Diccionario de NetLogo en español.

2 de agosto de 2016

Nota del traductor. El presente diccionario se ha elaborado tomando como base el correspondiente diccionario *NetLogo Dictionary* contenido en el manual: *NetLogo User Manual, version 5.2.1*, del 1 de octubre de 2015 y el cual se puede bajar del sitio oficial de NetLogo: https://ccl.northwestern.edu/netlogo/.

Las primitivas se presentan según el siguiente orden alfabético:
A B C D E F G H I J L M N O P R S T U V W X Y ?.

Antes de presentar las primitivas según su orden alfabético, se muestra una clasificación de dichas primitivas agrupadas por categorías, con los respectivos nombres en español entre paréntesis.

Las categorías del listado de primitivas son las siguientes:

Turtle (tortuga) - Patch (parcela) - Agentset (conjunto de agentes) - Color (color) - Task (tarea) - Control/Logic (control/lógica) - World (mundo) - Perspective (perspectiva) - Input/Output (entrada/salida) - Files (archivos) - List (lista) - String (cadena)- Math (matemática) - Plotting (ploteo)- Links (ligas) - Movie (película) - System (sistema) - HubNet (hubnet)
Categorías especiales: Variables (variables) - Keywords (palabras clave) - Constants (constantes)

Primitivas

Primitivas relacionados con tortugas
back (bk) (atrás), <breeds>-at (<familias>-en), <breeds>-here (<familias>-aquí), <breeds>-on (<familias>-sobre), can-move? (puedo-moverme?), clear-turtles (ct) (eliminar-tortugas), create-<breeds> (crear-<familias>), create-ordered-<breeds> (crear-<familias>-ordenadas), create-ordered-turtles (cro) (crear-tortugas-ordenadas),

create-turtles (crt), (crear-tortugas), die (morir), distance (distancia), distancexy (distanciaxy), downhill (pendiente-abajo), downhill4 (pendiente-abajo4), dx (dx), dy (dy), face (mirar-hacia), facexy (mirar-haciaxy), forward (fd), (adelante), hatch (engendrar), hatch-<breeds> (engendrar-<familias>), hide-turtle (ht) (esconder-tortuga), home (casa), inspect (inspeccionar), is-<breed>? (es-<familia>?), is-turtle? (es-tortuga?), jump (saltar), layout-circle (colocarse-en-círculo), left (lt) (izquierda), move-to (moverse-a), myself (yo mismo), nobody (nadie), no-turtles (no-tortugas), of (de), other (otro), patch-ahead (parcela-adelante), patch-at (parcela-en), patch-at-heading-and-distance (parcela-en-orientación-y-distancia), patch-here (parcela-aquí), patch-left-and-ahead (parcela-a-izquierda-y-adelante), patch-right-and-ahead (parcela-a-derecha-y-adelante), pen-down (pd) (pluma-abajo), pen-erase (pe) (pluma-de-borrar), pen-up (pu) (pluma-arriba), random-xcor (azar-xcor), random-ycor (azar-ycor), right (rt) (derecha), self (yo-mismo), set-default-shape (asignar-figura-por-defecto), set-line-thickness (asignar-grosor-de-línea), setxy (asignarxy), shapes (figuras), show-turtle (st) (mostrar-tortuga), sprout (brotar), sprout-<breeds> (engendrar-familias), stamp (estampar), stamp-erase (borrar-estampa), stop-inspecting (detener-inspección), stop-inspecting (detener-inspección), subject (sujeto), subtract-headings (sustraer-orientaciones), tie (atar), towards (hacia), towardsxy (haciaxy), turtle (tortuga), turtle-set (conjunto-de-tortugas), turtles (tortugas), turtles-at (tortugas-en), turtles-here (tortugas-aquí), turtles-on (tortugas-sobre), turtles-own (tortugas-poseen), untie (desatar), uphill (pendiente-arriba), uphill4 (pendiente-arriba4)

Primitivas relacionados con parcelas

clear-patches (cp) (limpiar-parcelas), diffuse (difuminar), diffuse4 (difuminar4), distance (distancia), distancexy (distanciaxy), import-pcolors (importar-pcolors), import-pcolors-rgb (importar-pcolores-rgb), inspect (inspeccionar), is-patch? (es-parcela?), myself (yo mismo), neighbors (vecinos), neighbors4 (vecinos4), nobody (nadie), no-patches (no-parcelas), of (de), other (otro), patch (parcela), patch-at (parcela-en), patch-ahead (parcela-adelante), patch-at-heading-and-distance (parcela-a-orientación-y-distancia), patch-here (parcela-aquí), patch-left-and-ahead (parcela-a-izquierda-y-adelante), patch-right-and-ahead (parcela-a-derecha-y-adelante), patch-set (conjunto-de-parcelas), patches (parcelas), patches-own (parcelas-poseen), random-pxcor (azar-pxcor), random-pycor (azar-pycor), self (yo-mismo), sprout (brotar), sprout-<breeds> (brotar-<familias>), subject (sujeto), turtles-here (tortugas-aquí)

Primitivas relacionados con conjuntos de agentes

all? (todos?), any? (algunos?), ask (solicitar), ask-concurrent (solicitud-concurente), at-points (en-los-puntos), <breeds>-at (<familias>-en), <breeds>-here (<familias>-

aquí), <breeds>-on (familias-sobre), count (contar), in-cone (dentro-del-cono), in-radius (dentro-del-radio), is-agent? (es-agente?), is-agentset? (es-conjunto-agentes?), is-patch-set? (es-conjunto-de-parcelas?), is-turtle-set? (es-conjunto-tortugas?), link-set (conjunto-de-ligas), max-n-of (max-n-de), max-one-of (max-uno-de), member? (miembro?), min-n-of (min-n-de), min-one-of (min-uno-de), n-of (n-de), neighbors (vecinos), neighbors4 (vecinos4), no-links (no-ligas), no-patches (no-parcelas), no-turtles (no-tortugas), of (de), one-of (uno-de), other (otro), patch-set (conjunto-de-parcelas), patches (parcelas), sort (reordenar), sort-by (reordenar-según), sort-on (reordenar-por), turtle-set (conjunto-de-tortugas), turtles (tortugas), turtles-at (tortugas-en), turtles-here (tortugas-aquí), turtles-on (tortugas-sobre), with (con), with-max (con-max), with-min (con-min)

Primitivas relacionados con el color

approximate-hsb (hsb-aproximado), approximate-rgb (rgb-aproximado), base-colors (colores-base), color (color), extract-hsb (extraer-hsb), extract-rgb (extraer-rgb), hsb (hsb), import-pcolors (importar-pcolors), import-pcolors-rgb (importar-pcolors-rgb), pcolor (pcolor), rgb (rgb), scale-color (escala-de-color), shade-of? (sombra-de?), wrap-color (enlazar-color)

Primitivas relacionados con flujo de control y lógica

and (y), ask (solicitar), ask-concurrent (solicitar-concurrente), carefully (cuidadosamente), end (fin), error (error), error-message (mensaje-de-error), every (cada), if (si), ifelse (si-otro), ifelse-value (si-otro-valor), let (permitir-asignación), loop (bucle), not (no), or (o), repeat (repetir), report (reportar), run (correr, ejecutar), runresult (correr-resultado), ; (punto y coma), set (asignar), stop (detener), startup (iniciar), to (para), to-report (para-reportar), wait (esperar), while (mientras), with-local-randomness (con-azar-local), without-interruption (sin-interrupción), xor (xcor)

Primitivas relacionados con tareas

filter (filtrar), foreach (para-cada-uno), is-command-task? (es-tarea-de-comando?), is-reporter-task? (es-tarea-reportadora?), map (mapear), n-values (n-valores), reduce (reducir), run (correr, ejecutar), runresult (correr-resultado), sort-by (reordenar-según), task (tarea)

Primitivas relacionados con el mundo

clear-all (ca) (limpiar-todo), clear-drawing (cd) (limpiar-dibujo), clear-patches (cp) (limpiar-parcelas), clear-ticks (limpiar-ticks), clear-turtles (ct) (limpiar-tortugas),

display (desplegar), import-drawing (importar-dibujo), import-pcolors (importar-pcolors), import-pcolors-rgb (importar-pcolors-rgb), no-display (no-desplegar), max-pxcor (max-pxcolor), max-pycor (max-pycolor), min-pxcor (min-pxcolor), min-pycor (min-pycolor), patch-size (tamaño-de-parcela), stop-inspecting-dead-agents, reset-ticks (restablecer-ticks), resize-world (redimensionar-mundo), set-patch-size (asignar-tamaño-de-parcela), tick (tick), tick-advance (avanzar-tick), ticks (ticks), world-width (ancho-de-mundo), world-height (altura-de-mundo)

Primitivas relacionados con la perspectiva

follow (seguir), follow-me (sígueme), reset-perspective (rp) (restablecer-perspectiva), ride (cabalgar), ride-me (cabálgame), subject (sujeto), watch (observar), watch-me (obsérvame)

Primitivas relacionados con Hubnet

hubnet-broadcast (difundir-hubnet), hubnet-difundir-clear-output (limpiar-salida-difusión-hubnet), hubnet-difundir-message (mensaje-de-difusión-hubnet), hubnet-clear-override (limpiar-deshabilitado-hubnet), hubnet-clear-overrides (limpiar-deshabilitados-hubnet), hubnet-clients-list (lista-de-clientes-hubnet), hubnet-enter-message? (mensaje-de-ingreso-hubnet?), hubnet-exit-message? (mensaje-de-salida-hubnet?), hubnet-kick-all-clients (expulsar-a-todos-clientes-hubnet), hubnet-kick-client (expulsar-cliente-hubnet), hubnet-fetch-message (buscar-mensaje-hubnet), hubnet-message (mensaje-hubnet), hubnet-message-source (fuente-de-mensaje-hubnet), hubnet-message-tag (etiqueta-de-mensaje-hubnet), hubnet-message-waiting? (mensaje-hubnet-esperando?), hubnet-reset (restablecer-hubnet), hubnet-reset-perspective (resetablecer-perspectiva-hubnet), hubnet-send (envío-hubnet), hubnet-send-clear-output (envío-hubnet-limpiar-salida), hubnet-send-follow (hubnet-envía-seguir), hubnet-send-message (hubnet-envía-mensaje), hubnet-send-override (hubnet-envía-deshabilitar), hubnet-send-watch (hubnet-envía-observar), hubnet-set-client-interface (asignar-interfaz-de-cliente-hubnet)

Primitivas relacionados con entrada/salida

beep (bip), clear-output (limpiar-salida), date-and-time (fecha-y-hora), export-view (exportar-vista), export-interface (exportar-interfaz), export-output (exportar-salida), export-plot (exportar-gráfico), export-all-plots (exportar-todos-los-gráficos), export-world (exportar-mundo), import-drawing (importar-dibujo), import-pcolors (importar-pcolors), import-pcolors-rgb (importar-pcolors-rgb), import-world (importar-mundo), mouse-down? (ratón-abajo?), mouse-inside? (ratón-adentro?), mouse-xcor (ratón-xcor), mouse-ycor (ratón-ycor), output-print (imprimir-salida), output-show (mostrar-salida), output-type (imprimir-salida), output-write (escribir-

salida), print (imprimir), read-from-string (leer-de-cadena), reset-timer (restablecer-cronómetro), set-current-directory (asignar-directorio-actual), show (mostrar), timer (cronómetro), type (imprimir), user-directory (directorio-del-usuario), user-file (archivo-del-usuario), user-new-file (archivo-nuevo-del-usuario), user-input (entrada-del-usuario), user-message (mensaje-del-usuario), user-one-of (uno-del-usuario), user-yes-or-no? (si-o-no-del-usuario?), write (escribir)

Primitivas relacionadas con archivos

file-at-end? (final-del-archivo?), file-close (cerrar-archivo), file-close-all (cerrar-todos-los-archivos), file-delete (borrar-archivo), file-exists? (existe-archivo?), file-flush (desaguar-archivo), file-open (abrir-archivo), file-print (imprimir-archivo), file-read (leer-archivo), file-read-characters (leer-caracteres-de-archivo), file-read-line (leer-línea-de-archivo), file-show (mostrar-archivo), file-type (imprimir-archivo), file-write (escribir-archivo), user-directory (directorio-del-usuario), user-file (archivo-del-usuario), user-new-file (archivo-nuevo-del-usuario)

Primitivas relacionadas con listas

but-first (menos-primero), but-last (menos-último), empty? (vacía?), filter (filtrar), first (primero), foreach (para-cada), fput (poner–de-primero), histogram (histograma), is-list? (es-lista?), item (item), last (último), length (longitud), list (lista), lput (poner-de-último), map (mapear), member? (miembro?), modes (modas), n-of (n-de), n-values (n-valores), of (de), position (posición), one-of (uno-de), reduce (reducir), remove (remover), remove-duplicates (remover-duplicados), remove-item (remover-item), replace-item (reemplazar-item), reverse (revertir), sentence (frase), shuffle (barajar), sort (reordenar), sort-by (reordenar-según), sort-on (reordenar-por), sublist (sublista)

Primitivas relacionadas con cadenas

Operadores (<, >, =, ! =, <=, >=) but-first (menos-primero), but-last (menos-último), empty? (vacía?), first (primero), is-string? (es-cadena?), item (item), last (último), length (longitud), member? (miembro?), position (posición), remove (remover) remove-item (remover-item), read-from-string (leer-de-cadena), replace-item (reemplazar-item), reverse (revertir), substring (subcadena), word (palabra)

Primitivas relacionadas con matemática

Operadores aritméticos (+, ∗, −, /, ˆ, <, >, =, ! =, <=, >=) abs (valor absoluto), acos (arco-coseno), asin (arco-seno), atan (arco-tangente), ceiling (techo), cos (coseno), e (número e), exp (exponencial), floor (piso), int (entero), is-number? (es-número?), ln (logaritmo natural), log (logaritmo), max (máximo), mean (me-

dia), median (mediana), min, (mínimo), mod (módulo), modes (modas), new-seed (nueva-semilla), pi (número pi), precision (precisión), random (azar), random-exponential (azar-exponencial), random-float (azar-flotante), random-gamma (azar-gama), random-normal (azar-normal), random-poisson (azar-poisson), random-seed (azar-semilla), remainder (residuo), round (redondear), sin (seno), sqrt (raíz cuadrada), standard-deviation (desviación-estándar), subtract-headings (sustraer-orientaciones), sum (sumar), tan (tangente), variance (varianza)

Primitivas relacionadas con graficación

autoplot? (auto-graficar?), auto-plot-off (auto-graficar-desconectado), auto-plot-on (auto-graficar-conectado), clear-all-plots (limpiar-todos-los-gráficos), clear-plot (limpiar-gráfico), create-temporary-plot-pen (crear-pluma-de-graficar-temporal), export-plot (exportar-gráfico), export-all-plots (exportar-todos-los-gráficos), histogram (histograma), plot (graficar), plot-name (nombre-de-gráfico), plot-pen-exists? (existe-pluma-de-graficar?), plot-pen-down (pluma-de-graficar-abajo), plot-pen-reset (restablecer-pluma-de-graficar), plot-pen-up (pluma-de-graficar-arriba), plot-x-max (max-x-de-gráfico), plot-x-min (min-x-de-gráfico), plot-y-max (max-y-de-gráfico), plot-y-min (min-y-de-gráfico), plotxy (graficarxy), set-current-plot (asignar-gráfico-actual), set-current-plot-pen (asignar-pluma-de-graficar-actual), set-histogram-num-bars (asignar-num-barras-de-histograma), set-plot-pen-color (asignar-color-pluma-de-graficar), set-plot-pen-interval (asignar-intervalo-pluma-de-graficar), set-plot-pen-mode (asignar-modo-pluma-de-graficar), set-plot-x-range (asignar-rango-x-gráfico), set-plot-y-range (asignar-rango-y-gráfico), setup-plots (asignar-gráficos), update-plots (actualizar-gráficos)

Primitivas relacionadas con ligas

both-ends (ambos-extremos), clear-links (limpiar-ligas), create-<breed>-from (crear-<familia>-desde), create-<breeds>-from (crear-<familias>-desde), create-<breed>-to (crear-<familia>-hacia), create-<breeds>-to (crear-<familias>-hacia), create-<breed>-with (crear-<familia>-con), create-<breeds>-with (crear-<familias>-con), create-link-from (crear-liga-desde), create-links-from (crear-ligas-desde), create-link-to (crear-liga-hacia), create-links-to (crear-ligas-hacia), create-link-with (crear-liga-con), create-links-with (crear-ligas-con), die (muere), hide-link (oculta-liga), in-<breed>-neighbor? (ingresa-<familia>-vecino?), in-<breed>-neighbors (ingresa-<familia>-vecinos), in-<breed>-from (ingresa-<familia>-desde), in-link-neighbor? (ingresa-liga-vecino?), in-link-neighbors (ingresa-liga-vecinos), in-link-from (ingresa-liga-desde), is-directed-link? (es-liga-dirigida?), is-link? (es-liga?), is-link-set? (es-conjunto-de-ligas?), is-undirected-link? (es-liga-no-dirigida?), layout-radial (colocación-radial), layout-spring (colocación-resorte), layout-tutte (colocación-tutte),

<breed>-neighbor? (vecino-<familia>?), <breed>-neighbors (vecinos-<familia>), <breed>-with (<familia>-con), link-heading (orientación-de-liga), link-length (longitud-de-liga), link-neighbor? (vecino-de-liga?), link (liga), links (ligas), links-own (ligas-poseen), <link-breeds>-own (<familias-de-ligas>-poseen), link-neighbors (vecinos-de-liga), link-with (liga-con), my-<breeds> (mis-<familas>), my-in-<breeds> (mis-<familias>-entrantes), my-in-links (mis-ligas-entrantes), my-links (mis-ligas), my-out-<breeds> (mis-<familias>-salientes), my-out-links (mis-ligas-salientes), no-links (no-ligas), other-end (otro-extremo), out-<breed>-neighbor? (<familia>-saliente-vecino?), out-<breed>-neighbors, (<familia>-saliente-vecinos), out-<breed>-to (saliente-<familia>-hacia), out-link-neighbor? (saliente-liga-vecino?), out-link-neighbors (saliente-liga-vecinos), out-link-to (saliente-liga-hacia), show-link (muestra-liga), tie (atar), untie (desatar)

Primitivas relacionadas con películas

movie-cancel (cancelar-película), movie-close (cerrar-película), movie-grab-view (capturar-vista-de-película), movie-grab-interface (capturar-interfaz-de-película), movie-set-frame-rate (asignar-razón-de-cuadros-de-película), movie-start (comenzar-película), movie-status (status-de-película)

Primitivas relacionadas Espacio-Conductual (BehaviorSpace)

behaviorspace-experiment-name (nombre-de-experimento-de-behaviorsapce), behaviorspace-run-number (número-de-corrida-de-behaviorspace)

Primitivas relacionadas con el sistema

netlogo-applet? (applet-de-netlogo?), netlogo-version (versión-de-netlogo, netlogo-web? (netlogo-web?)

Variables preinstaladas

Tortugas

breed (familia), color (color), heading (orientación), hidden? (oculto?), label (etiqueta), label-color (color-de-etiqueta), pen-mode (modo-de-pluma), pen-size (tamaño-de-pluma), shape (figura), size (tamaño), who (quien), xcor (xcor), ycor (ycor)

Parcelas

pcolor (pcolor), plabel (petiqueta), plabel-color (color-de-petiqueta), pxcor (pxcor), pycor (pycor)

Ligas

breed (familia), color (color), end1 (extremo1), end2 (extremo2), hidden? (oculta?), label (etiqueta), label-color (color-de-etiqueta), shape (figura), thickness (grosor), tie-mode (modo-de-atadura)

Otras

? (?)

Palabras clave

breed (familia), directed-link-breed (familia-de-ligas-dirigidas), end (fin), extensions (extensiones), globals (globales), includes (inclusiones), links-own (ligas-poseen), patches-own (parcelas-poseen), to (para), to-report (para-reportar), turtles-own (tortugas-poseen), undirected-link-breed (familia-de-ligas-no-dirigidas)

Constantes

Constantes matemáticas
e = 2.718281828459045
pi = 3.141592653589793

Constantes booleanas
false (falso)

true (verdadero)

Constantes de colores

black (negro) = 0
gray (gris) = 5
white (blanco) = 9.9
red (rojo) = 15
orange (naranja =) 25
brown (café) = 35
yellow (amarillo) = 45
green (verde) = 55
lime (lima) = 65
turquoise (turquesa) = 75
cyan (cian) = 85
sky (celeste) = 95
blue (azul) = 105 violet (violeta) = 115
magenta (magenta) 125
pink (rosado) = 135

Ver la sección de colores de la Guía de Programación (Programming Guide) para mayores detalles.

A

abs

abs número

Reporta el valor absoluto de número.

```
show abs -7
-> 7
show abs 5
-> 5
```

---

acos

acos número

Reporta el valor de la función arcocoseno (coseno inverso) de número. El valor de número debe estar en el rango -1 a 1 y la salida se reporta en grados, en el intervalo 0 a 180.

---

all?

all? agentset [reporter]

Reporta true (verdadero) si todos los miembros del conjunto-agentes reportan true. De otra manera reporta false en cuanto encuentra un contraejemplo. Si el conjunto de agentes estuviese vacío reportaría true. La primitiva debe reportar un valor booleano para cada agente (true o false), pues si alguno de los agentes no lo hace, se producirá un error.

```
if all? turtles [color = red]
  [show "todas las tortugas son rojas" ]
```

Ver también any.

---

and

condición1 and condición2

La primitiva and equivale a la conjunción "y" en español y reporta true (verdadero) si condición1 y condición2 son verdaderas. Nótese que si condición1 fuese falsa (false) la condición2 no será evaluada pues su valor ya no afectaría el resultado.

```
if (pxcor > 0) and (pycor > 0)
  [ set pcolor blue ]  ;; las baldosas del cuadrante
                       ;;superior derecho
                       ;; se vuelven azules
```

---

any?

any? conjunto-agentes

Reporta true (verdadero) si el conjunto-agentes no está vació, falso en otro caso. Equivalente a "count agentset > 0", pero es más eficiente y posiblemente más legible.

```
if any? turtles with [color = red]
  [ show "¡al menos una tortuga es roja!" ]
```

Nota: nobody (nadie) no es un conjunto-agentes. No se obtiene como resultado nobody en situaciones en donde se espera un solo agente y no todo un conjunto de agentes. Si a any? se le da nobody como entrada, el resultado será un error.

---

approximate-hsb

approximate-hsb hue saturation brightness

Reporta un número en el rango de 0 a 140, sin incluir 140, que representan el color dado, especificado en el espectro HSB en el espacio de colores de NetLogo. Los tres valores deben estar en el rango de 0 a 255. El color reportado podría ser sólo una aproximación, puesto que el espacio de colores de NetLogo no incluye todos los colores posibles (contiene sólo ciertos hues discretos y para cada hue la saturación o el brillo podría variar, aunque no mucho – al menos uno de ambos siempre es 255).

```
show approximate-hsb 0 0 0
=> 0  ;; (black)
show approximate-hsb 127.5 255 255
=> 85.2 ;; (cyan)
```

Ver también extract-hsb, approximate-rgb, extract-rgb.

---

approximate-rgb

approximate-rgb red green blue

Reporta un número en el rango de 0 a 140, sin incluir 140, el cual representa el color dado, especificado en el espectro RGB, en el espcio de colores de NetLogo. Las tres entradas deben estar en el rango de 0 a 255. El color reportado puede ser sólo una aproximación, puesto que el espacio de colores de NetLogo no incluye todos los posibles colores (ver approximate-hsb para una descripción de qué parte del espacio de colores HSB cubre NetLogo, lo cual es difícil de caracterizar en términos RGB).

```
show approximate-rgb 0 0 0
=> 0  ;; black
show approximate-rgb 0 255 255
=> 85.2 ;; cyan
```

Ver también extract-rgb, approximate-hsb y extract-hsb.

Operadores aritméticos

`(+, *, -, /, ^, <, >, =, !=, <=, >=)`

Todos estos operadores toman dos entradas y actúan "como operadores infijos"(se colocan entre las dos entradas, como se hace comúnmente en matemática). NetLogo respeta la jerarquía usual de los operadores infijos.

Los operadores funcionan del modo siguiente: + es adición, $*$ es multiplicación, − es sustracción, / es división, ^ es exponenciación, < es menor que, > es mayor que, = es igual a, ! = es diferente a, <= es menor o igual que, >= es mayor o igual que.

Note que el operador sustracción (-) siempre toma dos entradas, a menos que se pongan paréntesis a su alrededor, en cuyo caso puede tomar una entrada. Por ejemplo, para expresar el negativo de x, escribimos (- x), incluyendo los paréntesis.

Todos los operadores de comparación se pueden aplicar a las cadenas (strings) y también permiten comparar agentes. Las tortugas se comparan por su número de who (quién). Las parcelas se comparan de arriba hacia abajo y de izquierda a derecha, de modo que la parcela 0 10 es menor que la parcela 0 9 y la 9 0 es menor que la 10 0. Las ligas se ordenan por los agentes de destino y en caso de una unión (tie) por familia (breed). De modo que la liga 0 9 es anterior a la 1 10 pues su agente de destino es menor, y la liga 0 8 es menor que la 0 9. Si hay varias familias de ligas, en el caso de dos ligas con el mismo agente de destino, aquella que no tienen familia precede a la que la posee. Las ligas que pertenecen a familias se ordenan según como las familias se han declarado en el código.

