LOGO GRAFICO®

Versión 4.4

DESCRIPCION DEL LENGUAJE

Edita FUNDAUSTRAL
Gavilán 2116 -1416- CAPITAL FEDERAL - REP. ARGENTINA
Tel. (54 11) 4581-2768
Email info@fundaustral.com.ar

www.fundaustral.com.ar

Descripción del Lenguaje

DESCRIPCIÓN DEL LENGUAJE

A continuación damos una descripción de esta versión de LOGO para que el usuario pueda conocer las características principales de la misma y su diferencia con otras implementaciones de este lenguaje.

Está dirigida a personas que tienen cierto conocimiento aunque sea elemental de conceptos tales como "procedimiento", "función", "variable", "instrucción", etc. adquirido por medio del trabajo con otras versiones de LOGO u otros lenguajes.

Los que no tengan claros estos conceptos pueden adquirirlos recurriendo a la extensa literatura que existe sobre el tema.

Para el uso del programa se debe consultar las ventanas de ayuda, en las cuales está también las descripción de todas las primitivas. Ésta se encuentra además en el archivo PRIMITIV.WRI incluido en el disco de instalación.

I.- ELEMENTOS DEL LENGUAJE LOGO

El lenguaje LOGO trabaja con "palabras" (también llamadas "átomos") que son conjuntos de caracteres sin blancos y "listas", que son un número finito de palabras dadas en un cierto orden que serán los elementos de estas listas. También pueden ser elementos de un lista otras listas. Se define

además la palabra vacía y la lista vacía.

Ejemplos:

Palabras o átomos:

```
abcd   *  -  /   |  A    2.85E-3  -4.0  ABC
```

Listas:

```
[abc 2 5 [a b] 4]    [1 2 3 4]
```

Las listas se indican empleando corchetes. Las palabras no pueden contener blancos ni corchetes, pues estos caracteres se emplean como separadores.

La palabra vacía se representa como " y la lista vacía se simboliza como [ ] .

Cuando en una lista se escriben dos espacios en blanco seguidos (con la barra espaciadora), LOGO los elimina pues entre un elemento (palabra) y otro este espacio actúa como separador, y alcanza con uno solo.

Si queremos expresamente dejar lugares en blanco podemos emplear la combinación de teclas Ctrl + B (de Blanco) que dejan ver en pantalla un puntito al escribirlo, pero luego no se ve. (ver Uso de los espacios en blanco en IV c.)

Por ejemplo:

```
[A B         C] Espacios con la barra se interpreta como [A B C]
[A B........C] (con Ctrl+B)  se entiende como           [A B         C]
```

Se distinguen mayúsculas de minúsculas en todos los nombres definidos por el usuario, aunque no en los nombres de las primitivas.

II.- IDENTIFICADORES

Los nombres de los procedimientos, funciones o variables, también llamados identificadores, deben contener sólo caracteres alfanuméricos (letras o números) y deben comenzar con una letra.

En el caso de que se los emplee como nombres de procedimientos o funciones, deberán seguir la misma regla que para los nombres de las variables o contener tan sólo "signos", que son los caracteres:

```
+ - / * | = < > & ^ _
```

Se consideran alfanuméricos las letras mayúsculas o minúsculas, los dígitos numéricos y los caracteres:

```
. : " ? @ $ % _ á é í ó ú ñ Ñ ü
```

No pueden mezclarse en un identificador caracteres alfanuméricos y signos mezclados, excepto en el identificador de asignación := .

Los identificadores pueden contener hasta 31 caracteres.

No deben incluir ni paréntesis ni corchetes.

III.- INSTRUCCIONES

Las instrucciones LOGO consisten en nombres de procedimientos seguidos por sus argumentos u operandos (valores sobre los que el procedimiento opera), los que a su vez pueden ser nombres de funciones seguidas por sus operandos, por ejemplo:

```
HACER "a 2.81
```

le asigna al nombre "a el valor 2.81.

```
HACER "a :b
```

e asigna al nombre a el valor de la variable b, lo que se simboliza como :b .

Las instrucciones LOGO pueden consistir en más de una llamada a procedimientos:

```
HACER "a :b HACER "b 3 HACER "c 5
```

Cuando un nombre está precedido por los dos puntos (:) nos referimos al valor del mismo y si va precedido por comillas (") nos referimos al mismo nombre.

Los valores alfanuméricos pueden ir con comillas antepuestas, entre comillas, con apóstrofo antepuesto o entre apóstrofos ("a es lo mismo que "a", 'a ó 'a').

Los números o las palabras alfanuméricas que están incluídas en listas no necesitan ir precedidas por comillas.

Por ejemplo:

```
HACER "y 35
```

o bien

```
HACER "x [a b c N]
```

En el primer caso se le asigna a "y el valor numérico 35 que no lleva comillas. Y en el segundo a "x se le asigna la lista dada cuyos elementos no llevan comillas por estar en una lista.

Las instrucciones se dan por medio del Editor provisto por LOGO GRÁFICO, o desde el modo comando, pudiendo dar en una sola línea uno o varios llamados a procedimientos:

```
BT HACER "a 2 HACER "b 4 ESC :a
```

Esta sucesión de instrucciones pueden concluirse o no con un punto y coma, tal como se explica en el capítulo siguiente.

Los argumentos de un procedimiento pueden ser también valores devueltos por funciones:

```
HACER "a SUMA :a 2

HACER "a SUMA :a PROD :b :c
```

La primera de estas instrucciones le asigna a "a el valor :a más 2 y la otra el valor :a más el producto de :b por :c . Estas dos instrucciones podrían haberse escrito también con los operadores infijos '+', '*' y con el operador de asignación infijo := que puede reemplazar a la primitiva HACER (ver primitivas en las pantallas de ayuda o en el archivo PRIMITIV.WRI):

```
HACER "a :a + 2         equivale a:        'a' := :a + 2

HACER "a :a + :b * :c   equivale a:    'a' := :a + :b * :c
```

La palabra vacía puede representarse con ', ' ', " ó " " o emplear la función NIL, que devuelve una palabra vacía, a fin de dar una mayor claridad a los programas.

El orden de las operaciones se realiza de la misma forma como se resuelve una pila, es decir, se recorre de izquierda a derecha y cuando se encuentra un operador prefijo seguido por todos sus operandos o un operador infijo entre sus dos operandos, se ejecuta esta operación y se pone en lugar de ésta el resultado. Luego se vuelve a recorrer la lista de izquierda a derecha y así sucesivamente hasta que se la resuelve completamente.

Por ejemplo:

```
HACER "y RC :a * :b
```

equivale a asignarle a la variable "y" la raíz cuadrada del producto de :a por :b.

```
HACER "y ( RC :a ) * :b
```

equivale a tomar el valor de la raíz de :a, multiplicarlo por :b y asignárselo a la variable "y".

```
HACER "y :a * RC ( :k * :x )
```

equivale a tomar la raíz del producto de :k por :x, multiplicarlo por :a y asignárselo a la variable "y" .

Como se observa, las funciones y procedimientos prefijos tienen siempre la misma jerarquía y se ejecutan después de los infijos.

En una instrucción, si un nombre o palabra no está dentro de una lista y no está precedido por los signos (:) o ("), representa el nombre de un procedimiento o función (prefija o infija). Esta función o procedimiento puede estar incorporado a la implementación del lenguaje o haber sido definida o redefinida por el usuario en la forma que se indicará a continuación.

IV.- PROCEDIMIENTOS Y FUNCIONES

Existen dos clases de procedimientos y funciones: los incorporados, llamados primitivas y los definidos por el usuario.

a. Primitivas:

Son los que ya existen incorporados dentro de LOGO. La lista de ellos puede verse en las pantallas de ayuda o en el archivo PRIMITIV.WRI.

En esta lista las primitivas vienen en letras mayúsculas, pero en esta versión de LOGO pueden ser escritas en letras minúsculas, mezclando mayúsculas y minúsculas o intercalando guiones bajos (_) para una mayor comprensión en la lectura de los programas.

Por ejemplo: `'PONERPRIMERO'` es equivalente a `'ponerprimero'`, `'PonerPrimero'`, `'PONER_PRIMERO'`, `'poner_primero'` ó `'Poner_Primero'`.

Uso de palabras con tilde: Las primitivas que provienen de palabras con tilde pueden usarse sin él y también escritas en mayúsculas. Por ejemplo, `'ULTIMO'` equivale a `'ultimo'` o a `'último'` .

Los procedimientos y funciones definidos por el usuario deben llamarse textualmente tal como fueron definidos. Solamente para las primitivas es válido emplear indistintamente mayúsculas, minúsculas o vocales acentuadas o intercalar guiones bajos.

b. Procedimientos y funciones definidos por el usuario

Los define el usuario a partir del editor y en casos muy especiales empleando las primitivas DEFPROC y DEFFUNC para procedimientos y funciones respectivamente.

c. Edición de procedimientos y funciones

Los procedimientos y funciones son editados en un archivo de texto, por ejemplo:

```
FUNC CUBO :x
  RESPUESTA :x * :x * :x
  FIN

PARA mensaje
  ESCRIBIR [MAL EL VALOR]
  FIN

PROC RECTÁNGULO
  repetir 2 [ad 40 de 90 ad 50 de 90]
  ; Estas instrucciones dibujan un rectángulo
  fin
```

Esto se realiza empleando el editor provisto por LOGO GRÁFICO, en el que se ingresa a partir del modo comando pulsando ALT-D o picando con el mouse en Editor de la barra de menú. Puede también emplearse cualquier otro editor para escribir las funciones y procedimientos. Cada procedimiento o función en el editor comienza con la palabra PARA ó PROC para los procedimientos y FUNC para funciones, el nombre del mismo y la lista de parámetros. PARA, PROC ó FUNC pueden escribirse con minúsculas ('para', 'proc' o 'func') pero no mezclando mayúsculas con minúsculas. Deben terminar con la palabra FIN

Para poner un comentario en el editor: Lo que está después de un (;) dentro de la misma línea se considera como un comentario aclaratorio del programa.

Las funciones deben entregar siempre un valor mediante la primitiva RESPUESTA ó RESP. Los procedimientos retornan a otro que los llama con la instrucción VOLVER, PARAR ó FIN .

Uso del punto y coma para finalizar una línea:

Luego de cada instrucción comienza otra en una nueva línea y se puede abarcar varias líneas concluyendo con el punto y coma (;) la última de ellas, de manera tal que todas juntas constituyen un solo grupo dentro del procedimiento. Es decir que hasta que LOGO no encuentra el ";" no da por finalizada la instrucción. Por ejemplo, el procedimiento:

```
PARA tabla :n ;
  esc  :n
  esc :n * 2
  esc :n * 3 ;
  FIN ;
```

es lo mismo que:

```
PARA tabla :n ;
  esc  :n  esc :n * 2 esc :n * 3 ;
  FIN ;
```

Se indica a LOGO GRÁFICO el uso de los ";" para concluir instrucciones poniéndolo en la primera instrucción del procedimiento o función.

La última instrucción de cada procedimiento o función será FIN, la que puede darse en forma minúscula como 'fin' .

Es posible dejar líneas en blanco para una mayor claridad en el listado, que son ignoradas por LOGO. Cada línea puede contener hasta 120 caracteres.

Uso optativo del punto y coma

Como en casi todas las otras implementaciones de LOGO no se emplea el punto y coma para concluir grupos de instrucciones, sino que éstos concluyen al final de cada línea, a fin de no crearle problemas a los usuarios acostumbrados a esta modalidad, es posible trabajar de esa manera simplemente omitiendo el punto y coma en la primera instrucción (la que comienza con PARA, PROC o FUNC), por lo que LOGO GRÁFICO leerá cada línea de ese procedimiento o función como si tuviera dicho carácter al final de la misma.

Por ejemplo, el procedimiento

```
PARA verificar
  si :color=3 repetir 4 [cp adelante 30 derecha 60] sino fcolor 12
  fin
```

puede escribirse con los ";" en una manera más clara:

```
PARA verificar;
  si :color = 3
     repetir 4 [cp adelante 30 derecha 60]
  sino
     fcolor 12;
  fin;
```

El formato sin los ";" es más cómodo, especialmente para programas con instrucciones cortas y sencillas, pero obliga a que cada instrucción o grupo de instrucciones concluya en una sola línea. En los procedimientos con instrucciones muy complejas puede resultar un programa ilegible.

