TI-89 / Voyage™ 200 Programación

Cómo

- Ejencutar un programa existete
- Iniciar una sesión del editor de programas
- Llamar un programa desde otro
- Usar variables en un programa
- Controlar el flujo de programa
- Configurar la TI-89 / Voyage 200 PLT
- Usar comandos de reloj
- Crear un menú personalizado
- Crear una tabla o un gráfico
- Gestionar errores de programa

Ejemplos

- Uso de enfoques alternativos

Más información

- Ayuda al cliente

© 2002 Texas Instruments

Importante

Texas Instruments no ofrece garantía alguna, ya sea explícita o implícita, incluidas, sin limitarse a ellas, garantías implícitas de comerciabilidad o idoneidad para un uso concreto, en lo que respecta a los programas o manuales y ofrece dichos materiales únicamente “tal y como son”.

En ningún caso Texas Instruments puede hacerse responsable ante cualquier persona por daños especiales, colaterales, accidentales o consecuentes relacionados o causados por la adquisición o el uso de los materiales mencionados, y la responsabilidad única y exclusiva de Texas Instruments, independientemente de la forma de acción, no sobrepasará el precio de compra de este equipo. Asimismo, Texas Instruments no puede hacerse responsable de las reclamaciones de cualquier clase contra el uso de dichos materiales por cualquier otra parte.

Ejecución de un programa existente

Tras crear un programa (según lo descrito en las restantes secciones de este módulo), puede ejecutarlo en la pantalla Home. La salida del programa, si la hay, se presenta en la pantalla Program E/S, en un recuadro de diálogo o en la pantalla Graph.

Ejecución de un programa

En la pantalla Home:

1. Escriba el nombre del programa.
2. Debe escribir siempre paréntesis después del nombre.

   Algunos programas necesitan la introducción de un argumento.

   Consejo: Utilice [2nd] [VAR-LINK] para mostrar una lista de las variables **PRGM** existentes. Resalte una variable y pulse [ENTER] para pegar el nombre en la línea de entrada.

```
prog1()
```

Si no se necesitan argumentos

```
prog1(x,y)
```

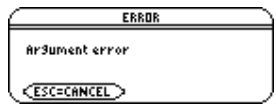
Si se necesitan argumentos

3. Pulse ENTER.

Nota: Los argumentos indican los valores iniciales de un programa. Consulte Introducción de valores en un programa.

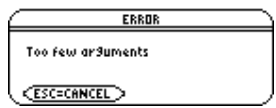
Al ejecutar un programa, la TI-89 / Voyage™ 200 PLT comprueba automáticamente la existencia de errores. Por ejemplo, el siguiente mensaje aparece si:

- No introduce ( ) después del nombre del programa.



Este mensaje de error aparece si:

- No introduce suficientes argumentos, cuando son necesarios.



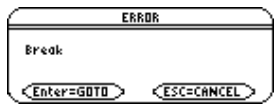
Para cancelar la ejecución del programa en caso de que se produzca un error, pulse ESC. A continuación, puede corregir el error y volver a ejecutarlo.

Nota: La TI-89 / Voyage 200 PLT también comprueba los errores de tiempo de ejecución dentro del programa. Consulte Errores de tiempo de ejecución.

“Interrupción” de un programa

El indicador **BUSY** se presenta en la línea de estado mientras el programa se está ejecutando.

Pulse [ON] para interrumpir la ejecución. A continuación se mostrará un mensaje.



- Para presentar el programa en **Program Editor**, pulse [ENTER]. El cursor se situará en la orden en la que se produjo la interrupción.
- Para cancelar la ejecución del programa, pulse [ESC].

¿Dónde se muestra la salida?

Dependiendo de las órdenes del programa, la TI-89 / Voyage™ 200 PLT presenta automáticamente la información en la pantalla correspondiente.

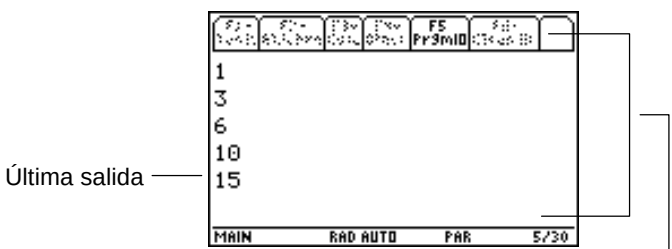
- La mayor parte de las órdenes de entrada y salida emplean la pantalla Program E/S. Las órdenes de entrada solicitan al usuario la introducción de información.

- Las órdenes de gráficas emplean normalmente la pantalla Graph.

Una vez que el programa se interrumpe, la TI-89 / Voyage™ 200 PLT muestra la última pantalla presentada.

La pantalla Program E/S

En la pantalla Program E/S, el nuevo resultado aparece debajo de los ya existentes previamente (que pueden haber aparecido anteriormente como consecuencia de la ejecución del mismo programa o de otro distinto). Una vez que la página de salida está completa, las salidas anteriores van desapareciendo por la parte superior de la pantalla.



En la pantalla Program E/S:
- El menú F5 está disponible en la barra de herramientas; los restantes están atenuados.
- No hay línea de entrada.

Consejo: Para borrar las salidas anteriores, introduzca la orden **ClrIO** en el programa. **ClrIO** también puede ejecutarse en la pantalla Home.

Si el programa se interrumpe en la pantalla Program E/S, deberá asegurarse de que no se encuentra en la pantalla Home (las dos pantallas son similares). La pantalla Program E/S sólo se emplea para mostrar la salida o solicitar la introducción de información y no permite la realización de operaciones.

Consejo: Si las operaciones de la pantalla Home no funcionan tras ejecutar un programa, puede que se encuentre en la pantalla Program E/S.

Abandonar la pantalla Program E/S

En la pantalla Program E/S:

- Pulse [F5] permite alternar las pantallas Home y Program E/S).
  — o —
- Pulse [ESC], [2nd] [QUIT], o
  TI-89: [HOME]
  Voyage™ 200 PLT: [♦] [CALC HOME]
  para presentar la pantalla Home.
  — o —
- Muestre otra pantalla de aplicación (con [APPS], [HOME], [♦] [Y=], etc.).

Inicio de una sesión de Program Editor

Con cada inicio de Program Editor se permite reanudar el programa o función actual (el que se mostraba la última vez que se empleó Program Editor), abrir un programa o función existente, o iniciar un programa o función nuevo.

Inicio de un nuevo programa o función

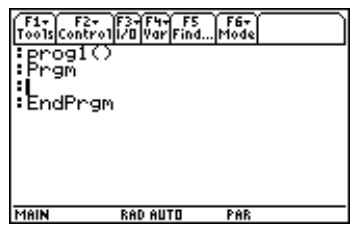
1. Pulse APPS y, a continuación, seleccione **Program Editor**.
2. Seleccione **3:New**.
3. Determine la información correspondiente del nuevo programa o función.

Elemento	Permite:	
Type	Elegir entre crear un programa o una función.	1:Program 2:Function
Folder	Seleccionar la carpeta en la que se va a almacenar el nuevo programa o función. Para obtener información sobre las carpetas, consulte el módulo *Pantalla principal de la calculadora*.	

Elemento	Permite:
Variable	Escribir un nombre de variable para el programa o función. Si especifica una variable que ya existe, al pulsar ENTER aparecerá un mensaje de error. Al pulsar ESC o ENTER para confirmar el error, se abrirá de nuevo el recuadro de diálogo **NEW**.

4. Pulse ENTER (tras escribir en un cuadro de entrada como **Variable**, deberá pulsar ENTER dos veces) para presentar una "plantilla" vacía.

Esta es la plantilla del programa. Las funciones tienen uno similar.



A continuación, puede utilizar Program Editor según lo descrito en las restantes secciones de este módulo.

Nota: El programa (o función) se guarda automáticamente al escribirlo. Por tanto, no es preciso almacenarlo manualmente antes de abandonar Program Editor, de iniciar un nuevo programa o abrir uno anterior.

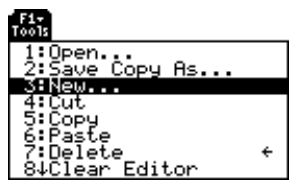
Continuación del programa actual

Puede abandonar Program Editor en cualquier momento para pasar a otra aplicación. Para volver al programa o función mostrado cuando salió de Program Editor, pulse APPS **7** y seleccione **1:Current**.

Inicio de un nuevo programa en Program Editor

Para abandonar el programa o función actual e iniciar uno nuevo:

1. Pulse F1 y seleccione **3:New**.
2. Especifique el tipo, carpeta y variable para el programa o función.
3. Pulse ENTER dos veces.



Apertura de un programa anterior

Los programas o funciones creados anteriormente pueden abrirse cuando se desee.

1. En Program Editor, pulse F1 y seleccione **1:Open**.
   — o —

   En una aplicación distinta, pulse APPS **7** y seleccione **2:Open**.

2. Seleccione el tipo, carpeta y variable correspondiente.

3. Pulse ENTER.

Nota: Por omisión, Variable muestra el primer programa o función existente en orden alfabético.

Copia de un programa

En algunos casos, puede interesarle copiar un programa o función para editar la copia y conservar el original.

1. Presente el programa o función que quiere copiar.

2. Pulse F1 y seleccione **2:Save Copy As**.

3. Especifique la carpeta y variable para la copia.

4. Pulse [ENTER] dos veces.

Nota sobre el borrado de un programa

Dado que todas las sesiones de Program Editor se almacenan automáticamente, los programas y funciones anteriores pueden ir acumulándose hasta agotar la memoria.

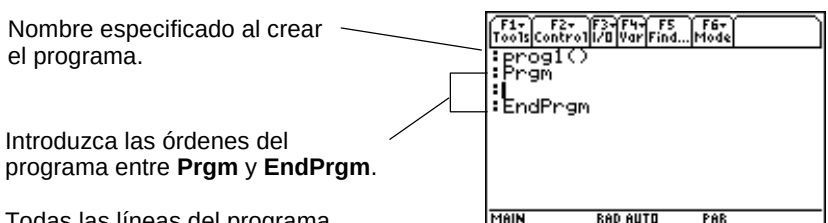
Para borrar programas y funciones, utilice la pantalla **VAR-LINK** ([2nd] [VAR-LINK]). Para obtener información sobre **VAR-LINK**, consulte *Gestión de la memoria y de las variables*.

Descripción de la introducción de un programa

Un programa es una serie de órdenes ejecutadas en orden secuencial (aunque algunas órdenes alteran el flujo del mismo). En general, todo lo que puede ejecutarse en la pantalla Home puede incluirse en un programa. La ejecución del programa continúa hasta llegar al final o hasta que se ejecuta la orden **Stop**.

Introducción y edición de instrucciones

Las órdenes para el nuevo programa se introducen en un listado vacío.



Nota: Utilice la tecla del cursor para desplazarse por el programa e introducir o editar las. Utilice ◆ ▲ o ◆ ▼ para ir a la parte superior o inferior de un programa, respectivamente.

Las órdenes del programa se introducen y editan en Program Editor utilizando las mismas técnicas empleadas para introducir y editar texto en Text Editor. Consulte “Introducción y edición de texto” en *Text Editor*.

