Computadora Personal

PB-80

MANUAL DEL PROPIETARIO

CASIO®

INTRODUCCION

Este manual contiene explicaciones sobre la computadora que permiten que los principiantes en la programación del lenguaje BASIC como así también aquellos que conocen bien dicho lenguaje y que busquen utilizarlo plenamente, puedan fácil e inmediatamente entender y usar la computadora.

Aquellas personas que son principiantes en la programación en lenguaje BASIC deben leer este manual desde el Capítulo 1 para poder dominar la programación. En especial, deberán leer cuidadosamente el Capítulo 3 que contiene explicaciones sobre los preparativos de la programación y los mandos. En el Capítulo 3 se explica el flujo de un programa. Consulte el Capítulo 4, MANDOS y SENTENCIAS, para obtener explicaciones detalladas sobre los patrones que tienen los mandos.

Aquellas personas que tuvieran conocimientos del lenguaje BASIC podrán utilizar la computadora leyendo el Capítulo 4, MANDOS y SENTENCIAS, después de haber dominado las operaciones fundamentales que se explican en los Capítulos 1 y 2. Las personas que quisieran introducir y usar programas inmediatamente podrán utilizar los programas que aparecen en el capítulo 5 con el nombre de PROGRAMOTECA.

ANTES DE INICIAR LA OPERACION

Esta computadora llega a sus manos después de haber pasado un estricto proceso de pruebas de su avanzada tecnología, así como un estricto control de calidad.
Para estar seguro de que su computadora tendrá una larga duración, por favor observe las siguientes precauciones.

■ **Precauciones para su utilización**

- Como esta computadora está compuesta de partes electrónicas de precisión, no la desarme. Tampoco la golpee o la deje caer, ni la someta a cambios bruscos de temperatura. No la almacene en lugares con altas temperaturas, excesiva humedad o polvo. Cuando la computadora se use en bajas temperaturas, la aparición de la imagen en la pantalla podrá ser lenta e incluso podrá no aparecer. Sin embargo, una vez que se reestablezcan las condiciones de temperatura normal, el funcionamiento de la computadora será normal también.
- Se deberá poner especial atención para no dañar la computadora al doblarla. Por ejemplo, no la lleve en los bolsillos del pantalón.
- Debido a que el símbolo "–" aparece en la pantalla durante los cálculos en los cuales la operación de teclas es inválida, con excepción de ciertas teclas, siempre verifique la pantalla antes de oprimir una tecla.
- Aun cuando en ocasiones la pantalla pierde intensidad cuando el zumbido se produce, esto no es un mal funcionamiento. Pero en caso de que la pantalla fuera muy débil, remplace las pilas por unas nuevas lo antes posible.
- Cambie las pilas cada dos años, aunque no haya usado la computadora mucho tiempo. No deje en la unidad pilas agitadas, ya que las mismas pueden sulfatarse y causar problemas.
- La electricidad estática puede afectar el funcionamiento de su computadora, ya sea alterando el contenido de la memoria o inutilizando el teclado. Cuando ello ocurra, quite las pilas y vuélvalas a colocar como estaban originalmente.
- Para limpiar la computadora, no use líquidos volátiles tales como bencina o rebajador (thinner); en su lugar use un paño seco y suave o bien un paño mojado con una solución de detergente neutro.
- No interrumpa la corriente durante la ejecución de un programa o una operación.
- Debido a que la computadora tiene partes electrónicas de precisión, evite golpearla mientras se ejecuta un programa; de otra manera la ejecución del programa se puede detener o los contenidos de la memoria podrían modificarse.
- Cuando ocurra un mal funcionamiento, pónganse en contacto con la tienda en donde compró la computadora o bien con el distribuidor más cercano.
- Antes de solicitar el servicio, por favor lea una vez más este manual, verifique la alimentación de corriente, revise posibles errores lógicos del programa, etc.

INDICE

CAPITULO 5 PROGRAMOTECA

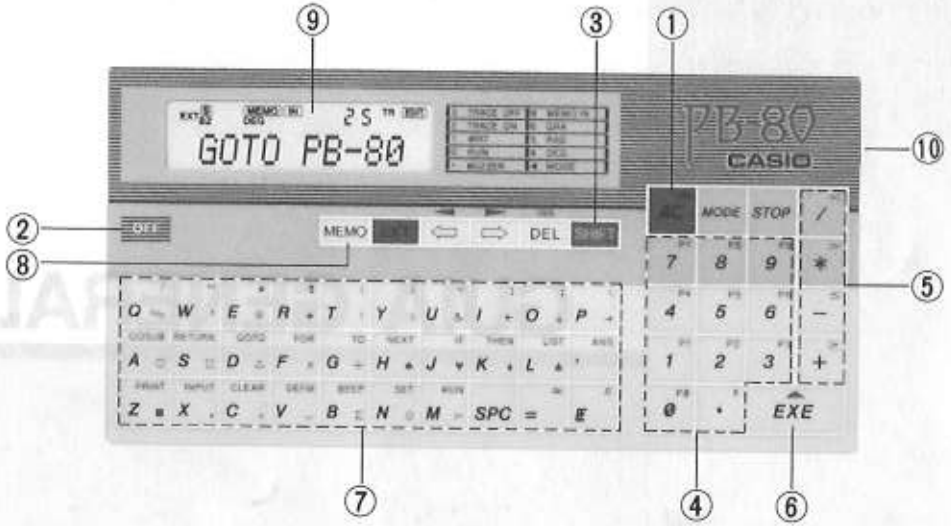
CAPITULO 6 MATERIAL DE REFERENCIA

CAPITULO 1

GUIA GENERAL

Es recomendable que aquellas personas que no hayan usado una computadora así como también aquellas que ya tienen conocimientos lean este capítulo.

1-1. NOMENCLATURA Y OPERACION



① Tecla de encendido/borrado total
② Tecla de apagado
③ Tecla de cambio de símbolo
④ Teclas de números y punto decimal
⑤ Teclas de cálculos
⑥ Tecla de ejecución
⑦ Teclas alfabéticas
⑧ Tecla de memorandum
⑨ Pantalla
⑩ Control del contraste de la imagen de la pantalla

Debido a que las unidades cuentan con teclas adicionales, comparándolas con las calculadoras normales, las teclas de función podrían producir confusión. Por esta razón explicamos cada tecla y su operación.

- **Tecla de encendido/borrado total (AC)**
  Presiónela para encender la unidad. Esta tecla borra todo lo indicado en la pantalla. También se oprime cuando se comete un error o cuando la pantalla se apaga mediante el mecanismo de apagado automático (consulte la página 17). Cuando se está ejecutando un programa, su ejecución se puede suspender oprimiendo esta tecla.

- **Tecla de apagado (OFF)**
  Apaga la unidad.

- **Tecla de cambio de símbolo (SHIFT)**
  Si se pulsa esta tecla, se entra en el modo de mayúsculas y puede visualizarse el mando o símbolo incripto en rojo sobre la tecla. Cuando se oprime una vez más, la modalidad de cambio de las teclas se suspende y desaparecerá "S"

- **Tecla de extensión (EXT)**
  La pulsación de la tecla EXT (aparece la indicación "EXT" en la pantalla) especifica el modo de extensión, el cual permite la entrada de letras minúsculas o de símbolos especiales (inscriptos en marrón). Una segunda pulsación de la misma tecla da por terminado el modo mencionado y hace que desaparezca de la pantalla la indicación "EXT".

- **Teclas de números, punto decimal, cálculos y ejecución.**
  Examine con cuidado este conjunto de teclas. Se trata del mismo grupo que hay en una calculadora normal, ¿no es verdad? Esta sección se usa cuando se hacen los cuatro cálculos aritméticos (suma, resta, multiplicación y división). Pero sin embargo existen las siguientes diferencias: Las teclas de × (multiplicación) y ÷ (división) son diferentes y no existe tecla de = mientras que existe una tecla EXE (ejecución). Esto se debe a que una computadora usa el signo de * (asterisco) para indicar el signo de multiplicación (x) y el de / (barra inclinada) para la división (÷), obteniéndose el resultado por medio de la tecla EXE en lugar de la de = .

Por ejemplo, una operación se realiza en una calculadora normal de la siguiente forma: 12 × 4 ÷ 3 + 7 − 5 = mientras que en una computadora es 12 × 4 / 3 + 7 − 5 EXE.
Esta computadora se puede usar como una calculadora normal tal y como se muestra antes. Una de las teclas de números (0 a 9), cuando es seguida por la tecla SHIFT , se puede usar para especificar un área de programa desde P0 hasta P9, mientras que la tecla . se usa para cálculos de potencias ($x^y \rightarrow x\uparrow y$) y las teclas + − × / para introducir operadores de relaciones ($\geqq, \leqq, >, <$).
También se presiona al final de cada entrada de línea en el modo WRT. Durante la edición de un programa (usando el comando LIST), presionar la tecla SHIFT EXE para moverse a la línea de programa inmediatamente anterior a aquella que está siendo visualizada.

< P7 P8 P9 >
P4 P5 P6 ≦
P1 P2 P3 ≧
P0 ↑ EXE

- **Teclas alfabéticas y de espacio**

Q W E R T Y U I O P
A S D F G H J K L "
Z X C V B N M SPC = E

Usando estas teclas se pueden introducir mandos o bien elaborar programas. Cada de las 26 teclas alfabéticas, desde A hasta Z , funciona como una memoria (para localización de almacenamiento).
Además, las teclas A — Z tienen otras funciones. Cuando se oprimen después de la tecla SHIFT , aparecerá un símbolo de un mando en BASIC. Oprima la tecla SPC cuando necesite un espacio.

Ejemplo) SHIFT A → GOSUB 、 SHIFT U → ?

! " # $ ( ) ? : ; ¥
GOSUB RETURN GOTO FOR TO NEXT IF THEN LIST ANS
PRINT INPUT CLEAR DEFM BEEP SET RUN SPC ≠ π

Además de esto, las teclas alfabéticas tienen otro uso en la modalidad de expansión (cuando la tecla EXT es oprimida, "EXT" aparece en la pantalla). Cuando son oprimidas directamente, los caracteres alfabéticos en minúsculas aparecerán en la pantalla, y cuando se oprimen después de la tecla SHIFT , entonces aparecerán símbolos especiales.

Funciones de modalidad de expansión:

q w e r t y u i o p
a s d f g h j k l "
z x c v b n m SPC = E

Funciones cuando se oprime una tecla después de la tecla SHIFT en la modalidad de expansión:

% " @ ¥ { } & ← ↓ →
○ □ △ × ÷ ♠ ♥ ♦ ♣ ANS
■ • ◦ — Σ Ω μ SPC ≠ π

Para salir del modo de extensión, pulse nuevamente la tecla EXT .

- **Tecla del símbolo igual (=)**
  Esta tecla no se usa para obtener el resultado de algún cálculo, se usa en cambio para establecer una sentencia de asignación (consulte la página 43) y una condición en una sentencia IF (consulte la página 68).
  Asimismo, cuando esta tecla se oprime después de la tecla SHIFT , el símbolo ≠ (no igual) aparecerá en la pantalla.

- **Tecla exponencial / Pi (π)**
  Cuando esta tecla se oprime directamente, se usa para proporcionar un exponente. Por ejemplo, oprima 1 . 2 3 E 4 para 1.23×10^4. Cuando un exponente es un número negativo, oprima la tecla − después de esta tecla. Por ejemplo, oprima 7 . 4 1 E − 9 para 7.41×10^{-9}.
  Cuando se oprime esta tecla después de haber oprimido la tecla SHIFT , aparece en la pantalla Pi (el cociente de la circunferencia de un círculo por su diámetro).

- **Tecla de resultados ([·])**
  Cuando se oprime esta tecla después de haber oprimido la tecla [SHIFT] , aparece en la pantalla el resultado del cálculo manual o de un programa ejecutado inmediatamente antes.

- **Tecla de modalidad ([MODE])**
  Esta tecla se usa junto con las teclas [·] y [0] a [9] para especificar la modalidad de la computadora o la unidad de ángulo.

[MODE][·]..... Esta activa y desactiva el sonido del zumbador de entrada de teclas. Cuando el zumbador esta activado, el símbolo "BUZZER" aparece en la visualización.
[MODE][0]..... "RUN" aparece en la pantalla para la ejecución de cálculos manuales y de programas.
[MODE][1]..... "WRT" aparece para realizar la grabación, verificación y edición de un programa.
[MODE][2]..... "TR" aparece en la pantalla para llevar a cabo la ejecución (Vea la página 63 para los detalles).
[MODE][3]..... Cuando "TR" aparece en la pantalla, la modalidad de ejecución secuencial se cancela y "TR" desaparece de la pantalla.
[MODE][4]..... "DEG" aparece en la pantalla para indicar que se han especificado grados como unidad de ángulo.
[MODE][5]..... "RAD" aparece en la pantalla para indicar que se han especificado radianes como unidad de ángulo.
[MODE][6]..... "GRA" aparece en la pantalla para indicar que se ha especificado gradianes como unidad de ángulo.
[MODE][9]..... "[MEMO] [IN] " aparece en la pantalla para indicar la modalidad de introducción para la función de Banco de Datos (vea la página 28). Para cancelar esta modalidad, oprima [MODE] [0] .

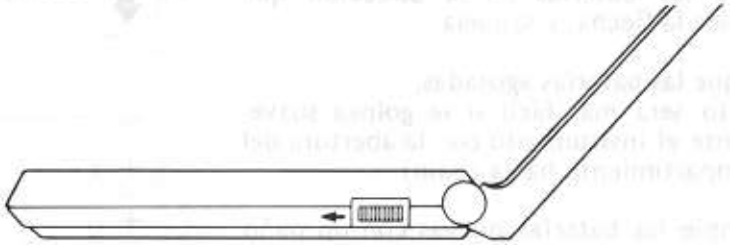
- **Tecla del Banco de Datos [MEMO]**
  Oprímala para usar la función de Banco de Datos. También se usa para volver a llamar los datos en secuencia o un carácter determinado que se haya oprimido, en la modalidad RUN (oprima [MODE] [0]) o en la modalidad de introducción (oprima [MODE] [9]).

