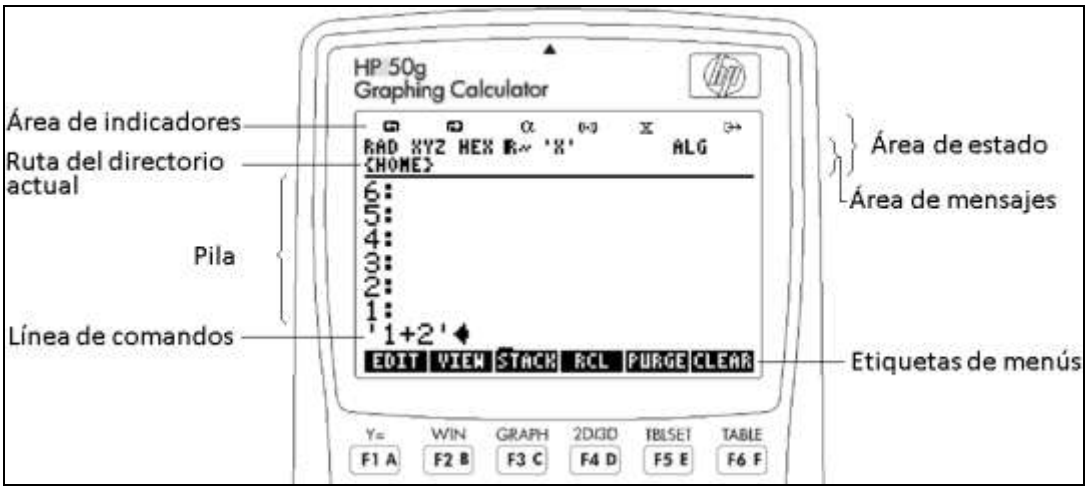
CONTENIDO

1 PANTALLA

En la mayoría de las situaciones la pantalla aparecerá dividida en cuatro secciones, como en el siguiente gráfico.



A esta configuración se le llama **pantalla de pila**.

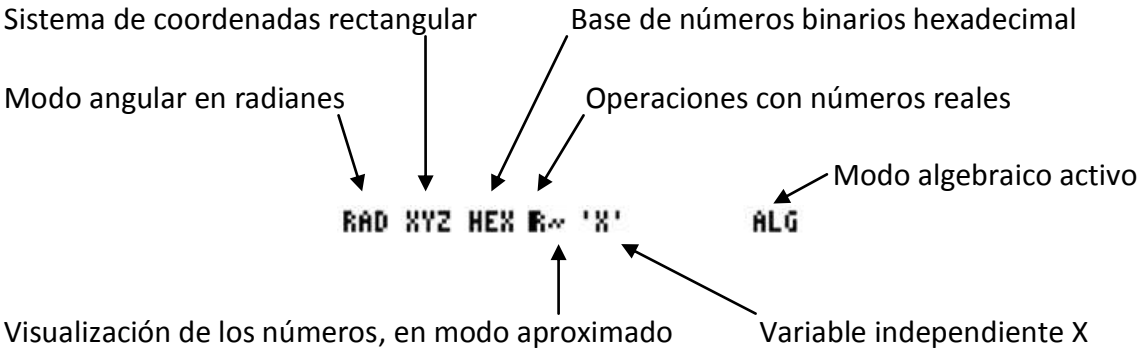
AREA DE ESTADO

El área de estado se encuentra en la parte superior de la pantalla. Se divide en las siguientes secciones:

AREA DE INDICADORES: Muestra el estado de la calculadora.

RUTA DEL DIRECTORIO ACTUAL: Muestra la ruta del directorio actual.

AREA DE MENSAJES: Proporciona informaciones para ayudar al usuario como: la medida angular, sistema de coordenadas, base de los números binarios, operaciones con números reales o complejos, visualización de los números en modo aproximado o exacto, la variable independiente actual y si se está ingresando un objeto en modo algebraico.



PILA

Es una serie de ubicaciones de almacenamiento en la memoria, para los objetos (números, cadenas, listas, etc.), estas ubicaciones se llaman niveles: nivel 1, nivel 2, etc.

LINEA DE COMANDOS

Es el área por donde se ingresa los objetos (números, operadores, comandos, funciones, etc.). Cuando no se ingresa o edite un objeto, la línea de comandos no aparece.

ETIQUETAS DE MENUS

Muestra los comandos, directorios, objetos, etc. correspondientes a las seis teclas superiores del teclado. Las teclas superiores son las teclas asociadas al menú.

COMANDOS

❶ **HEADER→** : Obtiene el tamaño del área de mensajes en líneas. Una línea es la altura necesaria para visualizar un texto.

SINTAXIS:

HEADER→ ⇨ n

Ejemplos:

El área de estado tiene dos líneas de texto: en la primera línea está el texto "RAD XYZ HEX R= 'X'" y en la segunda línea "{ HOME }".

❷ **→HEADER** : Establece el tamaño del área de mensajes en líneas, requiere el número de líneas, el número de líneas no debe ser mayor que dos.

SINTAXIS:

n →HEADER ⇨

Ejemplos:

→HEADER

→HEADER

2 TECLADO

Es el periférico utilizado para ingresar objetos, manipular objetos y otras acciones sobre los objetos en la pila de la calculadora.

ORGANIZACION DEL TECLADO

Las teclas de la calculadora tienen seis niveles o estratos. Cada tecla contiene un conjunto diferente de funciones.

TECLADO PRIMARIO: Representa al teclado principal. Son los caracteres que aparecen escritos sobre las teclas, son de color blanco o negro en la calculadora HP 49G+ y HP 50G.

Ejemplos: 

TECLADO DE CAMBIO IZQUIERDO: Se activa presionando la tecla color verde en la calculadora HP 49G+ y de color blanco en la HP 50G () .Estos caracteres están escritos de color verde (49G+) y blanco (50G) en la parte superior izquierda de las teclas primarias correspondientes.

Ejemplo: para activar la función **ABS**, se presiona la tecla  y luego la tecla primaria correspondiente  .

TECLADO DE CAMBIO DERECHO: Se activa presionando la tecla color rojo en la calculadora HP 49G+ y de color anaranjado en la HP 50G () .Estos caracteres están escritos de color rojo y anaranjado en las calculadoras HP 49G+ y HP 50G respectivamente, en la parte superior derecha de las teclas primarias correspondientes.

Ejemplo: para activar la función **LOG**, se presiona la tecla  y luego la tecla primaria correspondiente  .

TECLADO ALFABETICO: Se activa presionando la tecla  .Estos caracteres están escritos de color amarillo en la parte derecha o inferior sobre las teclas.

Ejemplo: para escribir la letra **R**, se presiona la tecla  y luego la tecla  .

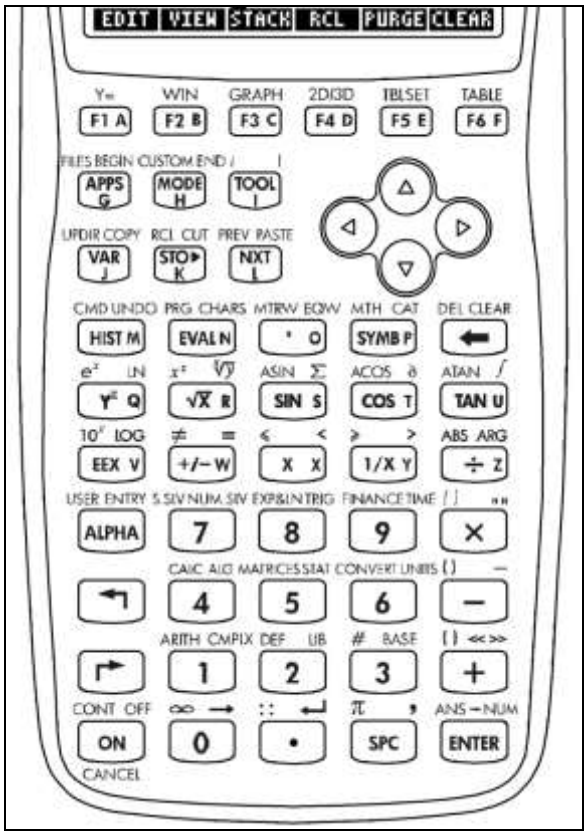
TECLADO ALFABETICO Y CAMBIO IZQUIERDO: Se activa presionando la tecla  y luego la tecla  .Estos

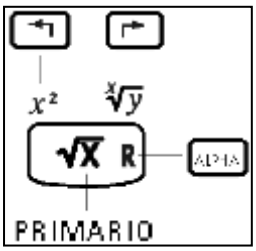
caracteres no están escritos en la calculadora, incluyen a las minúsculas y caracteres especiales.

Ejemplo: para escribir la letra **r**, se presiona la tecla [ALPHA], luego la tecla [↰] y por último la tecla [√x R].

TECLADO ALFABETICO Y CAMBIO DERECHO: Se activa presionando la tecla [ALPHA] y luego la tecla [↱].Estos caracteres no están escritos en la calculadora, estos caracteres incluyen a las letras griegas y caracteres especiales.

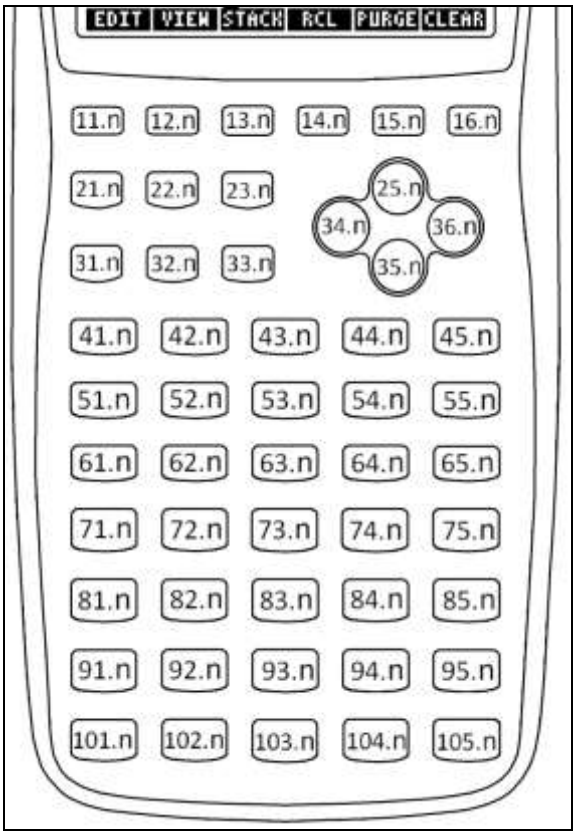
Ejemplo: para escribir la letra **α**, se presiona la tecla [ALPHA], luego la tecla [↱] y por último la tecla [F1 A].





CODIGOS DEL TECLADO

Cada tecla de la calculadora tienen un código general seguido de un código secundario, el código secundario indica de que nivel o estrato se trata (teclado primario, teclado de cambio izquierdo, etc.).



Los números que se observa son los códigos generales y después del punto está el código secundario representado por la letra n, la letra n puede tomar 6 valores de acuerdo al nivel o estrato seleccionado.

estrato	n
teclado primario	1
teclado de cambio izquierdo	2
teclado de cambio derecho	3
teclado alfabético	4
teclado alfabético y cambio izquierdo	5
teclado alfabético y cambio derecho	6

Ejemplo:

El código de la tecla donde se encuentra el número **2** es: **93.n** (ver los gráficos anteriores), el número **2** se encuentra en el estrato del teclado primario entonces n=1, por lo tanto el código de la tecla del número **2** es **93.1**.

COMANDOS

❶ **KEYEVAL** : Activa una función de una tecla como al presionarlo, requiere el código de la tecla.

SINTAXIS:

código_tecla **KEYEVAL** ⇨ activa_función

Ejemplo 1: colocar el número 1 en la pila.

El código de la tecla [1] es **92.1** (teclado primario).

KEYEVAL

Al ingresar el número 1 con **KEYEVAL** es como si se hubiera presionado la tecla 1, ahora solo falta presionar la tecla (**ENTER**).

Ejemplo 2: sacar la raíz cuadrada del número 16.

El código de la tecla [√x] es **52.1** (teclado primario).

KEYEVAL

❷ **KEYTIME→** : Devuelve el tiempo de accionamiento mínimo entre dos pulsaciones consecutivas de las teclas de la calculadora (valor por defecto 1138).

SINTAXIS:

KEYTIME→ ⇨ tiempo

❸ **→KEYTIME** : Establece el tiempo mínimo de accionamiento entre dos pulsaciones consecutivos de las teclas.

SINTAXIS:

tiempo **→KEYTIME** ⇨

Los dos últimos comandos son muy necesarios para configurar el tiempo de accionamiento de las teclas de la calculadora.

3 OBJETOS

Los objetos son los números, listas, vectores, matrices, programas, cadenas, etc. Se clasifican en 31 tipos diversos.

Número	Tipo	Ejemplo
0	Número real	-3.25
1	Número complejo	(6,-5)
2	Cadena	"hp 50g"
3	Sistema de números reales	[[1 2][3 4]]
4	Sistema de números complejos	[[(1,2) (3,–3)][(5,4) (3,1)]]
5	Lista	{ 10 s "hp 50g" }
6	Nombre global	X
7	Nombre local	J
8	Programa	«→ b h 'b*h' »
9	Objeto algebraico	'π*r^2'
10	Número entero binario	# EFAC11h
11	Objeto gráfico	Graphic 131 × 80
12	Objeto etiquetado	:Respuesta: 13
13	Objeto unidad	3_m/s
14	Nombre XLIB	XLIB 543 8
15	Directorio	DIR … END
16	Biblioteca	Library 1010: RIGIDECES
17	Objeto de reserva	Backup MYDIRECTORIO
18	Función incorporada	COS
19	Comando incorporado	CLEAR
20	Número entero binario interno	<123d>
21	Número real extendido	1.23E12
22	Número complejo extendido	(1.23E12, 1.234E10)
23	Serie enlazada	Serie enlazada
24	Objeto de carácter	Carácter
25	Objeto de código	Code
26	Datos de biblioteca	Datos de biblioteca
27	Objeto externo	External
28	Entero	3
29	Objeto externo	External
30	Objeto externo	External

COMANDOS

Estos comandos sirven para determinar el tipo de objeto que se encuentra en la pila o en una variable.

❶ **TYPE** : Obtiene el número del tipo de objeto.

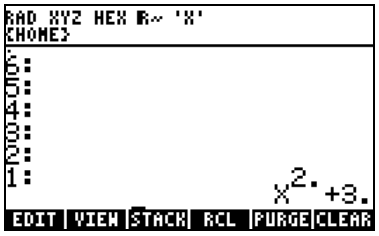
SINTAXIS:

objeto **TYPE** ⇨ n

Ejemplos:

TYPE

TYPE



TYPE

TYPE

❷ **VTYPE** : Obtiene el número del tipo de objeto que está almacenado en una variable, requiere el nombre de la variable.

SINTAXIS:

'variable' **VTYPE** ⇨ n

4 MANIPULACION DE LA PILA

La pila es una serie de ubicaciones de almacenamiento en la memoria, para números y otros objetos. Está formado por niveles en los cuales se visualiza los objetos almacenados para su utilización.

Ejemplo:

Del gráfico se observa la pila y se visualiza siete niveles.
En el primer nivel está el objeto: 2.x+5.
En el segundo nivel está el objeto: "HOLA"
En el tercer nivel está el objeto: 1.5
En los demás niveles no hay objetos.

COMANDOS PARA LA MANIPULACION DE LA PILA

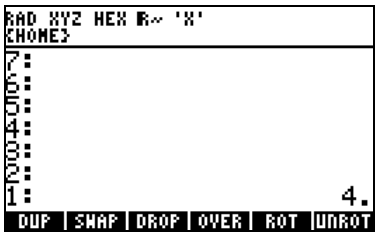
Manipulan los objetos que se encuentran en los niveles de la pila.

❶ **DUP** : Duplica el objeto que se encuentra en el nivel 1.

SINTAXIS:

obj	**DUP**	⇨	obj	obj

Ejemplo:



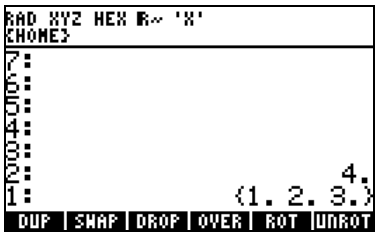
DUP

❷ **SWAP** : Intercambia los objetos que se encuentran en los niveles 1 y 2.

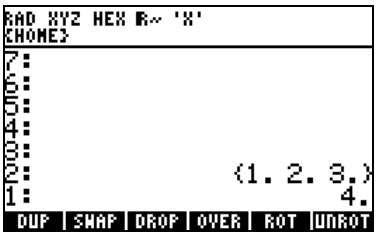
SINTAXIS:

obj_1	obj_2	**SWAP**	⇨	obj_2	obj_1

Ejemplo:



SWAP



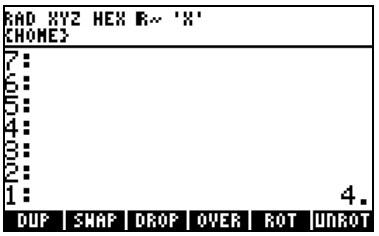
❸ **DROP** : Elimina el objeto que se encuentra en el nivel 1.

SINTAXIS:

obj **DROP** ⇨

Ejemplo:

DROP



❹ **OVER** : Crea una copia del objeto que se encuentra en el nivel 2.

SINTAXIS:

ob_1 obj_2 **OVER** ⇨ ob_1 obj_2 obj_1

Ejemplo:

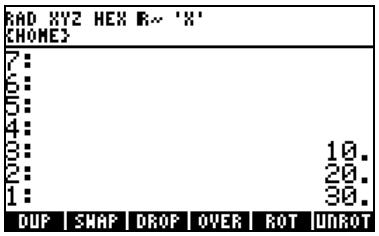
OVER

❺ **ROT** : Rota los objetos que se encuentran en los tres primeros niveles ascendentemente.

SINTAXIS:

obj_1 obj_2 obj_3 **ROT** ⇨ obj_2 obj_3 obj_1

Ejemplo:



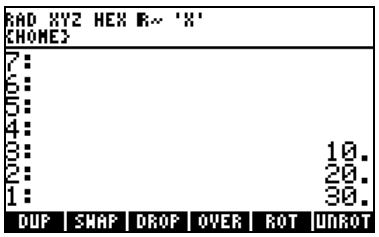
ROT

❻ **UNROT** : Rota los objetos que se encuentran en los tres primeros niveles descendentemente.

SINTAXIS:

obj_1	obj_2	obj_3	**UNROT**	⇨	obj_3	obj_1	obj_2

Ejemplo:



UNROT

❼ **ROLL** : Rota los objetos que se encuentran en los n primeros niveles ascendentemente, requiere el número de objetos a rotar.

SINTAXIS:

obj1	obj2	obj3 … objn	n	**ROLL**	⇨	obj2	obj3 … objn	obj1

Ejemplo:

4 **ROLL**

❽ **ROLLD** : Rota los objetos que se encuentran en los n primeros niveles descendentemente, requiere el número de objetos a rotar.

SINTAXIS:

obj1	obj2	obj3 … objn	n	**ROLLD**	⇨	objn	obj1	obj2	obj3 …

Ejemplo:

4 **ROLLD**

❾ **PICK** : Copia al nivel 1 cualquier objeto de los niveles, requiere el número del nivel que se desea copiar.

SINTAXIS:

obj1 …	obji …	objn	i	**PICK**	⇨	obj1 …	obji …	objn	obji

Ejemplo:

3 **PICK**

❿ **UNPICK** : Corta el objeto del nivel 1 y lo reemplaza al objeto de nivel i+1, requiere la posición i.

SINTAXIS:

objn …	obji …	obj2	obj1	i	**UNPICK**	⇨	objn …	obj1	obji … obj2

Ejemplo:

3 **UNPICK**

⓫ **PICK3** : Copia al nivel 1 el objeto que se encuentra en el nivel 3.

SINTAXIS:

obj_1	obj_2	obj_3	**PICK3**	⇨	obj_1	obj_2	obj_3	obj_1	

Ejemplo:

PICK3

⓬ **DEPTH** : Devuelve la cantidad de niveles que está en uso.

SINTAXIS:

obj_1	obj_2 … obj_n	**DEPTH** ⇨	obj_1	obj_2 … obj_n	n

Ejemplo:

DEPTH

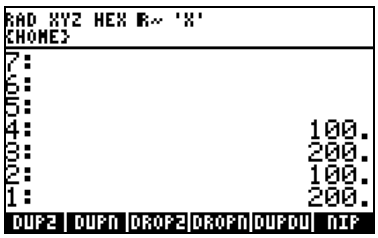
⓭ **DUP2** : Duplica los objetos que se encuentran en los niveles 1 y 2.

SINTAXIS:

obj_1	obj_2	**DUP2** ⇨	obj_1	obj_2	obj_1	obj_2

Ejemplo:

DUP2



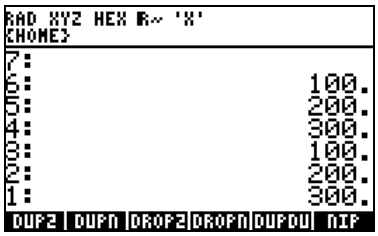
⓮ **DUPN** : Hace una copia de los n objetos que se encuentran en la pila, requiere el número de objetos a copiar.

SINTAXIS:

obj1	obj2 … objn	n	**DUPN** ⇨	obj1	obj2 … objn	obj1	obj2 … objn

Ejemplo:

3 **DUPN**



⓯ **DROP2** : Elimina los objetos que se encuentran en los dos primeros niveles.

SINTAXIS:

obj_n … obj_3 obj_2 obj_1 **DROP2** ⇨ obj_n … obj_3

Ejemplo:

DROP2

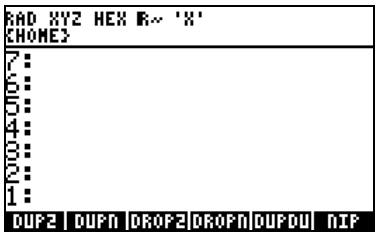
⓰ **DROPN** : Elimina los objetos que se encuentran en los n niveles, requiere la cantidad de niveles que se desea eliminar.

SINTAXIS:

obj_1 obj_2 … obj_n n **DROPN** ⇨

Ejemplo:

3 **DROPN**



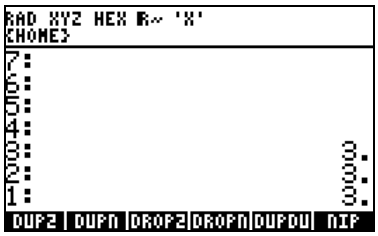
⓱ **DUPDUP** : triplica el objeto que se encuentra en el nivel 1.

SINTAXIS:

obj **DUPDUP** ⇨ obj obj obj

Ejemplo:

DUPDUP



⓲ **NIP** : Elimina el objeto que se encuentra en el nivel 2.

SINTAXIS:

obj_1	obj_2	**NIP**	⇨	obj_2

Ejemplo:

NIP

⓳ **NDUPN** : Multiplica el objeto del nivel 1 hasta obtener n objetos y además devuelve el número n, requiere la cantidad de objetos a obtener.

SINTAXIS:

obj	n	**NDUPN**	⇨	obj	obj …	obj	(n veces el objeto obj) n

Ejemplo:

3 **NDUPN**

⓴ **CLEAR** : Elimina todos los objetos de la pila.

SINTAXIS:

obj_1	obj_2 … obj_n	**CLEAR**	⇨	

5 FUNCIONES ESPECIALES

Estas funciones no son las comunes, estas se usan generalmente para programación.

❶ **%** : Calcula el x por ciento de y.

SINTAXIS:

x	y	**%**	⇨	x*y/100

❷ **%CH** : Calcula el cambio porcentual.

SINTAXIS:

x	y	**%CH**	⇨	100*(y-x)/x

❸ **%T** : Calcula la porción de un número que representa de otro en porcentaje.

SINTAXIS:

x	y	**%T**	⇨	(y/x)*100

❹ **MIN** : Devuelve el menor de dos números.

SINTAXIS:

x	y	**MIN**	⇨	mínimo(x, y)

Ejemplo:

MIN

❺ **MAX** : Devuelve el mayor de dos números.

SINTAXIS:

x	y	**MAX**	⇨	máximo(x, y)

Ejemplo:

MAX

❻ **MOD** : Este es un operador. Obtiene el residuo de dividir dos números enteros.

SINTAXIS:

x y **MOD** ⇨ residuo(x/y)

Ejemplos:

MOD

MOD

❼ **ABS** : Calcula el valor absoluto de un número.

SINTAXIS:

x **ABS** ⇨ |x|

Ejemplo:

ABS

❽ **SIGN** : Determina el signo de un número.
Si el número es negativo devuelve: -1.
Si el número es positivo devuelve: 1.
Si el número es cero devuelve: 0.

SINTAXIS:

x	**SIGN**	⇨	signo(x)

Ejemplo:

SIGN

❾ **MANT** : Determina la mantisa de un número basado en LOG_{10}.

SINTAXIS:

x	**MANT**	⇨	mantisa(x)

❿ **XPON** : Determina el exponente de la base 10 de un número, cuando este se encontraría en el formato científico.

SINTAXIS:

x	**XPON**	⇨	n

Ejemplo:

XPON

En este ejemplo el número 12365 es equivalente en el formato científico a: $1.2365*10^4$ y el exponente de 10 es 4.

⓫ **IP** : Determina la parte entera de un número real.

SINTAXIS:

x	**IP**	⇨	n

Ejemplos:

IP

IP

⓬ **FP** : Determina la parte decimal de un número real.

SINTAXIS:

x **FP** ⇨ y

Ejemplos:

FP

FP

⓭ **RND** : Redondea un número(x) con la cantidad de decimales deseados (y).

SINTAXIS:

x y **RND** ⇨ z

Ejemplo:

RND

⓮ **TRNC** : Trunca un número(x) con la cantidad de decimales deseados(y).

SINTAXIS:

x y **TRNC** ⇨ z

Ejemplo:

TRNC

⓯ **FLOOR** : Halla el entero más cercano que es menor o igual que el número indicado.

SINTAXIS:

x **FLOOR** ⇨ y

Ejemplos:

FLOOR

FLOOR

⓰ **CEIL** : Halla el entero más cercano que es mayor o igual que el número indicado.

SINTAXIS:

x **CEIL** ⇨ y

Ejemplos:

CEIL

CEIL

⓱ **D→R** : Convierte grados sexagesimales a radianes.

SINTAXIS:

grados **D→R** ⇨ rad

⓲ **R→D** : Convierte radianes a grados sexagesimales.

SINTAXIS:

rad **R→D** ⇨ grados

⓳ **RAND** : Genera un número aleatorio (pseudoaleatorio) comprendido entre 0 y 1.

SINTAXIS:

RAND ⇨ x

⓴ **→NUM** : Convierte un valor exacto en un valor equivalente aproximado.

SINTAXIS:

n **→NUM** ⇨ x

Ejemplo:


RAD XYZ HEX R= 'X'
{HOME}
1:
3/2+3/5
EDIT VIEW STACK RCL PURGE CLEAR

→NUM


RAD XYZ HEX R= 'X'
{HOME}
1: 2.1
EDIT VIEW STACK RCL PURGE CLEAR

6 LISTAS

Las listas son colecciones de objetos que están agrupados entre llaves. Las listas son los objetos que más se usan en programación ya que tienen: comandos, funciones y operadores con los que se pueden manipular sus elementos u objetos contenidos en ellas.

Ejemplos:

La primera lista tiene 4 elementos y los elementos son los objetos que están dentro de las llaves y los objetos pueden ser: números, cadenas, listas, matrices, gráficos, programas, etc.

El primer elemento de la primera lista es: 1.
El segundo elemento de la primera lista es: 3.
El tercer elemento de la primera lista es: "EJE"
El cuarto elemento de la primera lista es: 1.3
La dimensión o número de elementos de la primera lista es: 4

La segunda lista es una lista sin elementos o una lista vacía.

COMPOSICION DE UNA LISTA

Es la obtención de la misma a partir de objetos.

❶ →LIST : Construye una lista con los objetos de la pila, requiere la cantidad de objetos (n) que contendrá la lista.

SINTAXIS:

obj1	obj2	…	objn	n	→LIST ⇨	{obj1	obj2	…	objn}

obj1 obj2 … objn : son los objetos que tendrá la lista.
n : indica la cantidad de objetos que tendrá la lista, este valor debe estar en el primer nivel de la pila.

Ejemplo 1:

→LIST

En el nivel 1 está la cantidad de objetos o elementos que tendrá la lista, en este caso 4 y en el nivel 5, 4, 3 y 2 están los objetos que formaran la lista.

Ejemplo 2:

→LIST

En el nivel 1 está la cantidad de objetos o elementos que tendrá la lista, en este caso 3 por eso solo tomó los objetos de los niveles 4, 3 y 2 y el objeto del nivel 5 no lo consideró para la lista.

DESCOMPOSICION DE LISTAS

Es la obtención de los elementos de una lista.

❶ **LIST→** : Descompone una lista en sus elementos y devuelve el número de elementos de la lista (es lo contrario de **→LIST**).

SINTAXIS:

{obj1 obj2 … objn} **LIST→** ⇨ obj1 obj2 … objn n

Ejemplo:

LIST→

❷ **OBJ→** : Descompone una lista en sus elementos y devuelve el número de elementos de la lista. Es idéntico a **LIST→** cuando se trata de listas.

SINTAXIS:

{obj_1 obj_2 … obj_n} **OBJ→** ⇨ obj_1 obj_2 … obj_n n

Ejemplo:

OBJ→

❸ **EVAL** : Descompone una lista y se obtiene solo los elementos de la lista.

SINTAXIS:

{obj_1 obj_2 … obj_n} **EVAL** ⇨ obj_1 obj_2 … obj_n

Ejemplo:

EVAL

OPERACIONES CON LISTAS

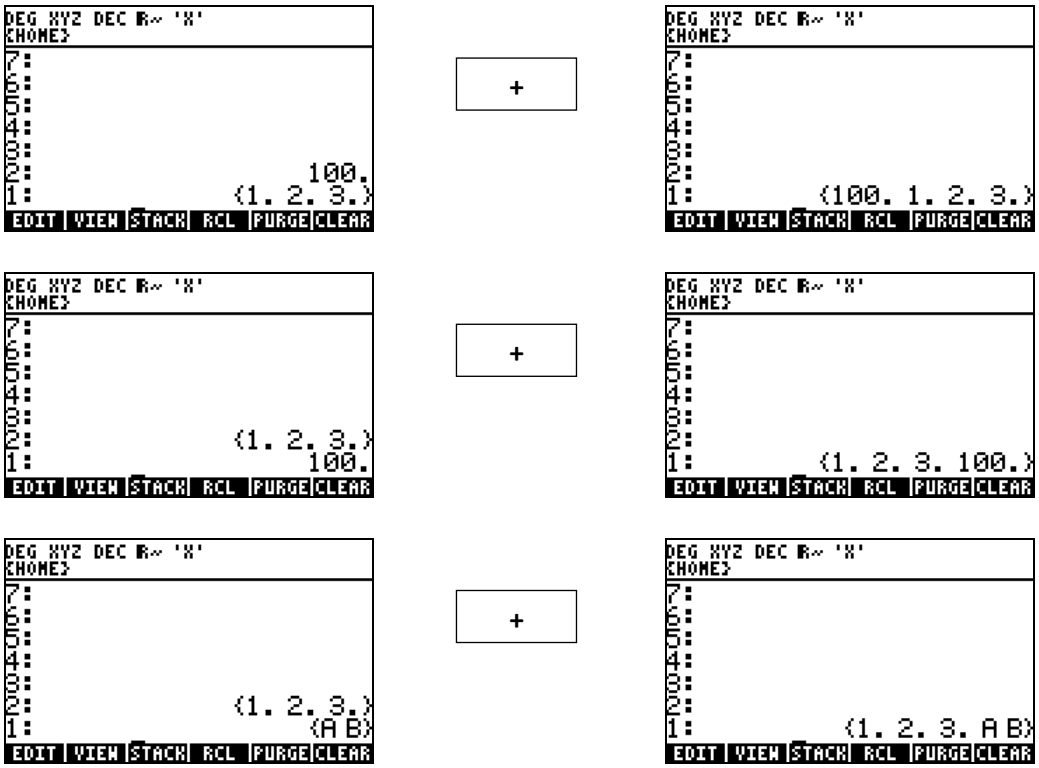
Las operaciones se pueden realizar entre listas, entre un objeto y una lista o una lista y un objeto. Para hacer las operaciones entre listas estas deben tener la misma cantidad de elementos a excepción de la suma.

❶ **+** : Añade un objeto o los elementos de una lista a otra lista, obteniendo una nueva lista, en donde sus elementos son todos los objetos (el orden importa).

SINTAXIS:

①	objeto lista_1	+	⇨	lista_2
②	lista_1 objeto	+	⇨	lista_2
③	lista_1 lista_2	+	⇨	lista_3

Ejemplos:



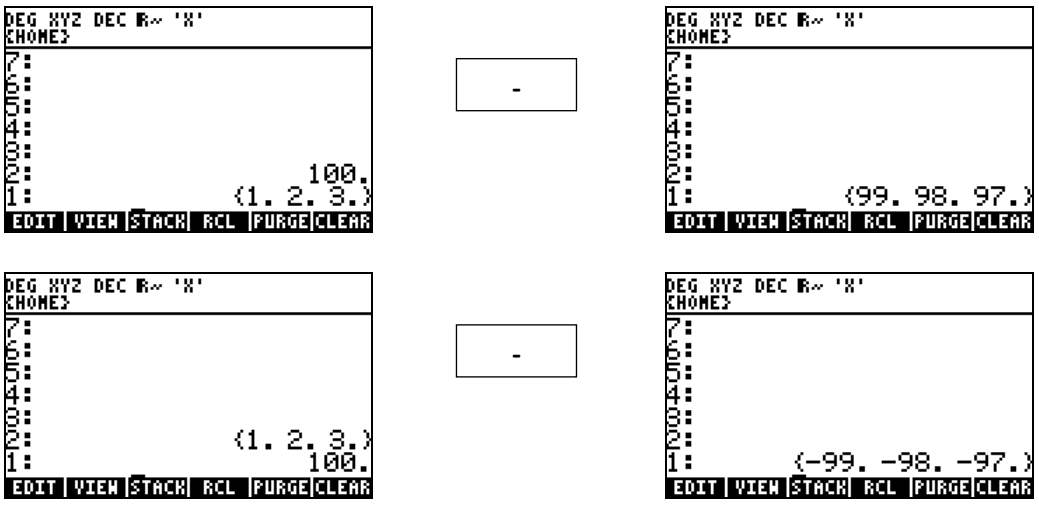
Se observa que el orden importa.

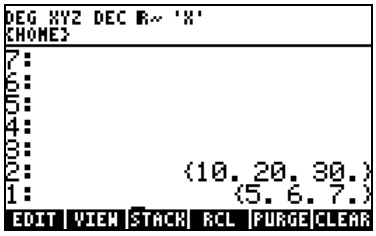
❷ [-] : Halla la diferencia entre un objeto y los elementos de una lista o viceversa o la diferencia entre los elementos que ocupan la misma posición de dos listas de igual dimensión (el orden importa).

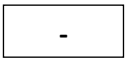
SINTAXIS:

①	objeto lista_1	-	⇨	lista_2
②	lista_1 objeto	-	⇨	lista_2
③	lista_1 lista_2	-	⇨	lista_3

Ejemplos:







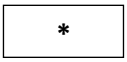
Se observa que el orden importa.

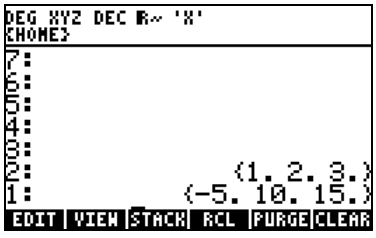
❸ [*] : Halla el producto entre un objeto y los elementos de una lista o viceversa, o el producto entre los elementos que ocupan la misma posición de dos listas de igual dimensión.

SINTAXIS:

①	objeto lista_1	*	⇨	lista_2
②	lista_1 objeto	*	⇨	lista_2
③	lista_1 lista_2	*	⇨	lista_3

Ejemplos:



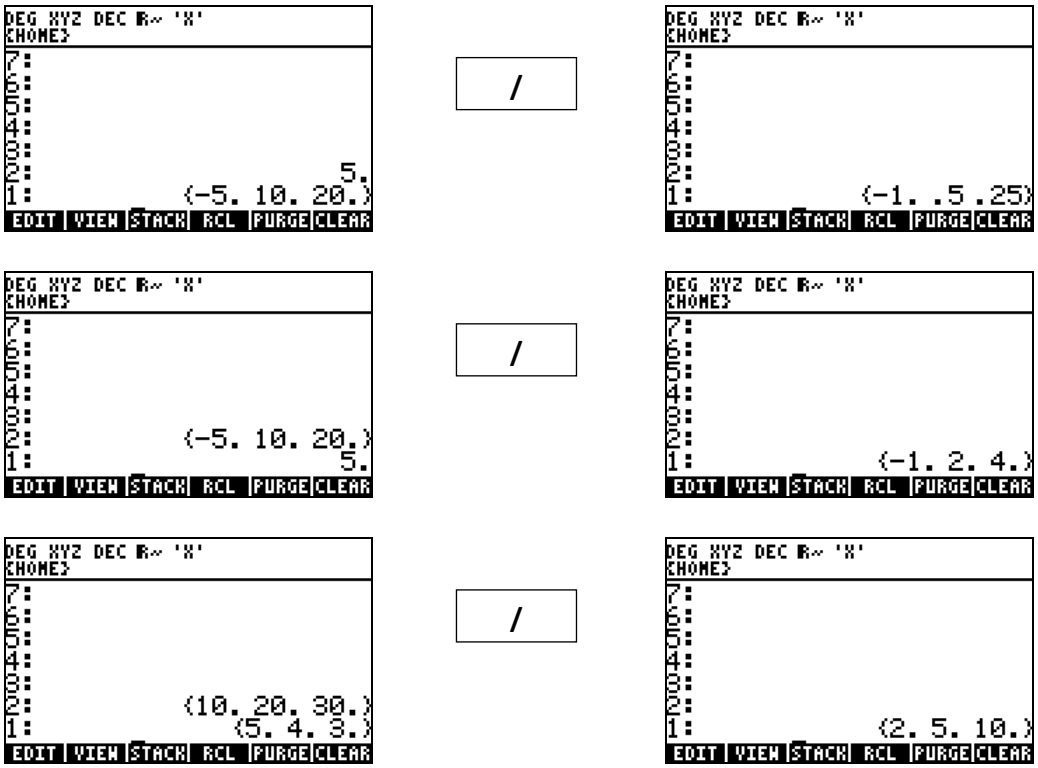


❹ [/] : Halla la división entre un objeto y todos los elementos de una lista o viceversa o la división entre los elementos que ocupan la misma posición de dos listas de igual dimensión.

SINTAXIS:

①	objeto	lista_1	/	⇨	lista_2
②	lista_1	objeto	/	⇨	lista_2
③	lista_1	lista_2	/	⇨	lista_3

Ejemplos:



Se observa que el orden importa.

MANIPULACION DE LISTAS

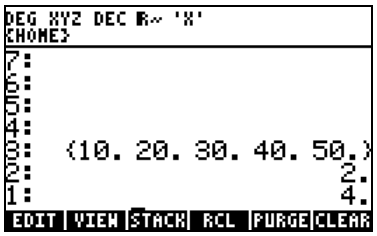
Se refiere al cambio de elementos ya sea por otro u otros, obtención de una parte de una lista, etc.

❶ **SUB** : Obtiene una lista cuyos elementos son parte de los elementos de la lista Inicial, requiere la posición inicial y final de la lista inicial.

SINTAXIS:

{obj_1…obj_i … obj_j … obj_n}	i	j	**SUB**	⇨	{obj_i … obj_j}

Ejemplos:



SUB

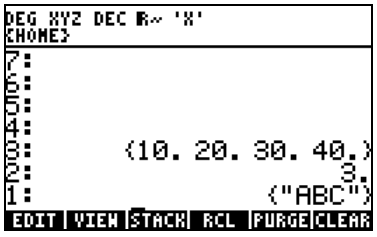
SUB

❷ **REPL** : Reemplaza uno o varios elementos de una lista por otros elementos de otra lista, desde la posición indicada.