Tras escribir cada línea del programa, pulse ENTER. De esta forma se inserta una nueva línea en blanco que permitirá continuar introduciendo otra. La línea del programa puede tener una longitud superior a la línea de la pantalla, en cuyo caso, pasará automáticamente a la siguiente línea de ésta.

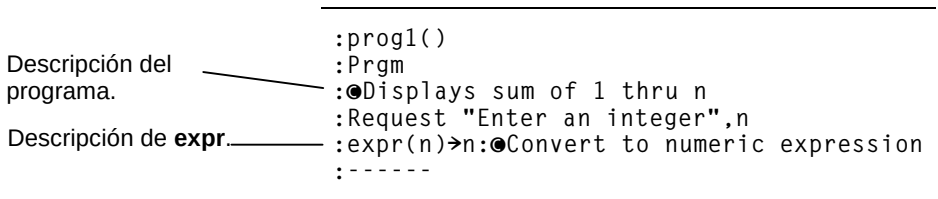
Nota: La introducción de una orden no implica su ejecución. Ésta se produce al ejecutar el programa.

Introducción de líneas con varias órdenes

Para introducir más de una orden en la misma línea, sepárelas mediante dos puntos pulsando 2nd [:].

Introducción de comentarios

El símbolo (◉) permite introducir comentarios en el programa. Al ejecutarlo, se ignorarán todos los caracteres situados a la derecha de ◉.



Consejo: Utilice comentarios para introducir información que resulte útil a quien lea la codificación del programa.

Para introducir el símbolo de comentarios pulse:

- TI-89: ◆)
  Voyage™ 200 PLT: 2nd **X**
  — o —
- Pulse F2 y seleccione 9:◉.

Control del flujo de un programa

Las instrucciones se ejecutan en orden secuencial. Sin embargo, algunas órdenes alteran el flujo del mismo. Por ejemplo:

- Las estructuras de control, como las órdenes **If...EndIf**, utilizan una prueba condicional para determinar la parte del programa que se va a ejecutar.
- Las órdenes de bucles, como **For...EndFor**, repiten un grupo de órdenes.

Consejo: Para obtener más información, consulte Uso de If, Lbl y Goto para controlar el flujo del programa y Uso de bucles para repetir un grupo de órdenes.

Uso del sangrado

Para programas más complejos que utilicen **If...EndIf** y estructuras de bucle como **For...EndFor**, el uso del sangrado puede hacer que sean fáciles de leer y entender.

```
:If x>5 Then
:  Disp "x is > 5"
:Else
:  Disp "x is < or = 5"
:EndIf
```

Presentación de los resultados de las operaciones

En los programas, los resultados no se presentan a menos que se utilice una orden de salida. Esta es la diferencia más importante entre la realización de operaciones en la pantalla Home y en un programa.

En un programa, los resultados de estas operaciones no se presentarían (aunque sí lo harían en la pantalla Home).

```
:12*6
:cos(π/4)
:solve(x^2-x-2=0,x)
```

Las órdenes de salida como Disp harán que se presenten los resultados al ejecutar un programa.

```
:Disp 12*6
:Disp cos(π/4)
:Disp solve(x^2-x-2=0,x)
```

Que aparezca el resultado de una operación no significa que se guarde para un posible uso posterior. Si necesita utilizar posteriormente un resultado, debe almacenarlo en una variable.

```
:cos(π/4)→maximum
:Disp maximum
```

Consejo: Para obtener una lista de las órdenes de salida disponibles, consulte Órdenes de salida.

Introducción de valores en un programa

Para introducir valores en un programa, puede:

- Solicitar al usuario que almacene un valor (con STO▸) en las variables necesarias antes de ejecutarlo. El programa podrá referirse a estas variables.
- Introducir los valores directamente.

```
:Disp 12*6
:cos(π/4)→maximum
```

- Incluir órdenes de entrada que soliciten al usuario la introducción de los valores necesarios al ejecutar el programa.

```
:Input "Enter a value",i
:Request "Enter an
integer",n
```

- Requerir al usuario que transfiera uno o más valores al ejecutarlo.

```
prog1(3,5)
```

Consejo: Para obtener una lista de las órdenes de entrada disponibles, consulte Órdenes de entrada.

Ejemplo de transferencia de valores a un programa

El siguiente programa dibuja una circunferencia en la pantalla Graph y, a continuación, traza una recta horizontal por la parte superior de dicha circunferencia. Se deben transferir tres valores al programa; las coordenadas **x** e **y** del centro de la circunferencia y el radio r de la misma.

- Al escribir el programa en Program Editor:

  Los nombres que aparecen entre ( ) junto al nombre del programa, indican las variables que se van a emplear para almacenar los valores que se transfieran.

  Observe que el programa también contiene órdenes que configuran la pantalla Graph.

```
:circ(x,y,r)
:Prgm
:FnOff
:ZoomStd
:ZoomSqr
:Circle x,y,r
:LineHorz y+r
:EndPrgm
```

En la plantilla, sólo aparece **circ( )** inicialmente; asegúrese de editar esta línea.

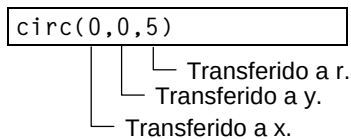
  Nota: En este ejemplo, no puede utilizar circle como nombre del programa por estar en conflicto con el nombre de una orden.

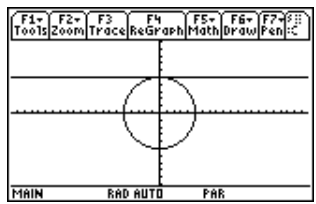
Antes de dibujar la circunferencia, el programa desactiva las funciones Y= Editor seleccionadas, presenta una ventana de visualización estándar y la convierte en “cuadrada”.

- Para ejecutar el programa en la pantalla Home:

  El usuario debe especificar entre **( )** los valores adecuados como argumentos.

  Los argumentos se transfieren al programa según el orden en que se introduzcan.





Nota: En este ejemplo se supone que se introducen valores que pueden presentarse en la ventana de visualización, definida mediante **ZoomStd** y **ZoomSqr**.

Descripción de la introducción de una función

Una función creada en Program Editor es muy similar a las funciones e instrucciones utilizadas habitualmente en la pantalla Home.

Razones para crear funciones definidas por el usuario

Las funciones (al igual que los programas) son idóneas para realizar operaciones o tareas repetitivas, ya que sólo es necesario escribirlas una vez para poder utilizarlas tantas veces como sea necesario. No obstante, las funciones ofrecen más ventajas que los programas.

- Pueden crearse funciones que amplíen las incorporadas en la TI-89 / Voyage™ 200 PLT, siendo su uso similar al de cualquier otra función.
- Las funciones devuelven valores que pueden representarse gráficamente o introducirse en una tabla; los programas carecen de esta ventaja.
- Las funciones (no los programas) pueden utilizarse en expresiones. Por ejemplo: **3∗func1(3)** es válido, no **3∗prog1(3)**.

- Dado que se transfieren argumentos a la función, pueden escribirse funciones genéricas no vinculadas a nombres concretos de variable.

Nota: Aunque puede crear funciones en la pantalla Home, Program Editor es más adecuado para funciones largas y complicadas.

Diferencias entre funciones y programas

Este manual emplea a veces el término orden como referencia genérica a instrucciones y funciones. Sin embargo, al escribir una función, es preciso establecer claramente las diferencias entre instrucciones y funciones.

Las funciones definidas por el usuario:

- Sólo pueden emplear las siguientes instrucciones. Cualesquiera otras no son válidas.

Cycle	**Define**	**Exit**
For...EndFor	**Goto**	**If...EndIf (en todas sus formas)**
Lbl	**Local**	**Loop...EndLoop**
Return	**While...EndWhile**	**→ (tecla STO▸)**

- Pueden emplear todas las funciones incorporadas en la TI-89 / Voyage™ 200 PLT excepto:

setFold	**setGraph**	**setMode**
setTable	**switch**	

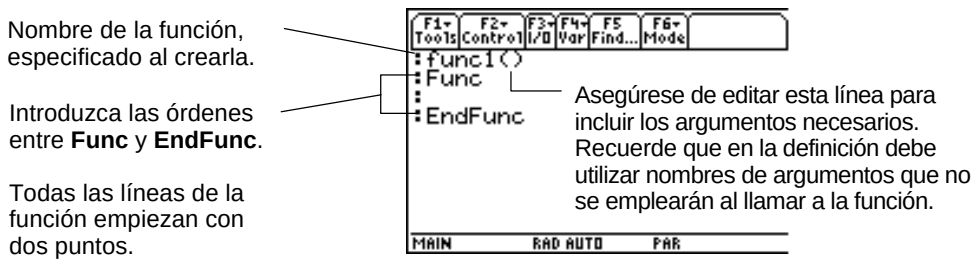
- Pueden referirse a cualquier variable; sin embargo, sólo pueden almacenar valores en variables locales.
  - Los argumentos utilizados para transferir los valores a la función se tratan automáticamente como variables locales. Si se almacenan en cualquier otra variable, deben definirse como locales dentro de la función.
- No permiten llamar a un programa como subrutina, aunque sí pueden recuperar otras funciones definidas por el usuario.
- No pueden definir un programa.
- No pueden definir una función global, pero sí una local.

Consejo: Para obtener información sobre las variables locales, consulte Uso de variables en un programa y Uso de variables locales en funciones o programas.

Introducción de una función

Al crear una nueva función en Program Editor, la TI-89 / Voyage™ 200 PLT muestra un “listado” en blanco.

Nota: Utilice la tecla del cursor para desplazarse por la función e introducir o editar órdenes.



Si la función necesita una entrada, deberán transferirse uno o más valores. Las funciones definidas por el usuario sólo pueden almacenarse en variables locales y no pueden emplear instrucciones que pidan una entrada al usuario.

Cómo devolver un valor desde una función

Existen dos formas de devolver un valor desde una función:

- Como última línea de la función (delante de **EndFunc**), calcule el valor que se va a devolver.

```
:cube(x)
:Func
:x^3
:EndFunc
```

- Utilice **Return**. Esto resulta útil para abandonar una función y devolver el valor a una posición distinta a la del final de la función.

```
:cube(x)
:Func
:If x<0
:  Return 0
:x^3
:EndFunc
```

Nota: En este ejemplo se calcula el cubo si **x**≥**0**; de lo contrario, devuelve el valor **0**.

El argumento x se trata automáticamente como variable local. Sin embargo, si en el ejemplo se hubiese necesitado otra variable, la función debería definirla como local mediante la orden Local.

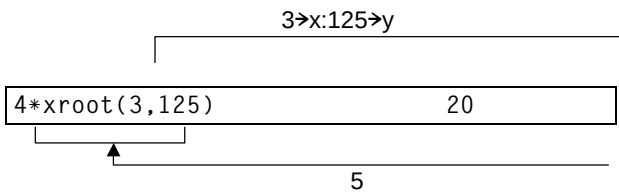
Al final de la función existe un **Return** implícito. Si la última línea no es una expresión, se producirá un error.