- **Teclas del cursor ([◄] [►])**
  Estas teclas se usan para desplazar el cursor ("—" centelleando en la pantalla) hacia la izquierda o la derecha con el objeto de corregir un carácter presente en la pantalla. Si se oprimen una sola vez, el indicador se desplaza un carácter y si se mantienen oprimidas, el indicador se desplaza continuamente dentro del rango de los caracteres presentes en la pantalla.

- **Tecla de Eliminación/Inserción ([INS])**
  Esta tecla se usa para eliminar el carácter en donde esté colocado el cursor. Después de la eliminación, el carácter que está a la derecha del cursor se desplazará hacia la izquierda. Cuando se oprime esta tecla después de haber oprimido la tecla [SHIFT] , el carácter en donde esté colocado el cursor se desplazará hacia la derecha para dejar un espacio.

- **Tecla de detención ([STOP])**
  Si se oprime esta tecla durante la ejecución de un programa, ésta se detiene temporalmente. Para proseguirla, oprima la tecla [EXE] .

- **Control del contraste de la pantalla**
  El contraste de la pantalla varía según la condición de las pilas o el ángulo desde donde se mire la pantalla. Ajústelo por medio de la perilla que se encuentra en el lateral derecho de la computadora.



La pantalla se oscurece cuando el control se mueve en la dirección de la flecha, y se aclara cuando se mueve en la dirección contraria. Si la pantalla sigue siendo débil aun después de haber colocado este control en la posición de oscuridad máxima, quiere decir que las pilas se han agotado y deberán remplazarse por unas nuevas.

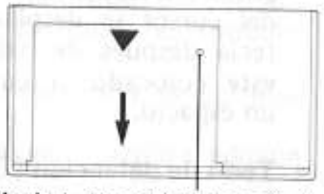
1-2. FUENTE DE ENERGIA

La fuente de la corriente de la computadora es dos pilas de litio (CR2032). Para detalles sobre la duración de las pilas, ver la página 168. Sin embargo, se reduce en caso de que se use frecuentemente el zumbador. Si la pantalla estuviera débil aun cuando hubiera sido ajustado el contraste de ella (consulte la página 15), esto se deberá a que las pilas están bajas por lo que deberán remplazarse lo antes posible. Siempre que lo haga coloque dos pilas nuevas en la misma ocasión.

* Remplace las pilas, siempre nuevas, cada dos años aun si no estuvieran usadas debido a que pudieran tener fugas.

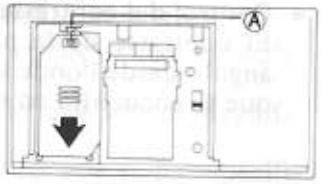
■ Reemplazo de las baterías

(1) Una vez apagado el aparato, deslice el panel posterior y quítelo.

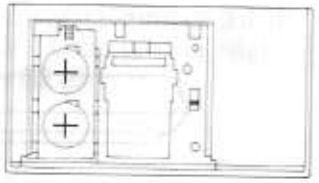


Botón ALL RESET (de reposición)
(Presiónese con un objeto punteagudo cada vez que se completa el reemplazo de las baterías).

(2) Al mismo tiempo que se presiona Ⓐ, deslícese la tapa del compartimiento para las baterías en la dirección que indica la flecha, y sáquela.



(3) Saque las baterías agotadas.
(Esto será más fácil si se golpea suavemente el instrumento con la abertura del compartimiento hacia abajo).



(4) Limpie las baterías nuevas con un paño seco, y colóquelas con el positivo ⊕ hacia arriba.

(5) Deslice la tapa para cerrar el compartimiento haciendo leve presión sobre las baterías.

(6) Vuelva a colocar el panel posterior en su posición original y pulse el botón ALL RESET.

* No incinere las baterías usadas, ya que corre peligro de explosión.
* Asegúrese de poner correctamente los terminales ⊕ y ⊖.
* Mantenga las pilas alejadas de los niños.
  Si se las tragan consulte a un médico inmediatamente.

■ Suspensión automática de la corriente

La suspensión automática de la corriente evita desperdicio de energía innecesario cuando usted olvida suspender la corriente. La corriente se suspende automáticamente 6 minutos después de la opresión de una tecla (excluyendo cálculo de programas). En este caso, la unidad puede encenderse pulsando la tecla [ON].

* Aun cuando el contenido de la memoria no se borra cuando la corriente se suspende, las especificaciones de ángulo y valor modal ("RAD", "WRT", "TR", etc.) se suspenden.

1-3. MODULO DE EXPANSION DE LA MEMORIA RAM (OPCIONAL)

Al área (de memoria) RAM estándar de esta unidad tiene 544 pasos/26 memorias. No obstante, este límite puede ampliarse hasta 1.568 pasos/222 memorias mediante el uso del módulo de memoria RAM opcional OR-1E.
Esta memoria RAM de expansión se utiliza del mismo modo que la RAM estándar, permitiendo aumentar la capacidad de memorias y, consecuentemente, el número de pasos disponibles (remitirse a la página 19).

■ **Instalación del módulo RAM**

(Preparativos)

Como los circuitos internos pueden dañarse por la electricidad estática, antes de tomar contacto con el módulo, descargue toda la electricidad estática de su cuerpo a través de algún objeto metálico, como ser un grifo, etc.

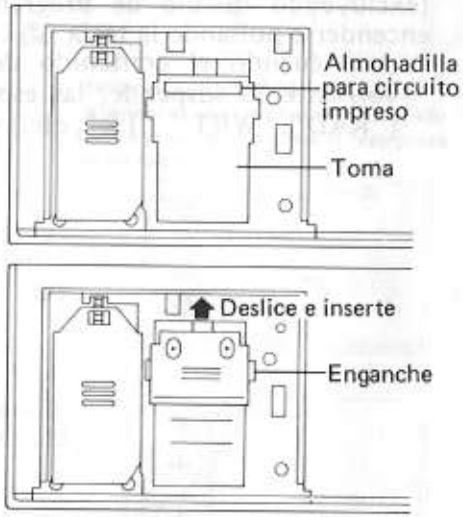
(Procedimiento)

(1) Apague la unidad.

(2) Deslice el panel posterior y quítelo.

(3) Inserte el módulo en la toma del cuerpo de la computadora y deslice el enganche a su posición de traba.
*No toque ni el conector del módulo RAM ni la tarjeta de circuito impreso que se encuentra dentro de la computadora.

(4) Vuelva a colocar el panel posterior en su posición original y pulse el botón ALL RESET.



- Siempre que coloque o saque el módulo opcional RAM, no olvide de presionar el botón ALL RESET con algún objeto punteagudo después de encender la unidad.
  Si no se presiona este botón, es posible que cambie el contenido de la memoria o se visualicen símbolos sin sentido alguno.
- Tenga cuidado que no caiga polvo o tierra sobre la porción del conector del módulo y en la tarjeta de circuito impreso de la computadora. Evite también tocar éstos con los dedos, ya que podría provocar alguna desconexión.
- No olvide colocar el módulo en su estuche y guardarlo en un sitio sin polvo después de haberlo usado.

1-4. EXPANSION DE LA MEMORIA

El número normal de memorias (variables) es de 26. En este caso, el número de pasos es de 544.
El número máximo de memorias estándar es de 94. Mediante el uso del módulo RAM, este límite puede extenderse hasta 222. Para expandir la memoria, los pasos de programa se convierten en memoria, usándose 8 pasos por cada una de ellas.

Número de memorias	Número de pasos de programa	
	Estándar	Expansión
26	544	1568
27	536	1560
28	528	1552
⋮	⋮	⋮
46	384	1408
⋮	⋮	⋮
94	0	1024
⋮	⋮	⋮
200	–	176
⋮	⋮	⋮
222	–	0

La expansión de la memoria se lleva cabo por unidades de a 1 usando el comando DEFM.

Ejemplo:
Expansión a 56 memorias.

Operación:
Seleccionar el modo RUN (presionando [MODE][RUN]) o el modo WRT (presionando [MODE][1]).

DEFM 30 [EXE]

```
***VAR:56
```

* El comando DEFM podrá entrarse presionando [D][E][F][M] ó [SHIFT][9].

Este comando DEFM se utiliza también para verificar el número de memorias designadas.

Ejemplo:
56 memorias designadas en total.

DEFM [EXE]

```
***VAR:56
```

- Cuando se encuentran en uso muchos pasos de programa, para proteger dicho programa, si se intenta una designación mayor que la que permite el número de pasos, ocurre un error (ERR1 ..... número de pasos insuficiente).
- La variable de carácter exclusiva ($) no se cuenta en la designación, ya que es una memoria especial.

1-5. ANTES DE HACER CALCULOS

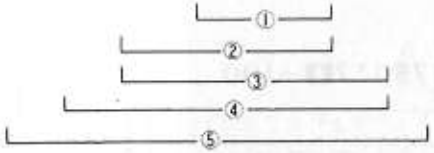
■ **Secuencia de prioridad de cálculos**

Los cálculos tienen reglas de "secuencia de prioridad" por las cuales las multiplicaciones y divisiones se realizan primero en relación con las restas y sumas. Esta computadora cuenta con una función que automáticamente distingue la prioridad en una secuencia. Esta función es tan conveniente que se puede obtener un resultado aun si se introduce una expresión de cálculo tal y como es.
La prioridad en el cálculo se determina de la siguiente manera.

① Funciones (SIN, COS, etc.)
② Potencias (↑)
③ × (∗), ÷ (/)
④ +, −

Aun cuando los cálculos se realizan de acuerdo con esta secuencia de prioridad, si dos operaciones tuvieran la misma prioridad, la izquierda tendrá entonces preferencia. Si se usan paréntesis, las operaciones que estén dentro de ellos tendrán prioridad.

Ejemplo) 2 + 3 ∗ SIN (17 + 13) ↑2 = 2.75



■ Introducción y extracción de digitos y operación de digitos

El número de dígitos que se pueden introducir son 12 para una mantisa y 2 para un exponente. Las operaciones internas también se realizan usando 12 dígitos para la mantisa y 2 dígitos para un exponente.
Aun cuando el número de dígitos que se pueden extraer es generalmente de 10 para una mantisa, difiere dependiendo de los resultados en la pantalla de un cálculo manual y del programa de cálculo. En el caso de cálculos manuales, el resultado aparecerá hasta con 12 dígitos, incluyendo la mantisa, el exponente y el signo de menos. En el caso de cálculos por programas, aparecerán 10 dígitos para la mantisa y 2 dígitos para exponentes. Sin embargo, si se sobrepasará de 12 dígitos, aparecerán 12 dígitos desde el principio, y en seguida el resto aparecerá en forma secuencial desplazándose hacia la izquierda.

Ejemplo)

Cálculos manuales

1⊡2345678912 [EXE]	1.234567891
12345678912[×]100[EXE]	1.2345678 E12
12345678912[×]−100[EXE]	−1.234567 E12

Cálculos del programa

para PRINT 12345678912[×]−100

	−1.234567891 E12	Cambia automáticamente
−	1.234567891E 12	
−1	.234567891 E1 2	
−1.	234567891 E12	

Desaparece de la pantalla

No ha aparecido en la pantalla

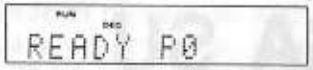
CAPITULO 2

PONGA SU COMPUTADORA EN FUNCIONAMIENTO

Si bien la teoría es importante en el estudio de la informática, más lo es la práctica. No hay nada mejor para familiarizarse con una computadora que justamente usarla. Veamos entonces, en las páginas que siguen, cómo acostumbrarnos al uso de su computadora.

2-1. MANOS A LA OBRA

Para comenzar, encienda su unidad deslizando el interruptor de la misma que se encuentra en el lateral derecho. Al hacerlo, el siguiente mensaje aparece en la pantalla.



Probemos, a continuación, pulsando la tecla [AC]. Esta hace desaparecer el mensaje previamente visualizado, en lugar del cual aparece en el extremo izquierdo de la pantalla un guión destellante llamado "cursor".

La aparición de este cursor destellante en la pantalla indica que la computadora se encuentra en un "estado de espera de entrada", durante el cual Vd. podrá entrar desde el teclado los caracteres que desee. Este cursor aparece en la pantalla bajo dos formatos. Uno de ellos es el visto en la figura anterior. El otro tiene la forma de un cuadrado que destella y aparece en la pantalla a modo de advertencia. Para darle instrucciones a la computadora, las mismas se entran desde el teclado y los caracteres correspondientes a las teclas pulsadas van apareciendo en la pantalla. En este caso, sólo pueden entrarse hasta 62 caracteres por línea (o renglón, por así llamarlo). Es justamente cuando uno entra el 56º carácter de una línea que aparece el cursor cuadrado, para avisarle que esta por llegar al límite de la capacidad de la línea.
En la pantalla aparecen también otros mensajes que indican las diferentes condiciónes y modalidades de funcionamiento. "RUN" indica el estado durante el cual se pueden ejecutar cálculos y programas. Tres diferentes mensajes indican la unidad angular en curso. Ellos son "DEG" para los grados (se especifica automáticamente al encender la unidad); "RAD" para los radianes (se especifica pulsando [MODE][5]); y "GRA" para los gradientes (se especifica pulsando [MODE][6]). Las unidades angulares se definen según sea necesario siempre que se hacen cálculos con funciones trigonométricas.
Las modalidades de funcionamiento también se visualizan en la pantalla. "WRT" corresponde al modo de entrada y se especifica pulsando [MODE][1]; "TR" corresponde al modo de ejecución secuencial y se especifica pulsando [MODE][2] (página 63); " [MEMO][IN] " corresponde al denominado "modo de memorándum electrónico", el cual se especifica pulsando [MODE][9] (página 28); y por último, "EXT" corresponde al llamado "modo de extensión", el cual se especifica por medio de [EXT]. El uso

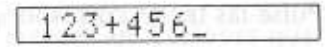
de su computadora la ayudará a familiarizarse con todas estas indicaciones que aparecen en la pantalla.
Veamos, entonces, a continuación algunos ejemplos prácticos.
En caso de que su computadora se bloquee en determinado modo de funcionamiento (o sea que, no puede pasar a otra modo sea cual fuere la tecla que pulse), apague la misma y vuelva a encenderla nuevamente.
Comencemos con un cálculo simple.

