Programación en User-Rpl

Versión 1.8

Jorge Ayala Niño de Guzmán

Diciembre del 2000

Índice

I INTRODUCCIÓN

Programar en la 49G no es muy complicado, lo único que se debe saber es algo de lógica, diagramas de flujo y aprender los comandos de codificación que existen. El lenguaje de programación de la Hp es propio y similar a muchos de los lenguajes que existen como Visual Basic, Delphi y otros.

En la 49G y 48Gx existen tres tipos de lenguajes estos son: User-Rpl, Sys-Rpl, Asambler. Este texto desarrolla el lenguaje de alto nivel o User-Rpl al cual se accede fácilmente mediante los comandos de usuario(menú PRG). Por ser de alto nivel un programa requiere mas memoria y más tiempo para ejecutarse por las pruebas que realiza al revisar la existencia de algún error en el código y de esta manera prevenirlo; no sucede así en los lenguajes de bajo nivel(Sys-Rpl y Asambler) ya que estos no verifican si existen errores en el código, si hubiese alguno esto ocasionaría una falla en el sistema, perdida de la memoria o un daño en el procesador de la calculadora.

El problema que existe de incompatibilidad entre la 48Gx y 49G es debido a que las direcciones de memoria que utiliza la 49G son diferentes a los de la 48Gx; este problema solamente afecta a los programas creados en bajo nivel(User-Rpl y Asambler) y programas de alto nivel(User-Rpl) que están comprimidos, codificados o en librerías. Esta incompatibilidad se debe a que los programas de compresión, codificación y creación de librerías están hechas en bajo nivel. La solución a este problema para los programas(User-Rpl) es descodificarlos, descomprimirlos o convertirlos de librería a variables antes de transferirlos a la 49G, luego corregir o ajustar algunos detalles. Para convertir los programas(Sys-Rpl, Asambler) compatibles a la 49G, la cosa se complica, ya que debe conocerse bien este lenguaje, que consiste en cambiar todas las direcciones de los comandos de memoria para la 49G mediante tablas(Entry Points).

El contenido de este texto de programación en User-Rpl es similar tanto para la 49G como para la 48Gx excepto algunos comandos que son exclusivamente para la 49G

Versiones actualizadas de este texto, programas, tutores y muchas otras cosas. Puede bajarlos visitando la pagina de la **49G Civil** en Internet cuya dirección es **www.angelfire.com/co/48gx/49g.html** y cualquier duda, comentario, sugerencia al correo electrónico hp_club@mixmail.com

Jorge Ayala N. de G.

2 VARIABLES

Son valores asignados en la memoria con un nombre en un programa, que se requieren constantemente para ser operados.
Esta parte es muy importante en la programación, y se debe dominar la forma conveniente de almacenar y utilizar las variables.

Existen dos maneras de almacenar variables, que son las siguientes;

2.1 VARIABLES LOCALES

Estas variables son almacenadas en la memoria temporal durante la ejecución de un programa, esta forma de guardar es recomendable ya que se utiliza menos memoria y son más rápidos al ejecutarse.

Ejemplo

En este programa los valores son almacenados en las variables **a** y **b,** que se encuentran en el nivel 1 y 2

```
<<\ → a b
    <<\   a b + 6 /
          'a' STO b 2 * a /
    \>>
\>>
```

Sintaxis :

1- Primeramente se deben encontrar los valores en los niveles o antes del símbolo(→).
2- Símbolo(→).
3- Nombre de las variables.
4- Subprograma que utilizara estas variables. (**<<** ..... **\>>**)
5- ' *Variable* ' STO: esto permite remplazar un nuevo valor a una variable

Lo que se debe tener presente en este tipo de variables es que solamente estarán disponibles en un subprograma asignado, luego de haber concluido el subprograma las variables ya no existen.

Ejemplo

```
<<\ → a          Almacenamiento del valor  en la variable a
    <<\  a 2 /   Subprograma que utiliza la variable a para dividirlo entre 2
    \>>          Fin del subprograma
   a 1 -         Esta operación ya no podrá llevarse acabo porque ya no existe a.
\>>
```

2.2 Variables Globales

Estos son los que se restauran en la memoria del menú VAR, y pueden ser recuperadas luego que el programa haya termina.

Se debe tener cuidado en usar esta clase de variables, ya que pueden haber programas o librerías con el mismo nombre de la variable, esto puede ocasionar errores en un programa.

Ejemplo

Este programa guarda el valor de 0 en una variable cuyo nombre es a.

```
<<\   0 'a' STO     Se almacena el valor 0 en a
      a 1 +         Realiza la operación a + 1
      'a' STO       Guarda el nuevo valor.
\>>
```

Sintaxis :

1- Primeramente se debe encontrar el valor en el nivel o antes del nombre.
2- Nombre de la variable, con el símbolo (')
3- Comando STO
6- ' *Variable* ' STO: esto permite remplazar un nuevo valor a una variable.

Para almacenar mas de una variable, una forma fácil es poner los valores en una lista, luego hacer lo mismo con los nombres y guardarlas

Ejemplo

```
<<\ { 9 20 8 1 } { a b c d } STO \>>
```

3 Programación Básica

Cualquier calculadora programable tiene esta herramienta de poder introducir una ecuación, sus valores y obtener sus valores. En esta parte se vera como crea este tipo de programas, y en forma RPN

Ejemplo 1

En este ejemplo se desea hacer un programa para obtener el valor de la ecuación:

Código Hp

```
<<\ → x
     <<\ 5 x * 2 x + √ *
          x Cos
          /
          4 x 3 ^ *
          + \>>
\>>
```

$$\frac{5x\sqrt{2+x}}{Cos(x)} + 4x^3$$

Ejemplo 2

En este ejemplo se utilizara la ecuación de 2do grado para obtener los valores x

Código Hp

```
<<\ → a b c
   <<\ b NEG
       b 2 ^ 4 a c * * - √
       +
       2 a *
         /
       b NEG
       b 2 ^ 4 a c * * - √
       -
       2 a *
         /   \>>
\>>
```

$$\frac{-b+\sqrt{b^2-4ac}}{2a}$$

$$\frac{-b-\sqrt{b^2-4ac}}{2a}$$

4 Comandos del menú PRG

En esta sección se vera detalladamente el uso de cada comando que se encuentran en el directorio PRG

4.1 Comandos de la sección Type

Estos comandos ayudan a combinar distintos tipos de objetos existentes.
Antes de entrar a estos comandos se deben conocer los diferentes tipos de objetos que existen. La siguiente tabla describe algunos de estos. (*vea también guía de bolsillo de la hp49 Pagina 80*)

Tipo	Descripción	Ejemplo
0	Número Real	1.23546
1	Número Complejo	(1,5)
2	Cadena (String)	"Umss"
3	Arreglo (Vector o matriz)	[2 3], [[1 2] [5 3]]
4	Arreglo números complejos	[(1,2) (2,5)]
5	Listas	{ "Civil" 657 }
6	Nombre	'X'
7	Variable en uso	'i' *<< FOR i 1 + NEXT >>*
8	Programa	<< B 2 / >>
9	Algebraico(Ecuación)	'X+5'
10	Número Binario	#05B15hd
11	Grob(gráficos)	Graphic 131 x 64
12	Objeto Etiquetado	Datos : 5354
13	Unidades	25_Km
14	Nombre de XLIB	XLIB 755 1
15	Directorio	DIR A 5 B 2 ..... END
16	Librería	Library 1687 : MetNum
19	Comandos de la HP	IF, FOR, START, NUM
21	Número Real Extendido	1.2E1
25	Objeto Codificado	Code
26	Datos de librería	Library Data
28	Números enteros	25
30	Fuentes	Ft8_25 : Wilancha

Tabla 3.1 Tipos de Objetos

TYPE : Este comando devuelve el numero del tipo de objeto que se encuentra.

VTYPE : Este comando devuelve el numero del tipo de objeto que se encuentra en una variable, para esto se necesita el nombre de la variable.

En la variable DATOS esta almacenada una lista

→ARRY : Construcción de arreglos(vectores o matrices)
Vectores elementos del vector y su dimensión
Matrices los elementos de la matriz ordenada en filas y columnas en una lista {#fila #columna}

→LIST : Construcción de listas, los elementos de las listas pueden ser cualquier tipo de objetos, se requiere la cantidad de elementos que contendra.