Los conjunto-agentes se pueden comparar mediante igualdad o desigualdad. Dos conjunto-agentes son iguales si son del mismo tipo (tortuga o parcela) y contienen los mismos agentes.

Si usted no está seguro sobre cómo NetLogo interpretará el código, se recomienda agregar paréntesis.

```
show 5 * 6 + 6 / 3
=> 32
show 5 * (6 + 6) / 3
=> 20
```

---

asin

asin número

Reporta el valor del arcoseno (seno inverso) de número. Número debe estar en el rango de -1 a 1. El resultado se reporta en grados y se encuentra en el

rango de -90 a 90.

---

ask

ask conjunto-agentes [**órdenes**]
ask agente [**órdenes**]
El agente o conjunto-agentes ejecutan las órdenes.

```
ask turtles [ fd 1 ]
  ;; todas las tortugas se mueven hacia adelante un paso
ask patches [ set pcolor red ]
  ;; todas las parcelas se vuelven rojas
ask turtle 4 [ rt 90 ]
  ;; sólo la tortuga con número de identidad 4 gira a la derecha
```

Nota: sólo el observador puede usar ask con todas las tortugas o todos los agentes. Esto evita que usted inadvertidamente pida a todas las tortugas que le pidan a todas la tortugas o que todas las parcelas usen ask con todas las parcelas, lo cual es un error común si no se tiene cuidado acerca de cuál agente ejecutará el código que usted está escribiendo.

Nota: Sólo ejecutan los comandos los agente que están en el conjunto-agentes en el momento en que ask comienza a operar.

---

ask-concurrent

ask-concurrent conjunto-agentes [**comandos**]

Esta primitiva existe sólo para mantener compatibilidad con versiones anteriores. No se recomienda su uso en nuevos modelos.

Los agentes en el conjunto-agentes dado ejecutan las órdenes dadas empleando un mecanismo por turnos, para producir concurrencia simulada. Ver la sección Ask-Concurrent de la Programming Guide (Guía de Programación)

para detalles sobre cómo esto funciona.

Nota: Solo los agentes que están en el conjunto-agentes en el momento en que ask comienza, ejecutarán las órdenes. Ver también without-interruption.

---

at-points

conjunto-agentes at-points [[x1 y1] [x2 y2] ...]

Reporta los miembros del conjunto-agentes que se encuentran ubicados en los puntos de coordenadas dadas en la lista de entrada entre corchetes. Esta lista está formada por listas de dos ítemes, que son las coordenadas x, y de cada punto.

Si el solicitante de la orden (the caller) es el observador, entonces las coordenadas están referidas al origen del mundo, donde quiera que éste se haya colocado. Si quien emite la orden es una tortuga, los puntos estan referidos a un sistema de coordenadas cuyo origen estaría colocado en el punto donde ésta se encuentra y dichas coordenadas pueden ser números no enteros.

```
ask turtles at-points [[2 4] [1 2] [10 15]]
  [ fd 1 ]  ;; sólo avanzan un paso las tortugas en los
            ;; puntos de coordenadas (2,4), (1,2) y (10,15),
            ;; relativas a quien da la orden
```

---

atan

atan x y

Convierte las coordenadas x y a ángulo de orientación de la tortuga en grados (de 0 a 360).

Note que esta versión de atan se ha diseñado para adaptarse a la geometría del mundo de NetLogo, donde una orientación de 0 es en dirección hacia arri-

ba, siguiendo luego en sentido horario alrededor del círculo (normalmente en geometría un ángulo de 0 se mide hacia la derecha, un ángulo de 90 hacia arriba y se sigue así en sentido antihorario, y atan se define de acuerdo a este criterio).

Cuando y es 0: si x es positivo, reporta 90, si x es negativo reporta 270 y si x es 0 se produce un error.

```
show atan 1 -1
=> 135
show atan -1 1
=> 315
crt 1 [ set heading 30  fd 1  print atan xcor ycor ]
=> 30
```

En el ejemplo final, note que el resultado de atan es igual a la orientación de la tortuga.

En caso de que usted necesite convertir la orientación de la tortuga (obtenida mediante atan o de otra manera) a un ángulo matemático normal, lo siguiente podría ser de utilidad:

```
to-report heading-to-angle [ h ]
  report (90 - h) mod 360
end
```

---

autoplot?
autoplot?

Reporta true (verdadero) si auto-ploting (auto-graficación) está en "on" para el gráfico (plot) actual, falso en otro caso.

---

auto-plot-off
auto-plot-on

auto-plot-off
auto-plot-on

Este par de órdenes se usan para controlar la característica de auto-graficación (auto-plotting) de NetLogo en un gráfico. El auto-plotting ajustará automáticamente los ejes x y y del gráfico cuando la pluma en uso excede sus límites. Resulta útil cuando se desea mostrar todos los valores graficados en un mismo gráfico, sin tomar en cuenta los rangos de cada uno.

B

---

back
bk
back número

La tortuga se mueve hacia atrás una cantidad de pasos dada por número, (si número es negativo la tortuga se mueve hacia adelante). Las tortugas utilizan un máximo de un paso por unidad de tiempo. Por ejemplo, bk 0.5 o bk 1 utilizan una unidad de tiempo pero bk 3 utiliza 3 unidades de tiempo. Si la tortuga no puede caminar la cantidad indicada porque la topología del mundo se lo impide, entonce caminará la cantidad máxima posible de pasos completos (redondeando la entrada número a su parte entera) y luego se detendrá.
Ver también forward, jump, can-move?

---

base-colors
base-colors

Reporta una lista de los 14 colores hue básicos de NetLogo.

```
print base-colors
=> [5 15 25 35 45 55 65 75 85 95 105 115 125 135]
ask turtles [ set color one-of base-colors ]
;; cada tortuga toma un color base al azar
ask turtles [ set color one-of remove gray base-colors ]
;; cada tortuga toma un color base al azar exceptuando el gris
```

---

beep
beep

Emite un sonido (bip). Note que el bip suena inmediatamente, de modo que varias órdenes beep en sucesión producen sólo un sonido.

```
beep                          ;; emite un beep
repeat 3 [ beep ]             ;; emite 3 beeps de una sola vez,
                              ;; de modo que uste oirá sólo un sonido
repeat 3 [ beep wait 0.1 ]    ;; produces 3 beeps una tras el otro,
                              ;; separados por una décima de segundo
```

Cuando se corre headless, esta orden no tiene efecto alguno.

---

behaviorspace-experiment-name
behaviorspace-experiment-name

Reporta el nombre del experimento actual en el experimento actual
Si no está corriendo ningún experimento de BehaviorSpace reporta “”.

---

behaviorspace-run-number

behaviorspace-run-number

Reporta el número actual en el experimento BehaviorSpace actual, comenzando en 1. Si no hay ningún experimento BehaviorSpace corriendo reporta 0.

---

both-ends
both-ends

Reporta el conjunto-agentes de los dos nodos que la liga conecta.

```
crt 2
ask turtle 0 [ create-link-with turtle 1 ]
ask link 0 1 [
  ask both-ends [ set color red ]
;; las tortugas 0 y 1 toman el color rojo ]
```

---

breed
breed

Esta variable pertenece a las tortugas y ligas y está preinstalada dentro del sistema. En ella se guarda el conjunto-agentes de todas las tortugas o ligas de la misma familia (breed) de la tortuga o liga. Para tortugas o ligas que no pertenecen a ninguna familia en particular, el conjunto-agentes sería entonces el conjunto de todas las tortugas o de todas la ligas.
Se puede usar esta variable para cambiar la familia de una tortuga o de una liga. Cuando una tortuga cambia de familia, su figura se cambia a la figura por defecto de la nueva familia. Ver set-default-shape.

Ver también directed-link-breed, undirected-link-breed.

```
breed [gatos gato]
breed [perros perro]
;; código de tortuga:
if breed = gatos [ show "¡miau!" ]
set breed perros
```

```
show "¡guau!"
directed-link-breed [ carreteras carretera ]
;; código de liga
if breed = carreteras [ set color gray ]
```

---

breed

breed [< *breeds* > < *breed* >]

Esta primitiva, al igual que las primitivas globales turtles-own y patches-own, sólo puede usarse al inicio del código, antes de las definir los procedimientos y sirve para definir una familia. La primera entrada define el nombre del conjunto-agentes asociado con la familia. La segunda entrada (en singular) define el nombre de un miembro individual de la familia.

Cualquier tortuga de la familia dada:
-Es parte del conjunto-agentes especificado por el nombre de la familia.
-Posee el conjunto de variables que el sistema asigna a este conjunto-agentes.

Con frecuencia el conjunto-agentes se usa junto con la orden ask, para dar órdenes solamente a las tortugas de una familia en particular.

```
breed [ratones ratón]
breed [sapos sapo]
to setup
  clear-all
  create-ratones 50
  ask ratones [ set color white ]
  create-sapos 50
  ask sapos [ set color green ]
  show [breed] of one-of ratones    ;; imprime ratones
  show [breed] of one-of sapos   ;; imprime sapos
end

show ratón 1
```

```
;; imprime (ratón 1)
show sapo 51
;; imprime (sapo 51)
show turtle 51
;; imprime (sapo 51)
```

Ver también las globales: patches-own, turtles-own, < *breeds* >-own, create-< *breeds* >, < *breeds* >-at, < *breeds* >-here.

---

but-first
butfirst
bf
but-last
butlast
bl

but-first list
butfirst string
but-last list
butlast string

Cuando se usa con una lista, but-first reporta todos los miembros de la lista con excepción de su primer miembro y but-last los reporta todos con excepción del último. En el caso de una cadena but-first reporta la cadena sin su primer carácter y but-last la reporta sin su último carácter.

```
show but-first [2 4 6 5 8 12]
;; imprime [4 6 5 8 12]
show but-last [2 4 6 5 8 12]
;; imprime [2 4 6 5 8]
show but-first "primavera"
;; imprime "rimavera"
show but-last "primavera"
;; imprime "primaver"
```

C

can-move?
can-move? distancia

Reporta true (verdadero) si la tortuga se puede mover la distancia indicada en la dirección en que apunta pero sin violar la topología del mundo. En caso contrario reporta false.

Es equivalente a:

patch-ahead distancia != nobody

---

carefully
carefully [**instrucciones1**] [**instrucciones2**]

Ejecuta las instrucciones1. En caso de que ocurra un error en las instrucciones1, NetLogo no se detendrá y alertará al usuario sobre la existencia del error, suprimirá el error y a cambio ejecutará instrucciones2.

La reportadora error-message (mensaje de error) se puede usar en instrucciones2 para encontrar el error que se suprimió en instrucciones1. Ver error-message.

```
carefully [ print one-of [1 2 3] ] [ print error-message ]
=> 3
observador> carefully [ print one-of [] ] [ print error-message ]
=> ONE-OF got an empty list as input
(traduc. ONE-OF recibió una lista vacía como entrada).
```

---

ceiling
ceiling número

Reporta el menor número que es mayor o igual a la entrada número.

```
show ceiling 4.5
=> 5
show ceiling -4.5
=> -4
See also floor, round, precision.
```

---

clear-all
ca
clear-all

Orden que pertenece al observador. Tiene el efecto combinado de clear-globals, clear-ticks, clear-turtles, clear-patches, clear-drawing, clear-all-plots, and clear-output.

---

clear-all-plots
clear-all-plots

Borra todos los gráficos en el modelo. Ver clear-plot para más información.

---

clear-drawing
cd
clear-drawing

Borra todas las líneas y estampas dibujadas por las tortugas.

---

clear-globals
clear-globals

Reasigna a todas las variables globales el valor 0.

---

clear-links
clear-links

Elimina (mata) todas las ligas.
Ver también die.

---

clear-output
clear-output

Borra todo texto del área de salida del modelo, caso de habérsele asignado alguna. De lo contrario no tiene ningún efecto.

---

clear-patches
cp
clear-patches

Limpia todas las parcelas, en el sentido de reestablecer todas las variables de parcelas a sus valores por defecto, incluso restableciendo su color negro.

---

clear-plot
clear-plot

Restablece todas las plumas en el gráfico actual, borra todas las plumas temporales, restablece el gráfico a sus valores por defecto (rango de la x, rango de la y, etc.) y restablece todas las plumas permanentes a sus valores por defecto. Los valores por defecto del gráfico y de las plumas permanentes se establecen en la ventana Edit, la cual se despliega cuando se edita el gráfico. Si después de borrar las plumas temporales no quedan plumas graficadoras,

es decir, si no quedan plumas permanentes, una pluma por defecto se crea con las especificaciones siguientes:

Pen: down (abajo)
Color: black
Mode: 0 (modo de línea)
Name: "default"
Interval: 1
Ver también clear-all-plots.

---

clear-ticks
clear-ticks

Borra el contador de ticks. No restablece el contador al valor cero. Después de esta instrucción, el contador de ticks no tiene valor alguno y cualquier intento de acceder al mismo o de actualizarlo generará un error, hasta tanto no se invoque la orden reset-ticks.
Ver también reset-ticks.

---

clear-turtles
ct
clear-turtles

Elimina todas las tortugas. También reestablece los número who, de modo que se le asignará el número 0 a la siguiente tortuga en ser creada.
Ver también die.

---

color
color

Esta es una variable preinstalada en el sistema la cual pertenece a las tortugas o a las ligas. La variable almacena el color de la tortuga o liga. Se puede

reasignar el valor de la variable y hacer que la tortuga o liga cambie de color. El color se puede representar como color de NetLogo (dado por un solo número) o como color RGB (dado por una lista de 3 números). Ver detalles en la sección Colors de la Guía de Programación.

Ver también pcolor.

---

cos
cos número

Reporta el coseno del ángulo dado. Se supone que el ángulo se dará en grados.

```
show cos 180
=> -1
```

---

count
count conjunto-agentes

Reporta el número de agentes en el conjunto-agentes dado.

```
show count turtles
;; imprime el número total de tortugas
show count patches with [pcolor = red]
;; imprime el número total de parcelas rojas
```

---

create-ordered-turtles
cro
create-ordered-< *breeds* >
create-ordered-turtles número
create-ordered-turtles número [**órdenes**]
create-ordered< *breeds* > **número**
create-ordered< *breeds* > **número** [**órdenes**]

Crea tantas nuevas tortugas como lo indica número. Las tortugas creadas nacen en la posición (0, 0) y se les asigna al azar un color entre los 14 colores primarios y sus orientaciones se distribuyen espaciándolas uniformemente entre 0 y 360 grados.
Si se usa la forma create-ordered-< *breeds* >, las nuevas tortugas nacen como miembros de la familia dada. Si se agregan órdenes entre corchetes, dichas órdenes son ejecutadas inmediatamente. Esto es útil para asignar colores u orientaciones específicas a las tortugas, entre otras cosas. (Aunque las tortugas son creadas de una sola vez, ellas ejecutan las órdenes una a la vez, en orden aleatorio).

```
cro 100 [ fd 10 ]  ;; las tortugas se posicionan uniformemente
                   ;; en conformación circular
```

---

create-< *breed* >-to
create-< *breeds* >-to
create-< *breed* >-from
create-< *breeds* >-from
create-< *breed* >-with
create-< *breeds* >-with
create-link-to
create-links-to
create-link-from
create-links-from
create-link-with
create-links-with

create-< *breed* >-to turtle
create-< *breed* >-to turtle [órdenes]
create-< *breed* >-from turtle
create-< *breed* >-from turtle [órdenes]
create-< *breed* >-with turtle
create-< *breed* >-with turtle [órdenes]
create-< *breeds* >-to conjunto-de-tortugas
create-< *breeds* >-to conjunto-de-tortugas [órdenes]
create-< *breeds* >-from conjunto-de-tortugas

create-< *breeds* >**-from conjunto-de-tortugas** [**órdenes**]
create-< *breeds* >**-with conjunto-de-tortugas**
create-< *breeds* >**-with conjunto-de-tortugas** [**órdenes**]
create-link-to turtle
create-link-to turtle [**órdenes**]
create-link-from turtle
create-link-from turtle [**órdenes**]
create-link-with turtle
create-link-with turtle [**órdenes**]
create-links-to conjunto-de-tortugas
create-links-to conjunto-de-tortugas [**órdenes**]
create-links-from conjunto-de-tortugas
create-links-from conjunto-de-tortugas [**órdenes**]
create-links-with conjunto-de-tortugas
create-links-with conjunto-de-tortugas [**órdenes**]

Se usa para crear ligas y tortugas pertenecientes o no a una familia.

create-link-with crea una liga no dirigida entre el solicitante que da la orden y el agente. create-link-to crea un liga dirigida del solicitante al agente. create-link-from crea una liga dirigida del agente al solicitante. Cuando se usa la forma plural del nombre de la familia, se espera un conjunto-agentes en vez de un agente y se crean ligas entre el solicitante que da la orden y todos los agentes en el conjunto-agentes.

El bloque de órdenes opcionales es el conjunto de órdenes que ejecuta cada liga nueva formada. Las ligas se crean todas juntas pero luego ejecutan las órdenes una a la vez en orden aleatorio.

Un nodo no puede establecer una liga consigo mismo. Tampoco se puede tener más de una liga no dirigida de la misma familia entre los dos mismos nodos, ni se puede tener dos ligas dirigidas y en la misma dirección de la misma familia entre dos nodos. Si usted trata de crear una liga en donde ya hay una (de la misma familia), no ocurre nada. Si trata de crear una liga de

una tortuga a sí misma se produce un runtime error (error de ejecución).

```
to setup
   crt 5
   ;; la tortuga 1 crea ligas con todas la otras tortugas
   ;; la liga entre la tortuga y ella misma se ignora
   ask turtle 0 [ create-links-with other turtles ]
   show count links ;; muestra 4
   ;; esto no produce ningún efecto pues la liga ya existe
   ask turtle 0 [ create-link-with turtle 1 ]
   show count links ;; muestra 4 puesto que la liga previa ya existía
   ask turtle 2 [ create-link-with turtle 1 ]
   show count links ;; muestra 5
end

directed-link-breed [red-links red-link]
undirected-link-breed [blue-links blue-link]

to setup
  crt 5
  ;; crea ligas en ambas direcciones entre la tortuga 0
  ;; y todas las otras tortugas
  ask turtle 0 [ create-red-links-to turtles ]
  ask turtle 0 [ create-red-links-from turtles ]
  show count links ;; muestra 8
  ;; ahora crea ligas no dirigidas entre la tortuga 0
  ;; y las demás tortugas
  ask turtle 0 [ create-blue-links-with turtles ]
  show count links ;; muestra 12
end
```

---

create-turtles
crt
create-< *breeds* >

create-turtles número

create-turtles número [**órdenes**]
create-< *breeds* > **número**
create-< *breeds* > **número** [**órdenes**]

Crea tantas tortugas en el origen como lo indica número. Las nuevas tortugas tienen orientaciones de números enteros elegidos al azar y su color se escoge también al azar entre los 14 colores primarios.

Si se usa la forma create-<breeds>, las nuevas tortugas serán miembros de la familia dada.

Si se incluyen órdenes entre los corchetes, las nuevas tortugas las ejecutarán de inmediato aunque de una en una y las tortugas se eligen siguiendo un orden aleatorio. Esto es útil para dar a las tortugas colores, orientaciones u otras características específicas.

```
crt 100 [ fd 10 ]     ;; las tortugas se colocan en un círculo
                      ;; espaciadas al azar.
breed [canarios canario]
breed [serpientes serpiente]
to setup
  clear-all
  create-canarios 50 [ set color yellow ]
  create-serpientes 50 [ set color green ]
end

Ver también hatch y sprout.
```

---

create-temporary-plot-pen

create-temporary-plot-pen cadena

En el gráfico actual se crea una nueva pluma temporal con el nombre dado y se establece como la pluma actual. Pocos modelos querrán usar esta primitiva, porque todas las plumas temporales desaparecen cuando se invoca clear-plot o clear-all-plots. La manera normal de crear una pluma es crearla

como pluma permanente en la correspondiente ventana de diálogo Edit del gráfico.

Si ya existe una pluma temporal con el mismo nombre en el gráfico actual, no se creará una nueva pluma y la que ya existe se converitrá en la pluma actual. Si ya existiera una pluma permanente con el mismo nombre, se produciría un runtime error (error de ejecución).

La nueva pluma temporal tiene las siguientes características iniciales:

Pen: down (bajo)
Color: black (negro)
Mode: 0 (modo de línea)
Interval: 1

Ver: clear-plot, clear-all-plots, y set-current-plot-pen.

D

---

date-and-time
date-and-time

Reporta una cadena que contiene la fecha y hora actual. El formato se muestra adelante. Todos los campos tienen una anchura fija, de modo que siempre quedan en los mismos lugares en la cadena. La resolución potencial del reloj es de milisegundos. La precisión de esta resolución puede variar de sistema a sistema, dependiendo de la capacidad de la Máquina Virtual Java subyacente.

```
show date-and-time
=> "01:19:36.685 PM 19-Sep-2002"
```

---

die
die

La tortuga o liga muere.

```
if xcor > 20 [ die ]
;; todas las tortugas con xcor mayor que 20 mueren
ask links with [color = blue] [ die ]
;; todas las ligas azules morirán
```

Un agente muerto deja de existir. Los efectos de esto incluyen:
El agente no ejecutará ningún otro código, de modo que si uted escribe ask turtles [die print "últimas palabras"], no se imprimirán esas útlimas palabras porque las tortugas ya están muertas antes de que puedan imprimir algo. El agente desaparecerá del conjunto-agentes al que pertenecía, quedando este conjunto disminuido en una unidad.

Cualquier variable que almacenaba el agente no tendrá ahora a nadie (nobody) en ella. Así por ejemplo, la instrucción let x one-of turtles ask x [die] print x imprimirá nobody. Si el agente era una tortuga, toda liga conectada a ella también muere. Si el observador estaba mirando (watching) o siguiendo (following) al agente, la perspectiva del observador se restablece, como si se invocara restablecer (reset) la perspectiva.
Ver también: clear-turtles clear-links

---

diffuse
diffuse variable-de-parcela número

Pide a cada parcela dar igual pocentage de (número * 100) del valor de variable-de-parcela a sus ocho parcelas vecinas, donde número debe tomar un valor entre 0 y 1. Independientemente de la topología, la suma de valores de variable-de- parcela se conserva por todo el mundo. Si una parcela tiene menos de ocho vecinos, cada vecino sigue recibiendo un octavo del valor y la parcela se queda con el sobrante.

Note que esta es una primitiva sólo del observador, aunque tal vez usted esperaba que fuese de las parcelas. La razón es que actúa sobre todas las parcelas a la vez, mientras que las primitivas de parcelas actúan sobre parcelas individuales.

```
diffuse hormona 0.5
;; cada parcela difumina 50% de su variable
;; hormona a sus 8 parcelas vecinas. Por tanto
;; cada parcela recibe 1/8 del 50% de la hormona
;; de cada parcela vecina.
```

---

diffuse4

diffuse4 variable-de-parcela número

Igual que diffuse, pero sólo difumina a las cuatro parcelas vecinas (al norte, sur, este y oeste), no a las parcelas diagonales.

```
diffuse4 hormona 0.5
;; cada parcela difumina 50% de su variable
;; hormona a sus 4 parcelas vecinas. Por tanto
;; cada parcela recibe 1/4 del 50% de la hormona
;; de cada parcela vecina.
```

---

directed-link-breed

directed-link-breed $[< link - breeds > < link - breed >]$

Esta primitiva, igual que las primitivas globales y de familias, sólo pueden usarse el comienzo del código, antes de las definiciones de los procedimientos y define una familia de ligas dirigidas. Las ligas de una familia particular deben ser todas dirigidas o no dirigidas. La primera entrada define el nombre del conjunto-agentes asociado a la familia de ligas. La segunda entrada define el nombre de un miembro individual de la familia. Las ligas dirigidas se pueden crear usando create-link(s)-to y create-link(s)-from, pero no con create-link(s)-with.

Cualquier liga de la familia dada:
-Es parte del conjunto-agentes denominado por el nombre de la familia de ligas,
-tiene las variables preinstaladas en el sistema configuradas a ese conjunto-agentes,
-es dirigida o no dirigida, según lo especifica la primitiva.

La mayoría de ls veces, el conjunto-agentes se emplea conjuntamente con la orden ask para dar órdenes sólo a las ligas de las familia en particular.

```
directed-link-breed [calles calle]
directed-link-breed [autopistas autopista]
to setup
  clear-all
  crt 2
  ;; crea una liga de la tortuga 0 a la tortuga 1
  ask turtle 0 [ create-calle-to turtle 1 ]
  ;; crea una liga de la tortuga 1 a la tortuga 0
  ask turtle 0 [ create-autopista-from turtle 1 ]
end

ask turtle 0 [ show one-of my-in-links ]
;; imprimer (calle 0 1)
ask turtle 0 [ show one-of my-out-links ]
;; imprime (autopista 1 0)
```

Ver también breed, undirected-link-breed

---

display
display

Hace que la vista (view) se actualice inmediatamente. Se exceptúa el caso cuando el usuario está usando el deslizador de velocidad para avanzar más rápido el modelo, entonces la actualización se puede omitir.