Ejemplo) 123+456=579

Presione [AC].

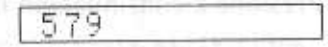
Pulse las teclas correspondientes a los números que componen la expresión numérica.

[1][2][3][+][4][5][6]



Luego, pulse la tecla [EXE] (y no [=]) para obtener el resultado.

[EXE]



¿Obtuvo el resultado esperado? Si así fuere, continuemos con otro ejemplo.

Ejemplo) 45×6+89=359

Supongamos que cometemos un error, y entramos el número 46 en lugar del 45 que aparece en la expresión.

[4][6][×][6][+][8][9]

Lleve entonces el cursor hasta la posición por debajo del número equivocado mediante la tecla [←].

[←][←][←][←][←][←]

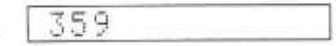
Parpadean el cursor y el 6.

Pulse entonces la tecla correspondiente al número correcto, [5] en este caso.

[5]

Una vez corregida la expresión numérica mediante este procedimiento, estamos en condiciones de obtener el resultado. Pulse para ello la tecla [EXE].

[EXE]



Del mismo modo que en el caso anterior, las teclas con las flechas permiten llevar el cursor a la posición deseada con fines de corrección, siempre que no se haya pulsado la tecla [EXT] con anterioridad. En caso de que se haya pulsado esta última tecla, vuelva a entrar la expresión nuevamente.
Con estos dos ejemplos, vimos la entrada de números desde el teclado. Veamos a continuación la entrada de caracteres alfabéticos (o sea, letras). Para la entrada de estos caracteres, se utilizan las teclas alfabéticas que forman parte del teclado. El teclado tiene una distribución tipo ASCII, universal para casi todas las computadoras personales y muy similar al de las máquinas de escribir convencionales.
Veamos algunos ejemplos.

Ejemplo) Entrada de "ABCXYZ".

Pulse las teclas correspondientes a ABC.

[A] [B] [C]

```
ABC_
```

Presione a continuación las correspondientes a XYZ.

[X] [Y] [Z]

```
ABCXYZ_
```

Intentemos ahora insertar un espacio entre ABC y XYZ.
Desplace el cursor hasta la posición por debajo de la X.

[←] [←] [←]

```
ABCXYZ
```

Abra un espacio pulsando:

[SHIFT] [INS DEL]

```
ABC_XYZ
```

Siempre que sea necesario insertar espacios entre caracteres, lleve el cursor hasta la posición donde desea insertar el espacio y presione entonces [SHIFT] [INS DEL] . Repita el mismo procedimiento para cada espacio adicional que quiera insertar.

Además de los caracteres alfanuméricos, esta computadora posee ciertos caracteres especiales, los cuales pueden utilizarse en juegos o como símbolos científicos. En la página 13 aparece una lista completa de los mismos.
Veamos a continuación el método para entrar estos caracteres especiales.

Ejemplo) Visualización de los símbolos ♤♡◇♧ .

Especifique primeramente el modo de extensión.

En pantalla

[AC] [EXT]

```
EXT
_
```

Para visualizar los caracteres especiales rotulados por encima de ciertas teclas, pulse la tecla correspondiente mientras presiona la tecla [SHIFT].

[SHIFT] [♠] [SHIFT] [♥] [SHIFT] [♦] [SHIFT] [♣]

```
♠♥♦♣_
```

Ejemplo) Visualización de los símbolos Σ , Ω y μ .

El modo de extensión ya ha sido especificado, por lo que será suficiente con que pulse las teclas a continuación:

[SHIFT] [Σ] [SHIFT] [Ω] [SHIFT] [μ]

```
♠♥♦♣ΣΩμ_
```

Con la práctica podrá ir dándose cuenta de las diferentes aplicaciones que podrá darle a estos caracteres especiales en sus programas. Volvamos, entonces, al modo en el cual se pueden entrar caracteres alfanuméricos. Pulse para ello la tecla [EXT].
Es probable que, durante la ejecución de los ejemplos hasta aquí citados, la computadora haya visualizado el mensaje "ERR2" y que el teclado deje de funcionar. La aparición de este mensaje no se debe a una avería de la máquina, sino a alguna equivocación en la entrada desde el teclado. En dicho caso, pulse la tecla [AC] y vuelva a repetir la operación. Para mayores detalles al respecto, sírvase remitirse a la página 57 y 162.

2-2. CONVENIENTE "BANCO DE DATOS"

Esta computadora incorpora un conveniente "banco de datos electrónico", el cual permite el almacenamiento y la visualización de información mediante el simple uso de la tecla [MEMO].
Entre las tantas aplicaciones que puede dársele a este banco de datos electrónico, podríamos mencionar la creación de listas de números telefónicos, itinerarios, tablas, etc.
Por otro lado, en vista de que el lenguaje BASIC de esta computadora también tiene acceso a este banco de datos, serviría también para la creación de listas de clientes y de productos, para llevar cabo cálculos de estimación o para preparar listas de libros.
Este banco de datos electrónico tiene una infinidad de usos además de los aquí mencionados. El "Manual de referencia del Banco de Datos" le dará algunas ideas adicionales para la aplicación del mismo.

2-3. OPERACIONES ARITMÉTICAS FUNDAMENTALES

Veamos en este apartado la realización de las operaciones aritméticas fundamentales. Para quienes nunca hayan usado una calculadora científica, cabe destacar que esta máquina computa en base a una "lógica algebraica real", según la cual la multiplicación y la división tienen prioridad sobre la suma y la resta.

Ejemplo 1) 23+4.5−53=−25.5
Operación) [2][3][+][4][.][5][−][5][3][EXE] → -25.5

* De aquí en más, las teclas numéricas aparecen sin su recuadro.

Ejemplo 2) 56×(−12)÷(−2.5)=268.8
Operación) 56 [×][−] 12 [÷][−] 2.5 [EXE] → 268.8

* Para los números negativos, pulse la tecla [−] antes de entrar el mismo.

Ejemplo 3) 7×8−4×5=36
Operación) 7 [×] 8 [−] 4 [×] 5 [EXE] → 36

* Las multiplicaciones se llevan a cabo antes que la resta.

Ejemplo 4) $(4.5\times10^{75})\times(-2.3\times10^{-78})=-0.01035$
Operación) 4.5 [E] 75 [×][−] 2.3 [E][−] 78 [EXE] → -0.01035

* Para el exponente, pulse la tecla [E] previo a la entrada del mismo.

En cálculos algebraicos, por lo general, se utiliza cierto valor numérico fijo, como en los ejemplos a continuación:

$$3x+5=$$
$$4x+6=$$
$$5x+7=$$

Supongamos que la x que aparece en estas tres expresiones algebraicas tiene un valor fijo igual a 123,456. ¿No le gustaría poder simplificar la entrada de esta número cada vez que aparece en las expresiones? Pues, el uso de una porción de la memoria, la cual llamaremos "variable", permite hacerlo. El procedimiento consistiría en asignar el valor deseado a la variable x, 123,456 en nuestro caso.

[×][−] 123.456 [EXE]

En esta asignación, el signo "=" no tiene el significado de igualdad empleado en matemáticas. En cambio, se utiliza para asignar a una variable (a la izquierda del signo) un determinado valor o el resultado de un expresión (a la derecha del signo). Una vez asignado el valor a x, lleve a cabo los cálculos del modo visto anteriormente.

```
3 [x] [+] 5 [EXE]
4 [x] [+] 6 [EXE]
5 [x] [+] 7 [EXE]
```

```
375.368
499.824
624.28
```

Esta computadora dispone de 26 variables, desde la A hasta la Z, en las cuales se pueden almacenar valores numéricos a conveniencia.
En estos ejemplos, la variable x tiene un valor fijo pero las expresiones algebraicas son diferentes. ¿Qué pasaría si quisiéramos utilizar una única expresión con diferentes valores para x?
Supongamos que tenemos la expresión "$3x+5=$" y que deseamos asignar a x tres siguientes valores, como ser: 123, 456 y 789. El procedimiento anteriormente citado ya no sería práctico. Pero sí hay un método para que la computadora memorice expresiones matemáticas o algebraicas y es mediante los cálculos por programa. Es justamente esta posibilidad de realizar cálculos por programa, la característica más destacable de esta computadora. El Capítulo 3 trata la realización de cálculos mediante el lenguaje de programación BASIC.

2-4. FUNCIONES CIENTÍFICAS

Esta computadora incorpora funciones científicas además de las cuatro operaciones aritméticas fundamentales. Estas funciones pueden usarse tanto en los cálculos manuales como en los cómputos por programa. En esta sección veremos su utilización para el primero de los casos.

Funciones		Formato
Trigonométricas	seno x	SIN x
	coseno x	COS x
	tangente x	TAN x
Trigonométricas inversas	seno^{-1} x	ASN x
	coseno^{-1} x	ACS x
	tangente^{-1} x	ATN x
Raíz cuadrada	$\sqrt{x}$	SQR x
Logaritmo común	log x	LOG x
Logaritmo neperiano	ln x	LN x
Función exponencial	e^x	EXP x
Potencias	x^y	x↑y
Decimal → sexagesimal		DMS $ (x)*
Sexagesimal → decimal		DEG (x,y,z)*
Entero de Gauss		INT x
Porción decimal		FRAC x
Valor absoluto	$\lvert x \rvert$	ABS x
Función signo	Nº positivo → 1 0 → 0 Nº negativo → −1	SGN x
Redondeo matemático	Se agrega una unidad a la posición en x especificada por 10^y cuando la primera descartada es igual o mayor que 5.	RND (x,y)*
Números aleatorios		RAN #

* Para DMS $, DEG y RND, los argumentos deben estar entre paréntesis.

Cálculos con funciones

• Funciones trigonométricas (seno, coseno y tangente) y trigonométricas inversas (seno^{-1}, coseno^{-1} y tangente^{-1}).
Siempre que se utilicen estas funciones, no olvide especificar la unidad angular (DEG, RAD o GRA).

Ejemplo) seno 12.3456° = 0.2138079201
Operación) MODE 4 → "DEG"
SIN 12.3456 EXE

0.2138079201

* De aquí en más, se omiten los recuadros de las teclas numéricas y alfabéticas.

Ejemplo) cos 63°52'41" = 0.4402830847
Operación) COS DEG SHIFT (63 ° 52 ° 41 SHIFT) EXE

0.4402830847

Ejemplo) 2·sin45° × cos65.1° = 0.5954345575
Operación) 2 × SIN 45 × COS 65.1 EXE

0.5954345575

Ejemplo) $\sin^{-1}0.5 = 30°$
Operación) ASN 0.5 EXE

30

Ejemplo) $\cos(\frac{\pi}{3}\text{rad}) = 0.5$
Operación) MODE 5 → "RAD"
COS SHIFT (SHIFT π ÷ 3 SHIFT) EXE

0.5

Ejemplo) $\cos^{-1}\frac{\sqrt{2}}{2} = 0.7853981634\text{rad}$
Operación) ACS SHIFT (SQR 2 ÷ 2 SHIFT) EXE

0.7853981634

Ejemplo) tan(−35gra) = −0.612800788
Operación) MODE 6 → "GRA"
TAN (−) 35 EXE

-0.612800788

• Funciones logarítmicas (log y ln) y exponenciales (e^x y x^y).

Ejemplo) log 1.23 (= $\log_{10}$1.23) = 0.0899051114
Operación) LOG 1.23 EXE

0.0899051114

Ejemplo) ln 90 (= $\log_e$ 90) = 4.49980967
Operación) LN 90 EXE

4.49980967

Ejemplo) $e^5 = 148.4131591$
Operación) EXP 5 EXE

148.4131591

Ejemplo) $10^{1.23} = 16.98243652$
(Se obtiene el número real del logaritmo común de 1,23).
10 SHIFT ↑ 1.23 EXE

16.98243652

Operación) $5.6^{2.3} = 52.58143837$
Ejemplo) 5.6 SHIFT ↑ 2.3 EXE

52.58143837

Operación) $123^{+}(=\sqrt[3]{123}) = 1.988647795$
Ejemplo) 123 SHIFT ↑ SHIFT (1 ÷ 7 SHIFT) EXE

1.988647795

* Cuando $x < 0$, y es un número natural.

Operación) log sin40° + log cos35° = −0.278567983
Ejemplo) El número real es 0,5265407845 (cálculo del logaritmo del seno 40° X coseno 35°).
Operación) MODE 4 → "DEG"
LOG SIN 40 + LOG COS 35 EXE
10 SHIFT ↑ SHIFT ANS EXE

-0.278567983
0.5265407845

• Otras funciones (√, SGN, RAN#, RND, ABS, INT y FRAC)

Ejemplo) $\sqrt{2} + \sqrt{5} = 3.65028154$
Operación) SQR 2 + SQR 5 EXE

3.65028154

Ejemplo) Se obtiene un "1" si el número es positivo, un "−1" si es negativo y un "0" si es igual a cero.

Operación) SGN 6 EXE → 1
SGN 0 EXE → 0
SGN (−) 2 EXE → -1

Ejemplo) Generación de números aleatorios (pseudo aleatorios entre 0 < RAN# < 1).

RAN SHIFT (#) EXE → 0.790373907 6

Ejemplo) Redondeo matemático del resultado de 12,3 × 4,56 en la posición de 10^{-2}.