Ejemplo de función

La siguiente función devuelve la raíz de índice x de un valor **y (** $\sqrt[x]{y}$ **)**. Los dos valores que deben transferirse a la función son **x** e **y**.

Nota: Dado que en la función **x** e **y** son locales, cualquier variable de nombre **x** o **y** no les afectaría.

Función llamada desde la pantalla Home	**Función definida en Program Editor**



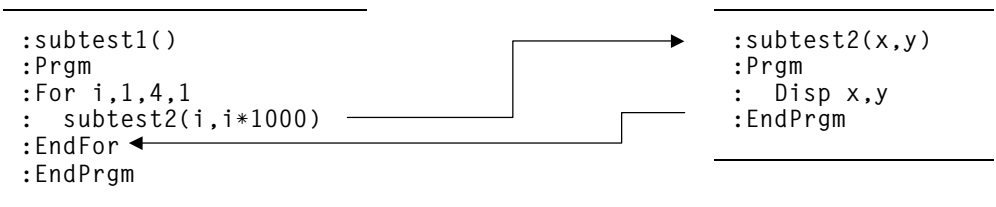

```
:xroot(x,y)
:Func
:y^(1/x)
:EndFunc
```

Llamada a un programa desde otro

Desde un programa se puede llamar a otro como subrutina. La subrutina puede ser externa (un programa aparte) o interna (incluida en el programa principal) y es útil cuando un programa necesita repetir el mismo grupo de órdenes en varias posiciones distintas.

Llamada a otro programa

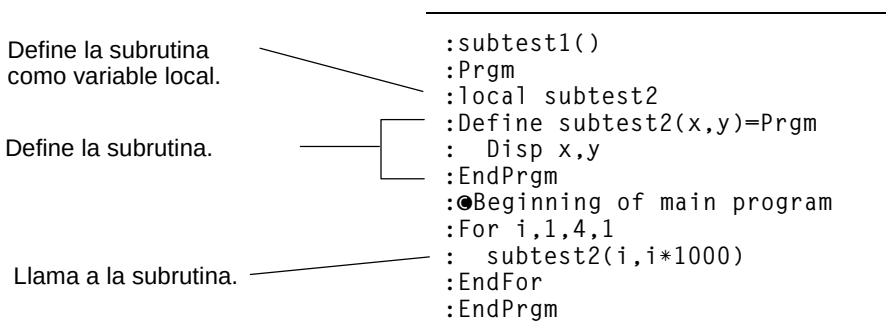
Para llamar a otro programa, utilice la misma sintaxis empleada para ejecutar el programa en la pantalla Home.



Llamada a una subrutina interna

Para definir una subrutina interna, utilice la orden **Define** con **Prgm...EndPrgm**. Dado que las subrutinas deben definirse antes de ser llamadas, se recomienda hacerlo al principio del programa principal.

Las subrutinas internas se llaman y ejecutan de la misma manera que los programas independientes.



Consejo: Utilice el menú F4 **Var** de la barra de herramientas de Program Editor para introducir las órdenes **Define** y **Prgm...EndPrgm**.

Notas sobre el uso de subrutinas

Al final de la subrutina, la ejecución vuelve al programa que la ha llamado. Para cancelar una subrutina en cualquier momento, utilice la orden **Return**.

Las subrutinas no tienen acceso a las variables locales establecidas en el programa que las llama. De la misma manera, el programa no puede acceder a las variables locales establecidas en una subrutina.

Las órdenes **Lbl** son componentes locales del programa en que se encuentran. Por tanto, la orden **Goto** del programa que las llama no puede extenderse hasta la etiqueta de una subrutina o viceversa.

Uso de variables en un programa

Los programas emplean variables de forma análoga a como se utilizan en la pantalla Home. Sin embargo, el “ámbito” de las variables afecta a la forma en que se almacenan y se accede a ellas.

Ámbito de las variables

Ámbito	Descripción
Variables del sistema (Global)	Variables de nombre reservado que se crean automáticamente para almacenar información sobre el estado de la TI-89 / Voyage™ 200 PLT. Por ejemplo, las variables de ventana (xmin, xmax, ymin, ymax, etc.) están disponibles de forma global para cualquier carpeta. • Es posible referirse a estas variables utilizando solamente el nombre de las mismas, independientemente de la carpeta que esté en uso. • Los programas no pueden crear variables del sistema, aunque pueden utilizar sus valores y, en la mayoría de los casos, almacenar nuevos valores.

Variables de carpeta (Para obtener información sobre las carpetas, consulte el módulo *Pantalla principal de la calculadora*.)	Variables que se almacenan en determinadas carpetas. • Si se almacena sólo en un nombre de variable, la variable se almacenará en la carpeta actual. Por ejemplo: 5→start • Si sólo se hace referencia a un nombre de variable, dicha variable debe encontrarse en la carpeta actual. De lo contrario, no se encontrará (aun cuando la variable exista en una carpeta distinta). • Para almacenar o hacer referencia a una variable de otra carpeta, será preciso especificar un nombre de camino. Por ejemplo: 5→class\start Nombre de variable Nombre de carpeta Después de interrumpir el programa, las variables de la carpeta creadas en el programa continúan existiendo y ocupando la memoria.

Variables locales	Variables provisionales que sólo existen mientras el programa se está ejecutando. Al interrumpir el programa, las variables locales se borran automáticamente. • Para crear variables locales en el programa, será preciso definirlas utilizando la orden **Local**. • Las variables locales se consideran como únicas aunque exista una variable en la carpeta con el mismo nombre. • Las variables locales son muy útiles para almacenar temporalmente los valores que no se quieren guardar.

Nota: Si el programa incluye variables locales, la gráfica de la función no podrá acceder a ellas. Por ejemplo:

```
Local a
 5→a
 Graph a*cos(x)
```

puede presentar un error o un resultado inesperado (si a es una variable que existe en la carpeta actual).

Errores de definición circular

Al hallar el valor de una función definida por el usuario o ejecutar un programa, puede especificar un argumento que incluya la misma variable que se usó para definir la función o crear el programa. Sin embargo, para evitar errores Circular definition, ha de asignar un valor a las variables x o i que se utilizan para hallar el valor de la función o ejecutar el programa. Por ejemplo:

```
x+1→x
```

— o —

```
For i,i,10,1
  Disp i
EndFor
```

Produce un mensaje de error **Circular definition** si x o i no tienen valor. El error no se produce si x o i ya tuvieran asignado un valor.

Órdenes relacionadas con variables

Orden	Descripción
STO▸ key	Almacena un valor en una variable. Al igual que en la pantalla Home, pulsando STO▸ se introduce el símbolo →.
Archive	Mueve las variables especificadas de la RAM a la memoria de archivo de datos del usuario.
BldData	Permite crear una variable de datos basada en la información gráfica introducida en Y=Editor, Window Editor, etc.
CopyVar	Copia el contenido de una variable.
Define	Define una variable de programa (subrutina) o de función dentro de un programa.
DelFold	Borra una carpeta. Primero deben borrarse todas las variables incluidas en dicha carpeta.
DelVar	Borra una variable.
getFold	Devuelve el nombre de la carpeta actual.
getType	Devuelve una cadena que indica el tipo de datos (**EXPR**, **LIST**, etc.) de la variable.
Local	Establece una o más variables como variables locales.
Lock	Bloquea una variable, de forma que no pueda modificarse o borrarse accidentalmente sin antes desbloquearla.
MoveVar	Desplaza una variable de una carpeta a otra.
NewData	Crea una variable de datos cuyas columnas consisten en una serie de listas.

Orden	Descripción
NewFold	Crea una nueva carpeta.
NewPic	Crea una variable de imagen gráfica basada en una matriz.
Rename	Asigna un nuevo nombre a la variable.
Unarchiv	Desplaza las variables especificadas de la memoria de archivo de datos del usuario a la RAM.
Unlock	Desbloquea una variable bloqueada.

Nota: Las órdenes **Define**, **DelVar** y **Local** se encuentran disponibles en el menú F4 **Var** de la barra de herramientas de Program Editor.

Uso de variables locales en funciones o programas

Las variables locales son variables temporales que sólo existen mientras la función se calcula o el programa se ejecuta.

Ejemplo de variable local

En el siguiente segmento del programa se muestra el bucle **For...EndFor** (descrito posteriormente en este módulo), donde la variable *i* cuenta los bucles. En la mayoría de los casos, la variable *i* sólo se emplea mientras se está ejecutando el programa.

Establece la variable i como local.

```
:Local I
:For i,0,5,1
:  Disp I
:EndFor
:Disp i
```

Consejo: Siempre que sea posible, utilice variables locales para aquellas empleadas exclusivamente en un programa y que no necesiten almacenarse cuando el mismo finalice.

Si establece la variable *i* como local, ésta se borrará automáticamente al interrumpir el programa para no agotar la memoria.

¿Qué produce un mensaje de error Undefined Variable?

Un mensaje de error **Undefined** variable aparece cuando se obtiene el valor de una función definida por el usuario o se ejecuta un programa definido por el usuario que hace referencia a una variable local que no se inicializa (asigna valor).

Este ejemplo es una función multisentencia, en lugar de un programa. Se muestra con saltos de línea, pero normalmente se escribiría el texto en la línea de entrada como una línea continua, como: **Define fact(n)=Func:Local…** donde la elipsis indica que el texto de la línea de entrada continúa fuera de pantalla.

Por ejemplo:

```
Define fact(n)=Func:
Local m:
While n>1:
  n*m→m: n-1→n:
EndWhile:
Return m:
EndFunc
```

A la variable local m no se le asigna un valor inicial.

En el ejemplo anterior, la variable local m existe independientemente de cualquier variable m que, a su vez, exista fuera de la función.

Debe inicializar las variables locales

Todas las variables locales deben tener un valor inicial asignado antes de poder hacerse referencia a ellas.

```
Define fact(n)=Func:
Local m: 1→m: ———— 1 se almacena como valor inicial para m.
While n>1:
  n*m→m: n-1→n:
EndWhile:
Return m:
EndFunc
```

La TI-89 / Voyage™ 200 PLT no puede utilizar una variable local para realizar cálculos simbólicos.

Para realizar cálculos simbólicos

Si desea que un programa o función realice cálculos simbólicos, debe utilizar una variable global en vez de una local. No obstante, debe asegurarse de que la variable no exista ya fuera del programa. Los siguientes métodos pueden ayudarle.

- Haga referencia a un nombre de variable global, habitualmente con uno o más caracteres, que es poco probable que exista fuera del programa o función.
- Incluya **DelVar** en el programa o función para borrar la variable global, si la hubiera, antes de hacer referencia a ella (**DelVar** no borra variables archivadas o inaccesibles).

Operaciones con cadenas

Las cadenas se utilizan para introducir y presentar caracteres de texto. Las cadenas pueden escribirse directamente o almacenarse en variables.

Cómo utilizar las cadenas

Una cadena es una secuencia de caracteres escritos entre “comillas”. En la programación, las cadenas permiten al programa presentar información o solicitan al usuario la realización de una acción. Por ejemplo:

```
Disp “The result is”,answer
```

— o —

```
Input “Enter the angle in degrees”,ang1
```

— o —

```
“Enter the angle in degrees”→str1
Input str1,ang1
```

Algunas órdenes de entrada (como **InputStr**) almacenan automáticamente las entradas del usuario como cadenas y no requieren el empleo de comillas.