Por ejemplo, si se emplean los ";" es posible en el editor abrir un corchete en una línea y cerrarlo varias líneas más abajo. LOGO GRÁFICO verifica el balance correcto de corchetes y paréntesis.

En este manual y en los ejemplos escribiremos las instrucciones en ambas formas.

Definición de procedimientos y funciones

En el caso de que se emplee el editor de LOGO GRÁFICO, una vez que se hayan escrito todos los procedimientos y funciones se pulsa nuevamente ALT-D o se pica con el mouse en Definir de la barra de menú y éstos son "definidos", lo que significa que son transformados a la forma de lista dada más adelante, con lo que se los hace ejecutables.

Si se emplea otro editor, los procedimientos se definen a partir de la opción 'CARGAR Y DEFINIR' del menú principal o releyendo el archivo con el editor LOGO GRÁFICO y definiendo los procedimientos desde ese lugar como en el caso anterior.

El proceso de "definición" de procedimientos o funciones procede a la "separación" de operadores y operandos cuando el usuario los escribe juntos en una sola palabra, por ejemplo, la expresión:

```
'a' := 3*(:a+2*:c) ;
```

se transforma en la lista:

```
['a' := 3 * ( :a + 2 * :c )]
```

En el proceso de separación de operadores y operandos no se incluye el operador de asignación

:= , el que debe darse ya separado de su contexto, tal como se indica en el ejemplo precedente.

Un procedimiento o función no puede contener a otro.

Al definir un archivo con procedimientos se definen todos los procedimientos y funciones contenidos en él, como nuevos si no existen o redefiniendo los anteriores si ya existían. Esto vale tanto cuando se definen a partir de un archivo desde el menú principal como desde el editor. Si existen dos procedimientos con el mismo nombre el que está más abajo en el editor anula al anterior.

No existen nombres reservados, lo que significa que es posible redefinir los nombres de los procedimientos y funciones incorporados a LOGO, tanto prefijos como infijos.

Uso de los espacios en blanco:

Todos los espacios en blanco sobrantes al escribir en el editor y en el modo comando son suprimidos automáticamente al interpretar las instrucciones. Por ejemplo, si escribimos:

```
HACER "PUNTOS 10
ESCRIBIR FRASE [PUNTAJE:       ] :PUNTOS
```

responde con:

```
PUNTAJE: 10
```

o sea que se interpreta como:

```
ESCRIBIR FRASE [PUNTAJE:] :PUNTOS
```

Si es nuestro propósito intercalar espacios en blanco y que éstos no sean suprimidos lo logramos tecleando CTRL-B en los lugares donde deseamos que haya un espacio blanco adicional, con lo que en lugar de éstos aparecen en el editor y en el modo comando pequeños puntos. Al interpretar la línea o instrucción estos puntos son reconocidos como espacios en blanco. Si queremos mantener el espacio entre la palabra PUNTAJE: y el número 10 usamos 6 veces CTRL-B:

```
ESCRIBIR FRASE [PUNTAJE:......] :PUNTOS
```

d. Procedimientos y funciones como listas

Al definir un procedimiento LOGO desde el editor se lo convierte en una lista que contiene los siguientes elementos:

- El primer elemento es una lista que contiene los parámetros, los cuales serán identificadores precedidos por dos puntos (:). Si no hay parámetros se da una lista vacía. Se permite un máximo de 10 parámetros.

- Siguen luego varias listas que serán instrucciones como las dadas en el apartado anterior.

- El último elemento es la lista unitaria [FIN] o [fin].

Por ejemplo, en lugar de emplear el editor es posible definir la función que calcula a*sen(k*x) mediante la primitiva DEFFUNC:

```
DEFFUNC "ONDA [[:a :k :x] [RESPUESTA :a * SEN :k * :x] [FIN]]
```

y un procedimiento mediante la primitiva DEFPROC:

```
DEFPROC "verificar
     [[:a] [SI :a >= 0 VOLVER] [ESCRIBIR [MAL VALOR]] [FIN]];
```

```
DEFPROC "salida
     [[:x :y] [escribe [X =] || :x] [escribe [Y =] || :y] [FIN]];
```

La forma de lista puede verse mediante la primitiva TEXTO.

f. Cómo se pasan los argumentos

Los argumentos se pasan por valor, es decir, que al alterar los valores de los parámetros no se alteran los de los argumentos.

Por ejemplo, si definimos el procedimiento:

```
PARA ver :a ;
  hacer "a :a + 2;
  esc frase [VALOR DE A DENTRO DE "VER":] :a ;
  FIN;

PARA prueba;
  hacer "b 2 ;
  ver :b ;
  esc frase [VALOR DE B DESPUÉS DE "VER":] :b ;
  FIN;
```

y llamamos al procedimiento prueba tecleando "prueba" desde el modo comando obtenemos la salida en pantalla:

```
  VALOR DE A DENTRO DE "VER": 4

  VALOR DE B DESPUÉS DE "VER": 2
```

es decir, desde "prueba" se llamó a "ver" pasándole como argumento a :b que tenía el valor 2. Este valor es tomado por el parámetro :a el entrar a "ver" y luego es alterado sumándole dos unidades y por eso al mostrarlo en pantalla desde dentro de este procedimiento da 4. Luego de salir de "ver" se comprueba que el valor de :b no ha sido alterado, es decir, al mostrarlo en pantalla conserva el valor anterior a la llamada a "ver".

V.- FUNCIONES INFIJAS

Una función infija es la que actúa entre medio de dos operandos, como lo hacen por ejemplo la función suma, resta, etc. con los operadores +, - ,etc. (que son llamados operadores infijos). Las funciones infijas se editan encabezándolas con la palabra FUNC seguida por su nombre y dos parámetros. El nombre de la función deberá estar compuesto únicamente por los signos: + - / * | = < > & ^

Por ejemplo podemos redefinir el signo + para usarlo en sumar vectores en el plano:

```
FUNC + :a :b ;
  SI (NUMERO? :A) & (NUMERO? :b) RESPUESTA SUMA :a :b ;

  SI (LISTA? :a) & (NUMERO? :b)
  RESPUESTA LISTA SUMA PRIMERO :a :b ULTIMO :a ;

  SI (LISTA? :b) & (NUMERO? :a)
  RESPUESTA LISTA SUMA PRIMERO :b :a ULTIMO :b ;

  SI (LISTA? :a) & (LISTA? :b)
  RESPUESTA LISTA SUMA PRIMERO :a PRIMERO :b
  SUMA ULTIMO :a ULTIMO :b ;
```

```
  FIN ;
```

Ahora podemos dar desde el modo comando:

```
[3 4] + [5 -1]
```

y LOGO responderá:

```
[8 3]
```

La función '+' discrimina si cada parámetro es una lista o un número y actúa en cada caso. Para sumar dos números emplea en este caso la forma prefija SUMA, pues la forma infija '+' es precisamente la que se está redefiniendo y en caso de emplearla en este lugar estaría llamándose a sí misma recursivamente.

Las funciones LISTA, PRIMERO, ULTIMO, LISTA? y CONTAR son funciones incorporadas para el procesamiento de listas cuya descripción puede verse en las pantallas de ayuda o en el archivo PRIMITIV.WRI .

VI.- JERARQUÍA DE LOS PROCEDIMIENTOS Y FUNCIONES

La jerarquía determina el orden de realización de las operaciones. La mayor jerarquía indica la operación que abarca más y es por consiguiente la que se efectúa en último término. A igual jerarquía entre operadores infijos se efectúa primero el de la izquierda.

Los procedimientos y funciones prefijos tienen todos la máxima jerarquía y los infijos definidos por el usuario la mínima.

Damos a continuación la tabla de jerarquías:

1.- Funciones y procedimientos prefijos incorporados o definidos por el usuario.

2.- Operadores infijos lógicos: | &

3.- Operadores infijos de comparación: < > = <= >= <>

4.- Operadores infijos de suma y resta: + -

5.- Operadores infijos de multiplicación y división: * /

6.- Operadores infijos definidos por el usuario que no redefinen los comprendidos en 2, 3, 4 y 5 y los operadores de concatenación y potenciación: || && ^

Los operadores incorporados al ser redefinidos por el usuario conservan la jerarquía. Por ejemplo, si se redefine la suma y multiplicación para vectores o complejos con los signos '+' y '*' las jerarquías se conservan.

VII.- ÁMBITO DE VALIDEZ DE LAS VARIABLES

El ámbito de validez de los nombres (identificadores) que representan procedimientos y funciones es siempre global, es decir, son reconocidos desde dentro del mismo o de cualquier otro procedimiento o función. Cuando se trata de variables, existen dos casos:

1.- Variables globales:

Se crean mediante una simple asignación desde cualquier procedimiento o función en cualquier nivel. Por ejemplo:

```
HACER "X 2.4
```

Si "X no existe, crea la variable como global y le asigna el valor 2.4. Si ya existe como global, simplemente le reasigna ese valor.

2.- Variables locales:

Se crean mediante el procedimiento LOCAL. Por ejemplo:

```
PARA P ;
  LOCAL "Y1
  HACER "Y1 0
  .....
  FIN
```

Al dar el procedimiento LOCAL toda referencia a la variable "Y1 dentro del procedimiento P ó cualquier otro procedimiento o función llamados por P o desde P (en que "Y1 no esté redefinido como variable local) se referirá a la instancia de "Y1 definida en este lugar. Al salir de P se liberan todas las variables locales. La instrucción LOCAL no inicializa la variable. Para ello se emplea una instrucción ASIGNA , HACER ó := . Por ejemplo, si definimos los procedimientos:

```
PARA PROC1
  LOCAL "a
  HACER "a 2
  escribir fr [VALOR DE "a" EN PROC1 = ] :a
  PROC2
  FIN

PARA PROC2
  escribir fr [VALOR DE "a" EN PROC2 = ]  :a
  FIN

PARA prueba
  HACER "a 1
  escribir fr [ANTES DE LLAMAR A "PROC1" "a" VALE: ]  :a
  PROC1
  escribir fr [DESPUÉS DE LLAMAR A "PROC1" "a" VALE: ]  :a
  FIN
```

Si ahora tecleamos "prueba" desde el modo comando obtenemos la siguiente salida:

```
ANTES DE LLAMAR A "PROC1" "a" VALE: 1

VALOR DE "a" EN PROC1 = 2

VALOR DE "a" EN PROC2 = 2

DESPUÉS DE LLAMAR A "PROC1" "a" VALE: 1
```

Al llamar a prueba definimos una variable global "a", le asignamos el valor 1 y mostramos dicho valor. Luego llamamos a "PROC1" y allí declaramos una variable local "a" a la que le asignamos el valor 2. Esto se comprueba al mostrar en pantalla dicho valor. Si desde "PROC1" llamamos a "PROC2" y mostramos allí el valor de "a" observamos que nuevamente se trata de la variable local definida en PROC1. Ahora volvemos a "prueba" y la variable local "a" ya no existe más, por lo que cuando mostramos el valor de "a" nos referimos a la variable global definida al entrar en "prueba", a la que le hemos asignado el valor 1.

VIII.- INSTRUCCIONES DE CONTROL

Esta versión de LOGO provee instrucciones condicionales e iterativas, aunque no posee bifurcación incondicional (GOTO), la que promueve malos hábitos de programación y no es necesaria en absoluto.

1.- Condicional:

```
SI {cond1} ENTONCES {acción1} SINO {acción2}
```

donde las cláusulas SI y SINO se escriben con letras minúsculas o mayúsculas.

El sentido de las llaves { y } es que dentro de ellas puede haber una o varias instrucciones. No se debe interpretar que estas instrucciones deben escribirse entre llaves.

{cond1}: expresión lógica
{acción1}, {acción2}: una o varias llamadas a procedimientos.
La palabra ENTONCES puede ser omitida.

Si la expresión lógica {cond1} da "VERDAD se ejecuta {acción1}, si da "FALSO se ejecuta {acción2}. La cláusula SINO y {acción2} pueden ser omitidas.