12.3×4.56=56.088

Operación) RND SHIFT (12.3 × 4.56) − 2 SHIFT) EXE → 56.1

* Cuando se usa RND (x, y), $|y| < 100$.

Ejemplo) |−78.9÷5.6|=14.08928571

Operación) ABS SHIFT ((−) 78.9 ÷ 5.6 SHIFT) EXE → 14.08928571

Ejemplo) Parte entera de 7800/96 81

Operación) INT SHIFT (7800 ÷ 96 SHIFT) EXE → 81

* Esta función obtiene el entero mayor que no exceda el valor numérico original.

Ejemplo) Parte decimal de 7800/96 0,25

Operación) FRAC SHIFT (7800 ÷ 96 SHIFT) EXE → 0.25

• Designación del número de posiciones significativas y del número de posiciones decimales.

Para la designación del número de posiciones, se hace uso del mando "SET".

Número de posiciones significativas ..... SET E n (n =0~8)
Número de posiciones decimales ..... SET F n (n =0~9)
Cancelación de designaciones previas ..... SET N

* En cálculos manuales, "SET E0" designa 8 posiciones significativas.
* Una vez designado el número de posiciones, los números visualizados en la pantalla se someten a un redondeo matemático a la posición del dígito menos significativo según la designación realizada. No obstante, en la memoria queda el número sin redondear.

Ejemplo) 100÷6=16.66666666·····

Operación) SHIFT SET E 4 EXE (Designación de 4 posiciones significativas)
(igual que S E T E4 EXE)
100 ÷ 6 EXE → 1.667 ε01

Ejemplo) 123÷7=17.57142857·····

Operación) SHIFT SET F 2 EXE (Designación de 2 posiciones decimales)
123 ÷ 7 EXE → 17.57

Ejemplo) 1÷3=0.3333333333·····

Operación) SHIFT SET N EXE (Cancelación de la designación previa)
1 ÷ 3 EXE → 0.3333333333

• Conversión decimal ↔ sexagesimal (DEG y DMS $)

Ejemplo) 14°25'36"=14.42666667

Operación) DEG SHIFT (14 ° 25 ' 36 " SHIFT) EXE → 14.42666667

Ejemplo) 12.3456° = 12°20'44.16"

Operación) DMS [SHIFT] [$] [SHIFT] [(] 12.3456 [SHIFT] [)] [EXE]

```
12° 20' 44.16
```

Ejemplo) sin 63°52'41" = 0.897859012

Operación) [MODE] [4]

SIN DEG [SHIFT] [(] 63 [°'"] 52 [°'"] 41 [SHIFT] [)] [EXE]

```
0.897859012
```


CAPITULO 3

PROGRAMACION EN BASIC

En este capítulo veremos numerosos métodos de programación en el lenguaje BASIC. Para quienes no tengan experiencia aluguna en el campo de la programación, se incluyen también algunos conceptos fundamentales al respecto.

3-1. QUE ES UN PROGRAMA?

En el lenguaje de programación BASIC, un programa consiste en un conjunto de instrucciones (mandos) que le dicen a la computadora qué es lo que debe hacer. Estas instrucciones se disponen dentro del programa con cierto orden, para determinar así la secuencia de ejecución del programa.

3-1-1 Aplicaciones de los programas

Ya hemos visto en el capítulo anterior cómo llevar a cabo cálculos manualmente. Podríamos pensar que las operaciones y las funciones tratadas, servirían para realizar cálculos comerciales y de matemáticas, mediciones o para cálculos de, por ejemplo, presupuestos. Si bien no habría problemas si estos cálculos tuvieran que hacerse tan sólo una vez, sí sería bastante complicado cuando es necesario repetir la misma operación muchas veces para direrentes valores dentro de la expresión matemática. Es aquí donde su computadora podrá serle útil.

```
10 INPUT X
20 Y=2*X↑2+5*X+13
30 PRINT Y
```

3-1-2 Estructura de un programa

Como ya dijimos anteriormente, un programa consiste en un conjunto de instrucciones, las cuales en BASIC se denominan "mandos", que le dicen a la computadora qué es lo que debe hacer. Estos mandos se disponen en líneas, las cuales se llaman "sentencias" y vienen precedidas por un "número de línea". El número de cada línea es el que determina el orden de ejecución del programa. Veamos un ejemplo simple.


```
10 INPUT X
20 Y=X+1
30 PRINT Y
```

Este programa se compone de tres líneas. La expresión de la línea 20 le resultará, seguramente, familiar. Sí, tal como Vd. cree, en esta línease asigna a la variable Y el resultado de la suma de X+1. Veamos la estructura de este programa paso por paso.

```
10 INPUT X ..... Entrada
20 Y=X-1 ..... Cómputo
30 PRINT Y ..... Visualización
```

La estructura básica de un programa se divide en la sección para la entrada de información, la sección para el proceso de la información entrada y la parte que visualiza en la pantalla o imprime en la impresora el resultado de los cómputos.
En el ejemplo, la sección para la entrada de la información es la correspondiente a la línea 10. La misma podría desglosarse del siguiente modo:

```
10  INPUT  X
|     |    |
Nº de línea  Mando  Operando
```

Ya sabemos que el número de línea determina el orden de ejecución de las líneas que componen el programa. Por lo general, se asignan números de línea de diez en diez. De tal modo uno puede, en caso de ser necesario, agregar líneas entre las actuales, ya sea para modificar el programa o para corregirlo. Sea como fuere, para numerar cada línea se pueden usar sólo números enteros.
El número de línea viene seguido por un mando, el cual le dice a la computadora la operación que debe realizar. Numerosos son los mandos incluidos en el lenguaje de programación BASIC. El tiempo y la práctica le ayudarán a memorizar la mayoría, sin bien no todos, de los mandos que se utilizan con mayor frecuencia. En el Capítulo 4 "MANDOS Y SENTENCIAS" (página 109) podrá encontrar todos los mandos y funciones de esta computadora.
El operando que se encuentra a continuación del mando es como un suplemento de este último. No todos los mandos requieren operandos. En nuestro ejemplo, el operando es la variable a la cual se asigna la entrada causada por el mando INPUT.

Desglosemos la línea siguiente:

```
20      Y=X+1
```

N° de línea — Sentencia de asignación

En la línea 20, se asigna el resultado de la operación que se encuentra a la derecha del signo de igualdad (=) a la variable a la izquierda del mismo signo. En realidad, el mando LET se utiliza para asignaciones de este tipo.

```
20 LET  Y=X+1
```

En esta línea, LET es un mando de asignación y la sentencia (expresión) de asignación el operando. De todos modos, y como puede observarse en el programa original, el mando LET por lo general se omite.

```
30 PRINT  Y
```

N° de línea — Mando — Operando

El mando PRINT de la línea 30 hace que se visualice en la pantalla el valor numérico asignado a la variable Y (operando del mando PRINT en este caso) en la línea anterior.
Así, este programa solicita la entrada de un valor numérico (línea 10), agrega una unidad al valor entrado (línea 20) y visualiza en la pantalla el resultado de la suma (línea 30). Si bien este programa es sumamente simple, la mayoría de los programas, aún los más complicados, se ajustan a una estructura similar.

3-1-3 CONCEPTOS DE PROGRAMACION

La preparación de programas no es tan difícil una vez que se haya comprendido la estructura del lenguaje utilizado, en este caso BASIC.
Numerosos tipos de programas pueden crearse en base a la estructura "entrada-cómputo-salida" vista anteriormente. En dicho caso, no es necesario que recuerde de una vez todos los mandos. Se recomienda, en cambio, comenzar con programas simples usando sólo los mandos INPUT y PRINT y las sentencias de asignación necesarias.
El secreto para llegar a ser un buen programador no consiste en tan sólo conocer el lenguaje de programación y memorizar todos sus mandos y funciones. Se requiere además una abundante imaginación para escoger, antes que nada, qué es lo que se quiere lograr con el programa propuesto, ya sean cálculos financieros, mediciones, contabilidad en la oficina o del

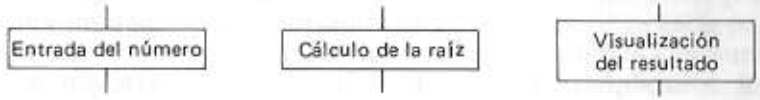
hogar, archivos de información, etc. Una buena mecánica sería, preparar un programa para lograr el objetivo deseado, memorizar los mandos en él utilizados y luego ir mejorándolo de a poco para que sea más conveniente para quien lo usa.
Otro secreto consiste en preparar los programas por partes (también llamados "módulos"), para luego unir las partes creadas por separado. Es más, suele suceder que el "módulo" de un programa pueda aprovecharse en otro programa como está o con algunas simples modificaciones.
Siempre que prepare un programa, hágalo recordando estos conceptos básicos. Una vez que se familiarice con la estructura del lenguaje BASIC, podrá preparar programas usando los mandos que se encuentran en el Capítulo 4 "MANDOS Y SENTENCIAS" y experimentando con los mismos.

3-2. PREPARACION DE UN PROGRAMA

En esta sección veremos prácticamente cómo se prepara un programa con el lenguaje BASIC.

3-2-1 Diagramas de flujo

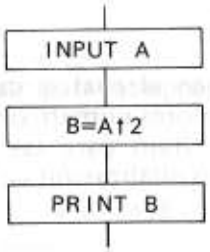
Se denomina diagrama de flujo al esquema que detalla los pasos de un trabajo, y las relaciones entre los diferentes pasos y componentes utilizados en cada uno de ellos.
Si bien en BASIC no se ha generalizado el uso de los diagramas de flujo, los mismos son recomendables para quienes recién se inician en el campo de la programación, ya que representan en forma visual cada uno de los pasos que componen un programa.
No obstante hay ciertos símbolos especiales que se utilizan en estos diagramas, por el momento sólo pondremos cada paso del programa dentro de un recuadro y uniremos cada recuadro con una línea.
Veamos a continuación un ejemplo práctico. Supongamos que necesitamos un programa para obtener la raíz cuadrada de un número. Coloquemos, entonces, cada paso del programa a continuación en un recuadro: entrada del número, cálculo de la raíz cuadrada y visualización en la pantalla del resultado.



Si unimos estas trea partes ordenadamente por medio de líneas, obtendríamos el diagrama de flujo para el programa en cuestión. Por lo general, por cuestiones de espacio, el detalle de cada paso puesto dentro de los recuadros se abrevia como en la figura a continuación.

A continuación, cambiemos lo que se encuentra dentro de cada recuadro por los mandos (en BASIC) que le corresponden.

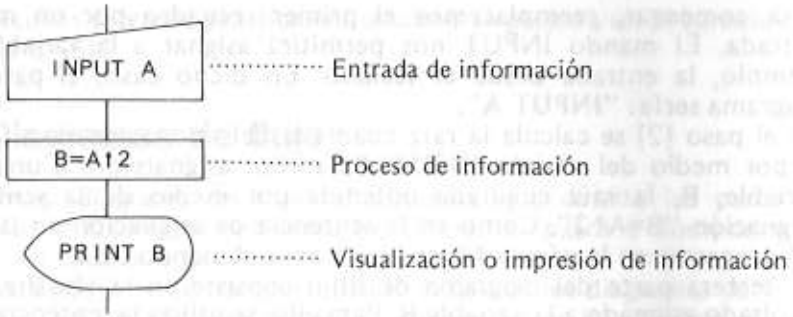
Para comenzar, reemplacemos el primer recuadro por un mando de entrada. El mando INPUT nos permitirá asignar a la variable A, por ejemplo, la entrada desde el teclado. En dicho caso, el paso (1) del diagrama sería: "INPUT A".
En el paso (2) se calcula la raíz cuadrada del valor asignado a la variable A por medio del mando INPUT. Para ello, asignaremos a una segunda variable, B, la raíz cuadrada obtenida por medio de la sentencia de asignación "B=A↑2". Como en la sentencia de asignación en la línea 20 del programa en la página 47, se omite aquí el mando LET.
La tercera parte del diagrama de flujo consiste en la visualización del resultado asignado a la variable B. Para ello, se utiliza la sentencia "PRINT B". Una vez reemplazado el contenido en cada recuadro que compone el diagrama de flujo y unidos los mismos por medio de líneas, se obtiene un esquema como el que sigue.



Si a cada paso de este diagrama de flujo le agregamos un número de línea, obtendríamos el siguiente programa.

```
10 INPUT A
20 B=A↑2
30 PRINT B
```

Del modo aquí descripto, es posible preparar programas similares para realizar diferentes operaciones matemáticas.
En realidad, en los diagramas de flujo se utilizan algunos símbolos especiales que permiten distinguir a simple vista el tipo de proceso implicado en cada paso. Veamos a continuación un ejemplo.



Al final de este manual, se encuentra una tabla con los símbolos utilizados en los diagramas de flujo.

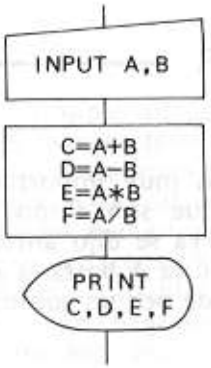
3-2-2 Otro Programa

Veamos aquí un programa en el cual se calculen la suma, resta, multiplicación y división de dos valores numéricos entrados.
Preparemos el diagrama de flujo para las secciones de entrada de la información, de cálculo y de visualización de los resultados.

Para entrar los dos valores numéricos y asignar cada uno de ellos a una variable diferente, use el mando INPUT. Para asignar información (en este caso, valores numéricos) por medio de este mando a más de una variable, coloque las mismas separadas por comas a continuación de INPUT. En nuestro ejemplo la sentencia correcta sería, entonces, "INPUT A, B".