También anula el efecto de la primitiva no-display, de modo que si las actualizaciones de la vista se hubieran suspendido por esta primitiva, las activaría de nuevo.

```
no-display
ask turtles [ jump 10 set color blue set size 5 ]
display
;; las tortugas se mueven, cambian de color y crecen, sin que
;; los estados intermedios sean visibles para el usuario,
;; únicamente el estado final
```

Aún si no-display no se ha usado, "display" puede ser útil porque, en su modo ordinario, NetLogo siempre puede omitir algunas actualizaciones de vistas con el fin de que el modelo corra más rápido. Con esta primitiva podemos forzar una actualización de vista para que cualquier cambio que ha tenido lugar en el mundo resulte visible para el usuario.

```
ask turtles [ set color red ]
display
ask turtles [ set color blue]
;; las tortugas se vuelven rojas, luego azules; el uso de ‘‘display’’
;; fuerza a las tortugas rojas a aparecer brevemente
```

Note que display y no-display operan independientemente del interruptor Actualizar Vista que congela la vista.
Ver también no-display.

---

distance

distance agente

Reporta la distancia del solicitante que da la orden (the caller) al agente dado como entrada.

La distancia hacia o desde una parcela se mide desde el centro de la misma. Si el mundo tiene una topología de enlazamiento (toro o cilindro), tanto tortugas como parcelas emplearán la distancia más corta según dicha topología.

```
ask turtles [ show max-one-of turtles [distance myself] ]
;; cada tortuga imprime el número de identificación
;; de la tortuga más alejada de sí misma.
```

---

distancexy

distancexy x y

Reportq la distancia del agente al punto de coordenadas (x y).

La distancia desde una parcela se mide desde el centro de la misma. Si el mundo tiene una topología de enlazamiento (toro o cilindro), tanto tortugas como parcelas emplearán la distancia más corta según dicha topología.

```
if (distancexy 0 0) > 10
  [ set color green ]
;; las tortugas a más de 10 unidades de distancia
;; del centro del mundo se vuelven verdes.
```

---

downhill
downhill4

downhill variable-de-parcela
downhill4 variable-de-parcela

Mueve la tortuga a la parcela vecina con el menor valor de variable-de-parcela. Si no hay una parcela con menor valor al de la parcela actual, la tortuga se queda quieta. Si hay varias parcelas con el mismo valor más bajo, la tortuga escoge una al azar. Valores no numéricos no se toman en cuenta.

La primitiva downhill toma en cuenta las ocho parcelas vecinas mientras que downhill4 sólo considera las cuatro vecinas en los cuatro puntos cardinales. Equivale al siguiente código (suponiendo que los valores de la variable son numéricos):

```
move-to patch-here  ;; se coloca en el centro de la parcela
let p min-one-of neighbors [variable-de-parcela]  ;; o bien neighbors4
if [patch-variable] of p < patch-variable [
  face p
  move-to p
]
```

Note que la tortuga siempre termina en el centro de una parcela y con una orientación que es múltiplo de 45 (downhill) o de 90 (downhill4).
Ver también uphill, uphill4.

---

dx
dy
dx
dy

Reporta el incremento-x o el incremento-y (la cantidad en que la coordenada xcor o ycor cambiaría) si la tortuga diera un paso hacia adelante en la dirección en que apunta su orientación.

Nota: dx es simplemente el seno de la orientación y dy es simplemente el coseno. Si esto es opuesto a lo que usted esperaba, la razón es que en NetLogo la orientación 0 es hacia el norte y 90 es hacia el este, contrariamente a como se miden usualmente los ángulos en geometría.

Nota: en versiones anteriores de NetLogo, estas primitivas se usaban en muchas situaciones en que ahora la nueva primitiva patch-ahead resulta más apropiada.

E

empty?
empty? lista
empty? cadena

Reporta true (verdadero) si la lista o cadena dada está vacía o false en caso contrario.

Nota: la lista vacía se escribe [ ] y la cadena vacía ¨¨.

end
end

Se usa al final de todo procedimiento. Ver to and to-report.

end1
end1

Es una variable preinstalada en el sistema, la cual indica el primer extremo (tortuga) de una liga. Para ligas dirigidas end1 siempre es el extremo inicial u origen de la liga, para ligas no dirigidas es la tortuga con el menor número who. No se puede reasignar el valor de end1 usando set.

```
crt 2
ask turtle 0
[ create-link-to turtle 1 ]
ask links
[ show end1 ] ;; muestra la turtle 0
```

---

end2
end2

Es una variable preinstalada en el sistema, la cual indica el segundo extremo (tortuga) de una liga. Para ligas dirigidas end2 siempre es el extremo final o destino de la liga, para ligas no dirigidas es la tortuga con el mayor número who. No se puede reasignar el valor de end2 usando set.

```
crt 2
ask turtle 1
[ create-link-with turtle 0 ] ;; liga no dirigida
ask links
[ show end2 ] ;; muestra la tortuga 1
```

---

error
error valor

Provoca un runtime error (error de ejecución). El valor dado es convertido a una cadena (caso que ya no lo fuera) y usado como mensaje de error.

Ver también error-message y carefully.

---

error-message
error-message

Reporta una cadena que describe el error que ha sido suprimido mediante carefully. Esta reportadora sólo se puede usar en el segundo bloque de la primitiva carefully.

Ver también error y carefully.

---

every
every número [órdenes]

Ejecuta las órdenes dadas solamente cuando ha transcurrido el mismo número de segundos desde la última vez que el agente las ejecutó dentro de este contexto. En caso contrario las órdenes son ignoradas.
Por sí misma, every no hace que las órdenes se ejecuten una y otra vez. Si usted desea que las órdenes se ejecuten una y otra vez, deberá usar every dentro de un lazo (loop) o dentro de un botón forever. Every sólo limita la frecuencia con que se repite el bloque de órdenes.

El sentido de la expresió "dentro de este contexto" empleada al inicio de esta explicación significa durante el mismo ask (o botón u orden escrita en la ventana del observador). Por tanto no tiene sentido escribir ask turtles [every 0.5 [...]], porque cuando el ask finaliza, todas las tortugas desechan sus cronómetros del "every". El uso correcto se muestra seguidamente.

```
every 0.5 [ ask turtles [ fd 1 ] ]
;; dos veces por segundo las tortugas se moverán hacia adelante 1 paso
every 2 [ set index index + 1 ]
;; cada 2 segundos se aumenta index en una unidad
```

Ver también wait.

---

exp
exp número

Reporta el valor del número e elevado al exponente número.

Nota: Esto es lo mismo que e ^ número .

---

export-view

export-interface
export-output
export-plot
export-all-plots
export-world

export-view nombre-de-archivo
export-interface nombre-de-archivo
export-output nombre-de-archivo
export-plot nombre-de-gráfico nombre-de-archivo
export-all-plots nombre-de-archivo
export-world nombre-de-archivo

export-view escribe el contenido actual de la vista vigente a un archivo externo dado por la cadena nombre-de-archivo. El archivo se guarda en formato PNG (Portable Network Graphics), por lo que se recomienda darle un nombre con extensión ".png".
export-interface es similar, pero para toda la pestaña Ejecutar.
Note que export-view aún funciona cuando NetLogo se corre en modo headless (sin cabeza), mientras que export-interface no.
export-output escribe el contenido del área de salida (output area) del modelo a un archivo externo dado por la cadena nombre-de-archivo. Si el modelo no tiene una área de salida aparte, se usa la Terminal de Instrucciones.

export-plot escribe los valores x y y de todos los puntos graficados por todas las plumas en el gráfico dado por la cadena nombre-de-gráfico a un archivo externo dado por la cadena nombre-de-archivo. Si una pluma está en modo de barras (modo 0) y el valor y del punto es mayor que 0, se exportará el punto de la esquina superior izquierda de la barra. Si el valor de y es menor que 0, se exportará el punto de la esquina inferior izquierda de la barra.

export-all-plots escribe cada gráfico del modelo actual a un archivo externo dado por la cadena nombre-de-archivo. Cada gráfico es idéntico en formato al producido por export-plot.

export-world escribe los valores de todas las variables, tanto las preinstaladas dentro del sistema como las definidas por el usuario, incluyendo todas las variables del observador, tortugas o celdillas, el dibujo, el contenido de la

área de salida (output area), caso de existir alguna, el contenido de cualesquiera gráficos y el estado del generador de números aleatorios, a un archivo externo dado por la cadena nombre-de-archivo. El archivo resultante puede ser leido de nuevo por NetLogo con la primitiva import-world. export-world no guarda el estado de archivos abiertos.

export-plot, export-all-plots and export-world guardan los archivos como texto sin formato en la modalidad de "comma-separated values" (valores separados por coma), con extensión .csv. Los archivos csv pueden ser leídos por los programas de hojas electrónicas o de bases de datos más comunes, lo mismo que por cualquier editor de texto.

Si usted desea exportar un archivo a un sitio diferente al lugar del modelo por defecto, usted debe incluir la ruta completa del archivo (usar el caracter "/" como separador de carpetas). Note que la funcionalidad de estas primitivas también se encuentra disponbile en el menú Archivos de NetLogo.

```
export-world "fuego.csv"
;; exporta el estado del modelo al archivo fuego.csv
;; ubicado en la carpeta de NetLogo
export-plot "Temperatura" "c:/Mis Documentos/grafico.csv"
;; exporta el gráfico llamado
;; "Temperatura" al archivo grafico.csv ubicado en
;; la carpeta C:\Mis Documentos
export-all-plots "c:/Mis Documentos/graficos.csv"
;; exporta todos los gráficos al archivo graficos.csv
;; ubicado en la carpeta C:\Mis Documentos
```

Si el archvio ya existe, se escribirá sobre él. Para evitar esto, tal vez usted querrá usar algún método para generar nombres nuevos. Ejemplos:

```
export-world user-new-file
export-world (word "resultados " date-and-time ".csv")
;; Usar el caracter ":" en la hora causa errores en Windows
export-world (word "resultados " random-float 1.0 ".csv")
```

---

extensions

extension [nombres...]

Permite que el modelo use primitivas de las extensiones nombradas entre los corchetes. Ver la sección de Extensiones del User's Manual para mayor información.

---

extract-hsb

extract-hsb color

Reporta una lista de tres valores en el rango de 0 a 255, que representan hue, saturación y brillo respectivamente, del color NetLogo dado en el rango de 0 a 140, excluyendo 140.

```
show extract-hsb red
=> [2.198 206.372 215]
show extract-hsb cyan
=> [127.5 145.714 196]
```

Ver también approximate-hsb, approximate-rgb, extract-rgb.

---

extract-rgb

extract-rgb color

Reporta una lista de tres valores en el rango de 0 a 255 que representan los niveles de rojo, verde y azul, respectivamente , del color NetLogo dado en el rango de 0 a 140, excluyendo 140.

```
show extract-rgb red
=> [215 50 41]
show extract-rgb cyan
=> [84 196 196]
```

Ver también approximate-rgb, approximate-hsb, extract-hsb.

F

---

face
face agente

Da al solicitante una orientación de cara hacia el agente. Bajo las topologías de enlazamiento (toro, cilindros), si la distancia atravesando los bordes es más corta que sin hacerlo, entonces face usará la trayectoria más corta. Si el solicitante y el agente se encuentran en exactamente la misma posición, entonces la orientación del solicitante no cambiará.

---

facexy
facexy número número

Da al solicitante una orientación hacia el punto (x, y). Bajo la topología de enlazamiento (toro, cilindros), si la distancia atravesando los bordes es más corta que sin hacerlo, entonces facexy usará esta última trayectoria. Si el solicitante y el agente se encuentran en exactamente la misma posición, entonces la orientación del solicitante no cambiará.

---

file-at-end?
file-at-end?

Reporta true (verdadero) cuando no hay más caracteres que leer en el archivo actual (que fue abierto previamente con file-open). En otro caso reporta false.

```
file-open "mi-archivo.txt"
print file-at-end?
=> false ;; Aún puede leer más caracteres
print file-read-line
=> This is the last line in file (Esta es la última línea del archivo)
print file-at-end?
=> true ;; Llegamos al final del archivo
```

Ver también file-open, file-close-all.

---

file-close
file-close

Cierra un archivo que se ha abierto previamente con file-open. Note que esta primitiva y file-close-all son las únicas maneras de ir al inicio de un archivo abierto o de cambiar entre modos de archivo.

Si no hay ningún archivo bierto nada ocurre.
Ver también file-close-all, file-open.

---

file-close-all
file-close-all

Cierra todos los archivos (caso de haber alguno) que han sido abiertos previamente con file-open.
Ver también file-close, file-open.

---

file-delete
file-delete cadena

Elimina el archivo especificado por la cadena. La cadena debe nombrar un archivo existente con permiso para que el usuario escriba en él. Además, el archivo no debe estar abierto. Use la primitiva file-close antes de borrar el archivo, caso de encontrarse abierto.
Note que la cadena puede ser el nombre de un archivo o la ruta completa del mismo. Si fuera el nombre de un archivo, será buscada en el directorio actual. Esto se puede modificar con la instrucción set-current-directory. La ruta por defecto es el directorio donde se encuentra el modelo.

---

file-exists?
file-exists? cadena

Reporta true (verdadero) si la cadena es el nombre de un archivo existente en el sistema, de otro modo reporta false.
Note que la cadena puede ser el nombre de un archivo con o sin la ruta completa. Si no se incluye la ruta se buscará el archivo en el directorio actual. Esto se puede cambiar con la orden set-current-directory. El directorio por defecto es el directorio del modelo.

---

file-flush
file-flush

Fuerza a las actualizaciones del archivo para que se escriban en el disco. Cuando usted usa file-write u otras órdenes de salida (out commands), los valores pueden no ser escritos inmediatamente en el disco, pues esto mejora el rendimiento de las órdenes de salida. Cuando un archivo es cerrado, todas las órdenes de salida van dirigidas al disco antes de que el archivo se cierre.

Sin embargo, algunas veces es necesario asegurarse de que los datos se escriban en el disco sin tener que cerrar el archivo. Por ejemplo, usted podría estar usando un archivo para comunicarse con otro programa en su máquina y querer que el otro programa pueda ver la salida (output) inmediatamente.

---

file-open
file-open cadena

Esta instrucción interpreta la cadena como el nombre de un archivo (con o sin la ruta completa) y lo abre. Usted puede luego usar las primitivas re-

portadoras file-read, file-read-line y file-read-characters para leer del archivo o file-write, file-print, file-type, file-show para escribir en él.

Note que usted sólo puede escribir un archivo para leer o escribir pero no para ambas cosas. La primitiva file i/o que se usa después de esta orden indica el modo en que el archivo es abierto. Para cambiar de modo es necesario cerrar el archivo usando file-close. El archivo debe existir si se lo desea abrir en modo de lectura.

Cuando se abre un archivo en modo de escritura, cualquier nuevo dato será añadido al final del archivo original. Si no existe un archivo original, un nuevo archivo en blanco será creado (se debe tener permiso de escritura en la carpeta del archivo). Si usted no desea añadir datos sino reemplazar el contenido del archivo existente, use file-delete para borrar primero el contenido, tal vez dentro de una orden carefully, en caso de no estar seguro de que el archivo exista.

Note que la cadena puede ser un nombre con o sin la ruta completa. Si es sólo el nombre, será buscado en el directorio actual. Esto se puede cambiar usando la orden command set-current-directory. La carpeta por defecto es la misma que la del modelo.

```
file-open "mi-archivo-entrada.txt"
print file-read-line
=> First line in file ;; el archivo está en modo de lectura (reading mode)
file-open "C:\\NetLogo\\mi-archivo-salida.txt"
;; suponiendo que su sistema operativo es Windows
file-print "Hola Mundo" ;; el archivo está en modo de escritura
```

Abrir una archivo no cierra los que se han abierto previamente. Usted puede usar file-open para ir de uno a otro entre los varios archivos que se encuentran abiertos. Ver también file-close y file-close-all.

---

file-print

file-print valor

Imprime valor en un archivo abierto, seguido de un retorno de carro (nueva línea)
Contrariamente a file-show, "este agente" (el agente solicitante) no se imprime antes de "valor".
Note que esta orden es el equivalente file i/o de print y se necesita llamar a file-open antes de poder usarla.

Ver también file-show, file-type y file-write.

---

file-read
file-read

Esta primitiva reportadora lee la siguiente constante del archivo abierto y lo interpreta como si hubiese sido escrito en la ventana del Observador (Centro de Mandos). Reporta el valor resultante, el cual puede ser un número, lista, cadena, booleano o el valor especial nobody (nadie).

Las constantes se separan por espacios en blanco. Cada llamada a file-read saltará sobre los espacios blancos antes y depués de la constante.
Note que las cadenas necesitan tener comillas alrededor. Use la orden file-write para incluir comillas.

También note que la orden file-open se debe llamar antes de usar esta primitiva reportadora y tiene que haber datos en el archivo. Use la primitiva reportadora file-at-end? para determinar si usted se encuentra al final del archivo.

```
file-open "mi-archivo.data"
print file-read + 5
;; el valor leído es 1
=> 6
print length file-read
;; el valor leído es la lista [1 2 3 4]
=> 4
```

Ver también file-open y file-write.

---

file-read-characters
file-read-characters número

Reporta la cadena formada por el número dado de caracteres de un archivo abierto. Si hay menos caracteres que los especificados por número, reportará los que hay.

Note que reportará cada caracter, incluyendo cambio de línea y espacios. También note que se debe invocar la orden file-open antes de usar esta primitiva y deben quedar algunos datos en el archivo.

```
file-open "mi-archivo.txt"
print file-read-characters 4
;; La línea actual en el archivo es "Hola Mundo"
=> Hola
```

Ver también file-open.

---

file-read-line
file-read-line

Lee la siguiente línea en el archivo y la reporta como una cadena. Determina el final del archivo por un retorno de carro, un caracter de final de archivo o ambos en una fila. No reporta el caracter de fin de línea.

También note que la orden file-open se debe invocar antes de usar esta primitiva y deben haber quedado datos en el archivo. Use la primitiva file-at-end? para determinar si usted ha llegado al final del archivo.

```
file-open "mi-archivo.txt"
print file-read-line
=> Hola Mundo
```

Ver también file-open.

---

file-show
file-show valor

Imprime la entrada valor a un archivo abierto, precedido por el agente solicitante y seguido de un caracter de retorno de carro. El solicitante se incluye para ayudarle a dar seguimiento acerca de cuáles agentes están produciendo cuáles líneas de texto. Además, todas las cadenas incluyen comillas, similarmente a file-write.

Note que este comando es el equivalente file i/o de show y es necesario invocar file-open antes de poder usarlo.

Ver también file-print, file-type y file-write.

---

file-type
file-type valor

Imprime la entrada valor a un archivo abierto, sin estar seguido de un caracter de retorno de carro (contrariamente a file-print y file-show). La ausencia del retorno de carro permite imprimir varios valores en la misma línea.

El agente solicitante no se imprime antecediendo a valor.

Note que este comando es el equivalente file i/o de type y es necesario invocar file-open antes de usarlo.

Ver también file-print, file-show y file-write.

---

file-write
file-write valor

Este comando envía la salida, que puede ser un número, cadena, lista, booleano o nobody (nadie) a un archivo abierto, sin añadir al final un comando de retorno de carro (contrariamente a file-print y file-show).

El agente solicitante no se imprime antes de valor, contrariamente a file-show. La salida enviada también incluye comillas alrededor de las cadenas y es antecedido de un espacio en blanco. La salida valor enviada podría ser interpretada por file-read.

Note que este comando es el equivalente file i/o de write y que file-open debe ser invocado antes de poder usarlo.

```
file-open "ubicaciones.txt"
ask turtles
  [ file-write xcor file-write ycor ]
```

Ver también file-print, file-show y file-type.

---

filter
filter condición-reportadora lista

Reporta una lista que contiene sólo aquellos miembros de lista de entrada para los cuales la condición-reportadora reporta true (verdadero) – en otras palabras, filter reporta solamente los miembros que satisfacen la condición dada.

```
show filter is-number? [1 "2" 3]
=> [1 3]
show filter [? < 3] [1 3 2]
=> [1 2]
show filter [first ? != "t"] ["salud" "tengan" "ustedes"]
=> ["salud" "ustedes"]
```

Ver también map, reduce y ?.

---

first
first lista
first cadena

En una lista reporta su primer ítem (el 0-ésimo ítem). En una cadena reporta una cadena que consiste en el primer caracter de la cadena original.

---

floor
floor número

Reporta el mayor entero que es menor o igual al número (redondeo al entero menor o igual a número).

```
show floor 4.5
=> 4
show floor -4.5
=> -5
```

Ver también ceiling, round y precision.

---

follow
follow tortuga

Similar a ride pero en la vista 3D, con el punto de observación del observador detrás y encima de la tortuga.

Ver también follow-me, ride, reset-perspective, watch y subject.

---

follow-me
follow-me

Pide al observador que sigue esta tortuga.

Ver también follow.

---

foreach

foreach lista comandos-de-tarea
(foreach lista1 ... comandos-de-tarea)

Con sólo una lista ejecuta la tarea para cada ítem de la lista.

```
foreach [1.1 2.2 2.6] show
=> 1.1
=> 2.2
=> 2.6
foreach [1.1 2.2 2.6] [ show (word ? " -> " round ?) ]
=> 1.1 -> 1
=> 2.2 -> 2
=> 2.6 -> 3
```

Con múltiples lista ejecuta los comandos para cada grupo de ítemes de cada lista. De modo que se ejecutan una vez para el primer ítem, una vez para el segundo y así sucesivamente. Todas las listas deben tener la misma longitud.

Algunos ejemplos ayudarán a aclarar esto:

```
(foreach [1 2 3] [2 4 6]
   [ show word "la suma es: " (?1 + ?2) ])
=> "la suma es: 3"
=> "la suma es: 6"
=> "la suma es: 9"
(foreach list (turtle 1) (turtle 2) [3 4]
  [ ask ?1 [ fd ?2 ] ])
;; la tortuga 1 se mueve 3 pasos adelante
;; la tortuga 2 se mueve 4 pasos adelante
```

Ver también map y ?

---

forward
fd
forward número

La tortuga se mueve hacia adelante la cantidad de pasos indicada por número, un paso a la vez. Si número es negativo, la tortuga se mueve hacia atrás.

fd 10 es equivalente a repeat 10 [jump 1]. fd 10.5 es equivalente a repeat 10 [jump 1] jump 0.5.

Si la tortuga no se puede mover hacia adelante el número de pasos indicado, porque la topología vigente no se lo permite, la tortuga completará tantos pasos de 1 unidad como sea posible y luego se detendrá.
Ver también jump y can-move?.

---

fput
fput ítem lista

Añade ítem al comienzo de lista y reporta la nueva lista ampliada.

```
;; suponga que milista es [5 7 10]
set milista fput 2 milista
;; milista ahora es [2 5 7 10]
```

G

---

globals
globals[variable1 variable2...]

Esta primitiva, lo mismo que las primitivas breed, <breeds>-own, patches-own y turtles-own, sólo se puede usar al comienzo del código de un programa, antes de definir ningún procedimiento. Se usa para definir variables globales.

Las variables globales son "globales" por ser accesibles a todos los agentes y se pueden usar en cualquier parte de un modelo. La mayoría de las veces las variables globales se usan para definir variables o constantes que es necesario usar en muchas partes del programa.

H

---

hatch
hatch-<breeds>
hatch número [**comandos**]
hatch <**breeds**> **número** [**comandos**]

Esta tortuga crea una cantidad dada por número de nuevas tortugas. Cada nueva tortuga hereda de su progenitora todas sus variables, incluyendo su posición. Excepciones: cada nueva tortuga tiene un nuevo número who y podría pertenecer a otra familia cuando se usa la forma $hatch- < breed >$. Las nuevas tortugas ejecutan los comandos entre corchetes. Se pueden usar los comandos para dar a las nuevas tortugas colores, orientaciones o ubicaciones nuevas o lo que fuere. Las nuevas tortugas son creadas de una sola vez y luego ejecutan los comandos una por una en orden aleatorio.

Si se usa la forma $hatch- < breeds >$, las tortugas son creadas como miembros de la familia dada. De lo contrario las nuevas tortugas pertenecerán a la misma familia que su progenitora.

```
hatch 1 [ lt 45 fd 1 ]
;; esta tortuga crea una nueva tortuga,
;; y la hija gira y se aleja
hatch-oveja 1 [ set color black ]
;; esta tortuga crea una nueva tortuga
;; de la familia oveja
```

Ver también create-turtles y sprout.

---

heading
heading

Esta es una variable preinstalada en el sistema. Indica la dirección en la que apunta la tortuga. Reporta un número mayor o igual a 0 y menor a 360. 0 es norte, 90 este y así sucesivamente. Usted puede reasignar el valor de esta variable para cambiar la orientación de la tortuga.

Ver también right, left, dx, dy

Ejemplo:

```
set heading 45   ;; la tortuga ahora apunta en dirección noreste
set heading heading + 10 ;; produce el mismo efecto que "rt 10"
```

---

hidden?
hidden?

