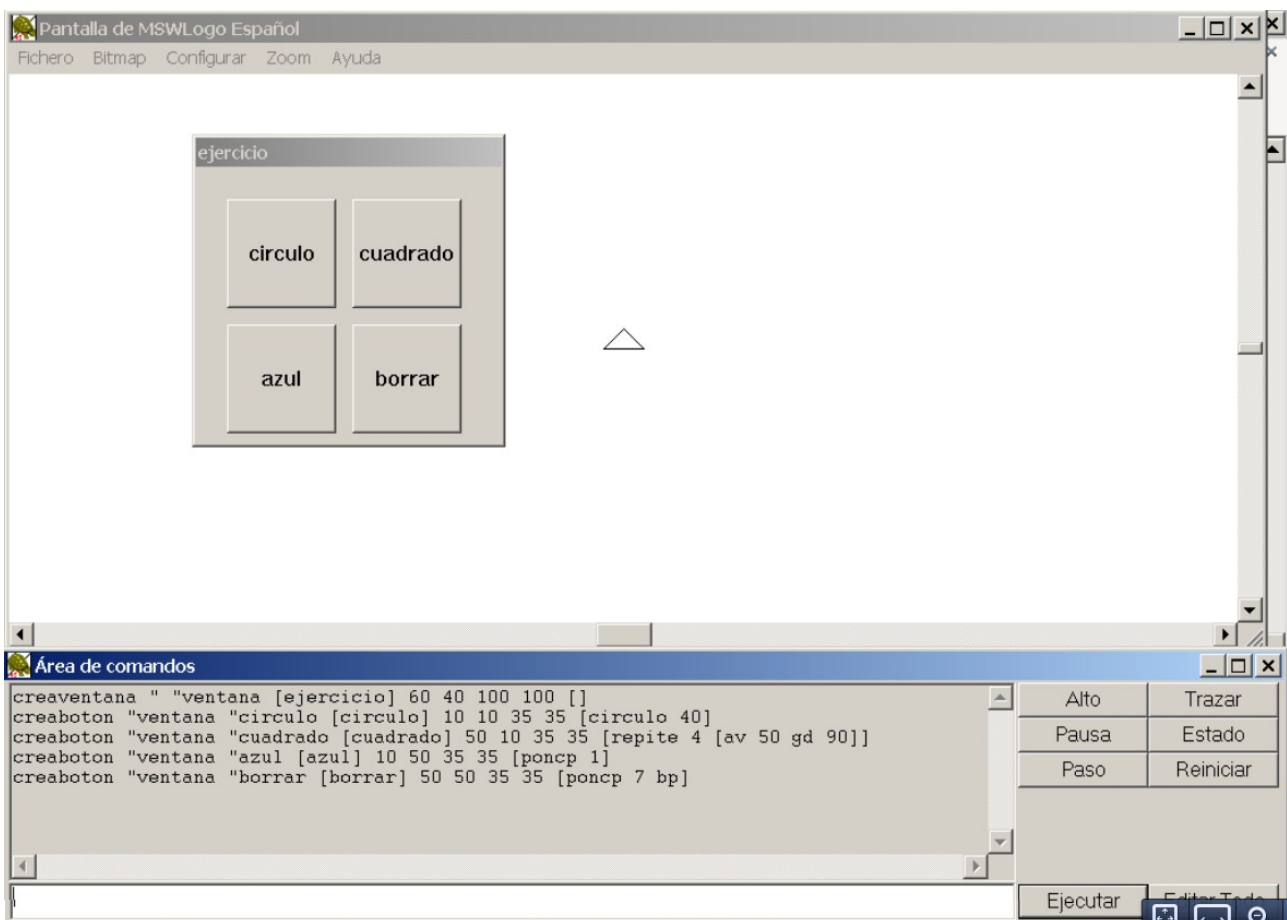
Para practicar

Ejercicio corregido

EJERCICIO 1:

Crea una ventana con cuatro botones, uno que dibuje círculos de radio 40, otro que dibuje cuadrados de lado 50, un tercero que cambie el color de fondo y lo ponga azul y el último que limpie la ventana y la vuelva a poner blanca. La ventana debe llamarse ejercicio, estar situada a unas coordenadas (60,40) del origen y medir 100 de alto x 100 de ancho. Los botones deben tener unas dimensiones de 35 x 35.

```
creaventana " "ventana [ejercicio] 60 40 100 100 []
creaboton "ventana "circulo [circulo] 10 10 35 35 [circulo 40]
creaboton "ventana "cuadrado [cuadrado] 50 10 35 35 [repite 4 [av 50 gd 90]]
creaboton "ventana "azul [azul] 10 50 35 35 [poncp 1]
creaboton "ventana "borrar [borrar] 50 50 35 35 [poncp 7 bp]
```