→STR : Convierte en cadena cualquier objeto

→TAG : Etiquetador de valores, primero debe encontrarse el valor y a continuación el nombre con el que se etiquetara.

→UNIT : Herramienta de la seccion de unidades, este coloca a un número la unidad que se desea.

OBJ→ : Este comando es lo inverso a los cinco últimos comandos vistos.

C→R : Descompone un número complejo, en su parte imaginaria y real. (o un sistema dimensional)

R→C : Crea un número complejo teniendo el valor real y el valor imaginario. o un sistema dimensional.

NUM : Devuelve el numero del codigo de un carácter.

CHR : Retorna el caracter(simbolo) de un numero

DTAG : Devuelve el valor de un objeto etiquetado.

EQ→ : Descompone una ecuación apartir del signo de igualdad.

4.2 Comandos de la sección MEM

MEM : Devuelve la cantidad de espacio libre que existe en el puerto 0
BYTES : Determina el tamaño y Checksun de cualquier objeto. →0
NEWOBJ : Crea un nuevo objeto →1
ARCHIVE : Crea un backup de todo lo que se encuentra en home
RESTORE : Restaura la copia backup en home
} →2

4.2.1 Comandos de la sección DIR

PURGE : Borra una variable o un directorio.
RCL : Recupera el objeto de una variable.
STO : Guarda una variable o directorio
CRDIR : Crea un nuevo directorio
PGDIR : Borra un directorio
} →1

PATH : Devuelve en una lista la ubicación actual
VARS : Devuelve en una lista las variables del menú actual
TVARS : Devuelve en una lista las variables de un tipo requerido →0
ORDER : Ordena las variables según el orden que se encuentran en una list →4

4.2.2 Comandos de la sección ARITH

STO+ : Restaura sumando un valor al anterior valor
STO- : Restaura restando un valor al anterior valor
STO* : Restaura multiplicando un valor al anterior valor
STO/ : Restaura dividiendo un valor al anterior valor
} →3

SINV : Restaura el inverso del valor anterior.
SNEG : Restaura el valor anterior cambiado de signo
SCONJ : Reúne un objeto a otra variable (Conjuga)
} →1

INCR : Restaura el valor anterior incrementando 1 y devuelve el valor actual
DECR : Restaura el valor anterior des incrementando y devuelve el valor actual
} →1

	Sintaxis		
0	Objeto	2	:Puerto: Nombre
1	Nombre de la variable	3	Número y nombre de variable
		4	Lista de Variables

4.3 Comandos de la sección STACK

Estas herramientas se utilizan para copiar, eliminar, cambiar posiciones y muchas otras cosas mas a los objetos que se encuentran en los niveles

DUP : Duplica el objeto del nivel 1

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

DROP : Elimina el objeto que se encuentra en el nivel 1

OVER : Copia al nivel 1 el objeto que se encuentraba en el nivel 2

ROT : Cambia la posición de los objetos que se encuentran en los primeros tres niveles de abajo hacia arriba.

UNROT : Cambia la posición de los objetos que se encuentran en los primeros tres niveles de arriba hacia abajo.

ROLL : Cambia la posición de los objetos que se encuentran en n niveles de abajo hacia arriba, este requiere el número de niveles.

ROLLD : Cambia la posición de los objetos que se encuentran en n niveles de abajo hacia arriba, este requiere el número de niveles.

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

UNPICK : Elimina el objeto del nivel deseado, este requiere el número de nivel que se desea eliminar.

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

DEPTH : Devuelve la cantidad de niveles que estan en uso.

DUP2 : Duplica los objetos de los niveles 1 y 2

DUPN : Duplica los objetos que se encuentran en n niveles, este requiere el número de los niveles que se desea copiar.

DROP2 : Elimina los objetos de los niveles 1 y 2.

DROPN : Elimina los objetos que se encuentran en n niveles, este requiere el número de niveles que se desea eliminar.

DUPDUP: Triplica el objeto que se encuntra en el nivel 1

NIP : Elimina el objeto que se encuentra en el nivel 2

NDUPN : Copia n veces el objeto que se encuentra en el nivel 1 y devuelve el número de copias, este requiere el número de veces a copiar.

4.4 Comandos de la sección RUN

Estos comandos son utilizados para verificar si existe algún error en el procedimiento de un programa ya que estos nos permiten ver la ejecución de un programa paso a paso.

DBUG : Este activa a un programa al modo RUN para ser ejecutado paso a paso, este programa puede estar en el nivel 1 o en una variable escribiendo el nombre de la variable.

SST : Este empieza a ejecutar el programa paso a paso, en la parte superior de la pantalla se observa el comando que se lleva a cabo, para continuar se debe presionar este nuevamente.

SST→ : Similar a SST

NEXT : Este muestra los dos siguientes comandos a ejecutarse en el programa.

HALT : Este interrumpe la ejecución de un programa similar al comando PROMPT.

KILL : Este comando interrumpe el modo RUN.

OFF : Este comando puede ser utilizado dentro de un programa para apagar la calculadora.

Ejemplo

En este ejemplo sé ejecutara un programa paso a paso, que este muestra el tamaño de un directorio que se encuentra en HOME.

Primeramente se debe copiar el código del programa:

```
<<\
  2 FIX HOME 15 TVARS
   "Memoria en Directorio" SWAP
    1 CHOOSE DROP DUP RCL
    BYTES 1024 / SWAP DROP "Kb" +
    SWAP " " + SWAP +
    MSGBOX STD
\>>
```

Luego de haber copiado el código, el programa se debe encontrar en el nivel 1 se lo lleva al modo RUN presionando DBUG, al efectuar esta operación el programa ya no estará en el nivel 1, y en la parte superior de la pantalla sé veraHLT(*esto indica que el programa esta en modo RUN o fue interrumpido*)

Se deberá presionar SST para ver paso a paso todos los comandos que utiliza dicho programa hasta terminar, para finalizar o abortar el modo RUN se debe presionar KILL

4.5 Comandos de la sección Test

En esta sección sé vera las herramientas que sirven para comparar: números, hacer pruebas lógicas y el modo de utilizar dentro de un programa los indicadores del sistema.

Comparaciones numéricas

En cualquiera de estas operaciones al efectuar la comparación si fuera verdad el sistema devuleve el número 1, y si fuera falsa el número 0. Estos operadores matemáticos son : ==, ≠, <, >, ≤ y ≥

Pruebas lógicas

Estas realizan pruebas lógicas compuestas.

AND : Este comando equivale al símbolo lógico conocido **(^)**
OR : Este comando equivale al símbolo lógico conocido **(v)**
XOR : Comando lógico

Prueba		AND	OR	XOR
1	1	1	1	0
0	0	0	0	0
1	0	0	1	1

Tabla que muestra las pruebas de los comandos AND, OR y XOR

NOT : Este devuelve lo inverso si es verdad o falso
SAME : Comando que compara dos objetos si son iguales

Indicadores del sistema(banderas)

Es la configuracion de la calculadora, para utilizar estos en un programa primeramente se debe conocer los indicadores(*Ver guia de bolsillo 49G Pag 76 - 79*)

SF : Activa el indicador seleccionado del sistema
CF : Desactiva el indicador seleccionado del sistema
FS? : Verifica si un indicador esta activado
FC? : Verifica si un indicador esta desactivado
FS?C : Verifica si un indicador esta activado, y luego lo desactiva
FC?C : Verifica si un indicador esta desactivado, y luego lo desactiva

Ejemplo

- Se desea desactivar el reloj que esta situado en la parte superior de la pantalla.

Para esto se debe conocer el número del indicador del reloj que es–40, se escribe este número y a continuación el comando**CF**.

- Se quiere verificar si los argumentos simbólicos devuelven números, si no lo es activarlo.

Sabemos que el indicador para este ejemplo es el número–3, primeramente se verifica si esta activado con el número del indicador y el comando**FS?**, si este no estuviera activado la maquina devuelve el número 0, luego se procedera a activarlo con **SF** como el ejemplo anterior.

LININ : Este comando verifica si una ecuación es lineal, este requiere la ecuación y la variable de la ecuación.

4.6 Comandos de la sección MODES

Estos sirven para hacer cambios en el sistema desde un programa permitiendo cambiar el formato numérico(FMT), medida de ángulo(ANGLE), indicadores de sistema(FLAG), teclas de usuario(KEYS) y menús de acceso(MENU).