Es una variable preinstalada en el sistema que pertenece a las tortugas o las ligas. Almacena un valor booleano (verdadero o falso) indicando si la tortuga o liga se encuentra oculta (es decir invisible). Usted puede reasignar el valor de esta variable para hacer que una tortuga o liga desaparezca o reaparezca.
Ver también hide-turtle, show-turtle, hide-link, show-link.

Ejemplo:

```
set hidden? not hidden?
;; si la tortuga está visible se oculta, si estaba oculta reaparece.
```

---

hide-link
hide-link

La ligas de vuelve invisible.

Nota: Este comando es equivalente a asignarle a la variable de liga “hiden?” el valor true (verddero).

Ver también show-link.

---

hide-turtle
hide-turtle

La tortuga se vuelve invisible.

Nota: Este comando es equivalente a asignar a la variable de tortuga “hidden?” el valor de true (verdadero).

Ver también show-turtle.

---

histogram
histogram lista

Crea un histograma con los valores en la lista dada.

Dibuja un histograma que muestra la distribución de frecuencias de los valores de la lista. La altura de las barras en el histograma representa el número de valores en cada subrango.

Antes de que el histograma sea dibujado, se remueven todos los puntos previamente dibujados por la pluma actual.

Se ignoran los valores no numéricos.

El histograma se dibuja en el gráfico actual, usando la pluma actual y su color. El escalado automático no afecta el rango horizontal del histograma, de modo que set-plot–x-range se debe usar para controlar el rango y el intervalo de la pluma se puede asignar (ya sea directamente con set-plot-pen-interval, o indirectamente vía set-histogram-num-bars) para controlar en cuántas barras se debe dividir el rango.

Si desea que el histograma se dibuje con barras, asegúrese que la pluma actual se encuentra en modo de barras (bar mode, mode 1).

Cuando se crear un histogramas no se considera incluido el valor máximo de X del gráfico. Los valores iguales al máximo X quedan fuera del rango del histograma.

```
histogram [color] of turtles
;; dibuja un histograma mostrando cuántas tortuga hay
;; de cada color
```

---

home
home

La tortuga solicitante se traslada al origen (0, 0). Es equivalente a setxy 0 0.

---

hsb
hsb saturación brillantez

Reporta una lista RGB cuando se dan tres números describiendo un color RGB. La saturación Hue y la brillantez son enteros en el rango 0-255. La lista RGB contiene tres enteros en el mismo rango.

Ver también rgb.

---

hubnet-broadcast
hubnet-broadcast nombre-etiqueta valor

Difunde la entrada valor desde NetLogo al elemento de la interfaz del cliente con nombre nombre-etiqueta.

Ver la HubNet Authoring Guide para detalles e instrucciones.

---

hubnet-broadcast-clear-output
hubnet-broadcast-clear-output

Limpia todos los mensajes impresos en el área de texto de todo cliente.

Ver también: hubnet-broadcast-message y hubnet-send-clear-output

---

hubnet-broadcast-message
hubnet-broadcast-message valor

Imprime la entrada valor en el área de texto de cada cliente. Tiene la misma funcionalidad que el botón en el Centro de Control de HubNet.

Ver también: hubnet-send-message

---

hubnet-clear-override
hubnet-clear-overrides
hubnet-clear-override cliente agente-o-conjunto nombre-de-variable
hubnet-clear-overrides cliente

Elimina overrides (invalidaciones) de la lista de invalidaciones en el cliente. hubnet-clear-override elimina sólo la invalidación para la variable especificada para el cliente o conjunto-agentes especificado. hubnet-clear-overrides elimina todas las invalidaciones del cliente especificado.

Ver también: hubnet-send-override

---

hubnet-clients-list
hubnet-clients-list

Reporta una lista que contiene los nombre de todos los clientes actualmente conectados al servidor HubNet.

---

hubnet-enter-message?
hubnet-enter-message?

Reporta true (verdadero) si un nuevo cliente acaba de entrar a la simulación y falso en caso contrario. hubnet-message-source contendría el nombre de usuario del cliente que se acaba de conectar.

Ver la HubNet Authoring Guide para detalles e instrucciones.

---

hubnet-exit-message?

hubnet-exit-message?

Reporta true (verdadero) si un cliente acaba de salir de la simulación, falso en otro caso. hubnet-message-source contendría el nombre de usuario del cliente que se acaba de desconectar.

Ver la HubNet Authoring Guide para detalles e instrucciones.

---

hubnet-fetch-message
hubnet-fetch-message

Si hay algún nuevo dato enviado por los clientes, se obtiene el nuevo trozo de información, de modo que puede ser accesado por hubnet-message, hubnet-message-source y hubnet-message-tag. Se produce un error si no hay nuevos datos de parte de los clientes.

Ver también la HubNet Authoring Guide para más detalles.

---

hubnet-kick-client
hubnet-kick-client nombre-de-cliente

Expulsa (patea) al cliente con el nombre nombre-de-cliente. Equivale a expulsar (patear en sentido figurado) el nombre del cliente en el Centro de Control de HubNet y oprimir el botón Kick (patear).

---

hubnet-kick-all-clients
hubnet-kick-all-clients

Expulsa a todos los clientes conectados a HubNet en ese momento. Equivale a seleccionar a todos los clientes en el Centro de Control de HubNet y oprimir el botón Kick (patear o expulsar).

---

hubnet-message
hubnet-message

Reporta el mensaje recuperado por hubnet-fetch-message.

Ver la HubNet Authoring Guide para más detalles.

---

hubnet-message-source
hubnet-message-source

Reporta el nombre del cliente que envió el mensaje recuperado por hubnet-fetch-message.

Ver la HubNet Authoring Guide para más detalles.

---

hubnet-message-tag
hubnet-message-tag

Reporta la etiqueta asociada con los datos recuperados por hubnet-fetch-message. La etiqueta sería una de los Display Names (Desplegar Nombres) de los elementos de la interfaz en la interfaz del cliente.

Ver la HubNet Authoring Guide para más detalles.

---

hubnet-message-waiting?
hubnet-message-waiting?

Busca un nuevo mensaje enviado por los clientes. Reporta verdadero si hay alguno y falso si no lo hay.

Ver la HubNet Authoring Guide para más detalles.

---

hubnet-reset
hubnet-reset

Reinicia el sistema HubNet. HubNet debe ser iniciado para poder usar cualquier otra primitiva de hubnet, con la excepción de hubnet-set-client-interface.

Ver la HubNet Authoring Guide para más detalles.

---

hubnet-reset-perspective
hubnet-reset-perspective nombre-de-etiqueta

Borra watch o follow enviado directamente al cliente. La perspectiva de la vista se revierte a la perspectiva del servidor.

Ver también: hubnet-send-watch hubnet-send-follow

---

hubnet-send
hubnet-send lista-de-cadenas nombre-etiqueta valor

Para una cadena envía valor desde NetLogo a la etiqueta nombre-etiqueta del cliente que tiene la cadena por nombre de usuario.

Para una lista-de-cadenas envía valor desde NetLogo a la etiqueta nombre-etiqueta de todos los clientes que tienen un nombre de usuario contenido en la lista-de-cadenas.

Enviar un mensaje a un cliente no existente usando hubnet-send genera un hubnet-exit-message (mensaje-de-salida-de-hubnet).

Ver la HubNet Authoring Guide para más detalles.

---

hubnet-send-clear-output

hubnet-send-clear-output cadena
hubnet-send-clear-output lista-de-cadenas

Limpia todos los mensajes impresos en el área de texto del cliente o clientes dados (especificados en la cadena o lista-de-cadenas).

Ver también: hubnet-send-message, hubnet-broadcast-clear-output

---

hubnet-send-follow

hubnet-send-follow nombre-de-cliente agente radio

Le dice al cliente asociado con nombre-de-cliente que siga al agente mostrando un vecindario Moore de tamaño dado por radio alrededor del agente.

Ver también: hubnet-send-watch, hubnet-reset-perspective

---

hubnet-send-message

hubnet-send-message cadena valor

Imprime valor en el área de texto del cliente especificado por cadena.

Ver también: hubnet-broadcast-message

---

hubnet-send-override

hubnet-send-override nombre-de-cliente agente-o-conjunto nombre-de-variable [reportadora]

Evalúa reportadora para el agente o conjunto-agentes indicado y luego envía los valores al cliente para "invalidar" ("override") el valor de nombre-de-variable sólo en nombre-de-cliente. Esto se usa para cambiar la apariencia de agentes en la vista del cliente, por tanto sólo se deben escoger variables preinstaladas del sistema que afectan la apariencia del agente. Por ejemplo, usted puede invalidar la variable color de una tortuga:

```
ask turtles [hubnet-send-override nombre-de-cliente self "color" [red]]
```

En este ejemplo suponga que hay una nombre-de-cliente que es una variable de tipo turtles-own, que está asociado a un cliente que está conectado y todas las tortugas son azules. Esta orden hace que la tortuga asociada con cada cliente aparezca en color rojo en su propia vista pero no en la de nadie más o en la del servidor.

Ver también: hubnet-clear-overrides

---

hubnet-send-watch

hubnet-send-watch nombre-de-cliente agente

Le dice al cliente asociado con nombre-de-cliente que observe al agente.

Ver también: hubnet-send-follow, hubnet-reset-perspective

---

hubnet-set-client-interface

hubnet-set-client-interface tipo-cliente info-cliente

Si tipo-cliente es "COMPUTER", se ignora info-cliente.

```
hubnet-set-client-interface "COMPUTER" []
```

Futuras versiones de HubNet soportarán otres tipos de clientes. Aún para computadoras clientes, el significado de la segunda entrada de este comando

podría cambiar.

Ver la HubNet Authoring Guide para más detalles.

I

---

if

if condición-reportadora [**comandos**]

Si la condición-reportadora reporta true (verdadero) se ejecutan los comandos.

La condición podría reportar distintos valores para agentes diferentes, de modo que algunos agentes podría ejecutar los comandos y otros no.

```
if xcor > 0[ set color blue ]
;; las tortugas en la mitad derecha del mundo
;; se vuelven azules
```

Ver también ifelse, ifelse-value.

---

ifelse

ifelse condición-reportadora [**reporter1**] [**reporter2**]

La condición-reportadora debe reportar un valor booleano verdadero o falso (true o false).

Si la condición reporta true, se ejecutan los comandos1, si reporta false se ejecutan los comandos2.

La condición podría reportar distintos valores para diferentes agentes, de mo-

do que algunos agentes podrian ejecutar comandos1 y otros comandos2.

```
ask patches
  [ ifelse pxcor > 0
      [ set pcolor blue ]
      [ set pcolor red ] ]
;; la mitad izquierda del mundo se vuelve roja y
;; la mitad derecha se vuelve azul
```

Ver también if, ifelse-value.

---

ifelse-value

ifelse-value condición-reportadora [**reportadora1**] [**reportadora2**]

La condición-reportadora debe reportar un valor booleano veradero o falso (true o false).

Si la condición reporta true, el resultado reportado es el valor de reportadora1. Si reporta false el valor reportado es reportadora2.

Esta primitiva se puede usar cuando se necesita una expresión condicional en el contexto de una reportadora, donde comandos (tales como ifelse) no son permitidos.

```
ask patches [
  set pcolor ifelse-value (pxcor > 0) [blue] [red]
]
;; la mitad izquierda del mundo se vuelve roja y
;; la mitad derecha se vuelve azul
show n-values 10 [ifelse-value (? < 5) [0] [1]]
=> [0 0 0 0 0 1 1 1 1 1]
show reduce [ifelse-value (?1 > ?2) [?1] [?2]]
  [1 3 2 5 3 8 3 2 1]
=> 8
```

Ver también if, ifelse.

---

import-drawing

import-drawing nombre-de-archivo

Lee un archivo de imagen dentro del dibujo, a escala con el tamaño del mundo y manteniendo el aspecto original en cuanto a las proporciones de la imagen. La imagen es centrada en el dibujo. El dibujo anterior no es previamente borrado.

Los agentes no pueden percibir el dibujo, de modo que estos no pueden interactuar o procesar las imágenes importadas con import-drawing. Si usted necesita que los agentes perciban una imagen, use import-pcolors o import-pcolors-rgb.

Los siguientes formatos de imagen son soportados: BMP, JPG, GIF y PNG. Si el formato de imagen soporta la transparencia (alpha), esta información será también importada.

---

import-pcolors

import-pcolors nombre-de-archivo

Lee un archivo de imagen, a escala con las mismas dimensiones que el entramado de parcelas, manteniendo el aspecto original de la imagen en cuanto a las proporciones y transfiere los colores de los pixeles resultantes a las parcelas. La imagen es centrada en el entramado de parcelas.

Los colores resultantes de las parcelas podrían verse distorsionados, pues el espacio de colores de NetLogo no incluye todos los posibles colores (ver la sección de colores de la Guía de Programación). La primitiva import-pcolors podría ser lenta para algunas imágenes, particularmente si usted tiene muchas parcelas y una imagen grande con muchos colores diferentes.

Puesto que import-colors fija el pcolor de las parcelas, los agentes pueden percibir la imagen. Esto es útil si los agentes necesitan analizar, procesar o interactuar con la imagen. Si usted simplemente desea desplegar un fondo estático, sin distorsión de colores, vea import-drawing.

Los siguientes formatos de archivos de imagen son soportados: BMP, JPG, GIF y PNG. Si el formato de imagen soporta la transparencia (alpha), entonces todos los pixeles totalmente transparentes serán ignorados (los pixeles parcialmente transparentes serán tratados como si fueran opacos).

---

import-pcolors-rgb

import-pcolors-rgb nombre-de-archivo

Lee un arvchivo de imagen y a escala con las mismas dimensiones del entramado de parcelas, manteniendo el aspecto original en cuanto a las proporciones de la imagen y transfiere los colores resultantes de los pixeles a las parcelas. La imagen es centrada en el entramada de parcelas. Contrariamente a import-pcolors, se mantienen los colores exactos de la imagen original. La variable pcolor de todas las parcelas será una lista RGB en vez de un color NetLogo (aproximado).

Los siguientes formatos de archivos de imagen son soportados: BMP, JPG, GIF y PNG. Si el formato de imagen soporta la transparencia (alpha), entonces todos los pixeles totalmente transparentes serán ignorados (los pixeles parcialmente transparentes serán tratados como si fueran opacos).

---

import-world

import-world nombre-de-archivo

Lee los valores de todas las variables de un modelo, tanto las preinstaladas en el sistema como las definidas por el usuario, incluyendo las del observador, tortugas y parcelas, de un archivo externo nombrado por la cadena dada. El archivo debe tener el formato usado por la primitiva export-world.

Note que la funcionalidad de esta primitiva se encuentra disponible directamente desde el menú archivo de NetLogo.

Cuando use import-world, para evitar errores, realice los siguientes pasos en el orden indicado:

1. Abra el modelo con el que usted creó el archivo de exportación.
2. Oprima el botón Setup, para tener el modelo en un estado en que se pueda correr.
3. Importe el archivo.
4. Vuelva a abrir cualesquiera archivos que el modelo hubiese abierto con el comando file-open
5. Si lo desea, oprima el botón Go para continuar corriendo el modelo desde el punto en que lo dejó.

Si desea importar un archivo desde una ubicación distinta a donde se encuentra el modelo, debe incluir la ruta completa del archivo que desea importar. Ver export-world para un ejemplo.

---

in-cone

conjunto-agentes in-cone distancia ángulo

Esta reportadora permite dar a una tortuga un "cono de visión" en frente de ella. El cono está definido por dos entradas, la distancia de visión (radio) y el ángulo de visión. El ángulo de visión puede variar de 0 a 360 y se mide a partir de la orientación actual de la tortuga. Si el ángulo es 360 in-cone es equivalente a in-radius.

in-cone reporta un conjunto-agentes que incluye sólo aquellos agentes del conjunto-agentes original que se encuentran dentro del cono (esto puede incluir al agente mismo). La distancia a una parcela se mide desde el centro de la parcela.

```
ask turtles
  [ ask patches in-cone 3 60
      [ set pcolor red ] ]
;; cada tortuga produce delante de ella una mancha
;; roja de parcelas en forma de cono con
;; ángulo de 60 grados y radio 3
```

---

in-<breed>-neighbor?
in-link-neighbor?

in-<breed>-neighbor? agente
in-link-neighbor? tortuga

Reporta true (verdadero) si hay una liga dirigida de la tortuga al agente solicitante.

```
crt 2
ask turtle 0 [
  create-link-to turtle 1
  show in-link-neighbor? turtle 1  ;; imprime false
  show out-link-neighbor? turtle 1 ;; imprimer true (verdadero)
]
ask turtle 1 [
  show in-link-neighbor? turtle 0  ;; imprime true
  show out-link-neighbor? turtle 0 ;; imprime false
]
```

---

in-<breed>-neighbors
in-link-neighbors

in-<breed>-neighbors
in-link-neighbors

Reporta el conjunto-agentes de todas las tortugas que tienen ligas dirigidas cuyo origen es el agente solicitante.

```
crt 4
ask turtle 0 [ create-links-to other turtles ]
ask turtle 1 [ ask in-link-neighbors
 [ set color blue ] ] ;; la tortuga 0 se vuelve azul
```

---

in-<breed>-from
in-link-from

in-<breed>-from turtle
in-link-from turtle

Reporta la liga dirigida que va de la tortuga al agente solicitante. Si no existen ligas reporta nobody (nadie).

```
crt 2
ask turtle 0 [ create-link-to turtle 1 ]
ask turtle 1 [ show in-link-from turtle 0 ] ;; muestra la liga 0 1
ask turtle 0 [ show in-link-from turtle 1 ] ;; muestra nobody
```

Ver también: out-link-to link-with

_includes
_includes [**nombre-de-archivo ...**]

Hace que los archivos NetLogo externos (con la extensión .nls) sean incluidos en el modelo. Los archivos incluidos pueden tener definiciones de familias (breeds), variables y procedimientos. La primitiva _includes sólo se puede usar una vez por archivo.

---

in-radius
agentset in-radius número

Reporta un conjunto-agentes que incluye sólo aquellos agentes del conjunto-agentes original cuya distancia del agente solicitante es menor o igual a número (eventualmente puede incluir al agente solicitante).

La distancia hacia o desde una parcela se mide desde el centro de la parcela.

```
ask turtles
  [ ask patches in-radius 3
      [ set pcolor red ] ]
;; cada tortuga produce una mancha roja a su alrededor
```

---

inspect
inspect agente

Abre un monitor para el agente dado (tortuga o parcela).

```
inspect patch 2 4
;; un monitor de agentes se abre para esa parcela
inspect one-of ovejas
;; un monitor de agentes se abre para una tortuga
;; escogida al azar de la familia "ovejas"
```

---

int
int número

Reporta la parte entera de número – la parte decimal o fraccionaria del número se desecha.

```
show int 4.7
=> 4
show int -3.5
=> -3
```

---

is-agent?
is-agentset?
is-boolean?
is-<breed>?
is-command-task?
is-directed-link?
is-link?
is-link-set?
is-list?
is-number?
is-patch?
is-patch-set?
is-reporter-task?
is-string?
is-turtle?
is-turtle-set?
is-undirected-link?

is-agent? valor
is-agentset? valor

is-boolean? valor
is-<breed>? valor
is-command-task? valor
is-directed-link? valor
is-link? valor
is-link-set? valor
is-list? valor
is-number? valor
is-patch? valor
is-patch-set? valor
is-reporter-task? valor
is-string? valor
is-turtle? valor
is-turtle-set? valor
is-undirected-link? valor

Reporta true (verdadero) si valor es del tipo dado, falso en caso contrario.

---

item

item índice lista
item índice cadena

En una lista reporta el valor del ítem de la lista que ocupa la posición dada por el índice.

En una cadenas reporta el caracter de la cadena que ocupa la posición dada por el índice.

Los índices comienzan en 0, no en 1 (el primer ítem es el ítem 0, el segundo ítem es el ítem 1 y así sucesivamente).

```
;; suponga que milista es [2 4 6 8 10]
show item 2 milista
=> 6
show item 3 "mi-zapato"
```

```
=> "z"
```

J

---

jump
jump número

La tortuga se mueve hacia adelante y de una sola vez una candidad de pasos dada por número (en vez de hacerlo un paso a la vez como con el comando forward).

Si la topología vigente del mundo le impide avanzar ese número de pasos entonces la tortuga no se mueve del todo.

Ver también forward y can-move?

L

---

label
label

Esta es una variable preinstalada de las tortugas o las ligas. Puede almacenar un valor de cualquier tipo. La totuga o liga aparece en la vista con el valor dado "adherido" a ella como texto. Se puede usar esta variable para agregar, cambiar o eliminar la etiqueta (label) de una tortuga o liga.

Ver también label-color y plabel, plabel-color.

```
Ejemplo:
```

```
ask turtles [ set label who ]
;; todas las tortugas están ahora etiquetadas
;; con sus número who
ask turtles [ set label "" ]
;; las tortugas ahora no están etiquetadas
```

---

label-color
label-color

Esta es una variable preinstalada de las tortugas o las ligas. Almacena un número mayor o igual a 0 y menor que 140. Este número determina en qué color aparece la etiqueta de la tortuga o liga (caso de tener una etiqueta). Se puede asignar el valor de esta variable para cambiar el color de la etiqueta de una tortuga o liga.

Ver también label, plabel, plabel-color.

Example:

```
ask turtles [ set label-color red ]
;; todas las tortugas tienen ahora etiquetas rojas
```

---

last
last lista
last cadena

En una lista reporta el último item de la lista.
En una cadena reporta una cadena formada por el último caracter de la cadena original.

---

layout-circle
layout-circle conjunto-agentes radio

layout-circle lista-de-tortugas radio

Despliega las tortugas dadas en un círculo centrado en el centro del mundo con el radio dado (si el mundo tiene un radio de tamaño impar el centro del círculo se redondea a la parcela más cercana). Las tortugas apuntan hacia afuera.

Si la primera entrada es un conjunto-agentes, las tortugas se disponen en orden aleatorio.
Si la primera entrada es una lista, las tortugas se disponen en sentido horario en el orden dado, comenzando en la parte superior del círculo (cualquier cosa que no sea una tortuga en la lista será ignorada).

```
;; en orden aleatorio
layout-circle turtles 10
;; en orden por número de who
layout-circle sort turtles 10
;; en orden de tamaño
layout-circle sort-by [[size] of ?1 < [size] of ?2] turtles 10
```

---

layout-radial

layout-radial conjunto-tortugas conjunto-ligas agente-raíz

Despliega las tortugas de conjunto-tortugas conectadas por las ligas de conjunto-ligas en forma de árbol radial, centrado alrededor del agente-raíz, el cual es movido al centro de la vista del mundo.

Sólo las ligas del conjunto-ligas se usarán para determinar la forma de disponer las tortugas. Si hay ligas conectadas con tortugas que no pertenecen a conjunto-tortugas, estas tortugas permanecen en su lugar.

Aún cuando la red contenga ciclos y no posea una verdadera estructura de árbol, esta dispoisicón aún funcionaría, aunque el resultado no será siempre lindo.

```
to hacer-un-árbol
  set-default-shape turtles "circle"
  crt 6
  ask turtle 0 [
    create-link-with turtle 1
    create-link-with turtle 2
    create-link-with turtle 3
  ]
  ask turtle 1 [
    create-link-with turtle 4
    create-link-with turtle 5
  ]
  ; disponerse en árbol radial, centrado an la tortuga 0
  layout-radial turtles links (turtle 0)
end
```

---

layout-spring

layout-spring conjunto-tortugas conjunto-ligas constante-resorte longitud-resorte constante-repulsión

Dispone las tortugas de conjunto-tortugas como si las ligas de conjunto-ligas fueran resortes y las tortugas se repelieran unas a otras. Aquellas tortugas no incluidas en conjunto-tortugas pero conectadas con ligas de conjunto-ligas son tratadas como anclas y no se mueven.
spring-constant es una medida de la "tirantez" del resorte. Es la "resistencia" a cambiar su longitud. spring-constant es la fuerza que ejercería el resorte si su longitud fuera cambiada en 1 unidad.
spring-length es la longitud de "fuerza-cero" o longitud natural de los resortes. Es la longitud que todos los resortes tratan de alcanzar ya sea tirando hacia adentro o repeliendo hacia afuera los nodos.
repulsion-constant es una medida de la repulsión entre los nodos. Es la fuerza repulsiva que actúa sobre 2 nodos situados a 1 unidad de distancia el uno del otro.

El efecto de repulsión trata de llevar los nodos lo más lejos posible los unos de los otros, a fin de evitar el apiñamiento y el efecto de los resortes es tratar

de mantenerlos a “una cierta” distancia de los nodos a los que están conectados. El resultado es una disposición extendida de la red en un modo que destaca las relaciones entre los nodos y a la vez es menos congestionada y más agradable a la vista.

El algoritmo de disposición está basado en el algoritmo de disposición de Fruchterman-Reingold. Más información sobre este algoritmo se puede obtener en http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.13.8444

```
to hacer-un-triángulo
  set-default-shape turtles "circle"
  crt 3
  ask turtle 0
  [
    create-links-with other turtles
  ]
  ask turtle 1
  [
    create-link-with turtle 2
  ]
  repeat 30 [ layout-spring turtles links 0.2 5 1 ]
  ;; dispone los nodos en triángulo
end
```

---

layout-tutte

layout-tutte conjunto-tortugas conjunto-ligas radio

Las tortugas no incluidas en conjunto-tortugas pero conectadas por ligas de conjunto-ligas son colocadas en círculo con el radio dado. Debe haber al menos 3 tortugas en este conjunto de agentes.