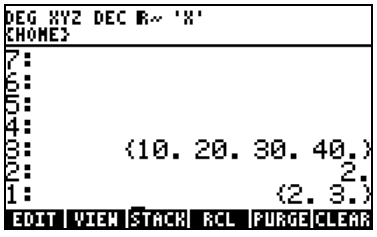
SINTAXIS:

{obj1 obj2 … obj**i** … objn} **i** {obje1… objej} **REPL** ⇨ {obj1 obj2… obje … objej … objn}

Ejemplos:



REPL



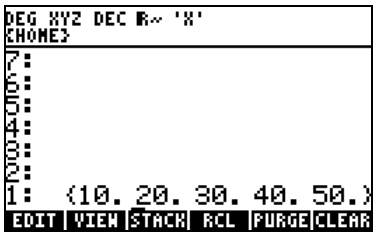
REPL

❸ **TAIL** : Elimina el primer elemento de una lista.

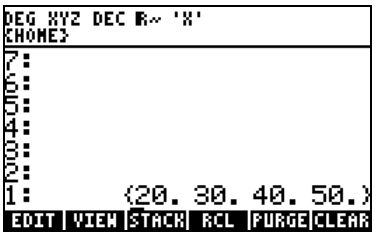
SINTAXIS:

{obj1 obj2 obj3 … objn} **TAIL** ⇨ {obj2 obj3 … objn}

Ejemplo:



TAIL



MANIPULACION DE LOS ELEMENTOS DE UNA LISTA

Se refiere a la obtención de un elemento o reemplazar un elemento de una lista por otro, etc.

❶ **GET** : Obtiene un elemento de una lista, requiere la posición del elemento a obtener.

SINTAXIS:

{obj_1 obj_2 … obj_i … obj_n} i **GET** ⇨ obj_i

Ejemplo:

GET

❷ **GETI** : Este comando obtiene la lista original, la posición del siguiente elemento y el elemento de posición indicada.

SINTAXIS:

{obj_1 … obj_i … obj_n} i **GETI** ⇨ {obj_1 … obj_i … obj_n} i+1 obj_i

Ejemplos:

GETI

GETI

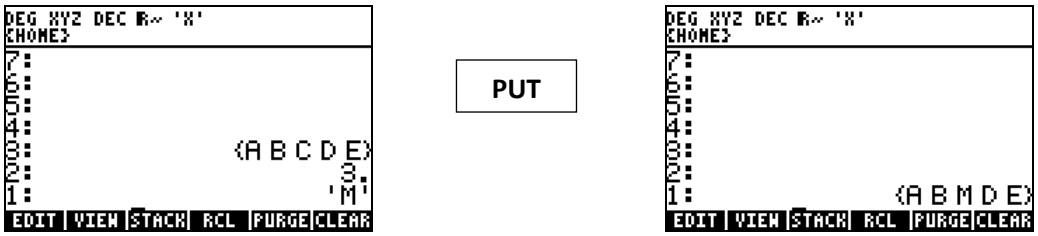
Cuando se indica la posición del último elemento la posición del siguiente elemento será 1.

❸ **PUT** : Este comando reemplaza un objeto de una lista por otro objeto, requiere la posición y el objeto por el cual reemplazar.

SINTAXIS:

{obj_1 … obj_i … obj_n} i objeto **PUT** ⇨ {obj_1 … objeto … obj_n}

Ejemplo:

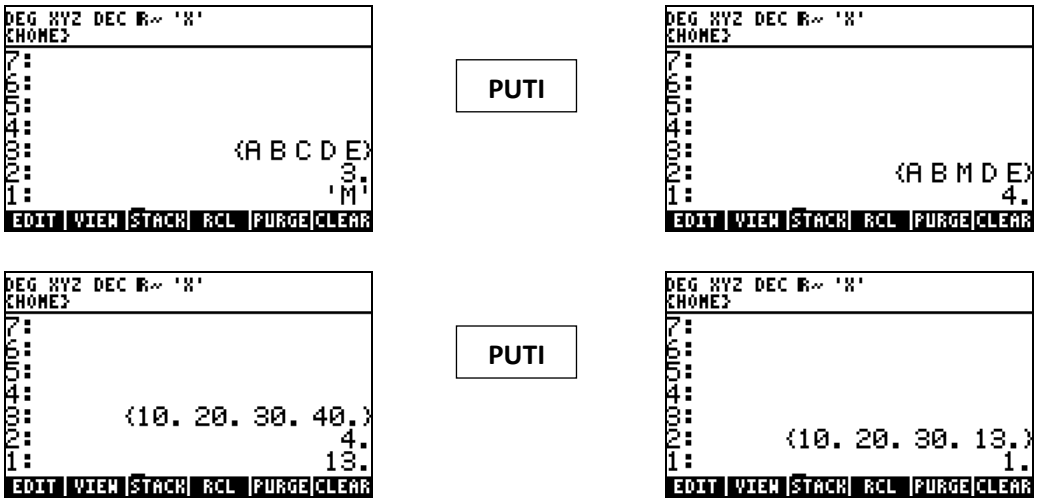


❹ **PUTI** : Este comando requiere los mismos argumentos que **PUT** y se obtiene el mismo resultado y además la posición del siguiente elemento.

SINTAXIS:

{obj_1 … obj_i … obj_n} i objeto **PUTI** ⇨ {obj_1 … objeto … obj_n} i+1

Ejemplos:



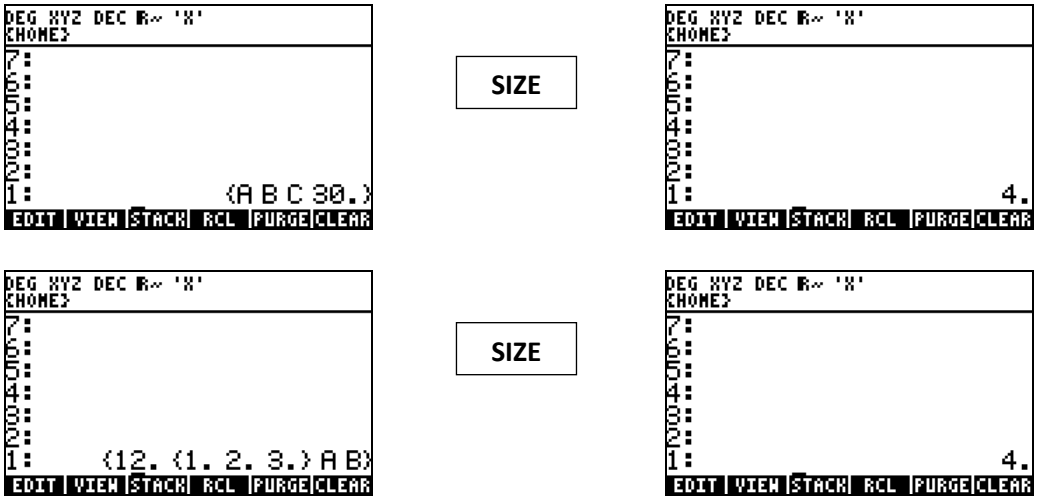
Cuando se indica la posición del último elemento la posición del siguiente elemento será 1.

❺ **SIZE** : Obtiene la dimensión o el número de elementos de una lista.

SINTAXIS:

{obj_1 obj_2 … obj_n} **SIZE** ⇨ n

Ejemplos:



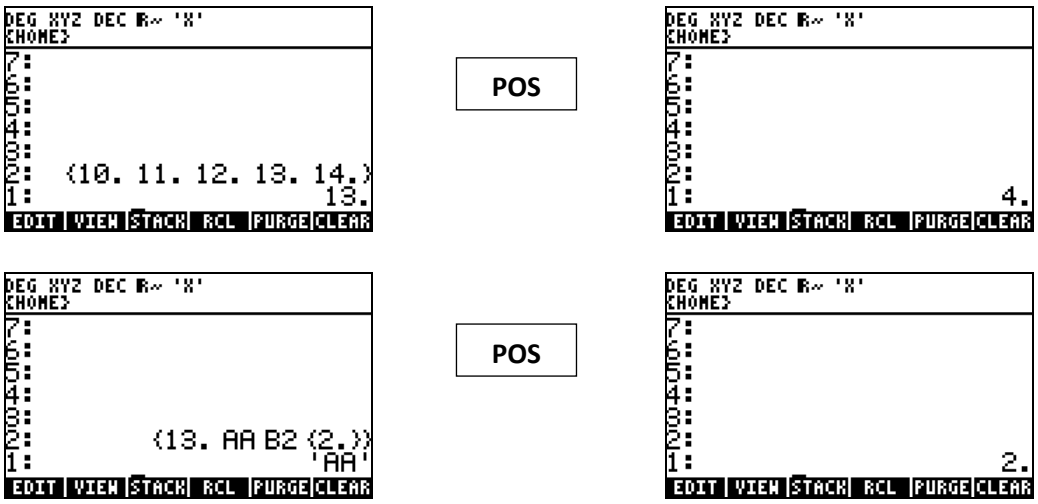
En el segundo ejemplo {1. 2. 3.} es un objeto por eso se considera como un solo elemento, si la lista no tiene elementos se obtiene 0.

❻ **POS** : Obtiene la posición de un objeto contenido en una lista, requiere el objeto. Este comando devuelve la posición del primer objeto que sea igual al buscado (este comando empieza a buscar el objeto en la lista de izquierda a derecha).

SINTAXIS:

{obj_1 obj_2 … obj_i … obj_n} obj_i **POS** ⇨ i

Ejemplos:



Si el objeto no está en la lista da como resultado 0.

❼ **HEAD** : Obtiene el primer objeto de una lista.

SINTAXIS:

{obj_1 obj_2 … obj_n} **HEAD** ⇨ obj_1

Ejemplos:

El primer objeto de la lista del segundo ejemplo es {12. 3.}.

PROCEDIMIENTOS EN UNA LISTA

Se refiere a la obtención de otra lista a partir de la lista original usando funciones u operaciones a los elementos o entre los elementos de una o más listas.

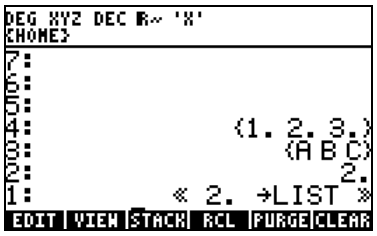
❶ **DOLIST** : Ejecuta un programa o una función sucesivamente a todos los elementos que tienen la misma posición de un grupo de n listas que tienen la misma dimensión, requiere la cantidad de listas a utilizar y el programa o función a ejecutar.

SINTAXIS:

lista_1 lista_2 … lista_n n <<programa>> **DOLIST** ⇨ lista

Ejemplo 1: con dos listas obtener otra lista, donde sus elementos son la suma de los elementos que tienen la misma posición de las dos listas iníciales.

Ejemplo 2: con dos listas obtener otra lista, donde sus elementos son listas cuyos elementos son los elementos que tienen la misma posición de las dos listas iníciales.



DOLIST

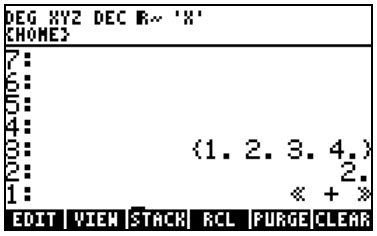
❷ **DOSUBS** : Ejecuta un programa a sucesivos grupos de elementos de una lista o ejecuta un programa o función a cada uno de los elementos de una lista.

SINTAXIS:

lista_1	n	<<programa>>	**DOSUBS**	⇨	lista_2

n : es el número de elementos que utilizará en cada procedimiento.

Ejemplo 1: con una lista obtener otra lista, donde sus elementos son la suma de dos elementos consecutivos de la lista inicial.



DOSUBS

Ejemplo 2: con una lista obtener otra donde sus elementos estén contenidos en listas.

DOSUBS

NSUB : Devuelve la posición del proceso que se está efectuando dentro de **DOSUBS**.

Ejemplo: con una lista obtener otra donde sus elementos sean listas de dos elementos en donde los primeros elementos son los elementos de la lista inicial y los segundos elementos son las posiciones de los elementos de la lista inicial.

DOSUBS

ENDSUB : Devuelve la cantidad de procesos que se efectuara dentro de **DOSUBS**.

❸ **STREAM** : Ejecuta un programa o función a los primeros elementos de una lista, luego al resultado obtenido ejecuta el programa o función con los siguientes elementos requeridos y a este nuevo resultado ejecuta el programa o función con los siguientes elementos requeridos y así sucesivamente hasta ejecutarlo con los últimos elementos.

SINTAXIS:

lista	<<programa>>	**STREAM**	⇨	obj

Ejemplo 1: obtener la suma de los elementos de una lista.

STREAM

Ejemplo 2: restar sucesivamente del primer elemento de una lista los demás elementos.

STREAM

Lo que hizo es lo siguiente: a 100 lo restó 20 obteniendo 80 y a este resultado lo vuelve a restar el siguiente elemento en este caso 30 obteniendo 80-30=50.

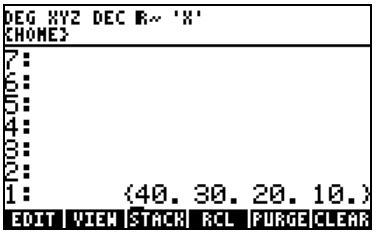
❹ **REVLIST** : Invierte el orden de los elementos de una lista.

SINTAXIS:

{obj_1 obj_2 … obj_n}	**REVLIST**	⇨	{ obj_n … obj_2 obj_1}	

Ejemplos:

REVLIST



REVLIST

❺ **SORT** : Ordena los elementos de una lista numéricamente o alfabéticamente en orden ascendente.

SINTAXIS:

lista_1	**SORT** ⇨	lista_2

Ejemplos:

SORT

SORT

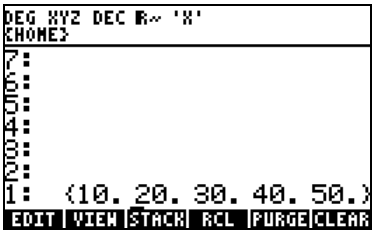
❻ **SEQ** : Reemplaza los valores de una sucesión, el cual varía desde un valor inicial hasta un valor final, con un incremento indicado en una variable de una función. Devuelve en una lista los valores obtenidos.

SINTAXIS:

función	variable	inicio	fin	incremento	**SEQ** ⇨	lista

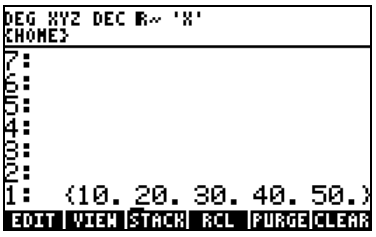
Ejemplo: obtener una lista, cuyos elementos son los números desde10 al 50 de 10 en 10.

SEQ



También se puede obtener de esta otra forma.

SEQ



❼ **ΔLIST** : Calcula una lista en donde sus elementos son el incremento de dos elementos consecutivos de otra lista.

SINTAXIS:

{obj_1 obj_2 … obj_n} **ΔLIST** ⇨ {obj_2-obj_1 obj_3-obj_2 … obj_n- obj_n-1}

Ejemplos:

ΔLIST

ΔLIST

❽ **ΣLIST** : Calcula la suma de todos los elementos de una lista.

SINTAXIS:

{obj_1 obj_2 obj_3 … obj_n} **ΣLIST** ⇨ obj_1+obj_2+obj_3 … +obj_n

Ejemplo:

ΣLIST

❾ **∏LIST** : Calcula el producto de todos los elementos de una lista.

SINTAXIS:

{obj_1 obj_2 … obj_n} **∏LIST** ⇨ obj_1*obj_2* … *obj_n

Ejemplo:

∏LIST

❿ **ADD** : Suma miembro a miembro los elementos de dos listas de la misma dimensión o un objeto a todos los elementos de una lista.

SINTAXIS:

{obj1 obj2… objn} {obje1 obje2… objen} **ADD** ⇨ {obj1+obje1 obj2+obje2… objn+objen}

Ejemplos:

ADD

ADD

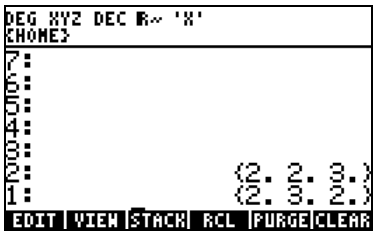
ADD

FUNCIONES Y OPERADORES EN LISTAS

Todas las funciones que se aplican a números o expresiones, se pueden aplicar a las listas.

Ejemplos: se aplicarán algunas funciones a una o dos listas.

SIN



^

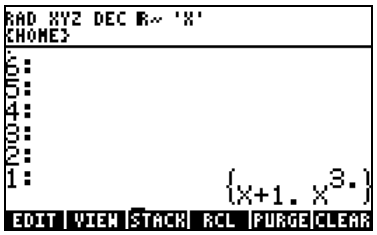
^

ABS

LOG

IP

CEIL



DERV

SIGN

EJEMPLOS DE MANIPULACIONES DE LISTAS

Ejemplo 1: obtener el menor valor de la siguiente lista: {5. 8. 9. 4. 3. 7.}

① Ya ingresado la lista a la calculadora se ordena los elementos de la lista de menor a mayor usando el comando **SORT**.

SORT

② En la lista ordenada el elemento de menor valor es el primero, se obtiene el primer elemento con el comando **HEAD**.

HEAD

Si se hace un programa quedaría de la siguiente forma:

Ejemplo 2: obtener el mayor valor de la siguiente lista: {5. 8. 9. 4. 3. 7.}.

① Ya ingresado la lista a la calculadora se ordena los elementos de la lista de menor a mayor usando el comando **SORT**.

② Ya ordenado de observa que el mayor valor es el último elemento de la lista ordenada, se invierte la lista usando el comando **REVLIST**.

③ El mayor valor es el primer elemento de la lista resultante se lo extrae con el comando **HEAD**.

Si se hace un programa quedaría de la siguiente forma:

Ejemplo 3: obtener el promedio de la siguiente lista de números: {5. 8. 9. 4. 3. 7.}.

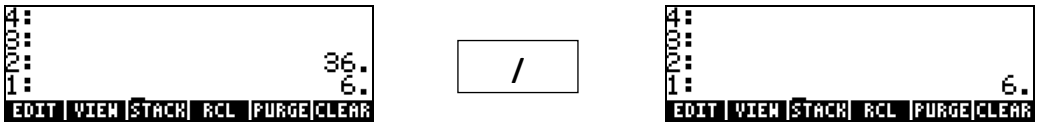
① Ya ingresado la lista a la calculadora, es necesario la suma de los elementos y la cantidad de elementos de la lista, se duplica la lista usando el comando **DUP**.

② Ya duplicado la lista se halla la suma de los elementos de la lista del nivel 1 usando el comando **ΣLIST**.

③ Ahora se tiene que intercambiar la posición del número y la lista usando el comando **SWAP**.

④ Se halla la cantidad de elementos de la lista usando el comando **SIZE**.

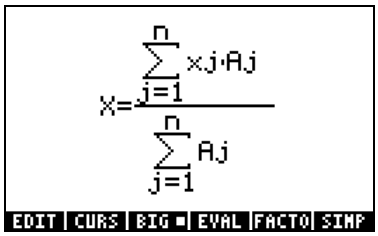
⑤ Se divide la suma de los elementos de la lista y la cantidad de elementos, obteniendo el promedio de los elementos de la lista.



Si se hace un programa quedaría de la siguiente forma:

Ejemplo 4: obtener la abscisa del centro de gravedad de un grupo de regiones en donde sus áreas son: {8. 4. 6. 5.}, las abscisas de los centros de gravedad de las regiones son: {1. 4. 3. 2.} respectivamente.

① La fórmula para hallar la abscisa del centro de gravedad es la siguiente.

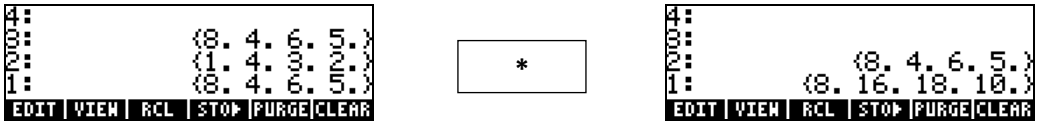


② Una vez ingresado las dos listas, se intercambia la posición de las listas usando el comando **SWAP**.

③ Se duplica la lista de las áreas usando el comando **DUP**.

④ Se rota las tres listas de arriba hacia abajo usando el comando **UNROT**.

⑤ Se multiplican las abscisas con sus respectivas áreas usando el operador *****.



⑥ Se suman los elementos de la lista obtenida usando el comando **ΣLIST**.

⑦ Se intercambia la posición de los objetos usando el comando **SWAP**.

⑧ Se suman las áreas de las regiones usando el comando **ΣLIST**.

ΣLIST

⑨ Se divide los valores obtenidos usando el operador **/**.

/

Si se hace un programa quedaría de la siguiente forma:

7 VECTORES

El vector es un arreglo de dos o más elementos, dispuestos en una fila o columna y se denominan vectores fila y vectores columna respectivamente.

Ejemplo:

Este vector tiene 4 elementos, los elementos son los objetos que están dentro de los corchetes.

El primer elemento del vector es: 1.
El segundo elemento del vector es: 2.
La dimensión del vector es: 4.

CONSTRUCCION DE UN VECTOR

Se puede construir de varias formas, las más comunes son dos, por la línea de comandos y por el editor de matrices.

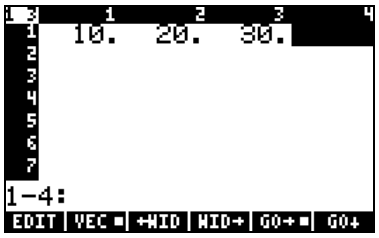
CONSTRUCCION POR LA LINEA DE COMANDOS: Se tiene que ingresar los elementos de los vectores entre corchetes.

Ejemplo:

CONSTRUCCION POR EL EDITOR DE MATRICES: Se ingresa al editor presionando las teclas [↰] y [' 0].

Ejemplo:

Una vez abierto el editor de matrices se ingresa los datos en las casillas correspondientes.



Se debe tener en cuenta del menú **VEC**, este menú debe estar activo para ingresar un vector. Si el menú no está activo se ingresará una matriz.

CONSTRUCCION DE UN VECTOR UTILIZANDO COMANDOS

❶ **→ARRY** : Construye un vector con los n elementos de la pila, requiere la cantidad de elementos.

SINTAXIS:

ele_1	ele_2	…	ele_n		n		**→ARRY**	⇨	[ele_1	ele_2	… ele_n]

Ejemplo 1:

❷ **CON** : Construye un vector de elementos iguales, requiere el número de elementos en una lista y el elemento que se repite.

SINTAXIS:

	{n}	elemento	**CON**	⇨	vector	

Ejemplo:

CON

MANEJO DE VECTORES

Se refiere a la manipulación de sus elementos.

❶ **ARRY→** : Descompone un vector en sus elementos además devuelve su dimensión en una lista. Es lo contrario del comando **→ARRY**.

SINTAXIS:

[ele_1 ele_2 … ele_n]	**ARRY→**	⇨	ele_1 ele_2 … ele_n	{n}

Ejemplo:

ARRY→

❷ **OBJ→** : Descompone un vector en sus elementos además devuelve su dimensión en una lista.

SINTAXIS:

[ele_1 ele_2 … ele_n]	**OBJ→**	⇨	ele_1 ele_2 … ele_n	{n}

Ejemplo 1:

OBJ→

❸ **GET** : Extrae un elemento de un vector, requiere la posición del elemento.

SINTAXIS:

[ele_1 … ele_i … ele_n]	i	**GET**	⇨	ele_i

Ejemplo:

GET

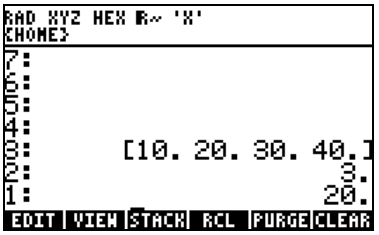
❹ **GETI** : Devuelve el vector original, la posición del siguiente elemento y un elemento del vector, requiere la posición del elemento a extraer.

SINTAXIS:

[ele_1 … ele_i … ele_n] i **GETI** ⇨ [ele_1 … ele_i … ele_n] i+1 ele_i

Ejemplo:

GETI



❺ **PUT** : Remplaza un elemento por otro elemento en un vector, requiere la posición a reemplazar y el objeto por cual reemplazar.

SINTAXIS:

[ele_1 … ele_i … ele_n] i elemento **PUT** ⇨ [ele_1 … elemento … ele_n]

Ejemplo:

PUT

❻ **PUTI** : Remplaza un elemento de un vector por otro elemento, además devuelve la posición siguiente elemento, requiere la posición para reemplazar y el objeto por cual reemplazar.

SINTAXIS:

[ele_1 … ele_i … ele_n] i elem **PUTI** ⇨ [ele_1 … elem … ele_n] i+1

Ejemplo:

PUTI

❼ **REPL** : Remplaza parte de los elementos de un vector por los elementos de otro vector, requiere la posición desde donde se va a reemplazar y el vector por cual se va a reemplazar.

SINTAXIS:

[ele1 ele2 … elei… elen] i [elem1 … elemj] **REPL** ⇨ [ele1 ele2… elem1 … elemj… elen]

Ejemplo:

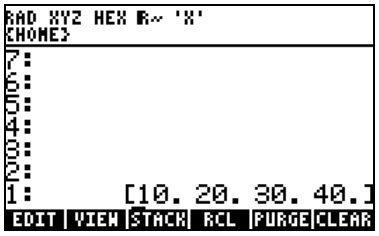
REPL

❽ **SIZE** : Devuelve la dimensión de un vector en una lista.

SINTAXIS:

[ele_1 ele_2… ele_n] **SIZE** ⇨ {n}

Ejemplo:



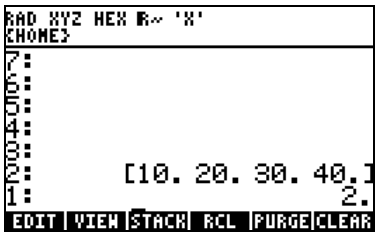
SIZE

❾ **COL-** : Extrae un elemento de un vector, requiere la posición del elemento a extraer.

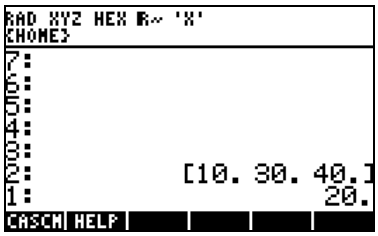
SINTAXIS:

[ele_1 … ele_i… ele_n]	i	**COL-**	⇨	[ele_1 … ele_n]	ele_i

Ejemplo:



COL-



❿ **COL+** : Inserta un elemento en un vector, requiere el elemento y la posición donde insertar.

SINTAXIS:

vector_1	element	i	**COL+**	⇨	vector_2

Ejemplo:

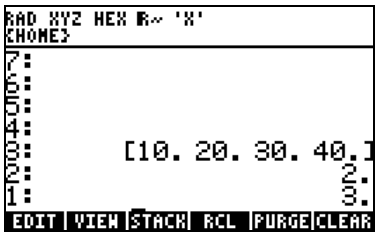
COL+

⓫ **CSWP** : Intercambia la posición de dos elementos de un vector, requiere las posiciones de los elementos a intercambiar.

SINTAXIS:

[ele_1 … ele_i… ele_j … ele_n]	i	j	**CSWP**	⇨	[ele_1 … ele_j… ele_i … ele_n]

Ejemplo:



CSWP

OPERACIONES CON VECTORES

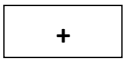
Las operaciones que se pueden realizar entre dos vectores de la misma dimensión son la suma y resta; la multiplicación o división de un vector con un escalar.

❶ **+** : Suma los elementos que tienen la misma posición de dos vectores que tienen la misma dimensión.

SINTAXIS:

[ele_1 ele_2… ele_n] [el_1 el_2 … el_n]	**+**	⇨	[ele_1+el_1 ele_2+el_2 … ele_n+ el_n]

Ejemplo:



❷  : Resta los elementos que tienen la misma posición de dos vectores que tienen la misma dimensión.

SINTAXIS:

[ele_1 ele_2… ele_n][el_1 el_2 … el_n] - ⇨ [ele_1- el_1 ele_2- el_2 … ele_n- el_n]

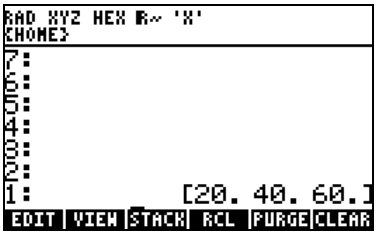
Ejemplo:

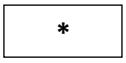
❸  : Multiplica un escalar con todos los elementos de u vector o viceversa.

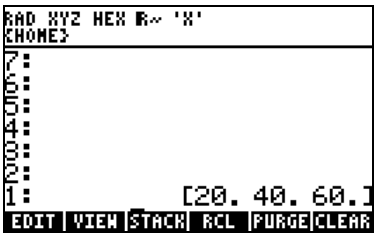
SINTAXIS:

①	x	[ele_1	ele_2	…	ele_n]		*	⇨	[x*ele_1	x*ele_2	… x*ele_n]
②	[ele_1	ele_2	…	ele_n]	x		*	⇨	[x*ele_1	x*ele_2	… x*ele_n]

Ejemplos:







❹ **/** : Divide todos los elementos de un vector con un escalar.

SINTAXIS:

[ele_1	ele_2	…	ele_n]	x	**/**	⇨	[ele_1/x	ele_2/x	…	ele_n/x]

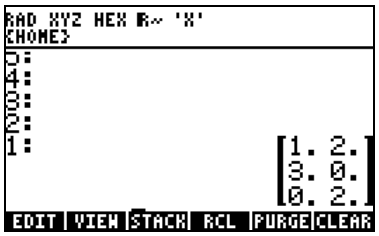
Ejemplo:

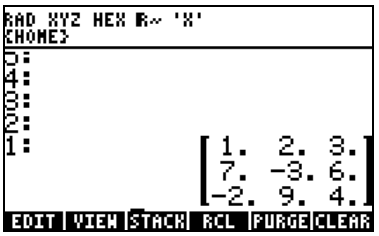
/

8 MATRICES

La matriz es un arreglo rectangular de elementos (objetos). Solo admite los objetos números y expresiones algebraicas y estos elementos están dispuestos en filas y columnas encerrados entre corchetes.

Ejemplos:





La primera matriz tiene 3 filas y 2 columnas por lo tanto es una matriz de orden de 3*2.
Los elementos son los objetos que están dentro de los corchetes.

La segunda matriz tiene 3 filas y 3 columnas, es una matriz de orden de 3*3.

El elemento de posición (1,1) de la primera matriz es: 1.
El elemento de posición (2,1) de la primera matriz es: 3.
El elemento de posición (2,1) de la segunda matriz es: 7.
El elemento de posición (2,3) de la segunda matriz es: 6.

CONSTRUCCION DE UNA MATRIZ

Se puede construir de varias formas, pero las más comunes son dos, por la línea de comandos y por el editor de matrices.

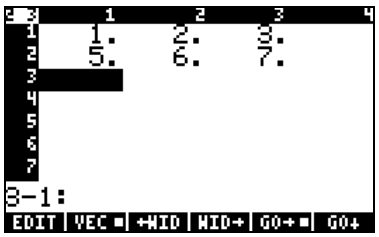
CONSTRUCCION POR LA LINEA DE COMANDOS: Se tiene que ingresar en filas utilizando corchetes para delimitar cada fila.

Ejemplo:

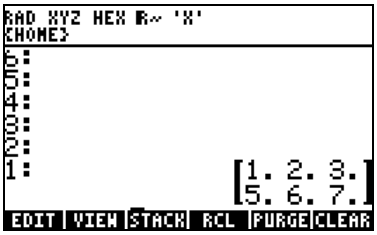
CONSTRUCCION POR EL EDITOR DE MATRICES: Se ingresar al editor presionando las teclas  y  .

Ejemplo:

Una vez abierto el editor de matrices se ingresa los datos en las casillas correspondientes.



ENTER



NOTA: Si se desea ingresar una matriz fila, se tiene que desactivar el menú **VEC** del editor de matrices.

CONSTRUCCION DE UNA MATRIZ UTILIZANDO COMANDOS

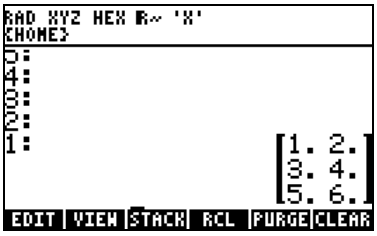
❶ **→ARRY** : Construye una matriz con los m*n elementos de la pila además requiere el orden de la matriz (m*n). En estos casos el orden de la matriz es de la forma {m n}.

SINTAXIS:

ele_1	ele_2	ele_3	…	ele_k	{m n}	**→ARRY**	⇨	matriz_de_m*n

Ejemplo 1:

→ARRY



En el nivel 1 indica el orden de la matriz {3. 2.}.

❷ **CON** : Construye una matriz de orden {m n}, los elementos son todos iguales, requiere el orden de la matriz {m n} y el elemento constante.

SINTAXIS:

{m n}	elemento	**CON**	⇨	matriz_de_m*n

Ejemplo:

CON

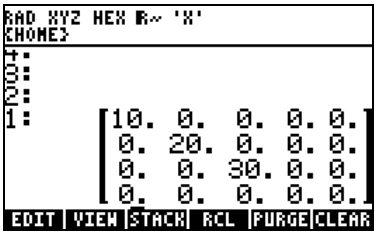
❸ **DIAG→** : Construye una matriz de orden {m n}, los elementos son todos iguales a 0, a excepción de su diagonal principal cuyos elementos serán los elementos de un vector o parte de los elementos del vector.

SINTAXIS:

vector_o_matriz fila	{m n} **DIAG→**	⇨	matriz_de_m*n

Ejemplo:

DIAG→



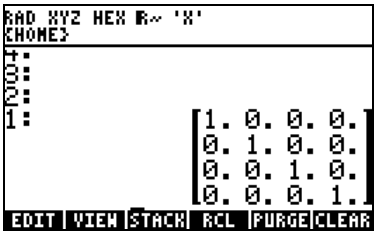
❹ **IDN** : Construye una matriz identidad de orden {m m}, requiere el orden m.

SINTAXIS:

m	**IDN**	⇨	matriz_de_m*m

Ejemplo:

IDN



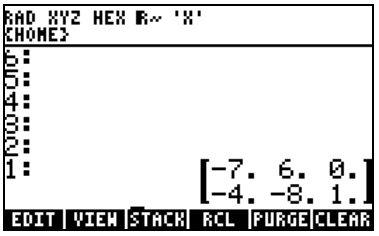
❺ **RANM** : Construye una matriz de orden {m n} cuyos elementos son números enteros aleatorios desde el -9 al 9, requiere el orden {m n}.

SINTAXIS:

{m n}	**RANM**	⇨	matriz_de_m*n

Ejemplo:

RANM



MANEJO DE MATRICES

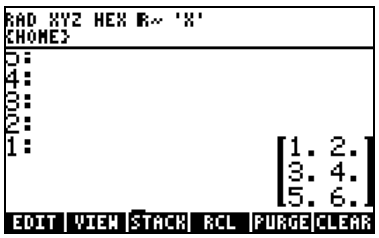
Se refiere a la manipulación de las matrices, sus elementos, columnas o filas.

❶ **ARRY→** : Descompone una matriz en sus elementos además devuelve el orden de la matriz. Es lo contrario del comando **→ARRY**.

SINTAXIS:

matriz_de_m*n	**ARRY→**	⇨	ele_1 ele_2 … ele_k {m n}

Ejemplo:



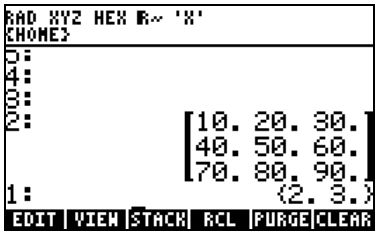
ARRY→

❷ **GET** : Obtiene un elemento de una matriz, requiere la posición del elemento ({i j}).

SINTAXIS:

matriz	{i j}	**GET**	⇨	elemento_i,j

Ejemplo:



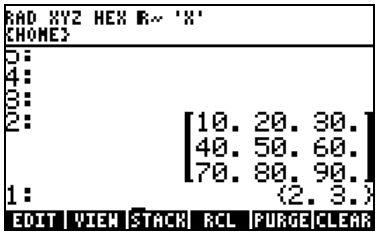
GET

❸ **GETI** : Devuelve la matriz original y la posición del siguiente elemento y extrae un elemento de la matriz, requiere la posición del elemento a extraer ({i j}).

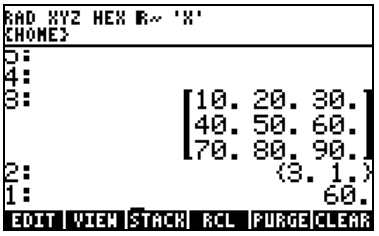
SINTAXIS:

matriz	{i j}	**GETI**	⇨	matriz	{k l}	elemento_i,j

Ejemplo:



GETI

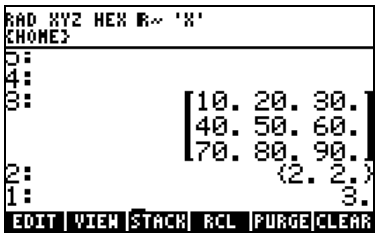


❹ **PUT** : Remplaza un elemento por otro elemento en una matriz, requiere la posición del elemento a reemplazar y el objeto por cual reemplazar.

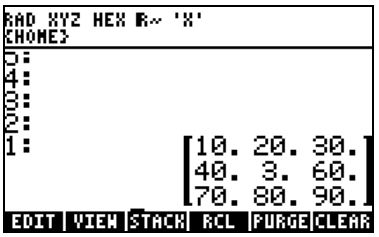
SINTAXIS:

matriz_1	{i j}	elemento	**PUT**	⇨	matriz_2

Ejemplo:



PUT

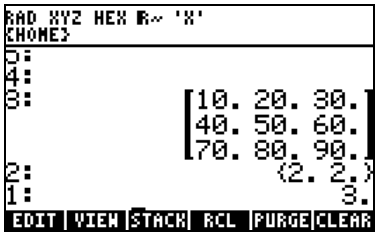


❺ **PUTI** : Remplaza un elemento de una matriz por otro elemento, además devuelve la posición del siguiente elemento, requiere la posición para reemplazar y el objeto por cual reemplazar.

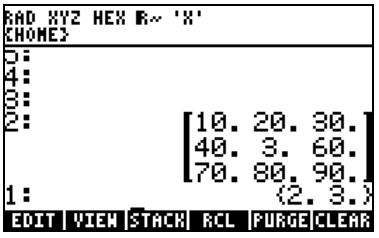
SINTAXIS:

matriz_1	{i j}	elemento	**PUTI**	⇨	matriz_2	{k l}

Ejemplo:



PUTI



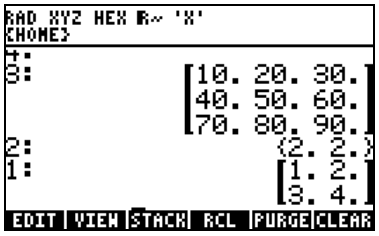
❻ **REPL** : Remplaza parte de una matriz por otra matriz, requiere la

posición desde donde se va a reemplazar y la matriz por cual se va reemplazar

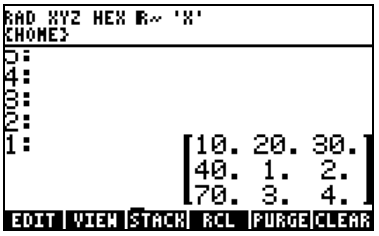
SINTAXIS:

matriz_1	{i j}	matriz_2	**REPL**	⇨	matriz_3

Ejemplo:



REPL

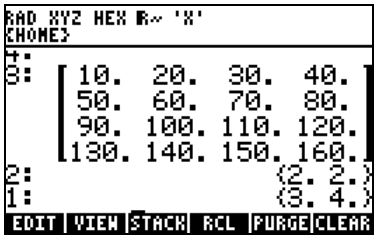


❼ **SUB** : Obtiene una sub matriz cuyos elementos son parte de los elementos de la matriz original, requiere la posición inicial y final de la matriz a extraer.