Hay dos tipos de variables a las cuales se pueden asignar valores numéricos. Ellas son las comunes, de la A hasta la Z, y las de matriz (o matriciales), las cuales consisten en una letra seguida por un subíndice entre paréntesis. Por el momento, sólo usaremos variables comunes.
Ya sabemos que para cada respuesta obtenida necesitamos una variable a la cual asignar el resultado. Prepararemos, por ello, la variables C, D, E y F. A la variable C asignaremos el resultado de la suma por medio de "C=A+B"; a D asignaremos el resultado de la resta mediante "D=A–B"; a E asignaremos el resultado de la multiplicación por medio de "E=A*B" ("*" es el signo de multiplicación en BASIC); a F asignaremos el resultado de la división mediante "F=A/B" ("/" es el signo de división en BASIC).
Para la visualización de los cuatro resultados, se utiliza el mando PRINT de modo similar al de entrada, o sea, con las cuatro variables que contienen los resultados a continuación del mando separadas entre sí por comas: "PRINT C, D, E, F".

El diagrama de flujo terminado para este programa sería entonces:



Para completar el programa, coloquemos los números de línea como a continuación.

```
10 INPUT A,B
20 C=A+B
30 D=A-B
40 E=A*B
50 F=A/B
60 PRINT C,D,E,F
```

Agreguemos al final del programa el mando "END". El mismo indica a la computadora que el programa ha llegado a su fin.

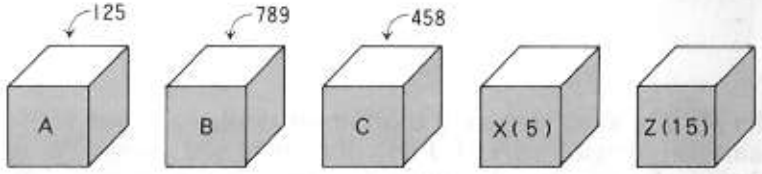
```
70 END
```

Así, damos por terminada la preparación de este simple programa. Con la práctica y siguiendo los conceptos aquí citados, Vd. irá adquiriendo la destreza necesaria para preparar programas más largos y más complicados.

PARA RECORDAR

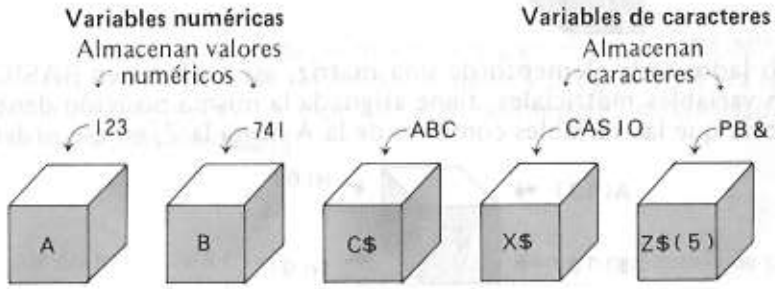
VARIABLES

Las variables son elementos muy importantes en la programación en BASIC. Podríamos decir que son como "cajas rotuladas" donde se guarda información. Como ya se dijo anteriormente, hay dos tipos de variables, las comunes desde la A hasta la Z, y las de matriz, las cuales consisten en una letra seguida por un subíndice entre pareántesis, como ser, A (5) ó B (50).



Además, las variables pueden clasificarse según el tipo de datos que almacenan, es decir, valores numéricos o información alfanumérica. Las variables para almacenar valores numéricos se denominan "variables

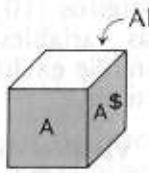
numéricas", mientras que las que almacenan información alfanumérica se llaman "variables de caracteres". En los ejemplos hasta aquí citados, hemos utilizado sólo variables numéricas. Las variables de caracteres se definen por medio de una letra, de la A hasta la Z, seguida por el signo "$". Hay, por otro lado, una variable de caracteres exclusiva, la cual se define por medio del signo "$" solamente.
Las variables numéricas aceptan hasta 10 dígitos (10 para la mantisa y 2 para el exponente), mientras que en las variables de caracteres se pueden almacenar hasta 7 caracteres. La variable exclusiva de caracteres, por su lado, puede almacenar hasta 30 caracteres.



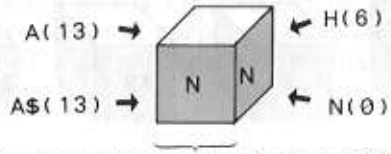
Tenga en cuenta que no es posible almacenar caracteres en variables numéricas, y que las variables de caracteres sí aceptan números, pero no como valores numéricos, sino como simples caracteres.
Por tal motivo, siempre que haga cálculos con valores numéricos, almacene los mismos en variables numéricas. Las variables de caracteres, por su lado, pueden usarse para almacenar mensajes, por ejemplo.
Siempre que se necesiten muchas variables, sean numéricas o de caracteres, para almacenar información, haga uso de las variables de matriz. Por medio de éstas, podrá almacenar gran cantidad de datos en variables con el mismo nombre y diferenciables por el subíndice entre paréntesis que les sigue. Más adelante veremos el uso de estas variables más en detalle.

<Precauciones en el uso de variables>

Tenga en cuenta que no se pueden usar variables numéricas y de caracteres con el mismo nombre. Ello se debe a que las variables, sean numéricas o de caracteres, con el mismo nombre son asignadas a la misma posición (dirección) dentro de la memoria.



Por otro lado, cada elemento de una matriz, así se llama en BASIC a las listas de variables matriciales, tiene asignada la misma posición dentro de la memoria que las variables comunes de la A hasta la Z, en ese orden.



Todos los nombres en la misma posición de memoria

```
A=A(0)=A$=A$(0)
B=A(1)=B(0)=B$=A$(1)=B$(0)
C=A(2)=B(1)=C(0)=C$=A$(2)=B$(1)=C$(0)
⋮
N=A(13)=B(12)=·····=N(0)=N$=A$(13)=····· N$(0)
⋮
Z=A(25)=B(24)=C(23)=·····=Z$=A$(25)=····· Z$(0)
```

Esta condición no cambia aún si se expande la memoria por medio del mando DEFM.

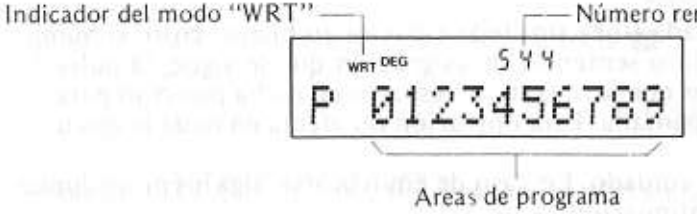
3-2-3 Entrada de un programa

Para poder ejecutar un programa, es necesario almacenarlo primeramente en la memoria entrando el mismo desde el teclado. Eso es lo haremos a continuación con el programa visto anteriormente, donde se entran dos valor numéricos y se calculan la suma, resta, multiplicación y división de los mismos.

Programa

```
10 INPUT A,B
20 C=A+B
30 D=A-B
40 E=A*B
50 F=A/B
60 PRINT C,D,E,F
70 END
```

Al encender la computadora, la misma entra en el modo "RUN", durante el cual se pueden ejecutar cálculos manuales y programas. Para la entrada del programa, es necesario especificar previamente la modalidad "WRT". Ello se logra pulsando MODE 1.



Tres son las indicaciones que aparecen en la pantalla durante esta modalidad. Los cuatro dígitos en la parte superior central de la pantalla indican el número de pasos de programa (memoria a disposición del usuario) que quedan. Este número va disminuyendo si se almacena algún programa o si se amplía la memoria disponible para las variables (ver la página 89). En nuestro caso, se puede observar, por el número de pasos visualizado, que no hay ningún programa en la memoria. Los números del 0 al 9 son los correspondientes a cada área de programa. El número que destella es el que corresponde al área de programa actualmente especificada. Quiere decir que, si entramos un programa en esta condición, lo estaremos almacenando en el área de programa P0. Siempre que la capacidad de memoria lo permita, es posible almacenar hasta 10 programas, uno en cada área de programa (de P0 a P9).

Para poder comenzar la entrada del programa, entre el siguiente mando.

NEW ALL [EXE]

Este mando borra todos los programas que puedan encontrarse en las 10 áreas de programa y hace que aparezca en la pantalla el cursor parpadelante (¿lo recuerda?). Pasemos ahora a entrar el programa desde el teclado.

[1][0][SHIFT][INPUT][A][,][B] [EXE] — Pulse esta tecla al final de cada línea.
└— Puede entrarse pulsando [I][N][P][U][T].

[2][0][C][=][A][+][B] [EXE]
[3][0][D][=][A][−][B] [EXE]
[4][0][E][=][A][×][B] [EXE]
[5][0][F][=][A][÷][B] [EXE]
[6][0][SHIFT][PRINT][C][,][D][,][E][,][F] [EXE]
[7][0][E][N][D] [EXE]

Aún si Vd. entra cada línea sin dejar espacios en blanco entre el número de línea y el mando o sentencia de asignación que le sigue, la pulsación de la tecla [EXE] hace que se inserte un espacio en dicha posición para una mejor visión en la pantalla. Esta operación no afecta en nada la ejecución del programa.
Pulse las teclas con cuidado. En caso de equivocarse, siga los procedimientos descriptos a continuación.

• Corrección de errores antes de pulsar la tecla [EXE].

Siempre que detecte un error antes de pulsar la tecla [EXE], podrá corregirlo desplazando el cursor por medio de la tecla [⇦] o [⇨] hasta la posición por debajo del carácter equivocado, y allí pulsar la tecla que corresponde a la letra correcta.

Ejemplo 1) `10 INPUT S_` Se entró "S" en lugar de "A".
Lleve el cursor hasta la posición por debajo de la "S".

[⇨]

```
10 INPUT S
```


Pulse la tecla [A].

[A]

```
10 INPUT A_
```

Prosiga con la entrada de dicha línea y pulse finalmente la tecla [EXE].

[,][B] [EXE]

```
10 INPUT A,B
```

Ejemplo 2) `40 EE=A*B_` Se entró un "E" de más.
Pulse cinco veces la tecla [⇦] para llevar el cursor hasta la posición por debajo de la "E".

[⇦][⇦][⇦][⇦][⇦]

```
40 EE=A*B
```

Pulse una vez la tecla [DEL] para suprimir el carácter que se encuentra por encima del cursor.

[DEL]

```
40 E=A*B
```

Una vez completada la corrección, pulse la tecla [EXE].

```
40 E=A*B
```

Ejemplo 3) `RINT C,,E,F_` Falta la "D" en la línea 60.
Pulse 4 veces la tecla [⇦] para llevar el cursor hasta la posición a continuación de donde desea llevar a cabo la inserción.

[⇦][⇦][⇦][⇦]

```
60 PRINT C,,
```

Inserte allí un espacio en blanco.

[SHIFT][INS]

```
60 PRINT C, _
```

Entre la "D" que faltaba.

[D]

```
0 PRINT C,D
```

Una vez finalizada la corrección, pulse la tecla [EXE].

[EXE]

```
60 PRINT C,D
```

• Correcciones después de pulsada la tecla EXE.

La pulsación de la tecla EXE hace que la línea entrada se almacene en la memoria como parte del programa. Por ello, para corregir una línea ya almacenada, es necesario visualizarla en la pantalla por medio del mando LIST.

Ejemplo) Se entró una "N" en lugar de "B" en la línea 50.
Visualice dicha línea mediante el mando LIST.

SHIFT LIST 50 EXE

Puede entrarse pulsando L I S T .

```
50 F=A/N_
```

Pulse la tecla ⇦ una vez para colocar el cursor por debajo de la "N".

⇦

```
50 F=A/N
```

Pulse la tecla correspondiente al carácter correcto.

B

```
50 F=A/B_
```

Una vez finalizada la corrección, pulse la tecla EXE .

EXE

En caso de que se haya ya almacenado la línea que sigue a la actual, la misma (en este caso la número 60) se visualiza en la pantalla.

```
60 PRINT C,D
```

Si terminó con todas las correcciones, pulse la tecla AC .

AC

Siempre que desee corregir una línea ya almacenada en la memoria, visualícela en la pantalla por medio del mando LIST y corríjala según sea necesario. O podría, si así lo deseara, volver a entrar la misma línea correctamente. Ello hace que la previamente entrada (la que contiene el error) sea reemplazada por la nueva.
Una vez finalizada la entrada del programa, volvamos al modo RUN, en el cual se pueden ejecutar cálculos manuales y programas, pulsando la tecla MODE . En caso de apagar la computadora sin cambiar del modo WRT (para la entrada de programas) a otro, el programa entrado se borraría de la memoria.

PARA RECORDAR

AREAS DE PROGRAMA

Esta computadora permite el almacenamiento simultáneo de 10 programas en las denominadas áreas de programa P0 a P9. Cada una de estas áreas se utilizan de la misma manera. Suponiendo que no se dispusiera de estas áreas de programa y fuera necesario el uso frecuente de tres diferentes programas, sería necessario recuperar cada programa desde cinta de cassette.
Esta división de la memoria evita este tipo de problemas.
Si bien se puede almacenar un programa por "área de programa" (así es cómo se las llama a estas divisiones), deben tomarse las precauciones debidas, de modo tal que el número total de pasos utilizados no exceda la capacidad total de la memoria
Cada área de programa se especifica en el modo WRT o RUN pulsando la tecla correspondiente a su número (0 hasta 9) mientras se presiona la tecla SHIFT . Cuando se especifica en el modo RUN, la ejecución del programa almacenado en el área especificada comienza automáticamente. En cambio, en el modo WRT, sólo se pasa al área de programa especificada.
Al encenderse la computadora, se especifica automáticamente el área de programa P0. El área de programa en curso puede confirmarse pulsándose MODE 0 . Al hacerlo, aparece en la pantalla el mensaje "READY" seguido por el área de programa actual.

Ejemplo) MODE 0 → READY P3 ····· Area de programa P3

3-2-4 Ejecución de programas

Probemos la ejecución del programa previamente entrado en la memoria. Para ello, la computadora debe estar en el modo "RUN" ([MODE] [0] — "RUN" en pantalla).
Hay dos métodos para la ejecución de un programa.