Las tortugas en conjunto-tortugas se disponen del siguiente modo: cada tortuga es colocada en el centroide (o baricentro) del polígono formado por su vecinas con que está conectada. El centroide es como un promedio bidimensional de las coordenadas de los vecinos.

El propósito del círculo de "agentes anclados" es prevenir que todas las tortugas colapsen en un punto.

Después de algunas iteraciones, la disposición se estabiliza.

Esta disposición toma su nombre del matemático William Thomas Tutte, quien la propuso como un método de disposición gráfica.

```
to hacer-un-árbol
  set-default-shape turtles "circle"
  crt 6
  ask turtle 0 [
    create-link-with turtle 1
    create-link-with turtle 2
    create-link-with turtle 3
  ]
  ask turtle 1 [
    create-link-with turtle 4
    create-link-with turtle 5
  ]
  ; colocar todas las tortugas con sólo un
  ; vecino en el perímetro de un círculo
  ; y luego colocar las tortugas restantes dentro
  ; de este círculo, esparcidas entre sus vecinos.
  repeat 10 [ layout-tutte (turtles with [link-neighbors = 1]) links 12 ]
end
```

---

left
lt
left número

La tortuga gira a la izquierda sobre su propio eje la cantidad de grados dada por número. Si número es negativo gira a la derecha.

---

length

length lista
length cadena

Reporta el número de ítemes en la lista dada por el número de caracteres en la cadena dada.

---

let

let nombre valor

Crea una nueva variable local llamada nombre y le asigna el valor dado. Una variable local existe sólo dentro del bloque de comandos en que fue creada.

Si desea luego cambiar el valor de la variable, use set.

Ejemplo

```
let presa one-of ovejas-here
if presa != nobody
  [ ask presa [ die ] ]
```

:

---

link
<breed>

link extremo1 extremo2
< *breed* > **extremo1 extremo2**

Dados los dos número who de los puntos extremos, reporta la liga que conecta las tortugas. Si no existe esa liga reporta nobody (nadie). Para referirse a familias de ligas se debe usar la forma singular con los puntos extremos.

```
ask link 0 1 [ set color green ]
;; la liga no dirigida que conecta la tortuga 0 con la tortuga 1
;; se vuelve verde
```

```
ask directed-link 0 1 [ set color red ]
;; la liga dirigida que conecta la tortuga 0 con la tortuga 1
;; se vuelve roja\\
```

Ver también patch-at

---

link-heading
link-heading

Reporta la orientación en grados (al menos 0, menos de 360) de extremo1 a extremo2 de la liga. Muestra un error de ejecución (runtime error) si ambos extremos tienen la misma ubicación.

```
ask link 0 1 [ print link-heading ]
;; imprime [[towards other-end] of end1] of link 0 1
```

Ver también link-length.

---

link-length
link-length

Reporta la distancia entre los extremos de la liga.

```
ask link 0 1 [ print link-length ]
;; imprime [[distance other-end] of end1] of link 0 1
```

Ver también link-heading

---

link-set
link-set valor1 valor2 ...

Reporta un conjunto-agentes que contiene todas las ligas en cualquier lugar de las entradas. Las entradas pueden ser ligas individuales, ligas de conjuntos-agente, nadie (nobody) o listas (o listas encajadas) que contengan cualquiera de los anteriores.

```
link-set self
```

```
link-set [my-links] of nodes with [color = red]
```

Ver también turtle-set, patch-set.

---

link-shapes
link-shapes

Reporta una lista de cadenas con todas las figuras de ligas en el modelo.

Se pueden crear o importar ligas de otros modelos del Editor de Figuras de Ligas (Link Shapes Editor).

```
show link-shapes
=> ["default"]
```

---

links
links

Reporta el conjunto-agentes que consiste de todas las ligas.

```
show count links
;; imprime el número de ligas
```

---

links-own
$< link - breeds >$**-own**

links-own [**var1 ...**]
$< link - breeds >$**-own** [**var1 ...**]

La primitiva links-own, como las globales, breed, $< breeds > -own$, turtles-own, y patches-own, sólo puede se usada al inicio de un programa, antes de la definición de los procedimientos. Esta primitiva define las variables pertenecientes a cada liga.

Si usted especifica una familia en vez de "links", sólo las ligas de esta familia tendrán las variables en la lista (más de una familia de ligas podría tener la misma variable en la lista).

```
undirected-link-breed [aceras acera]
directed-link-breed [calles calle]
links-own [tráfico]   ;; se aplica a todas las familias
aceras-own [peatones]
calles-own [autos bicicletas]
```

---

list

list valor1 valor2
(list valor1 ...)

Reporta una lista que contiene los ítemes dados, los cuales pueden ser de cualquier tipo, producidos por cualquier clase de reportadora.

```
show list (random 10) (random 10)
=> [4 9]  ;; o por azar lista similar
show (list 5)
=> [5]
show (list (random 10) 1 2 3 (random 10))
=> [4 1 2 3 9]  ;; o por azar lista similar
```

---

ln

ln número

Reporta el logartimo natural, esto es, el logaritmo de base e (2.71828...).

Ver tambiéne, log.

---

log

log número base

Reporta el logaritmo de número en la base dada.

```
show log 64 2
=> 6
```

Ver también ln.

---

loop
loop [órdenes]

Corre la lista de órdenes continuamente (por siempre) o hasta salir del procedimiento actual mediante el uso de las órdenes stop o report.

Nota: En la mayoría de las circunstancias usted debe usar un botón "continuamente" para que algo se repita continuamente (para siempre). La ventaja de esto último es que usted puede hacer clic sobre el botón "continuamente" para detener el bucle de repeticiones.

---

lput
lput valor lista

Se agrega el ítem valor en el último lugar a la lista dada y se reporta la lista ampliada.

```
;; suponga que milista es [2 7 10 "Pepe"]
set milista lput 42 milista
;; milista es ahora [2 7 10 "Pepe" 42]
```

M

---

map
map reportadora-tarea lista1 ...
(map reportadora-tarea lista1 ...)

Con una sola lista, la tarea dada es ejecutada para cada ítem de la lista. Se reporta una lista con los resultados.

```
show map round [1.1 2.2 2.7]
=> [1 2 3]
show map [? * ?] [1 2 3]
=> [1 4 9]
```

Con múltiples listas, la reportadora dada se ejecuta para cada grupo de ítemes de cada lista. Es decir, se ejecuta una vez para los primeros ítemes, una vez para los segundos ítemes y así sucesivamente. Todas las listas deben tener la misma longitud.

Algunos ejemplos podrían dejar esto más claro:

```
show (map + [1 2 3] [2 4 6])
=> [3 6 9]
show (map [?1 + ?2 = ?3] [1 2 3] [2 4 6] [3 5 9])
=> [true false true]
```

Ver también foreach y ?.

---

max
max lista

Reporta el número con el valor máximo de la lista dada. Ignora otros tipos de ítemes.

```
show max [xcor] of turtles
;; imprime la coordenada x de la tortuga más a la
;; derecha del mundo
show max list a b
;; imprimer la mayor de las dos variables a y b
show max (list a b c)
;; imprime la mayor de la tres variables a, b y c
```

---

max-n-of

max-n-of número conjunto-agentes [reportadora]

Reporta un conjunto-agentes que contiene el número dado de agentes del conjunto-agentes con los valores más altos de la reportadora. El conjunto-agentes se construye encontrando todos los agentes con el valor más alto de la reportadora, si no hay ese número de agentes con ese valor se toman agentes con el segundo valor más alto y así sucesivamente. Al final, si hay una atadura que haría el conjunto-agentes resultante demasiado grande, la atadura es rota al azar.

```
;; suponga que el mundo es 11 x 11
show max-n-of 5 patches [pxcor]
;; muestra 5 parcelas con pxcor = max-pxcor
show max-n-of 5 patches with [pycor = 0] [pxcor]
;; muestra un conjunto-agentes formado por:
;; (patch 1 0) (patch 2 0) (patch 3 0) (patch 4 0) (patch 5 0)
```

Ver también max-one-of y with-max.

---

max-one-of

max-one-of conjunto-agentes [reportadora]

Reporta el agente en el conjunto-agentes que tiene el valor más alto para la reportadora dada. Si hay una atadura este comando reporta un agente escogido al azar con el valor más alto. Si usted quiere a todos los agentes, use entonces with-max.

```
show max-one-of patches [count turtles-here]
;; imprime una parcela con la mayor cantidad de
;; tortugas en ella
```

Ver también max-n-of, with-max.

---

max-pxcor
max-pycor

max-pxcor
max-pycor

Estas reportadoras dan los valores máximos de las coordenadas x y y de las parcelas, lo cual determina el tamaño del mundo.

Contrariamente a versiones más antiguas de NetLogo, el origen no tiene que estar en el centro del mundo. Sin embargo, los valores máximos de x o y deben ser mayores o iguales a cero.

Nota: Usted sólo puede asignar el tamaño del mundo editando la vista –estas reportadoras no se pueden fijar por medio de comandos.

```
crt 100 [ setxy random-float max-pxcor
                random-float max-pycor ]
;; distribuye 100 tortugas al azar en el
;; primer cuadrante
```

Ver también min-pxcor, min-pycor, world-width y world-height

---

mean
mean lista

Reporta la media estadística de los ítemes numéricos dados en la lista, ignorando los ítemes no numéricos. La media es el promedio, es decir, es la suma de los ítemes dividida por el número total de estos.

```
show mean [xcor] de tortugas
;; imprime la media de las coordenadas x de todas las tortugas
```

---

median
median lista

Reporta la mediana estadística de los ítemes numéricos de la lista dada, ignorando los ítemes no numéricos. La mediana es el ítem que estaría en la mitad de todos los ítemes cuando se colocan en orden (si dos ítemes se encuentran en el medio, la mediana sería el promedio de los dos).

```
show median [xcor] de tortugas
;; imprime la mediana de las coordenadas x de las tortugas
```

---

member?

member? valor lista
member? cadena1 cadena2
member? agente conjunto-agentes

Para una lista, reporta true (verdadero) si el valor dado aparece en la lista dada, de otro modo reporta falso.
Para una cadena, reporta verdadero o falso según que la cadena1 aparezca en algún lugar como subcadena de cadena2.
Para un conjunto-agentes, reporta verdadero si el agente dado pertenece al conjunto-agentes, de otro modo reporta falso.

```
show member? 2 [1 2 3]
=> true
show member? 4 [1 2 3]
=> false
show member? "bat" "abate"
=> true
show member? turtle 0 turtles
=> true
show member? turtle 0 patches
=> false
```

Var también position.

---

min
min lista

Reporta el número cuyo valor es mínimo en la lista. Ignora los otros tipos de ítemes.

```
show min [xcor] of turtles
;; imprime el valor más bajo de x-coordinate de todas las tortugas
show min list a b
;; imprime el valor menor de las dos variables a y b
show min (list a b c)
;; imprime el menor valor de las tres variable a, b, y c
```

---

min-n-of
min-n-of número conjunto agentes [reportadora]

Reporta un conjunto-agentes que contiene la cantidad de agentes del conjunto-agentes dada por número y con los valores más bajos de la reportadora. El conjunto-agentes se construye encontrando dicho número de agentes con el valor más bajo de la reportadora, en caso de no haber ese número de agentes entonces se buscan aquellos con el segundo valor más bajo de la reportadora y así sucesivamente. Al final, si existe una atadura que podría hacer el conjunto-agentes demasiado grande, la atadura es rota al azar.

```
;; suponga que el mundo es 11 x 11
show min-n-of 5 patches [pxcor]
;; muestra 5 paracelas que cumplen pxcor = min-pxcor
show min-n-of 5 patches with [pycor = 0] [pxcor]
;; muestra un conjunto-agentes que contiene: (patch -5 0)
;; (patch -4 0) (patch -3 0) (patch -2 0) (patch -1 0)
```

Ver también min-one-of, with-min.

---

min-one-of
min-one-of conjunto-agentes [reportadora]

Reporta un agente al azar del conjunto-agentes, el cual tiene el valor más bajo de la reportadora dada. Si hay una atadura, esta orden reporta un agente tomado al azar que cumpla la condición. Si usted desea todos estos agentes, en su lugar use with-min.

```
show min-one-of turtles [xcor + ycor]
;; reporta la primera tortuga con el menor valor de la suma
;; de las coordenadas
```

Ver también with-min, min-n-of.

---

min-pxcor
min-pycor

min-pxcor
min-pycor

Estas reportadoras suministran respectivamente los valores mínimos de x-coordinate y y-coordinate de las parcelas, lo que determina el tamaño del mundo.

A diferencia de anteriores versiones de NetLogo, el origen no tiene que estar en el centro del mundo. No obstante, los valores mínimos de la coordenada x y la coordenada y deben ser mayores o a lo sumo iguales a cero.

Nota: Usted puede fijar el tamaño del mundo editando la vista – estas son reportadoras cuyos valores usted no puede asignar directamente.

```
crt 100 [ setxy random-float min-pxcor
                random-float min-pycor ]
;; distribuye 100 tortugas al azar en
;; el tercer cuadrante
```

Ver también max-pxcor, max-pycor, world-width y world-height

---

mod
número1 mod número2

Reporta número1 módulo número2: es decir, el residuo de la división número1 entre número2. mod es equivalente al siguiente código de NetLogo:

número1 - (floor (número1 / número2)) * número2

```
show 62 mod 5
=> 2
show -8 mod 3
=> 1
```

Note que el operador mod es "infijo", es decir, se coloca en medio de sus dos entradas.

Ver también remainder (residuo). mod y remainder se comportan igual para número positivos pero de modo diferente para número negativos.

---

modes
modes lista

Reporta una lista del ítem o de los ítemes más comunes en la lista. La lista de entrada puede contener cualesquiera valores NetLogo. Si la lista es vacía se reporta una lista vacía.

```
show modes [1 2 2 3 4]
=> [2]
show modes [1 2 2 3 3 4]
=> [2 3]
show modes [ [1 2 [3]] [1 2 [3]] [2 3 4] ]
=> [[1 2 [3]]
show modes [pxcor] of turtles
;; muestra cuales columnas de parcelas contienen
;; la mayor cantidad de tortugas en ellas
```

---

mouse-down?
mouse-down?

Reporta true (verdadero) si el botón del ratón está abajo, de lo contrario reporta false.

Nota: si el puntero del ratón está fuera de la vista actual, mouse-down? reportará siempre false.

---

mouse-inside?
mouse-inside?

Reporta true (verdadero) si el puntero del ratón se encuentra dentro de la vista actual, de lo contrario reporta false.

---

mouse-xcor
mouse-ycor
mouse-xcor
mouse-ycor

Reporta la coordenada x o y del ratón en la vista 2D. El valor se da en términos de coordenadas de tortuga, de manera que podría no ser un entero. Si usted desea coordenadas de parcela, use round mouse-xcor y round mouse-ycor.

Nota: Si el ratón se encuentra fuera de la vista 2D, reporta el valor de la última ocasión en que estuvo dentro.

```
;; para hacer que el ratón ‘‘dibuje’’ en rojo:
if mouse-down?
  [ ask patch mouse-xcor mouse-ycor [ set pcolor red ] ]
```

---

move-to
move-to agente

La tortuga fija sus coordenadas x y y iguales a las del agente dado. Si el agente es una parcela, el efecto es el de mover la tortuga al centro de dicha parcela.

```
move-to turtle 5
;; la tortuga se mueva al mismo punto que la tortuga 5
move-to one-of patches
;; la tortuga se mueve al centro de una parcela escogida al azar
move-to max-one-of turtles [size]
;; la tortuga se mueve al mismo punto que una de las tortugas más grandes
```

Note que la orientación de la tortuga permanece inalterada. Usted podría usar el comando face para orientar primero la tortuga en la dirección en que se moverá.

Ver también setxy.

---

movie-cancel
movie-cancel

Cancela la película actual.

---

movie-close
movie-close

Detiene la grabación de la película actual.

---

movie-grab-view

movie-grab-interface
movie-grab-view
movie-grab-interface

Añade una imagen a la vista actual (2D or 3D) o el panel de la interfaz a la película actual.

```
;; hacer una película de 20 pasos de la vista actual
setup
movie-start "out.mov"
repeat 20 [
  movie-grab-view
  go
]
movie-close
```

---

movie-set-frame-rate
movie-set-frame-rate frame-rate

Fija el "frame rate" ("razón de cuadros") de la película actual. El "frame rate" se mide en cuadros por segundo. Si usted no fija explícitamente el "frame rate", por defecto se fijará en 15 cuadros por segundo.

Debe ser llamado después de movie-start, pero antes de movie-grab-view o movie-grab-interface.

Ver también movie-status.

---

movie-start
movie-start nombre-de-archivo

Crea una nueva película, la cual se guardará en el archivo nombre-de-archivo. Este archivo será de tipo QuickTime, por lo que deberá tener extensión ".mov".

Ver también movie-grab-view, movie-grab-interface, movie-cancel, movie-status, movie-set-frame-rate, movie-close.

---

movie-status
movie-status

Reporta una cadena que describe la película actual.

```
print movie-status
=> No movie.
movie-start
print movie-status
=> 0 frames; frame rate = 15.
movie-grab-view
print movie-status
1 frames; frame rate = 15; size = 315x315.
```

---

my-breeds
my-links
my-breeds
my-links

Reporta el conjunto-agentes de todas las ligas no dirigidas conectadas al agente solicitante.

```
crt 5
ask turtle 0
[
  create-links-with other turtles
  show my-links
   ;; imprime el conjunto-agentes que contiene todas las ligas
   ;; (puesto que todas las ligas creadas fueron con la tortuga 0)
]
ask turtle 1
[
  show my-links ;; muestra un conjunto-agentes que contiene la liga 0 1
```

```
]
end
```

---

my-in-< *breeds* >
my-in-links
my-in-< *breeds* >
my-in-links

Reporta el conjunto-agentes de todas las ligas dirigidas entrando desde otros nodos hacia el agente solicitante.

crt 5 ask turtle 0 [create-links-to other turtles show my-in-links ;; muestra un conjunto-agentes vacío] ask turtle 1 [show my-in-links ;; muesstra un conjunto-agentes que contiene la liga 0 1]

---

my-out-< *breeds* >
my-out-links
my-out-< *breeds* >
my-out-links

Reporta el conjunto-agentes de todas las ligas saliendo del agente solicitante hacia otros nodos.

```
crt 5
ask turtle 0
[
  create-links-to other turtles
  show my-out-links ;; muestra el conjunto-agentes
                    ;; que contienetodas las ligas
]
ask turtle 1
[
  show my-out-links ;; muestra un conjunto-agentes vacío
]
```

---

myself
myself

"self" y "myself" son muy diferentes. "self" es simple; significa "yo". "myself" significa "la tortuga o parcela que me pidio a mí hacer lo que estoy haciendo ahora".

Cuando a un agente se le solicita ejecutar algún código, usar myself en ese código reporta al agente (tortuga o parcela) que hizo la solicitud.

myself se usa más frecuentemente en conjunción con of para leer o fijar variables del agente solicitante.

myself puede usarse dentro de bloques de código, no sólo en el comando ask, sino también en hatch, sprout, of, with, all?, with-min, with-max, min-one-of, max-one-of, min-n-of, max-n-of.

```
ask turtles
  [ ask patches in-radius 3
      [ set pcolor [color] of myself ] ]
;; cada tortuga hace una mancha de colores alrededor de ella
```

Ver el ejemplo de código "Myself Example"para más ejemplos.
Ver también self.

N

---

n-of

n-of size conjunto-agentes
n-of tamaño lista

De un conjunto-agentes dado reporta un conjunto-agentes del tamaño indicado con sus miembros escogidos al azar del conjunto-agentes de entrada, sin repeticiones.

De una lista dada reporta una lista del tamaño indicado, con miembros escogidos al azar de la lista de entrada, sin repeticiones. Los ítemes en la lista resultante mantienen un orden compatible con el que tenían en la lista de entrada (si usted los desea en orden aleatorio, use "shuffle" en el resultado).

Se produce un error si "tamaño" es mayor que el tamaño del conjunto-agentes o lista de entrada.

```
ask n-of 50 patches [ set pcolor green ]
;; 50 parcelas escogidas al azar se vuelven verdes
```

Ver también one-of.

---

n-values

n-values tamaño tarea-reportadora

Reporta una lista del tamaño dada por "tamaño", la cual contiene valores calculados obtenidos corriendo repetidas veces la tarea.

Si la tarea acepta entradas, la entrada será el número del ítem que se está procesando, comenzando a partir de cero.

```
show n-values 5 [1]
=> [1 1 1 1 1]
show n-values 5 [?]
=> [0 1 2 3 4]
```

```
show n-values 3 turtle
=> [(turtle 0) (turtle 1) (turtle 2)]
show n-values 5 [? * ?]
=> [0 1 4 9 16]
```

Ver también reduce, filter, ?

---

neighbors
neighbors4

neighbors
neighbors4

Reporta un conjunto-agentes que contiene las 8 o las 4 parcelas vecinas que rodean al agente.

```
   ;;
show sum [count turtles-here] of neighbors
  ;; imprime el número total de tortugas en las ocho parcelas
  ;; alrededor de esta tortuga o parcela
show count turtles-on neighbors
  ;; una manera más corta de decir la misma cosa
ask neighbors4 [ set pcolor red ]
  ;; vuelve rojas las cuatro parcelas vecinas
```

<breed>-neighbors
link-neighbors

<breed>-neighbors
link-neighbors

Reporta el conjunto-agentes de todas las tortugas que se encuentran en el otro extremo de las ligas no dirigidas conectadas a esta tortuga.

```
crt 3
```

```
ask turtle 0
 [
  create-links-with other turtles
  ask link-neighbors [ set color red ]
  ;; las tortugas 1 y 2 se vuelven rojas
 ]
ask turtle 1
 [
  ask link-neighbors [ set color blue ]
  ;; la tortuga 0 se vuelve azul
 ]
end
```

<breed>-neighbor?
link-neighbor?

<breed>-neighbor? tortuga
link-neighbor? tortuga

Reporta true (verdadero) si hay una liga no dirigida entre la tortuga y el agente solicitante.

```
crt 2
ask turtle 0
[
  create-link-with turtle 1
  show link-neighbor? turtle 1 ;; imprime true (verdadero)
]
ask turtle 1
[
  show link-neighbor? turtle 0     ;; imprime true
]
```

---

netlogo-applet?

netlogo-applet?

Reporta true (verdadero) si el modelo está corriendo un applet.

---

netlogo-version
netlogo-version

Reporta una cadena que contiene el número de versión de NetLogo que usted está corriendo.

```
show netlogo-version
=> "5.1.0"
```

---

netlogo-web?
netlogo-web?

Reporta true (verdadero) si NetLogo está corriendo en la Web.

---

new-seed
new-seed

Reporta un número apto para alimentar el generador de números aleatorios como semilla.

Los números reportados por new-seed están basados en la fecha y la hora actual en milisegundos y están en el rango usable de semillas del generador, de -2147483648 a 2147483647.

Ver también random-seed.

---

no-display
no-display

Desactiva todas las actualizaciones de la vista actual, hasta que se emita el comando display. Esto tiene dos usos principales.

Primero: usted puede controlar los momentos en que el usuario puede ver actualizaciones de la vista. Usted tal vez quiera cambiar varias cosas en la vista sin que el usuario lo sepa y luego hacer visibles los cambios de una sola vez.

Segundo: su modelo correrá más rápido cuando las actualizaciones de la vista estén desactivadas, de modo que cuando anda corto de tiempo, este comando le permite obtener resultados más rápidamente. Note que normalmente no necesitará usar no-display para esto último pues también puede usar el interruptor on/off en la banda de control de la vista, para congelarla.

Ver también display.

---

nobody
nobody

Este es un valor especial que reportan algunas primitivas tales como turtle, one-of, max-one-of, etc. para indicar que no se encontró ningún agente. También, cuando una tortuga muere se vuelve igual a nobody (nadie).

Nota: conjunto-agentes vacíos no son iguales a nobody. Si usted quiere verificar si un conjunto-agentes esta vacío, use any? Se obtiene como respuesta nobody sólo en situaciones donde uno esperaría un agente individual, no todo un conjunto-agentes.

```
set target one-of other turtles-here
if target != nobody
```

```
[ ask target [ set color red ] ]
```

---

no-links
no-links

Reporta un conjunto-agentes de ligas vacío.

---

no-patches
no-patches

Reporta un conjunto-agentes de parcelas vacío.

---

not
not booleano

Reporta true (verdadero) si booleano es falso, en otro caso reporta false.

```
if not any? turtles [ crt 10 ]
```

---

no-turtles
no-turtles

Reporta un conjunto-agentes de tortugas vacío.