No pueden realizarse operaciones matemáticas con los contenidos de las cadenas, aunque en apariencia sean expresiones numéricas. Por ejemplo, la cadena **“61”** representa los caracteres **“6” y “1”**, no el número **61**.

Aunque las cadenas como **“61”** o **“2x+4”** no pueden utilizarse en operaciones, pueden convertirse en expresiones numéricas mediante la orden **expr**.

Órdenes para cadenas

Nota: Consulte el módulo *Referencia t*écnica para la sintaxis de todas las órdenes y funciones de la TI-89 / Voyage™ 200 PLT.

Orden	Descripción
#	Convierte una cadena en un nombre de variable. Se le denomina direccionamiento indirecto.
&	Anexa (concatena) dos cadenas en una.
char	Devuelve el carácter correspondiente a un código de carácter especificado. Es la opuesta de la orden ord.
dim	Devuelve el número de caracteres de una cadena.

Orden	Descripción
expr	Convierte una cadena en una expresión numérica y la ejecuta. Es la opuesta de la orden **string**. **Importante**: Algunas órdenes de entrada del usuario almacenan los valores introducidos como cadenas. Antes de realizar operaciones matemáticas con dichos valores, será preciso convertirlos en expresiones numéricas.
format	Devuelve una expresión como cadena de caracteres basada en la plantilla de formato (fija, científica, ingeniería, etc.)
inString	Busca una cadena para verificar si contiene una subcadena determinada. En caso afirmativo, **inString** devuelve la posición del carácter donde se produce la primera ocurrencia de la subcadena.
left	Devuelve el número de caracteres especificado desde la parte izquierda (comienzo) de una cadena.
mid	Devuelve el número de caracteres especificado desde cualquier posición en la cadena.
ord	Devuelve el código de carácter del primer carácter de la cadena. Es la opuesta de la orden **char**.
right	Devuelve el número de caracteres especificado desde la parte derecha (final) de una cadena.
rotate	Rota los caracteres de una cadena. El valor predeterminado es -1 (rotar un carácter a la derecha).

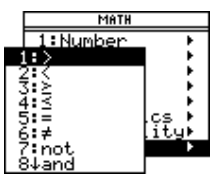
Orden	Descripción
shift	Sustituye por espacios una serie de caracteres de la cadena. El valor predeterminado es -1 (y sustituir por un espacio un carácter a la derecha). Ejemplos: shift("abcde",2)⇒"cde " y shift("abcde")⇒" abcd"
string	Convierte una expresión numérica en cadena. Es la opuesta de la orden **expr**.

Pruebas condicionales

Las pruebas condicionales permiten a los programas tomar decisiones. Por ejemplo, dependiendo de si la prueba es verdadera o falsa, el programa puede decidir cuál de entre dos acciones va a realizar. Las pruebas condicionales se emplean con estructuras de control, como **If...EndIf**, y con bucles, como **While...EndWhile** (descritos más adelante en este módulo).

Introducción de un operador

- Escriba el operador directamente con el teclado.
  — o —
- Pulse 2nd [MATH] y seleccione **8:Test**. A continuación, seleccione el operador en el menú.
  — o —



- Presenta las funciones de built-in.
  Pulse:
  TI-89: CATALOG
  Voyage™ 200 PLT: 2nd [CATALOG].
  La lista de operadores de prueba se muestra cerca de la parte inferior del menú F2 Built-in.

Operadores relacionales

Los operadores relacionales permiten definir una prueba condicional que compara dos valores. Estos números pueden ser números, expresiones, listas o matrices (pero deben coincidir en tipo y tamaño).

Operador	Verdadero si:	Ejemplo
>	Mayor que	a>8
<	Menor que	a<0
≥	Mayor o igual que	a+b≥100
≤	Menor o igual que	a+6≤b+1
=	Igual	list1=list2
≠	Distinto de	mat1≠mat2

Consejo: Puede escribir con el teclado:
>= para ≥
<= para ≤
/= para ≠
Para obtener el carácter /, pulse ÷.

Operadores booleanos

Los operadores booleanos permiten combinar los resultados de dos pruebas distintas.

Operador	Verdadero si:	Ejemplo
and	Ambas pruebas son verdaderas	a>0 and a≤10
or	Al menos una prueba es verdadera	a≤0 or b+c>10
xor	Una prueba es verdadera y la otra falsa	a+6<b+1 xor c<d

La función Not

La función not cambia el resultado de una prueba de verdadero a falso y viceversa. Por ejemplo:

`not x>2` es verdadero si `x≤2`
falso si `x>2`

Nota: Si utiliza **not** en la pantalla Home, en el área de historia aparecerá como ~. Por ejemplo, **not x>2** aparece como **~(x>2)**.

Uso de If, Lbl y Goto para controlar el flujo del programa

La estructura **If...EndIf** se sirve de las pruebas condicionales para decidir si se ejecutan una o varias órdenes. Las órdenes **Lbl** (etiqueta) y **Goto** también pueden utilizarse para trasladarse (o saltar) de una posición a otra en el programa.

Menú F2 Control de la barra de herramientas

Para introducir las estructuras **If...EndIf**, utilice el menú F2 **Control** de la barra de herramientas de Program Editor.

La orden If está directamente disponible en el menú F2.

Para ver un submenú que incluya una lista de otras estructuras If, seleccione **2:If...Then**.

Al seleccionar una estructura como **If...Then...EndIf**, se inserta una plantilla en la posición del cursor.

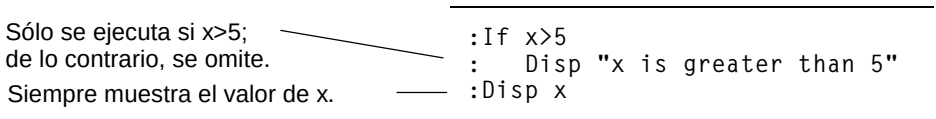
```
:If | Then

:EndIf
```

El cursor está situado de forma que pueda introducir una prueba condicional.

La orden If

Para ejecutar sólo una orden cuando la prueba condicional es verdadera, utilice la forma general:

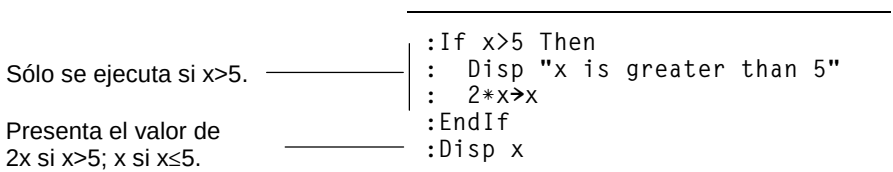


En este ejemplo, antes de ejecutar la orden **If** deberá almacenar un valor en **x**.

Consejo: Utilice el sangrado para facilitar la lectura y comprensión de los programas.

Las estructuras If...Then...EndIf

Para ejecutar varias órdenes cuando la prueba condicional es verdadera, utilice la estructura:



Nota: EndIf marca el final del bloque **Then** ejecutado cuando la condición es verdadera.

Las estructuras If...Then...Else... EndIf

Para ejecutar un grupo de órdenes cuando la prueba condicional es verdadera y otro grupo distinto cuando la condición es falsa, utilice esta estructura:

Sólo se ejecuta si x>5.

Sólo se ejecuta si x≤5

Presenta el valor de:
- 2x if x>5.
- 5x if x≤5.

```
:If x>5 Then
:  Disp "x is greater than 5"
:  2*x→x
:Else
:  Disp "x is less than or
   equal to 5"
:  5*x→x
:EndIf
:Disp x
```

Las estructuras If...Then...ElseIf... EndIf

Una forma más compleja de la orden If permite comprobar una serie de condiciones. Supongamos que el programa solicita al usuario un número que corresponde a una de cuatro opciones. Para comprobar cada opción (**If Choice=1**, **If Choice = 2**, etc.), utilice la estructura **If...Then...ElseIf...EndIf**.

Para obtener más información y ver un ejemplo, consulte el módulo *Referencia t*écnica.

Las órdenes Lbl and Goto

El flujo del programa también puede controlarse mediante las órdenes **Lbl** (etiqueta) y **Goto**.

Utilice la orden **Lbl** para marcar (asignar un nombre a) una posición determinada en el programa.

Lbl *Nombre de etiqueta*

└ nombre que se va a asignar a esta posición (utilice la misma convención que para asignar nombres a variables)

Puede utilizar **Goto** en cualquier parte del programa para trasladarse hasta la posición correspondiente a la etiqueta especificada.

Goto *Nombre de etiqueta*

└ especifica la orden **Lbl** hasta la que se va a trasladar

Dado que la orden **Goto** es incondicional (siempre se traslada hasta la etiqueta especificada), a menudo se utiliza con la orden **If** para definir pruebas condicionales. Por ejemplo:

Si x>5, se traslada directamente hasta la etiqueta GT5.

En este ejemplo, el programa debe incluir órdenes (como **Stop**) que eviten que Lbl GT5 se ejecute si x≤5

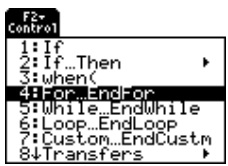
```
:If x>5
:  Goto GT5
:Disp x
:--------
:--------
:Lbl GT5
:Disp "The number was > 5"
```

Uso de bucles para repetir un grupo de órdenes

Los bucles permiten repetir sucesivamente el mismo grupo de órdenes. Se encuentran disponibles varios tipos de bucles, cada uno de los cuales proporciona una forma distinta de finalizarlo, basándose en pruebas condicionales.

Menú F2 Control de la barra de herramientas

Para introducir la mayor parte de las órdenes relacionadas con bucles, utilice el menú F2 **Control** de la barra de herramientas de Program Editor.



Al seleccionar un bucle, la orden de inicio y su correspondiente **End** se insertan en la posición del cursor.

```
:For |
:EndFor
```

Si el bucle requiere argumentos, el cursor se situará después de la orden.

A continuación, puede empezar a introducir las órdenes que se ejecutarán en el bucle.

Nota: La orden del bucle marca el inicio de éste. La orden **End** correspondiente marca su final.

Los bucles For...EndFor

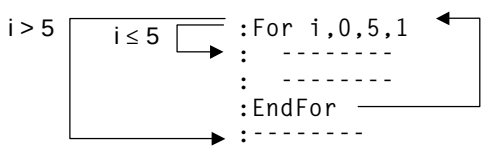
El bucle **For...EndFor** emplea un contador para controlar la cantidad de veces que se repite. La sintaxis de la orden **For** es:

Nota: El valor inicial puede ser inferior al final, pero el incremento debe ser negativo.