(1) Ejecución por la especificación del área de programa.
Pulse la tecla numérica correspondiente al área de programa deseada mientras presiona [SHIFT].

Ejemplo) [SHIFT] [P0]

(2) Ejecución por medio del mando RUN.

Ejemplo) [SHIFT] [RUN] (igual que [R] [U] [N]) [EXE]

Estos dos métodos de ejecución se diferencian en lo siguiente. Por medio del método (1), la ejecución se puede iniciar sólo a partir del comienzo del programa. En cambio, mediante el método (2), el programa puede ejecutarse a partir del número de línea deseado.

Probemos primero el método (1).

Operación **Pantalla**

[SHIFT] [P0]

```
?
```

Entre los dos números.

Ejemplo)

45 [EXE]
36 [EXE]

```
?
81
```

Al ejecutarse la sentencia PRINT, se enciende en la pantalla la indicación "STOP" (de parada).

Una vez entrados los dos números, se visualiza el resultado de la suma. Pulse la tecla [EXE] para llevar a la pantalla el resultado de la siguiente operación.

[EXE]
[EXE]
[EXE]

```
9
1620
1.25
```

A continuación, ejecutemos el mismo programa mediante el uso del mando RUN. Si se entrara sólo este mando, el resultado sería el mismo que el del procedimiento anterior. Sin embargo, si entraramos RUN seguido por un número de línea, el programa se ejecutaría del siguiente modo.

Operación

[SHIFT] [RUN] 20 [EXE]

```
81
```

(Igual que RUN 20 [EXE] .)

[EXE]
[EXE]
[EXE]

```
9
1620
1.25
```

Siempre que no se especifique número de línea alguno a continuación de RUN, el programa se ejecuta desde el comienzo. En cambio, si se especificara un número de línea a continuación de RUN, el programa comenzaría a ejecutarse a partir del número de línea especificado. En realidad, esta no es la única diferencia entre estos dos métodos.

Siempre que se usa el mando RUN, se ejecuta el programa almacenado en el área de programa en curso (P5 en este caso). ¿Cómo se podría, entonces, ejecutar desde P5 el programa almacenado en P0? La solución es la secuencia [SHIFT] [P0] .

Estos dos métodos para la ejecución de programas pueden usarse libremente, según sea el caso o la necesidad.
Suele suceder que se cometan errores en la entrada de un programa. En dicho caso, al intentar ejecutar tal programa, la computadora le avisa la presencia del o de los errores por medio de ciertos mensajes ("ERR" seguido por un número). En la siguiente sección veremos los métodos de identificación y corrección de los errores que aparecen en los programas.

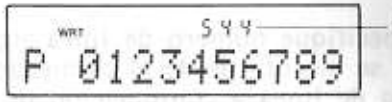
PARA RECORDAR

NUMERO DE PASOS

Esta computadora tiene una capacidad de memoria de 544 pasos. El uso del módulo de expansión de la memoria RAM OR-1(E) opcional permite ampliar la memorias hasta 1568 pasos.
Los pasos son la unidad que indican la capacidad de la memoria usada para el almacenamiento de programas. De tal modo, el número de pasos remanentes va disminuyendo a medida que se almacenan nuevos programas.
El número de pasos remanente se visualiza siempre que se especifica el modo WRT mediante MODE 1 .

Ejemplo)

MODE 1



Número de pasos remanente

El número de pasos que ocupa un programa se cuenta de acuerdo a los siguientes criterios:

- Nº de línea ..... 2 pasos (de 1 hasta 9999)
- Mando ..... 1 paso
- Función ..... 1 paso
- Variable ..... 1 paso por carácter
- Además de lo aquí citado, cada pulsación de la tecla EXE requiere 1 paso.

Ejemplo)

```
1  INPUT A EXE ..... 5 pasos
␣  ──── ␣ ␣
2    1   1 1

10 B=SIN A EXE ..... 7 pasos
␣  ␣␣ ␣ ␣ ␣
2  1 1 1 1 1

100 PRINT "B=";B EXE ..... 10 pasos
␣   ──── ␣␣␣
2     1  4 1 1 1
```

Total: 22 pasos

- Cuando se expande la memoria, se requieren 8 pasos por cada memoria que se agrega.

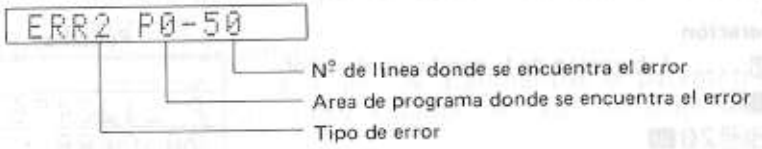
Ejemplo) Estado inicial ..... 26 memorias 544 pasos
DEFM 10 EXE ..... 36 memorias 464 pasos

3-2-5 Corrección de errores en un programa

La aparición de errores durante la ejecución de un programa es algo que sucede con cierta frecuencia. Ello hace, a veces, que sea imposible obtener el resultado. La localización y corrección de este tipo de errores en un programa se denomina "depuración de programas".
El método de depuración depende de la causa del error. En algunos casos, se visualiza un mensaje de error durante la ejecución del programa en cuestión; en otros, no aparece mensaje de error alguno, pero no se obtiene el resultado deseado. Estos mensajes indican la posición del error donde dentro del programa y el tipo de error localizado. Cuando el resultado obtenido no es el deseado a pesar de que no se visualiza ningún error, el procedimiento de depuración es un poco más complicado. Veamos ambos casos por orden.

(1) Cuando se visualiza un mensaje de error

Los mensajes de error aparecen en la pantalla con el siguiente formato.



El tipo de error se visualiza mediante un número a continuación del mensaje "ERR". La descripción de cada número, el cual puede ser del 1 al 9, puede encontrarse en la Tabla de Mensajes de Error que se encuentra en la página 162.
El área de programa donde se encuentra el programa con el error se visualiza a continuación del tipo de error, separado de éste por un guión.
A continuación de la indicación del número del área de programa se visualiza el número de línea donde se encuentra el error.
De este modo, los mensajes de error permiten al programador saber dónde y qué tipo de error ha detectado la computadora durante la ejecución del programa en cuestión. Veamos a continuación un ejemplo práctico.
El error más frecuente es el de sintaxis, indicado por el mensaje "ERR2". Este mensage aparece siempre que se comete un error en la entrada de un programa.

Supongamos, por ejemplo, que cometimos un error en la entrada del programa anterior.

Operación	Pantalla
MODE 1	P _123456789
SHIFT LIST 20 EXE	20 C=A+B_
← ← DEL	20 C=AB
EXE	30 C=A-B_
AC MODE ⊡	READY P0

En este ejemplo, para la línea 20 se entró "C=AB" en lugar de "C=A+B". Ejecute el programa con este error.

Operación	Pantalla
SHIFT P0	?
45 MODE	?
12 EXE	ERR2 P0-20

Al hacerlo, la computadora visualiza un mensaje indicando que ha encontrado un error en la línea 20 del programa almacenado en el área P0.

Operación	Pantalla
AC ..... Liberación de la condición de error	_
MODE 1	P _123456789
SHIFT LIST 20 EXE	20 C=AB_

Corrijamos, entonces, la línea 20 de acuerdo al programa original, o sea, agregando el signo "+" entre A y B.

Operación	Pantalla
⇨	20 C=AB
SHIFT INS	20 C=A_B
+ EXE	30 C=A-B
AC MODE ⊡	READY P0

Por lo general, ERR2 es provocado por un error en la entrada del programa. Por lo tanto, visualice en la pantalla la línea con el error y corríjala según sea necessario. Este error aparece también cuando se intenta asignar información de caracteres a una variable numérica por medio de la sentencia READ (en la página 92). Cuando este error aparece en una línea con esta sentencia, verifique también los datos (incluidos en la sentencia DATA) que se asignarán a las variables correspondientes por medio de la sentencia READ.
He a continuación una lista de los mensajes de error con sus respectivas definiciones.

- ERR1: Superación de la capacidad de la memoria.
  Verifique el número remanente de pasos. Confirme que no hayan sido muchos los pasos convertidos en memorias por medio de la sentencia DEFM.
- ERR2: Error de sintaxis corrobore que no haya errores de sintaxis en el programa almacenado.
- ERR3: Error matemático
  Verifique que el resultado de una expresión aritmética no exceda el límite 10^{99} y confirme que no se haya excedido la gama de entrada de una función. Tenga en cuenta que no se permiten ni divisiones por cero ni raíces cuadradas de números negativos.
- ERR4: Definición de un número de línea que no existe
  Una sentencia GOTO, GOSUB o RESTORE especifica un número de línea que no existe. Confirme el número de línea correcto.
- ERR5: Error de argumento
  Verifique el valor de los argumentos o parámetros en los mandos y funciones. Cuando se usan variables, verifique su contenido.

- ERR6: Error de variable
  Verifique que se haya ampliado la memoria mediante la sentencia DEFM siempre que utilice una variable de matriz. Corrobore, además, que no se haya utilizado el mismo nombre para una variable numérica y otra de caracteres.
- ERR 7: Error de inclusión de ciclos o subrutinas
  Siempre que en la línea donde se encuentra el error aparezca una sentencia RETURN o NEXT, confirme la presencia de la sentencia GOSUB o RETURN que les corresponde respectivamente. Cuando el error se encuentra en una línea donde hay una sentencia GOSUB o FOR, recuerde que se admiten hasta 8 subrutinas en cascada y hasta cuatro ciclos FOR-NEXT incluidos.
- ERR8: Error de clave
  Siempre que se haya especificado previamente una clave, verifique que no se haya entrado una clave equivocada o que no se haya intentado el uso de los mandos LIST, NEW, NEW ALL, etc.

Los mandos aquí mencionados se explican más adelante en este manual. La "Lista de Mandos", la cual se encuentra en la página 109, le servirá como referencia para mayores detalles.

(2) Cuando no se visualiza ningún mensaje de error pero no se logra el resultado deseado.

Esta condición suele suceder siempre que se intente llevar a cabo una operación incorrecta o cuando el uso de las variables no es el adecuado. Especialmente cuando no se ha logrado el resultado correcto, compare la expresión original con la expresión de cómputo.
Cuando la ejecución del programa no se interrumpe o si se interrumpe sin que se hayan realizado las operaciones asignadas al programa, revise las variables que controlan el flujo de dicho programa. En cuanto a las expresiones de cómputo, revíselas entrando primeramente en el modo WRT (mediante MODE 1), del mismo modo que en el paso (1) del ejemplo anterior.
El flujo del programa puede verificarse interrumpiéndolo momentáneamente mediante la introducción de sentencias STOP dentro del mismo o visualizando en la pantalla el valor de determinada variable por medio de la sentencia PRINT.

Entre el programa a continuación.

```
10 INPUT A
20 B=1
30 FOR C=1 TO A
40 B=B*C
50 C=C+1
60 NEXT C
70 PRINT B
80 END
```

Este programa obtiene el factorial de los datos entrados mediante una sentencia INPUT. En realidad, la línea 50 no se necesita y la variable utilizada para el lazo FOR no debe cambiar.
Las sentencias FOR y NEXT en las líneas 30 y 60 respectivamente son las que determinan el comienzo y el final del lazo, por medio del cual se repite cierta cantidad de veces un determinado ciclo. El uso de estas sentencias se explica más adelante en este manual. Veamos, no obstante, cómo se ejecuta este programa.

Operación	Pantalla
MODE 1	P -?123456789
SHIFT P1	P 1?123456789
NEW EXE	P -1123456789
└ Mando para borrar un programa.	
10 SHIFT INPUT A EXE	10 INPUT A
20 B=1 EXE	20 B=1
30 SHIFT FOR C=1 SHIFT TO A EXE	30 FOR C=1 T
40 B=B*C EXE	40 B=B*C
50 C=C+1 EXE	50 C=C+1
60 SHIFT NEXT C EXE	60 NEXT C
70 SHIFT PRINT B EXE	70 PRINT B
80 END EXE	80 END

Una vez entrado el programa, pulse [AC] [MODE] [0] para especificar el modo RUN. Ejecutemos entonces el programa.

Operación	Pantalla
Ejemplo) [SHIFT] [1]	?
12 [EXE]	10395

La respuesta correcta es 479001600.
Confirme que no haya errores en la expresión de cómputo. Luego verifique el flujo del lazo FOR-NEXT.
Inserte una sentencia STOP a continuación de la línea 50 a fin de interrumpir el programa momentáneamente en cada ejecución del lazo.

Operación	Pantalla
[MODE] [1]	P _|23456789
55 STOP [EXE]	55 STOP
[AC] [MODE] [0]	READY P1

Para insertar una línea con la sentencia STOP entre las líneas 50 y 60, use cualquier número de línea entre estos dos números. En nuestro caso, usaremos la línea 55.
Ejecutemos nuevamente el programa.

Operación	Pantalla
[SHIFT] [1]	?
12 [EXE]	_

Verifique el valor de la variable de control del lazo C.

C [EXE]	2

Continue con la ejecución.

[EXE]	_

Verifique nuevamente el valor dela variable C.

C [EXE]	4

Se supone que C debe aumentar de a 1, pero en realidad se está incrementando de a 2 unidades por ciclo. Quiere decir que el flujo del lazo FOR-NEXT no es correcto. Verifique entonces nuevamente el incremento de la variable C por cada ejecución del ciclo. Al hacerlo, se llega a la conclusión de que no se necesita la línea 50. Borre, por lo tanto, la línea 50 y la que se agregó con la sentencia STOP (en nuestro caso, la línea 55).