Para practicar

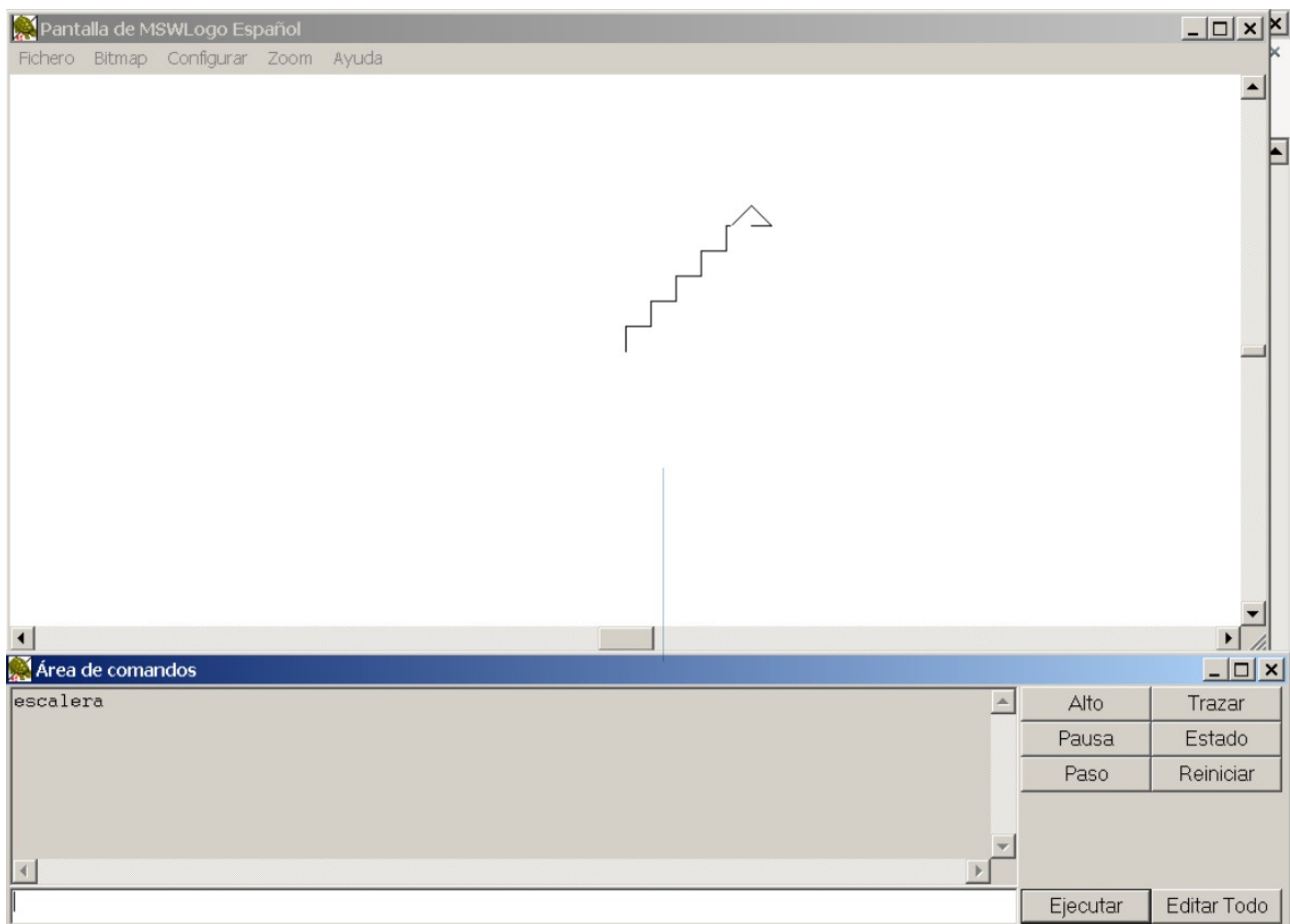
Ejercicio corregido

EJERCICIO 2

Define el procedimiento ESCALERA de forma que al ejecutarlo dibuje una escalera de 5 peldaños, cada peldaño de 20 por 20. Recuerda que este código no se escribe en el área de comandos, sino en el editor.

Recordemos que este código no se escribe en el área de comandos, sino en el editor:

```
para escalera
repite 5 [av 20 gd 90 av 20 gd 270]
fin
```



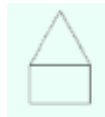


Para practicar

Ejercicio corregido

EJERCICIO 3

Crea el procedimiento CABAÑA, dibujando una cabaña como la del dibujo, donde el rectángulo tenga 80 de ancho por 50 de largo y el triángulo sea equilátero. Utiliza REPITE para la construcción del rectángulo y del triángulo.



Usando CABAÑA, construye un pueblo de nueve cabañas situadas en tres filas de tres cabañas cada una. La separación entre dos cabañas (medida entre el mismo punto en una y en la otra) será de 130 unidades, tanto en vertical como en horizontal.

Primero definimos el procedimiento a partir del código que escribimos en el ejercicio 5:

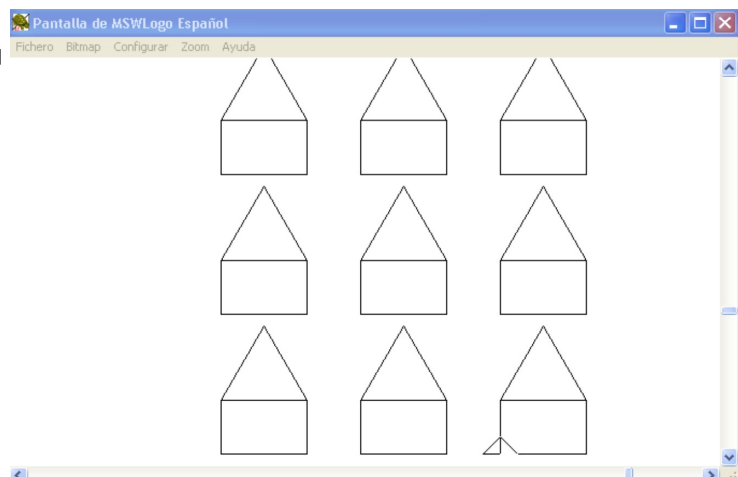
```

PARA CABAÑA
REPITE 2 [AV 50 GD 90 AV 80 GD 90] AV 50 GD 90 AV 80 GI 180 REPITE 3 [AV 80 GD 120] AV 80 GD 90 RE
50; an adimos esta línea para volver al origen.
FIN
  