For(*variable, inicio, fin* [, *incremento*])

- *incremento*: se añade al contador las veces subsiguientes que se ejecuta **For** (si este valor opcional se omite, el incremento es 1).
- *fin*: sale del bucle cuando *variable* supera este valor
- *inicio*: valor de contador utilizado la primera vez que se ejecuta
- *variable*: variable utilizada como contador

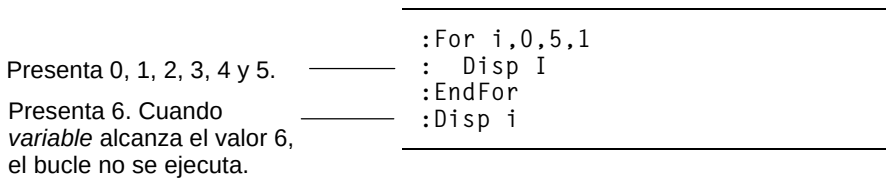
Al ejecutar **For**, el valor variable se compara con el valor fin. Si variable no supera el valor fin, el bucle se ejecuta; de lo contrario, el control del programa saltará a la orden siguiente a **EndFor**.



Nota: La orden **For** incrementa automáticamente la variable contador de forma que el programa pueda cancelar el bucle tras un determinado número de repeticiones.

Al final del bucle (**EndFor**), el control del programa retrocede hasta la orden **For**, donde variable se incrementa y se compara con fin.

Por ejemplo:



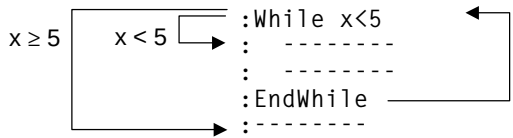
Consejo: Puede definir la variable contador como local siempre que no necesite almacenarla tras interrumpir el programa.

Los bucles While...EndWhile

El bucle **While...EndWhile** repite un bloque de órdenes siempre que la condición especificada sea cierta. La sintaxis de la orden While es:

While *condición*

Al ejecutar While, la condición se calcula. Si condición es verdadera, el bucle se ejecuta; de lo contrario, el control del programa pasará a la orden siguiente a EndWhile.



Nota: La orden **While** no cambia automáticamente la condición. Es preciso incluir órdenes que permitan al programa abandonar el bucle.

Al final del bucle (**EndWhile**), el control del programa retrocede hasta la orden **While**, donde se vuelve a calcular la condición.

Para ejecutar el bucle por primera vez, la condición debe ser verdadera al principio.

- Las variables referidas en la condición deben ajustarse antes que la orden **While**. Los valores pueden generarse en el programa o puede solicitarse al usuario la introducción de los mismos.

- El bucle debe contener órdenes que modifiquen los valores de la condición, permitiendo incluso convertirla en falsa. De lo contrario, la condición será siempre verdadera y el programa no podrá salir del bucle (denominado bucle infinito).

Por ejemplo:

Inicialmente, ajusta x.

Presenta 0, 1, 2, 3 y 4.

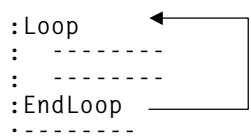
Incrementa x..

Presenta 5. Cuando x toma el valor 5, el bucle deja de ejecutarse.

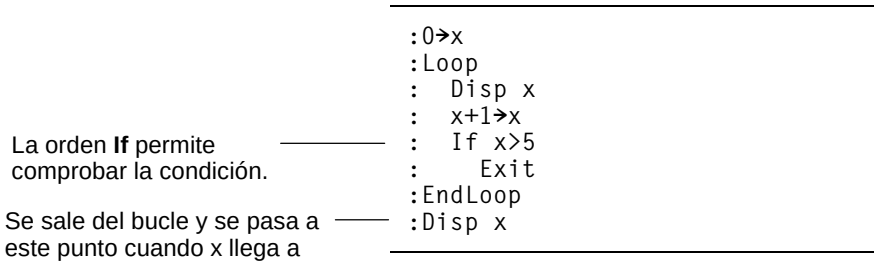
```
:0→x
:While x<5
:  Disp x
:  x+1→x
:EndWhile
:Disp x
```

Los bucles Loop...EndLoop

Loop...EndLoop crea un bucle infinito: se repite un número indefinido de veces. La orden Loop carece de argumentos.



Normalmente, el bucle contiene órdenes que permiten al programa salir del mismo. Las órdenes más utilizadas son: **If**, **Exit**, **Goto** y **Lbl** (etiqueta). Por ejemplo:



Nota: La orden **Exit** permite salir del bucle actual.

En este ejemplo, la orden If puede encontrarse en cualquier parte del bucle.

Si la orden If está:	**El bucle:**
Al principio del bucle	Se ejecuta sólo si la condición es verdadera.
Al final del bucle	Se ejecuta al menos una vez y sólo se repite si la condición es verdadera.

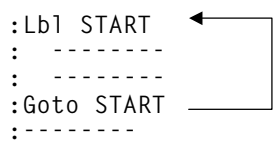
If también puede utilizar una orden **Goto** para transferir el control del programa a una orden **Lbl** (etiqueta) determinada.

Repetición inmediata de un bucle

La orden **Cycle** transfiere inmediatamente el control del programa a la siguiente repetición del bucle (antes de que termine la repetición actual). Esta orden funciona con **For...EndFor**, **While...EndWhile** y **Loop...EndLoop**.

Los bucles Lbl and Goto

Aunque las órdenes **Lbl** (etiqueta) y **Goto** no son estrictamente órdenes de bucle, pueden utilizarse para crear un bucle infinito. Por ejemplo:



Al igual que **Loop...EndLoop**, el bucle debe incluir órdenes que permitan al programa abandonarlo.

Configuración de la TI-89 / Voyage™ 200 PLT

Los programas pueden contener órdenes que modifiquen la configuración de la TI-89 / Voyage 200 PLT. Dado que los cambios de modos son especialmente útiles, el menú **Mode** de la barra de herramientas de Program Editor facilita la introducción de la sintaxis adecuada de la orden **setMode**.

Órdenes de configuración

Orden	Descripción
getConfg	Devuelve una lista con las características de la calculadora.
getFold	Devuelve el nombre de la carpeta actual.
getMode	Devuelve el estado actual del modo especificado.
getUnits	Muestra una lista de las unidades.
setFold	Ajusta la carpeta actual.
setGraph	Establece un formato de gráfico determinado (**Coordinates**, **Graph Order**, etc.).
setMode	Ajusta todos los modos excepto **Current Folder**.
setTable	Ajusta un parámetro de configuración de tabla específico (**tblStart**, **Δtbl**, etc.)
setUnits	Define las unidades predeterminadas de los resultados que aparecen.
switch	Define la ventana activa cuando la pantalla se encuentra dividida o devuelve el número de la ventana activa.

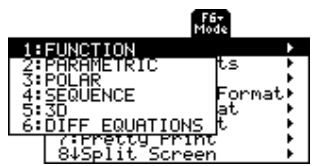
Nota: Las cadenas de parámetro/modo usadas en las funciones **setMode( )**, **getMode( )**, **setGraph( )**, y **setTable( )** no se traducen a otros idiomas cuando se usan en un programa. Consulte el módulo *Referencia t*écnica.

Introducción de la orden SetMode

En Program Editor:

1. Sitúe el cursor donde quiere insertar la orden setMode.

2. Pulse:
   TI-89: 2nd F6
   Voyage™ 200 PLT: F6
   para presentar una lista de modos.



   Nota: El menú Mode no permite ajustar el modo Current Folder. Para esto, utilice la orden setFold.

3. Seleccione un modo para mostrar un menú con los estados válidos.

4. Seleccione un ajuste.

En el programa se inserta la sintaxis correcta.

```
:setMode("Graph","FUNCTION")
```

Uso de órdenes de reloj

Las órdenes enumeradas en esta sección sirven para ejecutar funciones de reloj.

Orden	Descripción
checkTmr	Devuelve un entero que representa el número de segundos transcurridos desde que se inició un temporizador .
	Desactiva el reloj.
ClockOn	Activa el reloj.
dayOfWk	Devuelve un entero del 1 al 7 que representa el correspondiente día de la semana.
getDate	Devuelve una lista con la fecha correspondiente al valor actual del reloj.
getDtFmt	Devuelve un entero que representa el formato de fecha que hay definido en ese momento en el dispositivo.
getDtStr	Devuelve una cadena con la fecha actual.
getTime	Devuelve una lista con la hora correspondiente al valor actual del reloj. La hora se devuelve con formato de 24 horas.
getTmFmt	Devuelve un entero que representa el formato de hora del reloj que hay definido en ese momento en el dispositivo.
getTmStr	Devuelve una cadena con la hora actual del reloj.
getTmZn	Devuelve un entero que representa la zona horaria que hay definida en ese momento en el dispositivo.

Orden	Descripción
isClkOn()	Determina si el reloj está activado o desactivado.
setDate	Ajusta el reloj en la fecha indicada en el argumento y devuelve una lista que representa el valor de fecha anterior.
setDtFmt	Define el formato de fecha del escritorio de acuerdo con el argumento y devuelve el valor del formato de fecha anterior.
setTime	Ajusta el reloj en la hora indicada en el argumento y devuelve una lista que representa el valor de hora anterior.
setTmFmt	Define el formato de hora del escritorio de acuerdo con el argumento y devuelve el valor del formato de hora anterior.
setTmZn	Define el formato de hora de acuerdo con el argumento y devuelve el valor de la zona horaria anterior.
startTmr	Devuelve el valor actual del reloj expresado con un número entero, proporcionando el *tiempoinic* de un temporizador.
timeCnv	Convierte segundos a unidades de tiempo que pueden ser más fáciles de comprender al evaluar.

Solicitud de entradas al usuario y presentación de salidas

Aunque los valores pueden generarse en el mismo programa (o almacenarse antes en variables), éste puede solicitar al usuario que introduzca información durante su ejecución. De la misma forma, el programa puede mostrar información como, por ejemplo, los resultados de una operación.

Menú F3 E/S de la barra de herramientas

Para introducir la mayor parte de órdenes de entrada/salida empleadas habitualmente, utilice el menú F3 **E/S** de la barra de herramientas de Program Editor.

Para ver el submenú con las órdenes adicionales, seleccione **1:Dialog**.

Órdenes de entrada

Orden	Descripción
getKey	Devuelve el código de la siguiente tecla que se pulsa. Vea en el Apéndice A la lsita de los codigos de las teclas.
Input	Solicita al usuario la introducción de una expresión, que se tratará de acuerdo con la forma en que se haya introducido. Por ejemplo: • Las expresiones numéricas se tratan como expresiones. • Las expresiones entre “comillas” se tratan como cadenas. **Input** también presenta la pantalla Graph y permite al usuario actualizar las variables xc e yc (rc y θc en el modo polar) situando el cursor gráfico.
InputStr	Solicita al usuario la introducción de una expresión, que siempre se tratará como cadena. Por tanto, no se precisa el uso de “comillas”.
PopUp	Presenta un cuadro de menú desplegable que permite al usuario seleccionar un elemento.
Prompt	Solicita al usuario la introducción de una serie de expresiones. Al igual que con **Input**, las expresiones se tratan de acuerdo con la forma en que se han introducido.
Request	Presenta un recuadro de diálogo que solicita al usuario la introducción de una expresión. **Request** siempre trata las expresiones introducidas como cadenas.