4.6.1 FMT

STD : Cambia a formato standard
FIX : Cambia a formato Fix, requiere un número antes
SCI : Cambia a formato Científico, requiere un número antes
ENG : Cambia a formato Ingeniería, requiere un número antes

4.6.2 ANGLE

DEG : Cambia a formato Deg
RAD : Cambia a formato Rad
GRAD : Cambia a formato Grad
RECT : Cambia a formato Rectangular
CYLIN : Cambia a formato Cilíndrico
SPHERE : Cambia a formato Esférico

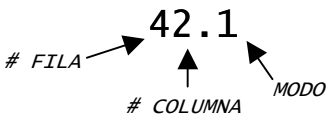
4.6.3 FLAG

STOF : Restaura la configuración de los indicadores del sistema
RCLF : Devuelve la configuración de los indicadores del sistema

4.6.4 KEYS

ASN : Restaura una tecla de usuario
STOKEYS : Restaura la configuración de teclas de usuario
RCLKEYS : Devuelve la configuración de teclas de usuario
DELKEYS : Borra una tecla de usuario

Formato de key :



MODOs

No	Tecla	No	Tecla
1	Ningún	4	Alpha
2	Azul (izquierda)	5	Alpha + Azul
3	Rojo (derecha)	6	Alpha +Rojo

4.6.5 MENU

MENU : Devuelve el menú llamado.
CST : Menú personalizado de acceso directo
TMENU : Menú temporal
RCLMENU : Devuelve el número del menú

Formato de menú : { {"*NOMBRE o GRAFICO 21x8* " *OBJETO* } {.... }..... n }

4.7 Comandos de la sección OUT

Estos comandos sirven para visualizar resultados y mensajes con sonidos de un programa.

MSGBOX: Muestra un mensaje en recuadro en la parte central de la pantalla.

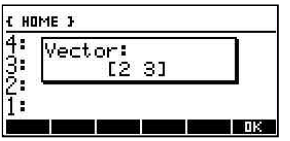
Sintaxis :

1 **Mensaje** debe estar en cadena.
2 **MSGBOX**

Ejemplo

En este ejemplo se desea visualizar en un recuadro un vector que se encuentra en el nivel 1.

```
<<\ →STR
  "Vector :
        " SWAP + MSGBOX \>>
```



CLLCD : Limpia la pantalla modo texto.

DISP : Muestra un objeto en una linea de la pantalla.

Sintaxis :

1 **Objeto** lo que se visualizara en la pantalla
2 **Número** de la linea en el que se desea visualizar en la pantalla (1 - 7)
3 **DISP**

Ejemplo

En este ejemplo se desea visualizar los valores que se encuentran en los niveles 1, 2 y 3 que son A, B y C respectivamente.

```
<<\ "C: " SWAP + SWAP
 "B: " SWAP + ROT
 "A: " SWAP + UNROT
 SWAP CLLCD 5 DISP 4
 DISP 3 DISP "Datos"
 1 DISP 0 WAIT DROP \>>
```

BEEP : Emite un sonido mediante frecuencia. y un determinado tiempo.

- **Sintaxis :**

1 **Frecuencia** esta dada en Hz
2 **Tiempo** de duración esta dada en segundos
3 **BEEP**

Ejemplo

En estos dos ejemplos escucharan los sonidos de error, y sonido de error de menú del sistema.

```
<<\  " Sonido de error " MSGBOX
      1392  0.075  BEEP  \>>

<<\  " Sonido de error  de menú" MSGBOX
      337  0.07  BEEP  \>>
```

Se debe tomar en cuenta que el Flag 57 no debe estar activado, para que se pueda escuchar el tono.

FREEZE : Congela una parte especifica de la pantalla.

- **Sintaxis :**

1 **Parte** de la pantalla: 1=área de estado; 2=pila o linea de comandos; 4=área menú
2 **FREEZE**

PVIEW : Cambia a la pantalla modo gráfico.

- **Sintaxis :**

1 **Tipo** de cambio: {}=cambio permanente; {#0 #0}=visualiza por un momento.
2 **PVIEW**

TEXT : Cambia a la pantalla modo texto.

4.8 Comandos de la sección IN

En esta parte se vera todos los comandos de introducción de datos para los programas.

INPUT : Este comando es utilizado para introducir datos.

Sintaxis :

1 **Mensaje** debe estar en cadena.

2 **Formato** {"VALOR INICIAL" {#FILA #COLUMNA } *MODO(S)* } o "VALOR INICIAL"

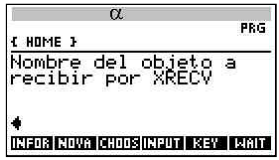
MODOS
- **V** : Verifica si existe algun error en el sintaxis
- **ALG** : Este activa el modo algebraico para introducir ecuaciones
- α : Activa alpha

3 **INPUT**

Ejemplo

Este programa permite recibir datos mediante el comando XRECV, escribiendo el nombre en el que se almacenara el archivo recibido.

```
<<\  "Nombre del objeto a
 recibir por XRECV "
 {" " {1 2} α }
 INPUT OBJ→ XRECV \>>
```



NOVAL : Este es un campo vacio que es utilizado junto al comando INFORM

INFORM : Este permite diseñar formularios para la introducción de datos. En el nivel 2 devuelve los valores almacenados en una lista, y en el nivel 1 un valor que puede ser 1 si el formulario fue llenado correctamente o el valor 0 en caso que se haya cancelado el formulario manualmente o por un error.

Sintaxis :

1 **Nombre** que lleva el formulario, el nombre debe estar en cadena.
2 **Label** son los elementos que existen
{ {"*NOMBRE*" "*AYUDA*" *TIPO DE OBJETO(S)* } {} . . . n }
3 **Número** de columnas, #COLUMNAS, { #COLUMNAS}, {#COLUMNAS #ESPACIO }
4 **Valores Reset** estos son los valores que se restauran cuando se presiona la opcion reset del menu del formulario{*VALORES RESET*}
5 **Valores Iniciales** estos son los valores de iniciales {*VALORES INICIALES*}
6 **INFORM**

Ejemplo

Este programa realiza un formulario que permite la entrada de una función, el valor de x y calcularlo.

```
<<\ "Introducción de datos"
  { { "F(X):" "Introdusca función" 9 }
  { "  X:" "Valor para X" 0 } }
  { 1 -1 } { } { } INFORM DROP OBJ→
  DROP X SWAP = SUBST →NUM
  "F(X)" →TAG →STR MSGBOX
\>>
```

CHOOSE : Crea un recuadro de selección de datos por el usuario en la parte central de la pantalla. Si de este es seleccionado alguna opción devuelve el objeto que contiene dicho y el número 1, si no se selecciona ninguna opción devuelve 0

Sintaxis :

1 **Nombre** este debe estar en cadena
2 **Items** opciones de selección { {"*NOMBRE*" *OBJETO* } { }... n }
3 **Posicion Inicial** número de un item que estara sombreado
4 **CHOOSE**

Ejemplo

Este ejemplo da a escoger tres opciones que puede se puede efectuar con una función, la primera puede derivarla, la segunda puede integrarla o hallar el valor de la función remplazando un valor para x.


```
<<\  "Introdusca Función"
{ "  ' '  "  {1 2} ALG}  INPUT  OBJ→
"Funcion" { {"Derivar"  DERVX }
 {"Integrar"  INTVX  }
 { "Reemplazar Valor"  <<\ "Valor para X"  " "
 INPUT  OBJ→  'X'  SWAP  =  SUBST →NUM  \>>  } }
1 CHOOSE DROP EVAL \>>
```

WAIT : Este comando interrumpe un determinado tiempo o hasta que se presione una tecla.

Sintaxis :

1 **Número** del tiempo de espera, si no hay ninguno este espera una tecla.
2 **WAIT**

El número de tecla que devuelve este comando es el formato explicado en la sección de MODES / KEYS en la página 20.

PROMPT : Comando que interrumpe la ejecución de un programa y muestra un mensaje en la parte superior.