Operación	Pantalla
[MODE] [1]	P _|23456789
50 [EXE]	_
55 [EXE]	_
[AC] [MODE] [0]	READY P1

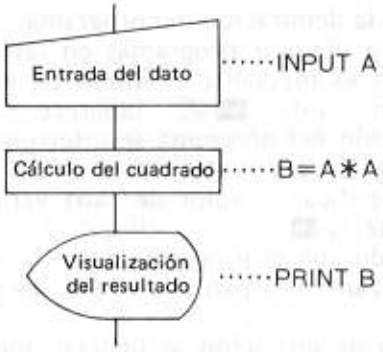
Con ello se completa la depuración del programa.
Hay otra forma para depurar programas en las cuales no se utiliza la sentencia STOP. Ella es mediante el modo de ejecución secuencial, el cual se especifica mediante [MODE] [2] (aparece "TR" en pantalla). En este modo, la ejecución del programa se interrumpe momentáneamente por cada mando que se encuentra en el mismo. Durante dicha interrupción se puede verificar el valor de cada variable y la ejecución se reanuda pulsando la tecla [EXE] .
Probemos este método con el programa anterior. Por cada pulsación de la tecla [EXE] se visualizan en la pantalla el área de programa y el número de línea.
Para liberar el modo de ejecución secuencial, pulse [MODE] [3] ("TR" desaparece de la pantalla).
Siempre que aparezca un mensaje de error o no se obtenga el resultado deseado al ejecutar determinado programa, lleve a cabo la depuración del mismo mediante los métodos aquí mencionados.

3-3. DESARROLLO DE PROGRAMAS

Ya hemos visto cómo preparar programas en base a la estructura básica conformada por las partes de entrada, cómputo y visualización. Si bien son muchos los programas que podrían prepararse en base a esta estructura, resulta conveniente conocer los mandos explicados en esta sección.

3-3-1 Cambio del flujo de un programa (sentencia GOTO)

La sentencia GOTO se utiliza ya sea para repetir varias veces una misma operación o para cambiar arbitrariamente el flujo del programa en lugar de seguir con la ejecución ordenada de las líneas que lo componen.
Preparemos, por ejemplo, un programa para obtener el cuadrado de cierto valor. Compongamos este programa con sus tres partes fundamentales de entrada (del valor numérico), cómputo (del cuadrado) y salida (o visualización del resultado). Preparemos el diagrama de flujo.



Preparemos el programa de acuerdo a este diagrama de flujo.

```
10 INPUT A
20 B=A*A
30 PRINT B
40 END
```


Por ejemplo, para obtener el cuadrado de 15 y de 43.

Operación	Pantalla
RUN EXE	?
15 EXE	225
RUN EXE	?
45 EXE	1849

Supongamos que deseamos obtener el cuadrado de muchos números. En dicho caso, sería necesario ejecutar el programa nuevamente para cada número, lo que sería bastante poco práctico.
Es aquí donde la sentencia GOTO puede agilizar la ejecución del programa. Esta sentencia permite que el programa de un "salto" al número de línea especificado por el número que se encuentre a continuación de GOTO.
Cambiemos, entonces, la sentencia END de la línea 40 por una sentencia GOTO, del siguiente modo.

```
40 GOTO 10
```

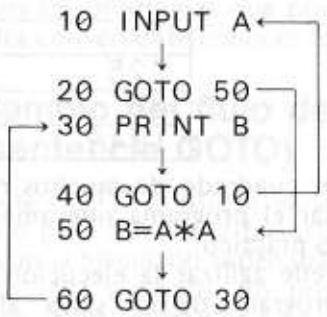
Por medio de esta sentencia, el programa volverá a la línea 10 después de visualizar el resultado. Ejecutemos el programa modificado.

Operación	Pantalla
RUN EXE	?
15 EXE	225
EXE	?
43 EXE	1849

La sentencia GOTO es muy conveniente ya que, como en el ejemplo anterior, permite repetir la misma operación un número indeterminado de veces.
Hay muchas otras maneras de utilizar la sentencia GOTO, ya sea para volver al comienzo del programa o para saltar arbitrariamente al número de línea deseado.
Por ejemplo,

```
10 INPUT A
20 GOTO 50
30 PRINT B
40 GOTO 10
50 B=A*A
60 GOTO 30
```

El flujo del programa sería el siguiente.



Por la naturaleza de la ejecución de esta sentencia, se la denomina "salto incondicional", ya que no requiere la satisfacción de condición alguna para ejecutarse.
La sentencia GOTO permite saltar ya sea a la línea o el área de programa deseado. Para especificar un área de programa, se agrega a continuación de GOTO el símbolo "#" seguido por el número de área (de 0 hasta 9).

Ejemplo) GOTO #1 ····· Salto al área P1.
GOTO #9 ····· Salto al área P9.

Siempre que se salte a otra área de programa, el programa almacenado en dicha área se ejecuta desde el comienzo.

PARA RECORDAR

SENTENCIA PRINT

La sentencia PRINT se utiliza para visualizar en la pantalla el contenido de variables, ya sean numéricas o de caracteres.

Cuando A=123 ····· PRINT A → 123
Cuando B$="ABC" ····· PRINT B$ → ABC

Las secuencias de caracteres entre comillas asignadas a la variable de caracteres pueden visualizarse en la pantalla a modo de mensajes.

Ejemplo) PRINT "CASIO" → CASIO

Cuando necesite visualizar más de dos ítems, podrá poner uno a continuación del otro separados por una coma "," o un punto y coma ";".

Ejemplo)
```
PRINT A,B,Z$
PRINT "TOTAL=";T
```

Cuando se separan los ítems por medio de una coma, se visualiza el primero de ellos y el programa se interrumpe momentáneamente (aparece "STOP" en la pantalla). El siguiente ítem puede visualizarse pulsando la tecla EXE . Sin embargo, cuando se utiliza un punto y coma para separar las variables cuyo contenido desea visualizarse, cada ítem aparece en la pantalla a continuación del otro, en la misma línea.

Ejemplo) Probemo con el siguiente programa.

```
10 A=123
20 B$="ABC"
30 PRINT A,B$ ····· Se visualiza primero el contenido de A y luego, después de pulsar EXE, el contenido de B$.
40 PRINT A;B$ ····· El contenido de A y B$ se visualizan uno a continuación del otro.
50 PRINT B$; ····· Se visualiza el contenido de B$ y la ejecución del programa continúa sin detenerse.
60 PRINT A ····· El programa se detiene después de visualizar el contenido de la variable A.
70 END
```

Este programa se ejecuta del siguiente modo.

Operación	Pantalla	
RUN EXE	123	PRINT A,B$
EXE	ABC	
EXE	123ABC	PRINT A;B$
EXE	ABC 123	PRINT B$; PRINT A

Si bien se utiliza ":" para visualizar el contenido de una variable a continuación del otro, el valor numérico (123 en este caso) aparece en la pantalla precedido por un espacio en blanco. Este espacio en blanco es reservado para el signo del número, sólo que el signo aparece en la pantalla solamente cuando es un número negativo.

[EJERCICIO]
Prepare un programa para obtener el área de un círculo para diferentes valores del radio. Use para ello la sentencia GOTO.

Expresión: $S = \pi r^2$ (para entrar Pi, pulse [SHIFT] [π]).

El diagrama de flujo es el siguiente. S y R se utilizan como variables para la expresión citada.

PROGRAMA

```
10 INPUT R
20 S=π∗R∗R
30 PRINT S
40 GOTO 10
```

ó

```
10 INPUT R
20 S=π∗R↑2
30 PRINT S
40 GOTO 10
```

"↑2" se utiliza para elevar un número al cuadrado.

3-3-2 Juicios y decisiones en un programa (IF y THEN)

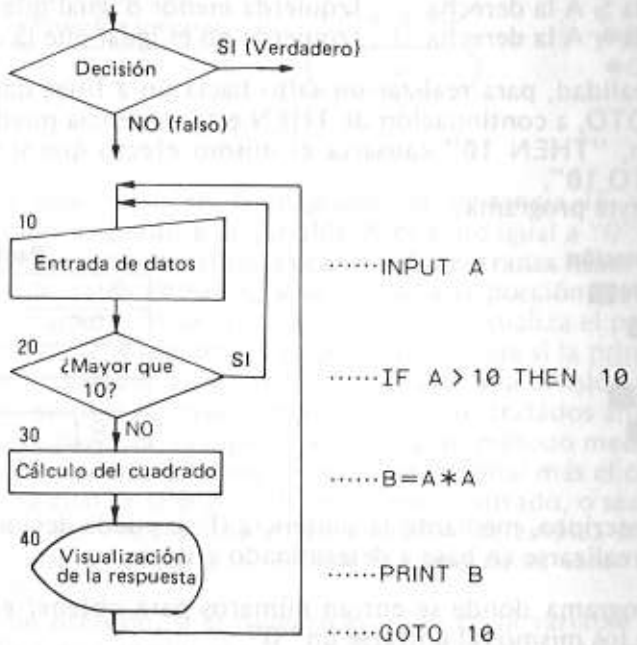
No hay duda que sería muy conveniente si la computadora pudiera tomar decisiones en cuanto a los pasos a seguir en base a ciertos criterios. La sentencia IF se utiliza en BASIC justamente para tomar decisiones dentro de un programa. Según si se cumple o no la expresión comparativa a continuación de IF, el programa decide qué paso seguir.

IF expresión comparativa THEN { Nº de línea o de área de programa / Sentencia de instrucción }

Siempre que se cumple la expresión entre IF y THEN, se lleva a cabo la operación especificada a continuación de esta última sentencia. En caso contario, el programa pasa a la línea siguiente.
Veamos la función de la sentencia IF en la práctica.

Ejemplo) Entre un número cualquiera. Si el número entrado es mayor que 10, haga que el programa vuelva a la línea para entrada del valor numérico. En caso contrario, eleve al cuadrado el número entrado, visualice el resultado y haga que el programa regrese a la línea para entrada del número.

En este caso, el programa se conformaría de cuatro partes: la de entrada, de decisión, de cómputo y de visualización. Para la porción de decisión en los diagramas de flujo se utiliza el siguiente símbolo.



Compongamos el programa en base al diagrama de flujo, con los números de línea correspondientes.

```
10 INPUT A
20 IF A>10 THEN 10
30 B=A∗A
40 PRINT B
50 GOTO 10
```

La línea 20 usa una sentencia IF. Si el resultado de la comparación es positivo, se ejecuta lo ordenado a continuación de la sentencia THEN (en este caso, se llevaría a cabo un salto a la línea 10).
En las expresiones de comparación, se utilizan los siguientes operadores.

A la izquierda > A la derecha . . . Izquierda mayor que la derecha
A la izquierda < A la derecha . . . Izquierda menor que la derecha
A la izquierda = A la derecha . . . Izquierda igual que la derecha
A la izquierda ≧ A la derecha . . . Izquierda mayor o igual que la derecha
A la izquierda ≦ A la derecha . . . Izquierda menor o igual que la derecha
A la izquierda ≠ A la derecha . . . Izquierda no es igual que la derecha

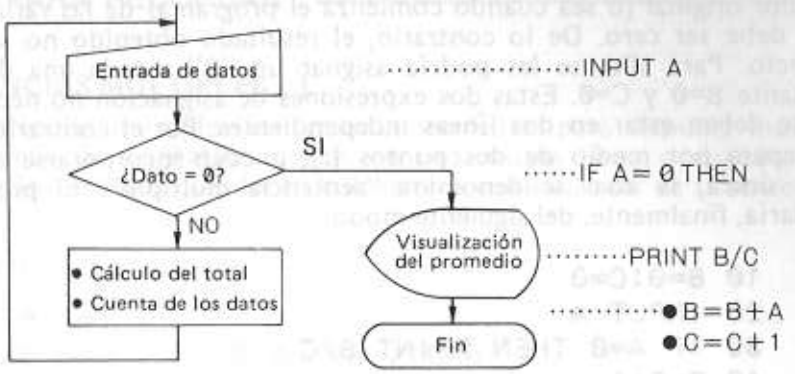
Si bien en realidad, para realizar un salto hacia un a línea dada se usa la sentencia GOTO, a continuación de THEN esta sentencia puede omitirse. De tal modo, "THEN 10" causaría el mismo efecto que si se utilizara "THEN GOTO 10".
Ejecutemos este programa.

Operación	Pantalla
RUN EXE	?
5 EXE	25
EXE	?
12 EXE	?
9 EXE	81

Del modo descripto, mediante la sentencia IF se puede decidir el tipo de operación a realizarse en base a determinado criterio.

Ejemplo) Programa donde se entran números para obtener el promedio de los mismos al entrarse un "0".

Este programa se compone de las partes para entrada, decisión, cálculo y visualización del resultado. Dentro de la parte de cómputo, se incluyen además tres operaciones: cálculo de la suma total, cuenta de la cantidad de números entrados y cálculo del promedio. En vista de que el promedio se obtiene sólo después que se entra un "0", la porción del programa que le corresponde se encontrará a continuación de la porción de decisión. En base a este análisis, obtenemos el siguiente diagrama de flujo.



Como podrá observarse en el diagrama, la sentencia IF verifica si el número entrado asignado a la variable A es o no igual a "0". En caso de no serlo, se obtienen el total de las cantidades entradas hasta el momento y el número de datos entrados, y se vuelve a la porción para la entrada del siguiente número. Si se entrara un "0", se visualiza el promedio y se da por terminado el programa. Tenga en cuenta que si la primera entrada es igual a cero, aparecería un error, al intentarse una división por "0".
Este ejemplo es un poco más complicado que los tratados anteriormente. En cuanto a la parte de cómputo, se utiliza un método mediante el cual se asigna a una variable la suma de su valor original más el contenido de la variable a la cual se asignó el último número entrado, o sea, "B=B+A". Para la cuenta del número de datos entrados, se emplea una mecánica similar, sólo que se asigna a la variable la suma de su valor original más 1, o sea, "C=C+1".
En la parte de decisión, si el número entrado en la variable A es igual a 0, se visualiza por medio de la sentencia PRINT el promedio de los números entrados hasta entonces. Para ello, se incluye a continuación de esta sentencia la expresión de cómputo resultando, en este caso, en "PRINT B/C".
En vista de que se pueden escribir sentencias a continuación de "THEN", "PRINT B/C" puede incorporarse a continuación de esta sentencia, y no necesariamente en otra línea.

