CAPÍTULO 10

ARRAYS

Hay dos grupos de objetos en las que realmente representa las series la hp49G. El primer grupo (que se describirá en este capítulo) ha existido desde la hp48: las series normales (éstas para el usuario puede ser solamente compuesto de números reales o complejos) y las series unidas que no son accesibles al usuario. En la hp49g se introdujo un nuevo tipo de objeto para representar las series: los matrices simbólicos. Desde que éstos, realmente, son una parte del CAS de la hp49g, estos objetos se describen en el capítulo 43.

En usuario RPL, las series sólo pueden ser de números reales o complejos. En el sistema RPL, usted puede componer series de cualquier objeto, incluso series de series. Note que las series no son objetos compuestos (vea capítulo 11) aún cuando se parece. También una serie puede contener un solo tipo de objeto.

Usando MASD, las series se ingresan así:

```
1   ARRY   [[ % 1. % 2. %3. ]
           [ % 4. % 5. %6. ]]
```

Esto no es diferente de ingresar la serie en forma normal en la hp49g.

Usted también puede crear una serie real o compleja poniéndolos en la pila en el orden adecuado, luego ingresando en una lista las dimensiones (en números reales, no en enteros binarios) de la serie en el nivel uno de la pila, después de esto ejecute ^XEQ>ARRY. Esta función verifica que existe suficientes argumentos y si son de los tipos poyados.

La función ^ARSIZE devuelve el número de elementos en una serie. Usted puede conseguir las dimensiones de la serie con ^DIMLIMITS que devuelve en una lista, en números binarios, las dimensiones de la serie. Para conseguir un elemento de una serie, ponga el número del elemento en nivel dos, la serie en nivel uno, y ejecuta GETATELN. Usted conseguirá el elemento y TRUE si fuera encontrado o sólo FALSE si el elemento no existe. Más funciones de la serie se describen debajo.

Las entradas que se describe aquí son para series normales (aunque algunos de ellos, trabajan en el CAS para matrices simbólicos), si necesita entradas para matrices simbólicos, consulte el capítulo 43.

10.1 REFERENCIA

10.1.1 FUNCIONES GENERALES

DRECCIÓN	NOMBRE	DESCRIPCIÓN
0371D	GETATELN	(# [] → ob T) (# [] → F) saca el elemento indicado de la serie
16D006	^MDIMS	([[]]→ #rows #cols T) ([] → #elem F) Devuelve las dimensiones de la serie. Equivalente a MDIMS en la hp48

35FD8	MDIMSDROP	([2D] → #m #n) ejecuta MDIMS, luego DROP
16E006	^DIMLIMITS	([]→ { # }) ([[]]→ {# #}) retorna las dimensiones de la serie, parecido al comando SIZE en user RPL, pero las dimensiones son en números binarios, en la hp48 DIMLIMITS
35E006	^ARSIZE	([]→ #) devuelve el número de elementos de la serie
36183	OVERARSIZE	([] ob → [] ob #elts) primero hace OVER, luego ARSIZE
36183	PULLREALEL	([%] # → [%] %) saca el número real indicado
260F3	PULLCMPEL	([C%] # → [C%] C%) saca el número complejo indicado
26102	PUTEL	([%] % # → [%]') ([C%] C% # → [C%]') inserta el elemento a la posición especificada,
26107	PUTREALEL	([%] % # → [%]') inserta un número real a la posición especificada
260FD	PUTCMPEL	([C%] C% # → [C%]') inserta un número complejo a la posición especificada
33B006	^MATTRAN	(M → M') halla la transpuesta de una matriz
331006	^Yext	(V2 V1 → ob) Producto escalar entre vectores.

10.1.2 CONVERSIÓN

DIRECCIÓN	NOMBRE	DESCRIPCIÓN
169006	^BESTMATRIXTYPE	(ob → ob) convierte una matriz simbólica que contiene números reales o complejos, a una serie numérica
172006	^CKNUMARRY	(ob → ob) Prueba si el objeto es una serie numérica. Si es una serie simbólica intenta convertir la serie simbólica a la serie numérica
178006	^MATRIX2ARRAY	([]→ []) ([[]]→ [[]]) Intenta convertir una matriz simbólica a un número uno
001007	^ListToArry	({}/{{}} → []/[[]] TRUE) ({}/{{}} → FALSE) Si es posible, convierte una lista a una serie, devolviendo la serie y TRUE, de lo contrario devuelve FALSE.