```

Ahora llamamos al procedimiento y nos vamos desplazando para ir creando las diferentes cabañas:

```

CABAÑA
SL GD 90 AV 130 BL GI 90
CABAÑA
SL GD 90 AV 130 BL GI 90
CABAÑA SL GI 90 AV 260 GD 90; vuelvo al punto de origen
RE 130 BL
CABAÑA
REPITE 2 [SL GD 90 AV 130 BL GI 90 CABAÑA]
SL GI 90 AV 260 GD 90
RE 130 BL
CABAÑA
REPITE 2 [SL GD 90 AV 130 BL GI 90 CABAÑA]
  
```





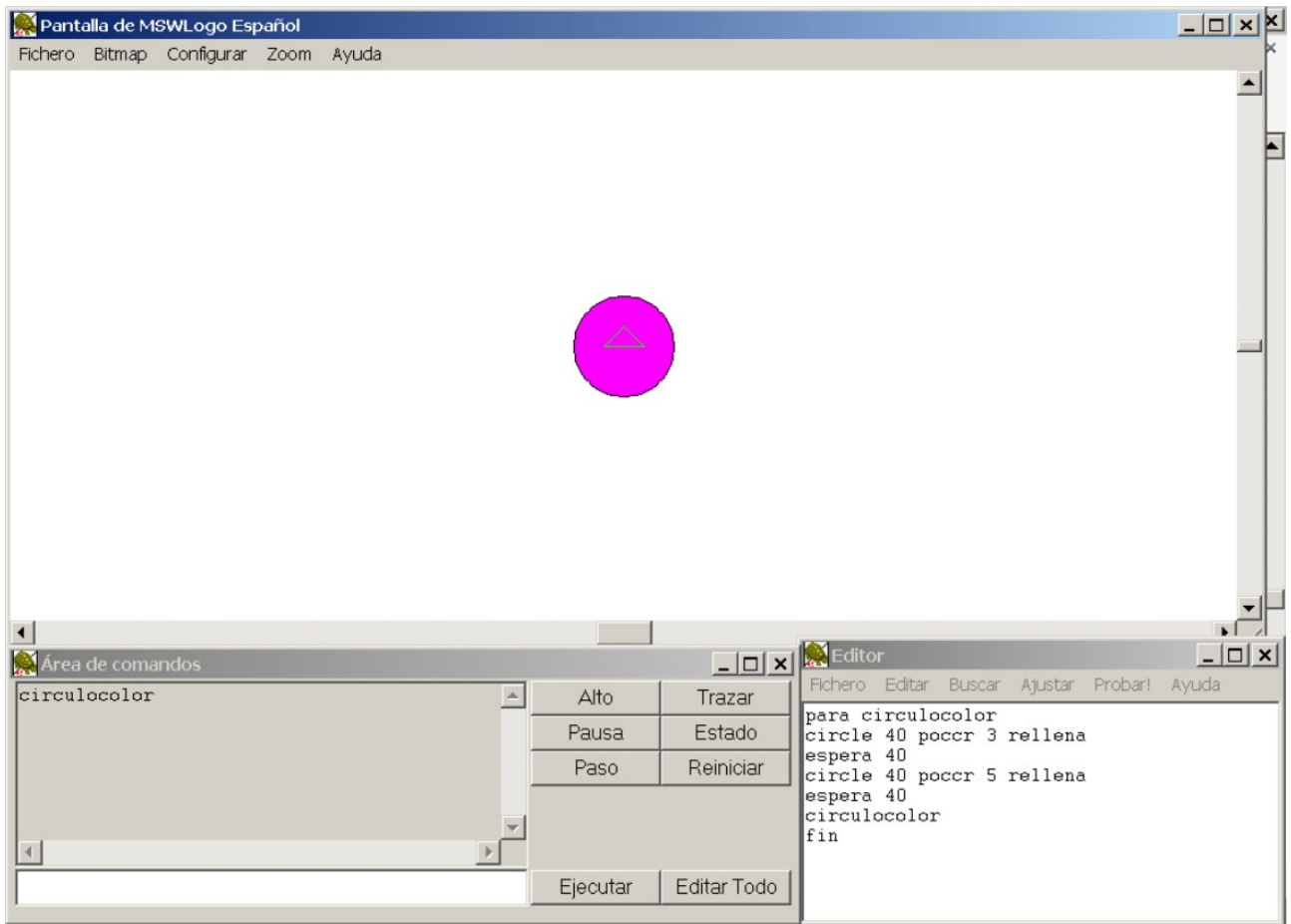
Para practicar

Ejercicio corregido

EJERCICIO 4

Define un procedimiento por el que un círculo de radio 40 cambie de color cada 40 unidades de tiempo. El programa debe repetirse indefinidamente.

```
para circulo color
circle 40 poccr 3 rellena
espera 40
circle 40 poccr 5 rellena
espera 40
circulo color ;(para que al acabarse vuelva a empezar indefinidamente)
fin
```