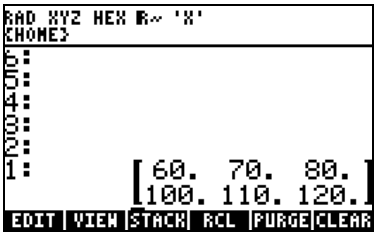
SINTAXIS:

matriz_1	{m n}	{o p}	**SUB**	⇨	matriz_2

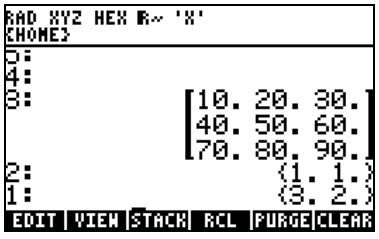
Ejemplo 1:



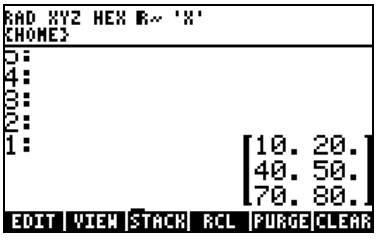
SUB



Ejemplo 2:



SUB

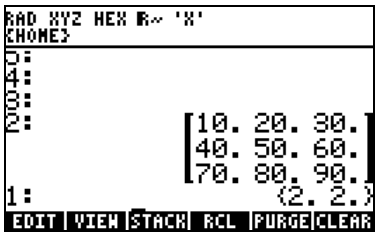


❽ **RDM** : Redimensiona una matriz al orden deseado ({m n}), requiere el nuevo orden que tendrá la matriz.

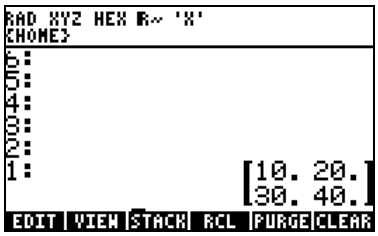
SINTAXIS:

matriz_1	{m n}	**RDM**	⇨	matriz_2

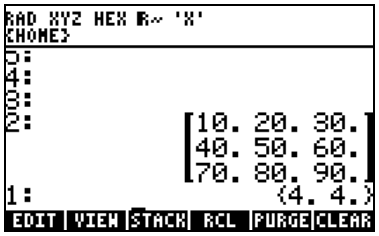
Ejemplo 1:



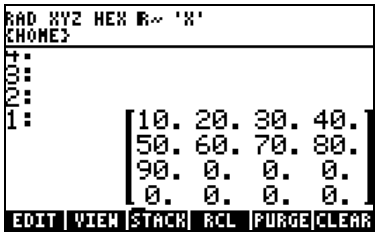
RDM



Ejemplo 2:



RDM

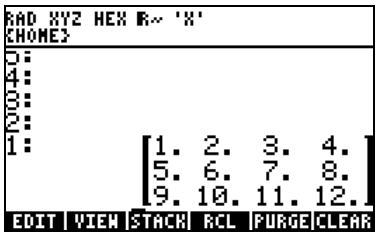


❾ **SIZE** : Obtiene la dimensión de una matriz o el orden de una matriz en una lista.

SINTAXIS:

matriz_de_m*n	**SIZE** ⇨	{m n}

Ejemplo:



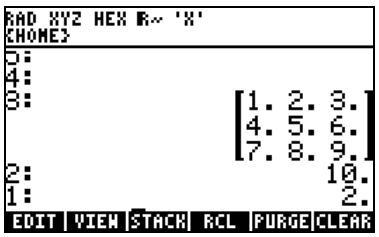
SIZE

❿ **RCI** : Multiplica una fila de una matriz por una constante, requiere la constante y el número de la fila a multiplicar.

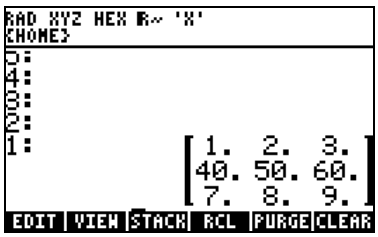
SINTAXIS:

matriz_1	constante	i	**RCI** ⇨	matriz_2

Ejemplo:



RCI



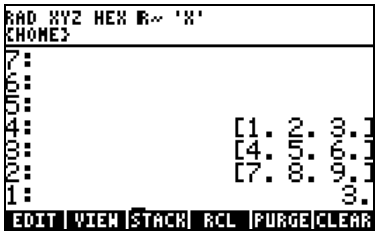
⓫ **ROW→** : Construye una matriz de m filas, requiere m filas y estas

filas deben ser vectores, también requiere el número de filas.

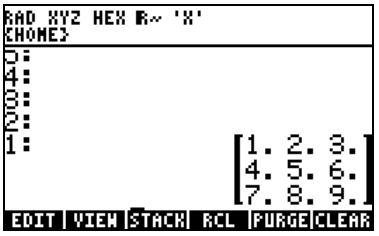
SINTAXIS:

vector_1	vector_2 …	vector_m	m	**ROW→**	⇨	matriz

Ejemplo:



ROW→

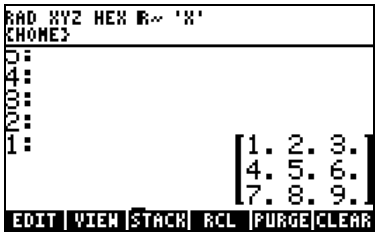


⑫ **→ROW** : Descompone una matriz en vectores filas y devuelve el número de filas. Es lo contrario de **ROW→**.

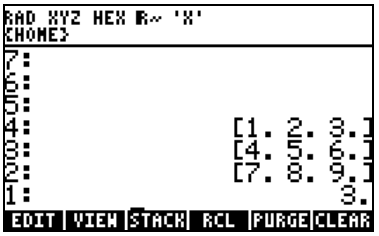
SINTAXIS:

matriz	**→ROW**	⇨	vector_1	vector_2 …	vector_m	m

Ejemplo:



→ROW

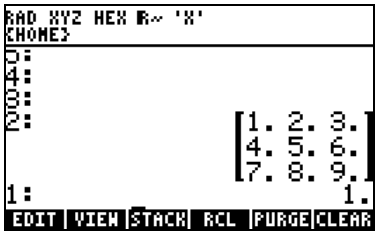


⑬ **ROW-** : Extrae una fila de una matriz, requiere la posición de la fila a extraer.

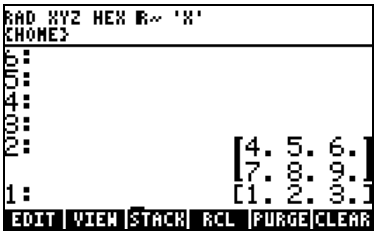
SINTAXIS:

matriz_1	i	**ROW-**	⇨	matriz_2	vector_i

Ejemplo:



ROW-

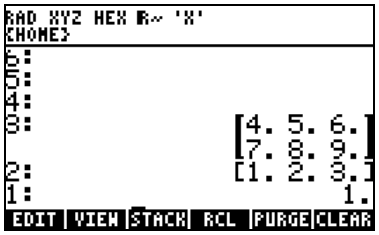


⑭ **ROW+** : Inserta una fila a una matriz, requiere el vector fila y la posición de la fila a insertar.

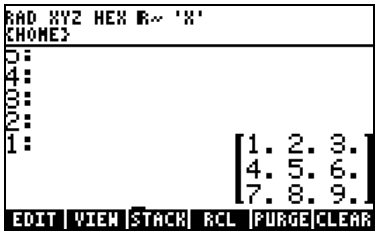
SINTAXIS:

matriz_1	vector	i	**ROW+** ⇨	matriz_2

Ejemplo:



ROW+

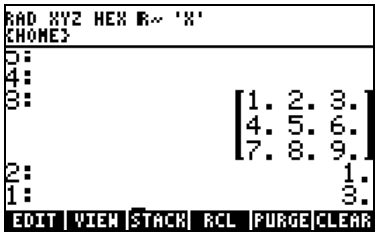


⓯ **RSWP** : Intercambia la posición de dos filas de una matriz, requiere las posiciones de las filas a intercambiar.

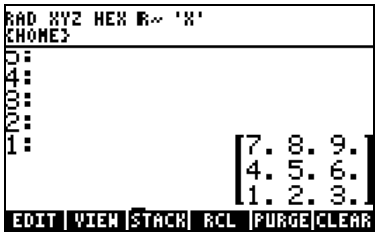
SINTAXIS:

matriz_1	k	l	**RSWP** ⇨	matriz_2

Ejemplo:



RSWP

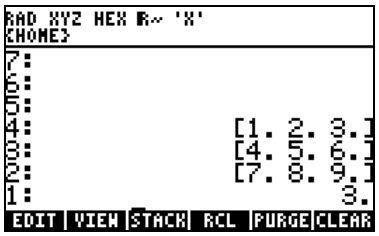


⓰ **COL→** : Construye una matriz de n columnas, requiere n columnas y estas columnas deben ser vectores, también el número de columnas.

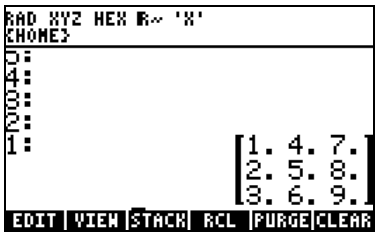
SINTAXIS:

vector_1	vector_2 …	vector_n	n	**COL→** ⇨	matriz

Ejemplo:



COL→

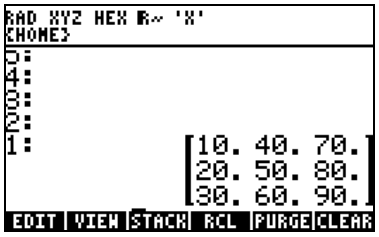


⓱ **→COL** : Descompone una matriz en vectores, estos vectores son las columnas de la matriz y devuelve el número columnas. Es lo contrario de **COL→**.

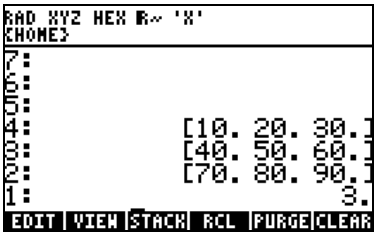
SINTAXIS:

matriz	→COL ⇨	vector_1	vector_2 ...	vector_n	n

Ejemplo:



→COL

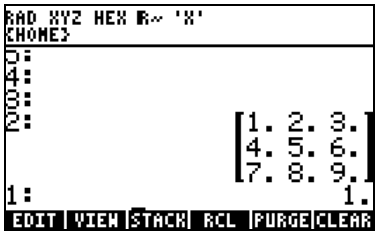


⓲ **COL-** : Extrae una columna de una matriz, requiere la posición de la columna a extraer.

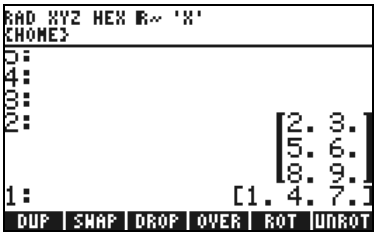
SINTAXIS:

matriz_1	j	**COL-** ⇨	matriz_2	vector_ j

Ejemplo:



COL-

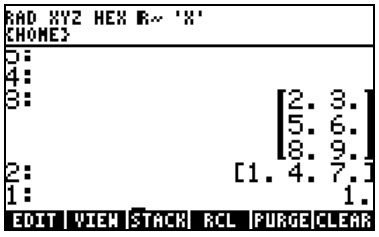


⓳ **COL+** : Inserta una columna a una matriz, requiere un vector y la posición de la columna a insertar.

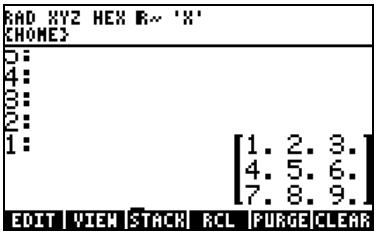
SINTAXIS:

matriz_1	vector	j	**COL+** ⇨	matriz_2

Ejemplo:



COL+

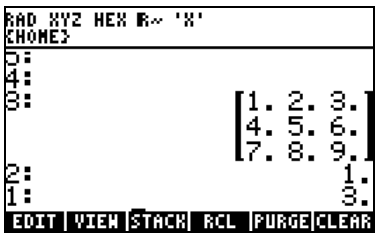


⓴ **CSWP** : Intercambia la posición de dos columnas de una matriz, requiere las posiciones de las columnas a intercambiar.

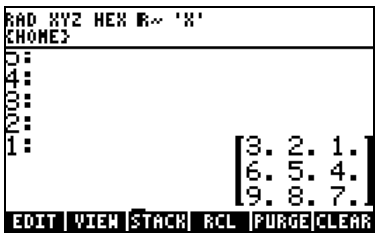
SINTAXIS:

matriz_1	o	p	**CSWP** ⇨	matriz_2

Ejemplo:



CSWP

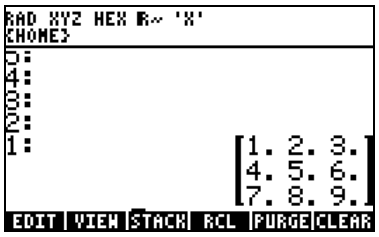


21 **TRAN** : Halla la transpuesta de una matriz.

SINTAXIS:

matriz **TRAN** ⇨ transpuesta (matriz)

Ejemplo:



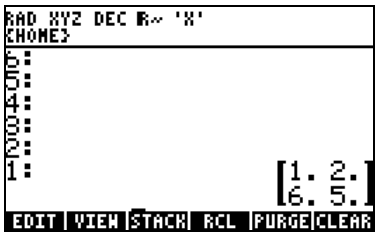
TRAN

22 **DET** : Obtiene la determinante de una matriz cuadrada.

SINTAXIS:

matriz **DET** ⇨ determinante (matriz)

Ejemplo:



DET

OPERACIONES Y FUNCIONES CON MATRICES

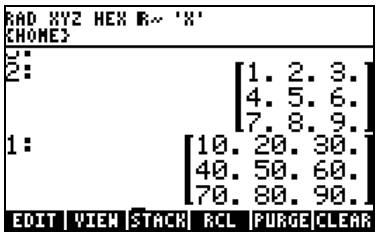
Las operaciones que se pueden realizar con matrices son la suma, resta y multiplicación, también algunas funciones.

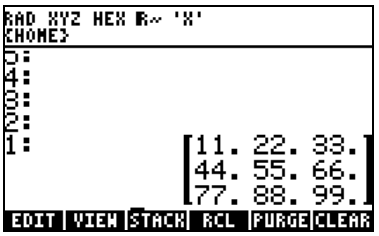
1 **+** : Halla una matriz, donde sus elementos son la suma de los elementos que ocupan la misma posición de dos matrices que tienen la misma dimensión.

SINTAXIS:

matriz_1_m*n	matriz_2_m*n	+	⇨	matriz_3_m*n

Ejemplo:



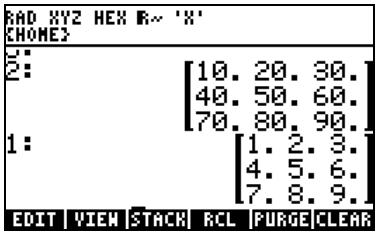


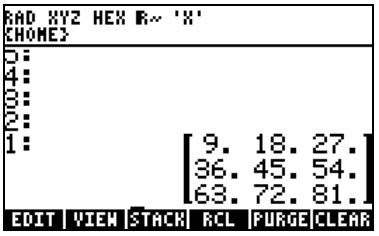
❷  : Halla una matriz, donde sus elementos son la resta de los elementos que ocupan la misma posición de dos matrices que tienen la misma dimensión.

SINTAXIS:

matriz_1_m*n	matriz_2_m*n	-	⇨	matriz_3_m*n

Ejemplo:



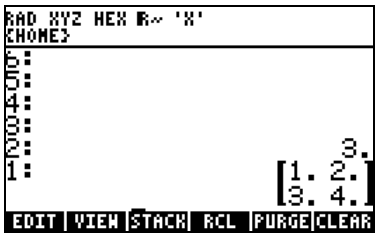


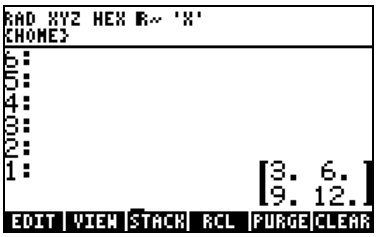
❸  : Existen varios tipos de multiplicación con una matriz, la multiplicación de un escalar por una matriz, matriz con vector (su dimensión debe ser igual que el número de columnas de la matriz) y matriz con matriz.

SINTAXIS:

①	x matriz_1_m*n	*	⇨	matriz_2_m*n
②	matriz_1_m*n [vector_1*n]	*	⇨	vector_1*m
③	matriz_1_m*n matriz_2_n*p	*	⇨	matriz_3_m*p

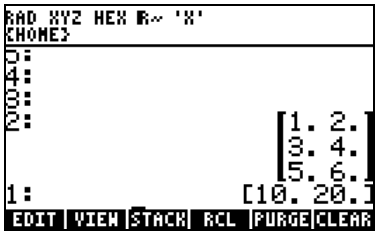
Ejemplo 1: multiplicación de un escalar por una matriz.





En este tipo de multiplicación cada elemento de la matriz resultante es igual al elemento original de la matriz multiplicado por el escalar.

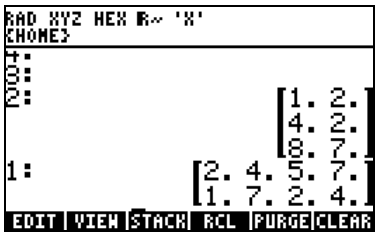
Ejemplo 2: multiplicación de una matriz por un vector.



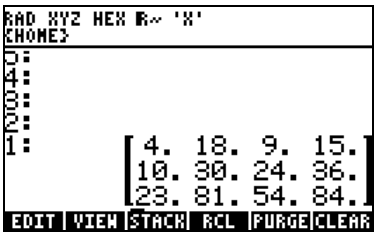
*

Multiplica los elementos de la primera fila de la matriz por los elementos del vector y luego suma todos los productos obtenidos, obteniendo 1*10+2*20 = 50, luego hace la misma operación con la segunda fila de la matriz y así sucesivamente con todas las filas de la matriz.

Ejemplo 3: multiplicación de matrices



*



Primeramente el número de columnas de la primera matriz debe ser igual al número de filas de la segunda matriz, resultando una matriz de orden igual al número de filas de la primera matriz por el número de columnas de la segunda matriz, cada elemento de la matriz resultante se calcula de la siguiente manera:

$$C_{ij} = \sum_{k=1}^{n} A_{ik} * B_{kj}$$

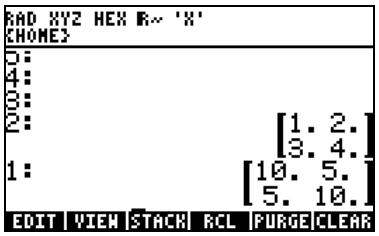
Donde : i = 1, 2, 3,…,m
j = 1, 2, 3,…,p
A_{ik} = elemento de la primera matriz
B_{kj} = elemento de la segunda matriz

❹ **HADAMARD** : Multiplica término a término los elementos de dos matrices de la misma dimensión.

SINTAXIS:

matriz_1_m*n	matriz_2_m*n	**HADAMARD** ⇨	matriz_3_m*n

Ejemplo:



HADAMARD

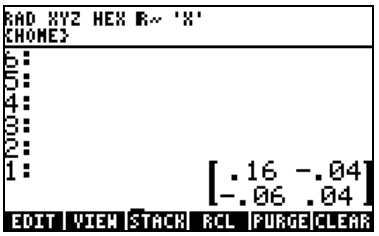
❺ ^ : Eleva una matriz cuadrada a un exponente, para realizar multiplicaciones sucesivas de una matriz obteniendo otra matriz de la misma dimensión. Si se lo eleva al exponente -1 se obtiene la matriz inversa.

SINTAXIS:

matriz	n	^	⇨	matriz * matriz * …* matriz (n factores)

Ejemplo:

^



9 CADENAS DE CARACTERES

Las cadenas son objetos que están delimitados por comillas (" "), los objetos que se encuentran dentro de las comillas son caracteres.

Ejemplos:

La primera cadena tiene 4 elementos y los elementos son los objetos que están dentro de las comillas.

En la primera cadena:
El primer elemento o carácter es: A
El segundo elemento o carácter es: 1
El tercer elemento o carácter es: B
El cuarto elemento o carácter es: 2

La dimensión o número de elementos de la primera cadena es: 4.

La segunda cadena es una cadena vacía sin elementos.
La tercera cadena no es una cadena vacía. Un espacio en blanco es un carácter.

COMPOSICION DE UNA CADENA O UN CARACTER

Es la obtención de las mismas a partir de un objeto.

❶ **→STR** : Convierte un objeto a una cadena de caracteres.

SINTAXIS:

objeto	**→STR**	⇨	"objeto"

Ejemplos:

→STR

→STR

En el segundo ejemplo el número no tiene punto decimal, en el tercer ejemplo lleva punto decimal. Al convertir el objeto en una cadena incluye todos los caracteres del objeto.

❷ **CHR** : Convierte el código de un carácter, en un carácter.

SINTAXIS:

número	**CHR**	⇨	"carácter"

Ejemplos:

CHR

CHR

CHR

CHR

Los códigos de los caracteres se encuentran en el manual de usuario, que viene en el CD.

OBTENCION DEL CODIGO DE UN CARACTER DE UNA CADENA

❶ **NUM** : Convierte un carácter o el primer carácter de una cadena, en su respectivo código de carácter.

SINTAXIS:

"carácter" **NUM** ⇨ número

Ejemplos:

NUM

NUM

NUM

NUM

En el último ejemplo se observa que el primer carácter es "@" y su código de carácter es 64.

DESCOMPOSICION DE CADENAS

Obtiene los objetos contenidos en una cadena o carácter.

❶ **OBJ→** : Obtiene los objetos contenidos en una cadena.

SINTAXIS:

"objetos" **OBJ→** ⇨ objetos

Ejemplos:

OBJ→

OBJ→

En el segundo ejemplo el comando obtuvo los objetos 10 5 + en este mismo orden y además los evaluó obteniendo la suma 15.

MANIPULACION DE CADENAS

Se refiere al cambio de elementos o caracteres ya sea por otro u otros, obtención de una parte de una cadena, etc.

❶ **SUB** : Obtiene una cadena cuyos elementos son parte de los elementos de la cadena original, requiere la posición inicial y final de la cadena a extraer.

SINTAXIS:

"cadena_1" i j **SUB** ⇨ "cadena_2"

Ejemplos:

SUB

SUB

❷ **REPL** : Reemplaza parte o todos los caracteres de una cadena por otra cadena, iniciando desde la posición indicada.

SINTAXIS:

"cadena_1" i "cadena_2" **REPL** ⇨ "cadena_3"

Ejemplos:

REPL

REPL

❸ **TAIL** : Elimina el primer carácter de una cadena.

SINTAXIS:

"cadena_1" **TAIL** ⇨ "cadena_2"

Ejemplo:

TAIL

❹ **HEAD** : Obtiene el primer carácter una cadena.

SINTAXIS:

"cadena" **HEAD** ⇨ "carácter"

Ejemplo:

HEAD

❺ **SREPL** : Reemplaza una sub cadena de una cadena, por otra cadena además devuelve 1 si la sub-cadena pertenece a la cadena inicial, en caso contrario devuelve la cadena original y el número 0.

SINTAXIS:

"cadena_1" "sub-cadena" cadena_2 **SREPL** ⇨ "cadena_3" 1 ó 0

Ejemplos:

SREPL

SREPL

❻ **POS** : Indica la posición de la primera ocurrencia de un carácter o sub cadena en una cadena.

SINTAXIS:

"cadena" "carácter" **POS** ⇨ i

Ejemplos:

POS

POS

En el segundo ejemplo se debe obtener la posición que ocupa el carácter "R" de la cadena "PROGRAMACION" obteniendo 2. Este comando ubica la posición del primer carácter que cumpla la condición.

❼ **SIZE** : Obtiene el número de caracteres de una cadena.

SINTAXIS:

"cadena" **SIZE** ⇨ n

Ejemplos:

SIZE

SIZE

En el segundo ejemplo hay 3 espacios, los cuales son caracteres entonces "1 2 3 " tiene 6 caracteres.

CONCATENACION DE CADENAS

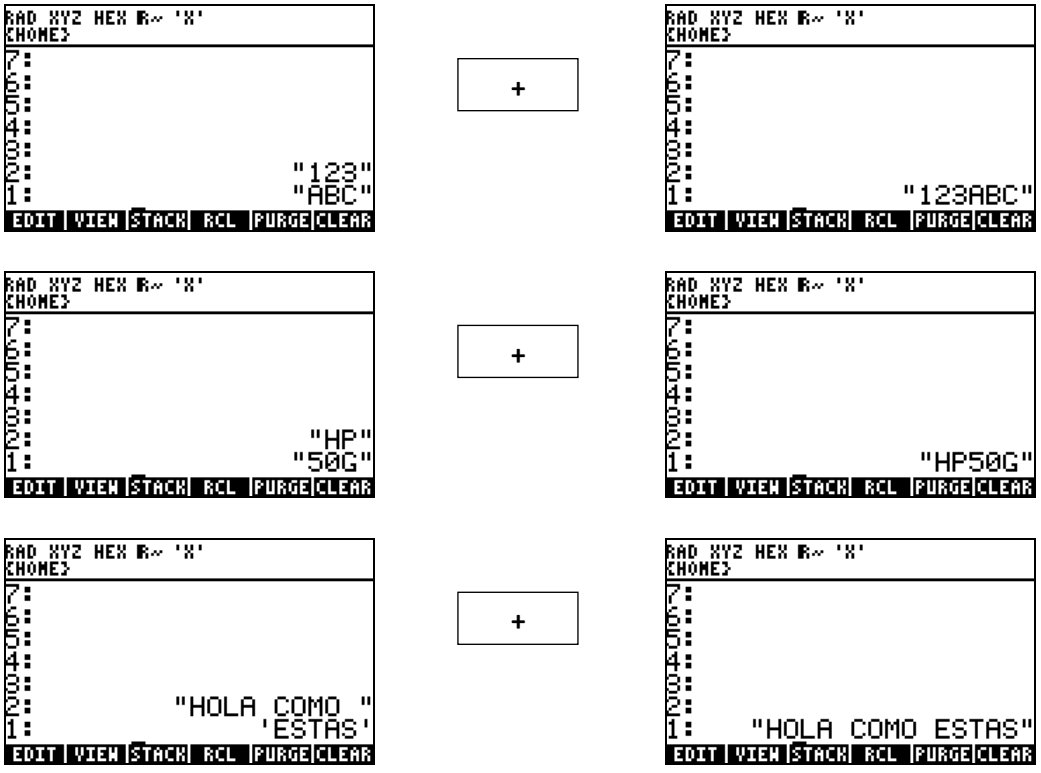
Se obtiene una cadena, en donde sus caracteres son los caracteres de dos cadenas.

❶ [+] : Concatena dos cadenas, una cadena con un objeto o viceversa.

SINTAXIS:

"cadena_1" "cadena_2" + ⇨ "cadena_1cadena_2"

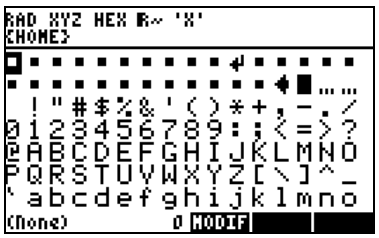
Ejemplos:



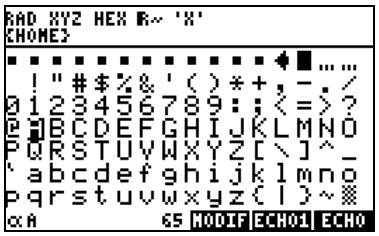
En el tercer ejemplo el objeto del primer nivel ('ESTAS') es una variable y no una cadena, el operador (+) también lo concatena, con la condición de que por lo menos un objeto sea una cadena.

ACCESO A LOS CARACTERES

Se puede ingresar a todos los caracteres que soporta la calculadora usando la aplicación CHARS, presionando la tecla [↱] seguido de la tecla [EVAL], aparece lo siguiente:



Al seleccionar un carácter se observa en el área de menús, que aparece el código del carácter, además la forma de cómo obtener el carácter utilizando el teclado.



Se observa que esta seleccionado el carácter "A" y su código de carácter es el número 65, utilizando el teclado se puede obtener presionando la tecla "**α**" (ALPHA) y luego "**A**" (F1 A).

10 CONFIGURACION DEL SISTEMA

Hace cambios en el sistema para modificar el formato numérico, angular e indicadores del sistema.

FORMATO NUMERICO

Estos comandos cambian el formato de visualización de los números.

❶ **STD** : Cambia al formato estándar (formato general), no requiere argumento.

❷ **FIX** : Cambia a un formato donde los números reales se visualizan con una cantidad exacta de dígitos decimales.

SINTAXIS:

	n **FIX** ⇨	

n : indica la cantidad de dígitos que se visualizará del número real después del punto decimal.

❸ **SCI** : Cambia al formato científico, requiere un número.

SINTAXIS:

	n **SCI** ⇨	

n : indica la cantidad de dígitos que se visualizará del número real después del punto decimal.

❹ **ENG** : Cambia al formato de ingeniería, requiere un número.

SINTAXIS:

	n **ENG** ⇨	

n : indica la cantidad de dígitos que se visualizará del número real después del punto decimal.

FORMATO ANGULAR Y DE COORDENADAS

Estos comandos no requieren de argumentos y cambian el formato angular o sistema de coordenadas.

❶ **DEG** : Cambia al formato angular sexagesimal.

❷ **RAD** : Cambia al formato angular de radianes.

❸ **GRAD** : Cambia al formato angular centesimal.

❹ **RECT** : Cambia las coordenadas a coordenadas rectangulares.

❺ **CYLIN** : Cambia las coordenadas a coordenadas cilíndricas.

❻ **SPHERE** : Cambia las coordenadas a coordenadas esféricas.

INIDICADORES DEL SISTEMA O BANDERAS

Estos comandos restauran u obtienen la configuración del sistema.

❶ **STOF** : Restaura la configuración de los indicadores del sistema, requiere una lista de números binario.

❷ **RCLF** : Obtiene la configuración de los indicadores del sistema actual.

Explicación:

En el primer gráfico de abajo se observa que aparece la hora y fecha (01:17 APR:16). Esta configuración es parte de toda la configuración del sistema. La configuración de todo el sistema se puede obtener usando el comando **RCLF**, este comando obtiene una lista de números binarios y en esta lista está codificada toda la configuración actual del sistema.

Ahora se cambiará la configuración del sistema por ejemplo borrando el reloj, quedando la pantalla de la calculadora como la primera figura de abajo. Si se quiere recuperar la anterior configuración, solo se debe escribir la lista de los números binarios obtenidas con el comando **RCLF**, luego se ejecuta el comando **STOF** obteniendo la configuración anterior.

❸ **SF** : Activa el indicador seleccionado del sistema.

SINTAXIS:

n	**SF**	⇨

n : indica el número del indicador del sistema. Estos números son todos números enteros negativos.

Ejemplo 1:

SF

El indicador del reloj es el número -40. En el primer gráfico del ejemplo no está activado el reloj pero al aplicarle el comando **SF** se activa el reloj.

Ejemplo 2:

SF

El indicador para designar la posición izquierda o derecha que tendrán los objetos en la pila es el número -74. Al activar el indicador el o los objetos se visualizarán al lado izquierdo.

❹ **CF** : Desactiva el indicador seleccionado del sistema.

SINTAXIS:

n	**CF**	⇨

n : indica el número del indicador del sistema. Estos números son todos números enteros negativos.

Ejemplo:

CF

El indicador del reloj es el número -40. En el primer gráfico del ejemplo está activado el reloj, al aplicarle el comando **CF** se desactiva el reloj.

❺ **FS?** : Verifica si un indicador está activado. Devuelve 1 si está activo y 0 si no lo está.

SINTAXIS:

n	**FS?**	⇨	0 ó 1

❻ **FC?** : Verifica si un indicador está desactivado. Devuelve 0 si está activo y 1 si no lo está.

SINTAXIS:

n	**FC?**	⇨	0 ó 1

❼ **FS?C** : Verifica si un indicador está activado, luego lo desactiva. devuelve 1 si está activo y 0 si no lo está.

SINTAXIS:

n	**FS?C**	⇨	0 ó 1

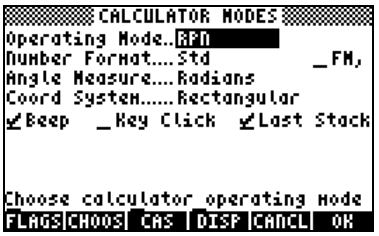
❽ **FC?C** : Verifica si un indicador está desactivado, luego lo desactiva. Devuelve 0 si está activo y 1 si no lo está.

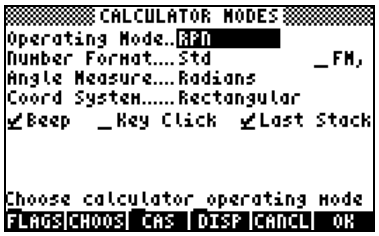
SINTAXIS:

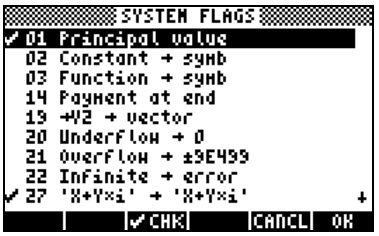
n	**FC?C**	⇨	0 ó 1

INGRESO A LOS INDICADORES DEL SISTEMA

Para ingresar a los indicadores se tiene que presionar las siguientes teclas: MODE y luego F1 A, observando las siguientes ventanas como en los siguientes gráficos:







Se posiciona en el indicador deseado. Para activarlo o desactivarlo presionando la tecla F3 C .

11 CONVERSION DE OBJETOS

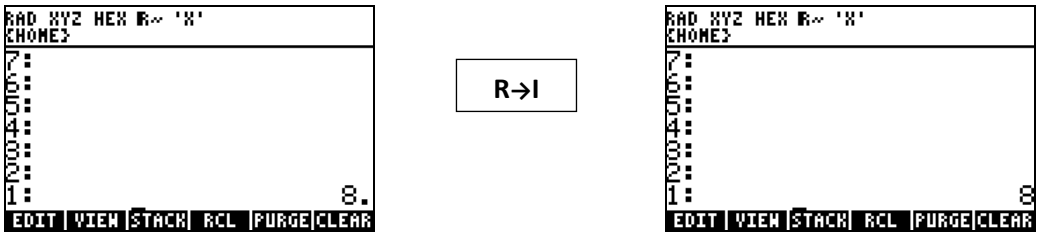
Se realizaran los cambios de un o unos objetos a otro u otros objetos.

❶ **R→I** : Convierte un número real que no tenga decimales a un número entero.

SINTAXIS:

x **R→I** ⇨ n

Ejemplo:

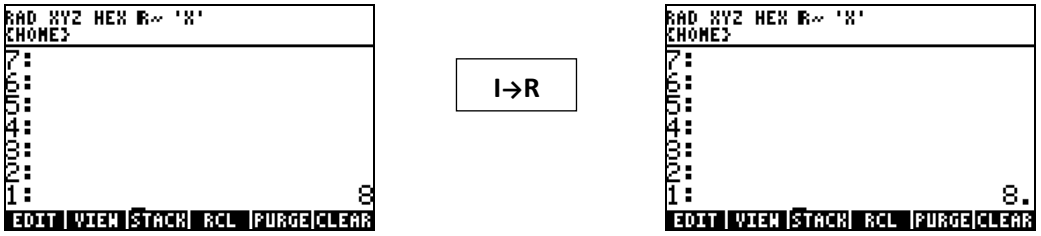


❷ **I→R** : Convierte un número entero a un número real.

SINTAXIS:

n **I→R** ⇨ x

Ejemplo:



❸ **C→PX** : Convierte una coordenada cartesiana o de usuario a coordenadas de pixel. Los valores obtenidos al ejecutar el comando, varían de acuerdo a la configuración del área gráfica.

SINTAXIS:

(x, y) **C→PX** ⇨ { #m #n }

Ejemplo:

C→PX

❹ **PX→C** : Convierte una coordenada de pixel a coordenadas cartesianas o de usuario. Los valores obtenidos al ejecutar el comando, varían de acuerdo a la configuración del área gráfica.

SINTAXIS:

{ #m #n } **PX→C** ⇨ (x, y)

Ejemplo:

PX→C

❺ **B→R** : Convierte un número entero binario a un número real positivo.

SINTAXIS:

#n **B→R** ⇨ n

Ejemplo:

B→R

❻ **R→B** : Convierte un número real positivo a un número entero binario.

SINTAXIS:

n **R→B** ⇨ #n

Ejemplo:

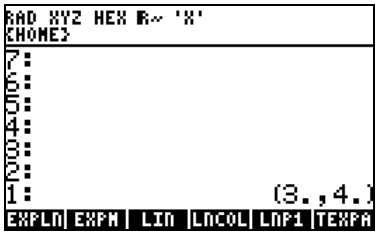
R→B

❼ **C→R** : Descompone un número complejo o coordenadas en sus componentes.

SINTAXIS:

(x, y) **C→R** ⇨ x y

Ejemplo:



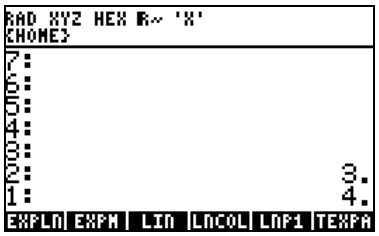
C→R

❽ **R→C** : Construye un número complejo o coordenadas de usuario con dos números reales.

SINTAXIS:

x y **R→C** ⇨ (x, y)

Ejemplo:



R→C

❾ **AXL** : Convierte un vector o matriz a una lista o una lista de lista respectivamente o viceversa.

SINTAXIS:

lista **AXL** ⇨ vector

Ejemplos:

AXL

AXL

❿ →Q : Convierte un argumento a su forma racional.

SINTAXIS:

x →Q ⇨ m/n

Ejemplo:

→Q

Se observa en el área de mensajes, que cambió los caracteres "**R~**" por "**R=**". El carácter "**~**", indica que los números se visualizarán en decimales o en forma aproximada y "**=**" en modo exacto.

12 OPERADORES RELACIONALES Y LOGICOS

Estos operadores son muy importantes y lo que hacen es comparar si una relación de números es correcta o hacen una prueba lógica simple.

OPERADORES RELACIONALES

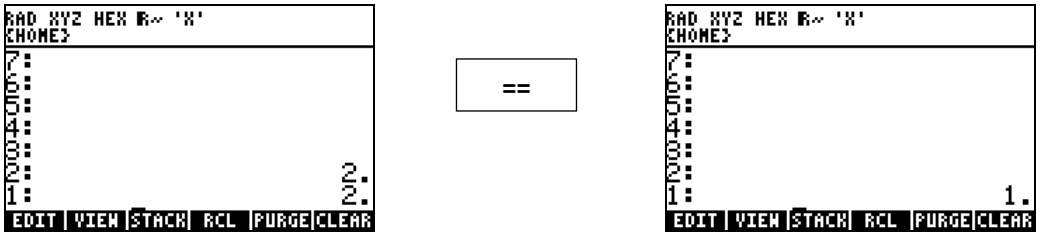
Lo que hacen estos operadores es comparar las posiciones relativas que tienen dos números, uno con respecto al otro y si cumple la comparación el operador devuelve el número 1 en caso contrario el número 0.

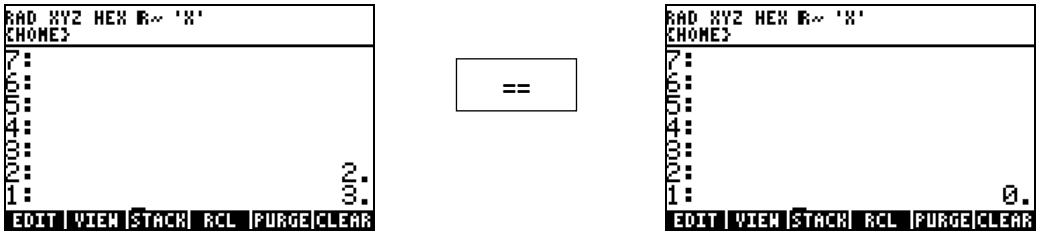
❶ `==` : Este operador compara la igualdad de dos números. Si son iguales devuelve el número 1 en caso contrario 0.

SINTAXIS:

x y == ⇨ 1 ó 0

Ejemplos:



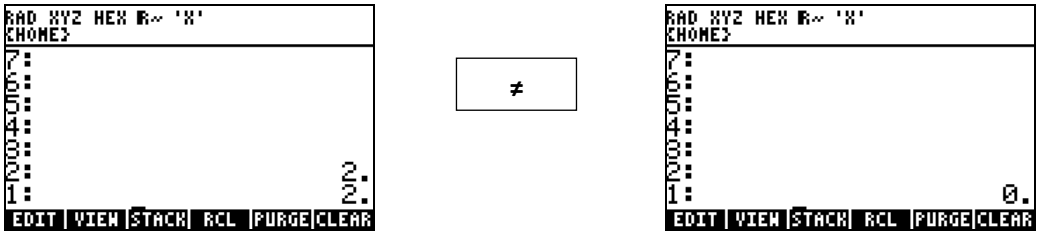


❷ `≠` : Este operador compara la desigualdad de dos números. Si son distintos devuelve el número 1 en caso contrario 0.