Sintaxis :

1 **Mensaje** este debe estar en cadena
2 **PROMPT**
3 **CONT** este comando sirve para continuar el programa

Ejemplo

Este ejemplo permite calcular la longitud, área y inercia de una circunferencia introduciendo como dato su radio. Este ejemplo se debe guardar en la variable P6

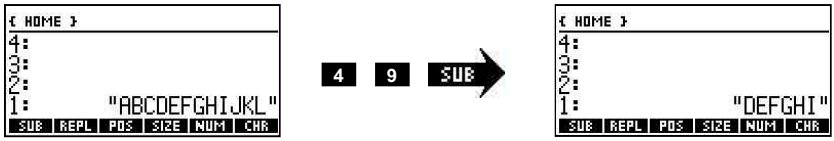

```
<<\  "Introdusca Radio
 de la circunferencia" { ":Radio: "  { 1 9 } V }  INPUT
   OBJ→ DTAG → R
      <<\ { {"Longi" <<\  2  PI  R  *  *  →NUM  "Longitud"  →TAG  \>> }
            {"Area"  <<\  R  2  ^  PI  *  →NUM  "Area"  →TAG  \>>}
            {"Inercia"  <<\  PI  R  4  ^  *  4  /  →NUM  "Inercia"  →TAG  \>>}
            {} { "Nuevo"  <<\  0  MENU  P6  CONT  \>> }
            {"Cancl"  <<\  0  MENU  CONT  \>> } } TMENU
          "Elija del menu"
          PROMPT  \>>  \>>
```

KEY : Comando que devuelve 1 y la ubicación del número de tecla o devuelve 0 cuando no se presiona ninguna tecla.

4.9 Comandos de la sección CHARS

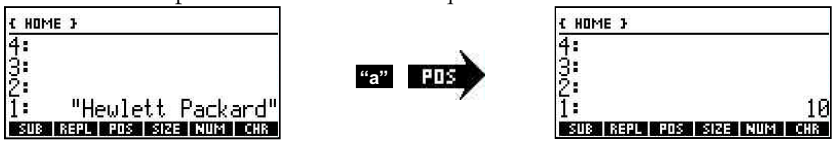
Estos comandos sirven para modificar los caracteres de una cadena(string) desde un programa.

SUB : Este devuelve una parte de una cadena mediante dos valores que son la posición inicial y la final.



REPL : Reemplaza una cadena de caracteres en otra cadena mediante el valor de la posición.

POS : Indica la posición de un carácter que se encuentra en una cadena.



SIZE : Devuelve la cantidad de caracteres que contiene una cadena.

HEAD : Devuelve el primer caracter de una cadena

TAIL : Devuelve todos los caracteres excepto el primero de una cadena.

4.10 Comandos de la sección BRCH

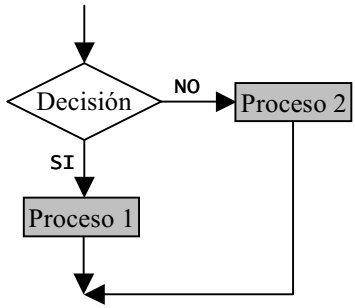
Los siguientes comandos a continuación son los más importantes y es la base de este texto de programación, en cada comando sé vera la relación con un diagrama de flujo para su mejor compresión.

4.10.1 IF

Este es un comando de decisión, comprueba dos valores y si la comparación es verdadera ejecuta una serie de instrucciones y si es falsa de igual manera se ejecuta otras instrucciones.

Significados

IF : Comparación(comando que compara)
THEN : Entonces (verdadero → 1)
ELSE : De otra manera (falso →0)
END : Fin (finaliza el comando IF)



Sintaxis 1 :

Este es cuando se tiene dos procedimientos uno cuando sea verdadero y el otro en caso que fuera falsa.

IF Comparación o número (1 o 0) SI THEN Proceso 1 NO ELSE Proceso 2 END

Sintaxis 2 :

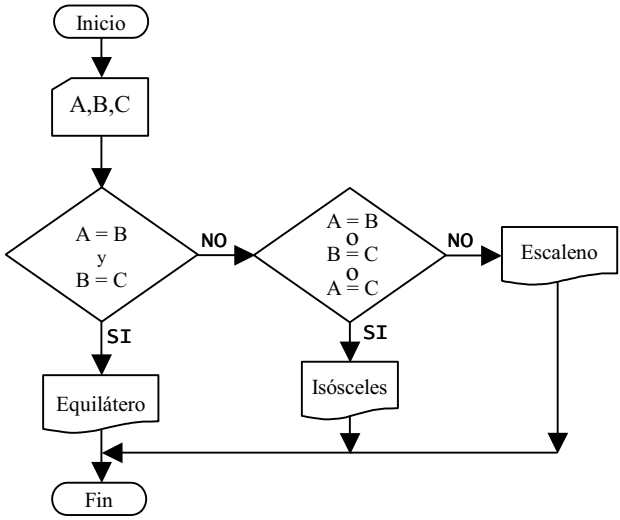
En este caso solamente se efectuara un procedimiento si fuera verdad y si no fuese a así no realizaría ningún procedimiento.

IF Comparación o número (1 o 0) SI THEN Proceso END

Ejemplo

Este programa determina si un triángulo es equilátero, isósceles o escaleno introduciendo como datos sus lados.

Diagrama de Flujo

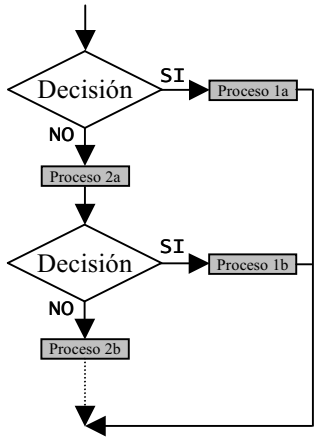


Código HP

```
<<\  "Introdusca los lados
      de un trinagulo"  {" : A :
                        : B :
                        : C : "  {1 4} V }  INPUT
     OBJ→  DTAG   ROT  DTAG  ROT   DTAG   ROT
      →  A  B  C
     <<\  IF  A  B  ==  B  C  ==    AND
          THEN "Equilátero"  MSGBOX
          ELSE
             A   B == B  C  ==    A    C  == OR  OR
               IF  THEN  "Isoceles"  MSGBOX
                    ELSE
                     "Escaleno"  MSGBOX
               END
       END
     \>>
\>>
```

4.10.2 CASE

Este comando es similar al anterior comando *IF*, este se utiliza mayormente cuando se tiene muchas alternativas de comparación.



Significados

CASE : Condición *(inicializa el comando Case)*
THEN : Entonces *(sí es verdad se ejecuta la instrucción)*
END : Fin *(finaliza la instrucción)*
END : Fin *(finaliza el comando Case)*

Sintaxis :

CASE
Comparación o número (1 o 0) THEN Proceso 1a END
Proceso 2a
Comparación o número (1 o 0) THEN Proceso 2b END
Proceso 2b
:
END

Ejemplo

Este programa determina si un número es múltiplo de 2, 3, 5, 7 o 11

```
<<\  "Introdusca número"
       " "  INPUT  OBJ→
 → N  <<\ CASE
              N  2  /  IP  2  *  N  ==
          THEN  "Multiplo 2"  MSGBOX  END
              N  3  /  IP  3  *  N  ==
          THEN  "Multiplo 3"  MSGBOX  END
              N  5  /  IP  5  *  N  ==
          THEN  "Multiplo 5"  MSGBOX  END
              N  7  /  IP  7  *  N  ==
          THEN  "Multiplo 7"  MSGBOX  END
              N  11  /  IP  11  *  N  ==
          THEN  "Multiplo 11"  MSGBOX  END
        END  \>>
\>>
```

Código HP

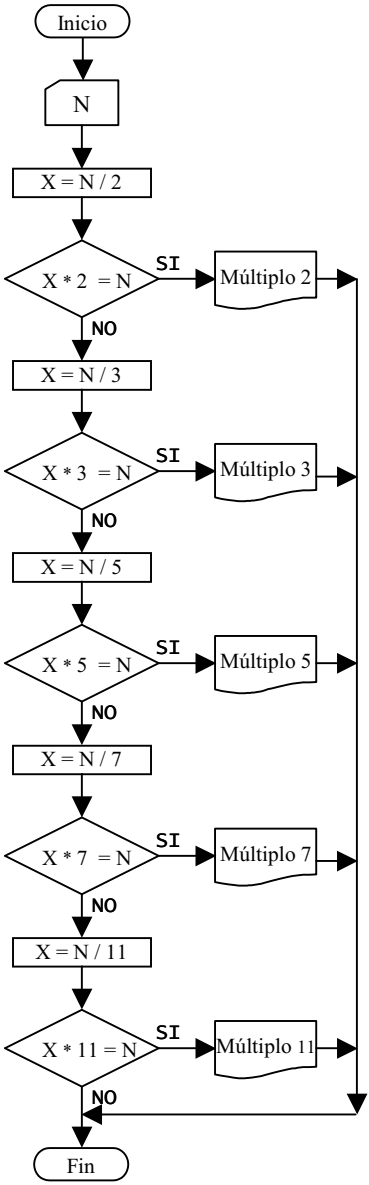


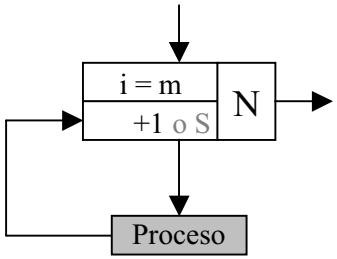
Diagrama de Flujo

4.10.3 FOR

Este es utilizado para elaborar procesos iterativos dentro de un programa, comenzando en valor inicial hasta llegar al valor final, de acuerdo al incremento.