Para practicar

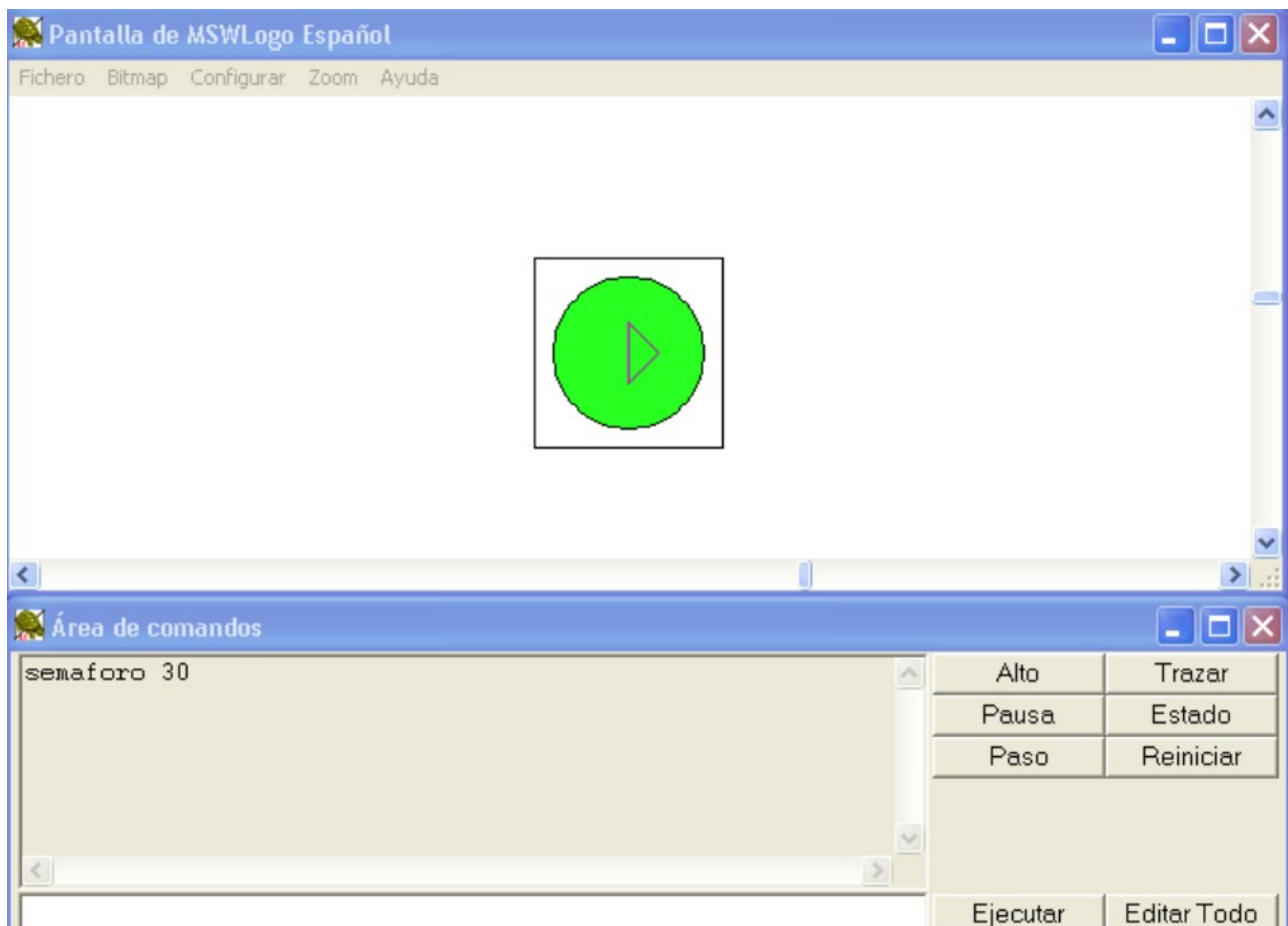
Ejercicio corregido

EJERCICIO 5

Diseña un semáforo con un disco de radio 40 metido dentro de un cuadrado de lado 100 (el centro del disco debe ser el centro del cuadrado). El semáforo debe cambiar del rojo al amarillo y luego al verde según la velocidad que le indiquemos al llamar al procedimiento (lo que indicaremos será el número de unidades de tiempo que permanecerá en cada color).

```

PARA semaforo :tiempo
BP BL
REPITE 4 [AV 100 GD 90]
SL AV 50 GD 90 AV 50 BL
CIRCLE 40
POCCR 4 RELLENA ESPERA :tiempo
POCCR 6 RELLENA ESPERA :tiempo
POCCR 2 RELLENA ESPERA :tiempo
FIN
  