Por ejemplo:

```
SI :a > 1 ENTONCES
  HACER "b :a + 1
SINO
  ESCRIBIR [valor de "a" no válido] ;
```

con el uso de ; o bien:

```
SI :a >1 hacer "b :a + 1 SINO escribir [valor de a no válido]
```

Esta secuencia en PASCAL se escribe:

```
if a > 1 then  b := a + 1
else write ('valor de "a" no valido ');
```

Se pueden encajar condiciones unas dentro de otras:

```
SI {cond1} SI {cond2} {acción21}
     SINO {acción22} SINO {acción12} ;
```

donde cada SINO se aparea con el SI más próximo anterior como en PASCAL.

Se puede emplear en todos los casos, si se desea, la palabra ENTONCES, que actúa como una instrucción nula.

2.- Primitiva REPETIR:

Funciona igual que en todas las versiones de LOGO:

```
REPETIR :n [lista de instrucciones]
```

en donde la lista dada se ejecuta :n veces. Puede usarse como sinónimo REPITE.

3.- Ciclo iterativo MIENTRAS:

Funciona como el WHILE de PASCAL:

```
MIENTRAS [condición] [lista de instrucciones]
```

[condición] es una expresión lógica. La lista de instrucciones se ejecuta indefinidamente mientras la condición expresada en `[condición]` sea verdadera. Puede no ejecutarse ninguna vez.

Por ejemplo, para que la tortuga avance hasta la coordenada y=70 no importando desde donde parte.

```
MIENTRAS [coory < 70] [adelante 1]
```

4.- Ciclo iterativo HASTA:

Funciona como el REPEAT... UNTIL de PASCAL:

```
HASTA [condición] [lista de instrucciones]
```

La [lista de instrucciones] se ejecutará indefinidamente hasta que la condición `[condición]` sea verdadera. Se ejecutará al menos una vez.

Mostramos el mismo ejemplo anterior, ahora con HASTA:

```
HASTA [coory >= 70] [adelante 1]
```

5.- Procedimientos PRUEBA , SICIERTO , SIFALSO :

```
PRUEBA Condición
```

donde `Condición` es una expresión lógica. Esto permite que más adelante puedan emplearse los condicionales:

```
SICIERTO Una o más instrucciones

SIFALSO Una o más instrucciones
```

en cualquier punto del programa, en donde las instrucciones que les siguen se ejecutan de acuerdo al resultado de la última ejecución de PRUEBA.

Por ejemplo en vez de: `SI LEERCAR = "D DE 90 SINO AD 20`

```
PRUEBA LEERCAR ="D
SICIERTO DE 90
SIFALSO AD 20
```

IX.- RECURSIVIDAD

La recursividad es posible hasta cualquier nivel mientras haya memoria disponible. Por ejemplo, mediante el procedimiento:

```
PROC raíces :n
  SI :n > 20 PARAR
  ESCRIBIR RC :n
  raíces :n + 1
  FIN
```

Al llamar al procedimiento con:

```
raíces 20
```

se logra escribir las raíces cuadradas de los primeros veinte números mediante un procedimiento recursivo.

Otro caso sería hallar la suma de todos los elementos de una lista:

```
FUNC sumar_lista :L ;
  SI VACÍA? :L RESP 0;
  RESP (PRIMERO :L) + sumar_lista MENOSPRIMERO :L ;
  FIN ;
```

Es posible además la "recusión de cola" o también llamada "recursión infinita", que ocurre cuando un procedimiento se llama a sí mismo en su última instrucción. Por ejemplo, en el procedimiento:

```
PARA CIRCUN
  AD 1 DE 4
  CIRCUN
  FIN
```

Como en CIRCUN el llamado recursivo se realiza en la última instrucción, el procedimiento no necesita guardar información en las pilas de recursión, por lo que este llamado funciona como una bifurcación incondicional a la primera línea del procedimiento después de haber asignado a los argumentos los valores correspondientes.

Dado que el costo en memoria de este sistema de recursión es cero, la misma puede continuar indefinidamente y no se terminará nunca por agotamiento de la memoria.

Sólo funciona con procedimientos.

No se puede hacer recursión infinita con funciones.

X.- SALIDA A LA PANTALLA DE TEXTO

Para escribir sobre la pantalla de texto o mixta se emplean los procedimientos ESCRIBIR, ESCRIBIRS y LINEA. Los dos primeros tienen un solo argumento. Si es una palabra, se escribe sin las comillas y si es una lista se suprimen los corchetes, y si tiene más de 2 los 2 más externos .

Por ejemplo:

```
ESCRIBIR "hola
hola

ESCRIBIR [Buen día]
Buen día

ESCRIBIR [[PERÚ LIMA] [ECUADOR QUITO]]
```

```
[PERÚ LIMA] [ECUADOR QUITO]
```

El procedimiento ESCRIBIR pasa a la próxima línea después de escribir la lista mientras que ESCRIBIRS se queda en la misma línea (si se usa desde un procedimiento). Ambos comienzan a escribir a continuación de donde se encuentra el cursor. Pueden abreviarse como ESC y ESCS.

Otro ejemplo:

```
ESCRIBIRS [Buen día] ESCRIBIR [, Señores y Señoras]
Buen día, Señores y Señoras
```

El procedimiento MOSTRAR funciona como ESCRIBIR pero no suprime corchetes, y precisamente resulta muy útil para distinguir por ejemplo la palabra "ABCDE de la lista [ABCDE]. Pasa a una nueva línea antes y después de escribir.

Para pasar a una nueva línea se usa el procedimiento LINEA.

XI.- ENTRADA/SALIDA A ARCHIVOS DE TEXTO

Se pueden abrir sólo dos al mismo tiempo, uno para entrada y otro para salida mediante los procedimientos:

```
ABRIR "NOME "E
ABRIR "NOMS "S
```

El segundo argumento contendrá las letras "E ó "S según se abra el archivo de entrada o el de salida. "NOME y "NOMS son los nombres externos de los mismos. Deben ser archivos de texto.

A estos archivos no se les supone ninguna extensión y si el directorio no fue especificado, se asume el directorio corriente.

Un archivo abierto como entrada puede leerse mediante la función LLARCH, que convierte cada línea o registro en una lista. Por ejemplo, si un registro contiene:

```
A X   2.1  **
```

la instrucción:

```
HACER "L LLARCH
```

le asigna a `"L` la lista `[A X 2.1 **]` , eliminando los blancos sobrantes del registro leído.

Si la línea está en blanco, LLARCH retorna una lista vacía y si se encuentra el fin del archivo retorna la palabra vacía.

La función LEERARCH en:

```
HACER "L LEERARCH
```

lee un registro del archivo de entrada y no modifica el contenido, respetando el número de blancos del registro original.

Así si el registro contiene:

```
'   abc  27 '
```

lo convierte en:

```
[   abc  27]
```

En un archivo abierto como salida se graba cada registro mediante la instrucción GRABAR:

```
GRABAR :L
```

en donde si :L es una lista se graba sin el primer y el último corchete y respetando los blancos que contenga.

Los archivos de entrada y salida se cierran mediante las llamadas al procedimiento CERRAR:

```
CERRAR "E
CERRAR "S
```

Archivos de Acceso Directo:

LOGO GRÁFICO permite emplear archivos de acceso directo. Éstos tienen registros a los cuales se los accede por medio de una clave (por ejemplo, nombre y apellido) ordenada en forma alfabética. Para crear uno de estos archivos se emplea la primitiva CrearArchivo, por ejemplo:

```
CrearArchivo "LISTADO 100 24 40
```

crea un archivo de acceso directo con capacidad para 100 registros de hasta 40 caracteres y se accede a ellos con claves de hasta 24 caracteres. En realidad, se crean dos archivos, uno de para las claves y otro para los registros. En este caso, al pedir el directorio se verán en el listado LISTADO.CLV y LISTADO.RGS . Es por eso que en el nombre no se da la extensión.

Luego, para grabarlos y leerlos primero se abre el archivo como acceso directo:

```
ABRIR "LISTADO "D
```

y luego se emplean las primitivas GrabarReg y RegrabarReg para garbarlos:

```
GrabarReg [González,Juan Carlos] [Alumno nuevo,23 años,tercer curso] ;
RegrabarReg [Pérez, Ana María] [21 años, primer curso] ;
```

En el primer caso la clave no debe existir, y se agrega un registro nuevo. En el segundo se altera el contenido de un registro ya existente.

Las claves se ordenan respetando el orden alfabético sin distinguir mayúsculas de minúsculas e intercalando la Ñ entre la N y la O. La comparar entre claves se hace palabra por palabra ignorando los signos de puntuación. Por ejemplo:

```
[González,Juan Carlos]   y   [GONZÁLEZ   JUAN CARLOS]
```

son claves equivalentes. Se distinguen vocales acentuadas de las que no lo son.

Para leer un registro LeerReg ó LLReg (este último suprime los blancos como lo hace LLARCH). Por ejemplo:

```
hacer "Dato LeerReg [González,Juan Carlos]
```

Devuelve una lista de dos elementos (clave y registro), por ejemplo:

```
[[González,Juan Carlos] [Alumno nuevo,23 años, tercer curso]]
```

Mediante la función ULTIMO se puede obtener el registro.

Se puede emplear la primitiva LLENO? para saber si en el archivo caben más registros. Por ejemplo:

```
SI NO LLENO? GrabarReg [Fernández, Pedro]
```

```
        [Jubilado, 73 años, Segundo curso]  ;
```

y la función CLAVE? para indagar si el registro con una dada clave existe:

```
SI CLAVE? [Fernández, Pedro]
     RegrabarReg [Fernández, Pedro]
        [Jubilado, 73 años, Segundo curso]  ;
```

Finalmente, debe cerrarse el archivo mediante:

```
CERRAR "D
```

XII.- PROCESAMIENTO DE LISTAS

Al ser el lenguaje LOGO un descendiente directo del LISP, aunque más sencillo, posee todas las posibilidades de éste en la manipulación de listas.

Para este fin LOGO provee las funciones incorporadas:

Se aplican a listas o palabras:

```
PRIMERO
ULTIMO
MENOSPRIMERO
MENOSULTIMO
```

Se aplican sólo a listas:

```
PONERPRIMERO
PONERULTIMO
```

Por ejemplo:

```
PRIMERO [A B C D E]          da   "A .
MENOSPRIMERO [A B C D]  da   [B C D] .
PONPRIMERO "X [A B C D] da   [X A B C D] .
PRIMERO "ABCD           da   "A .
MENOSPRIMERO "ABCD      da   "BCD .
```

En forma análoga se aplican ULTIMO, MENOSULTIMO y PONERULTIMO.

Existen las funciones NUMERO?, PALABRA?, LISTA?, VACIA? que retornan "VERDAD cuando el argumento posee un valor numérico, es una palabra, es una lista o es una palabra o lista vacía respectivamente y "FALSO en caso contrario.

La función LISTA convierte a sus argumentos en elementos de una lista:

```
LISTA "A [C D]          da   [A [C D]] .
```

La función FRASE une listas o palabras en una sola lista:

```
FRASE "A [C D]          da   [A C D].
FRASE [A X] [C D]       da   [A X C D].
FRASE "AB "KJ           da   [AB KJ].
```

La función PALABRA une dos palabras:

```
PALABRA "AB "KJ             da   "ABKJ .
```

En lugar de FRASE o PALABRA pueden emplearse también las funciones infijas || ó && respectivamente, especialmente útiles cuando se aplican a más de dos argumentos.

La función MIEMBRO? da "VERDAD si el primer argumento pertenece a la lista dada en el segundo argumento:

```
MIEMBRO? 2 [1 2 3]      da   "VERDAD .
```

Si se desea obtener un elemento de una lista o un carácter de una palabra se emplea la función ELEMENTO :

```
ELEMENTO 3 [A B C D E F]    da   "C .
ELEMENTO 3 "ABCDEF      da   "C .
```

Puede emplearse ITEM en lugar de ELEMENTO.