SINTAXIS:

x y ≠ ⇨ 1 ó 0

Ejemplos:



≠

❸ < : Este operador compara dos números. Si el primero es menor que el segundo devuelve el número 1 en caso contrario 0.

SINTAXIS:

x y < ⇨ 1 ó 0

Ejemplos:

<

<

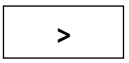
❹ > : Este operador compara dos números. Si el primero es mayor que el segundo devuelve el número 1 en caso contrario 0.

SINTAXIS:

x y > ⇨ 1 ó 0

Ejemplos:

>

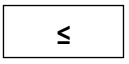


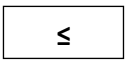
❺ ≤ : Este operador compara dos números. Si el primero es menor o igual que el segundo devuelve el número 1 en caso contrario 0.

SINTAXIS:

x y ≤ ⇨ 1 ó 0

Ejemplos:



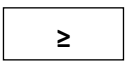


❻ ≥ : Este operador compara dos números. Si el primero es mayor o igual que el segundo devuelve el número 1. en caso contrario 0.

SINTAXIS:

x y ≥ ⇨ 1 ó 0

Ejemplos:



≥

OPERADORES LOGICOS

Lo que hacen estos operadores es comparar dos valores de verdad (verdadero y falso, en este caso los valores de verdad son números, el número 1 representa el valor verdadero y el 0 representa el valor falso), obteniendo un valor verdadero (1) en caso contrario falso (0).

❶ **AND** : Este operador obtienen el número 1 si los dos valores de verdad son 1 y en los demás casos el valor de verdad de 0.

SINTAXIS:

①	1	1	**AND**	⇨	1
②	1	0	**AND**	⇨	0
③	0	1	**AND**	⇨	0
④	0	0	**AND**	⇨	0

Ejemplos:

AND

AND

❷ **OR** : Este operador obtienen el número 1 si por lo menos uno de los valores de verdad es 1 y si no fuese el caso devuelve el valor de verdad de 0.

SINTAXIS:

①	1	1	**OR**	⇨	1
②	1	0	**OR**	⇨	1
③	0	1	**OR**	⇨	1
④	0	0	**OR**	⇨	0

Ejemplos:

OR

OR

❸ **XOR** : Este operador obtienen el número 1 si los dos valores de verdad son distintos, en los demás casos el valor de verdad de 0.

SINTAXIS:

①	1	1	**XOR**	⇨	0
②	1	0	**XOR**	⇨	1
③	0	1	**XOR**	⇨	1
④	0	0	**XOR**	⇨	0

Ejemplos:

XOR

XOR

❹ **SAME** : Este operador compara dos objetos, si son idénticos se obtiene el número 1 en caso contrario 0.

SINTAXIS:

objeto_1	objeto_2	**SAME** ⇨	1 ó 0

Ejemplos:

SAME

SAME

SAME

En el segundo ejemplo el número 4 es distinto del número 4. por tratarse de objetos diferentes. El primero es un objeto entero (número entero) y el segundo es un objeto real (número real). El operador **SAME** es muy estricto.

❺ **NOT** : Este operador requiere un argumento o valor de verdad. Con este operador se obtiene lo contrario del valor de verdad. Lo contrario de 1 es 0 y de 0 es 1.

SINTAXIS:

①	1	**NOT**	⇨	0
②	0	**NOT**	⇨	1

Ejemplos:

NOT


RAD XYZ HEX R↵ 'X'
{HOME}
7:
6:
5:
4:
3:
2:
1: 0.
EDIT VIEW STACK RCL PURGE CLEAR

NOT


RAD XYZ HEX R↵ 'X'
{HOME}
7:
6:
5:
4:
3:
2:
1: 1.
EDIT VIEW STACK RCL PURGE CLEAR

13 VARIABLES

Las variables son objetos en los cuales se pueden almacenar otros objetos. Estas variables pueden ser de uso momentáneo o pueden guardarse en la memoria.

VARIABLES GLOBALES

Estas variables son aquellas que pueden ser guardados permanentemente en la memoria hasta el momento que se los elimine. Las variables se pueden llamar solo si se encuentra en el directorio donde están guardadas dichas variables o en algún sub directorio. Para llamar las variables de cualquier directorio es necesario guardarlos en el directorio **HOME**.

❶ **STO** : Este comando crea una variable y además almacena cualquier objeto en ella, a excepción de operadores, comandos, funciones propias de la calculadora ,etc.

SINTAXIS:

objeto 'variable' **STO** ⇨

Ejemplo 1:

STO

En el primer gráfico del ejemplo, en el nivel 2 está el objeto (en este caso una función) y en el nivel 1 está el nombre de la variable donde se almacenará el objeto (función). En el primer gráfico, en el área de menús solo se ve el directorio **CASDIR** como en el gráfico de abajo.

Después de aplicar el comando **STO**, en el área de menús ya aparecen dos objetos como en el gráfico de abajo, el objeto que aumentó es la variable global de nombre **FUNC1** en donde está almacenado la función ($X^{2.}$+1.).

❷ **PURGE** : Este comando elimina una o varias variables, es necesario el nombre de la variable o nombres de las variables en una lista.

SINTAXIS:

①	'variable'	**PURGE**	⇨	
②	{ variable_1 variable_2 … variable_n }	**PURGE**	⇨	

Ejemplo :

PURGE

❸ **RCL** : Recupera el objeto guardado en una variable sin evaluarlo.

SINTAXIS:

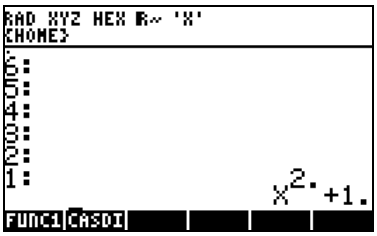
'variable'	**RCL**	⇨	objeto_almacenado_variable

LLAMAR EL OBJETO ALMACENADO EN UNA VARIABLE GLOBAL

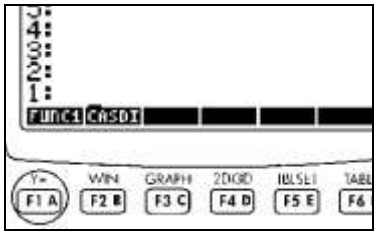
Para llamar u obtener el objeto almacenado en una variable global solo se escribe el nombre de la variable o presionar la tecla asociada al menú donde aparece la variable .

Ejemplo 1: se obtendrá el objeto almacenado en la variable de nombre **FUNC1** escribiendo su nombre:

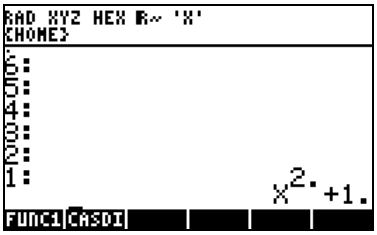
ENTER



Ejemplo 2: se llamará el objeto almacenado en la variable de nombre **FUNC1** presionando la tecla asociada al menú donde aparece el nombre de la variable.



F1 A



VARIABLES LOCALES

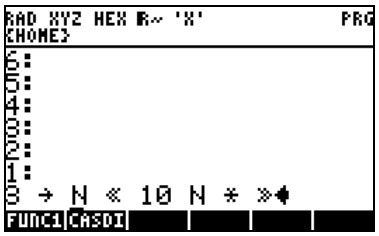
Estas variables solo pueden ser utilizadas después de escribir el nombre(s) de la(s) variable(s) local(es) precedido por los delimitadores de programa (**« »**), el contenido de estas variables solo pueden ser llamadas dentro de los limitadores de programa (**« »**). Estas variables se eliminan automáticamente cuando termina el programa.

❶ [→] : Este símbolo lo que hace es guardar temporalmente en la memoria de la calculadora un objeto en una variable local, la variable tendrá el nombre especificado

SINTAXIS:

objeto	→	variable	«programa»

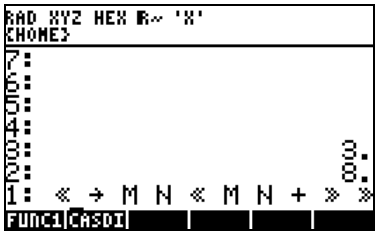
Ejemplo 1:



[ENTER]

En el primer gráfico se observa que el primer objeto (número 3), luego el símbolo →, luego el nombre de la variable local (N), hasta este momento lo que hizo el símbolo → es guardar el número 3 en la variable de nombre N, este valor almacenado solo puede ser usado dentro de los delimitadores de programa (**« »**), luego se ejecuta la expresión: 10 N *, pero en la variable de nombre N está almacenado el número 3, por lo que: 10 N * = 10 3 * = 10 * 3 = 30.

Ejemplo 2:



[EVAL N]

El objeto del nivel 1 es un programa porque está delimitado por los delimitadores de programa (**« »**). Lo que hace el programa primeramente es tomar los dos objetos que están en los niveles 3 y 2 y lo almacenan en dos variables con nombres M y N respectivamente, luego ejecuta la operación: M N + pero en los nombres de las variables M y N están almacenados los números 3 y 8 entonces: M N + es equivalente a: 3. 8. + = 11.

14 CARPETAS O DIRECTORIOS

Un directorio es un contenedor virtual, en la que se almacena un grupo de archivos de datos y otros directorios.

❶ **PURGE** : Borra una variable o un directorio vacío, requiere el nombre de la variable o directorio.

SINTAXIS:

'directorio' **PURGE** ⇨

Ejemplo:

PURGE

En el primer gráfico del ejemplo se observa el directorio DIR1, luego de aplicarle el comando el directorio ya no aparece.

❷ **CRDIR** : Este comando crea un nuevo directorio en el directorio actual, requiere el nombre del directorio.

SINTAXIS:

'directorio' **CRDIR** ⇨

Ejemplo:

CRDIR

Se observa en el segundo gráfico, en el área de menús se creó un nuevo directorio.

❸ **PGDIR** : Este comando borra un directorio contenido en el directorio actual, requiere el nombre del directorio.

SINTAXIS:

'nombre' **PGDIR** ⇨

❹ **PATH** : Devuelve en una lista la dirección del directorio actual.

SINTAXIS:

PATH ⇨ lista

Ejemplo:

PATH

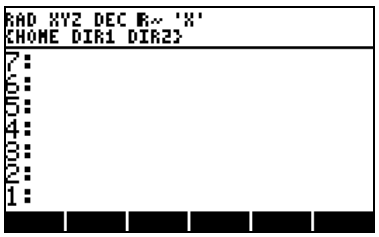
En el área estado la dirección del directorio actual coincide con el obtenido con el comando.

❺ **UPDIR** : Hace que el directorio primario del directorio actual, sea el nuevo directorio actual.

SINTAXIS:

UPDIR ⇨

Ejemplo:



UPDIR

En el primer gráfico la ruta del directorio actual es HOME/DIR1/DIR2, luego de aplicarle el comando sube un nivel al directorio primario, haciéndolo el actual (HOME/DIR1).

❻ **VARS** : Devuelve en una lista todos los objetos contenidos en el directorio actual.

SINTAXIS:

VARS ⇨ lista

Ejemplo:

VARS

Se observa en el área de menús, que efectivamente existe los objetos que se obtuvieron al aplicar el comando.

❼ **TVARS** : Devuelve en una lista todos los objetos de un solo tipo contenidos en el directorio actual, requiere el número del tipo de objeto.

SINTAXIS:

número_objeto **TVARS** ⇨ lista

Ejemplo:

TVARS

Al escribir el número 15, se refiere a los objetos del tipo DIRECTORIO.

❽ **ORDER** : Ordena los objetos contenidos en el directorio actual, requiere una lista con los nombres de los objetos en el orden que desea.

SINTAXIS:

lista **ORDER** ⇨

Ejemplo:

ORDER

En el primer gráfico en el área de menús, se observa el orden de los objetos y en el segundo el orden que se desea.

❾ **HOME** : Hace que el directorio **HOME** sea el actual.

SINTAXIS:

HOME ⇨

Ejemplo:

HOME

En el primer gráfico en el área de la ruta del directorio actual, se observa que está activo el directorio **DIR1**, luego en el segundo gráfico el directorio activo es el **HOME.**

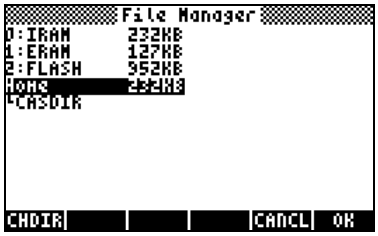
⑩ **FILER** : Abre el administrador de archivos (directorio, etc.).

SINTAXIS:

FILER ⇨

Ejemplo:

FILER



⑪ **CLVAR** : Elimina todos los objetos contenidos en el directorio actual. Si en el directorio actual existe otro directorio que no está vacía, el comando **CLVAR** no elimina ningún objeto.

SINTAXIS:

CLVAR ⇨

Ejemplo:

CLVAR

El directorio actual es el directorio DIR1. En este directorio antes de ejecutar el comando **CLVAR** había el directorio DIR2 y las variables VAR1 y VAR2, después ya no. El directorio DIR2 no contenía ningún objeto.

15 INSTRUCCIONES DE PROGRAMACION

Estos comandos son los más importantes, permiten tomar decisiones entre varias opciones en el programa (ramificaciones de programa) y realizar un proceso repetidas veces (procesos iterativos).

RAMIFICACIONES DEL PROGRAMA

Estas instrucciones evalúan uno o varios valores de verdad y en función a ello ejecuta una de entre dos o más posibles grupos de sentencias.

❶ **IF THEN END** : Esta es la más simple de las instrucciones, esta instrucción evalúa un determinado valor_verdad y en función a ello ejecuta o no unas sentencias.

SINTAXIS:

①		**IF**	valor_verdad
		THEN	sentencias_verdaderas
		END	
②	valor_verdad	**IF**	
		THEN	sentencias_verdaderas
		END	

valor_verdad : valor de verdad y pueden ser verdadero (1) o falso (0).

sentencias_verdaderas : sentencias que se ejecutarán si el valor de verdad es verdadero (1).

Ejemplo 1:

En el primer gráfico del ejemplo, el valor de verdad entre **IF** y **THEN** es el número uno (1), como el valor de verdad es 1 entonces ejecuta la expresión que está dentro de **THEN** y **END**, y esta expresión es "HOLA".

Ejemplo 2:

En el primer gráfico del ejemplo, el valor de verdad entre **IF** y **THEN** es el número cero (0), como el valor de verdad es 0 entonces **no** ejecuta la expresión que está dentro de **THEN** y **END**, y se va al final de **END,** como no hay nada después de **END** la pila queda vacía.

Ejemplo 3:

En el primer gráfico del ejemplo, el valor de verdad está antes de **IF** y es el número uno (1), como el valor de verdad es 1 entonces ejecuta la expresión que está dentro de **THEN** y **END**, y esta expresión es: 2 1 +, que es equivalente a: 2 + 1 = 3.

Ejemplo 4:

En el primer gráfico del ejemplo, el valor de verdad está antes de **IF** y es el número cero (0), como el valor de verdad es 0 entonces **no** ejecuta la expresión que está dentro de **THEN** y **END**, como no hay nada después de **END** la pila queda vacía.

Ejemplo 5: el ejemplo número 3 se puede escribir de la siguiente manera equivalente

Los símbolos **«** y **»** son los delimitadores de un programa y sirve para convertir uno o varios comandos, operadores, instrucciones, etc. en un solo objeto "PROGRAMA".

En el primer gráfico del ejemplo, el valor de verdad está antes de **IF** y es el número cero (1), como el valor de verdad es 1 entonces **se** ejecuta la expresión que está dentro de **THEN** y **END**, esta expresión es: 2. 1. + y es equivalente a: 2. + 1. = 3.

❷ **IF THEN ELSE END** : Estas instrucciones evalúa un determinado valor_verdad y en función a ello ejecuta una de entre dos posibles grupos de sentencias.

SINTAXIS:

①		**IF**	valor_verdad
		THEN	sentencias_verdaderas
		ELSE	sentencias_falsas
		END	
②	valor_verdad	**IF**	
		THEN	sentencias_verdaderas
		ELSE	sentencias_falsas
		END	

valor_verdad	:	valor de verdad y pueden ser verdadero (1) o falso (0).
sentencias_verdaderas	:	sentencias que se ejecutarán si el valor_verdad es verdadero (1).
sentencias_falsas	:	sentencias que se ejecutarán si el valor_verdad es falso (0).

Ejemplo 1:

Como se observa los números en los niveles 2 y 1 son 3. y 1. respectivamente, el valor de verdad entre **IF** y **THEN** es uno (1) por lo que se ejecutara la

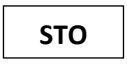
expresión que se encuentra entre **THEN** y **ELSE** y esta expresión es el operador + quedando la siguiente operación: 3. 1. +, el cual es equivalente a: 3. + 1. = 4.

Ejemplo 2:

Los números en los niveles 2 y 1 son 3. y 1. respectivamente, el valor de verdad entre **IF** y **THEN** es cero (0) por lo que se ejecutará la expresión que se encuentra entre **ELSE** y **END** y esta expresión es el operador - quedando la siguiente operación: 3. 1. -, el cual es equivalente a: 3. - 1. = 2.

Ejemplo 3: el ejemplo anterior se puede escribir de la siguiente manera equivalente:

Ejemplo 4: hacer un programa donde indique si una nota es aprobatoria o desaprobatoria (nota menor que 10.5).



Primeramente se guardará el programa que está en el nivel 2 en la variable de nombre PROG1, se observa en el área de menús del segundo gráfico, que el programa está almacenado.

Al ejecutar el programa PROG1 se obtiene "APROBADO", la calculadora ejecutó las siguientes instrucciones:

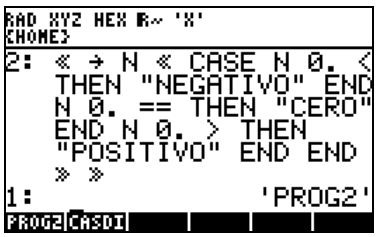
15.	→	N	⇨		:	almacena el valor **15.** en la variable local **N.**
N	10.5	<	⇨	0	:	compara si **N** es menor que **10.5**, esta relación es falsa por lo que se obtiene el número **0.**
IF	0		⇨		:	**IF** evalúa el valor de verdad, el valor de verdad es **0** Entonces se ejecutan las sentencias falsas entre **ELSE** y **END.**
ELSE "APROBADO" **END**			⇨		:	"APROBADO"

❸ **CASE THEN END… END** : Estas instrucción evalúa varias condiciones hasta que encuentre una que sea verdadera (1) y ejecuta las sentencias correspondientes.

SINTAXIS:

```
CASE
        cond_1 THEN   sentencias_1   END
        cond_2 THEN   sentencias_2   END
        .
        .
        .
        cond_n THEN   sentencias_n   END
        sentencias_por_defecto (opcional)
END
```

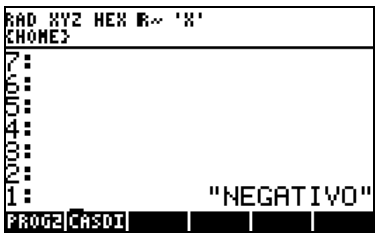
Ejemplo:



STO

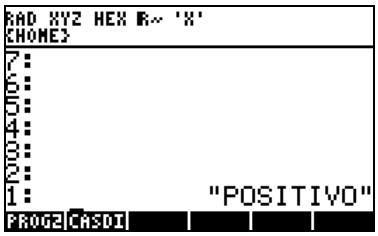
Ya guardado el programa con el nombre de PROG2 este se ejecutará con 3 números.

PROG2



PROG2

PROG2




```
«  →  N
     «  CASE
          N  0  <   THEN  "NEGATIVO"  END
          N  0  ==  THEN  "CERO"      END
          N  0  >   THEN  "POSITIVO"  END
END
     »
»
```

Al ejecutar el programa con el número 0 las sentencias que realiza el programa son las siguientes:

0.	→	N	⇨		:	almacena el valor **0.** en la variable local **N**
N	0.	<	⇨	0.	:	compara si **N** es menor que **0.** (falso pasa a la siguiente condición)
N	0.	==	⇨	1.	:	compara si **N** es igual que **0.** (verdadero pasa a evaluar sus respectivas sentencias).
THEN	"CERO"	**END**	⇨		:	"CERO"

❹ **IFERR THEN ELSE END** : Estas instrucciones evalúa un determinado proceso verificando si existe algún error y si existe alguno ejecuta un grupo de sentencias_erroneas en caso contrario ejecuta las sentencias_correctas

SINTAXIS:

```
①   IFERR  proceso
    THEN   sentencias_erroneas
    ELSE   sentencias_correctas
    END
②   IFERR  proceso
    THEN   sentencias_erroneas
    END
```

Ejemplo 1:

Entre **IFERR** y **THEN** solo existe un sumando 2 por lo que el operador **+** no podrá ejecutar la operación, entonces se produce un error por lo cual el programa pasa a la sentencias entre **THEN** y **END** en el cual está la cadena "FALTA UN SUMANDO**"**.

PROCESOS ITERATIVOS

Estas instrucciones permiten al programa ejecutar, un número exacto de veces las declaraciones contenidas entre ellas, también puede realizar varias veces las declaraciones contenidas entre ellas, hasta que cumpla una condición.

❶ **FOR** : Esta instrucción se utiliza para hacer una serie exacta de procesos iterativos y tiene dos sintaxis: el **FOR NEXT** y **FOR STEP**

SINTAXIS:

①	valor_inicial	valor_final	**FOR**	contador	procesos	**NEXT**	
②	valor_inicial	valor_final	**FOR**	contador	procesos	incremento	**ESTEP**

valor_inicial	:	es el primer valor que tomará el contador.
valor_final	:	es el último valor que tomará el contador.
contador	:	es una variable temporal el cual se incrementará desde un valor inicial (valor_inicial) hasta un valor final (valor_final). Para el caso de FOR NEXT este contador se incrementara de uno en uno y para el caso de FOR STEP se incrementara de acuerdo a un incremento designado.
procesos	:	son las instrucciones que se realizaran en el programa.
incremento	:	es el valor con el cual se irá incrementando el contador (solo para FOR STEP).

Ejemplo: hacer un programa que ponga los números del 1 hasta el 5 en la pila.

El número 1 es el valor inicial que tomará el contador i y el contador se irá incrementando de uno en uno hasta llegar al valor final 5, la primera i indica la variable del contador quiere decir que i irá incrementándose desde 1 hasta 5 de uno en uno, la segunda i es el proceso del programa lo que quiere decir que el programa solamente dará el valor del contador.

❷ **DO** : Esta instrucción se utiliza para hacer una serie de procesos iterativos hasta que se cumpla una condición. Cuando el valor_verdad es verdadero (1) termina las iteraciones.

SINTAXIS:

DO	procesos	**UNTIL**	valor_verdad	**END**

Ejemplo: hacer un programa que ponga los números, desde el número 1 hasta el 5 en la pila.

Primeramente el programa guarda en la variable global N el número 0. Luego procede con las iteraciones agregando a la variable N una unidad y este nuevo valor es guardado en la misma variable N y este nuevo valor que toma la variable N es enviado a la pila, este proceso se realizará hasta que la variable N sea igual que 5 y ahí termina las iteraciones.

❸ **WHILE** : Esta instrucción se utiliza para hacer una serie de procesos iterativos hasta que no se cumpla una condición, cuando el valor_verdad es falso (0) termina las iteraciones.

SINTAXIS:

WHILE	valor_verdad	**REPEAT**	proceso	**END**

Ejemplo: hacer un programa que ponga los números de 1 hasta 5 en la pila.

Primeramente el programa guarda en la variable global N el número 0. luego procede a iterar, verifica el valor de verdad y procede a realizar los procesos agregando a la variable global una unidad, luego reemplaza este nuevo valor en la variable global y este nuevo valor que toma la variable es enviado a la pila. Procede a iterar hasta que el valor_ verdad sea falso (0).

16 INTRODUCCION DE DATOS

En este capítulo se verá como ingresar datos usando unas plantillas propias del lenguaje RPL. Estas plantillas hacen que el ingreso de datos sea más fácil y así no cometer errores.

❶ **INPUT** : Esta secuencia de entrada es la más fácil y simple de usar, requiere los argumentos que se indican en la sintaxis.

SINTAXIS:

```
"texto_ayuda"
{ ":nombre_variable_1:
:nombre_variable_2:
:nombre_variable_3:
.
:nombre_variable_n:"  {número_fila  0}  modo }
INPUT  ⇨
":nombre_variable_1:dato_1
:nombre_variable_1:dato_2
.
.
:nombre_variable_n:dato_n"
```

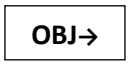
texto_ayuda	:	este es un texto que aparecerá en la parte superior del formulario, debe estar en cadena.
número_fila	:	indica la posición donde aparecerá el cursor al ingresar los datos.
modo	:	**V** verifica si existe un error al ingresar los datos. **ALG** activa el modo algebraico para introducir ecuaciones. **α** activa alpha.

Ejemplo: hacer un formulario que necesite como datos la base y altura de un rectángulo.

INPUT

Se observa que el número_fila es 1. por lo tanto el cursor aparece en la primera fila.

Una vez ingresado los datos se presiona la tecla ENTER, luego se obtiene una cadena de caracteres donde se encuentra los datos ingresados y para obtener los datos numéricos se tiene que utilizar el comando **OBJ→**, obteniendo los datos etiquetados.



❷ **INFORM** : Esta secuencia de entrada permite hacer formularios, para introducir datos de una manera más cómoda y práctica.

SINTAXIS:

```
"título_formulario"
{ { "nombre_variable_1 "  "texto_ayuda_1"      tipo_objeto }
          .
          .
          .
{ "nombre_variable_n "  "texto_ayuda_n"      tipo_objeto } }
{ número_columnas     número_espacios }
{ valores_reset }
{ valores_iniciales }
INFORM          ⇨
{ valores_variables_ingresados }
1 ó 0
```

título_formulario	:	es el título que aparecerá en la parte superior del formulario, debe ser una cadena.
nombre_variable_n	:	es la etiqueta que tendrá la variable, debe ser una cadena.
texto_ayuda_n	:	es un texto que aparece en la parte inferior del formulario, aquí se escribe la descripción de la variable correspondiente y debe ser una cadena.
tipo_objeto	:	aquí se indica el tipo de objeto que

		aceptará la variable correspondiente, se ingresa el número del objeto que se desea ingresar (ver cap. OBJETOS).
número_columnas	:	indica el número de columnas que tendrá el formulario.
número_espacios	:	es la separación que tendrá el nombre de la variable y el área de ingreso de datos de la variable respectiva.
valores_reset	:	son los valores que tomará las variables al presionar el menú reset.
valores_iniciales	:	son los valores con los que aparecen las variables al iniciar el formulario.
valores_variables_ingresados	:	estos son los valores ingresados en las variables. Aparecen en el orden declarado en el formulario.
1 ó 0	:	estos números indican lo siguiente: el número 1 indica que se ingresó correctamente los datos, el número 0 indica que se salió del formulario tocando la tecla **ON** o la tecla **F5**.

Ejemplo: hacer un formulario que necesite como datos la base y altura de un rectángulo.

INFORM

Se observa en el formulario que aparece con las etiquetas (nombre_variable_n) designadas, además aparece con los valores_iniciales en sus respectivas variables y con sus respectivo texto_ayuda_n y en la parte superior del formulario aparece el título_formulario.

F6 F

Los datos ingresados aparecen en una lista en el nivel 2 en el orden asignado al elaborar el formulario, en el nivel 1 aparece el número 1, esto quiere decir que se ingresó los datos de la manera correcta.

❸ **CHOOSE** : Crea un cuadro de selección, de opciones en la pantalla.

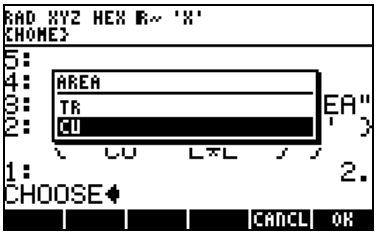
SINTAXIS:

```
                    "título_cuadro"
                    { { "texto_ayuda_1"     objeto_1 }
                            .
                      { "texto_ayuda_i"     objeto_i }
                            .
                      { "texto_ayuda_n"     objeto_n } }
                    posición
                    CHOOSE          ⇨
                    objeto_i
                    1 ó 0
```

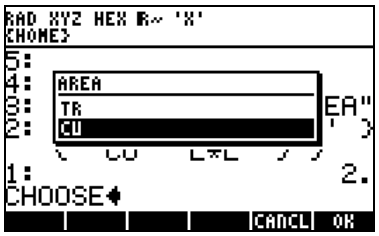
"título_ cuadro"	:	es el título que aparecerá en la parte superior del cuadro, debe ser una cadena.
"texto_ayuda_i"	:	es un texto que aparece en las regiones del cuadro, estas son las opciones que tendrá el cuadro de selección.
posición	:	es la posición inicial que estará seleccionado de todas las opciones, del cuadro de selección.
obeto_i	:	este es el objeto que devolverá o evaluará el programa.

Ejemplo: hacer un cuadro de selección que devuelva las fórmulas para hallar el área de un cuadrado o un triángulo.

CHOOSE



Se observa que esta seleccionado la segunda opción debido al número 2.



F6 F

El número 1 del nivel 1, indica que se continuó presionando la tecla **F6**, en el nivel 2 está el objeto correspondiente a la opción seleccionada.

❹ **WAIT** : Suspende la ejecución del programa por un tiempo especificado n. Si el tiempo de suspensión es de 0 este comando espera a que se oprima una tecla y devuelve como resultado el código de la tecla presionada.

SINTAXIS:

	n	**WAIT**	⇨	
0	**WAIT**	tecla	⇨	código_tecla

tecla : espera que se presione una tecla.
código_tecla : es el código de la tecla presionada (ver el capítulo de TECLADO).

Ejemplo:

WAIT

El comando está esperando que se presione una tecla. Se presionará la tecla F6.

F6 F

El código de la tecla F6 es16.1 en donde el primer 1 representa la fila de la tecla, 6 representa la columna de la tecla y el número 1 después del punto decimal representa el nivel o estrato de la tecla (ver el capítulo de TECLADO).

❺ **KEY** : Devuelve el código de tecla y el número 1 si se presiona una tecla en caso contrario devuelve 0. Este comando no se puede utilizar solo, se puede utilizar con las instrucciones de procesos iterativos.

SINTAXIS:

KEY	⇨	código_tecla	1
KEY	⇨	0	

17 SALIDA DE DATOS

En este capítulo se verá como mostrar datos usando unos comandos propios del lenguaje RPL, además de sonidos propios de la calculadora.

❶ **CLLCD** : Limpia la ventana de texto pero solo por un instante. Para que dure la limpieza se puede utilizar con el comando **WAIT** o con otros comandos como **MSGBOX**, **CHOOSE**, etc.

SINTAXIS:

CLLCD ⇨

❷ **MSGBOX** : Esta secuencia de salida es la más fácil de usar, muestra un mensaje en un cuadro en el centro de la ventana de texto, requiere una cadena.

SINTAXIS:

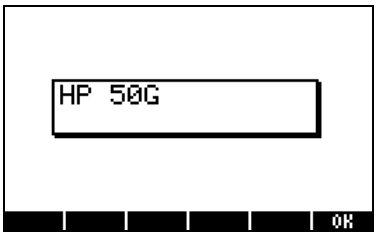
"cadena" **MSGBOX** ⇨

Ejemplo 1:

MSGBOX

Ejemplo 2: ahora se utilizará con el comando **CLLCD**.

ENTER



❸ **DISP** : Muestra un objeto en una línea de la pantalla pero solo por un instante, para que dure su visualización se puede usar con el comando **WAIT**.

SINTAXIS:

objeto número_linea **DISP** ⇨

número_ línea : indica la posición vertical donde se

visualizará el objeto, empezando desde arriba (1) hacia abajo.

Ejemplo 1:

En el segundo gráfico del ejemplo se observa que el texto está en la parte superior de la pantalla, además se utilizó con el comando WAIT, para que dure su visualización en la pantalla.

Ejemplo 2:

En el segundo gráfico del ejemplo se observa que el texto está en la tercera línea de la pantalla como se le indicó, además se utilizó con el comando WAIT, para que dure su visualización.

Ejemplo 3:

Al utilizarlo con el comando **CLLCD**, el fondo de la pantalla se limpió y luego se mostró el objeto en la pantalla.

❹ **FREEZE** : Congela la parte especificada de la pantalla para que no se actualice, hasta que se presione una tecla.

SINTAXIS:

	número_linea	**FREEZE**	⇨

número_línea : indica el área que se va a congelar

1 : área de estado
2 : pila y la línea de comandos
4 : área de menús

❺ **BEEP** : Produce un sonido con una frecuencia (Hz) y el tiempo (s) deseado.

SINTAXIS:

	frecuencia	tiempo	**BEEP** ⇨

Ejemplo:

BEEP

❻ **TEXT** : Cambia a la pantalla de modo texto, no requiere argumentos.

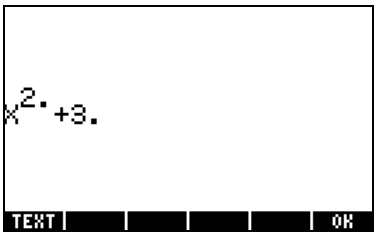
❼ **SCROLL** : Presenta un objeto en el formato más adecuado. Este comando es el más flexible para presentar objetos en la pantalla.

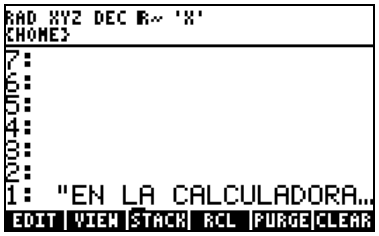
SINTAXIS:

	objeto	**SCROLL**	⇨

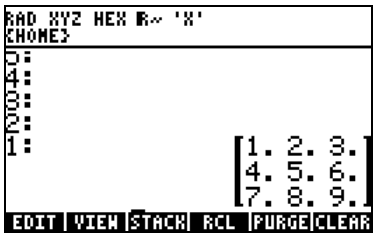
Ejemplos:

SCROLL

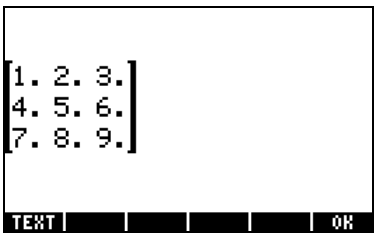




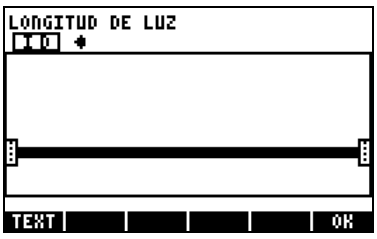
SCROLL



SCROLL



SCROLL



Si los objetos son más grandes solo se tiene que mover el objeto con las teclas del cursor.

❽ **DOERR** : Esta secuencia de salida es parecida a **MSGBOX**. Muestra un mensaje de error en un cuadro en el centro de la ventana de texto, junto con un sonido.

SINTAXIS:

"texto" **DOERR** ⇨

Ejemplo 1:

DOERR

18 ETIQUETAS

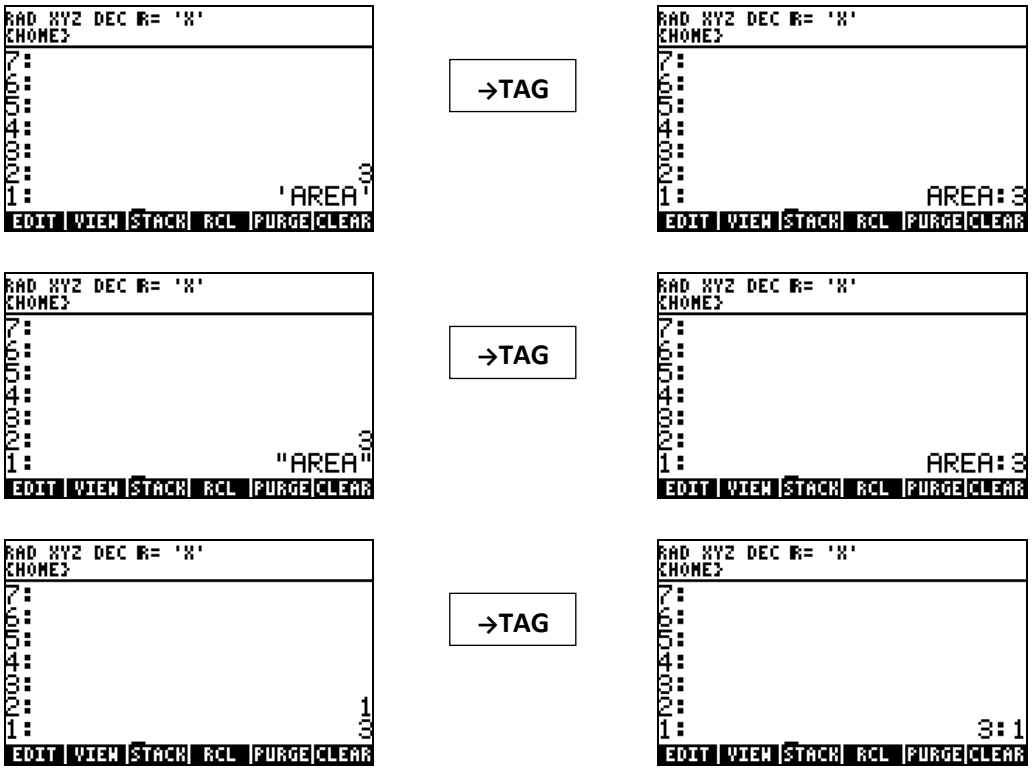
Las etiquetas o rótulos son objetos que sirven para identificar, clasificar objetos.
Se puede realizar operaciones con un objeto etiquetado y un objeto no etiquetado, obteniendo como resultado un objeto no etiquetado.

❶ **→TAG** : Etiqueta o rotula un objeto, requiere un objeto con el cual rotular.

SINTAXIS:

objeto objeto_etiqueta **→TAG** ⇨ :objeto_etiqueta: objeto

Ejemplos:



❷ **DTAG** : Elimina la etiqueta de un objeto.

SINTAXIS:

:objeto_etiqueta: objeto **DTAG** ⇨ objeto

Ejemplo:

19 MENUS

Es una serie de opciones que el usuario puede elegir para realizar determinadas tareas. Muestra las funciones, directorios, objetos, etc. correspondientes a las seis teclas superiores del teclado. Las teclas superiores son las teclas asociadas al menú.
El usuario puede crear un menú con la calculadora HP en donde los iconos del menú pueden ser textos y/o gráficos de 21 * 8 pixeles.

❶ **MENU** : Muestra el menú indicado, requiere el número del menú. Para saber el número de un menú se tiene que utilizar el comando **RCLMENU**.

SINTAXIS:

x **MENU** ⇨

Ejemplo:

Se observa que se activó el menú donde se encuentra los comandos para la manipulación de listas.

❷ **RCLMENU** : Devuelve el número del menú actual.

SINTAXIS:

RCLMENU ⇨ x

Ejemplos:

❸ **TMENU** : Muestra un menú elaborado por el usuario, requiere una lista de objetos, los cuáles serán visualizados en el área de menús o una lista de listas donde las listas están compuestos por dos objetos, el primero es el icono del menú y el segundo es el objeto que devolverá o evaluará en la pila.

SINTAXIS:

①	{obj_1 obj_2 … obj_n}	**TMENU**	⇨
②	{{"text_1" o graf 21*8_1 obje_1}… {"text_n" o graf 21*8_n obje_1}}	**TMENU**	⇨

Ejemplos:

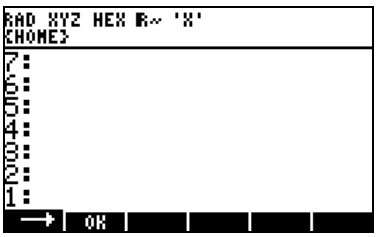
TMENU

Se observa que 10. es un número, SWAP es un comando y HOLA es una variable, y al presionar las tecla asociadas al menú se obtendrá el número 10. ejecutará el comando SWAP y devolverá la variable HOLA respectivamente.

TMENU

Al presionar la tecla correspondiente al menú MENU1 devolverá el número 100. y al tocar la tecla correspondiente al menú MENU2 devolverá el número 200.

TMENU



El primer menú es un gráfico y el segundo menú es una cadena. Al presionar el primer menú devuelve la variable SALIDA y al presiona el segundo menú devuelve la variable ACEPTAR. Estas variables pueden contener programas.