O

---

of

[reportadora] of agente
[reportadora] of conjunto-agentes

Para un agente, reporta el valor de la reportadora para ese agente (tortuga o parcela).

```
show [pxcor] of patch 3 5
;; imprime 3
show [pxcor] of one-of patches
;; imprime el valor de la variable pxcor de una parcela escogida
   ;; al azar
show [who * who] of turtle 5
=> 25
show [count turtles in-radius 3] of patch 0 0
;; imprime el número de tortugas ubicadas dentro de un radio de
;; tres parcelas centrado en el origen

Para un conjunto-agentes reporta una lista que contiene el valor
de la reportadora para cada agente en el conjunto-agentes
(en orden aleatorio).

crt 4
show sort [who] of turtles
=> [0 1 2 3]
show sort [who * who] of turtles
=> [0 1 4 9]
```

---

one-of

one-of conjunto-agentes
one-of lista

De un conjunto-agentes reporta un agente al azar. Si el conjunto-agentes está vacío reporta nobody (nadie).
De una lista reporta un ítem al azar. Se produce un error si la lista está vacía.

```
ask one-of patches [ set pcolor green ]
;; una parcela al azar se vuelve verde
ask patches with [any? turtles-here]
  [ show one-of turtles-here ]
;; para cada parcela que contiene tortugas imprime
;; una de las tortugas

;; suponga que milista es [1 2 3 4 5 6]
show one-of milista
;; imprime un valor al azar de milista
```

Ver también n-of.

---

or
booleano1 or booleano2

Reporta true (verdadero) si booleano1 o booleano2 es verdadero o si ambos lo son.

Note que si la primera condición es verdadera entonces no se evalúa la segunda condición (pues su valor ya no afecta el resultado).

```
if (pxcor > 0) or (pycor > 0) [ set pcolor red ]
;; las parcelas se vuelven rojas excepto las del cuadrante inferior izquierdo.
```

---

other
other conjunto-agentes

Reporta un conjunto-agentes igual al de entrada pero omitiendo al agente solicitante.

```
show count turtles-here
=> 10
```

```
show count other turtles-here
=> 9
```

---

other-end
other-end

Si lo ejecuta una tortuga reporta la tortuga en el otro extremo de la liga solicitante.

Si lo ejecuta una liga, reporta la tortuga en el extremo de la liga que no es la tortuga solicitante.

Estas definiciones son difíciles de entender abstractamente, pero los siguientes ejemplos deberían ayudar:

```
ask turtle 0 [ create-link-with turtle 1 ]
ask turtle 0 [ ask link 0 1 [ show other-end ] ] ;; imprime turtle 1
ask turtle 1 [ ask link 0 1 [ show other-end ] ] ;; imprimer turtle 0
ask link 0 1 [ ask turtle 0 [ show other-end ] ] ;; imprimer turtle 1
```

Esperamos que estos ejemplos aclaren que el "otro" extremo no es el extremos que llama o solicita ni aquel que es preguntado.

---

out-<breed>-neighbor?
out-link-neighbor?

out-<breed>-neighbor? tortuga
out-link-neighbor? tortuga

Reporta true (verdadero) si hay una liga dirigida que sale de la tortuga solicitante hacia la tortuga de entrada.

```
crt 2
ask turtle 0 [
  create-link-to turtle 1
  show in-link-neighbor? turtle 1  ;; imprime false
  show out-link-neighbor? turtle 1 ;; imprime true (verdadero)
]
ask turtle 1 [
  show in-link-neighbor? turtle 0  ;; imprime true
  show out-link-neighbor? turtle 0 ;; imprime false
]
```

---

out-<breed>-neighbors
out-link-neighbors

Reporta el conjunto-agentes de todas las tortugas que tiene ligas dirigidas saliendo del agente solicitante.

```
crt 4
ask turtle 0
[
  create-links-to other turtles
  ask out-link-neighbors [ set color pink ] ;; las tortugas 1-3
 ;; se vuelven rosadas
]
ask turtle 1
[
  ask out-link-neighbors [ set color orange ]  ;; ninguna tortuga
  ;; cambia de color pues la tortuga 1 sólo tiene ligas entrantes.
]
end
```

---

out-<breed>-to
out-link-to

out-<breed>-to tortuga
out-link-to tortuga

Reporta la liga dirigida del solicitante a la tortuga. Si no hay ligas entonces reporta nobody (nadie).

```
crt 2
ask turtle 0 [
  create-link-to turtle 1
  show out-link-to turtle 1 ;; muestra link 0 1
]
ask turtle 1
[
  show out-link-to turtle 0 ;; muestra nobody
]
```

Ver también: in-link-from link-with

---

output-print
output-show
output-type
output-write

output-print valor
output-show valor
output-type valor
output-write valor

Estos comandos hacen lo mismo que print, show, type y write, excepto que valor es impreso en el área de salida del modelo en vez de la Terminal de Instrucciones. Si el modelo no posee un área de salida separada, entonces se usará la Terminal de Instrucciones.

P

patch
patch xcor ycor

Dadas la coordenada x y y de un punto, reporta la parcela que contiene ese punto. Las coordenadas son coordenadas absolutas; no son relativas a este agente como en el caso de patch-at.

Si x y y son enteros el punto es el centro de la parcela. Si x o y no es entero, se redondea al entero más cercano para determinar la parcela que contiene al punto.

Si las topologías de la rosquilla o del tubo están habilitadas, las coordenadas se adaptan a la topología. En el caso de la topología del cuadrado si las coordenadas dadas se salen fuera del mundo, reporta nobody (nadie).

```
ask patch 3 -4 [ set pcolor green ]
;; la parcela con pxcor de 3 y pycor de -4 se vuelve verde
show patch 1.2 3.7
;; imprime (patch 1 4); nótese el redondeo
show patch 18 19
;; suponiendo que min-pxcor and min-pycor are -17
;; y max-pxcor y max-pycor son 17,
;; en la topología de la rosquilla imprime (patch -17 -16);
;; en la topología del cuadrado imprime nobody
```

Ver también patch-at

patch-ahead
patch-ahead distancia

Reporta la parcela que se encuentra a la distancia indicada por delante de la tortuga solicitante, es decir en la dirección de la orientación de la tortuga. Reporta nobody (nadie) si la parcela no existe porque está fuera del mundo (en la topología del cuadrado).

```
ask patch-ahead 1 [ set pcolor green ]
;; la parcela 1 paso delante de esta tortuga se vuelve
;; verde, note que esta podría ser la misma parcela
;; sobre la que se encuentra parada la tortuga
```

Ver también patch-at, patch-left-and-ahead, patch-right-and-ahead, patch-at-heading-and-distance.

---

patch-at
patch-at dx dy

Reporta la parcela a la distancia (dx, dy) del agente solicitante, es decir, la parcela que contiene el punto a dx parcelas este y dy parcelas norte del solicitante.

Reporta nadie (nobody) si el punto se encuentra fuera de los límites del mundo con la topología del cuadrado.

```
ask patch-at 1 -1 [ set pcolor green ]
;; si el solicitante es una tortuga o una parcela, la parcela
;; justo al sureste del solicitante se vuele verde
```

Ver también patch, patch-ahead, patch-left-and-ahead, patch-right-and-ahead, patch-at-heading-and-distance.

---

patch-at-heading-and-distance
patch-at-heading-and-distance orientación distancia

patch-at-heading-and-distance reporta la parcela que se encuentra a la dis-

tancia dada de la tortuga o parcela solicitante y en la dirección absoluta dada por la orientación. A diferencia con patch-left-and-ahead y patch-right-and-ahead, la orientación actual de la solicitante no se toma en cuenta. Reporta nobody (nadie) si la parcela no existe por estar fuera de los límites del mundo (en al topología del cuadrado).

```
ask patch-at-heading-and-distance -90 1 [ set pcolor green ]
;; la parcela 1 paso al oeste de esta parcela se vuelve verde
```

Ver también patch, patch-at, patch-left-and-ahead, patch-right-and-ahead.

---

patch-here
patch-here

patch-here reporta la parcela debajo de la tortuga.

Note que esta reportadora no está disponible para la parcela ya que las parcelas pueden decir "self".

---

patch-left-and-ahead
patch-right-and-ahead

patch-left-and-ahead ángulo distancia
patch-right-and-ahead ángulo distancia

Reporta la parcela que se encuentra a la distancia dada de la tortuga solicitante y en la dirección obtenida girando a izquierda o derecha el ángulo dado (en grados) a partir de la orientación actual de la tortuga. Reporta nobody (nadie) en caso de no existir esa parcela por estar fuera de los límites del mundo (con la topología del cuadrado).

Si usted desea referirse a una parcela dada en términos de una orientación

absoluta, en vez de relativa a la orientación actual de la tortuga solicitante, use entonces patch-at-heading-and-distance.

```
ask patch-right-and-ahead 30 1 [ set pcolor green ]
;; esta tortuga ‘‘mira’’ 30 grados a la derecha de su orientación
;;   actual a la parcela alejada 1 unidad de ella y vuelve esa
;;   parcela verde; note que esta parcela podría ser la misma que
;;   la parcela sobre la que la tortuga se encuentra
```

Ver también patch, patch-at, patch-at-heading-and-distance.

---

patch-set

patch-set valor1
patch-set valor1 valor2...

Reporta un conjunto-agentes de todas las parcelas en cualquiera de las entradas. Las entradas pueden ser parcelas individuales, conjunto-agentes, nobody (nadie) o listas (o listas anidadas) que contengan algunas de las anteriores.

```
patch-set self
patch-set patch-here
(patch-set self neighbors)
(patch-set patch-here neighbors)
(patch-set patch 0 0 patch 1 3 patch 4 -2)
(patch-set patch-at -1 1 patch-at 0 1 patch-at 1 1)
patch-set [patch-here] of turtles
patch-set [neighbors] of turtles
```

Ver también turtle-set, link-set.

---

patch-size

patch-size

Reporta el tamaño en pixeles de las parcelas en la vista. El tamaño es típi-

camente un entero, pero podría también ser un número de punto flotante (floating point number).

Ver también set-patch-size.

---

patches
patches

Reporta el conjunto-agentes que consiste de todas las parcelas.

---

patches-own
patches-own [**variable1...**]

Esta palabra clave, lo mismo que globals, breed, <breed>-own y turtles-own, sólo puede usarse al inicio de un programa, antes de la definición de los procedimientos. Define las variables que todas las parcelas pueden usar.

Todas las parcelas poseerán las variables dadas y podrán usarlas.

Todas las variables de una parcela también pueden ser accesadas por cualquier tortuga que se encuentre parada sobre la parcela.

Ver también globals, turtles-own, breed, ¡breeds¿-own.

---

pcolor
pcolor

Se trata de una variable preinstalada del sistema, la cual guarda el color de la parcela. Se puede reasignar el valor de esta variable para cambiar el color de la parcela.

Todas las variables de las parcelas pueden ser directamente accesadas por las tortugas que se encuentran sobre las ellas. El color puede representarse como un color NetLogo (un sólo número) o un color RGB (una lista de 3 números). Ver detalles en la sección Colors de la Guía de Programación (Programming Guide).

Ver también color.

---

pen-down, pd
pen-erase, pe
pen-up, pu

pendown
penerase
penup

La tortuga cambia de modo entre dibujar líneas, borrar líneas o ninguna de ambas cosas. Las líneas siempre se muestran sobre las parcelas y debajo de las tortugas. Se cambia el color de la pluma cambiando el color de la tortuga con set color.

Nota: Cuando la pluma de la tortuga está abajo, todos los comandos de movimiento tendrán como efecto que se dibujen líneas, incluyendo jump, setxy y move-to.

Nota: Estos comandos equivalen a fijar la variable de tortuga en “up”, “down” o “erase”. En Windows, dibujar y borrar una línea podría no borrar todos los pixeles.

---

pen-mode

pen-mode

Esta es una variable preinstalada del sistema, la cual guarda el estado de la pluma de la tortuga. Se asigna el valor de esta variable para dibujar líneas, borrarlas o detener cualquiera de estos efectos. Los valores posibles son “up”, “down” o “erase”.

---

pen-size
pen-size

Es una variable de tortuga preinstalada en el sistema que guarda la anchura en pixeles de las líneas que trazará (o borrará) la tortuga según el modo en que se encentre (dibujar o borrar).

---

plabel
plabel

Es una variable de parcela preinstalada en el sistema. La parcela aparece en la vista con el valor dado “adherido” a ella como texto. Se puede asignar valor a esta variable para añadir, cambiar o quitar la etiqueta de una parcela.

Todas las variables de parcelas pueden ser accesadas por cualquier tortuga que se encuentre sobre la parcela.

Ver también plabel-color, label, label-color.

---

plabel-color
plabel-color

Es una variable de parcela preinstalada en el sistema, la cual guarda un

número mayor o igual a 0 y menor a 140. Este número determina en qué color aparece la etiqueta de la parcela (caso de tener una). Se puede asignar esta variable para cambiar el color de la etiqueta de las parcelas.

Todas las variables de parcela pueden ser directamente accesadas por cualquier tortuga que se encuentre sobre la parcela.

Ver también plabel, label, label-color.

---

plot
plot número

Incrementa el x-value de la pluma graficadora por una cantidad de plot-pen-interval y luego marca un punto con este nuevo valor de x-value y un valor y-value dado por número. La primera vez que el comando se usa en un gráfico, el punto marcado tiene un valor x-value de 0.

---

plot-name
plot-name

Reporta una cadena con el nombre del gráfico (plot) actual.

---

plot-pen-exists?
plot-pen-exists? cadena

Reporta true (veradero) si en el gráfico actual se ha definido una pluma graficadora con el nombre dado por la cadena. De otro modo reporta false.

---

plot-pen-down
plot-pen-up

plot-pen-down
plot-pen-up

Pone abajo o arriba la pluma actual de graficar (plot-pen) de manera que dibuja si está abajo o no lo hace si está arriba. Por defecto todas las plumas están inicialmente abajo.

---

plot-pen-reset
plot-pen-reset

Limpia todo lo que la pluma de graficar actual ha dibujado, la lleva a la posición (0, 0) y la pone en modo abajo. Si se trata de una pluma permanente, el color y el modo se fijan en los valores por defecto que muestra la ventana Edit de la pluma.

---

plotxy
plotxy número1 número2

Mueve la pluma actual de graficar al punto de coordenadas número1, número2. Si la pluma está abajo dibujará una línea, barra o punto (dependiendo del modo de la pluma).

---

plot-x-min
plot-x-max
plot-y-min

plot-y-max

plot-x-min
plot-x-max
plot-y-min
plot-y-max

Reporta el mínimo o máximo valor el eje x o y del gráfico actual.

Estos valores pueden asignarse con los comandos set-plot-x-range y set-plot-y-range (sus valores por defecto se fijan desde la ventana de la opción Edit del gráfico.)

---

position

position ítem lista
position cadena1 cadena2

En una lista, reporta la primera posición del ítem en la lista o false si este no aparece.

En una cadena reporta la posición de la primera aparición de cadena1 como subcadena de cadena2 o falso si esta no aparece.

Nota: Las posiciones se numeran a partir de 0, no de 1.

```
;; suponga que milista es [2 7 4 7 "Pepe"]
show position 7 milista
=> 1
show position 10 mylist
=> false
show position "en" "cadena"
=> 3
```

---

precision
precision número lugares

Reporta el número dado redondeado a tantas cifras decimales como las indicadas por lugares.

Si lugares es negativo, el redondeo se lleva a cabo a la izquierda del punto decimal.

```
show precision 1.23456789 3
=> 1.235
show precision 3834 -3
=> 4000
```

Ver también round, ceiling, floor.

---

print
print valor

Imprime el valor dado en la Terminal de Instrucciones, seguido de un retorno de carro.

El agente solicitante no se imprime antes del valor, a diferencia de show.

Ver también show, type y write.

Ver también output-print.

---

pxcor
pycor
pxcor
pycor

Se trata de variables preinstaldas de las parcelas. Estas variables guardan las coordenadas x y y de la parcela, las cuales son siempre número enteros. No se pueden cambiar sus valores porque las parcelas no se mueven.

pxcor es mayor o a lo sumo igual a min-pxcor y menor o a lo sumo igual a max-pxcor; similarmente para pycor, min-pycor y max-pycor.

Todas las variables de las parcelas pueden ser directamente accesadas por las tortugas que se encuentran sobre ellas.

Ver también xcor, ycor.

R

---

random
random número

Si número es positivo, reporta un entero al azar, mayor o igual que 0 pero estrictamente menor que número.

Si número es negativo, reporta un entero al azar menor o igual que 0 pero estrictamente mayor que número.

Si número es cero, el resultdo es siempre cero.

Nota: En versiones de NetLogo anteriores a la versión 2.0, esta primitiva reportaba un número de punto flotante si se le daba como entrada un número no entero. Este ya no es el caso. Si usted desea como respuesta un número de punto flotante entonces debe usar random-float.

```
show random 3
;; imprime 0, 1 o 2
show random -3
;; imprime 0, -1 o -2
show random 3.5
;; imprime 0, 1, 2 o 3
```

Ver también random-float.

---

random-float
random-float número

Si número es positivo, reporta un número de punto flotante al azar, mayor o igual que 0 pero estrictamente menor que número.

Si número es negativo, reporta un número de punto flotante al azar menor o igual que 0 pero estrictamente mayor que número.

Si número es cero, el resultdo es siempre cero.

```
show random-float 3
;; imprime un número al menos igual que 0 pero menor que 3
;; por ejemplo 2.589444906014774
show random-float 2.5
;; imprime un número al menos igual que 0 pero menor a 2.5
;; por ejemplo 1.0897423196760796
```

---

random-exponential
random-gamma
random-normal
random-poisson

random-exponential media
random-gamma alfa lambda
random-normal media desviación-estándar
random-poisson media

Reporta un número al azar de la distribución correspondiente, con la media de entrada y -en el caso de la distribución normal- la desviación estándar (la desviación estándar puede no ser negativa).

random-exponential reporta un número de punto flotante al azar de una distribución exponencial. Es equivalente a (- mean) * ln random-float 1.0.

random-gamma reporta un número de punto flotante al azar de una distribución gama, controlado por el punto flotante de parámetros alfa y lambda. Ambas entradas deben ser mayores que cero. Nota: para obtener resultados con una media y variancia dadas, utilice entradas del modo siguiente:
alfa = mean * mean / variance; lambda = 1 / (variance / mean).

random-normal reporta un número de punto flotante al azar de una distribución normal.

random-poisson reporta un entero al azar de una distribución de Poisson.

```
show random-exponential 2
;; imprime un número de punto flotante al azar de una distribución
;; exponencial con una media de 2
show random-normal 10.1 5.2
;; imprime un número de punto flotante al azar de una distribución
;; normal con una media de 10.1 y una desviación estándar de 5.2
show random-poisson 3.4
;; imprime un entero al azar de una distribución de Poisson
;; con una media de 3.4
```

---

random-pxcor
random-pycor

random-pxcor
random-pycor

Reporta un entero al azar en el rango desde min-pxcor (o -y) a max-pxcor (o y) inclusive.

```
ask turtles [setxy random-pxcor random-pycor]
;; mueve cada tortuga a una parcela de coordenadas
  ;; random-pxcor random-pycor elegidas al azar
```

Ver también random-xcor, random-ycor.

---

random-seed

random-seed número

Fija la semilla (seed) del generador de números pseudo-aleatorios a la parte entera de número. La semilla debe estar en el rango de -2147483648 a 2147483647; note que este rango es más pequeño que el rango total de enteros que soporta NetLogo (-9007199254740992 a 9007199254740992).

Ver la sección de números aleatorios de la Guía de Programación para más detalles.

```
random-seed 47822
show random 100
=> 50
show random 100
=> 35
random-seed 47822
show random 100
=> 50
show random 100
=> 35
```

---

random-xcor

random-ycor

random-xcor
random-ycor

Reporta un número de punto flotante al azar en el rango de coordenadas de tortuga disponibles en los ejes dados x o y.

El rango de coordenadas varía horizontalmente de min-pxcor - 0.5 (inclusive) a max-pxcor + 0.5 (excluido); verticalmente de manera análoga sustituyendo -y por -x.

```
ask turtles [
  ;; mover cada tortuga a un punto elegido al azar
  setxy random-xcor random-ycor
]
```

Ver también random-pxcor, random-pycor.

---

read-from-string

read-from-string cadena

Interpreta la cadena dada como si se hubiese escrito en la Ventana del Observador (Command Center) y reporta el valor resultante. El resultado podría ser un número, lista, cadena o valor booleano o el valor especial "nobody"(nadie).

Útil en conjunción con la primitiva user-input para convertir lo ingresado por el usuario en algo utilizable.

```
show read-from-string "3" + read-from-string "5"
=> 8
show length read-from-string "[1 2 3]"
=> 3
```

```
crt read-from-string user-input "¿Crear cuántas tortugas?"
;; se crea una cantidad de tortugas igual al número
;; ingresado por el usuario
```

---

reduce

reduce tarea-reportadora lista

Reduce una lista de izquierda a derecha usando la tarea dada, dando como resultado un solo valor. Esto significa, por ejemplo, que reduce [?1 + ?2] [1 2 3 4] es equivalente a (((1 + 2) + 3) + 4). Si la lista tiene un único item, dicho item es reportado. Aplicar "reduce" a una lista vacía produce un error.

La primera entrada que se pasa a la tarea da como resultado ella misma, la segunda entrada es el segundo item en la lista.

Puesto que puede ser difícil desarrollar una intuición acerca de lo que hace reduce, aquí hay algunos ejemplos simples, los que -pese a no ser útiles en sí mismos- podrían ayudar a entender mejor esta primitiva:

```
show reduce + [1 2 3]
=> 6
show reduce - [1 2 3]
=> -4
show reduce [?2 - ?1] [1 2 3]
=> 2
show reduce [?1] [1 2 3]
=> 1
show reduce [?2] [1 2 3]
=> 3
show reduce sentence [[1 2] [3 [4]] 5]
=> [1 2 3 [4] 5]
show reduce [fput ?2 ?1] (fput [] [1 2 3 4 5])
=> [5 4 3 2 1]
```

Aquí hay algunos ejemplos de mayor utilidad:

```
;; encontrar la cadena más larga en una lista
```

```
to-report longest-string [strings]
  report reduce
    [ifelse-value (length ?1 >= length ?2) [?1] [?2]]
    strings
end

show longest-string ["hola" "ustedes" "!"]
=> "ustedes"

;; contar el número de veces que ocurre un item en una lista
to-report occurrences [x the-list]
  report reduce
    [ifelse-value (?2 = x) [?1 + 1] [?1]] (fput 0 the-list)
end

show occurrences 1 [1 2 1 3 1 2 3 1 1 4 5 1]
=> 6

;; evaluar un polinomio con coeficientes dados en x
to-report evaluar-polinomio [coeficientes x]
  report reduce [(x * ?1) + ?2] coeficientes
end

;; evaluar 3x^2 + 2x + 1 en x = 4
show evaluar-polinomio [3 2 1] 4
=> 57
```

---

remainder

remainder número1 número2

Reporta el residuo de la división de número1 entre número2. Esto es equivalente al siguiente código de NetLogo:

```
number1 - (int (number1 / number2)) * number2
show remainder 62 5
=> 2
show remainder -8 3
=> -2
```

Ver también mod. Note que mod y remainder se comportan de igual manera para números positivos, sin embargo no es así para números negativos.

---

remove

remove item lista
remove cadena1 cadena2

Para una lista reporta una copia de la lista en la cual se han removido todas las instancias o apariciones del item.

Para cadenas, reporta una copia de cadena2 en la cual se han removido todas las instancias o apariciones de la subcadena cadena1.

```
set milista [2 7 4 7 "Pepe"]
set milista remove 7 milista
;; milista es ahora [2 4 "Pepe"]
show remove "te" "inteligente"
=> "inligen"
```

---

remove-duplicates

remove-duplicates lista

Reporta una copia de la lista preservando un sólo ejemplar de cada item, es decir eliminando las duplicaciones de los ítemes.

```
set milista [2 7 4 7 "Pepe" 7]
set milista remove-duplicates milista
;; milista es ahora [2 7 4 "Pepe"]
```

---

remove-item

remove-item índice lista
remove-item índice cadena

En una lista reporta una copia de la misma con el item dado por índice removido.

En una cadena reporta una copia de la misma con el caracter en el índice dado removido.

Note que los índices comienzan en 0, no en 1 (el primer item es el item 0, el segundo es el item 1 y así sucesivamente).

```
set milista [2 7 4 7 "Pepe"]
set milista remove-item 2 milista
;; milista es ahora [2 7 7 "Pepe"]
show remove-item 2 "castillo"
=> "catillo"
```

---

repeat

repeat número [comandos]

Corre los comandos el número de veces indicado indicado por número.

```
pd repeat 36 [ fd 1 rt 10 ]
;; la tortuga dibuja un círculo
```

---

replace-item

replace-item índice lista valor
replace-item índice cadena1 cadena2

En una lista reemplaza el item cuyo índice es el indicado por valor. Los índices comienzan en 0 (el 6º item tendría un índice de 5). Note que "replace-item" se usa conjuntamente con "set" para cambiar una lista.

Igualmente para una cadena, pero removiendo el caracter de cadena1 que ocupa la posición dada por índice y poniendo en su lugar el contenido de cadena2.

```
show replace-item 2 [2 7 4 5] 15
=> [2 7 15 5]
show replace-item 1 "piedad" "ropi"
=> "propiedad"
```

---

report
report valor

Sale inmediatamente del procedimiento actual to-report y reporta el valor dado como resultado de dicho procedimiento. Las primitivas report y to-report se usan siempre en combinación una con otra. Para una discusión sobre cómo usarlas ver to-report.