```





Para practicar

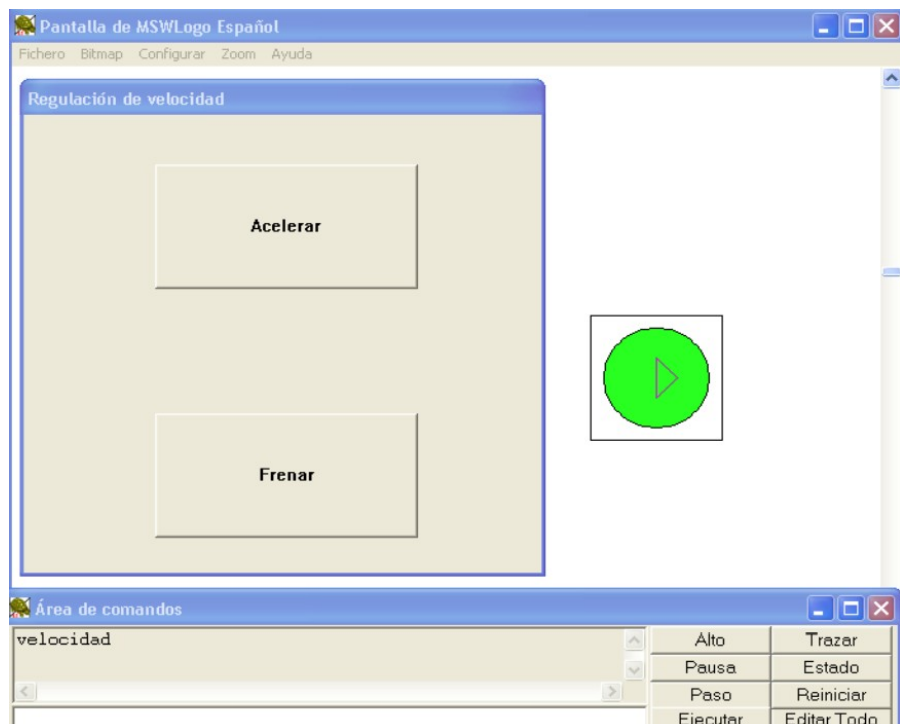
Ejercicio corregido

EJERCICIO 6

Mejora el semáforo anterior mediante una ventana con dos botones, uno de acelerador y otro de freno. Al ejecutar el programa aparecerá la ventana con los dos botones y se llamará al programa del semáforo, que cambiará la primera vez cada 32 unidades de tiempo. Al pulsar el acelerador se volverá a ejecutar el programa semáforo pero cambiando el tiempo a la mitad de su valor anterior, mientras que al pulsar el freno el semáforo cambiará en el doble de tiempo

```
PARA velocidad
BP BL
HAZ "tiempo 32
CREAVENTANA " "regulacion [Regulacion de velocidad] 50 50 200 200 []
CREABOTON "regulacion "acelerar [Acelerar] 50 20 100 50 [HAZ "tiempo :tiempo /2 semaforo]
CREABOTON "regulacion "frenar [Frenar] 50 120 100 50 [HAZ "tiempo :tiempo *2 semaforo]
semaforo
FIN
```

```
PARA semaforo
BP BL
REPITE 4 [AV 100 GD 90]
SL AV 50 GD 90 AV 50 BL
CIRCLE 40
POCCR 4 RELLENA ESPERA :tiempo
POCCR 6 RELLENA ESPERA :tiempo
POCCR 2 RELLENA ESPERA :tiempo
FIN
```





Para practicar

Ejercicio corregido

EJERCICIO 7

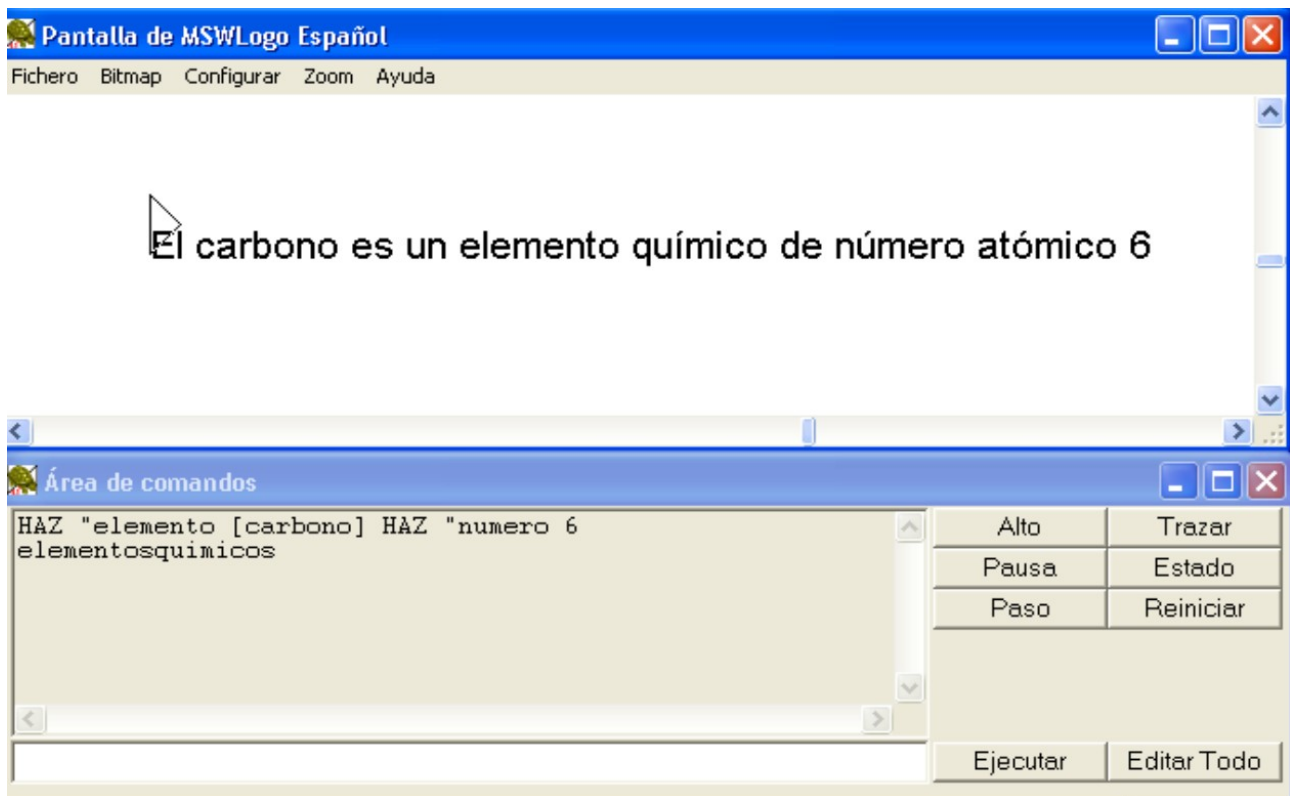
Diseña un programa al que le proporciones el nombre y el número atómico de un elemento y te escriba en pantalla la frase "El nombre del elemento es un elemento químico de número atómico número atómico"; antes de escribirla, debes desplazar el cursor 200 unidades a la izquierda para que la frase te quepa en el área de trabajo.