20 GRAFICOS

Los gráficos son objetos que se visualizan en su propia ventana. La calculadora hp tiene dos tipos de ventanas la ventana de texto (ventana común que se utiliza para hacer las operaciones comunes, etc.) y la ventana de gráficos en donde se puede realizar y visualizar todo tipo de gráficos. Existen dos tipos de comandos, los comandos que realizan directamente el gráfico en la ventana de gráficos y otros desde la ventana de texto.

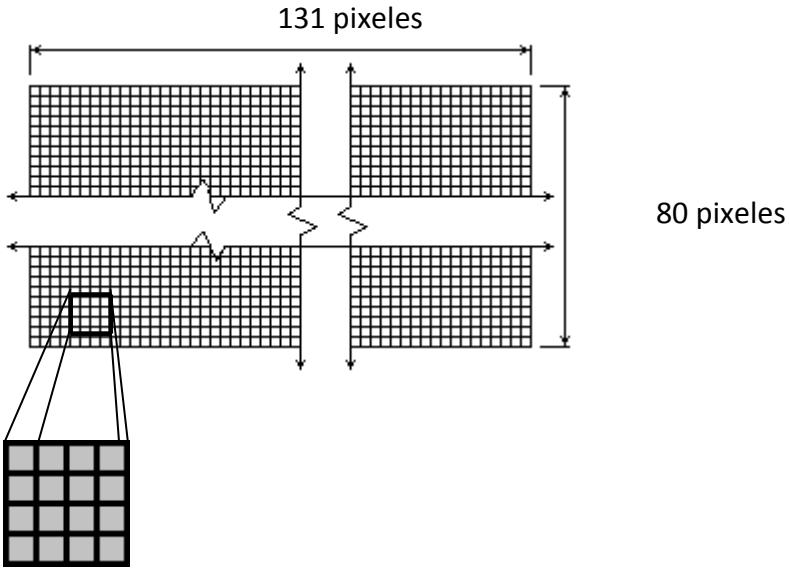
SISTEMAS DE COORDENADAS

La calculadora hp acepta dos tipos de sistemas de coordenadas para realizar gráficos, las coordenadas de los pixeles y coordenadas de usuario.

COORDENADAS DE LOS PIXELES

Son las coordenadas propias de la pantalla de la calculadora y están formados por pixeles físicos. Las coordenadas están dispuestas en la pantalla en forma ordenada en filas y columnas.

PIXELES: Los pixeles son unos cuadros pequeños en la pantalla de la calculadora, las cuales están dispuestas en filas y columnas. Las calculadoras HP 49G+ y HP 50 G tienen pantallas con filas de 131 pixeles y columnas de 80 pixeles.



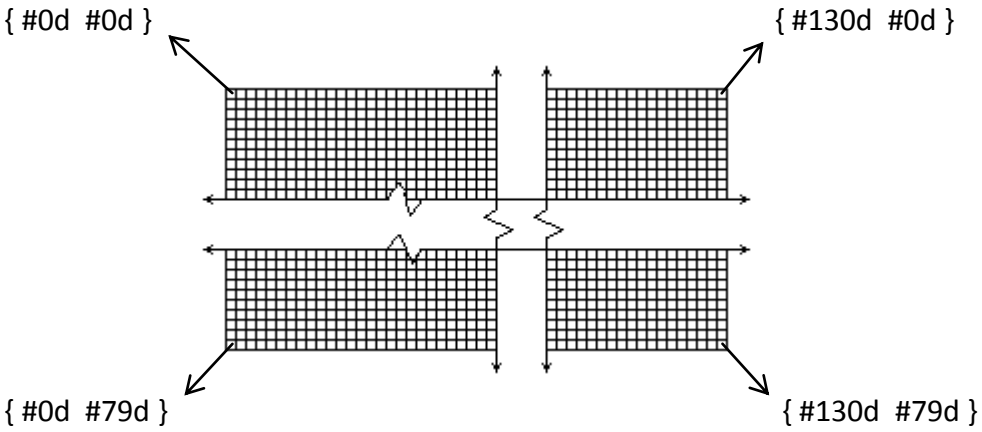
Cada cuadro de color claro es un pixel. Observen detenidamente la pantalla de su calculadora y podrán observar los cuadros pequeños.

REPRESENTACION: Las coordenadas de los pixeles se representan con un par de números enteros binarios contenidos en una lista, de la siguiente forma:

```
{ # md  # nd }
```

md : es un número entero binario y representa la posición del pixel en la dirección de las filas de **izquierda a derecha.** Inicia desde el número entero binario **# 0d** hasta el **# 130d**.

nd : es un número entero binario y representa la posición del pixel en la dirección de las columnas de **arriba hacia abajo.** Inicia desde el número entero binario **# 0d** hasta el **#79d.**



NUMERO ENTERO BINARIO:

Un número entero binario es aquel número entero mayor o igual que cero (0), antecedido por el símbolo # y precedido por la letra **h**, **d**, **o** o **b**, estas letras precedentes representan la base en que se encuentra los números enteros binarios y tiene la forma:

 # nd

BASE DEL NUMERO ENTERO BINARIO:

La base de un número entero binario está representado por las letras: **h**, **d**, **o** y **b**

h **:** base hexadecimal
d **:** base decimal
o : base octal
b : base binaria

INGRESAR UN NUMERO ENTERO BINARIO A LA PILA:

Solamente se escribe el símbolo # seguido del número entero mayor o igual que cero (0) y la letra que le precede indicando la base deseada. Si no se escribe la base en el número entero binario, este se escribe automáticamente

dependiendo de la base actual activada.

Ejemplo 1:

ENTER

Se observa en el segundo gráfico que la letra que le precede es la letra **d**, esto indica que está seleccionada la base decimal. Es recomendable seleccionar la base que se desea antes de ingresar los números enteros binarios.

Ejemplo 2:

ENTER

CAMBIO DE BASE DE NUMEROS ENTEROS BINARIOS: Para ello se utilizan los siguientes comandos:

1. **HEX** : Selecciona la base de los números enteros binarios a la base hexadecimal.
2. **DEC** : Selecciona la base de los números enteros binarios a la base decimal.
3. **OCT** : Selecciona la base de los números enteros binarios a la base octal.
4. **BIN** : Selecciona la base de los números enteros binarios a la base binaria.

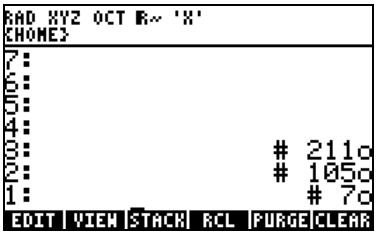
Ejemplo 1:

DEC

Se observar en el área de mensajes del primer gráfico un texto HEX, este texto indica la base seleccionada y en el segundo gráfico el texto cambio a DEC (decimal).

Ejemplo 2:

OCT



Se observa en el área de mensajes del primer gráfico un texto HEX, este texto indica la base seleccionada y en el segundo gráfico el texto cambió a OCT (octal). También los números enteros binarios cambiaron de base.

OPERACIONES CON NUMEROS ENTEROS BINARIOS: Se puede realizar algunas operaciones y siempre devolverá un número entero y no un decimal.

❶ [+] : Realiza la suma de dos números enteros binarios o un entero binario con un número o viceversa.

SINTAXIS:

m # n + ⇨ # (m+n)

Ejemplo:

[+]

❷ [-] : Resta dos números enteros binarios o un entero binario con un número o viceversa.

SINTAXIS:

m # n - ⇨ # (m – n)

Ejemplo:

[-]

❸ [*] : Multiplica dos enteros binarios o un número con un entero binario o viceversa

SINTAXIS:

# m	# n	*	⇨	# (m * n)

Ejemplo:

[*]

❹ [/] : Divide dos enteros binarios o un número con un entero binario o viceversa.

SINTAXIS:

# m	# n	/	⇨	# (m / n)

Ejemplo:

[/]

COORDENADAS DE USUARIO

Son las coordenadas cartesianas y se representan por pares ordenados. Estas coordenadas varían en la ventana de gráficos de acuerdo a los parámetros establecidos con los comandos: **XRNG** y **YRNG**

(x , y)

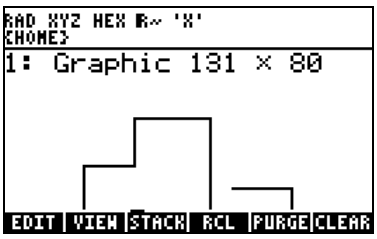
x : es un número real que representa el valor en el eje X.
y : es un número real que representa el valor en el eje Y.

PICT

Es una variable especial en el que se guarda el gráfico actual de la ventana de gráficos. Si se modifica la variable **PICT** en la ventana de texto usando algún comando y luego al visualizar la ventana de gráficos se observará que el gráfico también se modificó.
Al escribir la variable **PICT** en la pila solo se observa la variable y no su contenido. Para ver el gráfico contenido en **PICT** se utilizar el comando **RCL**.

Ejemplo:

RCL



Al aplicar el comando **RCL** se observa que muestra lo que tiene la variable **PICT** en la pila en el nivel 1.

VENTANA DE GRAFICOS

La ventana de gráficos es una ventana diferente a la ventana de texto. En la ventana de gráficos se puede dibujar cualquier gráfico, utilizando los comandos que aparecen en el menú de la misma ventana.
Para visualizar la ventana de gráficos se presiona la tecla obteniendo el siguiente entorno.

Aparece el cursor en el medio de la pantalla como una cruz (+).
Los menús que aparecen en la ventana de gráficos son tres **(X, Y)**, **EDIT** y **CANCL**

(X, Y) Este menú sirve para saber la ubicación del cursor en la pantalla, mostrando las coordenadas cartesianas o de usuario

EDIT Este menú accede a todas las herramientas para poder dibujar gráficos, enviar gráficos a la pila, etc.

CANCL Este menú sirve para salir de la ventana de gráficos a la ventana de texto.

DIBUJAR UN GRAFICO UTILIZANDO LA VENTANA DE GRAFICOS (EDITOR DE GRAFICOS)

Ya en la ventana de gráficos se ingresa al menú **EDIT** observando los siguientes menús:

Se observa los comandos en el área de menús.

DESCRIPCION DE LOS COMANDOS

DOT+ : activa el pixel seleccionado por el cursor
DOT- : desactiva el pixel seleccionado por el cursor
LINE : dibuja una línea desde un punto marcado con el cursor (pulse el menú **MARK** para marcar un punto) hasta otro punto seleccionado con el cursor
TLINE : es parecido a **LINE**, pero activa o desactiva los pixeles de la pantalla
BOX : dibuja un rectángulo desde un punto marcado hasta otro punto seleccionado con el cursor
CIRCL : dibuja una circunferencia, primeramente se marca el origen de la circunferencia luego se indica el radio con la ubicación del cursor
MARK : marca un punto
+/- : invierte la visualización del cursor cuando el cursor pasa por un objeto
LABEL : muestra solamente las etiquetas de los ejes
DEL : borra la parte indicada por un rectángulo, formado por un punto marcado y por la ubicación del cursor
ERASE : borra todos los objetos contenidos en la pantalla
MENU : oculta el menú hasta que se presione las teclas (**+**), (**-**), o cualquier tecla asociada al menú
SUB : **envía a la pila** un gráfico seleccionado por el rectángulo formado por dos puntos, uno marcado y el otro por la ubicación del cursor
REPL : pega el gráfico que se encuentra en la pila en la ubicación del cursor
PICT→ : copia el gráfico en la pila
X,Y→ : copia la coordenada donde se encuentra el cursor en la pila
PICT : muestra el menú inicial de la ventana de gráficos

MANIPULACION DE LA VENTANA DE GRAFICOS DESDE LA PILA

Estos comandos manipulan la ventana de gráficos desde la pila, a excepción de los últimos tres que comprueban si está activado un pixel o convierten coordenadas de usuario a coordenadas de pixel o viceversa.

❶ **ERASE** : Borra todos los objetos que se encuentran en la ventana de gráficos.

SINTAXIS:

	ERASE ⇨

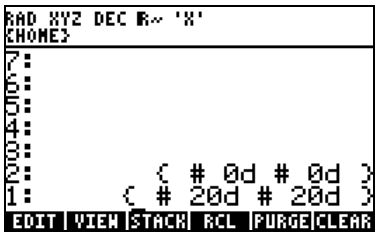
❷ **LINE** : Traza una línea entre dos coordenadas en la ventana de gráficos. Las coordenadas pueden ser de pixeles o de usuario. Es conveniente usar las coordenadas de pixeles.

SINTAXIS:

①	{ # m1 # n1 }	{ # m2 # n2 }	**LINE** ⇨
②	(x1 , y1)	(x2 , y2)	**LINE** ⇨

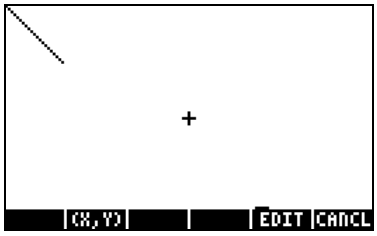
Ejemplo: dibujar una línea, primeramente se seleccione la base decimal.

Se borra la ventana de gráficos usando el comando **ERASE**.



LINE

Para visualizar la ventana de gráficos se presiona la tecla .



❸ **ARC** : Dibuja un arco en la ventana de gráficos, requiere la coordenada del centro en coordenadas de pixeles o de usuario, radio del arco en números binarios o reales, el ángulo inicial y el ángulo final. Se debe tener en cuenta el modo de ángulo seleccionado, por ejemplo si seleccionó en modo de radianes los ángulos tienen que ingresarse en radianes.

SINTAXIS:

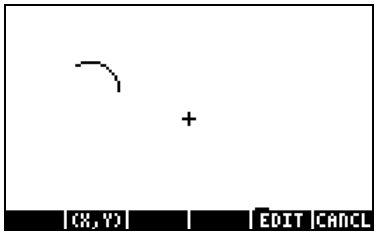
①	{ # m1 # n1 }	# r	Θ1	Θ2	**ARC**	⇨
②	(x1 , y1)	r	Θ1	Θ2	**ARC**	⇨

Ejemplo 1: dibujar un arco, primeramente seleccionar la base decimal y el ángulo en sexagesimales.

Se borra la ventana de gráficos usando el comando **ERASE**.

ARC

Para visualizar la ventana de gráficos se presiona la tecla .



Ejemplo 2: dibujar un arco con los mismos datos del ejemplo anterior, pero con los ángulos en orden inverso.

Se borra la ventana de gráficos usando el comando **ERASE**.

ARC

Para visualizar la ventana de gráficos se presiona la tecla .

El orden de los ángulos importa.

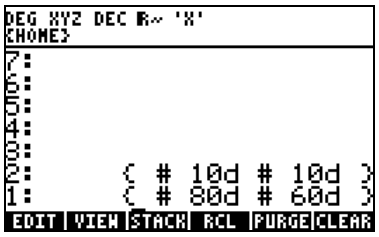
❹ **BOX** : Dibuja un rectángulo con sus lados paralelos a los ejes en la ventana de gráficos, requiere las coordenadas de dos vértices opuestos. Las coordenadas pueden ser de pixeles o de usuario.

SINTAXIS:

①	{ # m1 # n1 }	{ # m2 # n2 }	**BOX** ⇨
②	(x1 , y1)	(x2 , y2)	**BOX** ⇨

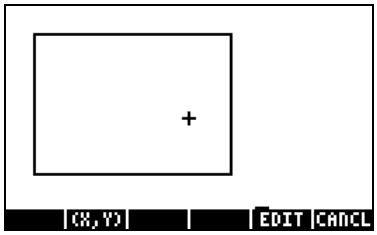
Ejemplo:

Se borra la ventana de gráficos usando el comando **ERASE**.



BOX

Para visualizar la ventana de gráficos se presiona la tecla .



❺ **TLINE** : Traza una línea entre dos coordenadas, cambiando el estado de los pixeles en la ventana de, las coordenadas pueden ser de pixeles o de usuario. Conviene usar las coordenadas de pixel.

SINTAXIS:

①	{ # m1 # n1 }	{ # m2 # n2 }	**TLINE** ⇨
②	(x1 , y1)	(x2 , y2)	**TLINE** ⇨

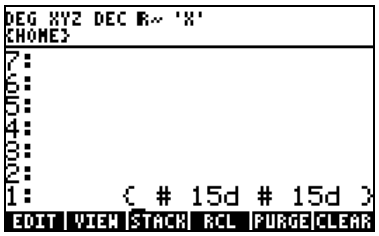
❻ **PIXON** : Activa un pixel en la venta de gráficos, requiere una coordenada en pixel o de usuario.

SINTAXIS:

①	{ # m # n }	**PIXON** ⇨
②	(x , y)	**PIXON** ⇨

Ejemplo:

Se borra la ventana de gráficos usando el comando **ERASE**.



PIXON

Para visualizar la ventana de gráficos se presiona la tecla

❼ **PIXOF** : Desactiva un pixel en la venta de gráficos, requiere una coordenada ya sea en pixel o de usuario.

SINTAXIS:

①	{ # m # n }	**PIXOF**	⇨
②	(x , y)	**PIXOF**	⇨

❽ **PIX?** : Comprueba si un pixel en la venta de gráficos está activado, requiere una coordenada ya sea en pixel o de usuario. Devuelve 1 si el pixel está activado en caso contrario 0.

SINTAXIS:

①	{ # m # n }	**PIX?**	⇨	1 ó 0
②	(x , y)	**PIX?**	⇨	1 ó 0

❾ **C→PX** : Convierte una coordenada cartesiana o de usuario a coordenada de pixel.

SINTAXIS:

(x , y)	**C→PX**	⇨	{ # m # n }

❿ **PX→ C** : Convierte una coordenada de pixel a coordenada cartesiana o de usuario.

SINTAXIS:

{ # m # n }	**PX→C**	⇨	(x , y)

MANIPULACION DE OBJETOS GRAFICOS

Estos comandos modifican o crean objetos gráficos **en la pila** y no en la ventana de gráficos, para poder modificar los objetos contenidos en la ventana de gráficos con estos comandos, es necesario modificar la variable **PICT** (contiene todos los objetos dibujados en la ventana de gráficos), como si fuera el objeto gráfico. Al modificar **PICT** se modifica instantáneamente la ventana de gráficos.

❶ **→GROB** : Convierte un objeto a gráfico, requiere un objeto y el modo de conversión **n**.

SINTAXIS:

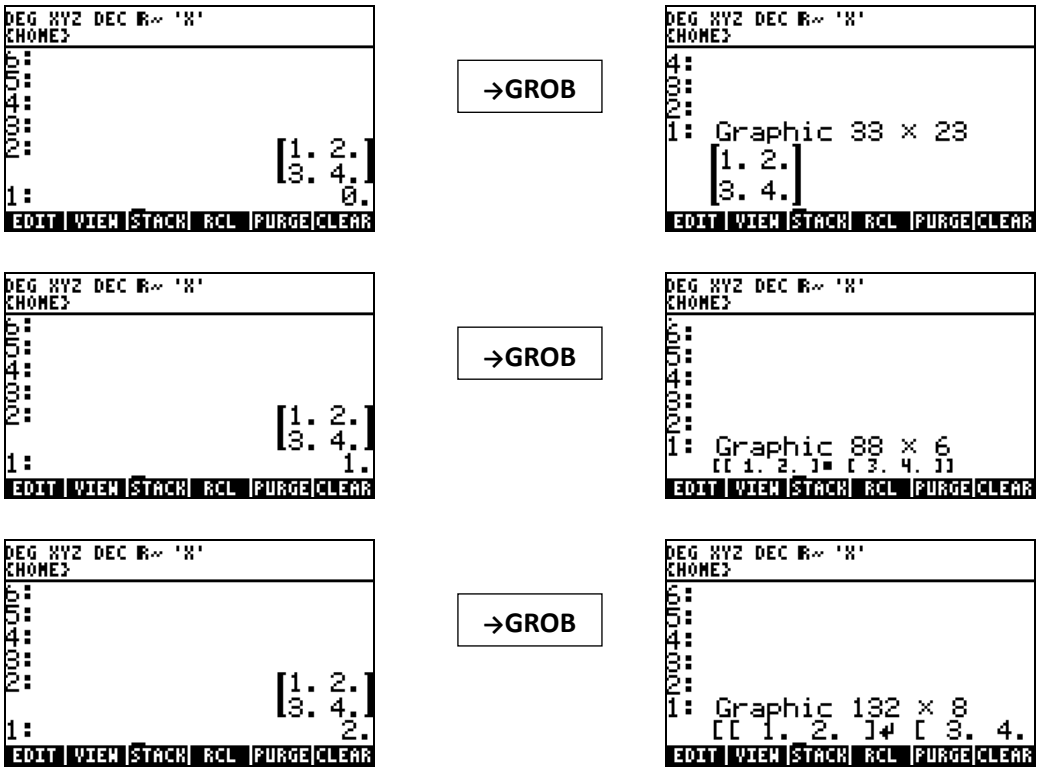
objeto	n	**→GROB**	⇨	gráfico

si n = 0 : convierte el objeto a gráfico casi exactamente como se observa en la pila.

si n = 1 : convierte el objeto a gráfico con los caracteres más pequeños, además el objeto se lo observa en su forma equivalente como si se escribiría en una sola línea .

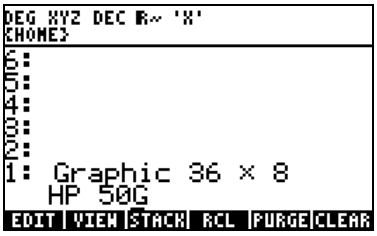
si n = 2 : convierte el objeto a gráfico con los caracteres más grandes, además el objeto se lo observa en su forma equivalente como si se escribiría en una sola línea.

Ejemplo 1: convertir una matriz utilizando los tres modos de conversión.



Ejemplo 2: convertir un texto (cadena) utilizando los tres modos de conversión.

→GROB



→GROB

→GROB

❷ **BLANK** : Construye un gráfico en blanco, requiere el ancho y el alto en números enteros binarios.

SINTAXIS:

# m	# n	**BLANK**	⇨	gráfico_en_blanco_de_m*n

Ejemplo:

BLANK

El gráfico está de color blanco. Para visualizarlo se puede usar el comando **NEG**, que invierte los pixeles de un gráfico contenido en la pila.

NEG

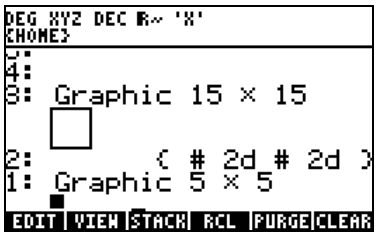
❸ **GOR** : Sobrepone un gráfico sobre otro gráfico, requiere la

ubicación y el gráfico a sobreponer.

SINTAXIS:

(gráfico_1 o **PICT**)	{ # m # n }	gráfico_2	**GOR**	⇨	gráfico_3

Ejemplo:



GOR

❹ **GXOR** : Sobrepone un gráfico invertido sobre otro gráfico, requiere la ubicación y el gráfico a sobreponer.

SINTAXIS:

(gráfico_1 o **PICT**)	{#m #n }	gráfico_2	**GXOR**	⇨	(gráfico_3 o **PICT**)

Ejemplo:

GXOR

❺ **SUB** : Extrae una porción rectangular de un gráfico, requiere las coordenadas de los vértices de una diagonal del rectángulo a extraer.

SINTAXIS:

(gráfico_1 o **PICT**)	{ # m1 # n1 }	{ # m2 # n2 }	**SUB**	⇨	gráfico_2

Ejemplo:

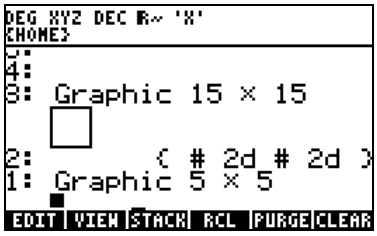
SUB

❻ **REPL** : Reemplaza una región de un gráfico por otro gráfico, requiere la posición desde donde se va a reemplazar y el gráfico por el cual se va a reemplazar. Es idéntico **GOR**.

SINTAXIS:

(gráfico_1 o **PICT**)	{ # m1 # n1 }	gráfico_2	**REPL** ⇨	(gráfico_3 o **PICT**)

Ejemplo:



REPL

❼ **SIZE** : Halla las dimensiones de un gráfico que se encuentra en la pila en números enteros binarios. Devuelve el ancho y el alto.

SINTAXIS:

gráfico_de_m*n	**SIZE** ⇨	# m	# n

Ejemplo:

SIZE

❽ **→LCD** : Muestra un gráfico que se encuentra en la pila, en la pantalla de texto en la esquina superior izquierda, solo por un instante. Para que dure la visualización del gráfico se puede utilizarlo con el comando **WAIT**.

SINTAXIS:

gráfico	**→LCD** ⇨

Ejemplo:

ENTER

❾ **LCD→** : Captura la ventana de texto y lo convierte en un gráfico.

SINTAXIS:

LCD→	⇨	gráfico

Ejemplo:

LCD→

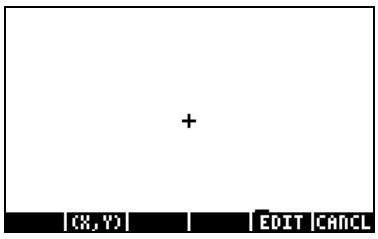
⑩ **PICTURE** : Muestra la ventana de gráficos. Realiza la misma acción como al presionar la tecla .

SINTAXIS:

PICTURE	⇨	

Ejemplo:

PICTURE



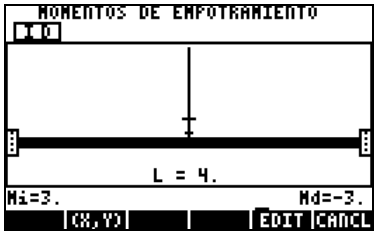
⑪ **PVIEW** : Este comando muestra la ventana de gráficos, pero desactiva el menu y el cursor, requiere una coordenada o una lista vacía ({ }).
Con este comando se puede visualizar la pantalla de gráficos de tres formas:

1: si se quiere visualizar la pantalla por solo unos instantes se utiliza { #0 #0 } junto con un número **n** y el comando **WAIT**.
2: si se quiere ver hasta que se presione una tecla se utiliza { #0 #0 } junto con el número **0** y el comando **WAIT**.
3: si se quiere ver hasta que se toque la tecla (**ON**), se utiliza { }.

SINTAXIS:

①	{ # 0 # 0 }	**PVIEW**	⇨
②	{ }	**PVIEW**	⇨

Ejemplo: Si en la ventana de gráficos esta dibujado el siguiente gráfico:



Visualiza la ventana de gráficos por 3 segundos.

Visualiza la ventana de gráficos, hasta que se presione una tecla.

Visualiza la ventana de gráficos hasta que se presione la tecla (**ON**).

⓬ **ANIMATE** : Realiza una animación con un determinado número de gráficos, requiere n gráficos y una lista en donde se indica la cantidad de gráficos a utilizar (n), la coordenada de visualización de los gráficos, tiempo de retraso para mostrar el siguiente gráfico y el número de ciclos. Una vez terminado la animación devuelve todos los objetos que se necesitaron para la animación.

SINTAXIS:

graf_1 graf_2 … graf_n { n { # m # n } t r } **ANIMATE** ⇨

graf_i : son los gráficos que se visualizaran

uno después de otro en el orden que se encuentra en la pila.

n : indica la cantidad de gráficos que se utilizará en la animación.

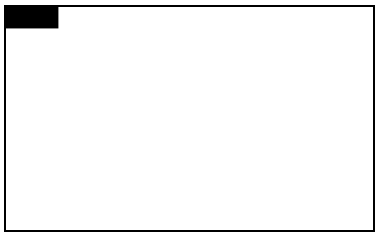
{ # m # n } : es la coordenada donde se visualizara los gráficos de la animación.

t : es el tiempo que demorara en mostrar el siguiente gráfico.

r : es el número de ciclos que tendrá la animación o cuantas veces se repetirá de visualizar los grupos de gráficos.

Ejemplo:

ANIMATE



En el ejemplo hay tres gráficos de la misma dimensión, la lista indica que hay tres gráficos, la coordenada de visualización es: { # 0 #0 } es lo mismo que { # 0d #0d }, el tiempo que demora en cambiar de gráfico es: 1seg y se repetirá 5 veces la animación de los 3 gráficos.

GRAFICACION DE DIAGRAMAS

En la calculadora hp también se pueden utilizar comandos para poder graficar varios tipos de diagramas, configurar los ejes y visualizar el diagrama.

LA VARIABLE EQ: Esta es una variable reservada de la calculadora. Se utiliza para almacenar una expresión para diagramas o una ecuación para la solución de ecuaciones. Para llamar el contenido almacenado en la variable **EQ** solo se escribe el nombre de la variable (**EQ**).

ALMACENAR UNA EXPRESION EN LA VARIABLE EQ: Se puede almacenar de dos maneras.

❶ **STO** : Almacena un objeto en una variable.

SINTAXIS:

objeto	**'EQ'**	**STO**	⇨

❷ **STEQ** : Almacena un objeto en la variable reservada **EQ** en el directorio actual.

SINTAXIS:

objeto **STEQ** ⇨

LA VARIABLE PPAR: Esta es una variable donde se almacena todos los parámetros de trazado de diagramas. Los parámetros de trazado están contenidos en una lista. Los parámetros que indica esta variable son las siguientes:

1: dos pares ordenados de números reales, el primer par indica las coordenadas de la esquina inferior izquierda y el segundo la esquina superior derecha de la región del plano cartesiano que se desea visualizar
(Xmin, Ymin) (Xmax, Ymax).
2: una lista donde de tres elementos el primer elemento indica la variable independiente el segundo y tercero indican el dominio del gráfico .Se puede colocar solamente la variable independiente
{ X Xinicial Xfinal } o X.
3: un número que indica el incremento de la variable independiente, este valor por defecto es cero (0).
4: una lista donde el primer elemento indica la coordenada del punto de intersección de los ejes del diagrama por defecto (0, 0), el segundo elemento es una lista donde sus elementos indican el espaciamiento de las marcas de los ejes X y Y respectivamente, luego dos caracteres que indican las etiquetas de los ejes
{ (0, 0) { n1 n2} "etiqueta_X" "etiqueta_Y" }.
5: el tipo de gráfico y la variable dependiente, por ejemplo: FUNCTION Y.

Para ingresar a la variable **PPAR** solo se tiene que escribir **PPAR**
obteniendo la siguiente lista:
{(Xmin,Ymin) (Xmax,Ymax) **{** X Xinicial Xfinal **}** 0 **{** (0,0) { n1 n2} "etiqueta_X" "etiqueta_Y"**}** **FUNCTION** Y**}.**

ELECCION DEL TIPO DE DIAGRAMA: Se elegirá el tipo de diagrama que trazara la calculadora, puede ser una función, diagrama polar, etc. Para configurar el tipo de diagrama de acuerdo a lo que se desee, se tiene que escribir lo siguiente:

FUNCTION : Establece el tipo en diagrama de función.

CONIC : Establece el tipo en diagrama cónica.

POLAR : Establece el tipo en diagrama polar.

PARAMETRIC : Establece el tipo en diagrama paramétrico.

Existen más tipos de diagramas que no se mencionaran.

CONFIGURACION DE LA VARIABLE PPAR Y GRAFICACION: Es necesario configurar esta variable para poder hacer un diagrama. Para configurar la variable **PPAR** existen comandos que se describirán a continuación.

❶ **XRNG** : Estableces el rango de visualización del eje X en la ventana de gráficos, requiere dos números reales que indiquen el rango.

SINTAXIS:

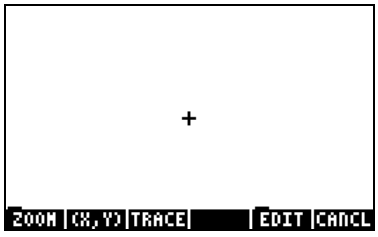
Xmin Xmax **XRNG** ⇨

Ejemplo:

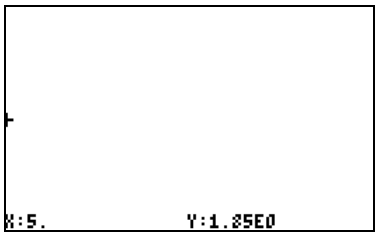
XRNG

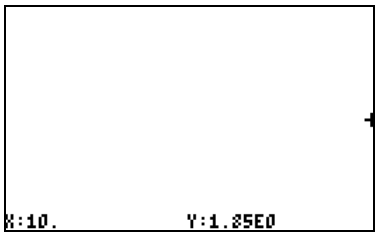
Para comprobar el rango ingresando en la ventana de gráficos se utilizará el comando **PICTURE**.

PICTURE



Ahora se presiona el menú **(X, Y)** y luego se ubica el cursor al lado izquierdo y al lado derecho de la pantalla observando lo siguiente:





En los dos últimos gráficos del ejemplo al observar el cursor y la parte inferior izquierda de la pantalla se observa x:5. y x:10. como lo establecido.

❷ **YRNG** : Estableces el rango de visualización del eje Y en la ventana de gráficos, requiere dos números reales que indiquen el rango.

SINTAXIS:

Ymin Ymax **YRNG** ⇨

Ejemplo: es similar como el ejemplo del comando **XRNG**.

❸ **INDEP** : Estableces la variable independiente.

SINTAXIS:

① 'variable' valor_inicial valor_final **INDEP** ⇨ ② 'variable' **INDEP** ⇨

valor_inicial : es el valor inicial que tomará la variable independiente.
valor_final : es el valor final que tomará la variable independiente.

❹ **DEPND** : Establece la variable dependiente.

SINTAXIS:

'variable' **DEPND** ⇨

❺ **AXES** : Establece la coordenada de intersección de los ejes X y Y, las anotaciones y las etiquetas de los ejes.

SINTAXIS:

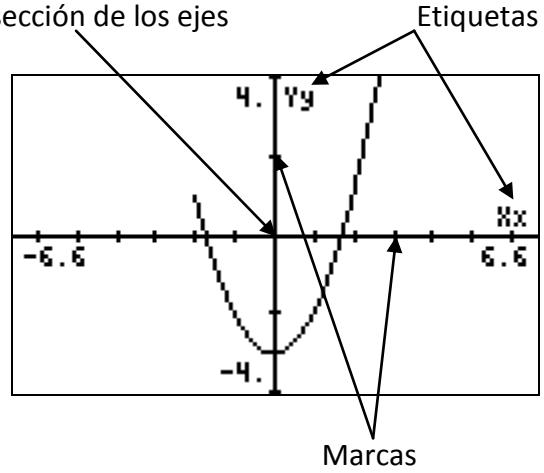
{ (x, y) { n1 n2 } "etiqueta_X" "etiqueta_Y" } **AXES** ⇨

(x, y) : es la coordenada de intersección de los ejes X y Y, si se asume la intersección como (0, 0), los ejes se intersectaran en el origen de coordenadas.
n1 : indica el espaciamiento de las marcas en el eje X.
n2 : indica el espaciamiento de las marcas en el eje y.
"etiqueta_X" : indica la etiqueta que tendrá el eje X, debe ser una cadena.

"etiqueta_Y" : indica la etiqueta que tendrá el eje Y, debe ser una cadena.

Coordenadas de intersección de los ejes

Etiquetas



Marcas

❻ **DRAX** : Dibuja los ejes y sus marcas, de acuerdo a la configuración hecha con el comando **AXES**.

SINTAXIS:

DRAX ⇨

❼ **LABEL** : Dibuja el valor mínimo (Xmin) y máximo (Xmax) establecido por el comando **XRNG**, el valor mínimo (Ymin) y máximo (Ymax) establecido por el comando **YRNG** en sus respectivas ubicaciones y las etiquetas establecidas por el comando **AXES**.

SINTAXIS:

LABEL ⇨

❽ **DRAW** : Traza la expresión almacenada en la variable **EQ**.

SINTAXIS:

DRAW ⇨

EJEMPLO DE TRAZADO DE UNA FUNCION

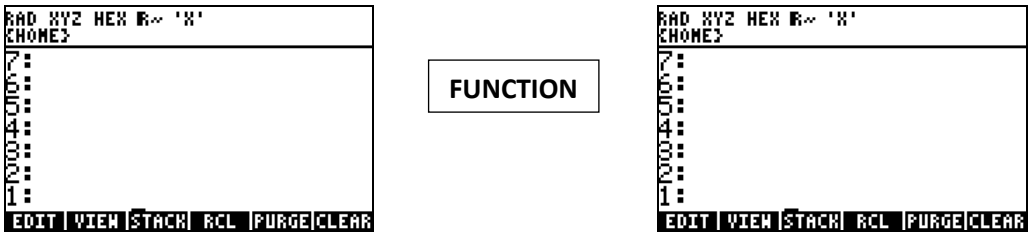
DATOS:	función	:	$y = x^2-3$
	variable independiente	:	x
	dominio de la función	:	[-2, 3]
	variable dependiente	:	y
	intersección de ejes	:	(0, 0)
	espaciamiento de las marcas en el eje X	:	1

espaciamiento de las marcas en el eje Y	:	2
etiqueta en el eje X	:	Xx
etiqueta en el eje Y	:	Yy
rango de visualización del eje X	:	[-6.5, 6.5]
rango de visualización del eje Y	:	[-4, 4]

Con los datos definidos se graficará la función utilizando solo comandos de la calculadora.

① borrar el contenido de las variables **PPAR** y **EQ** utilizando el comando **PURGE**.

② definir el tipo de diagrama en este caso **FUNCTION** (función).



③ almacenar la función en la variable **EQ** utilizando el comando **STEQ**.

④ definir la variable independiente y el dominio de la función utilizando el comando **INDEP**.

⑤ definir la variable dependiente utilizando el comando **DEPND**.

DEPND

⑥ definir la información de los ejes usando el comando **AXES**.

AXES

⑦ definir el rango de visualización del eje X utilizando el comando **XRNG**:

XRNG

⑧ definir el rango de visualización del eje Y utilizando el comando **YRNG**.

YRNG

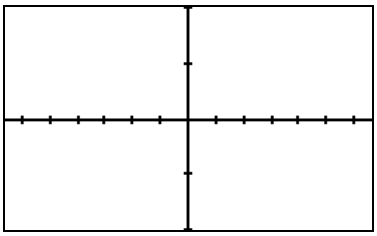
A partir de los siguientes comandos se mostrara la ventana de gráficos con fines didácticos.

⑨ borrar la ventana de gráficos utilizando el comando **ERASE**.

ERASE

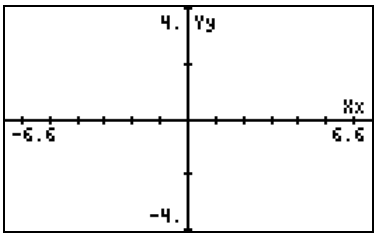
⑩ dibujar el eje y sus marcas usando el comando **DRAX**.

DRAX



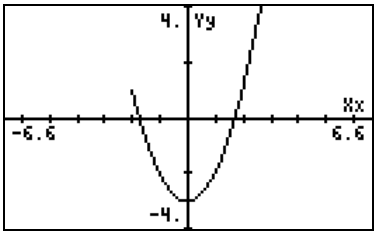
⑪ dibujar los valores extremos de los rangos de visualización y las etiquetas de los ejes usando el comando **LABEL**.

LABEL



⑫ dibujar la función usando el comando **DRAW**.

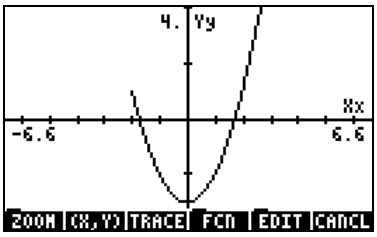
DRAW



Este comando muestra el gráfico solo el tiempo que demora en graficar la función, luego se visualiza la ventana de texto.

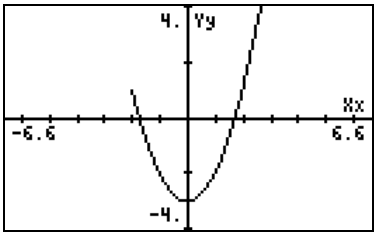
⑬ visualizar la ventana de gráficos usando el comando **PICTURE** o el comando **PVIEW**.

PICTURE



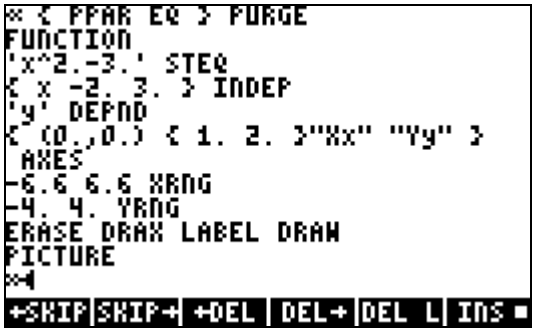
Al utilizarlo con **PICTURE** se observa que aparece el menú de edición, visualización, etc.