Consejo: Las entradas de cadenas no pueden emplearse en operaciones matemáticas. Para convertir la cadena en una expresión numérica, utilice la orden expr.

Órdenes de salida

Orden	Descripción
ClrIO	Vacía la pantalla Program E/S.
Disp	Presenta una expresión o cadena en la pantalla Program E/S. **Disp** también permite presentar el contenido actual de la pantalla Program E/S sin mostrar información adicional.
Error! Bookmark not defined.**DispG**	Presenta el contenido actual de la pantalla Graph.
DispHome	Muestra el contenido actual de la pantalla Home
DispTbl	Presenta el contenido actual de la pantalla Table.
Output	Presenta una expresión o cadena empezando por las coordenadas especificadas en la pantalla Program E/S.
Format	Asigna un formato a la presentación de información numérica.
Pause	Interrumpe la ejecución del programa hasta que se pulsa ENTER. De forma opcional, puede mostrarse una expresión durante la pausa. Una pausa permite al usuario leer la salida y decidir en qué momento está listo para continuar.
Text	Presenta un recuadro de diálogo que contiene una cadena de caracteres especificada.

Nota: En los programas, no basta con realizar una operación para que aparezca el resultado. Es preciso utilizar una orden de salida.

Consejo: Tras **Disp** y **Output**, el programa se reanuda inmediatamente. Puede añadir una orden **Pause**.

Órdenes de interfaz gráfica de usuario

Orden	Descripción
Dialog... **EndDlog**	Define un bloque del programa (que consta de órdenes **Title**, **Request**, etc.) que presenta un recuadro de diálogo.
Toolbar... **EndTbar**	Define un bloque del programa (que consta de órdenes **Title**, **Item**, etc.) que sustituye los menús de la barra de herramientas. La nueva barra de herramientas sólo funciona durante la ejecución del programa y sólo hasta que el usuario selecciona un elemento. A continuación, vuelve a mostrarse la barra de herramientas original.
CustmOn... **CustmOff**	Activa o anula la barra de herramientas personalizada.
EndCustm	Define un bloque del programa que presenta una barra de herramientas personalizada cuando pulse 2nd [CUSTOM]. Esta barra de herramientas permanece activa hasta que se vuelve a pulsar 2nd [CUSTOM] o se cambia la aplicación.
DropDown	Presenta un menú desplegable dentro de un recuadro de diálogo.

Orden	Descripción
Item	Presenta un elemento de menú de la barra de herramientas.
Request	Crea un cuadro de entrada dentro de un recuadro de diálogo.
Text	Presenta una cadena de caracteres dentro de un recuadro de diálogo.
Title	Presenta el título de un recuadro de diálogo o menú dentro de una barra de herramientas.

Consejo: Si se ejecuta un programa que configura una barra de herramientas personalizada, ésta se encuentra disponible incluso después de interrumpirlo.

Nota: **Request** y **Text** son órdenes independientes que también pueden utilizarse fuera del recuadro de diálogo o del bloque del programa de la barra de herramientas.

Creación de un menú Custom (Personalizado)

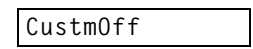
La función de menú personalizado de la TI-89 / Voyage™ 200 PLT permite crear su propio menú de barra de herramientas. Un menú personalizado puede contener cualquier función, instrucción o juego de caracteres disponibles. La TI-89 / Voyage 200 PLT tiene un menú personalizado predeterminado que puede ser modificado o redefinido.

Activación y desactivación del menú Custom

Al crear un menú personalizado, puede permitirse al usuario activarlo o desactivarlo manualmente, o bien dejar que lo haga automáticamente un programa.

Para:	**Realice lo siguiente:**
Activar el menú personalizado	En la pantalla Home o cualquier otra aplicación: • Pulse 2nd [CUSTOM]. En la pantalla Home o en un programa: • Ejecute la orden CustmOn.

Para:	Realice lo siguiente:
Desactivar el menú personalizado	Desde cualquier aplicación: • Pulse [2nd] [CUSTOM] otra vez. — o — • Vaya a otra aplicación. Uso del menú personalizado predeterminado en la pantalla Home: 1. Seleccione el menú Tools. TI-89: [2nd] [F7] Voyage™ 200 PLT: [F7] Después elija **3:CustmOff**. **CustmOff** se pega en la línea de entrada. 2. Pulse [ENTER]. También puede usar **CustmOff** en un programa.

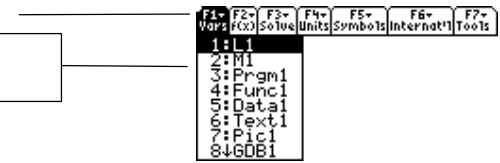


Nota: Cuando se activa el menú personalizado, sustituye al menú normal de la barra de herramientas. A no ser que se haya creado otro menú, se presenta el menú personalizado predeterminado.

Definición de un menú personalizado

Para crear un menú personalizado siga esta estructura general:

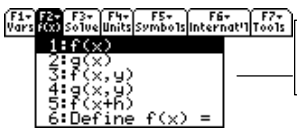
```
:Custom
:  Title título de menú F1
:    Item elemento 1
:    Item elemento 2
:    • • •
:  Title título de menú F2
:    • • •
:  Title título de menú F3
:    • • •
:EndCustm
```

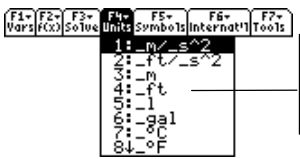


Nota: Cuando el usuario selecciona un elemento de menú, el texto definido por ese comando **Item** se pega en la posición actual del cursor.

Por ejemplo:

Nota: Este menú puede ser ligeramente distinto del menú personalizado predeterminado de su calculadora.





Nota: Observe como "_\o\C" y "_\o\F" aparecen como °C y °F en el menú. Observe también los caracteres acentuados.

```
:Custom

:Title "Vars"
:Item "L1":Item "M1":Item "Prgm1":Item
"Func1":Item "Data1"
:Item "Text1":Item "Pic1":Item "GDB1":Item
"Str1"

:Title "f(x)"
:Item "f(x)":Item "g(x)":Item "f(x,y)":Item
"g(x,y)"
:Item "f(x+h)":Item "Define f(x) ="

:Title "Solve"
:Item "Solve(":Item " and ":Item "{x,y}"
:Item "Solve( and ,{x,y})"

:Title "Units"
:Item "_m/_s^2":Item "_ft/_s^2":Item
"_m":Item "_ft":Item "_l"
:Item "_gal":Item "_\o\C":Item "_\o\F":Item
"_kph":Item "_mph"

:Title "Symbols"
:Item "#":Item "\beta\":Item "?":Item
"~":Item "&"

:Title "Internat'l"
:Item "\e`\":Item "\e'\":Item "\e^\":Item
"\a`\"
:Item "\u`\":Item "\u^\":Item "\o^\":Item
"\c,\":Item "\u..\"

:Title "Tools"
:Item "ClrHome":Item "NewProb":Item
"CustmOff"

:EndCustm

:CustmOn
```

Para modificar el menú personalizado predeterminado, utilice **3:Restore custom default** (como se describe más adelante) para acceder a las órdenes del menú predeterminado. Copie las órdenes, use el Program Editor para crear un programa nuevo y péguelas en el programa en blanco. Tras ello, modifique los programas según convenga.

Nota: Todas las órdenes se insertan en una línea. No es preciso dividirlas en varias líneas.

Puede crear y usar sólo un menú cada vez. Si necesita más, escriba un programa distinto para cada menú personalizado y ejecute el programa del menú que precise.

Restauración del menú personalizado predeterminado

Para restaurar el menú:

1. En el menú normal de la pantalla Home (no en el personalizado), elija **Clean Up**.
   TI-89: 2nd [F6]
   Voyage™ 200 PLT: F6

2. Seleccione **3:Restore custom default**.

   Las órdenes usadas para crear el menú predeterminado se pegan en la línea de entrada.

3. Pulse ENTER para ejecutar las órdenes y recuperar el valor predeterminado.

Cuando restaure el valor predeterminado, los menús anteriores se eliminan. Si el menú anterior se creó con un programa, puede ejecutar el programa de nuevo si desea reutilizar el menú más tarde.

Creación de una tabla o gráfica

Para crear una tabla o gráfica basada en una o varias funciones o ecuaciones, utilice las órdenes que se indican en esta sección.

Órdenes de tabla

Orden	Descripción
DispTbl	Presenta el contenido actual de la pantalla Table.
setTable	Ajusta los parámetros de tabla Graph <–> Table o Independent. Para ajustar los otros dos parámetros de tabla, puede almacenar los valores correspondientes en las variables del sistema tblStart y Δtbl.)
Table	Genera y presenta una tabla basada en una o varias expresiones o funciones.

Órdenes de gráficas

Orden	Descripción
ClrGraph	Borra las funciones o expresiones representadas gráficamente con la orden **Graph**.
Define	Crea una función definida por el usuario.
DispG	Presenta el contenido actual de la pantalla Graph.
FnOff	Anula la selección de todas las funciones Y= (o sólo las especificadas).

Orden	Descripción
FnOn	Selecciona todas las funciones Y= (o sólo las especificadas).
Graph	Representa gráficamente una o varias expresiones concretas utilizando el modo gráfico actual.
Input	Presenta la pantalla Graph y permite actualizar las variables **xc** e **yc** (**rc** y θ**c** en el modo polar) situando el cursor gráfico.
NewPlot	Crea una nueva definición para un gráfico estadístico.
PlotsOff	Anula la selección de todas las representaciones de datos estadísticos (o sólo las especificadas).
PlotsOn	Selecciona todas las representaciones de datos estadísticos (o sólo las especificadas).
setGraph	Modifica los ajustes de varios formatos de gráficos (**Coordinates**, **Graph Order**, etc.).
setMode	Ajusta el modo **Graph**, además de otros modos.
Style	Ajusta el estilo de visualización de una función.
Trace	Permite al programa trazar una gráfica.
ZoomBox – a – **ZoomTrig**	Realiza todas las operaciones de Zoom disponibles en el menú F2 de la barra de herramientas de Y= Editor, Window Editor y la pantalla Graph.

Nota: Para obtener más información sobre el empleo de setMode, consulte Introducción de la orden SetMode.

Órdenes de imagen gráfica y de base de datos

Orden	Descripción
AndPic	Presenta la pantalla Graph y superpone una imagen gráfica almacenada utilizando **AND**.
CyclePic	Anima una serie de imágenes gráficas almacenadas.
NewPic	Crea una variable de imagen gráfica basada en una matriz.
RclGDB	Restablece todos los ajustes almacenados en una base de datos gráfica.
RclPic	Presenta la pantalla Graph y superpone una imagen gráfica almacenada utilizando lógica **OR**.
RplcPic	Vacía la pantalla Graph y presenta una imagen gráfica almacenada.
StoGDB	Almacena los estados del formato gráfico actual en una variable de base de datos gráfica.
StoPic	Copia la pantalla Graph (o una parte rectangular determinada) en una variable de imagen gráfica.
XorPic	Presenta la pantalla Graph y superpone una imagen gráfica almacenada utilizando la lógica **XOR**.