Significados

FOR : Para *(requiere dos valores uno inicial y otro final)*
N : Variable del contador *(minúsculas preferentemente)*
NEXT : Siguiente *(empieza el contador de uno en uno)*
STEP : Paso *(requiere un valor de incremento para el contador)*



m = valor inicial
N = valor final
S = incremento del contador
i : variable del contador

Sintaxis 1 :

#Valor Inicial #Valor Final FOR *i* Proceso NEXT

Sintaxis 2 :

#Valor Inicial #Valor Final FOR *i* Proceso #Incremento STEP

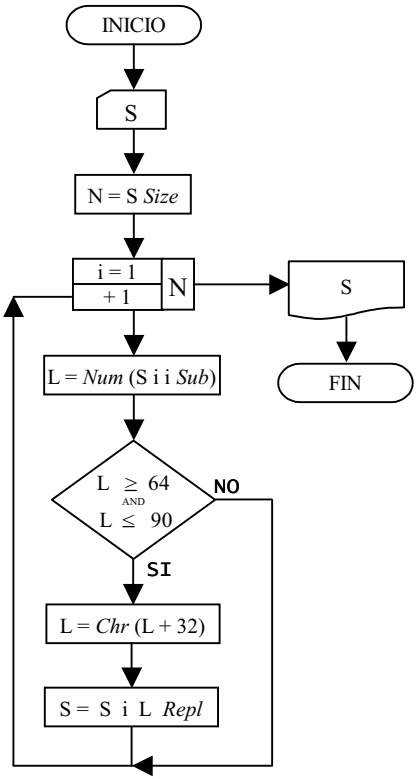
Ejemplo 1

En este ejemplo se utiliza NEXT en el programa que convierte un string de mayúsculas a minúsculas

Código HP

```
<<\  DUP  TYPE  2  ==
     IF  THEN  →  S
      <<\  S  1  S  SIZE
           FOR  i  S  i  i  SUB
             NUM  DUPDUP 64  ≥  SWAP
             90 ≤  AND
             IF  THEN  32  +  CHR
               i  SWAP  REPL
             ELSE  DROP  END
           NEXT
      \>>
     END
\>>
```

Diagrama de Flujo



Ejemplo II

En este ejemplo es utilizada la opción STEP para poder incrementar cualquier valor y así poder obtener múltiplo y finalmente mostrar los números que tienen raíz exacta.

```
<<\   " Raices exactas" { {}
      { "  Cantidad : " "Cantidad de números" 0 } { }
      { "  Multiplo : " " Multiplos a calcular ? " 0} }
      { 1 –2 } { }  { } INFORM
      IF  THEN OBJ→ DROP 0 [ 0 0 0 0 0 0 0 0 0 0 ]
        → N M C A
        <<\ M N
            FOR i i √ IP 2 ^ i ==
              IF THEN 'C' INCR DROP i 'A( C )' STO END
            M STEP
              1 C FOR i "Número " 'A( i )' EVAL
                  + MSGBOX
            NEXT
        \>>
      END
\>>
```

Código HP

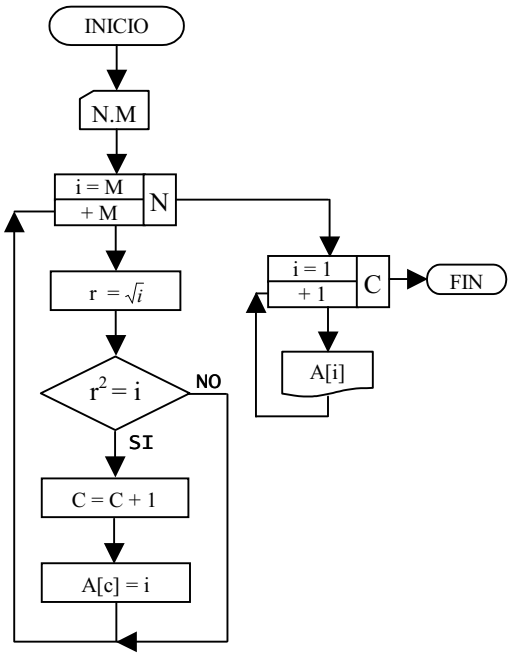


Diagrama de Flujo

4.10.4 START

Este comando es similar al anterior comando *FOR* excepto que no tiene un contador, este comando es poco utilizado en la programación

Significados

START : Empezar *(requiere dos valores uno inicial y otro final)*
NExt : Siguiente *(empieza con el proceso)*
STEP : Paso *(requiere un valor de incremento)*

Sintaxis 1 :

#Valor Inicial #Valor Final START Proceso NEXT

Sintaxis 2 :

#Valor Inicial #Valor Final START Proceso #Incremento STEP

Ejemplo

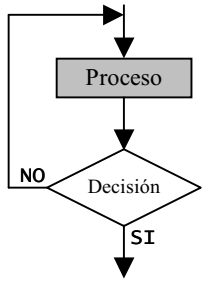
En el ejemplo se ve como este comando no utiliza un contador.

```
<<\  1  15  START   CLLCD
              " SYSTEM ERROR !!! "
              2  DISP  1000  0.1   BEEP
              0.05   WAIT   OFF
            NEXT
\>>
```

Código HP

4.1⊘.5 D⊕

Este comando repite un proceso hasta que la comparación efectuada o valor sea verdad para finalizar el proceso iterativo.



Significados

DO : Hacer *(empieza a realizar el proceso que contiene)*
UNTIL : Hasta que *(espera un valor verdadero para finalizar)*
END : Fin *(finaliza comando Do)*

Sintaxis :

DO Proceso UNTIL Numero o Comparación END

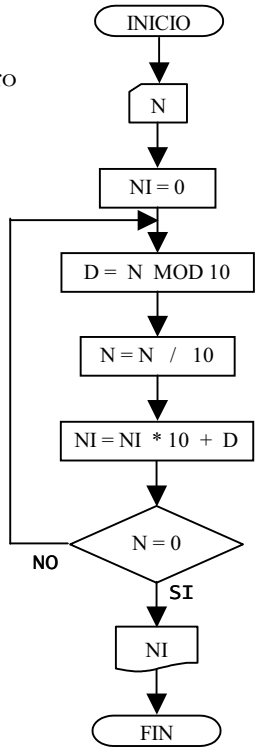
Ejemplo 1

Este ejemplo invierte los dígitos de cualquier número

```
<<\
 "DIGITOS INVERTIDOS

 Introduzca un número"
 " "  INPUT  OBJ→  DUP
 TYPE  0  ==
 IF  THEN  0
  → N  NI
  <<\
    DO  N  10  MOD
       N  10  /  IP  'N'  STO
       NI  10  *  +  'NI'  STO
    UNTIL
       N  0  ==
    END
    "Numero
     Invertido "  NI  +  MSGBOX
  \>>
 END
\>>
```

; Código HP



; Diagrama de Flujo

Ejemplo II

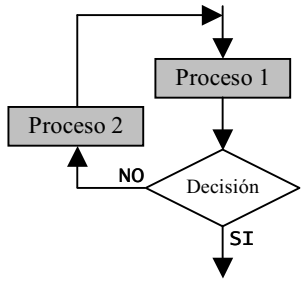
En este ejemplo se muestra el uso del comando**KEY**

Código HP:

```
<<\ CLLCD "Introduzca PassWord" 2 DISP
    1 5 START DO UNTIL KEY END 1000 0.1
    BEEP NEXT
    5 →LIST {31 43 52 21 15} ==
    IF THEN 1 6 FOR i i 100 * 0.2 BEEP NEXT
        "Archivos Abiertos" MSGBOX   Texto o gráficos
    ELSE "Usuario No Autorizado" 5000 0.3 BEEP
        MSGBOX
    END
\>>
```

4.1.5.1 DO (Caso Especial)

Este caso es cuando se tiene dos procesos a realizar, como se muestra en el diagrama de flujo, con el comando anterior esto no se podría codficar al lenguaje HP, la única manera es introduciendo el comando IF dentro del comando y la comparación a realizar seria lo inverso a la comparación que realiza el comando DO.