NUMMIEMBRO devuelve el número de orden que ocupa un elemento o carácter en una lista o palabra o cero si no está dentro de ella:

```
NUMMIEMBRO "C [A B C D E F] da   3 .
NUMMIEMBRO "C "ABCDEF       da   3 .
```

También es posible reemplazar, suprimir o insertar elementos en una lista o palabra, invertirlas o tomar partes de éstas:

```
REEMPLAZAR 2 "X [A B C D]   da   [A X C D]
REEMPLAZAR 2 "X "ABCD       da   AXCD
INSERTAR 2 "X [A B C D] da   [A X B C D]
INSERTAR 2 "X ABCD      da   AXBCD
SUPRIMIR 2 [A B C D]        da   [A C D]
SUPRIMIR 2 "ABCD        da   ACD
INVERTIR [a [1 2] b c d]    da   [d c b [1 2] a]
INVERTIR "abcd          da   dcba
PARTE [a b c d e f g] 2 3   da   [b c d]
PARTE "abcdefg 2 3      da   bcd
```

Se provee además la función NIL, que retorna una palabra vacía, a fin de dar mayor claridad a los programas.

Las funciones `PRIMITIVA? :pal` y `DEFINIDA? :pal` nos informan si una palabra es una primitiva o el nombre de un procedimiento o función ya definido en la memoria.

XIII.- FUNCIONES LÓGICAS Y DE COMPARACIÓN

Las constantes lógicas son "VERDAD y "FALSO . Si en algún lugar donde se espera un valor lógico aparece otra cosa LOGO señala error. Pueden emplearse también en forma minúscula como "verdad y "falso, pero no mezclando mayúsculas con minúsculas.

Para comparar se emplean las funciones infijas:

```
=  <  >  <>  >=  <=
```

o las funciones prefijas:

```
IGUAL?    MENOR?    MAYOR?
```

que equivalen respectivamente a `'=', '<' y '>'` .

Al hacerse las comparaciones se verifica antes si los argumentos son numéricos. Si lo son se comparan como números y en caso contrario se comparan alfabéticamente.

Por ejemplo:

```
1.0 = 1.0E+00 da   "VERDAD .
1   < 1.0     da   "FALSO .
"AB > "AA     da   "VERDAD .
```

Los operadores lógicos son los infijos:

```
&    |
```

que pueden ser remplazados por los prefijos:

```
Y    O
```

y el operador de negación es el NO .

XIV.- OPERACIONES NUMÉRICAS

Las constantes numéricas se dan en los formatos usuales, con o sin punto decimal, en notación de punto fijo o flotante empleando la letra E para el exponente, por ejemplo:

```
2    2.1  3.4E-04   -0.016
```

No deben contener blancos. El rango del exponente varía entre 1.0E-37 y 1.0E+37.

Las operaciones numéricas se realizan con los operadores infijos:

```
+    -    *    /
```

o los prefijos:

```
SUMA  DIF  PROD  DIVISION
```

LOGO provee además funciones usuales tales como:

```
EXP   LOG   SEN   COS   ARCTAN   RESTO   RC   ENTERO   FRAC
```

La potenciación se realiza con el signo '^', por ejemplo:

```
3 ^ 4          da   81
```

Puede emplearse en la potenciación un exponente no entero, pero en este caso la base debe ser siempre positiva. Si el exponente es entero puede darse una base negativa:

```
(-2) ^ 5       da   -32

(-2) ^ 4       da   16
```

Se provee la constante PI = 3.141592....

XV.- LISTAS EJECUTABLES. OTRAS ESTRUCTURAS LÓGICAS

En LOGO es posible dar listas que pueden ser secciones de programa a ejecutar. Estas listas son argumentos de algunas primitivas tal como REPITE, MIENTRAS, HASTA y las que manejan eventos en la animación simultánea. Con la primitiva CUMPLIR se hace que una lista se ejecute, por ejemplo:

```
...
HACER "L [HACER "A 1 HACER "B 2]
CUMPLIR :L
....
```

equivale a escribir directamente:

```
....
HACER "A 1 HACER "B 2
....
```

Al emplear listas ejecutables debe tenerse en cuenta dos reglas importantes:

Dentro de una lista ejecutable no puede retornarse del procedimiento o función mediante primitivas tales como RESPUESTA, VOLVER o PARAR.

Por ejemplo, la instrucción:

```
....
MIENTRAS [:n > 0] [procesar SI :k > 0 VOLVER]
....
```

(donde procesar es un procedimiento ya definido)

no es aceptada, pues la lista contiene la primitiva VOLVER. Para lograr el efecto deseado debe escribirse:

```
....
HACER "retornar "FALSO ;
MIENTRAS [(:n > 0 ) & (NO :retornar)]
[procesar SI :k > 0 HACER "retornar "VERDAD] ;

SI :retornar VOLVER ;
....
```

Excepto las condiciones de MIENTRAS o HASTA, una lista ejecutable no puede devolver un valor.

Por ejemplo, si la lista :L contiene una expresión que es dato de un problema que ha sido pedida al usuario, como en el caso de que :L tenga asignado como valor la lista [:x+2*:y], y deseamos asignarle el valor de esta expresión a una variable "z mediante:

```
HACER "z CUMPLIR :L
```

Al ejecutar CUMPLIR :L obtendríamos un valor que no puede ser procesado y el programa daría un mensaje de error. Si queremos asignarle a "z el valor de la expresión contenida en :L debemos escribir:

```
CUMPLIR FRASE [HACER "z] :L
```

o bien:

```
CUMPLIR [HACER "z] || :L
```

de tal manera que la lista afectada por CUMPLIR no devuelve ningún valor.

Otras estructuras lógicas:

Hay otras estructuras lógicas de otros lenguajes que se pueden construir con los mismos recursos de LOGO y la primitiva CUMPLIR.

Por ejemplo, si se desea construir el ciclo iterativo de PASCAL:

```
for i:=1 to n do begin
  a := a + i;
  j := j + 2*i
  end;
```

en LOGO puede hacerse:

```
CICLO "i 1 :n [HACER "a :a + :i HACER "j :j + 2 * :i] ;
```

si se inserta el procedimiento:

```
PARA CICLO :var :inf :sup :acción ;

HACER :var :inf ;
MIENTRAS [(VALOR :var) <= :sup]
     [CUMPLIR :acción  HACER :var (VALOR :var) + 1] ;
FIN ;
```

El procedimiento CUMPLIR ejecuta el contenido de una lista.

La función VALOR da el valor aplicado de su argumento. Se emplea en este caso 'VALOR :var' para no escribir ::var. Por ejemplo, si damos:

```
HACER "i 23
HACER "A23 [A B C]
```

entonces:

```
VALOR PALABRA "A :i          nos da la lista    [A B C] .
```

XVI.- POSIBILIDADES GRÁFICAS

Desde el modo gráfico están disponibles todos los procedimientos y funciones para el dibujo de gráficos en dos dimensiones, los cuales podrán ser guardados en un archivo mediante la opción Archivos - Guardar Gráfico del menú principal y ser impresos en alta calidad con los recursos de impresión de WINDOWS (tanto en colores como en blanco y negro y en cualquier impresora) con la opción Archivos - Imprimir de dicho menú. Las figuras grabadas pueden ser leídas mediante la opción Archivos - Cargar Gráfico del menú principal desde cualquier otro controlador gráfico, aún cuando éste sea distinto al de la instalación donde la figura fue grabada. Existe también la primitiva CARGARGRAF. Se puede leer dibujos realizados en la versión DOS de LOGO GRÁFICO.

1. Geometría de la tortuga

Se encuentran los procedimientos ADELANTE, ATRÁS, IZQUIERDA, DERECHA, FRUMBO, CENTRO, BORRARPANTALLA, SINPLUMA, CONPLUMA, etc. y sus apócopes AD, AT, IZ, DE, BP, SP, CP, etc. Pueden verse todos en la descripción de las primitivas.

Se dan procedimientos para ubicar la tortuga en cualquier punto y en cualquier dirección, girarla alrededor de un punto, dibujar elipses, arcos de elipse, círculos, arcos de círculo y sectores elípticos y

circulares.

Se puede dibujar en 21 grosores diferentes y en línea llena, cortada o de puntos. Para ello se emplean los procedimientos FGROSOR y FTRAZO.

2. Textos sobre la pantalla gráfica

Existen dos recursos diferentes para escribir sobre la pantalla gráfica:

a) Con caracteres tomados de un font de WINDOWS en cualquier tamaño, inclinación o grosor. Simplemente, la tortuga o el actor corriente indican la dirección en que se escribe y su posición, desde dónde se escribe o alrededor de qué punto se centra el texto.

La secuencia:

```
...
FTIPO "ARIAL
FGROSOR 2 FANCHOTIPO 1.6 ;(se leen FIJAR GROSOR y FIJAR ANCHO TIPO)
CENTRO IZQUIERDA 45 ;
ROTULAR [PRUEBA DE TEXTO SOBRE PANTALLA]
```

escribe sobre la pantalla gráfica el texto indicado a partir de la posición de la tortuga y en la dirección de 45 grados en el font de WINDOWS denominado "ARIAL".

El procedimiento ROTULAR escribe las letras a partir del lugar donde se encuentra la tortuga y en la dirección que ésta indica y finalmente ubica a la misma al final del texto. ROTULARC escribe el texto centrado en la posición de la tortuga sin cambiar su posición.

El font puede cambiarse mediante la primitiva FTIPO o entrando al cuadro de diálogo de WINDOWS para elegir el font a través de la opción Pantalla - Tipo de letra.

Los textos escritos de esta manera aparecen en los gráficos de alta calidad obtenidos mediante la opción Archivo - Imprimir... del menú principal, pero ROTULAR y ROTULARC no pueden ser empleadas cuando se está en el modo de animación simultánea.

b) Otra posibilidad de escribir textos sobre la pantalla gráfica la da la primitiva CARTEL. Por ejemplo, la secuencia:

```
...
CARTEL [PRUEBA DE CARTEL SOBRE PANTALLA] [40 40] 18
...
```

escribe el contenido de la lista centrado en el punto de coordenadas [40 40] sobre un fondo de color rojo. Solamente se puede escribir en forma horizontal. El tamaño de los caracteres se fija con FANCHOTIPO (ver primitivas) o eligiendo directamente el font de WINDOWS en el menú principal.

La primitiva CARTEL es la única que se puede emplear en animación simultánea para escribir textos en la pantalla.

3. Sombreados y rayados

Para dibujar en LOGO GRÁFICO figuras sombreadas o rellenadas se emplean los procedimientos FRELLENO, RELLENAR y PINTAR . En especial, para círculos y elipses puede usarse PINTARCIRCULO y PINTARELIPSE.

a) Mediante la primitiva RELLENAR

Se da primeramente el procedimiento FRELLENO :s :c (se lee FIJAR RELLENO) donde :s es un

tipo de sombreado (de 0 a 20) y :c el color del sombreado, se dibuja una figura cerrada y luego se da RELLENAR . Por ejemplo, la secuencia:

```
FRELLENO 8 1
REPETIR 6 [ADELANTE 60 IZQUIERDA 60]
RELLENAR
```

dibuja a partir de la posición de la tortuga un hexágono de lado 60 con un sombreado cuadriculado horizontal. Ver el tema de los sombreados en la descripción de la primitiva FRELLENO.

La secuencia de dar la primitiva FRELLENO, dibujar una figura cerrada y luego la primitiva RELLENAR debe hacerse en forma consecutiva. Si se la interrumpe por medio de algún comando de menú (por ejemplo, entrar en el editor y luego volver al modo comando), al dar FRELLENO se recibirá un mensaje de error. En este caso se debe reiniciar el proceso o emplear la primitiva PINTAR.

Los rellenos hechos de esta manera aparecen en los gráficos de alta calidad obtenidos mediante la opción Archivo - Imprimir... del menú principal.

b) Mediante la primitiva PINTAR

Para hacer sombreados se puede emplear también la primitiva PINTAR, que simplemente inunda con el sombreado pedido toda una zona a la que pertenece el punto dado por las coordenadas de la tortuga y está delimitado por la frontera más próxima de cualquier color diferente del de dicho punto. Por ejemplo, se puede rellenar un cuadrado por medio de la secuencia:

```
FRELLENO 8 1
REPETIR 4 [ADELANTE 60 IZQUIERDA 90]
SINPLUMA IZ 45 ADELANTE 30
PINTAR
```

La primitiva PINTAR sirve también para rellenar dibujos con los actores.