Nota: Para obtener información sobre las imágenes gráficas y las bases de datos, consulte también *Temas complementarios de gráficos*.

Dibujo en la pantalla Graph

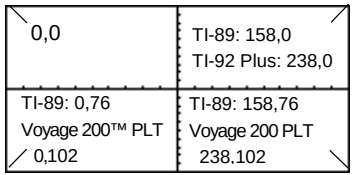
Para crear un objeto de dibujo en la pantalla Graph, utilice las órdenes que se explican en esta sección.

Coordenadas del punto frente a las del pixel

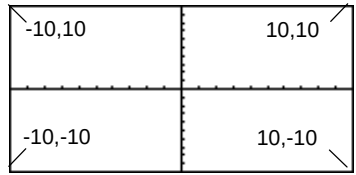
Al dibujar un objeto, puede utilizar cualquiera de los dos sistemas de coordenadas para determinar una posición en la pantalla.

- **Coordenadas del pixel** — Se refieren a los pixels que conforman físicamente la pantalla. Los pixels no dependen de la ventana de visualización, ya que la pantalla tiene siempre:
  TI-89: 159 (0 to 158) pixels wide and 77 (0 to 76) pixels tall.
  Voyage™ 200 PLT: 239 (0 to 238) pixels wide and 103 (0 to 102) pixels tall.

- **Coordenadas del punto** — Se refieren a las coordenadas aplicadas a la ventana de visualización actual (según se haya definido en Window Editor).



Coordenadas del pixel (independientes de la ventana de visualización



Coordenadas del punto (para ventanas de visualización estándar)

Consejo: Para obtener información sobre las coordenadas del pixel en la pantalla dividida, consulte *Data/Matrix Editor*.

La mayor parte de las órdenes de dibujo tienen dos formas, una para las coordenadas del pixel y otra para las del punto.

Nota: Las órdenes de pixel empiezan por **PxI**, como **PxIChg**.

Borrado de objetos dibujados

Orden	Descripción
ClrDraw	Borra todos los objetos dibujados en la pantalla Graph.

Dibujo de un punto o pixel

Orden	Descripción
PtChg or **PxlChg**	Alterna (invierte) un pixel en unas coordenadas determinadas. **PtChg**, que emplea coordenadas de puntos, afecta al pixel más próximo al punto especificado. Si el pixel está desactivado, se activa. Si está activado, se desactiva.
PtOff or **PxlOff**	Desactiva (borra) un pixel en unas coordenadas determinadas. **PtOff**, que emplea coordenadas de puntos, afecta al pixel más próximo al punto especificado.
PtOn or **PxlOn**	Activa (muestra) un pixel en unas coordenadas determinadas. **PtOn**, que emplea coordenadas de puntos, afecta al pixel más próximo al punto especificado.
PtTest or **PxlTest**	Devuelve verdadero o falso para indicar si la coordenada especificada está activa o inactiva, respectivamente.
PtText or **PxlText**	Presenta una cadena de caracteres en las coordenadas determinadas.

Dibujo de rectas y circunferencias

Orden	Descripción
Circle or **PxlCrcl**	Dibuja, borra o invierte una circunferencia que tiene un centro y un radio especificados.
DrawSlp	Dibuja una recta con una pendiente determinada que pasa por un punto.
Line or **PxlLine**	Dibuja, borra o invierte una recta entre dos pares de coordenadas.

LineHorz or **PxlHorz**	Dibuja, borra o invierte una recta horizontal en la coordenada de la fila especificada.
LineTan	Dibuja una recta tangente a la función que se indique, por un punto. Sólo dibuja la recta tangente, no la función.
LineVert or **PxlVert**	Dibuja, borra o invierte una recta vertical en la coordenada de la columna especificada.

Dibujo de expresiones

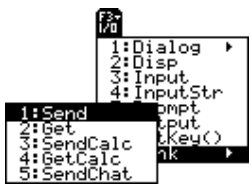
Orden	Descripción
DrawFunc	Dibuja una función.
DrawInv	Dibuja la inversa de la función especificada.
DrawParm	Dibuja una función en paramétricas utilizando expresiones como componentes **x** e **y**.
DrawPol	Dibuja una función en polares.
DrwCtour	Dibuja contornos en modo de gráficos 3D.
Shade	Dibuja dos funciones y muestra sombreadas las áreas para *expresión1* < *expresión2*.

Acceso a otra TI-89 / Voyage™ 200 PLT, a un CBL 2/CBL o a un CBR

La conexión de dos TI-89 / Voyage™ 200 PLT (descrita en *Conexión y actualización*) permite el intercambio de variables entre las dos unidades. Si la TI-89 / Voyage 200 PLT se conecta a un sistema Calculator-Based Laboratory™ (CBL 2™/CBL™), o a un sistema Calculator-Based Ranger™ (CBR), la TI-89 / Voyage™ 200 PLT podrá acceder a los mismos a través de un programa.

Menú F3 E/S de la barra de herramientas

Utilice el menú F3 **E/S** de la barra de herramientas de Program Editor para introducir las órdenes descritas en esta sección.



1. Pulse F3 y seleccione **8:Link**.

2. Seleccione una orden.

Acceso a otra TI-89 / Voyage™ 200 PLT

Al conectar dos TI-89 / Voyage 200 PLT, una actúa de unidad receptora y la otra de unidad transmisora.

Orden	Descripción
GetCalc	Se ejecuta en la unidad receptora. Configura la unidad para recibir una variable a través del puerto E/S. • Después de que la unidad receptora ejecute **GetCalc**, la unidad transmisora debe ejecutar **SendCalc**. • Después de que la unidad transmisora ejecute **SendCalc**, la variable enviada se almacenará en la unidad receptora (en el nombre de variable especificado por **GetCalc**).
SendCalc	Se ejecuta en la unidad transmisora. Envía una variable a la unidad receptora a través del puerto E/S. • Antes de que la unidad transmisora ejecute **SendCalc**, la unidad receptora deberá ejecutar **GetCalc**.
SendChat	Se ejecuta en la unidad transmisora como alternativa general a **SendCalc**. Resulta útil si la unidad receptora es una TI-92 (o para un programa de "charla" general que permita usar una TI-92, TI-92 Plus o una Voyage 200 PLT).

Nota: Para obtener un ejemplo de programa que sincronice las unidades de recepción y transmisión de forma que **GetCalc** y **SendCalc** se ejecuten en la secuencia adecuada, consulte “Transmisión de variables con el control de un programa” en *Conexión y actualización.*

Acceso a un CBL 2/CBL o a un CBR

Para obtener información complementaria, consulte el manual que se adjunta con la unidad CBL 2/CBL o CBR.

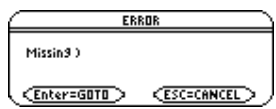
Orden	Descripción
Get	Obtiene una variable del CBL 2/CBL o CBR y la almacena en la TI-89 / Voyage™ 200 PLT.
Send	Envía una lista desde la TI-89 / Voyage 200 PLT hasta el CBL 2/CBL o CBR.

Depuración de programas y tratamiento de errores

Tras escribir un programa, pueden utilizarse varias técnicas para localizar y corregir los errores. En el programa también puede crearse una orden de gestión de errores.

Errores de tiempo de ejecución

El primer paso en la depuración del programa consiste en ejecutarlo. La TI-89 / Voyage™ 200 PLT comprueba automáticamente los errores de sintaxis en las órdenes ejecutadas. Cuando se detecta un error, aparece un mensaje que indica la naturaleza del mismo.



- Para mostrar el programa en Program Editor, pulse ENTER. El cursor aparece en un área cercana al error.
- Para cancelar la ejecución del programa y regresar a la pantalla Home, pulse ESC.

Si el programa permite seleccionar entre varias opciones, asegúrese de ejecutarlo y comprobar cada una de las mismas.

Técnicas de depuración

Los mensajes de error durante el tiempo de ejecución permiten detectar errores de sintaxis, aunque no encuentran errores en la lógica de un programa. Las técnicas siguientes pueden ser de utilidad.

- Durante la prueba, no utilice variables locales, para así poder comprobar los valores de las variables tras la interrupción del programa. Una vez depurado éste, defina las variables que procedan como locales.
- Inserte en el programa, de forma provisional, las órdenes **Disp** y **Pause** para mostrar los valores de las variables importantes.
  - **Disp** y **Pause** no pueden utilizarse en funciones definidas por el usuario. Para convertir temporalmente una función en programa, cambie **Func** y **EndFunc** a **Prgm** y **EndPrgm** y utilice **Disp** y **Pause** para depurar el programa. A continuación, anule **Disp** y **Pause** y vuelva a convertir el programa en función.
- Para confirmar que el bucle se ejecuta el número de veces correcto, presente la variable de contador o las variables incluidas en la prueba condicional.

- Para confirmar la ejecución de la subrutina, presente mensajes como “**Entering subroutine**” y “**Exiting subroutine**” al principio y final de la subrutina.

Órdenes de gestión de errores

Orden	Descripción
Try...EndTry	Define un bloque del programa que permite a éste ejecutar una orden y, en caso necesario, soluciona el error generado por dicha orden.
ClrErr	Borra el estado del error y ajusta el número de la variable del sistema Errornum en cero.
PassErr	Transfiere el error al siguiente nivel del bloque **Try...EndTry**.

Ejemplo: Uso de enfoques alternativos

En el ejemplo de programación del módulo *Matemáticas: Comienzo rápido*, un programa pide al usuario que introduzca un número entero, sume todos los enteros del 1 al que ha introducido y muestra el resultado.

Ejemplo 1

En este ejemplo se emplea **InputStr** para la entrada, el bucle While...EndWhile para calcular el resultado y Text para presentarlo.

Solicita una entrada en la pantalla Program

Convierte la cadena introducida con **InputStr** en una expresión.

Cálculo del bucle.

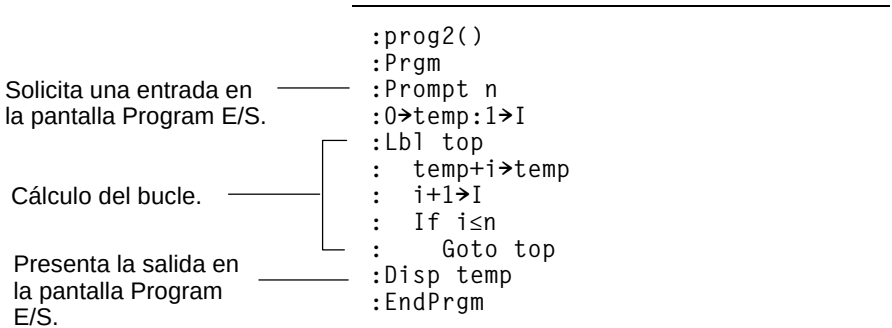
Presenta la salida en un recuadro de diálogo.

```
:prog1()
:Prgm
:InputStr "Enter an integer",n
:expr(n)→n
:0→temp:1→I
:While i≤n
:  temp+i→temp
:  i+1→I
:EndWhile
:Text "The answer is
"&string(temp)
:EndPrgm
```

Consejo: Para obtener ≤, escriba ◆ 0 (cero). Para escribir **&**, pulse:
TI-89: ◆ × (times); Voyage™ 200 PLT: 2nd **H**

Ejemplo 2

En este ejemplo se emplea **Prompt** para la **entrada**, **Lbl** y **Goto** para crear un bucle y **Disp** para presentar el resultado.