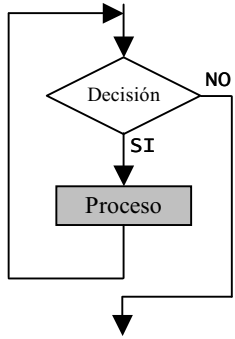


Sintaxis :

DO
Proceso 1
UNTIL Número o Comparación (*)
IF Número o Comparación(*) **NOT THEN** Proceso 2 **END**
END

4.10.6 WHILE

Comando similar al anterior comando *DO*, con la diferencia que este primero comprueba, luego hasta que el valor o comparación no sea verdadera repite el proceso que contiene este.



Significados

WHILE : Mientras *(espera un valor verdadero para continuar)*
REPEAT: Repetir *(vuelve a realizar el procedimiento)*
END : Fin *(finaliza comando While)*

Ejemplo

Este ejemplo encuentra los números primos que existen en una cantidad determinada de números.

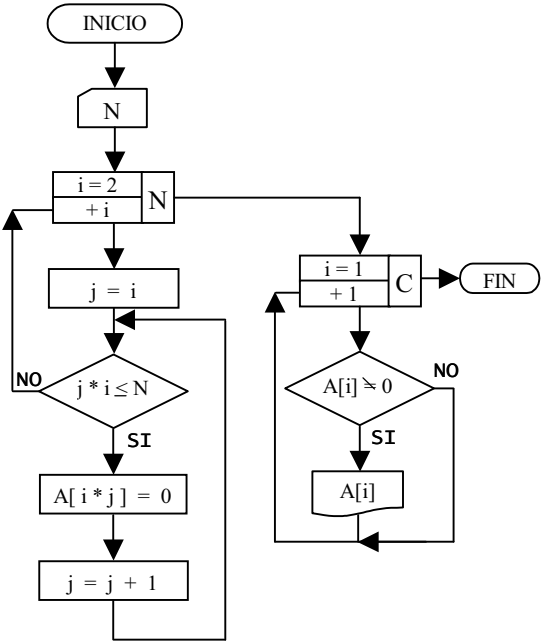


Diagrama de Flujo

```
<<\ "   NUMEROS PRIMOS

   Cantidad de números? "
   " "   INPUT OBJ→ DUP 1 SWAP
   FOR i i   NEXT   DUP   →ARRY   0
   → N A j
     <<\  2   N
          FOR   i    i   'j' STO
           WHILE i  j  *   N  ≤
           REPEAT    0   'A(i*j)'   STO
              'j'   INCR  DROP
           END
          NEXT  1   N
          FOR  i   'A(i)'   EVAL   0   ≠
             IF   THEN    "No. Primo→ "    'A(i)'
                  EVAL   +  MSGBOX
             END
           NEXT
     \>>
\>>
```

Código HP

4.10.7 IFT

Este comando es igual al comando *IF* la única diferencia que este solamente acepta una sola acción en cada parte.

IF **A** THEN **B** END → **A** **B** **IFT**

4.10.8 IFTE

Este comando es similar también al comando *IF* con la diferencia que este utiliza la opción *ELSE*, de igual forma solamente acepta una acción cada parte.

IF **A** THEN **B** ELSE **C** END → **A** **B** **C** **IFTE**

A	= Comparación o número
B	= Proceso 1 (Verdad)
C	= Proceso 2 (Falso)

4.10.A Relación diagramas de Flujo

En esta sección se muestra la relación de los símbolos de diagramas de flujo con los comandos aprendidos hasta esta parte.

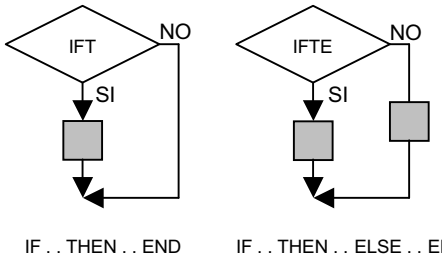
INICIO o FIN	<<\ \>>

ENTRADA DE DATOS	SECCION IN INPUT NOVAL INFORM CHOOSE PROMPT WAIT KEY

SALIDA DE DATOS	SECCION OUT MSGBOX CLLCD DISP BEEP FREEZE PVIEW TEXT

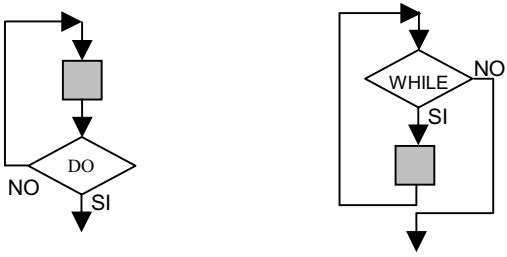
PROCESO DE DATOS	TODOS LOS COMANDOS HP

DECISIÓN DE DATOS	SECCION BRCH IF CASE IFT IFTE



IF . . THEN . . END IF . . THEN . . ELSE . . END

PROCESOS ITERATIVO	SECCION BRCH DO WHILE

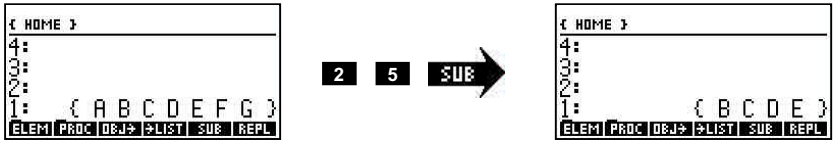


CONTADOR	SECCION BRCH FOR START

4.II Comandos de la sección LIST

En esta sección se describen los comandos de manipulación y procesos de listas

SUB : Requiere dos valores uno inicial y otro final para seleccionar los elementos que se desea obtener de una lista.



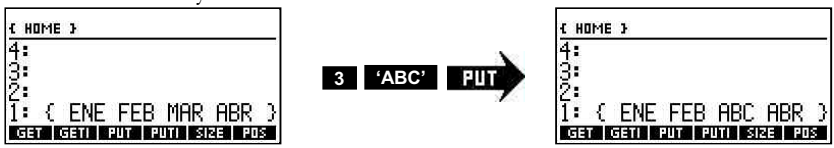
REPL : Este requiere el valor de la posición donde se desea remplazar un nuevo elemento o conjunto de elementos de una lista.

4.II.I ELEM

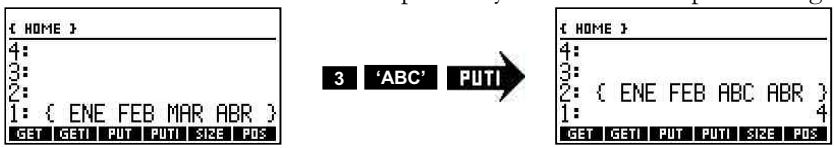
GET : Este comando es utilizado para obtener un determinado elemento de una lista, requiere la posición del demento.

GETI : Este es similar al comando *GET* con la diferencia que mediante este se obtiene la lista original, el número de la posición siguiente y el elemento, requiere la posición del elemento

PUT : Este reemplaza el elemento de una lista por otro, requiere la posición del elemento y valor.



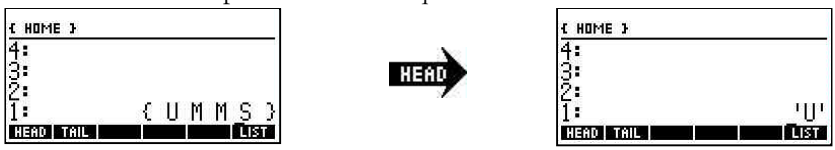
PUTI : Este es similar a al comando *PUT* con la diferencia que en este se obtiene la lista con el elemento reemplazado y el número de la posición siguiente



SIZE : Indica la cantidad de elementos que contiene una lista.