Lo rellenado mediante esta primitiva no sale en la impresora si se imprime el dibujo como gráfico. En este caso se lo debe imprimir como Bitmap.

c) Mediante las primitivas PINTARCIRCULO y PINTARELIPSE

En combinación con FRELLENO puede pintarse círculos y elipses mediante estas dos instrucciones. Por ejemplo, para pintar de rojo un círculo de radio 60:

```
FRELLENO 20 18
PINTARCIRCULO 60
```

y para llenar una elipse con semiejes horizontal y vertical 80 y 40 respectivamente con un relleno cuadriculado oblícuo de color verde oscuro:

```
FRELLENO 8 2
PINTARELIPSE 80 40
```

Para más detalle sobre estas primitivas consultar las ventanas de ayuda.

4. Dimensiones de la pantalla

Se supone que la tortuga dibuja sobre una superficie igual a la de la pantalla del monitor, cuya relación entre el alto y el ancho es de 0.75. Sin embargo, en WINDOWS se muestra solamente una parte de dicha pantalla tanto en modo mixto como en la pantalla entera. En el primer caso se recorta aproximadamente un cuarto del la pantalla para dar lugar a la pantalla de texto y en el segundo también se recorta la parte correspondiente al título, la barra de menúes y eventualmente la barra de herramientas.

Si se desea visualizar la pantalla completa se debe elegir la opción del menú principal Edición - Visualizar (o pulsar F8) y podemos apreciar la pantalla gráfica en toda su extensión.

En los modos con y sin pantalla de texto (modo mixto y modo gráfico) se recorta siempre la parte inferior de la ventana gráfica completa.

Por omisión se supone que el rango de la coordenada X dentro de la pantalla va entre -160 y +160 y la coordenada Y entre -120 y +120 para la ventana gráfica completa, que es la que se ve cuando ésta ocupa la totalidad de la pantalla del monitor al pulsar F8 en el modo comando. Si se desea alterar estos rangos se pueden emplear los procedimientos FRANGOX, FRANGOY o FRANGOS (se leen FIJAR RANGO/S). En los dos primeros pueden definirse escalas diferentes para ambas direcciones.

El eje X va de izquierda a derecha y el eje Y de abajo hacia arriba.

Se pueden emplear dos sistemas angulares diferentes para dar la orientación de la tortuga:

a) El rumbo, que es el ángulo de la tortuga con la dirección hacia el Norte (dirección positiva del eje Y) de tal manera que es positivo en el mismo sentido de las agujas del reloj. Estos ángulos se miden en grados sexagesimales de 0 a 360 grados.

Se emplea la primitiva FRUMBO :RUM (apócope de FIJAR RUMBO) orienta la tortuga hacia el rumbo :RUM y la función RUMBO nos da el rumbo hacia el cual está orientada.

b) El ángulo de la tortuga con la dirección positiva del eje X de tal manera que es positivo en el sentido contrario a las agujas del reloj. Los ángulos se miden en grados sexagesimales de 0 a 360 grados. Se utilizan las primitivas FANGULO :ang (se lee FIJAR ÁNGULO) para orientar a la tortuga hacia el ángulo :ang y la función ÁNGULO nos da el ángulo hacia el cual está orientada.

Si se trabaja en modo LIBRE (por omisión) la tortuga puede salir fuera de la pantalla, y sus límites serán un ancho de pantalla hacia la izquierda, derecha, arriba y abajo. En el caso de las escalas iniciales, estos valores varían desde -480 hasta +480 para ambas coordenadas. Sin embargo, la pantalla será siempre una ventana que mostrará los valores entre -160 hasta +160 para X y entre -120 hasta +120 para Y.

Mediante las primitivas FRANGOX, FRANGOY o FRANGOS puede alterarse las escalas de X e Y.

Si se da la primitiva JAULA, no se permite que la tortuga salga fuera de la pantalla, en cuyo caso se da un mensaje de error.

Existe también la posibilidad de trabajar en el modo VUELTA, que consiste en que cuando se emplean las primitivas ADELANTE o ATRÁS , si la tortuga o el actor corriente salen por uno de los bordes de la pantalla, reaparecen por el modo opuesto para seguir dibujando desde allí o para continuar su movimiento si no dibujan. Se entra en este modo mediante la primitiva VUELTA y se sale con LIBRE o JAULA.

El modo VUELTA solamente se recomienda para niños pequeños, pues tiene la particularidad de que la tortuga o el actor corriente nunca pueden salir de la pantalla. Se desaconseja su uso normal, pues no permite el funcionamiento de algunas primitivas tales como RELLENAR. Tampoco tiene ningún efecto sobre el modo de animación simultánea.

XVII.- ANIMACIÓN

En la presente versión de LOGO es posible programar animación con figuras o formas en el controlador VGA ó Super VGA (16 ó 256 colores).

Estas figuras se diseñan en grillas (cuadrículas) de forma rectangular y pueden tener dimensiones elegidas por el usuario desde 3 por 3 puntos hasta 125 por 125 puntos en la dirección horizontal y

vertical respectivamente

Desde las primeras versiones de LOGO se ha adoptado - para explicar el hecho de mover porciones de pantalla la fantasía de simular actores que se disfrazan con formas diseñadas por los usuarios.

En esta versión existen 120 actores que pueden llevar (disfrazarse) hasta 120 figuras o formas distintas. La primitiva FFORMA (se lee Fijar FORMA) permite asignar a cada actor activo una forma dada y éstas se hacen visibles con la primitiva VISIBLE.

La ubicación de las formas en la pantalla se da por medio de las coordenadas de sus puntos centrales con las primitivas FX (Fijar X), FY (Fijar Y), FXY (Fijar X e Y) o FPOS (Fijar Posición).

Para poder generar estas formas LOGO GRÁFICO provee un editor de formas que está descripto en las pantallas de ayuda.

Debe tenerse en cuenta lo siguiente al programar animación:

a) La animación es realizada por "actores" que se mueven en la pantalla. En esta versión de LOGO los actores no son "tortugas disfrazadas" y la tortuga no es un actor.

Desde el momento en que se activa el modo animación por medio de la primitiva DECIR :n o ACTIVAR :n la tortuga desaparece de la pantalla y no recibe más comandos, los que son dirigidos de ahora en más hacia los actores.

Estos pueden dibujar (aunque su función principal es moverse y actuar) en diferentes colores y en general todas las primitivas dirigidas a la tortuga pueden dirigirse a ellos (ADELANTE, ATRÁS, DERECHA, IZQUIERDA, FRUMBO, etc.) , pero lo que dibujan no puede salir como gráfico en la impresora por medio de la opción del menú Archivos - Imprimir ni ser grabadas por medio de GUARDARGRAF.

La tortuga tiene como misión principal graficar y guarda los dibujos registrando las coordenadas de los puntos iniciales y finales de cada segmento, sus colores y los sombreados que delimitan y su forma de funcionamiento no puede ser imitado por los actores, pues el movimiento de éstos se haría muy lento.

Para volver a emplear la tortuga gráfica debe ordenarse BORRARPANTALLA (BP) o RG. Existe la primitiva LP (Limpiar Pantalla) que borra los dibujos sin afectar las posiciones de la tortuga o actores.

Si se desea guardar los dibujos hechos por la tortuga juntamente con los realizados por los actores y las formas estampadas debe emplearse la primitiva GUARDARDEC o la opción Archivo - Guardar Bitmap del menú principal.

Gráficos y decorados

En general, denominaremos "gráfico" al dibujo hecho por la tortuga y "decorado" o "Bitmap" al realizado por otros medios sumado a lo graficado por la tortuga.

b) En animación no se deben cambiar los rangos:

Los rangos entre los que se maneja la pantalla en animación son los empleados en la pantalla gráfica por omisión (X entre -160 y +160, Y entre -120 y +120 para la pantalla gráfica completa) y no pueden estar cambiados mediante FRANGOX, FRANGOY o FRANGOS. Si este fuera el caso, se los debe restaurar previamente a su modo inicial con RG (restaurar gráfico).

Los actores sólo se pueden mover dentro de los límites de la pantalla gráfica, los que pueden ser ampliados con la primitiva LIBRE que los extiende para X desde -480 a +480 e Y entre -480 y +480. Pero en este caso, la pantalla es una ventana por donde se ve nada más que los valores de X desde -160 hasta +160 y los de Y según la pantalla sea mixta o entera y si una porción de una figura sale fuera de estos límites es recortada.

c) Cómo se hace animación:

Para comenzar a realizar una animación debe seguirse aproximadamente los siguientes pasos:

1. Cargar las formas que se van a emplear por medio de la primitiva CARGARFORMAS o elaborarlas con el editor de formas.

Existe también la posibilidad de dibujar algo con la tortuga y convertirlo a FORMA con la primitiva CONVERTIR o elaborar formas copiándolas desde el portapapeles.

2. Si se va a emplear un decorado, éste se carga mediante la primitiva CARGARDEC. Es posible cargar decorados aún con actores activos, aunque no en modo de animación simultánea.

3. Hacer todos los dibujos que sea necesario para armar o complementar el fondo con la tortuga gráfica.

4. Mediante la primitiva DECIR :n o ACTIVAR :n se entra en el modo animación.

De este modo se activa un actor y los rangos de variación de las coordenadas de los actores se ponen en modo JAULA automáticamente, es decir, no pueden salir de la pantalla. Si se desea permitir que puedan salir de la pantalla usar la primitiva LIBRE.

Luego le fijamos a cada actor la forma que va a llevar con FFORMA y la hacemos visible mediante VISIBLE (se permite usar MT y OT aunque no sean tortugas). Ninguna primitiva que se refiera a un actor especifica a cuál de ellos está dirigida la instrucción, pues se referirá siempre al "actor activo", que será el que ha sido mencionado en la última vez que se empleó la primitiva DECIR :n o ACTIVAR :n. Se puede saber cuál es el actor activo mediante la primitiva QUIEN.

La animación se puede realizar por dos técnicas diferentes: animación secuencial y animación simultánea, las que pasamos a describir.

ANIMACIÓN SECUENCIAL

Es la única manera en que se realiza la animación en las primeras versiones de LOGO. Consiste en enviarle a los actores comandos mediante las mismas primitivas que se emplean para la tortuga gráfica (ADELANTE, ATRÁS, IZQUIERDA, DERECHA, FRUMBO, etc.) complementadas con otras tales como FFORMA, VISIBLE, NOVISIBLE, ESPERAR, etc. Por ejemplo, para mostrar una figura humana caminando se puede hacer:

```
PARA HOMBRE
  ACTIVAR 1 FPOS [-100 0] FFORMA 1 VISIBLE
  FIN

PARA PASO
  AD 3 FFORMA 1 ESPERAR 10
  AD 3 FFORMA 2 ESPERAR 10
  AD 3 FFORMA 3 ESPERAR 10
  AD 3 FFORMA 4 ESPERAR 10
  FIN

PARA CAMINAR
  CARGARFORMAS "caminar.frm
  BP
  HOMBRE
  FRUMBO 90
  REPITE 10 [paso]
  FIN
```

Las formas 1 a 4 describen a la figura humana en diferentes posiciones durante la acción de caminar.

ANIMACIÓN SIMULTÁNEA

Emplea una novedosa e interesante técnica similar a la utilizada en los videojuegos, en la que no se le indica en cada momento a cada actor qué es lo que se debe hacer sino que se lo deja evolucionar independientemente hasta que ocurra algún "evento" que es controlado por el programador por medio de primitivas especiales que son dadas antes de que esa situación se haya producido.

Una circunstancia a prever puede consistir en el choque de un actor con un borde de la pantalla, con otro actor, con un bloque definido mediante la primitiva DEFBLOQUE, por haber ingresado a alguna zona de la pantalla pintada de un determinado color, porque se ha pulsado alguna tecla o porque se ha pulsado un botón del Mouse.

La manera de proceder es la siguiente:

- Se definen bloques rectangulares con DEFBLOQUE para controlar choques de formas con los actores.

- Luego se hacen procedimientos de "presentación" para que cada actor aparezca en un lugar con determinada forma como el procedimiento HOMBRE listado más arriba.

- Se fija a cada actor su rumbo y velocidad con FRUMBO y FVEL (o FVELX, y FVELY) - opcionalmente pueden fijárseles aceleraciones con FACELX y FACELY .