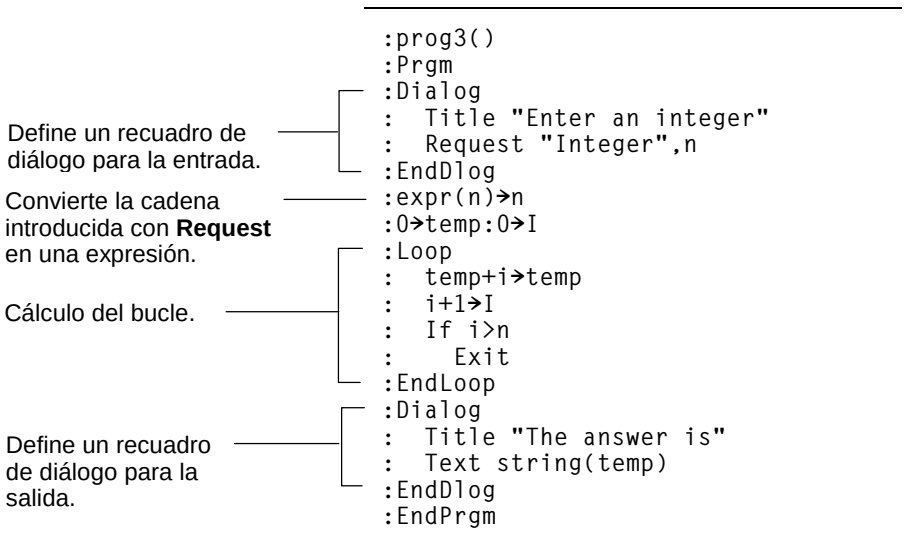



```
:prog2()
:Prgm
:Prompt n
:0→temp:1→I
:Lbl top
:  temp+i→temp
:  i+1→I
:  If i≤n
:    Goto top
:Disp temp
:EndPrgm
```

Nota: Dado que **Prompt** devuelve n como un número, no es preciso utilizar **expr** para convertir **n**.

Ejemplo 3

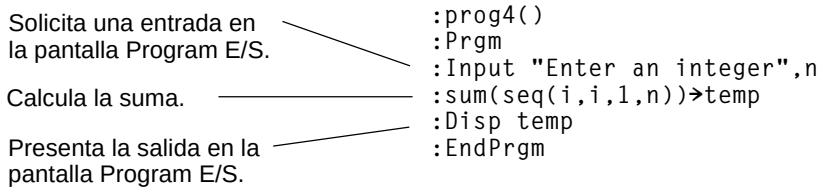
En este ejemplo se emplea **Dialog...EndDlog** para crear recuadros de diálogo para la entrada y la salida. **Loop...EndLoop** se emplea en el cálculo del resultado.




```
:prog3()
:Prgm
:Dialog
:  Title "Enter an integer"
:  Request "Integer",n
:EndDlog
:expr(n)→n
:0→temp:0→I
:Loop
:  temp+i→temp
:  i+1→I
:  If i>n
:    Exit
:EndLoop
:Dialog
:  Title "The answer is"
:  Text string(temp)
:EndDlog
:EndPrgm
```

Ejemplo 4

En este ejemplo se emplean las funciones incorporadas a la TI-89 / TI-92 Plus para calcular el resultado sin utilizar un bucle.



Nota: Dado que **Input** devuelve **n** como un número, no es preciso utilizar expr para convertir **n**.

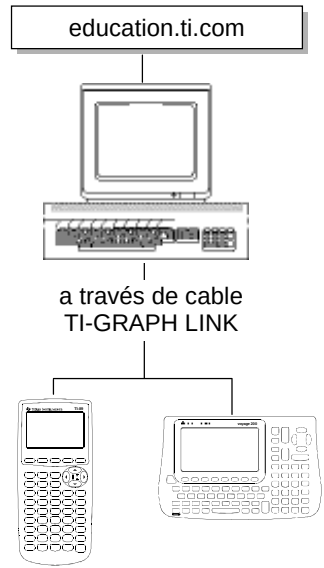
Función	Utilizada en este ejemplo para:
seq	Generar la sucesión de números enteros de **1** a **n**. **seq(***expression*, *var*, *inferior*, *superior* [,*paso*]) incremento de *var* ; si se omite. utiliza 1. valores inicial y final de *var* variable que se va a incrementar expresión utilizada para generar la sucesión
sum	Sum the integers in the list generated by seq.

Programas en lenguaje ensamblador

Con la TI-89 / Voyage™ 200 PLT pueden ejecutarse programas escritos en lenguaje ensamblador. Normalmente, los programas en lenguaje ensamblador se ejecutan más rápido y ofrecen un mayor control que los programas escritos con el Program Editor incorporado.

Dónde conseguir programas en lenguaje ensamblador
Where to Get Assembly-Language Programs

Tanto los programas en lenguaje ensemblador, como el código de otros programas, están disponibles en el sitio web de TI en: education.ti.com.



Los programas disponibles en este sitio proporcionan funciones adicionales o no incorporadas en la TI-89 / Voyage™ 200 PLT. Acceda al sitio web de TI para obtener información actualizada.

Después de descargar un programa del sitio web en el ordenador, utilice TI-GRAPH LINK™ para enviar el programa a la TI-89 / Voyage 200 PLT.

Para obtener información sobre la instalación, consulte las instrucciones de aplicaciones Flash en education.ti.com/guides.

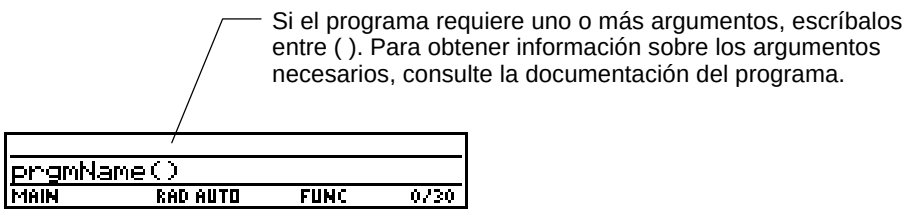
Nota sobre TI-GRAPH LINK

Si tiene un cable TI-GRAPH LINK de ordenador a calculadora y el software correspondiente para la TI-92, tenga en cuenta que el software de TI-92 TI-GRAPH LINK no es compatible con la TI-89, Voyage™ 200 PLT, ni la TI-92 Plus. Sin embargo, el cable funciona con todas las unidades.

Puede adquirir cables de conexión ordenador-calculadora y unidad-unidad en TI Online Store en education.ti.com/buy.

Ejecución de un programa en lenguaje ensamblador

Una vez almacenado el programa en lenguaje ensamblador de la TI-89 / Voyage™ 200 PLT en la unidad, se puede ejecutar el programa desde la pantalla Home con el mismo procedimiento con el que se ejecutaría cualquier otro programa.



Consejo: Si el programa no se encuentra en la carpeta actual, asegúrese de especificar el nombre de ruta.

Puede llamar a un programa en lenguaje ensamblador desde otro programa como una subrutina, eliminarlo o utilizarlo del mismo modo que cualquier otro programa.

Métodos abreviados para ejecutar un programa

En la pantalla Home, puede utilizar métodos abreviados de teclado para ejecutar hasta nueve programas definidos por el usuario o de lenguaje ensamblador. Sin embargo, los programas han de tener los nombres siguientes.

En la pantalla Home, pulse:	Para ejecutar un programa, si lo hay, llamado:
◆ 1	kbdprgm1()
⋮	⋮
◆ 9	kbdprgm9()

Los programas han de guardarse en la carpeta **MAIN**. Además, no puede usarse para ejecutar un programa un método abreviado que requiera argumento.

Si tiene un programa con nombre distinto y quiere ejecutarlo con un método abreviado de teclado, copie o renombre el programa existente como **kbdprgm1()**, etc.

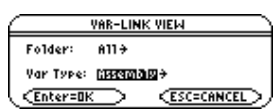
No es posible editar un programa en lenguaje ensamblador

La TI-89 / Voyage™ 200 PLT no puede emplearse para editar un programa en lenguaje ensamblador. El Program Editor incorporado no abre este tipo de programas.

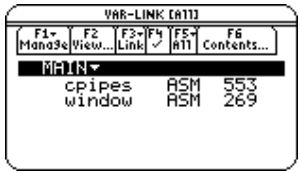
Presentación de una lista de programas en lenguaje ensamblador

Para obtener una lista de los programas en lenguaje ensamblador almacenados en la memoria:

1. Presente la pantalla VAR-LINK (2nd [VAR-LINK]).
2. Pulse F2 **View**.
3. Seleccione la carpeta apropiada (o All las carpetas) y establezca **Var Type = Assembly.**



4. Pulse ENTER para presentar la lista de programas.



Nota: Los programas en lenguaje ensamblador tienen el tipo de datos **ASM**.

Para obtener información sobre la escritura de un programa en lenguaje ensamblador

La información necesaria para enseñar a un programador sin experiencia cómo escribir un programa en lenguaje ensamblador se encuentra fuera del ámbito de este manual. No obstante, acceda al sitio web de TI (education.ti.com) para obtener información específica sobre cómo acceder a las funciones de la TI-89 / Voyage™ 200 PLT.

La TI-89 / TI-92 Plus incluye también una orden Exec que ejecuta una cadena consistente en una serie de códigos op de Motorola 68000. Estos códigos actúan como otra forma de programa en lenguaje ensamblador. Acceda al sitio web de TI para obtener la información disponible.

Nota: Es necesario utilizar un ordenador para escribir programas en lenguaje ensamblador. No es posible generar este tipo de programas con el teclado de la calculadora.

Advertencia: **Exec** permite acceder a las funciones del microprocesador. Tenga presente que es fácil cometer algún error que bloquee la calculadora y provoque la pérdida de datos. Recomendamos realizar una copia de seguridad de los datos contenidos en la calculadora antes de utilizar el comando **Exec**.

Texas Instruments (TI) Información sobre soporte y servicio técnico

Información general

Correo electrónico: ti-cares@ti.com

Teléfono: 1-800-TI-CARES (1-800-842-2737)
Sólo para EE.UU., Canadá, México, Puerto Rico e Islas Vírgenes

Página web: education.ti.com

Consultas técnicas

Teléfono: 1-972-917-8324

Servicio técnico de producto (hardware)

Clientes de EE.UU., Canadá, México, Puerto Rico e Islas Vírgenes: Antes de enviar un producto al servicio técnico, pónganse siempre en contacto con el Soporte al cliente de TI.

Todos los demás clientes: Consulten el prospecto adjunto al producto (hardware) o pónganse en contacto con su concesionario/distribuidor local de TI.

Referencias de página

Este documento PDF contiene marcadores electrónicos diseñados para facilitar el desplazamiento en pantalla. Si decide imprimir este documento, utilice los números de página siguientes para localizar temas específicos.