---

reset-perspective
rp
reset-perspective

El observador deja de observar, seguir o cabalgar cualquier tortuga (o parcela). Si no estaba haciendo ninguna de estas cosas nada ocurre. En la vista 3D, el observador también regresa a su posición por defecto (sobre el origen, mirando directamente hacia abajo)

Ver también follow, ride, watch.

---

reset-ticks
reset-ticks

Restablece el contador de ticks en cero, alista todos los gráficos, luego los actualiza (de modo que el estado inicial del mundo es graficado).

Normalmente reset-ticks va al final de un procedimiento setup.

Ver también clear-ticks, tick, ticks, tick-advance, setup-plots, update-plots.

---

reset-timer

reset-timer

Restablece el cronómetro (timer) a cero segundos. Ver también timer.

Note que el cronómetro (timer) es distinto al contador de ticks. El cronómetro mide el tiempo real transcurrido en segundos, el contador de ticks mide el tiempo transcurrido para el modelo en ticks.

---

resize-world

resize-world min-pxcor max-pxcor min-pycor max-pycor

Cambia el tamaño del cuadriculado de las parcelas.

Como efecto colateral, todas las tortugas y ligas mueren, el cuadriculado actual es eliminado y se crean nuevas parcelas.

Se desaconseja mantener referencias a viejas parcelas o conjuntos de estas pues esto puede causar errores de ejecución o comportamientos inesperados.

Ver también set-patch-size.

---

reverse

reverse lista
reverse cadena

Reporta una copia en orden inverso de la lista o cadena dada.

```
show milista
;; milista es [2 7 4 "Pepe"]
set milista reverse milista
;; milista es ahora ["Pepe" 4 7 2]
show reverse "notar"
=> "raton"
```

---

rgb

rgb red green blue

Reporta una lista RGB cuando se le dan tres números que describen un color RGB. El rango de los números debe estar entre 0 y 255.

Ver también hsb

---

ride

ride tortuga

Fija la perspectiva en turtle (tortuga).

Cada vez que la tortuga se mueve, el observador se mueve. Por lo tanto, en la vista 2D la tortuga permanece en el centro de la vista. En la vista 3D es como si viéramos a través de los ojos de la tortuga. Si la tortuga muere, la perspectiva se restablece al estado por defecto.

Ver también reset-perspective, watch, follow, subject.

---

ride-me
ride-me

Le pide al observador que cabalgue la tortuga solicitante.

Ver también ride.

---

right
rt

right número

La tortuga gira hacia la derecha sobre su propio eje una cantidad de grados dada por número. Si número es negativo el giro es hacia la izquierda.

---

round
round número

Reporta el entero más cercano a número.

Si la parte decimal de número es exactamente .5, el número es redondeado en la dirección positiva.

Note que el redondeo en la dirección positiva no es como se hace el redondeo en otros programas de software (en particular, no corresponde a la conducta de StarLogoT, que siempre redondea los número terminados en 0.5 al entero par más cercano). La justificación para este comportamiento corresponde a la manera como las coordenadas de tortuga se relacionan con las coordenadas de parcela en NetLogo. Por ejemplo, si la coordenada xcor de una tortuga es -4.5, entonces se encuentra en la frontera entre una parcela cuya pxcor es -4

y una cuya pxcor es -5, pero como la tortuga debe estar en alguna de las dos parcelas, se considera que se encuentra en la parcela cuya pxcor es -4, debido a que el redondeo se hace hacia los número positivos.

```
show round 4.2
=> 4
show round 4.5
=> 5
show round -4.5
=> -4
```

Ver también precision, ceiling, floor.

---

run
runresult

run command-task
run string
runresult reporter-task
runresult string

La forma run espera una tarea de tipo comando (command task) o una cadena que contiene comandos. El agente solicitante entonces corre dichos comandos.

La forma runresult espera una tarea reportadora o una cadena que contiene una reportadora. El solicitante entonces lo corre y reporta el resultado.

Note que no se puede usar run para definir o redefinir procedimientos. Si usted se preocupa por eficiencia, note que el código debe compilarse primero, lo cual toma tiempo. Sin embargo los bits compilados son "cacheados" (guardados en un cache) por NetLogo y usar run sobre la misma cadena una y otra vez es mucho más rápido que correrlo sobre diferentes cadenas. El primer run, no obstante, será muchas veces más lento que correr el mismo

código directamente o en una tarea de comandos.

En lo posible se recomienda las tareas (tasks) sobre las cadenas. Un ejemplo donde usted debe usar cadenas es cuando usted acepta trozos de código del usuario del modelo.

Las tareas pueden libremente leer o reasignar variables locales y entradas de procedimientos. Si tratamos de hacer lo mismo con cadenas, no podemos garantizar que vaya a funcionar.

S

---

scale-color

scale-color color número rango1 rango2

Reporta una sombra de color proporcional al valor de número.

Típicamente número es una variable de agente, pero podría ser cualquier reportadora numérica.

Si rango1 es menor a rango2 entonces entre más grande sea número más liviana será la sombra de color. Pero si rango2 es menor que rango1, la escala de color se invierte.

Si número es menor que rango1 entonces se escoge la sombra más oscura de color.

Si número es mayor que rango2 entonces se escoge la sombra más liviana de color.

Nota: para color la sombra es irrelevante, es decir, green y green + 2 son equivalentes y se usa el mismo espectro de color.

```
ask turtles [ set color scale-color red edad 0 50 ]
;; colorea cada tortuga con una sombra de rojo proporcional
;; al valor de la variable edad
```

---

self
self

Reporta esta tortuga, parcela o liga.

"self" y "myself" son muy diferentes. "self" es simple pues significa "yo". "myself" signfica "el agente que me pidió hacer lo que estoy haciendo ahora mismo".

Note que siempre es redundante escribir [cualquier cosa] de self. Esto siempre equivale a escribir simplemente cualquier-cosa.

Ver también myself.

;(punto y coma)
; comentarios

Después de un punto y coma, el resto de la línea es ignorado por el intérprete de NetLogo. Esto es útil para añadir comentarios a su código – texto que explica el código a los lectores humanos. Se pueden agregar algunos punto y coma extras para efectos visuales.

El menú Edit de NetLogo tiene items que permiten comentar o descomentar secciones enteras de código.

---

sentence
se

sentence valor1 valor2
(sentence valor1 ...)

Construye una lista a partir de los valores de entrada. Si alguno de los valores es una lista, sus miembros se incluyen directamente en la lista resultante, en vez de ser incluidos como una sublista. Los siguientes ejemplo dejarán esto más claro:

```
show sentence 1 2
=> [1 2]
show sentence [1 2] 3
=> [1 2 3]
show sentence 1 [2 3]
=> [1 2 3]
show sentence [1 2] [3 4]
=> [1 2 3 4]
show sentence [[1 2]] [[3 4]]
=> [[1 2] [3 4]]
show (sentence [1 2] 3 [4 5] (3 + 3) 7)
=> [1 2 3 4 5 6 7]
```

---

set
set variable valor

Asigna la entrada valor a la variable.

La variable puede ser cualquiera de las siguientes:

```
- Una variable global declarada usando ‘‘globals’’.
- La variable global asociada a un deslizador, interruptor,
  selecciondor o caja de entrada.
- Una variable que pertenece a este agente.
- Si este agente es una tortuga, una variable que pertenece
  a la parcela debajo de la tortuga.
- Una variable local creada por el comando let.
- Una entrada al procedimiento actual.
```

---

set-current-directory

set-current-directory cadena

Asigna el directorio actual al usado por las primitivas file-delete, file-exists? y file-open.

El directorio actual no se usa si a los comandos anteriores se les suministra la ruta completa, la cual se usará por defecto como directorio raíz para los nuevos modelos y se cambia al directorio del modelo cuando un modelo se abre.

Note en Windows para escribir la barra inclinada inversa dentro de una cadena es necesario usar otra barra inclinada inversa:

```
"C:\\"
```

El cambio es temporal y no se guarda con el modelo.

Nota: en applets este comando no tiene efecto, ya que a los applets sólo se les permiten leer archivos del mismo directorio en el servidor donde se guarda el modelo.

```
set-current-directory "C:\\NetLogo"
;; Supongamos que se trata de una computadora Windows
file-open "mi-archivo.txt"
```

```
;; Abre el archivo "C:\\NetLogo\\mi-archivo.txt"
```

---

set-current-plot

set-current-plot nombre-de-gráfico

Asigna el gráfico actual al gráfico con el nombre dado (una cadena). Subsecuentes comandos de graficación afectarán al gráfico actual.

---

set-current-plot-pen

set-current-plot-pen nombre-de-pluma

La pluma actual del gráfico se asigna a la pluma llamada nombre-de-pluma (una cadena). Si no existe tal pluma en el gráfico actual, se produce un error de ejecución (runtime error).

---

set-default-shape

set-default-shape turtles cadena
set-default-shape ligas cadena
set-default-shape familia cadena

Especifica una figura inicial por defecto para todas las tortugas o ligas, o para alguna familia particular de tortugas o de ligas. Cuando se crea una tortuga o liga o cuando se cambia de familia, su figura es fijada en el valor de la figura dada:

Este comando no afecta los agentes que ya existen, sólo aquellos que se crean después.

La familia dada debe ser de tortugas o ligas o el nombre de una familia. La cadena dada debe correponder al nombre de una figura definida actualmente.

En modelos nuevos, la figura por defecto de todas las tortugas es “default”.

Note que especificar una figura por defecto no le impide a usted cambiar la figura de un agente posteriormente. Los agentes no tienen por qué quedar atados al valor por defecto de la familia a la que pertenecen.

```
create-turtles 1 ;; la figura de la nueva tortuga es ‘‘default’’
create-cats 1    ;; la figura de la nueva tortuga es ‘‘default’’
set-default-shape turtles "circle"
create-turtles 1 ;; la figura de la nueva tortuga es ‘‘circle’’
create-cats 1    ;; la figura de la nueva tortuga es ‘‘circle’’

set-default-shape cats "cat"
set-default-shape dogs "dog"
create-cats 1   ;; la figura de la nueva tortuga es ‘‘cat’’ (gato)
ask cats [ set breed dogs ]
  ;; todos los cats se vuelven dogs (perros) y automáticamente
  ;; cambian su aspecto a ‘‘dog’’
```

Ver también shape.

---

set-histogram-num-bars

set-histogram-num-bars número

Asigna el valor actual del intervalo de pluma de graficar de modo que, dado el rango x para el gráfico, se dibujará la cantidad de barras dada por número cuando se llame al comando histogram.

Ver también histogram.

_set-line-thickness
_set-line-thickness número

Especifica el grosor de las líneas y el contorno de los elementos en la figura de la tortuga.

El valor por defecto es 0. Esto produce siempre líneas de un pixel de grosor.

Valores no cero se interpretan como el grosor medido en parcelas. Un grosor de 1, por ejemplo, produce líneas con el grosor de una parcela. Es común usar valores más pequeños como 0.5 or 0.2.

Las líneas tienen siempre un grosor de al menos un pixel.

Este comando es experimental y podría cambiar en futuras versiones.

---

set-patch-size
set-patch-size tamaño

Fija el tamaño de las parcelas en pixeles en la vista. El tamaño es típicamente un entero, pero también podría ser un número de punto flotante.

Ver también patch-size, resize-world.

---

set-plot-pen-color
set-plot-pen-color número

Asigna el color de la pluma de graficar actual al valor dado por número.

---

set-plot-pen-interval
set-plot-pen-interval número

Le indica al gráfico actual moverse una distancia dada por número en la dirección x, cada vez que se usa el comando plot. El intervalo de la pluma de graficar también afecta el comportamiento del comando histogram.

---

set-plot-pen-mode
set-plot-pen-mode número

Fija el modo como dibuja la pluma de graficar según número. Los modos permitidas para la pluma de graficar son:
0 (modo de línea) la pluma dibuja una línea conectando dos puntos.
1 (modo de barra): la pluma dibuja una barra de anchura plot-pen-interval con el punto graficado en la esquina superior izquierda de la barra (inferior izquierda si está graficando un número negativo).
2 (modo de punto): la pluma dibuja un punto en el lugar indicado. Los puntos no se interconectan.
El modo por defecto es 0 (modo de línea).

---

setup-plots
setup-plots

Para cada gráfico correr los comandos de configuración de ese gráfico, incluyendo el código que configura cualesquiera plumas que haya en el gráfico.

reset-ticks tiene el mismo efecto, de modo que en los modelos en que se usa el contador de ticks, esta primitiva normalmente no se usa.

Ver la sección de graficación (the Plotting section) de la Guía del Programa-

dor para más detalles.

Ver también update-plots.

---

set-plot-x-range
set-plot-y-range

set-plot-x-range min max
set-plot-y-range min max

Fija el mínimo y el máximo valor de los ejes x o y del gráfico actual.

El cambio es temporal si no se guarda con el modelo. Cuando el gráfico se borra, los rangos se reestablecen a sus valores por defecto, como se encuentran en le ventana Edit del gráfico.

---

setxy
setxy x y

La tortuga fija su coordenada x y su coordenada y en los valores de entrada x, y.

Equivale a xcor x y ycor y, excepto que en una sola órden en vez de en dos.

Si x o y está fuera del mundo, NetLogo mostrará un error de ejecución (runtime error), a menos que alguna topología de enlazamiento esté vigente. Por ejemplo, con la topología de la rosquilla y las dimensiones por defecto del mundo min-pxcor = -16, max-pxcor = 16, min-pycor = -16 and max-pycor = 16, si se le pide a la tortuga setxy 17 17, ésta se moverá al centro de la parcela (-16, -16).

```
setxy 0 0
;; la tortuga se mueve al centro de la parcela (0, 0)
setxy random-xcor random-ycor
;; la tortuga se mueve a un punto al azar
setxy random-pxcor random-pycor
;; la tortuga se mueve al centro de una parcela al azar
```

Ver también move-to.

---

shade-of?
shade-of? color1 color2

Reporta true (verdadero) si ambos colores son sombra el uno del otro, falso en caso contrario.

```
show shade-of? blue red
=> false
show shade-of? blue (blue + 1)
=> true
show shade-of? gray white
=> true
```

---

shape
shape

Esta es una variable preinstalada de las tortugas y las ligas, la cual contiene una cadena que es el nombre de la figura de la tortuga o liga. Se puede reasignar el valor de esta variable para cambiar la figura. Las tortugas o ligas nuevas adquiren la figura por defecto, a menos que se haya especificado una figura diferente usando la primitiva set-default-shape.

Ejemplo:

```
ask turtles [ set shape "lobo" ]
;; se supone que usted ha hecho un figura "lobo"
;; en en editor de figuras (Turtle Shapes Editor)
ask links [ set shape "liga 1" ]
```

```
;; se supone que usted a hecho una figura de ligas
;; "liga 1" en el editor de ligas (Link Shapes Editor)
```

Ver también set-default-shape, shapes.

---

shapes
shapes

Reporta una lista de cadenas que contiene los nombres de todas las figuras del modelo.

Se pueden crear nuevas figuras o importarlas de la biblioteca de figuras o de otros modelos en el editor de figuras.

```
show shapes
=> ["default" "airplane" "arrow" "box" "bug" ...
ask turtles [ set shape one-of shapes ]
```

---

show
show valor

Imprime valor en la Terminal de instrucciones, precedido por el agente solicitante y seguido de un retorno de carro. El agente solicitante se incluye para ayudar a llevar la cuenta de cuáles agentes están produciendo cuáles líneas de resultados. Todas las cadenas tienen sus comillas incluidas, de forma similar a write.

Ver también print, type y write y output-show.

---

show-turtle
st
show-turtle

La tortuga se vuelve de nuevo visible.

Nota: Este comando es equivalente a fijar la variable "hidden?"(escondida?) en el valor false.

Ver también hide-turtle.

---

show-link
show-link

La liga se vuelve de nuevo visible.

Nota: Este comando es equivalente a fijar la variable de liga "hidden?" en el valor false.

Ver también hide-link.

---

shuffle
shuffle lista

Reporta una nueva lista que contiene los mismos ítemes de la lista de entrada, pero barajados en un orden al azar.

```
show shuffle [1 2 3 4 5]
=> [5 2 4 1 3]
show shuffle [1 2 3 4 5]
=> [1 3 5 2 4]
```

---

sin
sin número

Reporta el seno del ángulo dado. Se supone que el ángulo se da en grados.

```
show sin 270
=> -1
```

---

size

size

Esta es una variable presinstalada de la tortuga. Almacena el número del tamaño aparente de la tortuga. El tamaño por defecto es 1, lo que significa que la tortuga tiene el mismo tamaño que las parcelas. Se puede reasignar el valor de esta variable para cambiar el tamaño de la tortuga.

---

sort

sort lista
sort conjunto-agentes

Reporta una lista reordenada de números, cadenas o agentes.

Si la entrada no contiene números, cadenas o agentes, el resultado es la lista vacía.

Si la entrada contiene al menos un número, los números de la lista se reordenan en orden ascendente y se reporta esa nueva lista, donde los ítemes no numéricos son ignorados.

Si la entrada contiene al menos una cadena (pero no números), las cadenas se reordenan en orden ascendente y se reporta esa nueva lista. Los ítemes que no son cadenas son ignorados.

Si la entrada es un conjunto agentes o una lista que contiene al menos un agente, los agentes de la lista se reordenan en orden ascendente y se reporta esa nueva lista (nunca un conjunto-agentes). Los no agentes son ignorados. Los agentes se reordenan en el mismo orden dado por el operador <.

```
show sort [3 1 4 2]
=> [1 2 3 4]
let n 0
foreach sort patches [
  ask ? [
    set plabel n
    set n n + 1
  ]
]
;; las parcelas se etiquetan con números en orden
;; de izquierda a derecha y de arriba hacia abajo
```

Ver también sort-by, sort-on

---

sort-by

sort-by tarea-reportadora lista
sort-by tarea-reportadora conjunto-agentes

Si la entrada es una lista, reporta una nueva lista con los mismos ítemes, en una reordenación dada por la tarea reportadora booleana.

Las dos entradas de la tarea reportadora son los valores que se comparan. La tarea debe reportar verdadero (true) si ?1 viene estricamente antes que ?2 en la reordenación deseada y falso en caso contrario.

Si la entrada es un conjunto-agentes o una lista de agentes, reporta una lista de agentes (nunca un conjunto-agentes).

Si la entrada es una lista, la reordenación es estable, es decir, el orden de íte-

mes que se consideran iguales por la reportadora no se altera. Si la entrada es un conjunto-agentes, las ataduras se rompen al azar.

```
show sort-by < [3 1 4 2]
=> [1 2 3 4]
show sort-by > [3 1 4 2]
=> [4 3 2 1]
show sort-by [length ?1 < length ?2] ["contento" "dama" "verano"]
=> ["dama" "verano" "contento"]
```

Ver también sort, sort-on.

---

sort-on

sort-on [reportadora] conjunto-agentes

Reporta una lista de agentes reordenados según el valor de la reportadora para cada agente. Las ataduras se rompen al azar.

Todos los valores deben ser números o cadenas o bien agentes del mismo tipo.

```
crt 3
show sort-on [who] [turtles]
=> [(turtle 0) (turtle 1) (turtle 2)]
show sort-on [(- who)] [turtles]
=> [(turtle 2) (turtle 1) (turtle 0)]
foreach sort-on [size] turtles
  [ ask ? [ hacer-algo ] ]
;; las tortugas ejecutan "hacer-algo" una a la vez en
;; orden de tamaño ascendente
```

Ver también sort, sort-by.

---

sprout

sprout <breeds>

sprout número [**comandos**]
sprout-<breeds> número [**comandos**]

Crea una cantidad de tortugas dada por número en la parcela actual. Las nuevas tortugas tienen orientaciónes al azar dadas por número enteros y su color se selecciona al azar de entre los 14 colores primarios. Las tortugas pueden ejecutar comandos inmediatamente. Esto es útil para dar a las nuevas tortugas colores u otras características diferentes. Las nuevas tortugas se crean todas de una sola vez pero luego ejecutan las órdenes una a la vez, en orden aleatorio.

Si se usa la forma sprout-<breeds>, las nuevas tortugas se crean como miembros de la familia (breed) dada.

```
sprout 5
sprout-lobos 10
sprout 1 [ set color red ]
sprout-oveja 1 [ set color black ]
```

Ver también create-turtles, hatch.

---

sqrt
sqrt número

Reporta la raíz cuadrada del número dado como entrada.

---

stamp
stamp

La tortuga o liga deja una imagen de su figura en el dibujo, en el lugar donde se encuentra en ese momento.

Nota: Las figuras estampadas por stamp pueden no ser idénticas pixel por pixel de una computadora a otra.

---

stamp-erase
stamp-erase

La tortuga o liga remueve cualquier pixel del dibujo debajo de ella, el cual se encuentre dentro de el contorno de su figura.

Nota: Las figuras removidas por stamp-erase pueden no ser idénticas pixel por pixel de una computadora a otra.

---

standard-deviation
standard-deviation lista

Reporta la desviación estándar de la muestra de una lista de números. Ignora otro tipo de ítemes.

Note que estima la desviación estándar de una muestra, en vez de toda una población, usando la correción de Bessel.

```
show standard-deviation [1 2 3 4 5 6]
=> 1.8708286933869707
show standard-deviation [energía] de las tortugas
;; imprime la desviación estándar de la variable "energía"
;; de todas las tortugas
```

---

startup
startup

Procedimiento que puede definir el usuario, el cual sería el primero en ser ejecutado cuando se carga el modelo.

```
to startup
  setup
end
```

El procedimiento startup no se ejecuta cuando un modelo se corre "decapitado" (headless), desde la Ventana del Observador o en paralelo en el BehaviorSpace.

---

stop
stop

El agente sale inmediatamente del procedimiento que lo contiene, dentro de un ask o ask-like (o bien crt, hatch, sprout). Sólo el procedimiento actual se detiene, no toda ejecución para el agente.

```
if not any? turtles [ stop ]
;; sale del procedimiento si no hay más tortugas.
```

Nota: stop se puede usar para detener un botón de tipo continuamente o para siempre (forever button). Si el botón continuamente llama directamente a un procedimiento, entonces cuando ese procedimiento se detiene, el botón se detiene. En el caso del botón de tipo continuamente de una tortuga o parcela, el botón no se detendrá hasta que cada tortuga o parcela se detenga – una tortuga o parcela no tiene el poder de detener todo el botón.

También se puede usar para detener la corrida de un modelo en el BehaviorSpace. Si los comandos de go llaman directamente a un procedimiento, entonces cuando termine el procedimiento, la corrida se detiene.

---

stop-inspecting
stop-inspecting agente

Cierra el monitor del agente dado (tortuga o parcela). En caso de que no haya ningún monitor de agente abierto, stop-inspecting no hace nada.

```
stop-inspecting patch 2 4
;; el monitor de esta parcela se cierra
ask oveja [stop-inspecting self]
;; cierra todos los monitores de agente para oveja
```

---

stop-inspecting-dead-agents
stop-inspecting-dead-agents

Cierra todos los monitores para los agentes muertos.

Ver inspect y stop-inspecting

---

subject
subject

Reporta la tortuga o parcela que el observador está observando, siguiendo o cabalgando. Reporta nobody (nadie) si no hay tal tortuga o parcela.

Ver también watch, follow, ride.

---

sublist
substring

sublist lista posición1 posición2
sustring cadena posición1 posición2

Reporta sólo una sección de la lista o cadena dada, que va de la primera posición (incluida) a la segunda posición (excluida).

Nota: Las posiciones se enumeran comenzando a partir de 0, no de 1.

```
show sublist [99 88 77 66] 1 3
=> [88 77]
show substring "apartar" 1 5
=> "part"
```

---

subtract-headings

subtract-headings orientación1 orientación2

Calcula la diferencia entre las orientaciones dadas, es decir, el número de grados del menor ángulo mediante el cual encabezado2 se podría rotar para producir encabezado1. Una respuesta positiva significa una rotación en sentido horario y negativa en sentido anti-horario. El resultado está siempre en el rango de -180 a 180, pero nunca es exactamente -180.

Note que sustraer las dos orientaciones simplemente usando el operador - (menos) no funcionará. Sustraer simplemente corresponde a rotar en sentido horario de orientación2 a orientación1; pero a veces la rotación en sentido anti-horario es más corta. Por ejemplo, la diferencia entre 5 y 355 grados es 10 grados, no -350 grados.

```
show subtract-headings 80 60
=> 20
show subtract-headings 60 80
=> -20
show subtract-headings 5 355
=> 10
show subtract-headings 355 5
=> -10
show subtract-headings 180 0
=> 180
show subtract-headings 0 180
=> 180
```

---

sum
sum lista

Reporta la suma de los ítemes en la lista.

```
show sum [energía] of turtles
;; imprime la suma total de la variable "energía"
;; de todas las tortugas
```

T

---

tan

tan número

Reporta la tangente del ángulo dado por número. Se asume que el ángulo se da en grados.

---

task
task [comandos]
task [reportadora]
task nombre-de-comando
task nombre-de-reportadora

Crea y reporta una tarea, ya sea como comando o como tarea reportadora, dependiendo de la entrada.

Ver la sección Tasks en la Guía de Progamación (Programming Guide) para más detalles.