- Se emplean las instrucciones que manejan los eventos, las que comienzan con "CUANDO...", tales como CUANDOBLOQUE, CUANDOCHOCA, CUANDOTECLA, etc., para decidir cuál será el comportamiento de cada actor cuando ocurra cada una de las situaciones previstas.

- Se pone en movimiento el conjunto con la primitiva ACTUAR o MOVERSE :n. Los actores comienzan a moverse de acuerdo a las condiciones iniciales fijadas y siguen en movimiento a no ser que les ocurra algún evento. En este caso, para el actor al que este evento le ocurrió se ejecutará la lista que acompaña la definición previa de la acción a efectuar en la primitiva correspondiente del tipo CUANDO... mientras que los demás actores prosiguen independientemente.

- Cuando se desarrolla la animación simultánea existe un conjunto de primitivas que no pueden emplearse, tal como las que modifican el estado de la pantalla (MODOTEXTO, MODOMIXTO, BORRARPANTALLA, BORRARTEXTO, etc.).

No es posible escribir textos ni emplear las primitivas ROTULAR o ROTULARC para escribir algo con la tortuga gráfica, pues ésta está inhabilitada. Para comunicarse con el usuario se debe usar la primitiva CARTEL o estampar mensajes gráficamente con ESTAMPARFORMA o ESTAMPAR. Tampoco es posible emplear las primitivas ADELANTE y ATRÁS aunque sí IZQUIERDA y DERECHA. Puede emplearse las primitivas TOCAR y TOCARSIEMPRE para ejecutar en forma asincrónica sonidos en formato ".WAV" o música con archivos ".MID".

La pantalla de texto está habilitada en modo mixto durante la animación simultánea y se puede escribir y ejecutar comandos desde allí.

- Se sale del modo de animación simultánea de las siguientes maneras:

a) Por algún error durante la ejecución.
b) Pulsando ESC o CTRL-Break.
c) Mediante la primitiva INTERRUMPIR, que se usa dentro de las listas ejecutables que manejan los eventos. Indica que al concluir la ejecución de la lista se interrumpe la animación simultánea.
d) Mediante el botón correspondiente de las barras de herramientas.

Por ejemplo, si ya hemos cargado formas con CARGARFORMAS, escribimos en modo comando:

```
BORRARPANTALLA
ACTIVAR 1
FFORMA 1
FPOS [0 40]
VISIBLE
FRUMBO 45
FVEL 10
ACTUAR
```

el actor toma la forma 1 y comienza a moverse en una dirección diagonal ascendente hacia la derecha a 45 grados con la horizontal. Al llegar a algún borde de la pantalla, si no se indicó otra cosa con las primitivas CUANDOIZQ, CUANDODER, CUANDOABAJO y CUANDOARRIBA rebota en forma elástica sin pérdida de energía (es decir que no disminuirá su velocidad en cada choque).

Podemos detener el movimiento de la figura pulsando ESC.

Podemos mover dos actores del mismo modo con:

```
BP
ACTIVAR 1 FPOS [-100 0] FFORMA 1 VISIBLE
FRUMBO 90 FVEL 10

ACTIVAR 2 FPOS [100 0] FFORMA 2 VISIBLE
FRUMBO 270 FVEL 15
CUANDOCHOCA 1 [REBOTAR]
ACTUAR
```

En este caso se mueven las dos figuras y cuando chocan entre ellas rebotan de acuerdo a las leyes del choque elástico de dos cuerpos con la misma masa, pues por medio de la primitiva CUANDOCHOCA le hemos dicho al actor 2 que cuando choque con el actor 1 rebote en forma elástica.

Lo dado anteriormente lo podemos completar ahora con un bloque:

```
PARA PRUEBA
  BP
  FGROSOR 3 FRELLENO 10 1 REPITE 4 [AD 50 DE 90] RELLENAR
  DEFBLOQUE 1 [0 50] 50 50
  ACTIVAR 1 FFORMA 1 FPOS [100 100] VISIBLE
  CUANDOBLOQUE 1 [REBOTAR]
  FVEL 10 FRUMBO 45
  ACTIVAR 2 FFORMA 2 FPOS [0 100] VISIBLE
  CUANDOBLOQUE 1 [REBOTAR]
  FVEL 15 FRUMBO 135
  CUANDOCHOCA 1 [REBOTAR]
  ACTUAR
  FIN
```

las figuras, además de rebotar entre sí y con los bordes de la pantalla rebotan con los bordes de un cuadrado de lado 50 .

EVENTOS:

La programación por eventos es una técnica novedosa en los LOGOS actuales, que permite desarrollar estrategias de anticipación en los niños. De hecho proviene de los modelos de programación orientados a objeto, que es un tema de mucho interés en programadores e investigadores.

Se pueden manejar 13 eventos diferentes, que se enumeran a continuación, donde :L es una lista ejecutable que se ejecuta en el momento en que el evento se produce.

Los eventos que se refieren a actores individuales son dados para el actor activo en el momento de dar la primitiva CUANDO..., de modo que todos los comandos dentro de :L se dirigen a él.

`CUANDOCHOCA :n :L`
para cuando el actor activo choca con el actor :n .

`CUANDOBLOQUE :nb :L`
para cuando el actor activo choca con el bloque :nb, definido previamente con DEFBLOQUE.

`CUANDOIZQ :L - CUANDODER :L - CUANDOARRIBA :L - CUANDOABAJO :L`
para cuando el actor activo choca con algún borde de la pantalla visible o la definida por la primitiva FLIMITES para cada actor. Si se dio LIBRE, para los actores a los que no se les haya fijado cotas mediante FLIMITES, la pantalla visible pasa a ser una ventana central a una pantalla más grande y los actores no pueden pasar de los límites -480 y +480 en ambas direcciones. Si esto último ocurre da mensaje de error y se detiene la ejecución, y estas cuatro primitivas no tienen efecto alguno. En este caso los límites deben manejarse con CUANDOX y CUANDOY.

El modo VUELTA no tiene efecto sobre la animación simultánea.

`CUANDOACTOR :t :L`
Determina que cada :t centésimos de segundo se ejecute la lista :L para el actor activo.

`CUANDOPASEN :t :L`
Determina cada :t centésimos de segundo se ejecute la lista :L .

`CUANDOX :x1 :x2 :L - CUANDOY :y1 :y2 :L`
Se da esta primitiva, en el caso de CUANDOX, cuando el actor activo se encuentra al dar este comando dentro de los límites :x1 y :x2. Se activa al salir éste fuera de los mismos. Lo mismo ocurre con CUANDOY en la dirección vertical.

`CUANDOTECLA :L - CUANDOBOTON :L`
La primera se activa cuando se pulsa una tecla, la que se puede determinar por medio de la función LEERCAR. La segunda cuando se pulsa algún botón del Mouse, lo que se puede luego discriminar con las funciones BOTONDER? y BOTONIZQ? .

`CUANDOPISACOLOR :C :L`
:C puede ser un número de color o una lista de números de color. Se activa cuando el punto central de la figura toca el color :C o alguno de la lista de colores de :C. El color pisado puede hallarse con la primitiva COLORDEBAJO.

- Para desactivar un evento

Existen dos métodos de los cuales el segundo es más claro.

a) Dando en lugar de la lista ejecutable una lista vacía. Por ejemplo:

```
...
CUANDOTECLA [HACER "c LC SI :c = "f CUANDOTECLA [] SINO procesar]
...
```

Mediante esta instrucción se activa el evento CUANDOTECLA de tal manera que si pulsamos un 'f' se desactiva y si pulsamos cualquier otra tecla se llama al procedimiento "procesar", pero en este caso sigue activado.

b) Mediante la primitiva DESACTIVAR, por ejemplo, en el caso anterior:

```
...
CUANDOTECLA [HACER "c LC SI :c = "f DESACTIVAR SINO procesar]
...
```

La primitiva DESACTIVAR hace que al terminar de ejecutarse la lista el evento CUANDOTECLA se desactive. Es importante tener esto en cuenta, pues si escribimos, en otro ejemplo:

```
CUANDOTECLA [DESACTIVAR FFORMA 5] ;
```

esto tiene el mismo efecto que:

```
CUANDOTECLA [FFORMA 5 DESACTIVAR] ;
```

pues DESACTIVAR sólo desactiva el evento recién después de ejecutada la lista, por lo que la instrucción "FFORMA 5" se ejecutará en ambos casos.

- Primitiva REBOTAR

Se emplea para que se produzca un rebote elástico entre dos actores o entre un actor y un obstáculo después de ser ejecutada la lista ejecutable de los eventos CUANDOCHOQUE, CUANDOX, CUANDOY y CUANDOBLOQUE. Por ejemplo, si escribimos:

```
CUANDOX (-100) 100 [REBOTAR] ;
```

el actor va a rebotar elásticamente contra las líneas verticales (-100) y (+100). Si escribimos:

```
CUANDOX (-100) 100 [FVELX 0.98 * VELX REBOTAR]
```

el actor, antes de rebotar, va a perder un dos por ciento de su velocidad y el rebote será inelástico.

- Primitiva INTERRUMPIR

Del mismo modo que DESACTIVAR y REBOTAR, actúa solamente cuando la lista de un evento ya ha sido ejecutada. sirve para interrumpir la animación simultánea. Por ejemplo, el evento:

```
CUANDOX (-100) 300
[CARTEL [ACTOR FUERA DE LIMITES] [0 0] 12 INTERRUMPIR]
```

hacer que se muestre en el centro de la pantalla un mensaje sobre un fondo rojo e interrumpe la animación simultánea.

- Primitiva FLIMITES

Se emplea para acotar el movimiento de un actor dentro de una zona rectangular de tal modo que su comportamiento dentro de la misma es el que tendría en la totalidad de la pantalla visible si no se dio LIBRE. Es decir, rebota sin pérdida de energía sobre los bordes de dicha zona y las primitivas que manejan eventos CUANDOIZQ, CUANDODER, CUANDOABAJO, CUANDOARRIBA pasan a tener vigencia sobre los bordes izquierdo, derecho, inferior y superior de la misma.

Por ejemplo: si hacemos:

```
DECIR 1 FFORMA 4 VISIBLE FVEL 18 FRUMBO 30
FLIMITES [-50 50] 100 100
ACTUAR
```

el actor rebota sobre los lados de un cuadrado de lado 100.

- Primitiva FAREAVISIBLE

Fija un rectángulo de tal manera que el actor será visible dentro del mismo, fuera de él no se verá y si se encuentra sobre uno de sus lados será recortado. Aunque no sea visible, seguirá interactuando con los otros actores, los bordes de la pantalla o los fijados por FLIMITES.

Por ejemplo, en el caso:

```
DECIR 1 FFORMA 4 VISIBLE FVEL 18 FRUMBO 30
FAREAVISIBLE [-50 50] 100 100
ACTUAR
```

el actor rebotará contra los bordes de la pantalla pero podrá ser visto solamente dentro de un rectángulo de lado 100 definido por FAREAVISIBLE.

OTRAS FACILIDADES PARA ANIMACIÓN:

- Formas que rotan

Normalmente, al cambiarle el rumbo a una forma mediante FFORMA la figura que el actor lleva no cambia. Sin embargo, es posible lograr que un actor tome formas dependientes de su dirección dando como argumento de FFORMA una lista de dos números:

```
DECIR 1 FFORMA [41 12] VISIBLE
```

El primer número de la lista es la forma para el rumbo cero y el otro es el número total de formas para la vuelta completa de tal manera que, para este caso, el actor 1 adopta la forma 41 para el rumbo 0, la 42 para el rumbo 30, la 43 para el rumbo 60 y así siguiendo hasta la 52 para el rumbo 330. Los cambios de forma se producirán cuando se ejecute ADELANTE, ATRAS o en los rebotes con bloques, bordes u otros actores.

Las formas para cada dirección se dibujan en el Editor de Formas, primero para la dirección cero y luego se la rota mediante la opción del menú Corrimientos y Giros - Rotar Bloque para las demás direcciones.