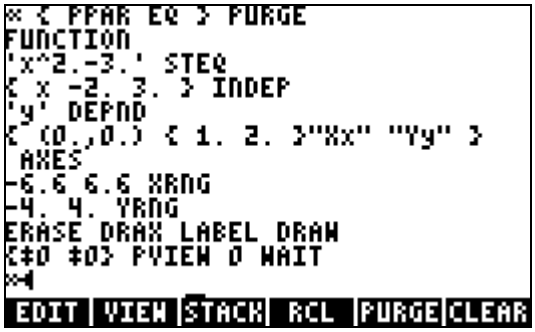
ENTER



Al utilizarlo con **PVIEW** y junto con el comando **WAIT** se observa que se visualiza el gráfico, hasta que se presione una tecla.

El programa quedaría de las siguientes formas:





21 CONSTRUCCION DE GRAFICOS USANDO CARACTERES HEXADECIMALES

En este capítulo se elaborarán gráficos de cualquier dimensión, usando solo caracteres hexadecimales, esto no quiere decir que se utilizará la base hexadecimal (solo los caracteres).

El formato general de un gráfico, para la calculadora hp es:

GROB m n código

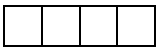
Donde:

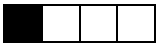
GROB	:	solo es un texto que siempre será el mismo, indica que se trata de un gráfico.
m	:	representa el ancho en pixeles que tiene o tendrá el gráfico.
n	:	representa la altura en pixeles que tiene o tendrá el gráfico.
código	:	indica la activación o desactivación de grupos de pixeles en el gráfico. Este código representa el aspecto que tendrá el gráfico.

Ejemplo:

GRUPOS DE PIXELES

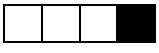
Para hacer el código del gráfico más fácil separaron los pixeles en grupos de cuatro, cada grupo de pixeles tienen su propio código (carácter hexadecimal). En los gráficos siguientes un cuadro en blanco representa un pixel desactivado y un cuadro de color negro representa un pixel activado.

 Código 0

 Código 1

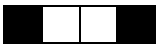
 Código 2

 Código 4

 Código 8

 Código 3

 Código 5

 Código 9

 Código 6

 Código A

 Código C

 Código 7

 Código B

 Código D

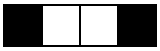
 Código E

 Código F

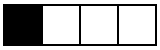
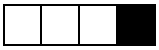
Se observa que el grupo de pixeles pueden tomar 16 formas, por ese motivo se utiliza los caracteres de la base hexadecimal.
Solo es necesario memorizar los códigos de los cinco primeros gráficos, los demás solo resultan de la suma de los cinco primeros. Al sumar los códigos se pueden obtener números mayores que nueve y a estos números se les representa con los siguientes caracteres:

10 = A
11 = B
12 = C
13 = D
14 = E
15 = F

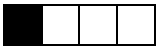
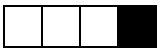
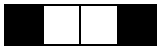
Ejemplo 1: obtener el código del siguiente arreglo de pixeles:



Se observa que está activo el primero y cuarto pixel, estos pixeles activos representan a los siguientes:

Ahora al sumar los dos pixeles activos manteniendo su posición, también sus respectivos códigos resultan:

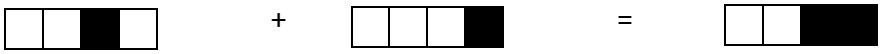
 +  = 

1 + 8 = **9**

Ejemplo 2: obtener el código del siguiente arreglo de pixeles:

Se observa que está activo el tercero y cuarto pixel, estos pixeles activos representan a los siguientes:

Ahora al sumar los dos pixeles activos manteniendo su posición también sus respectivos códigos resultan:



4 + 8 = **C** (**C** representa al número 12)

CODIFICACION DE UN GRAFICO

Se mostrará la forma general de hacer un gráfico por medio de caracteres, se siguen los siguientes pasos:

① Se coloca el texto GROB.

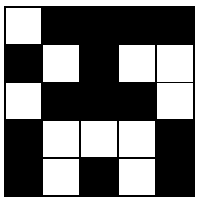
② Se coloca el número de pixeles horizontales y el número de pixeles verticales que tendrá el gráfico en números enteros.

③ Se separa el gráfico en columnas de cuatro pixeles de izquierda a derecha.

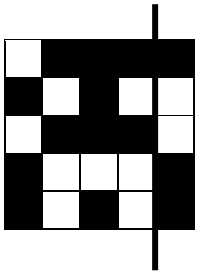
④ Si en la última columna hay menos de cuatro pixeles se completa con pixeles en blanco hasta tener cuatro.

⑤ Se coloca los códigos de los grupos de pixeles empezando con la primera fila de izquierda a derecha, luego con la segunda de izquierda a derecha, etc. Si se termina de colocar los códigos de los grupos de alguna fila y si el número de columnas es impar se coloca el código cero (0) al final de tosas las filas.

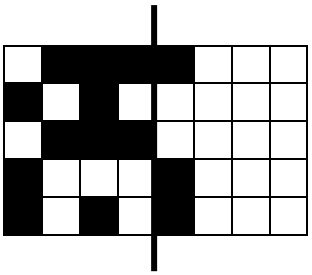
Ejemplo 1: construir el siguiente gráfico:



① Se separa el gráfico en columnas de cuatro pixeles.



② La segunda columna solo tiene un pixel, se lo completa con pixeles hasta tener cuatro.

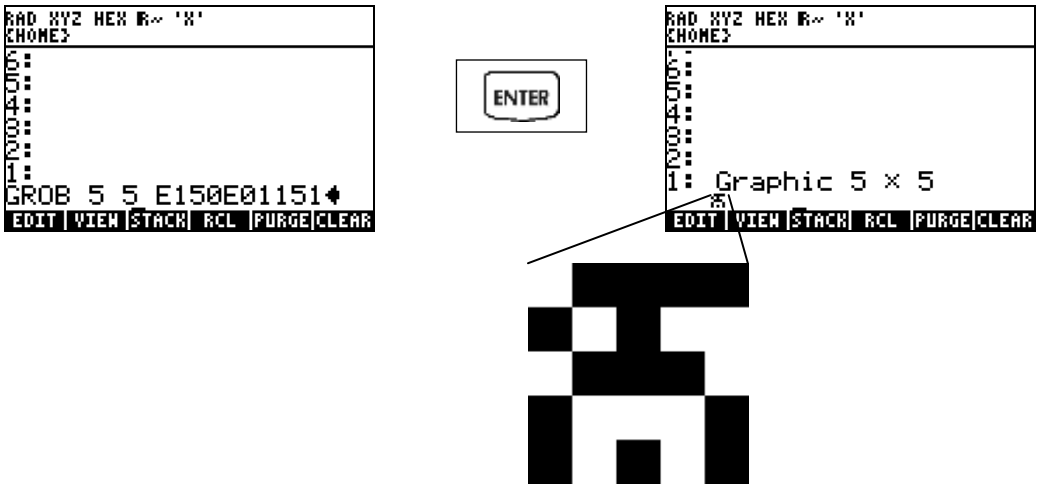


③ Se obtiene el código de los grupos de cada fila de pixeles, revisando los grupos de pixeles obteniendo los siguientes códigos:

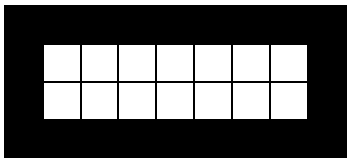
E	1
5	0
E	0
1	1
5	1

④ Ahora se coloca cada código en orden, de izquierda a derecha de arriba hacia abajo, es un gráfico de cinco filas y cinco columnas, el número de columnas es par por lo que no se aumenta el código cero (al final de cada fila)

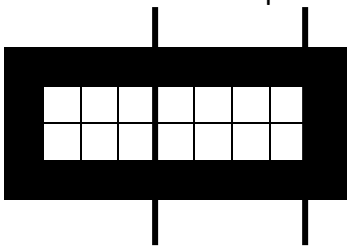
GROB 5 5 E150E01151



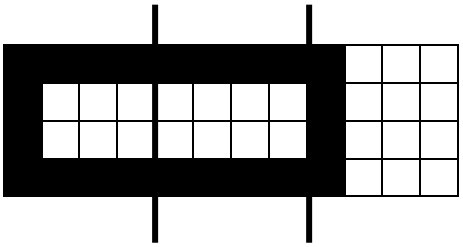
Ejemplo 2: construir el siguiente gráfico:



① Se separa el gráfico en columnas de cuatro pixeles



② La tercera columna solo tiene un pixel, se lo completa con pixeles hasta tener cuatro



③ Se obtiene el código de los grupos de cada fila de pixeles, revisando los grupos de pixeles obteniendo los siguientes códigos:

F	F	1
1	0	1
1	0	1
F	F	1

④ El número de columnas es impar por lo que se aumenta el código cero (al final de cada fila).

F	F	1	0
1	0	1	0
1	0	1	0
F	F	1	0

Es un gráfico de nueve pixeles horizontales y cuatro verticales.

GROB 9 4 FF1010101010FF10


RAD XYZ HEX R↵ 'X'
{HOME}
7:
6:
5:
4:
3:
2:
1:
GROB 9 4 FF1010101010FF104
EDIT VIEW STACK RCL PURGE CLEAR


ENTER


RAD XYZ HEX R↵ 'X'
{HOME}
7:
6:
5:
4:
3:
2:
1: Graphic 9 × 4
EDIT VIEW STACK RCL PURGE CLEAR

22 EDITORES

Los editores son ventanas en donde se puede ingresar objetos de la manera más cómoda y eficiente posible. La calculadora tiene varios editores como por ejemplo: el editor de ecuaciones, de matrices, gráficos, etc.

COMANDOS PARA ABRIR EDITORES

Los editores son una forma más fácil y cómoda de ingresar datos ya sea para programación o para ingresar objetos en la pila.

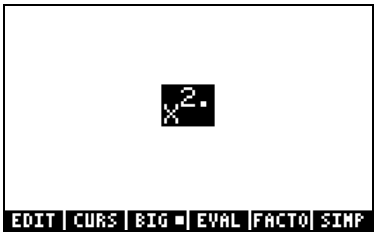
❶ **EQW** : Abre el editor de ecuaciones, requiere una expresión Adecuada.

SINTAXIS:

expresión_1 **EQW** ⇨ expresión_2

Ejemplo:

EQW



❷ **EDIT** : Coloca el objeto del nivel 1 en la línea de comandos para poder editarlo.

SINTAXIS:

objeto_1 **EDIT** ⇨ objeto_2

Ejemplo:

EDIT

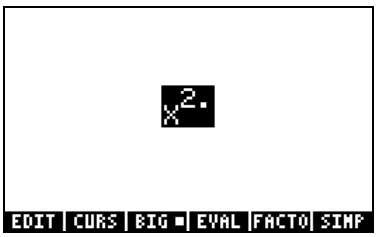
❸ **EDITB** : Abre un objeto en el entorno de edición más adecuado y devuelve el objeto modificado.

SINTAXIS:

objeto_1 **EDITB** ⇨ objeto_2

Ejemplo 1: editar una función en el editor de ecuaciones.

EDITB



Ejemplo 2: editar una matriz en el editor de matrices.

EDITB

Ejemplo 3: editar una cadena en el editor de textos.

EDITB

Ejemplo 4: editar un gráfico en el editor de gráficos.

EDITB

El comando **EDITB** se puede aplicar prácticamente en casi todos los programas para ingresar datos.

❹ **VISIT** : Coloca el contenido de una variable en la línea de comandos para poder visualizarlo o editarlo. Modifica el contenido de la variable.

SINTAXIS:

'variable' **VISIT** ⇨

Ejemplo: en la variable **VAR1** está guardado la siguiente expresión: X^2+3

VISIT

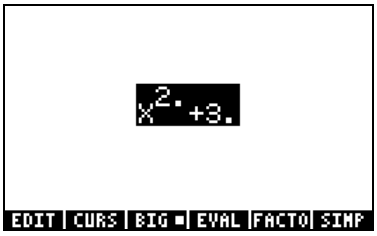
❺ **VISITB** : Abre el contenido de una variable en el entorno de edición más adecuado para su visualización o edición. Modifica el contenido de la variable.

SINTAXIS:

'variable' **VISITB** ⇨

Ejemplo:

VISITB



23 FECHA Y HORA

Estos comandos manipulan la fecha y hora.

❶ **DATE** : Devuelve la fecha actual que indica la calculadora en el siguiente formato : MM.DDAAAA
MM : indica el mes
DD : indica el día
AAAA : indica el año1.

SINTAXIS:

DATE ⇨ MM.DDAAAA

Ejemplo: primeramente se activa el reloj usando el comando **SF**, de la siguiente forma: -40 **SF**

DATE

Se observa en el primer gráfico la fecha 11 de mayo, en el segundo gráfico el número 5 representa el mes de mayo, el número 11 representa el día y por ultimo 2011 el año.

❷ **→DATE** : Fija la fecha del sistema al valor especificado, requiere la fecha en el formato siguiente: MM.DDAAAA.

SINTAXIS:

MM.DDAAAA **→DATE** ⇨

Ejemplo:

→DATE

La fecha en la calculadora ahora indica 14 de febrero al inicio era 11 de mayo.

❸ **TIME** : Devuelve la hora actual configurada en la calculadora en el siguiente formato : HH.MMSS
HH : indica la hora.

MM : indica los minutos.
SS : indica los segundos.

SINTAXIS:

TIME ⇨ HH.MMSS

Ejemplo:

TIME

En el segundo gráfico el número 7 indica la hora, el número 53 los minutos y el número 389060058 indica que pasaron 38.9060058 segundos.

❹ **→TIME** : Fija la hora del sistema al valor especificado, requiere la hora en el formato siguiente: HH.MMSS.

SINTAXIS:

HH.MMSS **→TIME** ⇨

Ejemplo:

→TIME

En el segundo gráfico la hora indica las 07:58 como se configuró.

❺ **DATE+** : Agrega o resta un número de días a una fecha.

SINTAXIS:

MM.DDAAAA_1 número_días **DATE+** ⇨ MM.DDAAAA_2

Ejemplo:

DATE+

En el primer gráfico la fecha indicada es el 12 de enero del 2011 y al agregarle 30 días la fecha será el 11 de febrero del 2011.

❻ **DDAYS** : Calcula el número de días entre dos fechas.

SINTAXIS:

MM.DDAAAA_1 MM.DDAAAA_2 **DDAYS** ⇨ MM.DDAAAA_2- MM.DDAAAA_1

Ejemplo:

DDAYS

En el primer gráfico la fecha indicada es el 12 de enero del 2011 y al agregarle 30 días la fecha será el 11 de febrero del 2011.

❼ **→HMS** : Convierte la hora de formato decimal a formato HH.MMSS. Este comando se puede utilizar para convertir un ángulo en grados sexagesimales a un ángulo es grados, minutos y segundos.

SINTAXIS:

x **→HMS** ⇨ HH.MMSS

Ejemplos:

→HMS

→HMS

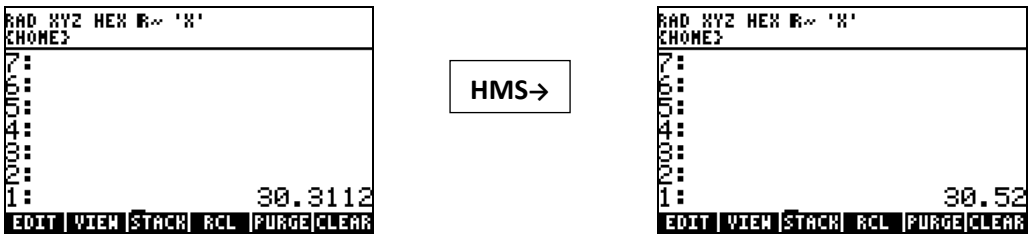
También se puede utilizar estos comandos para transformar ángulos. El equivalente de 30.52° es 30°31’12”

❽ **HMS→** : Convierte la hora de formato HH.MMSS a formato decimal. Este comando se puede utilizar para convertir un ángulo en grados, minutos y segundos sexagesimales a un ángulo en grados.

SINTAXIS:

HH.MMSS **HMS→** ⇨ x

Ejemplo:



❾ **HMS+** : Suma dos valores de horas en el formato HH.MMSS. Este comando se puede utilizar para sumar dos ángulo en grados, minutos y segundos sexagesimales.

SINTAXIS:

HH.MMSS_1 HH.MMSS_2 **HMS+** ⇨ HH.MMSS_1 + HH.MMSS_2

Ejemplo:

❿ **HMS-** : Resta dos valores de horas en el formato HH.MMSS. Este comando se puede utilizar para restar dos ángulo en grados, minutos y segundos sexagesimales.

SINTAXIS:

HH.MMSS_1 HH.MMSS_2 **HMS-** ⇨ HH.MMSS_1 - HH.MMSS_2

Ejemplo:

El resultado es 40.203 que es equivalente a 40 horas, 20 minutos y 30 segundos. Si se trataba de ángulos sexagesimales el resultado es equivalente a 40°20’30”.

⓫ **TSTR** : Convierte una fecha y una hora en una cadena, requiere la fecha en el formato MM.DDAAAA y la hora en el formato HH.MMSS obteniendo la cadena en el siguiente formato "día MM/DD/AAAA HH/MM/SS AoP".

SINTAXIS:

MM.DDAAAA HH.MMSS **TSTR** ⇨ "día MM/DD/AAAA HH/MM/SS AoP"

Ejemplo:

TSTR

24 SOLUCION DE ECUACIONES

En este capítulo se verán los comandos para la resolución de ecuaciones de la forma f(x)=0, algunas ecuaciones simbólicas y soluciones numéricas.

SOLUCION DE ECUACIONES SIMBOLICAS

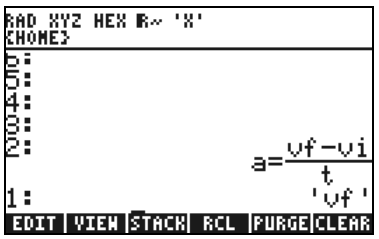
Estos comandos solucionan ecuaciones de una sola variable o ecuaciones racionales con dos o más variables.

❶ **ISOL** : Despeja si es posible una variable de una ecuación o expresión, requiere la variable a despejar.

SINTAXIS:

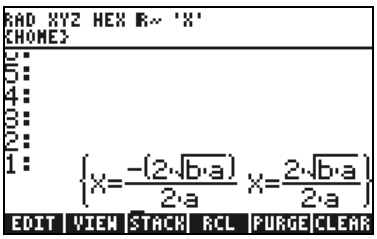
'expresión' 'variable' **ISOL** ⇨ solución

Ejemplos:



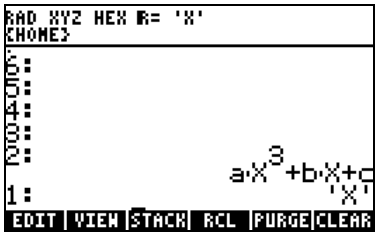
ISOL

ISOL

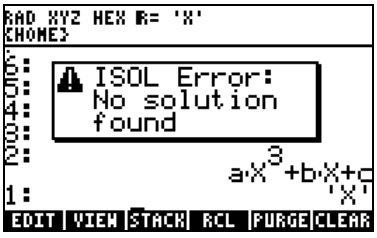


En este caso el flag 1 (solución general) está desactivado, por eso se puede observar las dos soluciones.

La ecuación se puede ingresar sin necesidad del símbolo "=".



ISOL



En este caso el comando **ISOL** no pudo despejar la variable X. **ISOL** solo resuelve algunas expresiones o ecuaciones racionales.

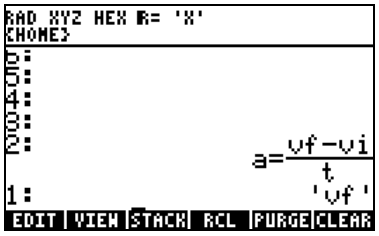
❷ **SOLVE** : Despeja si es posible una variable de una ecuación o

resuelve una ecuación polinómica, requiere la variable a despejar.

SINTAXIS:

'expresión'	'variable'	**SOLVE** ⇨	solución

Ejemplos:



SOLVE

SOLVE

Para que pueda resolver la ecuación es recomendable configurar en modo aproximado a la calculadora y también en modo complejo para visualizar las soluciones complejas.

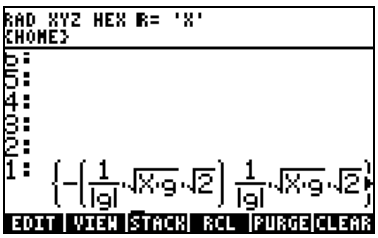
❸ **ZEROS** : Despeja si es posible una variable de una ecuación, de una expresión o una ecuación polinómica, requiere la variable a despejar.

SINTAXIS:

'expresión'	'variable'	**ZEROS** ⇨	solución

Ejemplo 1:

ZEROS



Ejemplo 2:

ZEROS

Se observa que devuelve una lista vacía, pero la ecuación tiene soluciones. Para solucionar esto es necesario poner a la calculadora en modo aproximado. Se observara la siguiente solución:

ZEROS

Para visualizar las soluciones complejas es necesario configurar a la calculadora en modo complejo. Se observara las siguientes soluciones en una lista:

ZEROS

❹ **SOLVEVX** : Resuelve una ecuación con respecto a la variable actual. El valor predefinido de la variable es la variable "**X**". La variable actual está almacenada en una variable del sistema de nombre "**VX**". **VX** es la variable predeterminada para hacer operaciones simbólicas.

SINTAXIS:

'expresión' **SOLVEVX** ⇨ solución

Ejemplo:

SOLVEVX

Para visualizar las soluciones no racionales y complejas es necesario configurar a la calculadora en modo aproximado y complejo.

SOLUCION DE ECUACIONES NUMERICAS

Estos comandos solucionan ecuaciones de una sola variable.

❶ **ROOT** : Resuelve una ecuación de una variable, requiere la variable y un valor inicial. La solución que devolverá este comando será la más próxima al valor inicial.

SINTAXIS:

'expresión' 'variable' valor_inicial **ROOT** ⇨ solución

Ejemplo:

ROOT

Si se trabaja con funciones trigonométricas es necesario configurar el modo angular al deseado.

❷ **PROOT** : Resuelve una ecuación polinómica, requiere los coeficientes del polinomio ordenado decrecientemente en un vector.

SINTAXIS:

vector_1 **PROOT** ⇨ solución_en_vector

Ejemplo:

PROOT

Al ingresar el vector [1. -5. 6.] se refiere a la ecuación: X^2-5*X+6=0, el vector resultante [2. 3.] representa las soluciones de la ecuación y son: X=2 y X=3.

SOLUCION DE UNA ECUACION USANDO SOLVE EQUATION

Esta herramienta sirve para resolver todo tipo de ecuaciones. Se ingresará al menú SOLVE EQUATION llamándolo directamente utilizando su XLIB correspondiente.

LA VARIABLE EQ: Esta es una variable reservada de la calculadora, en este caso se utiliza para almacenar una expresión para la solución de ecuaciones. Para llamar el contenido almacenado en la variable **EQ** se escribe el nombre de la variable (**EQ**). Para almacenar una expresión en esta variable es necesario utilizar los comandos **STO** o **STEQ** (descritos en el capítulo de GRAFICOS-GRAFICACION DE DIAGRAMAS).

LLAMAR EL MENU SOLVE ECUATION: No existe un nombre que llame a esta biblioteca, se lo llama utilizando la función **LIBEVAL** (evalúa funciones de aquellas bibliotecas sin nombre), el número de esta biblioteca es el número binario hexadecimal #B4001h
Se llama la biblioteca de la siguiente forma: #B4001h **LIBEVAL**. Si se hace un programa primero se guarda la expresión a resolver en la variable **EQ** y luego se activa la biblioteca.

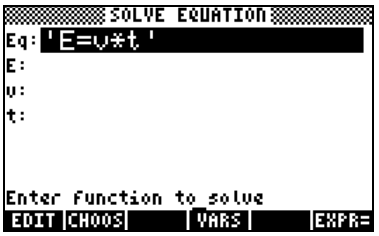
Ejemplo: Hallar el valor de "t" en la siguiente ecuación: E=v*t, para E=10 y v=2.

① Primero se almacena la ecuación en la variable **EQ** utilizando el comando **STEQ**.

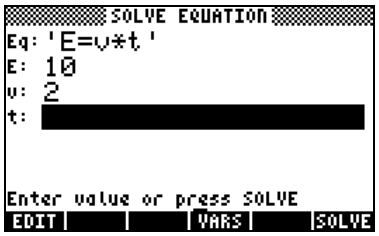
STEQ

② Ya almacenado la ecuación ahora se llama la biblioteca usando el comando **LIBEVAL**.

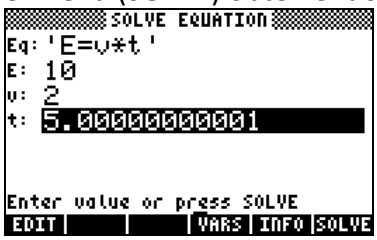
LIBEVAL



③ Dentro de la biblioteca se ingresa los datos en las variables correspondientes.



④ Ya ingresado los datos, se posiciona en la variable a calcular y luego se presiona el menú (**SOLVE**) obteniendo la respuesta.



SOLUCION DE MULTIPLES ECUACIONES USANDO EL MES

Con esta herramienta se puede resolver sistemas de ecuaciones teniendo solo los datos necesarios de algunas variables.

LA VARIABLE EQ: En este caso se almacena en la variable una lista de ecuaciones de la siguiente forma: { EQ1 EQ2 EQ3 }; EQ1, EQ2 y EQ3 son las ecuaciones.

❶ **MINIT** : Inicializa las variables de las ecuaciones almacenadas en la variable **EQ**.

SINTAXIS:

MINIT ⇨

❷ **MITM** : Configura la visualización del **MES**, requiere una cadena que será el título de la ventana y una lista de las variables de las ecuaciones en el orden que se los desee visualizar en el área de menús.

SINTAXIS:

"título" { var1 var2 … varn } **MITM** ⇨

❸ **MSOLVR** : Activa el **MES**, requiere que en la variable **EQ** este almacenado una lista de ecuaciones que resolverá el **MES**.

SINTAXIS:

MSOLVR ⇨

Ejemplo: Se resolverá las ecuaciones referentes a la caída libre, se utilizará dos ecuaciones:

$$h = V0 * t + \frac{g * t^2}{2} \qquad g = \frac{V - V0}{t}$$

① Primero se almacena la lista de ecuaciones en la variable **EQ**, utilizando el comando **STEQ**.

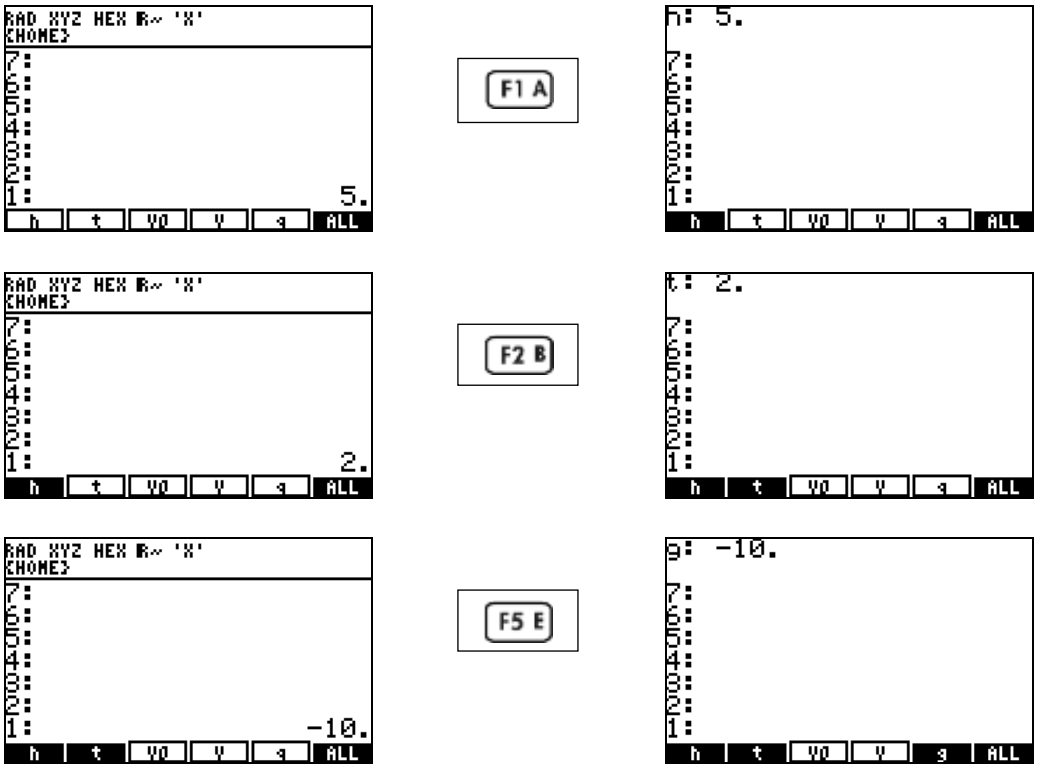
② Se inicializa las variables de las ecuaciones almacenadas en **EQ**, utilizando el comando **MINIT**.

③ Se configura la visualización de la ventana del **MES**, utilizando el comando **MITM**.

④ Se activa el **MES**, utilizando el comando **MSOLVR**.

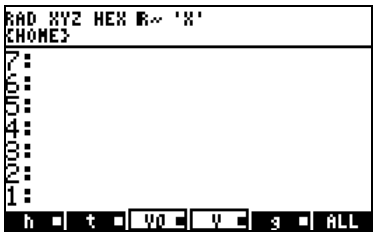
⑤ Para ingresar los valores de las variables conocidas se ingresa el valor numérico, luego se presiona la tecla de la variable correspondiente. Se asumirá los siguientes datos numéricos para las variables necesarias:

Datos: h = 5m
t = 2s
$g = -10m/s^2$



⑤ Para calcular la solución de una variable desconocida primero se presiona la tecla de cambio izquierdo ( ), seguido de la tecla correspondiente a la variable a calcular. Si se desea calcular todas las variables desconocidas a la vez, se presiona la tecla de cambio izquierdo ( ) seguido de la tecla correspondiente al menu **ALL**, si no se visualiza el menú **ALL** se presiona la tecla NXT hasta visualizarlo**.** Se calculará todas las variables.

⑥ Para visualizar todas las variables calculadas a la vez se presiona la tecla de cambio derecho ( ) seguido de la tecla correspondiente al menu **ALL.** Se visualizará todas las variables.



*) Para reiniciar el ingreso de datos se presiona la tecla correspondiente del menú **ALL**.

OTROS COMANDOS

Estos comandos son de ayuda para la resolución de ecuaciones.

❶ **LININ** : Prueba si una expresión algebraica es lineal con respecto a una variable.

SINTAXIS:

'expresión' **LININ** ⇨ 1 ó 0

Ejemplos:

LININ

LININ

❷ **LNAME** : Devuelve la expresión original y obtiene los nombres de las variables en un vector.

SINTAXIS:

'expresión' **LNAME** ⇨ 'expresión' vector_variables

Ejemplo:

LNAME

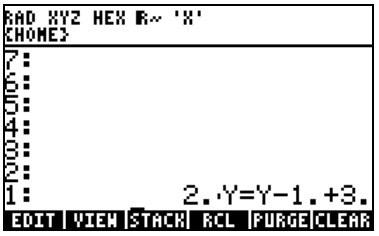
❸ **SUBST** : Substituye una variable de una expresión por otra expresión o número, requiere la expresión inicial y la variable igualada a otra expresión.

SINTAXIS:

'expresión_1' 'variable_n=expresión_2' **SUBST** ⇨ 'expresión_3'

Ejemplos:

SUBST



Reemplazó a la variable X de la primera expresión por la segunda expresión(X=Y-1), sin evaluarlo.

SUBST

Para reemplazar varias variables de una sola vez, es necesario ingresar las variables con sus respectivas expresiones en un vector.

❹ **EQ→** : Obtiene los dos miembros de una ecuación o igualdad.

SINTAXIS:

'expresión_1 = expresión_2' **EQ→** ⇨ 'expresión_1' 'expresión_2'

Ejemplos:


RAD XYZ HEX R↔ 'X'
{HOME}
7:
6:
5:
4:
3:
2:
1: X+3.=2.√X-6.
EDIT VIEW STACK RCL PURGE CLEAR

EQ→


RAD XYZ HEX R↔ 'X'
{HOME}
7:
6:
5:
4:
3:
2: X+3.
1: 2.√X-6.
EDIT VIEW STACK RCL PURGE CLEAR

25 UNIDADES

Es una cantidad estandarizada de una determinada magnitud física. La calculadora hp, puede trabajar con unidades, manipulándolos, cambiando de unidad, hacer operaciones, etc.

UNIDADES DE LA CALCULADORA

Solo se mostrara algunas unidades de la calculadora y su respectivo valor en el sistema internacional (SI). En la columna de nombre están los nombres de las unidades, en la de unidad están los símbolos que reconoce la calculadora hp y en la tercera columna están los equivalentes en el SI.

nombre	unidad	valor en unidades del SI
Metro	m	1_m
Yarda internacional	yd	0.9144_m
Pie internacional	ft	0.3048_m
Pulgada	in	0.0254 m
Año luz	lyr	$9.46052840488 * 10^{15}$_m
Milla internacional	mi	1609.344_m
Milla náutica	nmi	1852_m
Angstrom	Å	$1 * 10^{-10}$_m
Hectárea	ha	10000_m^2
Área	a	100_m^2
Acre	acre	4046.87260987_m^2
kilolitro	st	1_m^3
Litro	l	0.001_m^3
Galón, EE.UU.	gal	0.003785411784_m^3
Galón, Canadá	galC	0.00454609_m^3
Galón, Reino Unido	galUK	0.004546092_m^3
Segundo	s	1_s
Año	yr	31556925.9747_s
Día	d	86400_s
Hora	h	3600_s
Minuto	min	60_s
Hertz	Hz	1_1/s
Kilómetros por hora	kph	0.277777777778_m/s
Millas por hora	mph	0.44704_m/s
Millas náuticas por hora	kmot	0.514444444444_m/s
Velocidad de la luz	c	299792458_m/s
Caída libre estándar	ga	9.80665_m/s^2

nombre	Unidad	valor en unidades del SI
Kilogramo	kg	1_kg
Gramo	g	0.001_kg
Libra	lb	0.45359237_kg
Onza	oz	0.028349523125_kg
Tonelada métrica	t	1000_kg
Tonelada corta	ton	907.18474_kg
Quilate	ct	0.0002_kg
Masa atómica unificada	u	$1.6605402 * 10^{-27}$_kg
Mol	mol	1_mol
Newton	N	1_kg*m/s^2
Dina	dyn	0.00001_kg*m/s^2
Fuerza-gramo	gf	0.00980665_kg*m/s^2
Fuerza-kilopondio	kip	4448.22161526_kg*m/s^2
Fuerza-libra	lbf	4.44822161526_kg*m/s^2
Poundal	pdl	0.138254954376_kg*m/s^2
Julio	J	1_kg*m^2/s^2
Ergio	erg	0.0000001_kg*m^2/s^2
Caloría	cal	4.1868_kg*m^2/s^2
Vatio	W	1_kg*m^2/s^3
Potencia en C.V.	hp	745.699871582_kg*m^2/s^3
Pascal	Pa	1_kg/m*s^2
Atmósfera	atm	101325_kg/m*s^2
Bara	bar	100000_kg/m*s^2
Libras por pulgada cuadrada	psi	6894.75729317_kg/m*s^2
Torr (mmHg)	torr	133.322368421_kg/m*s^2
Milímetro de mercurio, 0°C	mmHg	133.322368421_kg/m*s^2
Kelvin	K	1_K
Grados Celsius	°C	274.15_K
Grados Fahrenheit	°F	255.927777778_K
Grados Rankine	°R	0.555555555556_K
Poise	P	1_kg/m*s
Estokesio	St	0.0001_ m^2/s

PREFIJOS DE UNIDADES DE MEDIDA

Las unidades vistas anteriormente y la que faltan se les puede insertar un prefijo en la unidad de medida, para indicar la potencia de base 10 con un exponente indicado con el cual se multiplicará.

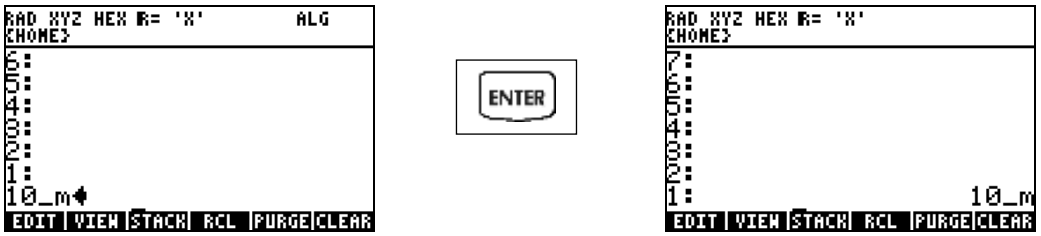
prefijo	nombre	Exponente de diez
Y	yotta	24
Z	zetta	21
E	exa	18
P	peta	15
T	tera	12
G	giga	9
M	mega	6
K o k	kilo	3
H o h	hecto	2
D	deka	1
d	deci	-1
c	centi	-2
m	mili	-3
µ	micro	-6
n	nano	-9
P	pico	-12
f	femto	-15
a	atto	-18
z	zepto	-21
y	yocto	-24

No se puede utilizar un prefijo a una unidad que tiene la calculadora si el texto de la unidad resultante resulta ser igual a otra unidad, por ejemplo: la unidad área(a) se desea que tenga el prefijo de hecto (h) La unidad resultante es (ha), pero esta unidad representa a la unidad hectárea (ha).

INGRESAR UNA UNIDAD A LA CALCULADORA

Se ingresa el escalar luego el símbolo "**_**" seguido de la unidad, para ingresar el símbolo anterior se puede utilizar la aplicación CHARS (véase el CAPITULO CADENAS DE CARACTERES) o el teclado utilizando el estrato de cambio derecho correspondiente a la tecla primaria menos("-") (véase el CAPITULO TECLADO).

Ejemplos:



Ejemplos: se ingresarán unidades prefijadas.

Se ingresará 10 milímetros:

La unidad es el metro (m) el prefijo es mili (m) por lo tanto la unidad prefijada es “mm”.

Se ingresará 1 kilolitro equivalente a 1000 litros:

La unidad es el litro (l) el prefijo es kilo (K) por lo que la unidad prefijada es “Kl”.

OPERACIONES CON UNIDADES

Se puede realizar las cuatro operaciones, radicación, potenciación, etc.

Ejemplos:

+

-

*

/

+

Al sumar dos magnitudes escalares iguales pero de diferente unidad el resultado estará en la unidad que tenga el ultimo objeto unidad (objeto del nivel 1), en este caso es el metro (m).

COMANDOS DE UNIDADES

Estas herramientas son los siguientes comandos:

❶ →UNIT : Construye una unidad con un número real y una

expresión de unidad.

SINTAXIS:

x y_unidad **→UNIT** ⇨ x_unidad

Ejemplos:

→UNIT

→UNIT

❷ **UVAL** : Obtiene la parte numérica de un objeto unidad.

SINTAXIS:

x _unidad **UVAL** ⇨ x

Ejemplos:

UVAL

UVAL

❸ **UBASE** : Convierte unidades a unidades del sistema internacional.

SINTAXIS:

x _unidad **UBASE** ⇨ y_unidad(SI)

Ejemplos:

UBASE

UBASE

❹ **CONVERT** : Convierte un objeto unidad a su equivalente en otra unidad.

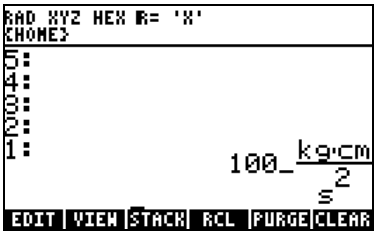
SINTAXIS:

x _unidad_1	y _unidad_2	**CONVERT**	⇨	z _unidad_2

Ejemplos:

CONVERT

CONVERT



En el primer ejemplo no importó el número que tenga el último objeto (objeto del nivel 1).

❺ **UFACT** : Factoriza el objeto unidad del nivel 2, uno de sus factores será la unidad del objeto unidad del nivel 1.