Prueba el programa haciendo que escriba la frase:

El carbono es un elemento químico de número atómico 6

```
para elementosquimicos
BP GI 90 SL AV 200 GI 180 BL
ROTULA (FRASE [El ] :elemento [ es un elemento qui mico de nu mero ato mico ] :numero)
FIN
```

```
HAZ "elemento [carbono] HAZ "numero 6
elementosquimicos
```



Para practicar

Ejercicio corregido

EJERCICIO 8

Mejora el programa del ejercicio 7 para que te pregunte el nombre del elemento químico y el número atómico y que escriba la frase con esos datos. Haz la prueba del nuevo programa con el oxígeno (número atómico 8) y con el litio (número atómico 3).

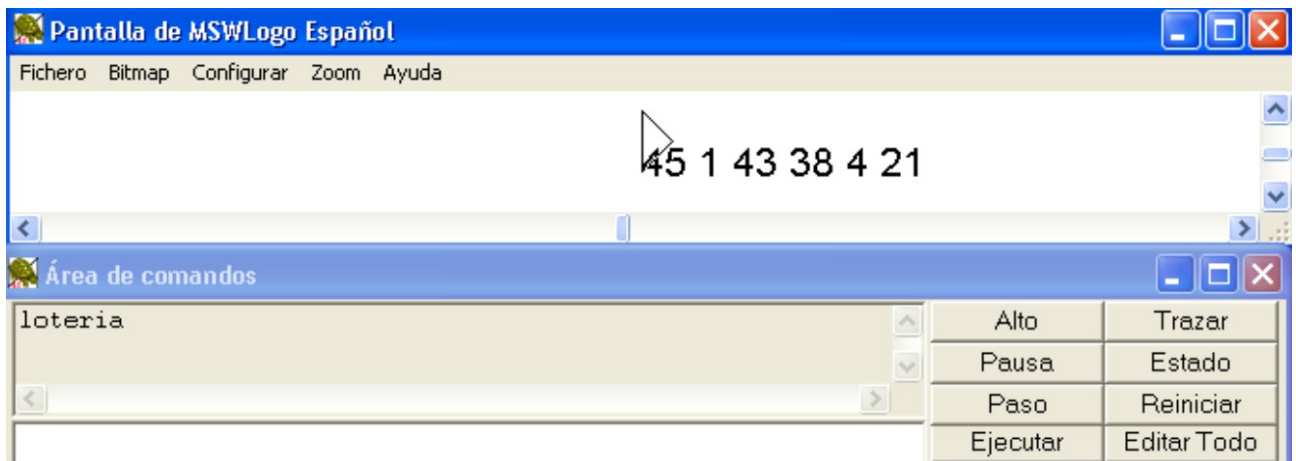
```
para elementosquimicos
BP GI 90 SL AV 200 GI 180 BL
HAZ "elemento PREGUNTABOX [Nombre del elemento] [Escribe el nombre del elemento qui mico]
HAZ "numero PREGUNTABOX [Número atómico] [Escribe el número atómico correspondiente]
ROTULA (FRASE [El ] :elemento [ es un elemento qui mico de número ] :numero)
FIN
```

Ejercicio corregido

EJERCICIO 9

Diseña un programa que te permita jugar a la lotería primitiva al escribirte en la pantalla seis números entre el 1 y el 49.

```
PARA loteria
HAZ "numero1 1 + AZAR 49
HAZ "numero2 1 + AZAR 49
HAZ "numero3 1 + AZAR 49
HAZ "numero4 1 + AZAR 49
HAZ "numero5 1 + AZAR 49
HAZ "numero6 1 + AZAR 49
BP BL GD 90
ROTULA (FRASE :numero1 [ ] :numero2 [ ] :numero3 [ ] :numero4 [ ] :numero5 [ ] :numero6)
FIN
```



Para practicar

Ejercicio corregido

EJERCICIO 10

En el semáforo del ejercicio 6 si se pulsa demasiadas veces el botón de aceleración llegará un momento en que el tiempo de cambio será menor que 1 y el programa dará error. Mejora el programa de forma que si el tiempo de respuesta es menor que 1 de un mensaje de "Demasiado rápido. Pulse frenar para continuar".