- Movimiento Gravitatorio

Se pueden simular movimientos gravitatorios alrededor de un planeta mediante la primitiva:

```
GRAVITATORIO [xc yc] :KG
```

donde [xc yc] es el centro de giro (donde debería estar el cuerpo que atrae) y :KG es una constante que simula la atracción gravitacional de una masa a otra y es del orden de 300 a 400. Cuanto mayor es el número que se asigna a :KG será mayor la atracción y por lo tanto los móviles que pasen cerca del punto de coordenadas [xc, yc] desviarán su trayectoria proporcionalmente al valor de :KG, o inclusive entrarán en órbita alrededor de ese punto. Se sale del modo gravitatorio mediante las primitivas FACELX y FACELY. Ver programa PLANETAS.LGO en los discos de instalación.

- Movimiento Oscilatorio

Si se desea efectuar movimientos oscilatorios:

```
OSCILATORIO [xc yc] :KX :KY
```

donde [xc yc] fija los valores de ordenada y abscisa respecto a los cuales se oscila y :KX y :KY las constantes elásticas en ambas direcciones que oscilan de 0 a 3.

Ver programa "OSCILAR.LGO" en los discos de instalación. Se sale del modo oscilatorio mediante las primitivas FACELX y FACELY.

- Cambio de formas con animación simultánea:

Para cambiar las formas de un actor alternativamente de una manera regular puede emplearse las primitivas:

```
CAMBIARFORMA :inic :final :t
ROTARFORMA :inic :final :t
```

En ambos casos se comienza con la forma :inic, después con la forma :inic+1 y así sucesivamente hasta llegar a la forma :final. Cada forma queda visible un tiempo dado por :t medido en centésimos de segundo.

En el caso de ROTARFORMA se vuelve nuevamente a la forma :inic, por lo que resulta ideal para simular movimientos tales como vuelos de pájaros, hombres caminando, helicópteros volando, molinos, cuerpos rotando, etc. Puede verse el programa VUELOS.LGO en los discos de instalación.

Con CAMBIARFORMA el actor queda con la forma :final.

- Primitivas FESTADO :num_estado y ESTADO

Mediante la primera se fija un número que determina en qué estado o etapa de su evolución, está un actor durante una animación. Esto sirve para requerir este número mediante ESTADO y facilitar la toma de decisiones dentro de las listas ejecutables :L de las primitivas CUANDO... .

Por ejemplo a un monstruo se le asigna el estado 1 (con FESTADO 1) cuando está calmo y estado 2 cuando está agresivo. O bien fijarle estado 1 a una nave si está vulnerable y estado 2 si tiene un campo de fuerza que lo torna invulnerable. En este caso puede haber distintos grados de vulnerabilidad (0, 1 , 2, 3...) y si lo alcanza un proyectil perder altura o puntos, de acuerdo a su estado, etc.

XVIII.- MOUSE

El Mouse, además de ser empleado por LOGO en el manejo de sus comandos interactivos, puede ser utilizado dentro de los programas, aunque con ciertas limitaciones.

Por ejemplo, si se da el procedimiento:

```
PARA mover_con_mouse
  MIENTRAS [NO BOTON?] []
  SI BOTONIZQ? FPOS POSMOUSE SINO PARAR
  mover_con_mouse
  FIN
```

es posible mediante el Mouse obligar a la tortuga a moverse a los puntos donde éste se encuentra. La línea MIENTRAS... ejecuta una lista vacía mientras no se pulse algún botón del Mouse, lo que será aprovechado para ubicar a éste en el punto deseado de la pantalla. En este caso, si se pulsa el botón izquierdo la tortuga es llevada hasta la posición del Mouse dibujando una línea recta desde el lugar donde estaba y si se pulsa el botón derecho se detiene la ejecución del programa.

Las demás primitivas que controlan al Mouse están explicadas en la lista de las primitivas y no necesitan mayor aclaración.

B-XIX CARGA DE PROGRAMAS

En los casos en que un conjunto de procedimientos y funciones ya están elaborados y verificados y solamente se desea ejecutarlos sin pasar por el editor existe la opción Archivos - Cargar y definir del menú principal que equivale a la primitiva CARGAR. En ambos casos se carga y se definen todos los procedimientos y funciones contenidos dentro de un archivo de texto para poder luego ejecutarlos llamándolos desde el modo comando. Estos se agregan de esta manera a los ya definidos anteriormente

desde el editor o por medio de otras llamadas con el procedimiento CARGAR.

Existe el caso en que se desea cargar un programa y comenzar a ejecutarlo. Para ello se emplea la primitiva CARGAREXEC o la opción de menú Archivo - Cargar y Ejecutar que, luego de reinicializar todo, carga todos los procedimientos y funciones contenidos en un archivo e inmediatamente llama a un procedimiento denominado "inicio" (en minúsculas) que estará contenido dentro del archivo a definir.

Por ejemplo, si escribimos:

```
CARGAREXEC "BLOQUES
```

luego de reinicializar todo, carga el archivo BLOQUES.LGO (está contenido dentro de los programas de demostración provisto por LOGO GRÁFICO) e inmediatamente llama al procedimiento inicio, el cual realiza el llamado a los procedimientos que ejecutan el programa.

XX.- PRIMITIVAS ESPECIALES PARA WINDOWS

La implementación en WINDOWS de LOGO GRÁFICO permite aprovechar con grandes ventajas varias de las facilidades que esta plataforma nos da para el manejo de la pantalla a través de menúes y cuadros de diálogo.

Por ejemplo, en WINDOWS puede imprimirse con cualquier impresora en su mejor calidad posible los gráficos dibujados por la tortuga, pues se emplea aquí las interfaces que ya están instaladas con todas las impresoras. Se puede emplear también cualquier tipo (font) de letra para dibujar letras en la pantalla gráfica con los que están implementados en WINDOWS en la impresión de gráficos.

a] Menúes:

Existen un conjunto de primitivas especiales para armar un menú.

Ejemplo:

```
para nuevo_menu
  CrearMenu
    AgregarAlMenú NIL [redondas]
      ColgarDelMenú "circ "circunferencia
      ColgarDelMenú "elip "elipse
    AgregarAlMenú NIL [otras figuras]
      ColgarDelMenú [repite 4 [ad 40 iz 90]] "cuadrado
      ColgarDelMenú "pent "pentágono
      ColgarDelMenú "bp "borrar
    AgregarAlMenú "volviendo "salir
  PonerMenú
  PonerTítulo [FIGURAS GEOMÉTRICAS]
  fin
```

De esta manera se crea el menú, con sus submenúes respectivos:

Principal redondas otras figuras salir

Submenúes circunferencia cuadrado
elipse pentágono
borrar

Al dar PonerMenú el menú creado reemplaza al de LOGO GRÁFICO en el modo comando. Luego, al elegir las opciones de los submenúes se llaman los procedimientos de nombre "circ, "elip, "pent y "volviendo, que deben estar ya programados, que obviamente dibujan un círculo, una elipse y un pentágono y el último de ellos sale del programa. En el caso del cuadrado, se lo dibuja directamente

dando una lista ejecutable.

AgregarAlMenú crea los nuevos ítems principales del menú. Si el primer argumento es una palabra vacía (NIL) se espera que vengan ítems de un submenú que colgarán de éste mediante ColgarDelMenú. Si este argumento es un nombre de un procedimiento o una lista ejecutable, no existe el submenú y se ejecuta directamente dicho procedimiento al elegir el ítem principal.

En ColgarDelMenú se procede del mismo modo, excepto que el primer argumento deberá ser siempre el nombre de un procedimiento o una lista ejecutable.

Entonces, al ejecutarse este programa, simplemente se ejecuta el procedimiento nuevo_menu y luego se vuelve al modo comando de tal modo que ya puesto el menú de nuestro programa, se elige de entre las opciones de éste.

La primitiva QuitarMenú vuelve el menú del modo comando al menú principal de LOGO GRÁFICO y vuelve a poner el título original. Es por eso que el procedimiento "volviendo toma la forma:

```
para volviendo
  quitarMenú
  fin
```

b] Mensajes y consultas al usuario:

Se da la posibilidad de pedir al usuario, por medio de pequeños cuadros de diálogo, decisiones acerca de lo que debe hacer el programa o simplemente darle alguna información.

La función ACEPTA? :texto presenta un cuadro de diálogo con un texto y los botones "SI" y "NO". Según por cual se salga devolverá el valor "VERDAD o "FALSO, lo que permite controlar en forma interactiva el flujo del programa. Por ejemplo:

```
SI ACEPTA? [¿Va a grabar todo?] grabar_todo SINO no_grabar_nada
```

De modo similar funciona la función CONSULTAR :titulo :texto, que contiene tres botones: "SI", "NO" y "cancelar" y según por cuál de ellos se salga devolverá "S, "N o "C en cada caso.

Mediante la función ENTRARDATOS :titulo :subtítulo se le presenta al usuario un cuadro de diálogo que contiene, además de un título sobre la barra superior, un subtítulo interno con un campo de edición donde se espera que se entren datos desde el teclado. Se sale por medio de los botones "aceptar" y "cancelar". En el primer caso la función devuelve los datos entrados en forma de lista y en el segundo caso devuelve una palabra vacía. Por ejemplo:

```
HACER "coeficientes LEERDATOS [Nuevo Caso] [Valores de A, B y C:]
```

en donde la variable "coeficientes recibirá una lista con los valores de A, B y C para realizar algún cálculo matemático.

El procedimiento INFORMAR :titulo :mensaje simplemente da una información al usuario y espera a que se pulse el botón "aceptar". Por ejemplo:

```
INFORMAR [Mensaje:] [Proceso terminado]
```

c] Cuadros de diálogo de LOGO GRÁFICO

Los cuadros de diálogo ya implementados en LOGO GRÁFICO pueden emplearse por medio de algunas primitivas implementadas a tal efecto. Tenemos, por ejemplo:

BUSCARARCHIVO :P permite presentar el cuadro de diálogo para cargar archivos.
GUARDARCOMO :P sirve para presentar el cuadro de diálogo para guardar archivos.

Los procesos interactivos para imprimir un gráfico y visualizar un gráfico en toda la pantalla del monitor realizar mediante IMPRIMIRGRAF y VISUALIZAR. Con PEDIRTIPO se ejecuta el diálogo para elegir el Font, su tamaño, color y estilo.

d] Cuadros de diálogo

Para armar un cuadro de diálogo de mayor complejidad en LOGO GRAFICO se emplean cinco primitivas que permiten construirlo paso a paso:

CrearCuadro :tit

Crea un cuadro de diálogo vacío con el título dado por la lista o palabra :tit.

AgregarTexto :tex

Agrega al cuadro de diálogo el texto estático :tex

AgregarBoton :var :tit

Agrega al cuadro de diálogo un botón lógico que tiene delante un título :tit cuyo estado inicial será el de marcado o no según el valor de la variable :var ("FALSO o "VERDAD) y su estado final quedará contenido en la variable :var si se sale del cuadro con del botón "Aceptar".

AgregarCampo :var :tit :long

Agrega al cuadro de diálogo un campo de edición de longitud :long caracteres que tiene delante un título :tit cuyo texto inicial será el dado por el valor de la variable :var y su contenido final será asignado a la variable :var si se sale del cuadro con del botón "Aceptar".

AceptaCuadro?

Es una función que expone el cuadro de diálogo.

LOGO GRÁFICO va armando el cuadro de diálogo poniendo sus elementos donde pueda, ordenándolos a medida que son definidos. Esto exime al usuario de tener que definir otros parámetros, tales como las coordenadas de los botones, altura de los campos, ubicación y medidas de la ventana, etc.

Si se ha salido con el botón "Aceptar" la primitiva AceptaCuadro? devuelve "VERDAD y asigna a las variables definidas en los argumentos :var de AgregarBoton y AgregarCampo los valores correspondiente. Si se ha salido con el botón "Cancelar" devuelve "FALSO y las variables citadas continúan con los valores que tenían.

En el cuadro es posible también poner botones de opción y texto estático. Recomendamos ver el ejemplo en el programa CUADRO.LGO que viene con los discos de instalación y consultar la descripción de las primitivas.