COMENTARIOS: edcivilic@lycos.es

17F006	^XEQ>ARRY	(ob1...obn {%n} → []) (ob11...obmn {%m %n} → [[mxn]]) retorna los objetos en una matriz especificada.
17C006	^XEQARRY>	([]→ ob1...obn meta-arry) Extrae todo los elementos de la matriz
002007	^ArryToMatrix	([]→ M) Convierte a una serie simbólica.

10.1.3 ESTADÍSTICAS

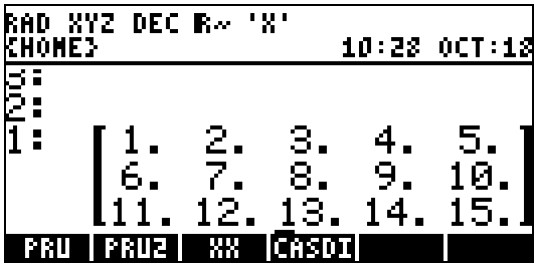
DIRECCIÓN	NOMBRE	DESCRIPCIÓN
2EEDA	STATCLST	(→) Aclara(borra) DAT
2EEDB	STATSADD%	(% →) Equivalente al comando Σ+ incorporado en CAS
2EEDC	STATN	(→ N) Equivalente al comando NΣ incorporado en CAS
2EEDF	STATSMIN	(→ %) Equivalente al comando MINΣ incorporado en CAS
2EEDD	STATSMAX	(→ %) Equivalente al comando MAXΣ incorporado en CAS
2EEDE	STATMEAN	(→ %) (→ []) Equivalente al comando MEAN incorporado en CAS
2EEE0	STATSTDEV	(→ %) (→ []) Equivalente al comando SDEV incorporado en CAS
2EEE1	STATTOT	(→ %) (→ []) Equivalente al comando TOT incorporado en CAS
2EEE2	STATVAR	(→ %) (→ []) Equivalente al comando VAR incorporado en CAS

EJEMPLOS

EJEMPLO #1: Este programa genera 15 números reales y luego lo agrupa en una serie de 3*5(matriz 3*5, es decir la matriz cuenta con tres filas y cinco columnas)

```
::                  @inicia system
  CK0NOLASTWD       @no necesita argumento
  BINT15            @fin del bucle
  #1+_ONE_DO        @inicia bucle con #1 y parada #15
  INDEX@            @esto sería el contador(en #i)
  UNCOERCE          @convierte a un número real
  LOOP              @fin del bucle
  {                 * Todo dentro de la lista es la dimensión de la matriz
    %3              @número de filas
    % 5.            @número de columnas
  }
  FPTR 6 17F        @devuelve la serie, equivalente a: ^XEQ>ARRY
;                   @fin del programa
@                   @código para ensamblar con masd.
```

- después de ejecutar el programa, en el primer nivel de la pila, devuelve la matriz.

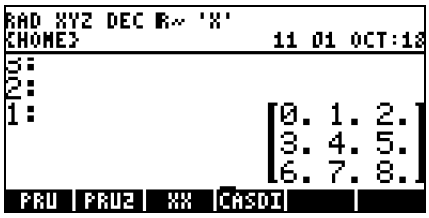


EJEMPLO #2: Este programa necesita como argumento una serie compuesto por números reales en el primer nivel de la pila, devolviendo la serie modificada.

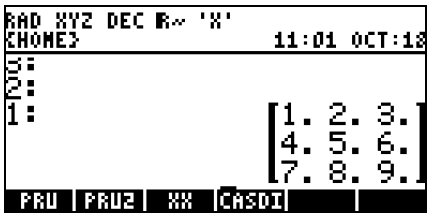
```
::
  CK1NOLASTWD       @necesita un argumento
  DUPTYPEARRY?      @duplica y pregunta si es una serie
  Case              *el subprograma se ejecuta si es cierto
  ::                @inicia subprograma
    DUP             @duplica
    FPTR 6 35E      @obtiene número de elementos: ^ARSIZE
    #1+_ONE_DO      @inicia bucle DO con inicio #1 y parada #elemetos
    INDEX@
    PULLREALEL      @saca el elemento del lugar indicado
    %1+             @suma uno(número real)
```

```
   INDEX@
   PUTREALEL          @inserta el número modificado
   LOOP               @fin del bucle
  ;                   @Termina subprograma
  DROP                     *Esto se ejecuta si es falso @elimina objeto nivel uno
  "Error: Ingrese una serie!"   @mensaje
  FlashWarning               @muestra el mensaje
;
@
```