---

thickness
thickness

Esta es una variable preinstalada de las ligas. Almacena un número que da la anchura aparente de las ligas como fracción del tamaño de una parcela. La anchura por defecto es 0, lo cual significa que, independientemente del tamaño de las parcelas, la liga aparecerá siempre con una anchura de 1 pixel. Se puede reasignar el valor de esta variable para cambiar la anchura de las ligas.

---

tick
tick

Avanza el contador de ticks en una unidad y actualiza todos los gráficos.

Si el contador de ticks no ha sido inicializado con la primitiva reset-ticks, ocurre un error.

Normalmente tick va al final del procedimiento go.

Ver también ticks, tick-advance, reset-ticks, clear-ticks, update-plots.

---

tick-advance
tick-advance número

Avanza el contador de ticks en número. La entrada podría ser un entero o un número de punto flotante (algunos modelos dividen los ticks más finamente que por unidades). La entrada no debe ser negativa.

Cuando se usan actualizaciones de la vista basadas en ticks, la vista se actualiza normalmente cada 1.0 ticks, de manera que usar tick-advance con un número menor 1.0 no siempre disparará una actualización. Si usted quiere asegurarse que la vista se actualice, puede usar el comando display.

Si el contador de ticks no se ha inicializado mediante reset-ticks, ocurrirá un error.

No actualiza los gráficos.

Ver también tick, ticks, reset-ticks, clear-ticks.

---

ticks
ticks

Reporta el valor actual del contador de ticks. El resultado es siempre un número, el cual nunca es negativo.

Si el contador de ticks no ha sido iniciado con reset-ticks, el resultado es un error.

La mayoría de los modelos emplean el comando tick para avanzar el contador de ticks, en cuyo caso ticks reporta siempre un entero. Si se emplea el comando tick-advance entonces ticks podría reportar un número de punto flotante.

Ver también tick, tick-advance, reset-ticks, clear-ticks.

---

tie
tie

Ata los extremos extremo1 y extremo2 de la liga. Si la liga es dirigida, extremo1 es la tortuga raíz y extremo2 es la tortuga hoja. El movimiento de la tortuga raíz afecta la posición y orientación de la tortuga hoja. Si la liga es no dirigida, la atadura es recíproca de modo que ambas tortugas se pueden considerar tortuga raíz y tortuga hoja. Un cambio en la posición u orientación de cualquiera de las tortugas afecta la posición y orientación de la otra tortuga.

Cuando la tortuga raíz se mueve, la tortuga hoja se mueve la misma distancia en la misma dirección. La orientación de la tortuga hoja no es afectada. Esto trabaja con forward, jump y reasignar la xcor o la ycor de la tortuga raíz.

Cuando la tortuga raíz gira a la derecha o izquierda, la tortuga hoja gira alrededor de la tortuga raíz la misma cantidad. La orientación de la tortuga raíz también cambia en la misma cantidad.

Si la liga muere, la relación de atadura es removida.

```
crt 2 [ fd 3 ]
;; crea una liga y ata la tortuga 1 a la tortuga 0
ask turtle 0 [ create-link-to turtle 1 [ tie ] ]
```

Ver también untie

---

tie-mode
tie-mode

Esta es una variable preinstalada de las ligas, la cual almacena una cadena con el estado que tiene la liga en ese momento. El uso de los comandos tie y untie cambia el estado de la liga. Se puede fijar tie-mode en “free” (libre) para crear una unión no rígida entre dos tortugas (ver la sección Tie de la Guía de Programación para mayores detalles). Por defecto las ligas no se encuentran en estado de atadura.

Ver también: tie, untie

---

timer

timer

Reporta cuántos segundos han pasado desde que se corrió por última vez el comando reset-timer (o desde que se inició NetLogo). La resolución potencial del reloj es de milisegundos (el que usted obtenga una resolución tan alta, en la práctica varía de sistema a sistema, dependiendo de las capacidades de la Máquina Virtual de Java subyacente).

Ver también reset-timer.

Note que timer es diferente que el contador de ticks. Timer mide en segundos el tiempo real transcurrido; el contador de ticks mide el tiempo transcurrido del modelo en ticks.

---

to

to nombre-de-procedimiento
to nombre-de-procedimiento [entrada1...]

Primitiva usada para iniciar un procedimiento de comandos (no reportador).

```
to setup
  clear-all
  crt 500
end

to circle [radius]
  crt 100 [ fd radius ]
end
```

---

to-report

to-report nombre-de-procedimiento
to-report nombre-de-procedimiento [entrada1...]

Primitiva usada para iniciar un procedimiento reportador.

El cuerpo del procedimiento debe usar la primitiva report para reportar un valor a otro procedimiento. Ver report.

```
to-report average [a b]
  report (a + b) / 2
end

to-report absolute-value [number]
  ifelse number >= 0
    [ report number ]
    [ report (- number) ]
end

to-report first-turtle?
  report who = 0  ;; reporta true o false
end
```

---

towards

towards agente

Reporta la orientación del agente al agente dado como entrada.

En las topologías de enlazamiento (rosquilla y cilindro), si la distancia a través de los bordes (aparentes) es la menor, entonces se usa esa distancia.

Nota: preguntar por la orientación de un agente a sí mismo o a un agente en

la misma ubicación causará un error de ejecución (runtime error).

```
set heading towards turtle 1
;; es lo mismo que "face turtle 1"
```

Ver tambiéno face.

---

towardsxy

towardsxy x y

Reporta la orientación desde la tortuga o parcela hacia el punto (x, y).

En las topologías de enlazamiento (rosquilla o cilindro), si la distancia a través de los bordes (aparentes) es la menor, entonces se usa esa trayectoria.

Nota: preguntar por la orientación hacia el punto en donde se encuentra el agente causará un error de ejecución (runtime error).

Ver también facexy.

---

turtle

turtle número
<familia> número

Reporta la tortuga con el número dado o nadie (nobody) si no existe tal tortuga. Para tortugas de una familia se puede también usar la forma simple "breed" para referirse a las tortugas.

```
ask turtle 5 [ set color red ]
;; la tortuga con número who 5 se vuelve roja
```

---

turtle-set
turtle-set valor1
turtle-set valor1 valor2...

Reporta un conjunto-agentes que contiene todas las tortugas en cualquier parte de cualquiera de las entradas. Las entradas pueden ser tortugas individuales, conjunto-agentes de tortugas, nobody (nadie) o listas (simples o anidadas) que contengan cualquiera de los anteriores.

```
turtle-set self
(turtle-set self turtles-on neighbors)
(turtle-set turtle 0 turtle 2 turtle 9)
(turtle-set sapos ratones)
```

Ver también patch-set, link-set.

---

turtles
turtles

Reporta el conjunto-agentes que consiste de todas las tortugas.

```
show count turtles
;; imprime el número de tortugas
```

---

turtles-at
<breeds-at>

turtles-at dx dy
<breeds-at> dx dy

Reporta un conjunto-agentes que contiene todas las tortugas sobre la parcela que se encuetra desplazada (dx, dy) unidades del agente solicitante. El resultado podría incluir a la solicitante misma, si la solicitante es una tortuga.

```
create-turtles 5 [ setxy 2 3 ]
```

```
show count [turtles-at 1 1] of patch 1 2
=> 5
```

Si se reemplaza el nombre de una familia por el de “turtle”, entonces sólo se incluyen las tortugas de esa familia.

---

turtles-here
<breeds>-here

turtles-here
<breeds-here>

Reporta un conjunto-agentes que contiene todas las tortugas sobre la parcela donde se encuentra el agente solicitante (incluyendo a la solicitante misma, caso de ser una tortuga).

```
crt 10
ask turtle 0 [ show count turtles-here ]
=> 10
```

Si el nombre de una familia se sustituye por el de ”turtles”, entonces sólo se incluyen las tortugas de esa familia.

```
breed [gatos gato]
breed [perros perro]
create-gatos 5
create-perros 1
ask perros [ show count gatos-here ]
=> 5
```

---

turtles-on
<breeds-on>

turtles-on agente
turtles-on conjunto-agente
<breeds>-on agente
<breeds>-on conjunto-agentes

Reporta un conjunto-agentes que contiene todas las tortugas que están en la parcela o parcelas sobre las que se encuentra la tortuga o tortugas dadas.

```
ask turtles [
  if not any? turtles-on patch-ahead 1
    [ fd 1 ]
]
ask turtles [
  if not any? turtles-on neighbors [
    morir-de-soledad
  ]
]
```

Si se reemplaza el nombre de una familia (breed) por el de "turtles", entonces sólo se incluyen las tortugas de esa familia.

---

turtles-own
<breeds>-own
turtles-own [**var1 ...**]
<breeds>-own [**var1 ...**]

La primitiva turtles-own, al igual que globals, breed, <breeds>-own, and patches-own, sólo se puede usar al comienzo de un programa, antes de la definición de los procedimientos. Define las variables que pertenecen a cada tortuga.

Si se especifica una familia (breed) en vez de "turtles", sólo las tortugas de esa familia poseerán las variables enlistadas. Más de una familia de tortugas

podría enlistar la misma variable.

```
breed [gatos gato]
breed [perros perro]
breed [conejos conejo]
turtles-own [ojos patas]   ;; se aplica a todas las familias
gatos-own [piel juguetes]
conejos-own [piel jaula]
perros-own [pelo bolas]
```

Ver también globals, patches-own, breed, <breeds>-own.

---

type
type valor

Imprime el valor dado en la Terminal de Instrucciones, seguido de un retorno de carro (contrariamente a print y show). La ausencia de un retorno de carro le permite imprimir varios valores en la misma línea.
El agente solicitante no se imprime antes del valor, contrariamente a show.

```
type 3 type " " print 4
=> 3 4
```

Ver también print, show y write. Ver también output-type

U

---

undirected-link-breed

undirected-link-breed [<**link-breeds**><**link-breed**>]

Esta primitiva, al igual que globals y breeds, sólo se puede usar al comienzo del área de código, antes de la definición de los procedimientos. Define una familia de ligas no dirigidas. Las ligas de una familia particular son siempre de tipo dirigido o no dirigido. La primera entrada define el nombre del conjunto-agentes asociado con la familia de ligas. La segunda entrada define el nombre de un miembro individual de dicha familia.

Cualquier liga de la familia de ligas dada:
es parte del conjunto-agentes nombrado por el nombre de la familia de ligas,
tiene su conjunto de variables de ligas fijadas en ese conjunto agentes,
es dirigida o no dirigida, según se ha declarado por la primitiva,
la mayoría de las veces, el conjunto agente se usa en conjunción con el comando ask para dar órdenes sólo a las ligas de una familia en particular.

```
undirected-link-breed [calles calle]
undirected-link-breed [autopistas autopista]
to setup
  clear-all
  crt 2
  ask turtle 0 [ create-calle-with turtle 1 ]
  ask turtle 0 [ create-autopista-with turtle 1 ]
end

ask turtle 0 [ show sort my-links ]
;; imprime [(calle 0 1) (autopista 0 1)]
```

Ver también breed, directed-link-breed

---

untie
untie

Desata el extremo2 del extremo1 (fija tie-mode en “none” (ninguno) si existía antes una atadura). Si la liga es no dirigida, entonces también desata el extremos 1 del extremo2. No elimina la liga entre las dos tortugas.

Ver también tie. Ver la sección Tie de la Guía del Programador (Program-

ming Guide) para más detalles.

---

update-plots
update-plots

Para cada gráfico, ejecuta los comandos de actualización del gráfico, incluyendo el código de actualización de cualquiera de sus plumas.

tick tiene el mismo efecto, de modo que en los modelos que usan el contador de ticks, esta primitiva normalmente no se usa. Modelos que usan ticks fraccionados podrían necesitar update-plots, pues tick-advance no actualiza los gráficos.

Ver la sección de graficación (Plotting section) de la Guía del Programador (Programming Guide) para más detalles.

Ver también setup-plots.

---

uphill
up-hill4
uphill variable-de-parcela
uphill4 variable-de-parcela

Mueve la tortuga a la parcela vecina con el valor más alto de variable-de-parcela. Si ninguna parcela vecina tiene un valor más alto que el de la parcela presente, la tortuga permanece sin moverse. Si hay varias parcelas con el mismo valor más alto, la tortuga escoge una parcela al azar. Los valores no numéricos son ignorados.

uphill considera las ocho parcelas vecinas; uphill4 sólo considera las cuatro vecinas.

Equivale al código siguiente (suponiendo que los valores de la variable son numéricos):

```
move-to patch-here  ;; ir al centro de la parcela
let p max-one-of neighbors [patch-variable]  ;; o neighbors4
if [patch-variable] of p > patch-variable [
  face p
  move-to p
]
```

Note que la trotuga siempre termina en el centro de una parcela y con una orientación de 45 (uphill) o 90 (uphill4).

Ver también downhill, downhill4.

---

user-directory
user-directory

Abre una ventana de diálogo que permite al usuario escoger uno de los directorios existentes en el sistema.

Reporta una cadena con la ruta absoluta o falso si el usuario escoge cancelar.

```
set-current-directory user-directory
;; Supone que el usuario escogerá un directorio
```

---

user-file
user-file

Abre una ventana de diálogo que permite al usuario escoger uno de los archivos existentes en el sistema.

Reporta una cadena con la ruta absoluta del archivo o falso si el usuario

escoge cancelar.

```
file-open user-file
;; Supone que el usuario escogerá un archivo
```

---

user-new-file
user-new-file

Abre una ventana de diálogo que permite al usuario escoger una ubicación y un nombre para un nuevo archivo que será creado. Reporta una cadena con la ruta absoluta o falso si el usuario escoge cancelar.

```
file-open user-new-file
;; Supone que el usuario escogerá un archivo
```

Nota: esta reportadora realmente no crea un archivo; normalmente se crea un archivo usando file-open, como en el ejemplo.

Si el usuario escoge un archivo existente, se le preguntará si desea reemplazarlo o no, pero la reportadora misma no ocasiona que el archivo sea reemplazado. Para hacer esto se debe usar file-delete.

---

user-input
user-input valor

Reporta la cadena que el usuario escribe en un campo de una ventana de diálogo, con el título dado por la entrada "valor". Esta entrada puede se de cualquier tipo pero típicamente es una cadena.

```
show user-input "¿Cuál es tu nombre?"
```

---

user-message
user-message valor

Abre una ventana de diálogo que contiene el mensaje.

La entrada “valor” puede ser de culaquier tipo pero típicamente es una cadena.

```
user-message (word "Hay " count turtles " tortugas.")
```

---

user-one-of

user-one-of valor lista-de-opciones

Abre una ventana de diálogo que contiene el mensaje y la lista de opciones se despliega como un menú emergente en el que el usuario hará su elección.

Reporta el item en la lista-de-opciones seleccionado por el usuario.

La entrada “valor” puede ser de cualquier tipo pero típicamente es una cadena.

```
if "sí" = user-one-of "¿Instalar el modelo?" ["sí" "no"]
  [ setup ]
```

---

user-yes-or-no?

user-yes-or-no? valor

Reporta true o false (verdadero o falso) basándose en la respuesta del usuario a la entrada “valor”.

La entrada “valor” puede ser de cualquier tipo pero típicamente es una cadena.

```
if user-yes-or-no? "¿Instalar el modelo?"
  [ setup ]
```

V

---

variance

variance lista

Reporta la varianza de la muestra de una lista de números. Ignora otro tipo de ítemes.

(Note que este cálculo es una estimación no sesgada de la varianza de una muestra, y no de toda una población, usando la corrección de Bessel.)

La varianza de la muestra es la suma de los cuadrados de las desviaciones de la media de los números de la lista, dividido la cantidad de números de la lista disminuida en una unidad.

```
show variance [2 7 4 3 5]
=> 3.7
```

W

---

wait

wait número

Produce una espera de una cantidad de segundos dada por la entrada número (número no debe ser necesariamente un entero; se pueden especificar fracciones de segundo). Note que no se puede esperar una precisión completa: el agente nunca esperará menos de la cantidad dada, pero podría ser ligeramente mayor.

```
repeat 10 [ fd 1 wait 0.5 ]
```

Mientras el agente está esperando, ningún otro agente puede hacer nada. Todo se detiene hasta que la espera haya terminado.

Ver también every

---

watch
watch agente

Pone una luz de rastreo sobre un agente. En la vista 3D el observador se voltea de cara al agente.

Ver también follow, subject, reset-perspective, watch-me.

---

watch-me
watch-me

Le pide al abservador que observe al agente actual.

Ver también watch.

---

while
while [reportadora][comandos]

Si la reportadora arroja falso, sale del bucle de comandos. De otra manera ejecuta los comandos y los repite.

La reportadora puede arrojar diferentes valores para agentes distintos, de manera que algunos agentes podrían correr los comandos un número diferen-

te de veces que otros agentes.

```
while [any? other turtles-here]
  [ fd 1 ]
;; la tortuga avanza hasta encontrar una parcela
;; en la que no hay ninguna otra tortuga
```

---

who
who

Los números who comienzan en 0. El número who de una tortuga muerta no se reasignará a ningun otra tortuga sino hasta que se use el comando clear-turtles o clear-all, en cuyo caso la numeración vuelve a comenzar en 0. Ejemplo:

```
show [who] of turtles with [color = red]
;; imprime una lista de los números who de todas las tortugas rojas
;; en la Terminal de Instrucciones, en orden aleatorio
crt 100
  [ ifelse who < 50
      [ set color red ]
      [ set color blue ] ]
;; las tortugas 0 a la 49 son rojas, las tortuga de la 50 a
;; la 99 son azules
```

Se puede usar la reportadora turtle para recuperar una tortuga con un número who dado. Ver también turtle.
Note que los número who no son específicos de las familias. Dos tortugas distintas no pueden tener el mismo número who, aunque pertenezcan a familias distintas:

```
clear-turtles
create-sapos 1
create-ratones 1
ask turtles [ print who ]
;; imprime (en algún orden aleatorio):
;; (sapo 0): 0
;; (ratón 1): 1
```

Aunque sólo hay un ratón, es el ratón 1 no el ratón 0, porque el número who

0 ya estaba asignado a un sapo.

---

with
conjunto-agentes with [reportadora]

Opera con dos entradas: a la izquierda un conjunto-agentes (usualmente "turtles" o "patches") y a la derecha una reportadora booleana. Reporta un nuevo conjunto-agentes (subconjunto del conjunto-agentes dado como entrada) que contiene solamente aquellos agentes que reportaron true (verdadero) –en otras palabras, los agentes que satisfacen la condición dada.

```
show count patches with [pcolor = red]
;; imprime el  número de parcelas rojas.
```

<breed>-with
link-with

<breed>-with tortuga
link-with tortuga

Reporta la liga no dirigida entre la tortuga y el agente solicitante. Si la liga no existe entonces reporta nobody (nadie).

```
crt 2
ask turtle 0 [
  create-link-with turtle 1
  show link-with turtle 1 ;; imprime link 0 1
]
```

Ver también: in-link-from, out-link-to

---

with-max
conjunto-agentes with-max [reportadora]

Opera con dos entradas: a la izquierda un conjunto-agentes (usualmente "turtles" o "patches") y a la derecha una reportadora. Reporta un nuevo conjunto-agentes (subconjunto del conjunto-agentes dado como entrada) que contiene todos los agentes que reportan el valor máximo de la reportadora dada.

```
show count patches with-max [pxcor]
;; imprime el número de parcelas en el borde derecho (donde
;; el valor de la coordenada x es máximo)
```

Ver también max-one-of, max-n-of.

---

with-min

conjunto-agentes with-min [reportadora]

Opera con dos entradas: a la izquierda un conjunto-agentes (usualmente "turtles" o "patches") y a la derecha una reportadora. Reporta un nuevo conjunto-agentes (subconjunto del conjunto-agentes dado como entrada) que contiene todos los agentes que reportan el valor mínimo de la reportadora dada.

```
show count patches with-min [pycor]
;; imprime el número de parcelas en el borde inferior (donde
;; el valor de la coordenada y es mínimo)
```

Ver también min-one-of, min-n-of.

---

with-local-randomness

with-local-randomness [comandos]

Los comandos se ejecutan sin afectar eventos aleatorios subsecuentes. Esto es útil para llevar acabo operaciones extra (tales como output) sin cambiar el resutado de un modelo.

Ejemplo:

```
;; Corrida #1:
random-seed 50 setup repeat 10 [ go ]
;; Corrida #2:
random-seed 50 setup
with-local-randomness [ watch one-of turtles ]
repeat 10 [ go ]
```

Puesto que one-of es usado dentro de with-local-randomness, ambas corridas serán idénticas.

Especificamente, la manera como funciona es que el estado del generador de números aleatorios es recordado antes de ejecutar los comandos y luego reestablecido. Si usted quisiera correr los comandos con un estado aleatorio fresco, en vez del mismo estado aleatorio que va a ser reestablecido, puede comenzar los comandos con random-seed new-seed.

El siguiente ejemplo demuestra que el generador de números aleatorios es el mismo antes y después de correr los comandos.

```
random-seed 10
with-local-randomness [ print n-values 10 [random 10] ]
;; imprime [8 9 8 4 2 4 5 4 7 9]
print n-values 10 [random 10]
;; imprime [8 9 8 4 2 4 5 4 7 9]
```

---

without-interruption

without-interruption [**comandos**]

Esta primitiva existe solamente para guardar compatibilidad con versiones anteriores. No recomendamos usarla en modelos nuevos.

El agente ejecuta todos los comandos en el bloque sin permitir que otros agentes que están usando ask-concurrent “interrumpan”. Es decir, otros agentes

son puestos “en espera” y no ejecutan ningún comando hasta que se hayan terminado de ejecutar los comandos en el bloque.

Nota: Este comando sólo tiene utilidad en conjunción con ask-concurrent.

Ver también ask-concurrent.

---

word

word valor1 valor2
(word valor1 ...)

Concatena las entradas y reporta el resultado como una cadena.

```
show word "tur" "tle"
=> "turtle"
word "a" 6
=> "a6"
set directory "c:\\foo\\pez\\"
show word directory "bar.txt"
=> "c:\foo\pez\bar.txt"
show word [1 54 8] "pescado"
=> "[1 54 8]pescado"
show (word 3)
=> "3"
show (word "a" "b" "c" 1 23)
=> "abc123"
```

---

world-width
world-height

world-width
world-height

Estas reportadoras dan la anchura (width) y la altura (height) total del mundo de NetLogo.

La anchura es igual a max-pxcor - min-pxcor + 1 y la altura es igual a max-pycor - min-pycor + 1.

Ver también max-pxcor, max-pycor, min-pxcor, and min-pycor

---

wrap-color

wrap-color número

wrap-color verifica si el número se encuentra en el rango de colores de NetLogo de 0 a 140 (excluyendo el 140). En caso de no encontrarse wrap-color "enlaza" ("wrap") la entrada número al rango de 0 a 140.

El enlazado se hace sumando o restando repetidamente 140 del número dado hasta obtener un resultado en el rango de 0 a 140. Este es el mismo proceso de enlazado que se hace automáticamente cuando se asigna un número fuera de rango (out-of-range) a la variable color de la tortuga o pcolor de la parcela.

```
show wrap-color 150
=> 10
show wrap-color -10
=> 130
```

---

write

write valor

Este comando envía a la Termina de instrucciones la entrada valor, la cual puede ser un número, cadena, lista, booleano o nobody (nadie) y no va se-

guido de un retorno de carro (a diferencia de print y show).

El agente solicitante no se imprimer antes de la entrada valor, a diferencia de show. El resultado también incluye comillas encerrando las cadenas y se antepone un espacio.

```
write "hola mundo"
=>  "hola mundo"
```

Ver también print, show y type. Ver también output-write.

X

---

xcor

xcor

Esta es una variable de la tortuga preinstalada en el sistema. Almacena el valor de la coordenada x de la tortuga. Se puede asignar su valor para cambiar la ubicación de la tortuga.

Esta variable es siempre mayor o igual a (min-pxcor - 0.5) y estrictamente menor que (max-pxcor + 0.5).

Ver también setxy, ycor, pxcor, pycor

---

xor
booleano1 xor booleano2

Reporta true (verdadero) si booleano1 o booleano2 es verdadero, pero no si ambos lo son.

```
if (pxcor > 0) xor (pycor > 0)
  [ set pcolor blue ]
;; los cuadrantes superior izquierdo e inferior derecho se vuelven azules
```

Y

---

ycor

ycor

Esta es una variable de la tortuga preinstalada en el sistema. Almacena el valor que tiene la coordenada y de la tortuga. Se puede asignar su valor para cambiar la ubicación de la tortuga.

Esta variable es siempre mayor o igual a (min-pycor - 0.5) y estrictamente menor que (max-pycor + 0.5).

Ver también setxy, ycor, pxcor, pycor

?

?, ?1, ?2, ?3 ...

?, ?1, ?2, ?3 ...

Estos nombres especiales de variables se refieren a las entradas de una tarea (task), ordenadas por número.

? es equivalente siempre a ?1.

Sus valores no se pueden asignar y no se usan excepto en el contexto de una tarea.

Las tareas se usan comúnmente con las primitivas foreach, map, reduce, filter, sort-by, y n-values. Consultar esas primitivas para ver ejemplos de sus usos.

Ver la sección de tareas de la Guía de Programación (Programming Guide) para más detalles.