Co digo (so lo cambia el procedimiento semaforo, el de velocidad se queda igual):

```

PARA semaforo
SI :tiempo <1 [BP SL GI 90 AV 100 GD 180 BL ROTULA [Demasiado ra pido. Pulse el boto n de freno para
continuar] ALTO]; el texto no cabe en la pantalla, por lo que hay que desplazar el cursor a la
izquierda antes de escribir; el resto del co digo es ide ntico al ejercicio 2.
BP BL
REPITE 4 [AV 100 GD 90]
SL AV 50 GD 90 AV 50 BL
CIRCLE 40
POCCR 4 RELLENA ESPERA :tiempo
POCCR 6 RELLENA ESPERA :tiempo
POCCR 2 RELLENA ESPERA :tiempo
FIN
  
```

Ejercicio corregido

EJERCICIO 11

Haz una última modificación en el semáforo del ejercicio anterior haciendo que el semáforo no cambie sólo una vez sino que al cambiar a verde vuelva a rojo de forma automática.

Basta con llamar al programa para que se repita en la u ltima li nea antes de FIN.

```

PARA semaforo
SI :tiempo <1 [BP SL GI 90 AV 100 GD 180 BL ROTULA [Demasiado ra pido. Pulse el boto n de freno para
continuar] ALTO]
BP BL
REPITE 4 [AV 100 GD 90]
SL AV 50 GD 90 AV 50 BL
CIRCLE 40
POCCR 4 RELLENA ESPERA :tiempo
POCCR 6 RELLENA ESPERA :tiempo
POCCR 2 RELLENA ESPERA :tiempo

semaforo

FIN
  
```



Autoevaluación

EJERCICIO 1

1. Si queremos encender y apagar una bombilla, ¿qué tipo de salida de una tarjeta controladora emplearemos ?



1 Digital

2 Analógica

EJERCICIO 2

2. Para escribir en una sola línea un programa que dibuje un cuadrado de lado 40 debo hacer

1 repite 4 "AV 40 GD 90"

2 repite 4 (AV 40 GD 90)

3 repite 4 AV 40 GD 90

4 repite 4 [AV 40 GD 90]



Autoevaluación

EJERCICIO 3

3. La instrucción POCCR:

- 1 se usa para seleccionar el color de relleno de las figuras que dibujemos
- 2 se usa para seleccionar el color de los trazos del cursor al avanzar
- 3 se usa para seleccionar el color de la pantalla

EJERCICIO 4

4. Identifica las instrucciones con la imagen

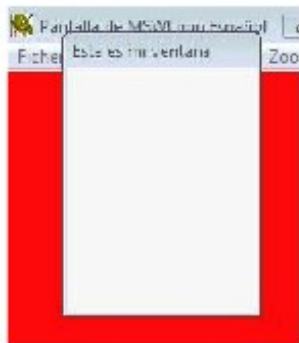
		
gd 90 rotula "hola	Rotula "Hola	gd 90 rotula "hola AV 50 SL



Autoevaluación

EJERCICIO 5

5. Dada la instrucción CREAVENTANA "MIVENTANA [Esta es mi ventana] 80 20 70 100 [poncp 4], señala la afirmación INCORRECTA:



- 1 70 100 son las longitudes del alto y el ancho de la ventana.
- 2 80 20 son las coordenadas que nos indican la posición de la ventana en la pantalla.
- 3 poncp 4 es la instrucción que se da al aparecer la ventana; en este caso se vuelve la pantalla roja.
- 4 70 100 son las coordenadas que nos indican la posición de la ventana en la pantalla.

EJERCICIO 6

6. Creaboton "ven "bot [AD] 10 20 50 15 [AV 50]

La cifra es la altura del botón

El texto entre corchetes es el texto que está dentro del botón

La instrucción entre comillas es el nombre del botón

La instrucción entre comillas es el nombre de la ventana donde está el botón

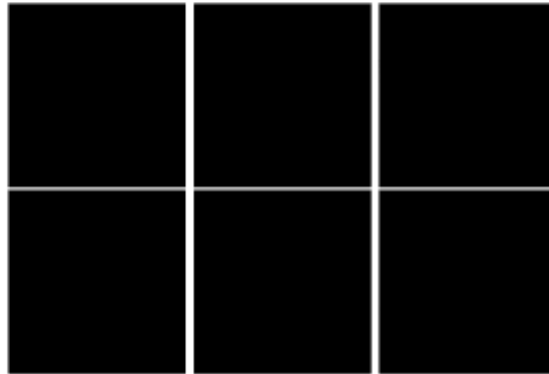
La cifra es posición X del botón respecto de la ventana



Autoevaluación

EJERCICIO 7

7. Empareja las figuras y sus procedimientos



EJERCICIO 8

8. Di qué frase es VERDADERA. El siguiente procedimiento:
PARA saludo ROTULA "Hola ESPERA 30 BP ROTULA [¿Qué tal
estás?] ESPERA 30 BP FIN



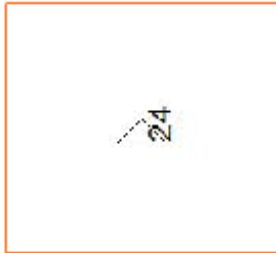



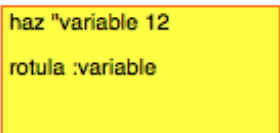
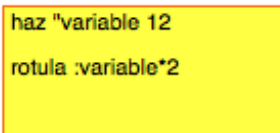
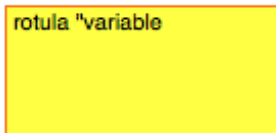
- 1 Como tiene la instrucción ESPERA, primero pone Hola y a los 30 segundos pone ¿Qué tal estás?
- 2 Primero escribe Hola y avanza 30 para escribir ¿Qué tal estás?
- 3 Como tiene la instrucción ESPERA, primero pone Hola y al medio segundo pone ¿Qué tal estás?
- 4 Pone en pantalla Hola, ¿Qué tal estás? a la vez.