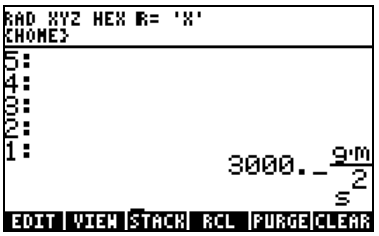
SINTAXIS:

x _unidad_2	y _unidad_1	**UFACT**	⇨	z _unidad_1*unidad_3

Ejemplo:


RAD XYZ HEX R= 'X'
{HOME}
7:
6:
5:
4:
3:
2: 3_N
1: 1_g
EDIT VIEW STACK RCL PURGE CLEAR

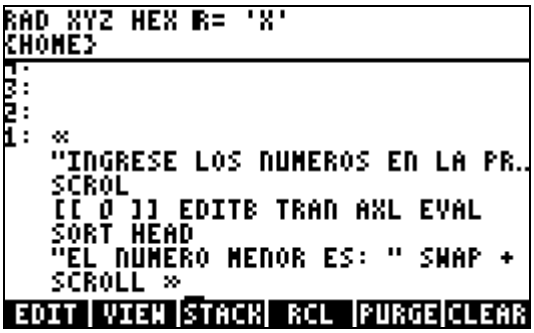
UFACT


RAD XYZ HEX R= 'X'
{HOME}
5:
4:
3:
2:
1: 3000._g·m/s^2
EDIT VIEW STACK RCL PURGE CLEAR

26 EJEMPLOS DE PROGRAMACION

HALLAR EL MENOR DE UN GRUPO DE NUMEROS

El programa quedaría de la siguiente forma:



Este programa se tiene que guardar en una variable para utilizarlo.

EXPLICACION:

«

"INGRESE LOS NUMEROS EN LA PRIMERA COLUMNA"	:	pone en la pila la cadena
SCROLL	:	presenta la cadena anterior en la pantalla
[[0]]	:	pone en la pila esta matriz
EDITB	:	abre el editor más idóneo (editor de matrices) Para ingresar los datos
TRAN	:	halla la transpuesta de una matriz
AXL	:	convierte una matriz en una lista
EVAL	:	obtiene el o los elementos de una lista
SORT	:	ordena la lista en orden ascendente
HEAD	:	obtiene el primer elemento de la lista
"EL NUMERO MENOR ES: "	:	pone a la pila la cadena
SWAP	:	intercambia la posición de la cadena y el número anterior
+	:	concatena o une una cadena y un objeto
SCROLL	:	presenta la cadena obtenida en la pantalla

»

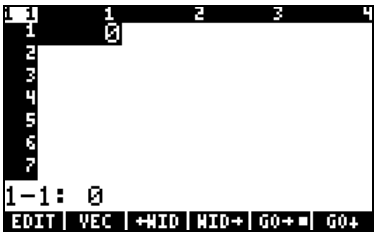
EXPLICACION CON ARGUMETOS: la mayoría de las ventanas no aparecerán al correr el programa pero se los mostrará para fines didácticos

① Primero ingresó el programa la cadena "INGRESE LOS NUMEROS EN LA PRIMERA COLUMNA", luego el comando **SCROLL**, este visualiza la cadena anterior con un formato adecuado.

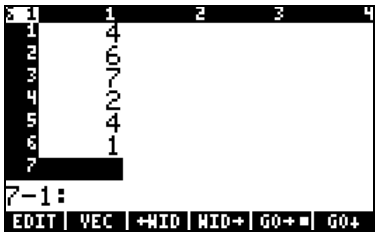
SCROLL

② Después de presionar el menú (**OK**) ingresará el programa a la pila la matriz [[0]] y el comando **EDITB**, este comando sirve para visualizar la matriz anterior y luego editarlo.

EDITB



Una vez que se visualice el editor de matrices se tiene que ingresar los datos en la primera columna.

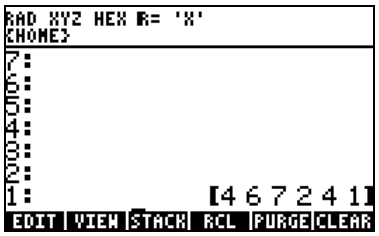


ENTER

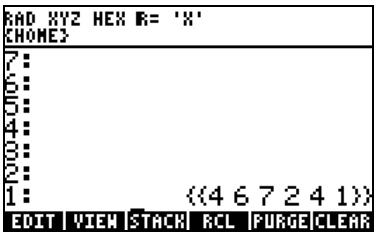
③ El programa ejecuta **TRAN**, halla la transpuesta de una matriz.

TRAN

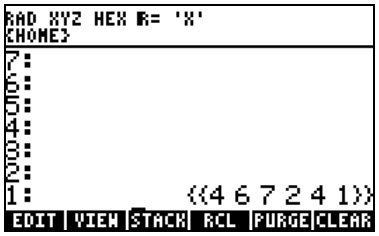
④ El programa ejecuta **AXL**, convierte un vector o matriz en una lista o viceversa.



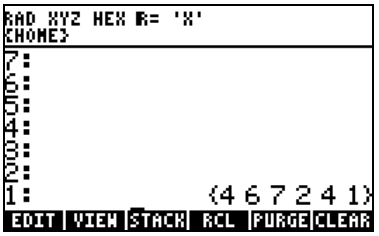
AXL



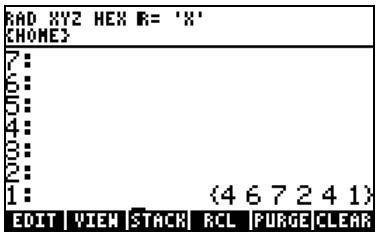
⑤ El programa ejecuta **EVAL**, en este caso obtiene el o los elementos de una lista



EVAL



⑥ El programa ejecuta **SORT**, ordena los elementos de una lista de menor a mayor.



SORT

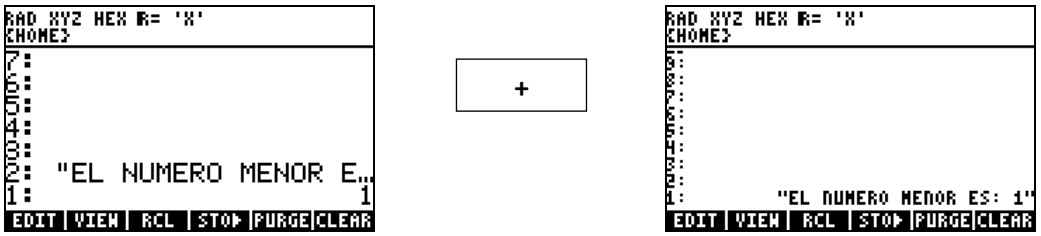
⑦ El programa ejecuta **HEAD**, devuelve el primer elemento de una lista.

HEAD

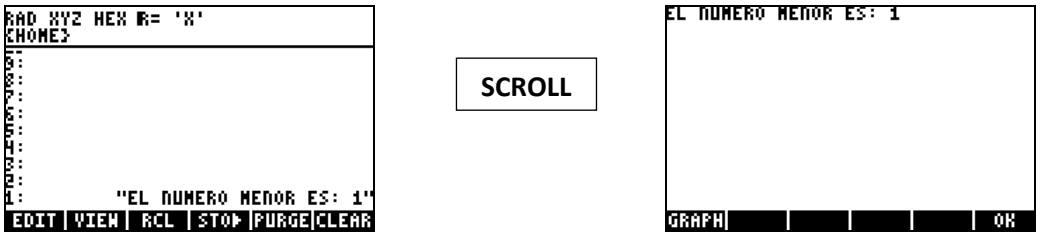
⑧ En el programa sigue la cadena "EL NUMERO MENOR ES: ", luego el comando **SWAP**, este comando intercambia los objetos que se encuentran en los niveles 1 y 2 de la pila.

SWAP

⑨ En el programa sigue el operador **+**, este operador cuando uno de sus argumentos es una cadena lo concatena con el segundo argumento. Se modificará la pantalla para poder visualizar mejor la concatenación.



⑩ En el programa sigue el comando **SCROLL**, este comando visualiza un objeto en la pantalla con un formato adecuado.



PRESENTAR EN LA PANTALLA LOS CARACTERES DE UNA CADENA UNO POR UNO

```
« ?→ cadena1
  «  1. cadena SIZE
  FOR i
        cadena1  1.  i  SUB  1.  DISP  .3  WAIT
  NEXT
  »
»
```

EXPLICACION CON ARGUMETOS: La mayoría de las ventanas no aparecerán al correr el programa pero se los mostrará para fines didácticos.

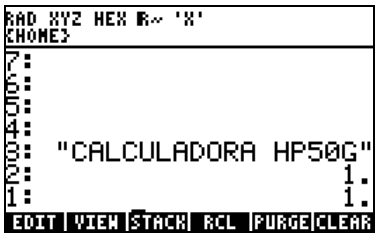
① Primeramente se ingresa una cadena en la pila, luego se ejecuta el programa. Se ingresará la cadena "CALCULADORA HP 50G" en la pila. El programa ejecuta el símbolo →, este símbolo almacena en la variable local (cadena) la cadena anterior.

② El programa coloca en la pila el número 1. y el contenido de la variable cadena1 ("CALCULADORA HP 50G") y luego ejecuta el comando **SIZE**, en este caso este comando obtiene la dimensión de una cadena.

SIZE

③ El programa ejecuta el comando **FOR NEXT**, los argumentos necesarios para este comando son: los números 1. , 17. y la variable temporal que se encuentra después de **FOR** (i).
Las instrucciones que se realizaran son:
cadena1 1. i **SUB** 1. **DISP** .3 **WAIT**

① El programa coloca el contenido de la variable cadena1 ("CALCULADORA HP 50G"), luego el número 1., luego el valor de la variable temporal i (1.) y ejecuta el comando **SUB**, en este caso este comando obtiene una sub cadena de la cadena anterior.



SUB

② El programa coloca en la pila el número 1., luego ejecuta el comando **DISP**, este comando muestra en la pantalla una cadena en la posición indicada. El número 1. indica que la cadena se visualizara en la parte superior de la pantalla. El programa coloca el número 0.3 en la pila y ejecuta el comando **WAIT**, este comando hace que el programa haga una pausa el tiempo deseado.

DISP 0.3 **WAIT**

Este proceso iterativo se realizará 17 veces. En la última iteracion se verá la cadena completa.

DIBUJAR Y CALCULAR EL AREA DE UN POLIGONO

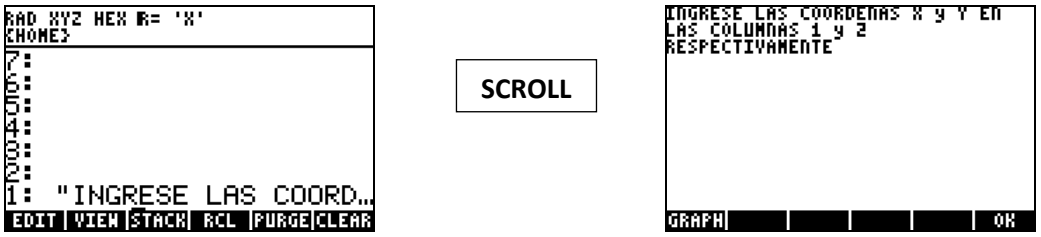
Determinará el área con las coordenadas indicadas y dibujará el polígono que se forma con las mismas.

```
«
"INGRESE LAS COORDENAS X y Y EN LAS COLUMNAS 1 y 2 RESPECTIVAMENTE"
```

```
SCROLL
[ [ 0 0 ] ] EDITB
TRAN
AXL
EVAL
'CY' STO
'CX' STO
CX SORT  HEAD CX SORT REVLIST HEAD XRNG
CY SORT  HEAD CY SORT  REVLIST HEAD YRNG
ERASE
{#0 #0} PVIEW
PICT {#4d #4d} "espere... " 1 →GROB REPL
PICT {#4d #4d}  "AREA : "
0 CX + CY CY HEAD + * ΣLIST
CX CX HEAD + 0 CY + * ΣLIST
- ABS 2 / + 1 →GROB
    1 CX SIZE FOR i
        CX CX HEAD + i GET CY CY HEAD + i GET R→C
        CX CX HEAD + i 1 + GET CY CY HEAD + i 1 + GET R→C
        LINE
    NEXT
    REPL
0 WAIT DROP
{ CX CY } PURGE
        »
```

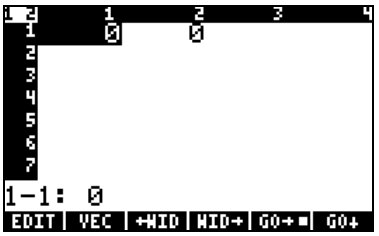
EXPLICACION CON ARGUMETOS: La mayoría de las ventanas no aparecerán al correr el programa pero se los mostrará para fines didácticos.

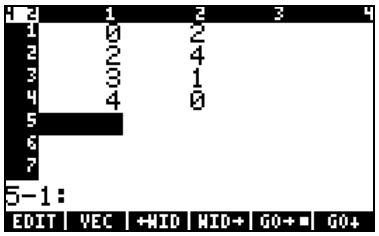
① Primero el programa ingresó la cadena "INGRESE LAS COORDENAS X y Y EN LAS COLUMNAS 1 y 2 RESPECTIVAMENTE", luego **SCROLL**, este comando sirve para visualizar un objeto con el formato más adecuado.



② Después de presionar el menú (**OK**), ingresará el programa a la pila la matriz [[0 0]] y el comando **EDITB**, este comando sirve para visualizar la matriz anterior y luego editarlo. Ya ingresado los datos en la matriz se presiona la tecla (**ENTER**).

EDITB



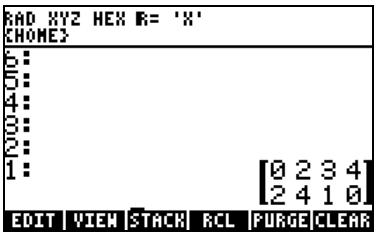


ENTER

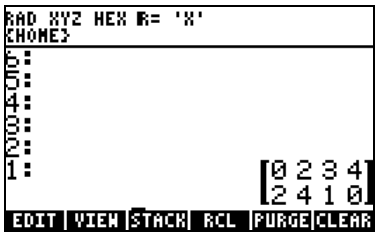
③ Luego se obtendrá dos listas de los componentes de las coordenadas, las cuales se guardarán en las variables globales CX y CY.

*) El programa ejecuta **TRAN** este comando halla la transpuesta de una matriz.

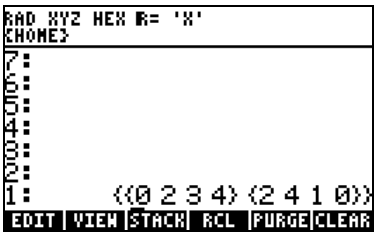
TRAN



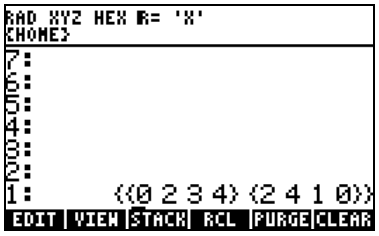
*) El programa ejecuta **AXL**, convierte una matriz en una lista de listas o viceversa.



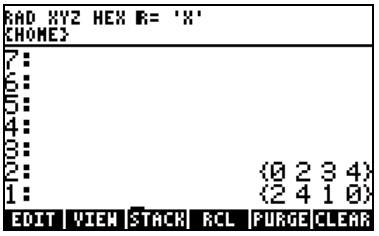
AXL



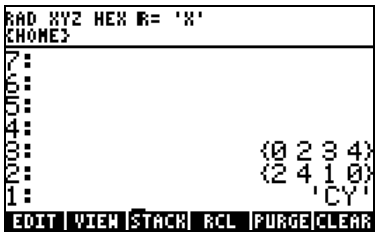
*) El programa ejecuta **EVAL**, en este caso obtiene los elementos de una lista.



EVAL



*) El programa coloca la variable 'CY' y luego ejecuta **STO**, guarda un objeto en la variable especificada.



STO

*) El programa coloca la variable 'CX' y luego ejecuta **STO**, guarda un objeto en la variable especificada.

STO

④ El programa definirá los rangos de visualización de Las coordenadas de usuario o cartesianas en la ventana de gráficos.

*) El programa llama al contenido de la variable global CX (en la variable global CX se encuentra la lista {0 2 3 4}) luego ejecuta **SORT** este comando ordena los elementos de una lista de menor a mayor.

SORT

*) El programa ejecuta **HEAD**, obtiene el primer elemento de una lista.

HEAD

*) El programa llama al contenido de la variable global CX (en la variable global CX se encuentra la lista {0 2 3 4}), luego ejecuta **SORT** este comando ordena los elementos de una lista de menor a mayor.

SORT

*) El programa ejecuta **REVLIST**, invierte el contenido de una lista.

REVLIST

*) El programa ejecuta **HEAD**, obtiene el primer elemento de una lista.

HEAD

*) El programa ejecuta el comando **XRNG**, establece el rango de visualización en el eje de las abscisas en la ventana de gráficos.

XRNG

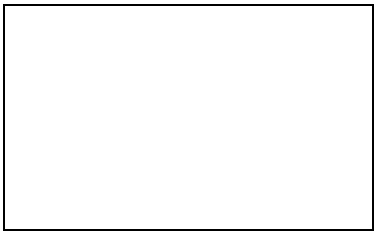
*) El programa llama el contenido de la variable CY, luego ejecuta los mismos procedimientos anteriores pero al final ejecuta el comando **YRNG**, este comando establece el rango de visualización en el eje de las ordenadas en la ventana de gráficos.

⑤ El programa calculará el área usando las listas obtenidas inicialmente y además dibujará el polígono pero sin escala. En el proceso no se verá la pantalla de texto solo se visualizará la ventana de gráficos, pero se mostrará la pantalla de texto por fines didácticos.

*) El programa ejecuta el comando **ERASE**, este comando borra la pantalla de gráficos.

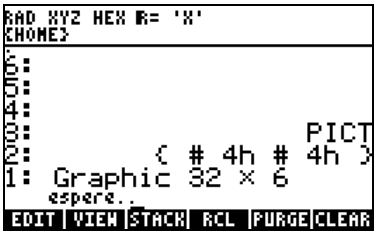
*) El programa pone en la pila la coordenada del pixel {#0 #0}, luego ejecuta el comando **PVIEW**, este comando muestra la pantalla de gráficos solo por un instante, pero cuando sigue ejecutando el programa más sentencias se seguirá visualizando la ventana de gráficos hasta que ya no ejecute alguna sentencia.

PVIEW

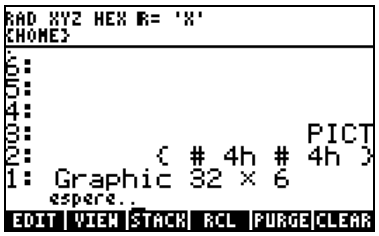


*) El programa pone en la pila los siguientes objetos: PICT, {#4d #4d}, "procesando... " y 1 luego ejecuta el comando **→GROB**, este comando convierte un objeto en gráfico.
NOTA: en el programa se colocó la coordenada de en pixeles {#4d #4d}, pero en el gráfico de abajo aparece como { # 4h # 4h }. Esto sucedió porque está activo la base hexadecimal y convirtió a {#4d #4d} en su equivalente en el sistema hexadecimal { # 4h # 4h }.

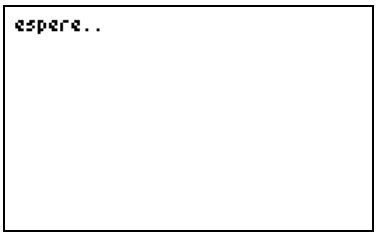
→GROB



*) El programa ejecuta el comando **REPL**, este comando reemplaza una parte de un gráfico por otro. En este caso reemplazara una parte de PICT (objeto contenido en la ventana de gráficos).

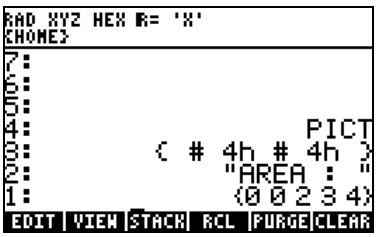


REPL

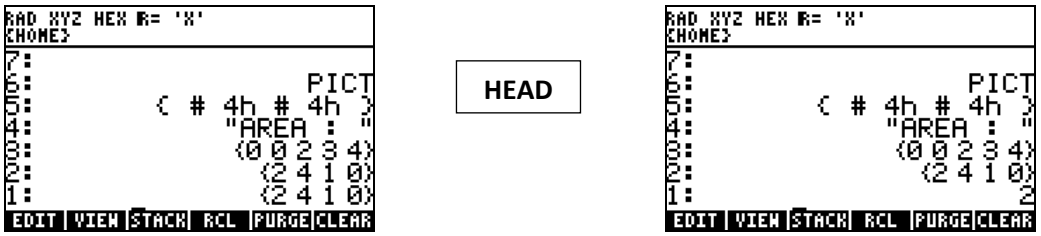


*) El programa pone en la pila los siguientes objetos: PICT, {#4d #4d}, "AREA : ", 0 y CX, luego ejecuta el operador **+**, en este caso el operador une el número 0 con la lista contenida en la variable CX.

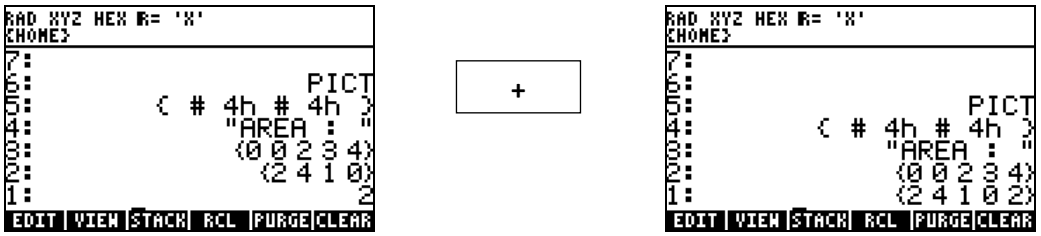
+



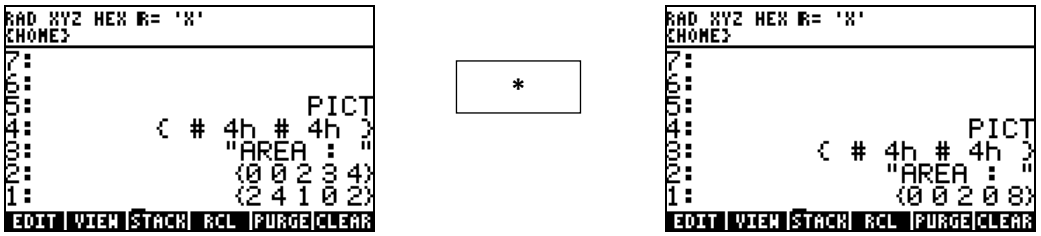
*) El programa pone en la pila los siguientes objetos: CY y CY, luego ejecuta el comando **HEAD**, este comando obtiene el primer elemento de una lista.



*) Luego ejecuta el operador **+**, en este caso el operador inserta a la lista el número 2.

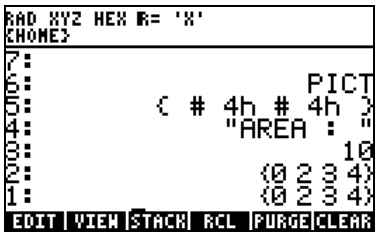


*) Luego ejecuta el operador *, en este caso el operador multiplica miembro a miembro los elementos de las dos listas.



*) Ejecuta **ΣLIST**, suma todos los elementos contenidos en una lista.

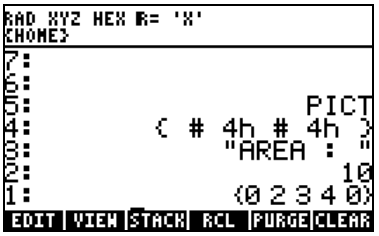
*) Pone a la pila dos veces CX, luego ejecuta el comando **HEAD**, este comando obtiene el primer elemento de una lista.



HEAD

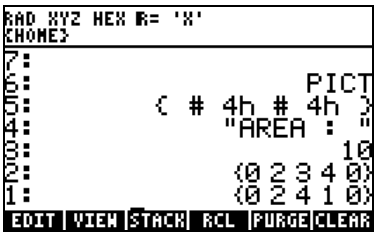
*) Ejecuta **+**

+

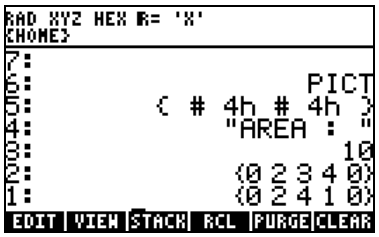


*) Pone a la pila 0 y CY, luego ejecuta **+**

+



*) Ejecuta ***

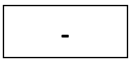


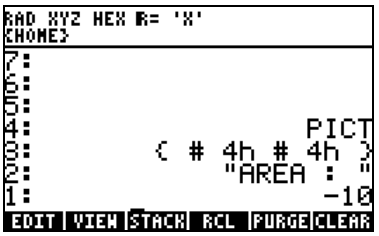
*****

*) Ejecuta **ΣLIST**

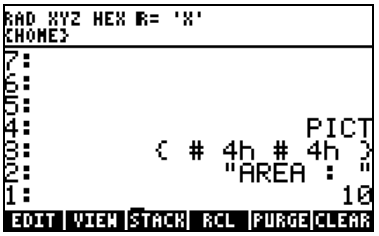
ΣLIST

*) Ejecuta **-**

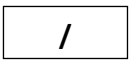




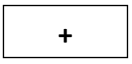
*) Ejecuta **ABS**, determina el valor absoluto de un número.

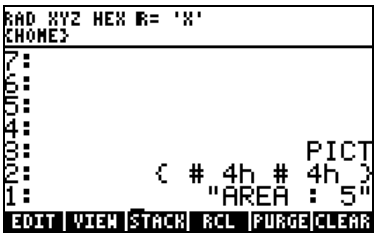


*) Pone a la pila el número 2 y ejecuta **/**

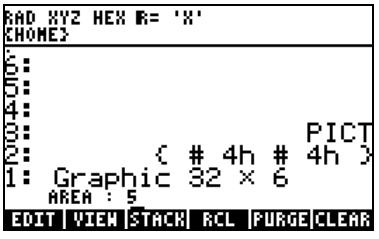


*) Ejecuta **+**, en este caso concatena la cadena y el número.



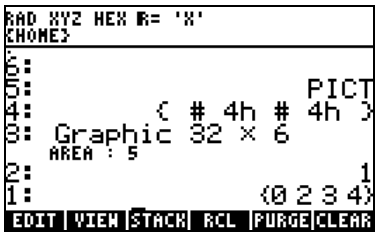


*) Pone a la pila el número 1 y ejecuta el comando **→GROB**, este comando convierte un objeto en un gráfico.

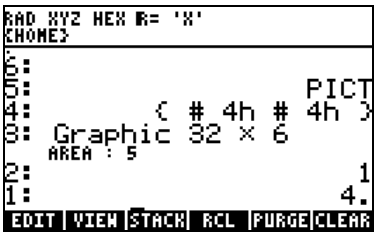


LOS SIGUIENTES PROCEDIMIENTOS DIBUJAN EL POLIGONO

*) Pone a la pila el número 1, el contenido de la variable CX y ejecuta el comando **SIZE**, este comando en este caso determina la cantidad de elementos de una lista.



SIZE



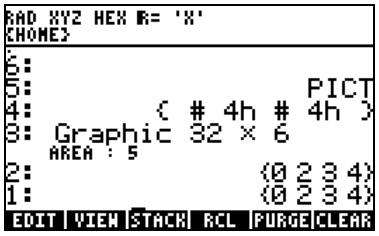
Estos dos últimos números representan los valores inicial y final que tomará el contador i de la instrucción **FOR NEXT**, quiere decir que las expresiones contenidas entre las instrucciones **FOR NEXT** se ejecutaran 4 veces con la única variación de que el contador i variará de 1 en 1 desde el número 1 hasta el número 4. Solo se hará una iteración de los cuatro, ya que es el mismo procedimiento para todos.

*) Al ejecutar la instrucción o comando **FOR**, ya no habrá los números 1, 4 y tampoco la variable i que se encontraba delante de **FOR**, porque este comando lo utilizó para poder definir el número de iteraciones y la variable temporal. Ejecutará las siguientes expresiones:

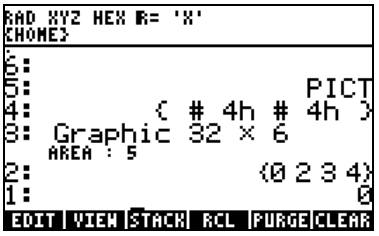
CX CX **HEAD +** i **GET** CY CY **HEAD +** i **GET R→C**
CX CX **HEAD +** i 1 **+ GET** CY CY **HEAD +** i 1 **+ GET R→C**
LINE.

Para la primera iteración el contador i tomará el valor de 1 (i=1).

) Pone en la pila el contenido de la variable CX dos veces y ejecuta **HEAD.

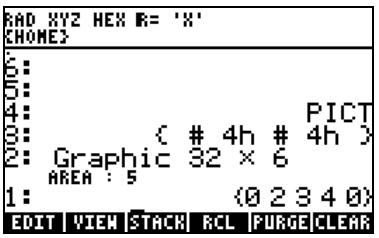


HEAD

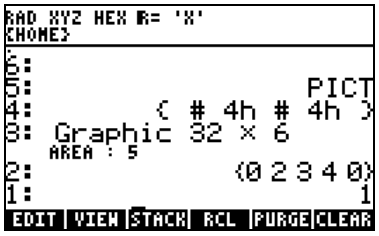


) Ejecuta **+, en este caso introduce el número a la lista.

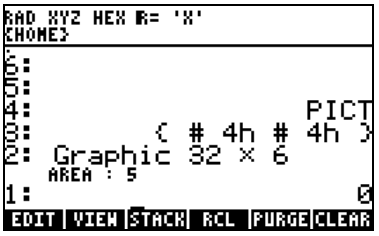
+



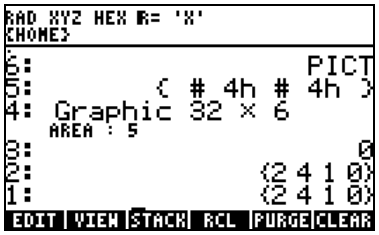
) Llama al contador i (en este caso el número 1) y ejecuta el comando **GET, este comando obtiene un elemento de posición indicada de la lista.



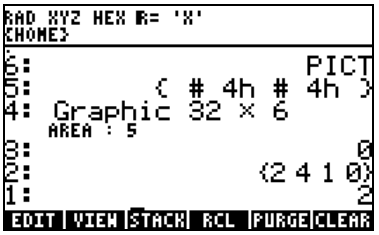
GET



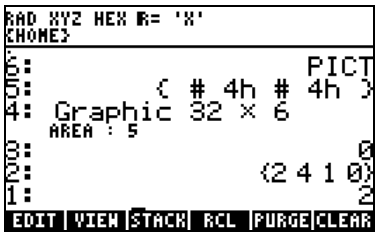
) Pone a la pila el contenido de la variable CY dos veces y ejecuta **HEAD.



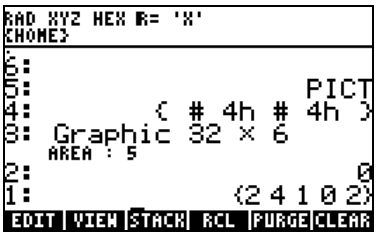
HEAD



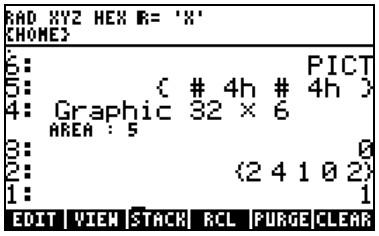
) Ejecuta **+, en este caso introduce el número a la lista.



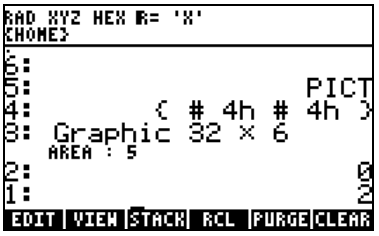
+



) Llama al contador i (en este caso el número 1) y ejecuta el comando **GET, este comando obtiene un elemento de posición indicada de la lista.

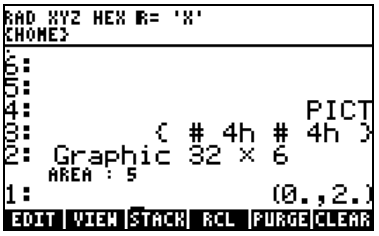


GET



) Ejecuta **R→C, convierte dos números reales en las componentes de una coordenada cartesiana.

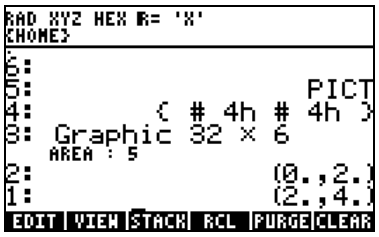
R→C



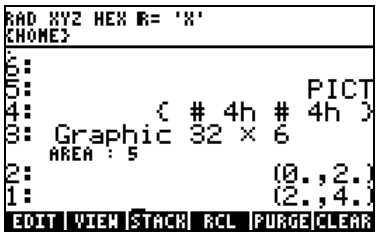
**) Ejecutará las siguientes expresiones:

CX CX **HEAD +** i 1 **+ GET** CY CY **HEAD +** i 1 **+ GET R→C**

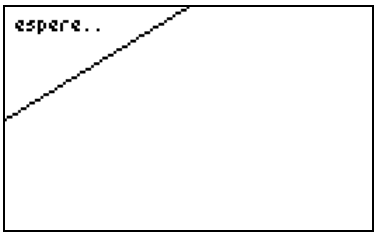
Estas expresiones son prácticamente iguales a las anteriores, con la única diferencia que le añade una unidad al contador i y hace esto para obtener la segunda coordenada del polígono, obteniendo así las dos coordenadas de los extremos de un lado del polígono.



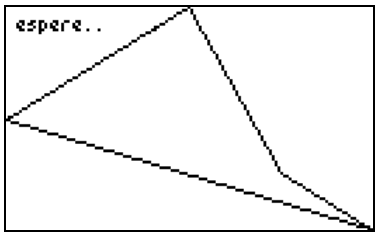
) Ejecuta el comando **LINE, este comando dibuja una línea en la ventana de gráficos, requiere dos coordenadas.



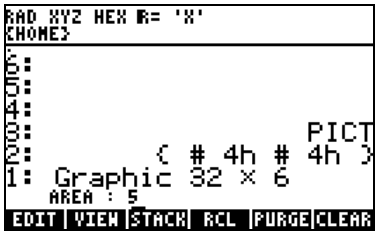
LINE



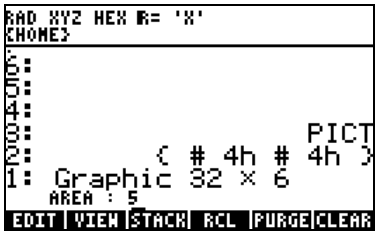
Al ejecutarse todas las iteraciones, la ventana de gráficos quedará de la siguiente forma:



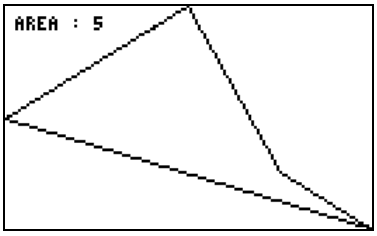
En la pila aún quedaría objetos como en el gráfico de abajo.



) Después de termina las instrucciones de **FOR NEXT se ejecuta el comando **REPL**, este comando reemplaza una parte de un gráfico por otro, en este caso reemplazará una parte de PICT (objeto contenido en la ventana de gráficos).



REPL



) Después la pila queda vacía luego el programa pone a la pila el número 0 y ejecuta el comando **WAIT, este comando hace que se visualice la ventana actual (ventana de gráficos) hasta que se presione una tecla y devuelve el código de la tecla presionada.
) Después de presionar una tecla el programa ejecuta el comando **DROP, este comando elimina el objeto que se encuentra en el nivel 1, en este caso el código de tecla obtenido con el comando **WAIT**.
) Pone a la pila una lista de las variables globales creadas { CX CY } y ejecuta el comando **PURGE, este comando elimina las variables globales indicadas.

FORMULARIO

En este formulario se podrán crear nuevas fórmulas, ejecutar las formulas, visualizar la ayuda de la formulas y borrar formulas. Será necesario un programa con nombre de FORMULARIO y subprogramas: DIRECTORIO, INGRESO, EJECUTAR, VISUALIZAR y BORRAR. Para que funcione el programa y los subprogramas, se los debe almacenar directamente en el directorio HOME.
Nota: para poder visualizar mejor los objetos en la pantalla de la calculadora, se modificará la configuración de la pantalla como: el tamaño de texto en la pila, en la línea de comando, etc.

DIRECTORIO: Este subprograma creará un directorio en el directorio HOME e ingresará en ella. En este directorio se almacenarán todos los datos de las fórmulas ingresadas por el formulario. Si el directorio ya está creado, solo ingresara en él.

```
«
HOME
IF 'DatForm' VTYPE 15 ==
THEN DatForm
ELSE 'DatForm' CRDIR DatForm
END
»
```

EXPLICACION:

① El programa hará que el directorio actual sea el HOME, con el comando **HOME**.

HOME

② Se analizarán las instrucciones que se encuentran entre **IF**, **THEN**, **ELSE** y **END**.

El objeto de nombre 'DatForm' será el nombre del directorio donde se almacenarán los datos de las formulas. Hay dos casos: el primero que exista la carpeta con nombre 'DatForm' en el directorio HOME y en el segundo caso no.

*) En el caso de que exista la carpeta 'DatForm' en el directorio HOME.

① El programa coloca en la pila el nombre 'DatForm' y luego obtiene el tipo de objeto, utilizando el comando **VTYPE**. Este comando obtiene el tipo de objeto contenido en una variable. Como en este caso existe el directorio, se obtendrá el tipo de objeto número 15. (objeto directorio o carpeta).

VTYPE

② El programa introduce el número 15. en la pila y luego ejecuta el operador relacional **==**, este operador verifica si dos números son iguales, obteniendo 1. en caso contrario 0.
Para poder ingresar el símbolo = dos veces en la línea de comandos, se puede utilizar la aplicación CHARS.

==

③ Este valor obtenido (1.) se encuentra entre las instrucciones **IF** y **THEN**. Como el número 1. representa el valor de verdad de verdadero, entonces se ejecutarán las sentencias contenidas entre **THEN** y **ELSE.**

④ La sentencia entre **THEN** y **ELSE** es el objeto directorio DatForm. Al enviar el objeto DatForm a la pila el programa abre el directorio.

DatForm

*) En el caso de que no exista la carpeta DatForm en el directorio HOME.

① El programa pone en la pila el nombre 'DatForm' y luego obtiene el tipo de objeto, utilizando el comando **VTYPE**. Este comando obtiene el tipo de objeto contenido en una variable.

VTYPE

Cuando se obtiene -1. indica que no existe ningún objeto con el nombre DatForm.

② El programa introduce el número 15. en la pila y luego ejecuta el operador relacional **==**, este operador verifica si dos números son iguales, obteniendo 1. si lo son, en caso contrario 0.

==

③ Como el valor de verdad contenido entre las instrucciones **IF** y **THEN** es 0. (falso), entonces se ejecutarán las sentencias contenidas entre **ELSE** y **END.**

④ El programa coloca el nombre 'DatForm' luego ejecuta el comando **CRDIR**. Este comando crea un directorio en la ruta actual y con el nombre dado.

CRDIR

⑤ El programa envía el objeto DatForm a la pila. El programa ingresa al directorio.

DatForm

INGRESO: Este subprograma creará un directorio e ingresará en ella, luego creará una carpeta con el nombre de la fórmula y guardará en ella la ecuación, variables y la descripción de las variables.

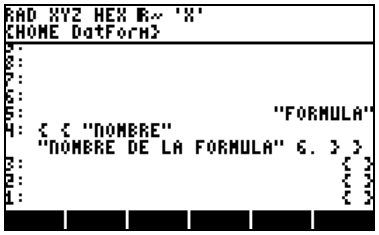
Antes de ejecutar el subprograma INGRESO, el programa principal ingresará al directorio DatForm.

```
«
"FORMULA"
{ { "NOMBRE" "NOMBRE DE LA FORMULA" 6 } }
{ }{ }{ } INFORM
IF
THEN
    OBJ→ DROP
    DUP VTYPE 15 ==
    IF
    THEN
        "YA EXISTE ESTE NOMBRE" MSGBOX DROP
    ELSE
        DUP CRDIR EVAL 'X' EQW
        LNAME AXL 'VARIABLES' STO 'ECUACION'  STO
        { }
        1 VARIABLES SIZE
        FOR i
        "VARIABLE"
        "DESCRIPCION"
        "DESCRIPCION DE LA VARIABLE "
        VARIABLES i GET + 2 3 →LIST 1 →LIST
        { }{ }{ "" } INFORM
        NOT
        IF
        THEN
           VARIABLES i GET "" +
        END
           +
        NEXT
        'DESCRIPCION' STO
```

```
        UPDIR
    END
END
    »
```

EXPLICACION:

① El programa coloca en la pila los siguientes objetos: "FORMULA", { { "NOMBRE" "NOMBRE DE LA FORMULA" 6 } }, { }, { } y { } luego ejecuta el comando **INFORM**, este comando sirve para ingresar datos en la pila. Se observa dentro del segundo objeto de la pila el número 6. este valor indica que se debe ingresar un nombre global.