POS : Indica la posición de un elemento, requiere el nombre del elemento

HEAD : Devuelve el primer elemento que se encuentra en una lista.



TAIL : Devuelve toda la lista menos el primer elemento.

4.II.2 PR⊕C

DOLIST : Este comando ejecuta un programa o una función a partir de listas.

Sintaxis :

1 **Listas** a utilizar
2 **Cantidad** de listas a utilizar
3 **Programa o función** que se efectuara con los valores de las listas
4 **DOLIST**

Ejemplo

En este ejemplo se desea calcular la hipotenusa para n valores diferentes

```
<<\  "Ingrese los  valores de A"  { "{}"  { 1 2 } V }  INPUT OBJ→
     "Ingrese los valores de B"  { "{}"  { 1 2 } V }  INPUT OBJ→
      2   <<\  2  ^  SWAP  2  ^  +  √  \>>
     DOLIST
\>>
```

DOSUBS : Este comando aplica un procedimiento secuencial con los elementos de una lista.

Sintaxis 1 :

1 **Lista** donde se encuentran los valores a utilizar
2 **Número** de elementos a utilizar en cada procedimiento
3 **Programa o función** que se efectuara con esta secuencia
4 **DOSUBS**

Ejemplo

En este ejemplo se desea sumar secuencialmente tres elementos de una lista y elevarlo al cubo.

```
<<\  "Ingrese Lista"  { "{}"  { 1 2 } V }  INPUT OBJ→
      3   <<\  +  +  3  ^  \>>
     DOSUBS  \>>
```

Sintaxis 2 :

1 **Lista** donde se encuentran los valores a utilizar
2 **Programa o función** a efectuarse con cada elemento
3 **DOSUBS**

Ejemplo

En este ejemplo se efectuara la siguiente función Ln (Abs (x * Cos(x))) para n valores de x

```
<<\ "Ingrese Valores" { "{}" { 1 2 } V } INPUT OBJ→
    <<\ → x <<\ x x COS * ABS LN \>> \>>
    DOSUBS \>>
```

NSUB : Devuelve la posición en que se esta efectuando el proceso dentro DOSUBS

ENDSUB : Devuelve la cantidad de veces que efectuara el programa para cada valor, de igual manera este se efectúa dentro del comandoDOSUBS

STREAM : Este comando ejecuta una función con cada elemento que exista dentro de una lista

Sintaxis :

1 **Lista** donde se encuentran los valores a utilizar por STREAM
2 **Programa o función** a efectuarse con los valores
3 **STREAM**

Ejemplo

En este ejemplo se desea el valor de la función Ln (Abs (x * Cos(x))) para n valores de x

```
<<\ "Ingrese Valores" { "{}" { 1 2 } V } INPUT OBJ→
    <<\ → x <<\ x x COS * ABS LN \>> \>>
  DOSUBS \>>
```

REVLIST : Invierte la posición de los elementos de una lista.

SORT : Ordena los elementos de una lista numéricamente o alfabéticamente

SEQ : Reemplaza en una variable de una función u objeto valores dados uno inicial y otro final, variando en un incremento, devuelve en una lista los valores obtenidos.

Sintaxis :

1 **Función u objeto** que se utilizara para remplazar en una determinada variable
2 **Variable** de la función donde se desea remplazar los valores
3 **Valor inicial** a remplazar en la función
4 **Valor final** a remplazar en la función
5 **Incremento** dado para cada repetición
6 **SEQ**

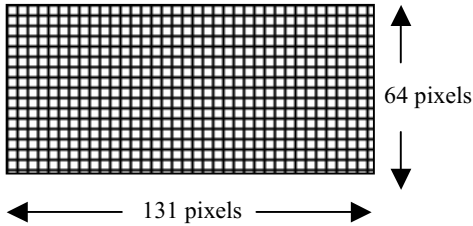
Ejemplo

En este ejemplo se desea remplazar solamente los números pares en una función, introduciendo el valor inicial, final y la función

```
<<\  "Ingrese la función" { " '' " { 1 2 } ALG } INPUT OBJ→
     "Ingrese la variable" { "" { 1 1} ALG } INPUT OBJ→
     "Ingrese Valor inicial y final" { ":Vi:
     :Vf:" { 1 5} V } INPUT OBJ→ DTAG SWAP
     DTAG DUPDUP 2 / IP 2 * =
     <<\ 1 + \>> IFT SWAP
     2 SEQ
\>>
```

4.12 Comandos de la sección PICT

En esta sección se encuentran todos los comandos de dibujo, que alguna vez se utilizo manualmente para dibujar algo. Mediante estos comandos se puede dibujar gráficos por medio de un programas de la misma forma que se lo realizaba de la forma manual, antes de continuar se debe conocer el modo de la pantalla gráfica.



Para poder ubicar un punto desde unprograma en esta pantalla existe dos maneras.

- NUMEROS BINARIOS

  Antes de proceder de esta forma la calculadora debe encontrarse en base 10, para cambiar el modo binario a la base 10**DEC** se realiza lo siguiente:

  Flecha Azul(Izquierda) + SYMB(P) → BASE → DEC

  La forma de ubicación es muy sencilla ya que nuestro origen se encuentra en la parte superior izquierda de la pantalla. Desde este punto de origen podemos ubicar las coordenadas de cualquier punto, nótese que al ser el origen (0,0) los limites de nuestra pantalla serán de 0 a 130 de largo y de 0 a 63 de ancho.
  La forma en que la 49G lee este tipo de coordenadas es:

  { # *No. De Fila* # *No. De Columna* }

  Se debe tener muy en cuenta, que antes del número de fila o columna debe encontrarse el símbolo **#** esto se reconocerá como un número binario y automáticamente la 49G pondrá seguido de este número la letra **d** indicando que esta en base 10 **DEC**

- SISTEMA DE CORDENADAS (x ,y)

  En este caso nuestro origen se encuentra justamenteen la parte central de la pantalla, esta forma no es muy aconsejable, porque primeramente se debe conocer la escala en que se encuentran las coordenadas, ademas este va cambiandose según los requerimientos del comando PLOT para el trazo de ecuaciones cuando se lo utiliza. La forma de introducir este tipo de coordenadas es:

  (*valor del eje* X , *valor del eje* Y)

PICT : Es algo parecido a una variable donde se guarda todo lo que se ve en el modo gráfico. Para poder obtener o sacar el gráfico de esta variable se presiona PICT en el nivel 1 se vera PICT, luego el comando RCL (Flecha Azul Izquierda + K)

PDIM : Crea un PICT en blanco(vació) con dimensiones personalizadas y lo restaura en la pantalla del modo gráfico.

Sintaxis :

1 **Número binario** valor de la longitud del PICT
2 **Número binario** valor del ancho del PICT
3 **PDIM**

Desde esta parte todos los comandos a continuación de esta sección necesitan las coordenadas de los puntos a utilizar que pueden estar en forma binaria o en coordenadas(x,y) conociendo la escala. Para poder notar y apreciar los gráficos realizados por medio de estos comandos de deberá ir al modo pantalla presionando Cursor Izquierda de las teclas

LINE : Dibuja una línea recta, requiere dos puntos

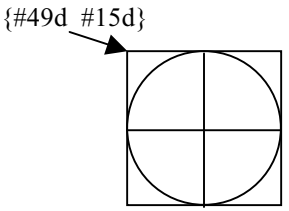
TLINE : Dibuja una línea también, requiere dos puntos

BOX : Dibuja un rectángulo, requiere dos puntos

ARC : Dibuja un arco, requiere centro, radio y los ángulos que determinan el arco. Se debe encontrar en modo DEG para el ángulo

Ejemplo

En este ejemplo se desea dibujar elsiguiente gráfico mediante un programa.