Por ejemplo:

```
FUNC EntrarLosDatos
  hacer "a 1 hacer "bot "VERDAD

  CrearCuadro [Caso de Prueba]
  AgregarTexto [Esto es un texto estático]
  AgregarBoton "bot [Esto es un botón lógico]
  AgregarCampo "a [Entrar valor de a] 30

  resp AceptaCuadro?
  FIN
```

Se expone el cuadro de diálogo y se espera que el usuario escriba sobre los campos expuestos o cambie el estado de los botones lógicos, para luego salir con "Aceptar" ó "Cancelar". En el primer caso, las variables "a y "bot toman los valores dados en su campo de edición o botón lógico dentro del cuadro de diálogo. En el segundo caso quedan con los valores que tenían antes de la ejecución del mismo.

Al entrar en este cuadro de diálogo, en el campo de edición correspondiente a "a y el botón lógico correspondiente a "bot se ponen los valores que éstas variables tenían antes de la ejecución de esta primitiva.

e] Botones y Controles:

1.- Botones WINDOWS:

Mediante la primitiva FBOTON :titulo :proc :AX :AY se puede crear un botón de diseño WINDOWS que se ubica en el lugar donde se encuentra la tortuga con dimensiones :AX por :AY. Lleva escrito encima el contenido de la lista o palabra :titulo y en el caso de que se lo oprima llama al procedimiento :proc. Este argumento puede también ser una lista ejecutable.

Los botones del diseño de WINDOWS sólo responden desde el modo comando, por lo que deben ser creados juntamente con el menú y los controles en un procedimiento inicial (generalmente "inicio") y volver luego a dicho modo para disponer de ellos como opción del programa.

Por ejemplo, si damos la secuencia:

```
CENTRO OCULTARTORTUGA
FBOTON  "MT "MT  40 20
AD 40
FBOTON "OT "OT 40 20
AD 30
```

Se crean dos botones, tal que al oprimir uno de ellos la tortuga se muestra y al oprimir el otro se oculta.

2.- Controles o Barras de Deslizamiento WINDOWS

Se los crea mediante la primitiva FCONTROL :Nom :proc :long :N1 :N2. De este modo se ubica una barra de deslizamiento centrada en la posición de la tortuga de longitud :long.

:Nom es una lista que contiene dos palabras: la primera es el nombre con el que se identifica al control y la segunda da la dirección de la barra, "V si es vertical y H si es horizontal. :N1 y :N2 son los valores mínimo y máximo que nos devolverá la función POSCONTROL :N de acuerdo a la posición del indicador de la barra y :proc es el nombre de un procedimiento o una lista ejecutable que serán llamados cada vez que dicho indicador cambie de lugar. En el caso de que no se llame a ningún procedimiento se da una lista o palabra vacía.

Podemos crear el control:

```
CENTRO
FCONTROL [Velocidad H] [] 100 (-50) 50
FPOSCONTROL "Velocidad 25
```

con lo que queda dibujado una barra de deslizamiento horizontal que la identificaremos con el nombre "Velocidad . Cuando movemos el indicador de esta barra no se llama a ningún procedimiento.

Si alteramos la posición del indicador con el Mouse y pedimos el valor correspondiente a su posición mediante POSCONTROL "Velocidad se nos dará un número que corresponderá a dicha posición de acuerdo a los límites definidos.

Los controles, botones del diseño de WINDOWS sólo responden desde el modo comando, por lo que deben ser creados juntamente con el menú en un procedimiento inicial (generalmente "inicio") y volver luego a dicho modo para disponer de ellos como opción del programa. Ver programa CONTROL.LGO en los discos de instalación.

3.- Botones diseñados por el usuario:

En este caso se dan dos formas, una para el botón suelto :NF1 y otra para el botón oprimido :NF2. Se lo puede crear mediante FBOTONLOGO :proc :NF1 :NF2 , de tal modo que al ser oprimido llama al procedimiento :proc. Este argumento puede también ser una lista ejecutable.

Este tipo de botones actúa no sólo en el modo comando, sino también durante la ejecución de un programa, en animación tanto secuencial como simultánea.

Por ejemplo, si damos:

```
PARA probar
  RG FCF 0
  FBOTONLOGO "cambio 34 35
  DECIR 1 FFORMA 1 VISIBLE FVEL 18 FRUMBO 30
  ACTUAR
  FIN
```

siendo el procedimiento cambio:

```
PARA cambio
  DECIR 1
  SI FORMA = 1 FFORMA 2 SINO FFORMA 1
  FIN
```

al correr el programa probar una forma se desplazará por la pantalla y cambiará cada vez que se pulse el botón creado.

f] Para programadores experimentados

Los programadores experimentados pueden emplear dos recursos que le permitirán hacer más amigables sus programas.

Uno de ellos es la primitiva PEDIRAYUDA :archivo :clave, que permite pedir una ayuda por medio de la clave :clave en el archivo :archivo con terminación '.HLP'. Este archivo se genera por medio de los editores de pantallas de ayuda que existen para WINDOWS. Por ejemplo, si se desea pedir ayuda para el ítem "Cómo de juega", simplemente se llama:

```
PEDIRAYUDA "JUEGO.HLP "como_se_juega
```

Se provee también la primitiva PUEDECERRAR?, que es llamada automáticamente por LOGO GRÁFICO al salir del intérprete. Normalmente, consulta al usuario si desea volver a WINDOWS y si éste responde con el botón SI sale del programa.

Se puede redefinir PUEDECERRAR? (únicamente en mayúsculas) desde un programa de tal manera que haga otras consultas al usuario y realice otras tareas, con la condición de que debe devolver solamente los valores "VERDAD y "FALSO.

Es especialmente útil en la versión RUNTIME de LOGO GRÁFICO.

Por ejemplo:

```
FUNC PUEDECERRAR?
  HACER "salida CONSULTAR [Va a Salir] [¿Graba lo anterior?]
```

```
  SI :salida = "S grabar_lo_anterior RESPUESTA "VERDAD
  SI :salida = "N RESPUESTA "VERDAD
  RESPUESTA "FALSO
  FIN
```

Como se observa, se consulta sobre si se va a grabar alguna información anterior antes de salir. Si se sale con el botón "cancelar" se responde con "FALSO para que no se vuelva a WINDOWS, si se responde con SI se graba y sale del programa y con NO se sale del programa directamente.

XXI.- PUERTOS Y MANDOS

Existen en esta versión de LOGO GRÁFICO un conjunto de primitivas que se emplean para programar los puertos de comunicaciones y los mandos (Joysticks).

a] Puertos:

Se puede operar tanto en los puertos serie como en los paralelos.

Para los puertos serie se dan primitivas para abrir, leer un carácter, enviar un carácter, cerrar y averiguar si hay caracteres pendientes en un puerto serie (ABRIRCANAL, LEERCANAL, ESCCANAL, CERRARCANAL, CARPENDIENTE? respectivamente).

Para los puertos paralelo se dan LEERPUERTO y ESCPUERTO.

Todas estas primitivas pueden verse descriptas en las pantallas de ayuda o en el archivo PRIMITIV.WRI .

b] Mandos:

Se dan primitivas para manejar los mandos o joysticks. Se suponen que pueden existir hasta dos mandos. Mediante la primitiva EXISTEMANDO? :N se puede averiguar si existe el mando :N y con AJUSTARMANDO :N se puede ajustar el cero.

Los mandos se manejan principalmente mediante las funciones INCLMANDO :N, que nos da un número que es proporcional a cuánto ha sido inclinado el mando :N y RUMBOMANDO :N que da hacia dónde apunta el mando :N.

BOTONMANDO? :N :nb nos dice si ha sido pulsado el botón :nb del mando :N.

Por ejemplo:

```
...
SI BOTONMANDO? 1 1
  DECIR 1 FFORMA 23 VISIBLE FPOS [0 0] FVEL (INCLMANDO 1) / 20
  FRUMBO RUMBOMANDO 1 ACTUAR ;
...
```

Esta secuencia, a partir de los valores que nos da la inclinación del mando y hacia dónde apunta pone un móvil en el centro de la pantalla, le da una velocidad inicial y un rumbo y lo dispara.

XXII.- SONIDO Y VIDEO

LOGO GRÁFICO tiene la posibilidad de emplear dispositivos de sonido para ejecutar el contenido de archivos en formato ".WAV" y ".MID" si la instalación cuenta con alguno de ellos. También se puede pasar videos del tipo ".AVI". En todos los casos se emplea la primitiva TOCAR, pero para música y

videos puede usarse también las primitivas TOCARMUSICA y PASARVIDEO.

Para escuchar un archivo del tipo ".WAV" se escribe:

```
TOCAR  "DADA.WAV
```

Para escuchar música de un archivo del tipo ".MID" se escribe:

```
TOCAR  "JAZZ.MID
```

Para escuchar un archivo del tipo ".AVI" se escribe:

```
TOCAR  "ACCION.AVI
```

y al ejecutar esta orden LOGO carga el archivo con el nombre dado y hace sonar su contenido por los parlantes en los dos primeros casos. En el tercer caso, abre una ventana, pasa el video y cuando éste se termina lo cierra. Para los archivos MIDI existe también la primitiva TOCARMUSICA.

Para el caso de los archivo ".WAV" es posible que la ejecución se repita indefinidamente con:

```
TOCARSIEMPRE "GALOPE.WAV
```

por ejemplo, para reproducir por el parlante el galope de un caballo. El sonido se interrumpe en cualquier caso con la primitiva PARARSONIDO o comenzando a hacer sonar los parlantes con otra instrucción TOCAR o TOCARSIEMPRE.

Los sonidos se ejecutan en forma "asincrónica", es decir, una vez cargado el archivo el programa sigue ejecutándose mientras suenan los parlantes. Pero como durante la carga del mismo se interrumpe la ejecución del programa, esto puede resultar un serio inconveniente para la animación secuencial o simultánea.

En el caso de los ".WAV", este problema es subsanado si cargamos previamente el archivo en algún lugar de la memoria mediante:

```
CARGARSONIDO "GALOPE.WAV  1
CARGARSONIDO "FANFARRIA.WAV  2
```

lo que carga los archivos mencionados en los registros de sonido 1 y 2. Puede haber hasta 8 archivos de sonido cargados en memoria.

Luego, los registros pueden emplearse en las primitivas TOCAR y TOCARSIEMPRE dando como argumento el número de registro de sonido en lugar del nombre del archivo en la forma:

```
TOCAR  1
...
TOCARSIEMPRE 2
...
PARASONIDO
etc. etc.
```

La instrucciones para hacer sonar en los parlantes los archivos ".WAV" ó ".MID" pueden emplearse tanto en animación secuencial como simultánea y son un interesante recurso para agregarle efectos sonoros a los programas.

Procesamiento de Videos:

Se provee la primitiva PasarVideo que permite exponer un video en cualquier lugar de la pantalla gráfica.

Por ejemplo:

```
PasarVideo "PASEO [POS [-100 80]
  TITULO [Paseo por el Parque] AMPLIAR 1.8] ;
```

expone un video del archivo PASEO.AVI con el ángulo superior izquierdo de la ventana en el punto de coordenadas [-100 80] ampliado por un factor de 1.8 .

Los comandos se dan en una lista a fin de que en futuras versiones de LOGO GRÁFICO se puedan agregar nuevas posibilidades, tales como manejar el volumen de sonido, canales de sonido, etc. .

La opción "CON PALETA" es útil cuando la ventana de video se ubica sobre un fondo armado con un decorado BMP ó PCX. Como el Video cambia la paleta, los colores de los decorados se deforman totalmente, a no ser que éstos se hayan adaptado previamente a la paleta del Video.

Para resolver este problema primeramente se ejecuta desde el modo directo el Video con la opción "CON PALETA" , con lo que la paleta del Video reemplazará a la actual. Por ejemplo:

```
PasarVideo "PASEO [POS [-100 80] CON PALETA]
```

Luego se entra en el menú a la opción Colores - Editar Paleta y con el botón Grabar se graba la paleta en un archivo de extensión ".PAL". por ejemplo, PALVIDEO.PAL. Al ejecutar el programa se carga esta paleta por medio de CARGARPAL , luego el decorado con CARGARDEC :

```
CARGARPAL "PALVIDEO.PAL
CARGARDEC "FONDO.BMP
PASARVIDEO "PASEO [POS [-100 80] AMPLIAR 2.2]
etc. etc. ...
```

y ahora los colores del decorado no serán cambiados.