INFORM

Una vez que se visualice el formulario de ingreso, se tiene que ingresar el nombre de la fórmula.
Se ingresará el nombre 'MRU' (movimiento rectilíneo uniforme).

② Se analizarán las instrucciones que se encuentran entre **IF**, **THEN** y **END**.

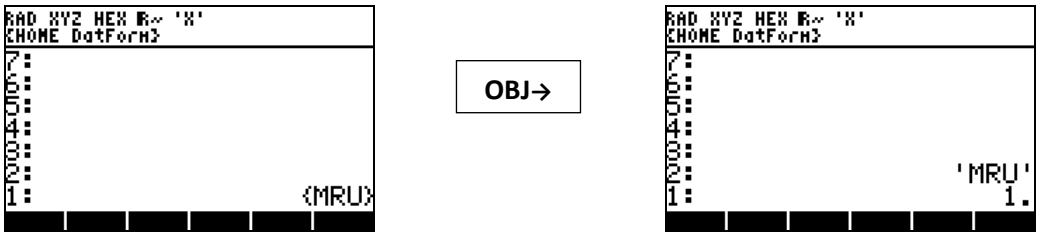
Al ejecutar el comando **INFORM** se obtiene dos objetos, el primer objeto es una lista con los datos ingresados y el segundo objeto es el número 1 ó 0. Se obtiene el número 1 si se salió del formulario presionando la tecla ENTER o la tecla F6 correspondiente al menú OK, se obtiene 0 cuando se presiona la tecla ON o la tecla F5 correspondiente al menú CANCEL.

*) En el caso en que se presione la tecla ENTER o la tecla correspondiente al menú OK.

① Como se presionó la tecla ENTER se obtendrá una lista y el número 1.
Este valor obtenido (1.) se encuentra antes de la instrucción **IF** y además entre las instrucciones **IF** y **THEN** no hay ningún objeto, entonces se tomará el valor que se encuentra antes de **IF** como valor de verdad, por lo que se ejecutarán las sentencias contenidas entre **THEN** y **END.**

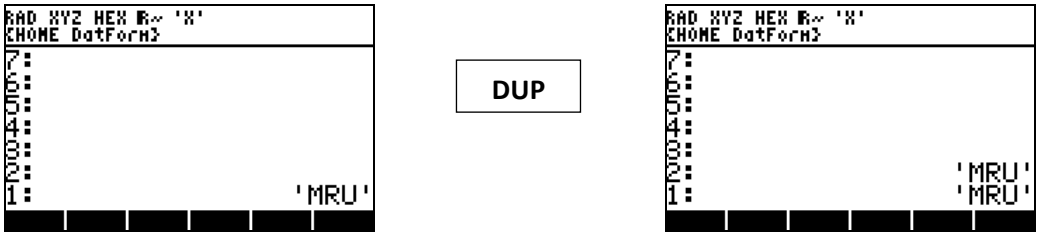
Después de la sentencia **IF** en la pila solo quedará la lista:

② El programa ejecuta el comando **OBJ→**, este comando en este caso descompone la lista, obteniendo sus elementos y la cantidad de elementos.

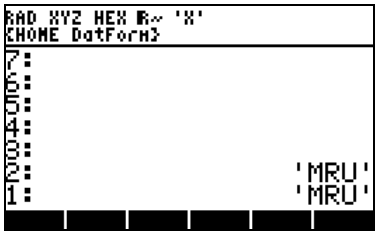


③ El programa ejecuta el comando **DROP**, este comando elimina el objeto del nivel 1.

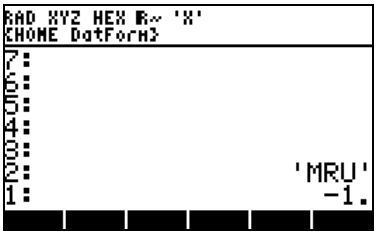
④ El programa ejecuta el comando **DUP**, este comando duplica el objeto del nivel 1.



⑤ El programa ejecuta el comando **VTYPE**, este comando obtiene el número del objeto almacenado en una variable.

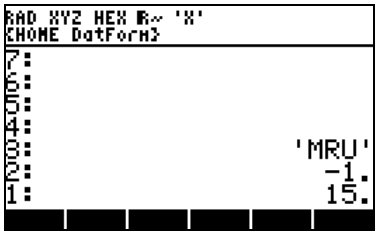


VTYPE

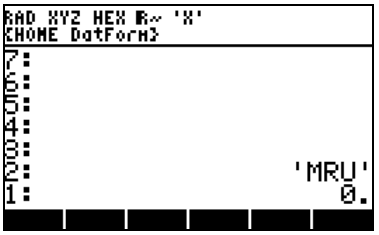


Se obtuvo el número -1, porque en la variable MRU no existe ningún objeto almacenado. En este caso MRU es un nombre global.

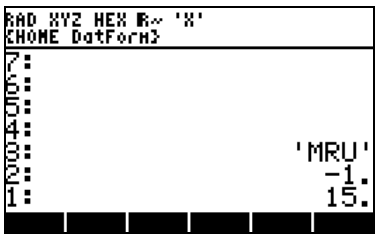
⑥ El programa coloca el número 15 en la pila y luego ejecuta el operador relacional **==**, este operador verifica si dos números son iguales. Si son iguales se obtiene 1. En caso contrario 0.



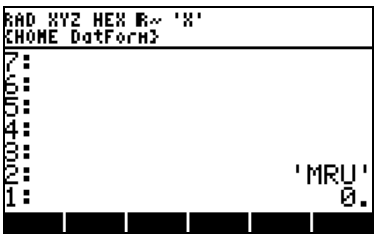
==



⑦ El programa coloca el número 15 en la pila y luego ejecuta el operador relacional **==**, este operador verifica si dos números son iguales. Si son iguales se obtiene 1. en caso contrario 0.

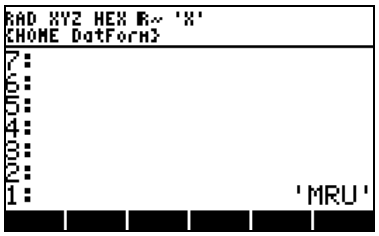


==



El número 15. representa a un objeto de tipo directorio.

⑧ En el programa se ejecutarán las instrucciones **IF THEN ELSE END.** Entre **IF** y **THEN** no se encuentra el valor de verdad, por lo que tomará el objeto que se encuentra antes de **IF** como valor de verdad y este valor es 0. (falso) por lo que el programa ejecutará las sentencias contenidas entre **ELSE** y **END**.
El programa ya utilizó el objeto 0. entonces en la pila queda solo el objeto 'MRU'



Cuando el valor de verdad es 1. (verdadero) el programa ejecutará las sentencias contenidas entre **THEN** y **ELSE** y estas sentencias son:
"YA EXISTE ESTE NOMBRE" **MSGBOX DROP**.

⑨El programa ejecuta el comando **DUP**, este comando duplica el objeto del nivel 1.

El número 15. representa a un objeto de tipo directorio.

⑩El programa ejecuta el comando **CRDIR**, este comando crea un directorio con el nombre indicado.

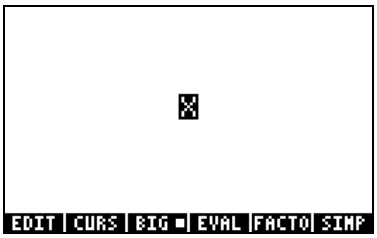
Se observa en el área de menús el directorio MRU.

⑪El programa ejecuta el comando **EVAL**, en este caso el comando ejecuta el contenido de la variable global, abriendo la carpeta con el nombre MRU

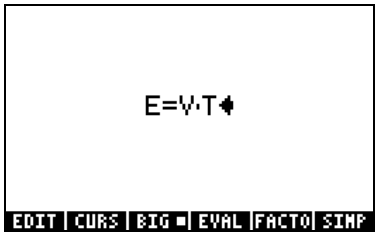
Se observa el área de la ruta del directorio actual y el directorio activo es MRU.

⑫El programa coloca en la pila la variable o nombre global 'X' y luego ejecuta el comando **EQW**, este comando abre en el editor de ecuaciones un objeto algebraico.

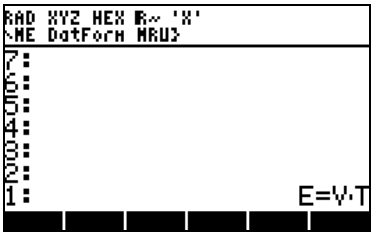
EQW



Luego se escribe la expresión de la fórmula y se presiona la tecla **ENTER**.



ENTER



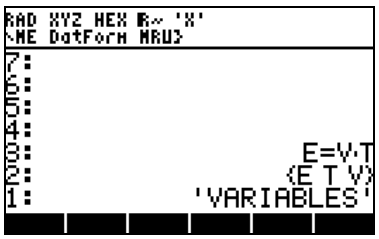
⑬El programa ejecuta el comando **LNAME**, este comando devuelve la expresión original y obtiene los nombres de las variables en un vector.

LNAME

⑭El programa ejecuta el comando **AXL**, en este caso el comando convierte un vector en una lista.

AXL

⑮El programa coloca el nombre global 'VARIABLES' y ejecuta el comando **STO**, este comando crea una variable almacenando un objeto ella.



STO

En la variable de nombre VARIABLES se encuentra almacenada la lista: {E V T}.

⑯El programa coloca el nombre global 'ECUACION' en la pila y ejecuta el comando **STO**, este comando crea una variable almacenando un objeto en ella.

STO

En la variable de nombre ECUACION se encuentra almacenada la ecuación: E=V.T

⑰El programa coloca en la pila la lista vacía { }, el número 1. y el contenido de la variable VARIABLES y luego ejecuta el comando **SIZE**, este comando obtiene la cantidad de objetos que tiene una lista, vector, cadena y matriz.

SIZE

⑱El programa ejecutará el proceso iterativo **FOR NEXT**, el número 1. será el primer valor que asumirá la variable temporal i(contador) y esta variable se incrementará de uno en uno hasta el valor 3.
Una vez que el programa empiece a iterar, el número 1., 3. y la variable i después de **FOR** serán utilizados por **FOR**, luego la pila quedará de la siguiente forma:

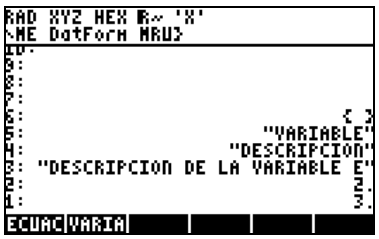
*) En la primera iteración el programa coloca las cadenas "VARIABLE", "DESCRIPCION" y "DESCRIPCION DE LA VARIABLE ", luego el contenido de la variable VARIABLES seguido de la variable temporal i=1. (esta variable es el contador del proceso iterativo **FOR NEXT**) y luego ejecuta el comando **GET**, en este caso este comando obtiene el objeto indicado de una lista.

GET

*) El programa ejecuta el operador **+**, en este caso el operador concatena la cadena con el nombre global.

+

*) El programa coloca en la pila los números 2 y 3, luego ejecuta el comando **→LIST**, este comando construye una lista.



→LIST

*) El programa coloca en la pila el número 1, luego ejecuta el comando **→LIST**, este comando construye una lista.

→LIST

*) El programa coloca en la pila las siguientes listas: { }, { } y { "" }, luego ejecuta el comando **INFORM**, este comando sirve para ingresar datos en la pila. Se observa dentro del tercer objeto de la pila el número 2. este valor indica que se debe ingresar una cadena.

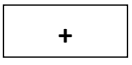
INFORM

Luego se ingresa una cadena que indique la descripción de la variable E, se ingresará la cadena "Espacio", luego se presiona la tecla ENTER o la tecla correspondiente al menú OK.

*) El programa ejecuta el operador lógico **NOT**, este operador obtiene lo contrario de un valor de verdad.

*) En el programa se ejecutarán las instrucciones **IF THEN END.** Entre **IF** y **THEN** no se encuentra el valor de verdad, por lo que tomará el objeto que se encuentra antes de **IF** como valor de verdad y este valor es 0. (falso) por lo que el programa no ejecutará las sentencias contenidas entre **THEN** y **END**. Luego en la pila ya no aparecerá el número 0.

Cuando el valor de verdad es 1. (verdadero) el programa ejecutará las sentencias contenidas entre **THEN** y **END** y estas sentencias son:
VARIABLES, i, **GET**, "" y **+**.
*) El programa ejecuta el operador **+**, en este caso este operador junta las dos listas que se encuentran en la pila



**) El programa vuelve a realizar las mismas instrucciones, con la diferencia del contador i, este contador tomará los valores de i=2 e i=3.
Luego de haber ingresado la descripción de las variables, para T=Tiempo y V=Velocidad la pila quedará de la siguiente forma:

⑲El programa coloca en la pila el nombre global 'DESCRIPCION' y ejecuta el comando **STO**, este comando crea una variable almacenando un objeto en ella

STO

⑳El programa ejecuta el comando **UPDIR**, este comando sube un nivel en el directorio.

UPDIR

El directorio antes de aplicar el comando es el directorio MRU, después de aplicar el comando el nuevo directorio será DatForm.

EJECUTAR: Este subprograma ejecutará una fórmula que se guardó en el directorio DatForm, usando el SOLVE EQUATION de la calculadora .

Antes de ejecutar el subprograma INGRESO, el programa principal ingresará al directorio DatForm.

```
«
15 TVARS DUP SIZE 0 ==
IF
THEN
DROP
      "NO HAY FORMULAS" MSGBOX
ELSE
```

```
        DUP "" ADD SWAP
        2 « 2 →LIST » DOLIST
        "EJECUTAR FORMULA" SWAP
        1
        CHOOSE
        IF
        THEN
                EVAL
                ECUACION STEQ
                #B4001h LIBEVAL
                UPDIR
        END
END
        »
```

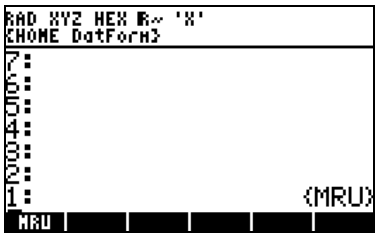
EXPLICACION:

① El programa coloca en la pila el número 15 luego ejecuta el comando **TVARS**, este comando obtiene en una lista los nombres de todos los objetos de un solo tipo indicado, contenidos en el directorio actual.

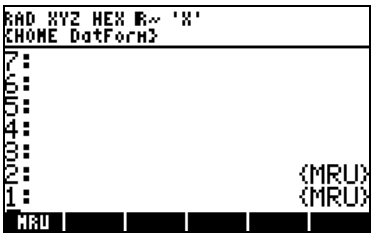
TVARS

El número 15. indica que se trata de objetos de tipo directorio.

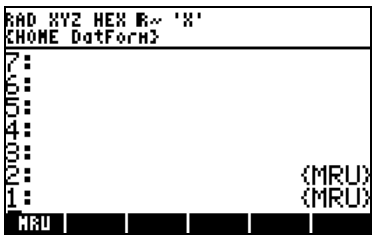
② El programa ejecuta el comando **DUP**, este comando duplica el objeto que se encuentra en el nivel 1.



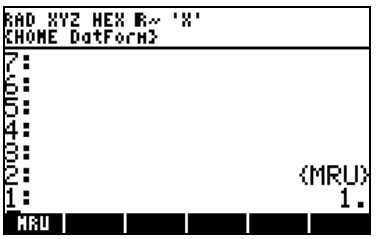
DUP



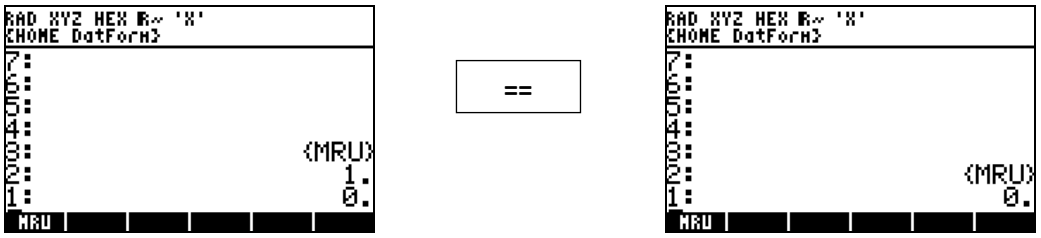
③ El programa ejecuta el comando **SIZE**, este comando obtiene el la dimensión de un objeto.



SIZE



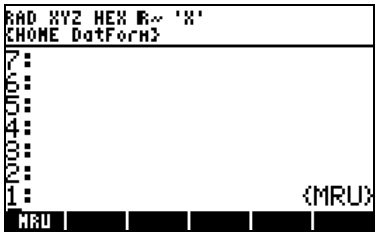
④ El programa coloca en la pila el número 0. y ejecuta el operador relacional **==**, este operador verifica si dos objetos son iguales.



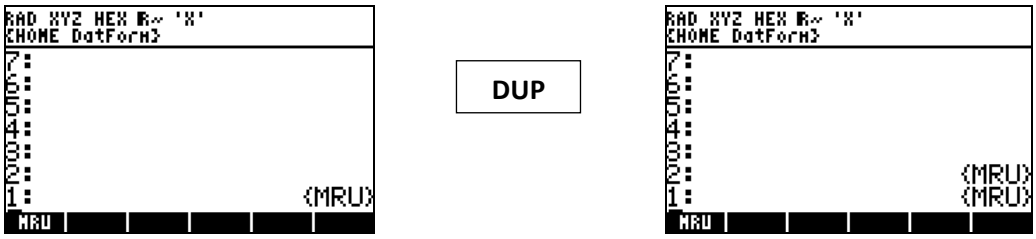
Para poder ingresar el operador **==**, se puede utilizar la aplicación **CHARS**.

⑤ Se analizarán las instrucciones que se encuentran entre **IF**, **THEN**, **ELSE** y **END**.

El valor de verdad no se encuentra entre **IF** y **THEN**, por lo que el programa asumirá como valor de verdad el objeto que se encuentre antes de la sentencia **IF**. El valor de verdad es 0. (falso), entonces el programa ejecutará las sentencias contenidas entre **ELSE** y **END**.
En la pila ya no se encuentra el número 0. quedando de la siguiente forma:

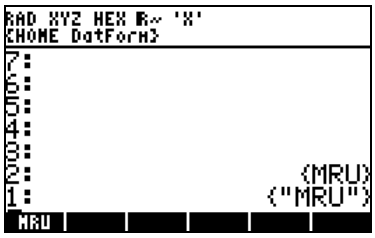


① El programa ejecuta el comando **DUP**, este comando duplica el objeto de nivel 1.

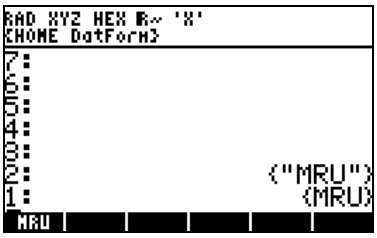


② El programa coloca en la pila la cadena vacía "" y luego ejecuta el comando **ADD**, en este caso este comando suma un objeto a todos los elementos de una lista.

③ El programa ejecuta el comando **SWAP**, este comando intercambia la posición de los dos objetos que se encuentran en los niveles 1 y 2.

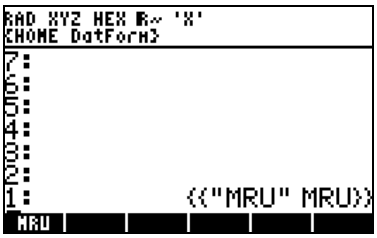


SWAP

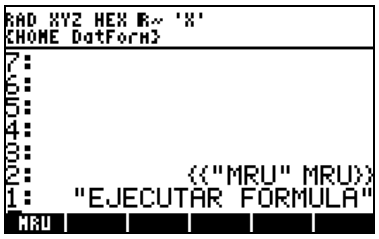


④ El programa coloca en la pila el número 2 y el programa « 2 **→LIST** » luego ejecuta el comando **DOLIST**, en este caso este comando ejecuta un programa sucesivamente a todos los elementos que tienen la misma posición de las dos listas. El número 2(del nivel 2) indica que se utilizará dos listas.

DOLIST



⑤ El programa coloca en la pila la cadena "EJECUTAR FORMULA" y ejecuta el comando **SWAP**, este comando intercambia la posición de los dos objetos que se encuentran en los niveles 1 y 2.

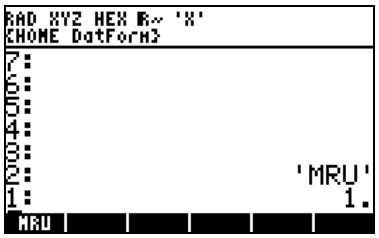


SWAP

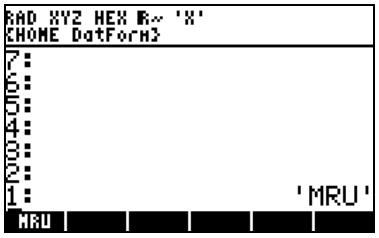
⑥ El programa coloca en la pila el número 1 y ejecuta el comando **CHOOSE**, este comando muestra un cuadro de opciones.

CHOOSE

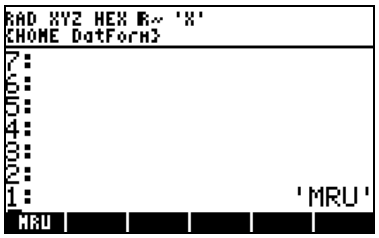
Luego de seleccionar la fórmula deseada (en este caso solo existe una fórmula MRU) se presiona la tecla correspondiente al menú OK o la tecla ENTER, obteniendo lo siguiente:



⑦ El programa ejecutará las sentencias **IF**, **THEN** y **END**. Como el valor de verdad es 1. (verdadero)el programa ejecutará las sentencias entre **THEN** y **END.**
En la pila ya no se encuentra el número 1. quedando lo siguiente:



*) El programa ejecuta el comando **EVAL**, en este caso el comando ejecuta el contenido de la variable global, abriendo la carpeta con el nombre MRU.



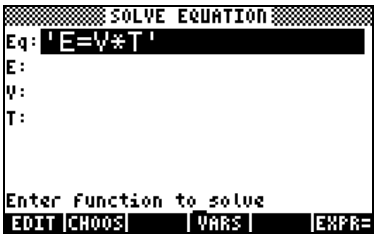
EVAL

*) El programa pone en la pila el contenido de la variable ECUACION y ejecuta el comando **STEQ**, este comando guarda en la variable EQ el contenido de una expresión.

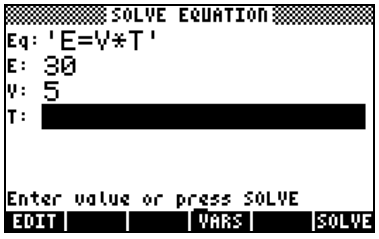
STEQ

*) El programa pone en la pila el número entero binario #B4001h y ejecuta el comando **LIBEVAL**, este comando abre la librería especificada. En este caso **LIBEVAL**, abre la librería SOLVE EQUATION.

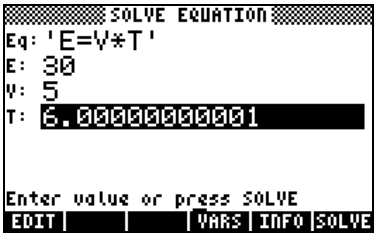
LIBEVAL



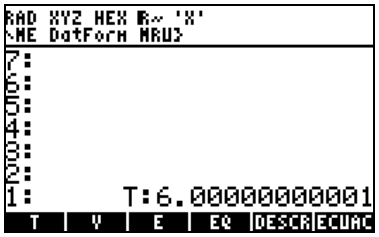
Luego se ingresa los datos conocidos en las casillas correspondientes, después se posiciona en la variable desconocida y se presiona la tecla del menú correspondiente a SOLVE



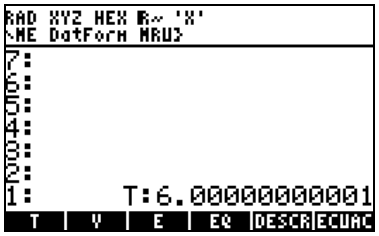
F6 F



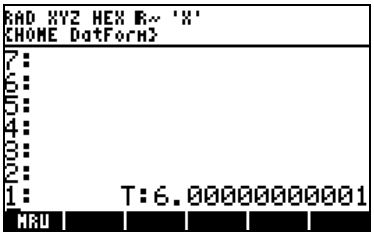
Luego se presiona la tecla ENTER o la tecla ON, obteniendo la siguiente:



*) El programa ejecuta el comando **UPDIR**, este comando sube un nivel en el directorio.



UPDIR



El directorio antes de aplicar el comando es el directorio MRU, después de aplicar el comando el nuevo directorio será DatForm.

VISUALIZAR: Este subprograma mostrará la descripción ingresada de las variables, de la ecuación.

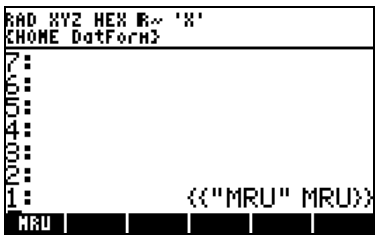
Antes de ejecutar el subprograma VISUALIZAR, el programa principal ingresará al directorio DatForm.

«
15 **TVARS DUP SIZE** 0 **==**

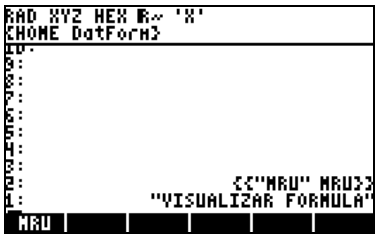
```
IF
THEN
        DROP
        "NO HAY FORMULAS" MSGBOX
ELSE
        DUP "" ADD SWAP
        2 « 2 →LIST » DOLIST
        "VISUALIZAR FORMULA" SWAP
        1
        CHOOSE
        IF
        THEN
            EVAL
            "" 1 VARIABLES SIZE
            FOR i
            VARIABLES " : " ADD i GET
            DESCRIPCION i GET
            +
            "

            "
            +
            +
            NEXT
            SCROLL
            UPDIR
        END
END
        »
```

EXPLICACION:

① Se observa que las sentencias hasta el comando **DOLIST**, son las mismas que en el subprograma EJECUTAR. Se continuará después del comando **DOLIST**. En la pila quedará la siguiente lista:



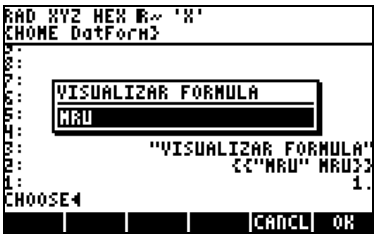
② El programa coloca en la pila la cadena "VISUALIZAR FORMULA" y ejecuta el comando **SWAP**, este comando intercambia la posición de los dos objetos que se encuentran en los niveles 1 y 2.



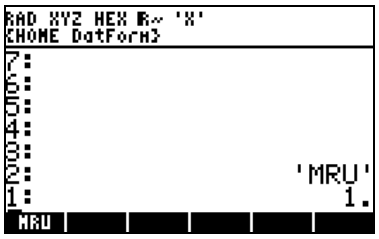
SWAP

③ El programa coloca en la pila el número 1 y ejecuta el comando **CHOOSE**, este comando muestra un cuadro de opciones.

CHOOSE

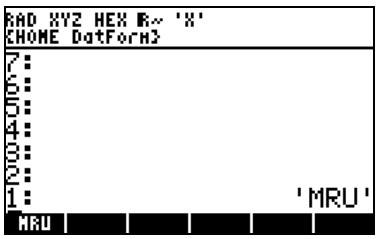


Luego de seleccionar la fórmula deseada (en este caso solo existe una fórmula MRU) se presiona la tecla correspondiente al menú OK o la tecla ENTER, obteniendo lo siguiente:



④ El programa ejecutará las sentencias **IF**, **THEN** y **END**. Como el valor de verdad es 1. (verdadero)el programa ejecutará las sentencias contenidas entre **THEN** y **END.**

En la pila ya no se encuentra el número 1. quedando lo siguiente:



① El programa ejecuta el comando **EVAL**, en este caso el comando ejecuta el contenido de la variable global, abriendo la carpeta con el nombre MRU.

EVAL

② El programa coloca en la pila la cadena vacía "", el número 1 y el contenido de la variable VARIABLES y luego ejecuta el comando **SIZE**, este comando devuelve la dimensión de un objeto.

SIZE

③ El programa ejecutará el proceso iterativo **FOR NEXT**, el número 1. será el primer valor que asumirá la variable temporal i(contador) y esta variable se incrementará de uno en uno hasta el valor 3.
Una vez que el programa empiece a iterar, el número 1., 3. y la variable i después de **FOR** serán utilizados por **FOR**, luego la pila quedará de la siguiente forma:

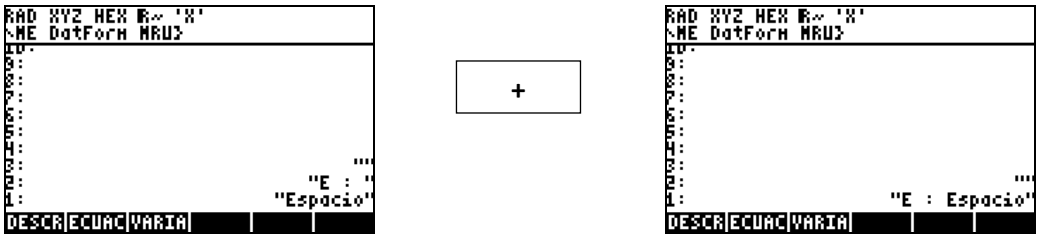
*) En la primera iteración el programa coloca el contenido de la variable VARIABLES, la cadena " : " y luego ejecuta el comando **ADD**, en este caso el comando concatena la cadena con todos los elementos de la lista almacenada en la variable VARIABLES.

ADD

*) El programa coloca en la pila el valor del contador i=1 y luego ejecuta el comando **GET**, este comando obtiene el elemento de posición indicada de una lista.

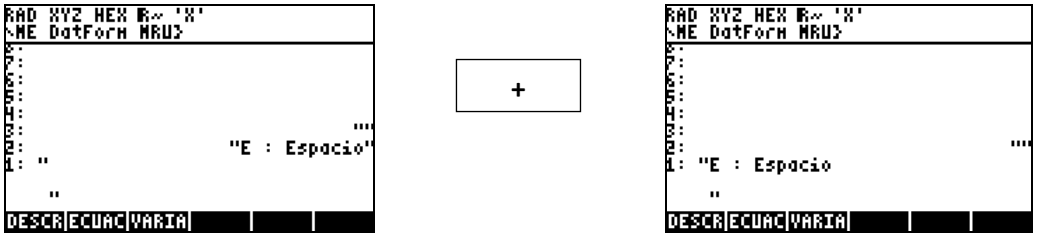
*) El programa coloca en la pila el contenido de la variable DESCRIPCION y el valor del contador i=1 y luego ejecuta el comando **GET**, este comando obtiene el elemento de posición indicada de una lista.

*) El programa ejecuta el operador **+**, en este caso el operador concatena las dos últimas cadenas de la pila.



*) El programa coloca en la pila la cadena:
"

", y luego ejecuta el operador **+**, en este caso el operador concatena las dos últimas cadenas de la pila.



*) El programa ejecuta el operador **+**, en este caso el operador concatena las dos últimas cadenas de la pila.

+

**) El programa vuelve a realizar las mismas instrucciones, con la diferencia del contador i, este contador tomará los valores de i=2 e i=3.
Luego de haber iterado tres veces en la pila quedará la siguiente cadena:

④ El programa ejecuta el comando **SCROLL**, este comando muestra un objeto en el formato más adecuado.

SCROLL

Luego de presionar la tecla correspondiente al menú OK o la tecla ENTER, la pila quedará vacía.

⑤ El programa ejecuta el comando **UPDIR**, este comando sube un nivel en el directorio.

UPDIR

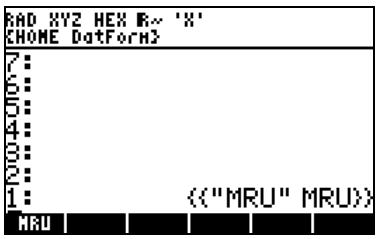
BORRAR: Este subprograma borrará la fórmula indicada.

Antes de ejecutar el subprograma VISUALIZAR, el programa principal ingresará al directorio DatForm.

```
    «
15 TVARS DUP SIZE 0 ==
IF
THEN
   DROP
   "NO HAY FORMULAS" MSGBOX
ELSE
   DUP "" ADD SWAP
   2 « 2 →LIST » DOLIST
   "ELIMINAR FORMULA" SWAP
   1
   CHOOSE
   IF
   THEN
      PGDIR
   END
END
    »
```

EXPLICACION:

① Se observa que las sentencias hasta el comando **DOLIST**, son las mismas que en el subprograma EJECUTAR. Se continuará después del comando **DOLIST**. En la pila quedará la siguiente lista:



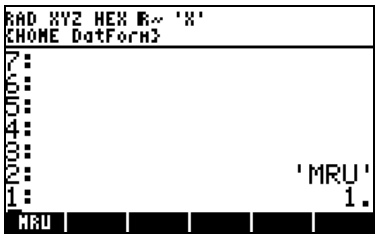
② El programa coloca en la pila la cadena "ELIMINAR FORMULA" y ejecuta el comando **SWAP**, este comando intercambia la posición de los dos objetos que se encuentran en los niveles 1 y 2.

SWAP

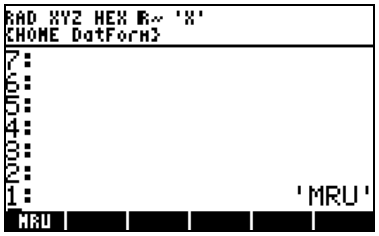
③ El programa coloca en la pila el número 1 y ejecuta el comando **CHOOSE**, este comando muestra un cuadro de opciones.

CHOOSE

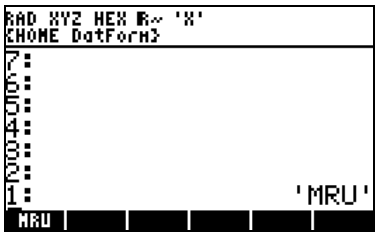
Solo existe la fórmula MRU se presiona la tecla correspondiente al menú OK o la tecla ENTER, obteniendo lo siguiente:



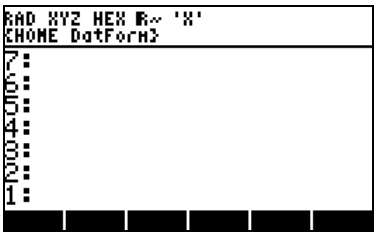
④ El programa ejecutará las sentencias **IF**, **THEN** y **END**. Como el valor de verdad que se encuentra antes de **IF** es 1. (verdadero) entonces el programa ejecutará las sentencias contenidas entre **THEN** y **END.**
En la pila ya no se encuentra el número 1. quedando lo siguiente:



① El programa ejecuta el comando **PGDIR**, este comando elimina un directorio contenido en el directorio actual.



PGDIR



FORMULARIO: Este es el programa principal y mostrará una ventana de selección, para seleccionar uno de los procedimientos que se encuentran en los subprogramas.

«
CLLCD

```
DO
        "FORMULARIO"
   {
   { "NUEVA FORMULA" INGRESO}
   { "EJECUTAR FORMULA" EJECUTAR}
   { "AYUDA FORMULA" VISUALIZAR }
   { "BORRAR FORMULA" BORRAR }
   { "SALIR" « HOME KILL » }
   }
   1
   CHOOSE
   IF
   THEN
        DIRECTORIO CLLCD EVAL CLLCD
    ELSE
        HOME KILL
    END
UNTIL
    0
END
        »
```

EXPLICACION:

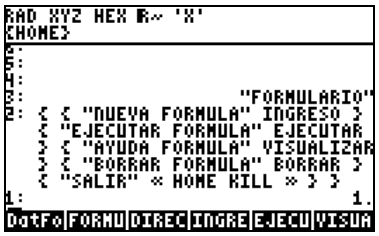
Solo se realizará un solo procedimiento y en el directorio DatForm existe dos fórmulas.

① El programa ejecuta el comando **CLLCD**, este comando limpia la pantalla.

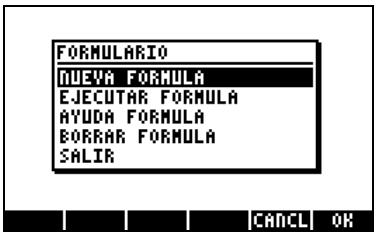
② El programa empezará un proceso iterativo con las instrucciones **DO**, **UNTIL** y **END**. El valor de verdad entre **UNTIL** y **END** es 0 (falso), por lo que el proceso iterativo no terminará hasta que el usuario lo cancele usando el comando **KILL**.

① El programa coloca en la pila la cadena "FORMULARIO", la lista
{
{ "NUEVA FORMULA" INGRESO}
{ "EJECUTAR FORMULA" EJECUTAR}
{ "AYUDA FORMULA" VISUALIZAR }
{ "BORRAR FORMULA" BORRAR }
{ "SALIR" « **HOME KILL** » }

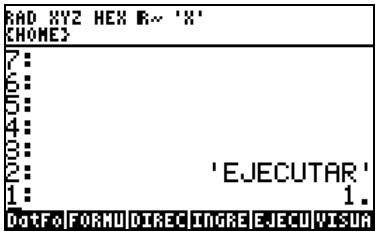
} y el número 1 y luego ejecuta el comando **CHOOSE**, este comando muestra un cuadro de opciones. En la pantalla no se observará ninguno de los objetos, pero se los mostrará para fines didácticos.



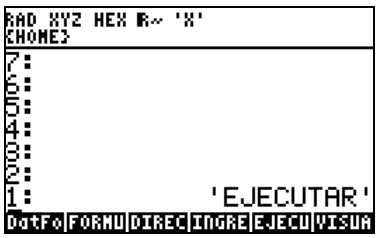
CHOOSE



Se seleccionará la opción "EJECUTAR FORMULA" y luego se presiona la tecla ENTER. Ahora se mostrará los objetos que se obtienen en la pila.



② El programa ejecutará las sentencias **IF**, **THEN**, **ELSE** y **END**. Como el valor de verdad que se encuentra antes de **IF** es 1. (verdadero) entonces el programa ejecutará las sentencias contenidas entre **THEN** y **ELSE.**
En la pila ya no se encuentra el número 1. quedando lo siguiente:



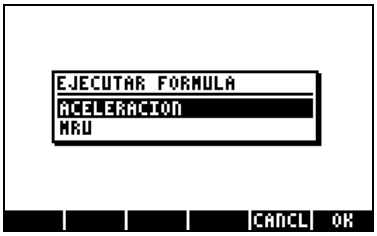
*) El programa ejecuta el subprograma DIRECTORIO, este subprograma crea o abre el directorio DatForm.

*) El programa ejecuta el comando **CLLCD**, este comando limpia la pantalla.

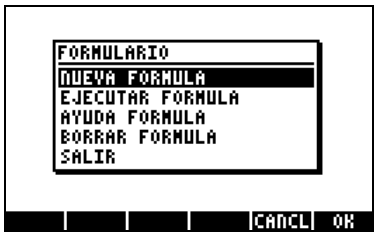
CLLCD

*) El programa ejecuta el comando **EVAL**, en este caso este comando evalúa el contenido de un nombre global.
Al evaluar el nombre global 'EJECUTAR' se ejecutará el subprograma EJECUTAR.

EVAL



Luego se selecciona la fórmula que se desea ejecutar y una vez terminado de usar la fórmula, el programa ejecuta el comando **CLLCD** y vuelve a regresar al cuadro principal como en el siguiente gráfico:



27 BIBLIOGRAFIA CONSULTADA

Según la relevancia.

1. **HP 48G Series** User's Guide
2. Guía de HP 49G Pocket
3. **hp** 49g+ calculadora gráfica **guía del usuario**
4. **hp** 50g calculadora gráfica **guía del usuario**