Por ejemplo:



devuelve



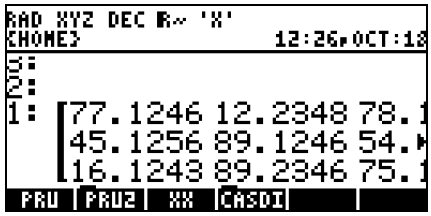
EJEMPLO #3: esto es parte del programa "C.angular" (compensación de los ángulos internos de un polígono con un punto central), en esto, los ángulos de cada triángulo (que forman el polígono), están en una misma fila, con la condición de que en la columna tres estén los ángulos que forman el punto central. Entonces, para que tengas una idea el programa dirá: cada fila debe sumar 180º00'00" y la columna tres 360º00'00'.

El programa devuelve una matriz, de (m+1)*3, en la primera columna la suma teórica, segunda columna, sumatoria de los ángulos de cada fila y la tercera columna la diferencia (fila1 – fila2) en h.mmss. la última fila representa del punto central.

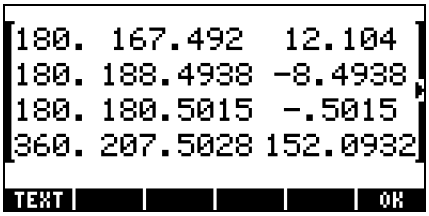
```
::                          @inicia sytem
  CK1                       @verifica que existe un argumento
  DUPTYPEARRY?              @duplica y pregunta si es un serie
  ITE                       @decision
  ::                  *este subprograma se ejecuta si es TRUE
    DUP                     @duplica
    FPTR 6 16D              @dimensión de la serie:  ^MDIMS
    NOT?SEMI                @si es verda continua
    BINT3
    #=case                  @decición
    ::                  ** ejecuta este subprograma si es verdad y termina
      DROPDUP               @elimina y duplica
      FPTR 6 35E            @número de elementos:  ^ARSIZE
      ONE_DO
      %180
      SWAP
      INDEX@
      PULLREALEL            @obtiene el elemento indicado
      SWAP
      INDEX@
      #1+
```

```
    PULLREALEL
    ROT
    %HMS+                    @suma dos números reales en h.mmss
    SWAP
    INDEX@
    #2+
    PULLREALEL
    ROT
    %HMS+
    DUP
    %180
    SWAP
    %HMS-                    @resta dos números reales en h.mmss
    ROT
    BINT3                    @incermento
    +LOOP
    %360
    %0
    ROTDUP
    FPTR 6 35E               @ número de elementos:  ^ARSIZE
    #3+
    BINT3
    DO
    INDEX@
    PULLREALEL
    ROT
    %HMS+
    SWAP
    BINT3
    +LOOP
    SWAPDUP
    %360
    SWAP
    %HMS-
    ROT
    MDIMSDROP                @dimensión de  la serie
    SWAP#1+SWAP
    UNCOERCE2                @transforma dos reales a bynarios
    TWO{}N                   @retorna dos objetos enuna lista
    FPTR 6 17F               @ordena:  ^XEQ>ARRY
  ;
  2DROP                      **ejecuta este otro si es falso y termina
  "Error: Dimensión incorrecta!"
  FlashWarning
 ;
 ::                                  *esto si es falso
  DROP
  "Error:Ingrese matriz de m*3"
  FlashWarning
 ;
;
@
```

Por ejemplo cuando tenemos:



y se ejecuta el programa se tiene:



EJEMPLO #4: A veces, es necesario recibir datos en el propio editor de matrices de la calculadora, para luego evaluarlo, procesarlo,… en fin. Eso es lo que hace este programa:

```
::
  CK0NOLASTWD
  DoNewMatrix              @abre el editor de matrices.
;
@
```

EJEMPLO #5: Cuando se desea mostrar resultados en una matriz, en el nivel uno de la pila debe tener la serie y luego ejecutar este programa:

```
::
  CK1NOLASTWD
  DUPTYPEARRY?
  case
  DoOldMatrix                    @edita la matriz existente en el primer nivel
  "Error:Argumento incorrecto"
  FlashWarning
;
@
```

AUTOR: CANCHARI GUTIÉRREZ, Edmundo.

COMENTARIOS: edcivilic@lycos.es