Autoevaluación

EJERCICIO 9

9. Identifica cada instrucción con su salida

EJERCICIO 10

10. Completa con la primitiva necesaria :

La primitiva elige al azar un número entre el 0 y el inferior que se le ponga como parámetro.

La primitiva es condicional; ofrece dos posibilidades, escritas entre corchetes y separadas.

La primitiva realiza bucles que se repiten indefinidamente.

La primitiva permite al usuario escribir datos.



Autoevaluación

Ejercicio corregido

EJERCICIO 1

1. Si queremos encender y apagar una bombilla, ¿qué tipo de salida de una tarjeta controladora emplearemos ?



1 Digital

Ejercicio corregido

EJERCICIO 2

2. Para escribir en una sola línea un programa que dibuje un cuadrado de lado 40 debo hacer

0 repite 4 [AV 40 GD 90]



Autoevaluación

Ejercicio corregido

EJERCICIO 3

3. La instrucción POCCR:

se usa para seleccionar el color de relleno de las figuras que dibujemos

Ejercicio corregido

EJERCICIO 4

4. Identifica las instrucciones con la imagen

		
Rotula "Hola	gd 90 rotula "hola	gd 90 rotula "hola AV 50 SL

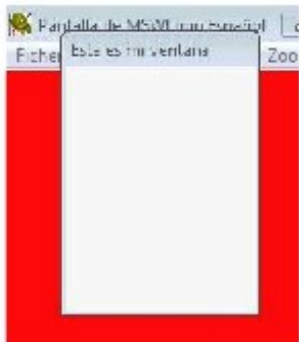


Autoevaluación

Ejercicio corregido

EJERCICIO 5

5. Dada la instrucción CREAVENTANA "MIVENTANA [Esta es mi ventana] 80 20 70 100 [poncp 4], señala la afirmación INCORRECTA:



70 100 son las coordenadas que nos indican la posición de la ventana en la pantalla.

Ejercicio corregido

EJERCICIO 6

6. Creaboton "ven "bot [AD] 10 20 50 15 [AV 50]

La cifra **10** es posición X del botón respecto de la ventana

La instrucción entre comillas **bot** es el nombre del botón

La cifra **15** es la altura del botón

El texto entre corchetes **AD** es el texto que está dentro del botón

La instrucción entre comillas **ven** es el nombre de la ventana donde está el botón

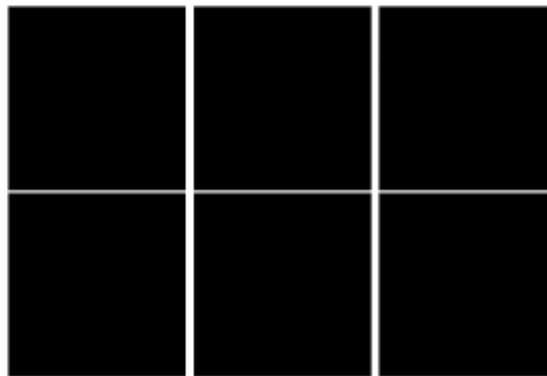


Autoevaluación

Ejercicio corregido

EJERCICIO 7

7. Empareja las figuras y sus procedimientos



Ejercicio corregido

EJERCICIO 8

8. Di qué frase es VERDADERA. El siguiente procedimiento:
PARA saludo ROTULA "Hola ESPERA 30 BP ROTULA [¿Qué tal
estás?] ESPERA 30 BP FIN

- Como tiene la instrucción ESPERA, primero pone Hola y al medio segundo pone ¿Qué tal estás?

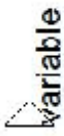




Autoevaluación

Ejercicio corregido

EJERCICIO 9

9. Identifica cada instrucción con su salida

		
rotula "variable"	haz "variable 12 rotula :variable"	haz "variable 12 rotula :variable*2"

Ejercicio corregido

EJERCICIO 10

10. Completa con la primitiva necesaria :

La primitiva **azar** elige al azar un número entre el 0 y el inferior que se le ponga como parámetro.

La primitiva **PREGUNTABOX** permite al usuario escribir datos.

La primitiva **siempre** realiza bucles que se repiten indefinidamente.

La primitiva **sisino** es condicional; ofrece dos posibilidades, escritas entre corchetes y separadas.



Para saber más

LOGO en Linux y Mac

Hay otros muchos intérpretes de LOGO, que puedes consultar en este documento:

http://roble.pntic.mec.es/cgee0028/4esotecnologia/quincena12/pdf/Logo_TreeProject.pdf

De entre éstos podemos destacar xLogo, que es software libre y gratuito y tiene versiones para LINUX y MacOS.

El enlace para descargarte xLogo y acceder a una cuidada y completa documentación lo tienes aquí:

<http://xlogo.tuxfamily.org/sp/presentacion.html>

Controlabot

Es una aplicación práctica de LOGO en Tecnología que encuentras publicada en la página del Observatorio Tecnológico cuya web es:

<http://recursostic.educacion.es/observatorio/web/es/home>

La programación se hace con MSWLogo, tal y como se ha visto en esta unidad.

Descarga de la aplicación:

<http://recursostic.educacion.es/observatorio/web/es/equipamiento-tecnologico/didactica-de-la-tecnologia/792-monografico-controlabot?start=0>