Se tiene una circunferencia inscrita en ur cuadrado y dos rectas perpendiculares que pasan por el centro de la circunferencia La medida del radio de la circunferencia es 15 pixels, conociendo el radio y un punto del grafico se puede dibujar lo demás sin necesidad de otros datos.

```
<<\   ERASE { #0d   #0d } PVIEW
      { #49d   #15d }  { #79d   #45d } BOX
      { #64d   #30d }    #15    0    360  ARC
      { #64d   #15d } { #64d   #45d } TLINE
      { #49d   #15d } { #64d   #45d } LINE
      { } PVIEW
 \>>
```

PIXON : Activa un determinado pixel de la pantalla, requiere un punto

PIXOF : Desactiva un determinado pixel, requiere un punto

PIX? : Comprueba si un pict esta activado

PX→C : Convierte las coordenadas pixel(binario) a usuario(sist. de coordenadas)

C→PX : Convierte coordenadas de usuario(sist. de coordenadas) a pixel(binario)

Ejemplo

En este programa se utiliza las segunda forma de coordenadas que son (x y) , el comando **R→C** convierte dos números reales a coordenadas y el comando **RAND** (Random)

```
<<\  ERASE  { #0d   #0d } PVIEW     0    30
       FOR i RAND  i  + RAND i + →  X   Y  2
   <<\  X  Y   R→C  X  NEG  Y   R→C  X Y NEG  R→C  X  NEG  Y
        NEG  R→C  Y   X   R→C  Y  NEG  X   R→C  Y   X   NEG
         R→C  Y  NEG  X  NEG  R→C  \>>
      PIXON  PIXON  PIXON  PIXON  PIXON
    PIXON  PIXON  PIXON  0.1  STEP
   \>>
```

4.13 Comandos de la sección GROB

Estos comandos sirven para reemplazar, sobreponer, capturar, crear, animar imágenes.

→GROB : Convierte un texto a gráfico, requiere un número para el tamaño

Sintaxis :

1 **Objeto** o texto que se desea convertir a grob
2 **Tamaño** a que se quiere convertir el objeto a gráfico 1 o 2
3 **→GROB**

BLANK : Crea un grob en blanco, requiere sus dimensiones.

Sintaxis :

1 **Ancho** del grob en número entero binario
2 **Alto** del grob en número entero binario
3 **BLANK**

GOR : Sobrepone un grob sobre otro grob, se requiere la ubicación.

GXOR : Este comando es similar al anterior comando *GOR* con la única diferencia que este invierte la imagen antes de sobreponerla.

Sintaxis :

1 **Grob** donde se sobrepondrá una imagen
2 **Ubicación** puede ser un vector o en números enteros binarios
3 **Grob** que se desea sobreponer
4 **GOR** o **GXOR**

SUB : Este extrae una porción de un grob, requiere valor inicial y final.

- **Sintaxis :**

1 **Grob** o PICT en caso que se desee utilizar la imagen que se encuentra en el modo gráfico
2 **Valor inicial** es la ubicación de origen, vector o en números binarios
3 **Valor final** es la ubicación final, vector o en números binarios
4 **SUB**

REPL : Reemplaza o sobrepone una porción de grob en otro grob o Pict .

- **Sintaxis :**

1 **Grob** o PICT en caso que se desee utilizar la imagen que se encuentra en el modo gráfico
2 **Ubicación** donde se desea remplazar o sobreponer un grob, vector o en números binarios
3 **Grob** que se desea remplazar
4 **REPL**

→LCD : Muestra el grob dado en la pantalla en modo texto.

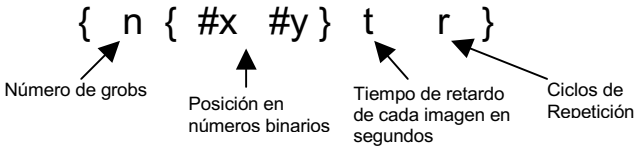
LCD→ : Convierte la pantalla de modo texto a un grob.

SIZE : Devuelve el tamaño del grob en números binarios

ANIMATE : Comando de animación requiere varios grobs para visualizar secuencialmente

- **Sintaxis :**

1 **Grobs** a utilizar en la secuencia de animación
2 **Número** de grobs que se utiliza para la animación o una lista de la siguiente forma:



3 **ANIMATE**

4.14 Comandos de la sección ERROR

DOERR : Permite crear un mensaje de error en un recuadro en la parte central de la pantalla, al efectuarse un error en un programa este finaliza. También este comando llama a un error especifico mediante su número hexadecimal.

Ejemplo

En este ejemplo se requiere crear un error de alerta en caso que alguien no este autorizado a alguna entrada, para efectuar este se escribe el mensaje "Error: No autorizado" este debe estar en strings, a continuación el comando DOERR.

ERRN : Devuelve el número del ultimo error ocasionado en modo hexadecimal

ERRM : Devuelve el mensaje del ultimo error ocasionado en strings

Ejemplo

Para este ejemplo se requiere efectuar cualquier error para obtener su número y mensaje:

ERRN → #201h
ERRM → "Too Few Arguments"

ERRO : Elimina de la memoria temporal él ultimo error ocasionado

LASTARG: Devuelve en el nivel 1 el ultimo comando (*similar al comando manual UNDO*)

4.14.1 IFERR

Este comando verifica si existe algún error en el momento de ejecución de una determinada sección de un programa

Significados

IFERR : Verifica si existe un error(*Inicio del comando*)
THEN : Entonces (*si existiera algún error efectúa una instrucción determinada*)
ELSE : De otra manera (*sino existiera ningún error efectúa otra segunda instrucción determinada*)
END : Fin (*finaliza el comando IFERR*)

Ejemplo

En este ejemplo mientras no se presione la tecla ALPHA + ROJO + T el programa no finalizara.

```
<<\ "Presione una tecla" 1 DISP
 IFERR DO 0 WAIT
       UNTIL 54.6 ==
       END
 THEN DROP P15 END
  "Programa Finalizado"
  DOERR
\>>
```

La parte seleccionada es la sección en que el comando IFERR verifica si hubiera algún error en el momento de ejecución.
El único error que pudiera ocurrir fuera si se presiona la tecla ON (Cancel) en el momento de ejecución, si fuera así procedería a efectuarse las instrucciones en caso de error.
P15 es el nombre del programa donde debe estar guardado

4.15 Comandos de la sección TIME

Estos comandos sirven para obtener, restaurar fechas, horas, etc.

DATE : Devuelve la fecha actual formato MM.DDAAAA

→DATE : Restaura una nueva fecha.

TIME : Devuelve la hora actual formato HH.MMSS

→TIME : Restaura una nueva hora.

TICKS : Devuelve la hora del sistema en Ticks

DATE+ : Calcula la fecha, a x días de la fecha

DDAYS : Calcula el numero de días que hay entre dos fechas.

→HMS : Convierte una hora en formato HMS a decimales

HMS→ : Convierte una hora en decimales a una hora en formato HMS

HMS+ : Suma dos horas en formato HMS

HMS- : Resta dos horas en formato HMS

TSTR : Crea una cadena con la fecha y hora.

CLKADJ : Ajusta el reloj a sistema x/8192 seg.

INFORMACION:

USER RPL v1.8
Este texto es una versión beta, la ultima versión la puede adquirir de la página HP-Civil, donde se incluyen los comandos mas usados y ejemplos de aplicación a la Ingeniería Civil, además esta nueva versión es tanto para la 49G como para la 48G(x).

SOBRE EL AUTOR
- Estudiante de Ingeniería Civil -- **UNIVERSIDAD MAYOR DE SAN SIMON**
- Auxiliar de la materia de "Computación para Ingeniería" (CIV-217)
- Lenguajes de Programación: Visual Basic, Delphi
- Diseñador de paginas Web
- Diseñador Grafico

HP-CIVIL
Invita a todos que desean formar parte de este grupo de investigación sobre la HP aplicada a la Ingeniería Civil de BOLIVIA. Todo lo publicado hasta el momento fue realizado por mi persona.

NOTICIAS HP-CIVIL
- Ahora puedes bajar el **EMU48** la versión en español.
- La sección de tutorías esta actualizada donde encontraras mucha ayuda para tu HP
- Muy pronto estará el texto de programación en Sys Rpl, el primer número.
- En Julio tendrá otro diseño la pagina HP-Civil (Totalmente Interactivo)

www.angelfire.com/co/48gx/49g.html

*Si deseas el texto USER RPL v1.8 en formato normal que incluye un disquet donde se encuentran todos los códigos de los programas y herramientas útiles. Contáctate a **hp_club@mixmail.com** el costo de este es de 9$ que incluye gastos de envió dentro Sudamérica.
Para Bolivia comunicarse al teléfono 04-480742 (Cochabamba) el costo es de 5$