El valor original (o sea cuando comienza el programa) de las variables B y C debe ser cero. De lo contrario, el resultado obtenido no sería el correcto. Para ello, se les podría asignar un "0" a cada una de ellas mediante B=0 y C=0. Estas dos expresiones de asignación no necesariamente deben estar en dos líneas independientes. Por el contrario, si se las separa por medio de dos puntos (:), pueden incorporarse en una línea única, la cual se denomina "sentencia múltiple". El programa quedaría, finalmente, del siguiente modo:

```
10 B=0:C=0
20 INPUT A
30 IF A=0 THEN PRINT B/C
40 B=B+A
50 C=C+1
60 GOTO 20
```

Debe notarse que, en caso de entrarse un cero, se visualiza el promedio pero el programa no termina. Para hacer que el programa finalice después de visualizar el resultado, agregue una sentencia END al final de la línea 30, separando esta sentencia de la previa por medio de dos puntos.

```
30 IF A=0 THEN PRINT B/C:END
```

De tal modo, la sentencia IF permite al programa juzgar y determinar la operación que debe realizarse a continuación en base a ciertos criterios establecidos.

- **Aplicaciones de la sentencia IF**

En el ejemplo anterior, se determinó el flujo del programa en base a una decisión. ¿Que pasaría si hubiera que tomar cierta decisión en base al cumplimiento de dos condiciones?
Supongamos que debemos preparar un programa en el cual sólo se acepta la entrada de números del 1 hasta el 9. Debemos, entonces, por cada número entrado, ver si el mismo es mayor que 0 y menor que 10. Para ello se podrá utilizar la expresión:

```
IF 0<variable THEN IF variable < 10 THEN.....
```

Si bien este tipo de expresión podría utilizarse para un número mayor de decisiones (en una misma línea), se recomienda un máximo de dos comparaciones para no complicar el programa y para que la línea no sea excesivamente larga.

PARA RECORDAR

SENTENCIAS MULTIPLES (:)

Las sentencias múltiples son muy convenientes para disponer muchas expresiones de asignación cortas en una misma línea o cuando son muchos los mandos que deben colocarse a continuación de la sentencia THEN.

Ejemplo 1)

```
10 A=1
20 B=2   ⇒ 10 A=1:B=2:C=3
30 C=3
```

Ejemplo 2)

```
50 C=A+B
60 D=A-B   ⇒ 50 C=A+B:D=A-B:E=A*B:F=A/B
70 E=A*B
80 F=A/B
```

Siempre que use una sentencia múltiple a continuación de THEN, las instrucciones que se encuentran después de esta sentencia se ejecutan sólo si se satisface la condición a continuación de IF. Toménse, entonces, las precauciones del caso.

Ejemplo 3)

```
IF A<0 THEN A=10:B=20:.....
```

Se ejecutan sólo si se cumple la comparación.

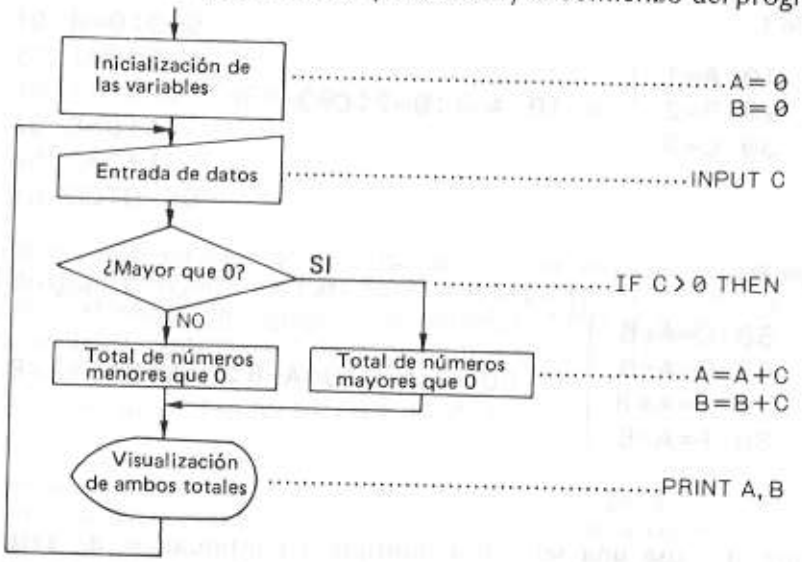
Si bien las sentencias múltiples son muy convenientes en la preparación de programas, no se debe abusar de su uso, y en posible mantener las líneas a una longitud razonable.

[EJERCICIOS]

Prepare un programa para entrar números y agrupar por un lado los mayores que cero y por el otro los menores que cero.

<PLANTEO>

Una vez entrado el número, se juzga a qué grupo pertenece mediante una sentencia IF y se lo suma a dicho grupo. Las variables utilizadas para obtener los totales se llevan a cero (inicializan) al comienzo del programa.



PROGRAMA

```
10 A=0:B=0
20 INPUT C
30 IF C>0 THEN A=A+C:GOTO 50
40 B=B+C
50 PRINT A;B
60 GOTO 20
```

La sentencia IF de la línea 30 juzga si el valor entrado (en la variable C) es o no mayor que 0. Si los es, el mismo se suma a la variable A (a continuación de THEN). En caso contrario, se suma a la variable B mediante la línea 40. La línea 50 visualiza cada total por cada entrada de un número.

3-3-3 Repetición de un ciclo (sentencias FOR-NEXT)

La combinación de las sentencias GOTO e IF para repetir una serie de instrucciones una cantidad de veces determinada es en algunos muy complicado y, además, el programa se extiende más de lo necesario. Siempre que se sepa el número de veces que debe repetirse determinado ciclo, se pueden utilizar las sentencias FOR-NEXT para simplificar el programa y hacerlo más corto.
Estas sentencias hacen que se repitan el número especificado de veces las instrucciones incluidas entre FOR y NEXT.
El formato de la sentencia FOR es el siguiente:

FOR variable de control = valor inicial TO valor final STEP incremento

He a continuación del formato de la sentencia NEXT.

NEXT variable de control

Como variable de control para FOR se pueden usar solamente variables numéricas de una sola letra (como ser A, B, etc.). No pueden utilizarse variables de matriz. Los valores iniciales y final, y el incremento pueden ser expresiones o variables numéricas. La variable de control va cambiando en base al incremento desde el valor inicial hasta el final. El incremento es igual a 1 siempre que se lo omita.
Entre y ejecute el programa siguiente para comprender la mecánica de las sentencias FOR-NEXT.

```
10 INPUT A
20 FOR B=1 TO 10 STEP A
30 PRINT B
40 NEXT B
50 GOTO 10
```

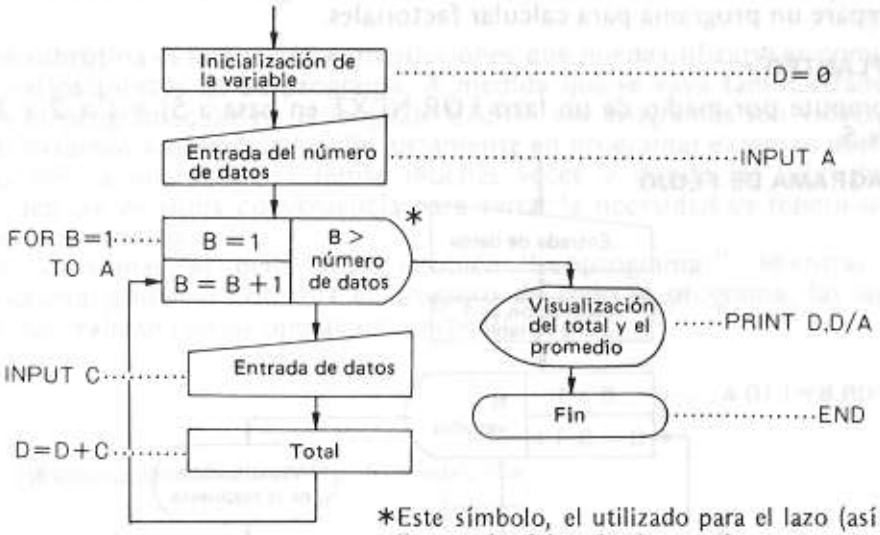
Este programa se ejecuta del siguiente modo:

Operación	Pantalla
RUN [EXE]	?
3 [EXE]	1
[EXE]	4
[EXE]	7
[EXE]	10
[EXE]	?
0.8 [EXE]	1
[EXE]	1.8
[EXE]	2.6
[EXE]	3.4
⋮	⋮

Ejemplo) Prepare un programa para obtener el total y el promedio de un número específico de datos entrados.

Para este programa, entre primero el número de datos y luego entre cada dato y obtenga el total utilizando para ello las sentencias FOR-NEXT. Una vez finalizada la entrada de los datos, haga que se visualicen el total y el promedio.

El diagrama de flujo sería como a continuación.



*Este símbolo, el utilizado para el lazo (así se llama al ciclo de instrucciones entre las sentencias) FOR-NEXT, indica que la variable B va aumentando de a 1 y que, una vez que supera el número de datos, hace que el programa salga del lazo FOR-NEXT para pasar a la siguiente operación.

Preparemos, entonces, el programa en base a este diagrama de flujo.

```
10 D=0
20 INPUT A
30 FOR B=1 TO A
40 INPUT C
50 D=D+C
60 NEXT B
70 PRINT D,D/A
80 END
```

Siempre que se sepa el número de veces que debe repetirse el lazo, como en el caso anterior, la operación puede realizarse mediante las sentencias FOR-NEXT. El número de datos podría también ser fijo, lo que evitaría la necesidad de entrarlo (línea 20). En dicho caso, la línea 20 ya no sería necesaria.

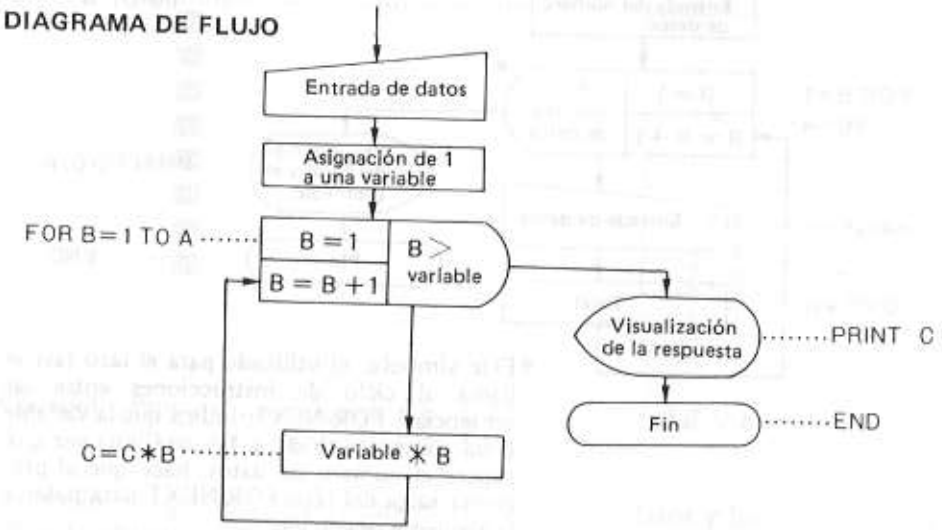
[EJERCICIOS]

Prepare un programa para calcular factoriales.

<PLANTEO>

Compute por medio de un lazo FOR-NEXT en base a 5! = 1 x 2 x 3 x 4 x 5.

DIAGRAMA DE FLUJO



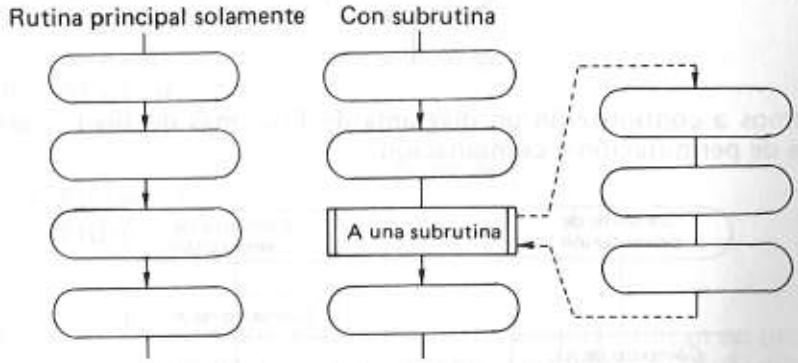
PROGRAMA

```
10 INPUT A
20 C=1
30 FOR B=1 TO A
40 C=C*B
50 NEXT B
60 PRINT C
70 END
```

En la línea 20, se prepara la variable C para que tenga un valor inicial igual a 1. De lo contrario, la respuesta obtenida puede ser incorrecta.
El cálculo del factorial se lleva a cabo mediante las sentencias FOR-NEXT desde la línea 30 hasta la 50. En ellas el valor de B va incrementando de a 1, y el factorial se obtiene computando 1 x 2 x 3 x ... Una vez completado el lazo, se dan por terminados tanto el cálculo como el programa, ya que la línea que sigue contiene una sentencia END. Si de deseara repetir el programa desde el comienzo indefinidamente, la línea 70 deberá ser "GOTO 10".

3-3-4 Subrutinas (sentencias GOSUB y RETURN)

Una subrutina es una serie de instrucciones que puede utilizarse en común en varios puntos de un programa. A medida que se vaya familiarizando con la programación en el lenguaje BASIC, sus programas son volverán más extensos y complicados. Es justamente en programas extensos donde una misma operación se repite muchas veces y donde las subrutinas pueden ser de suma conveniencia para evitar la necesidad de repetir una misma operación.
Las subrutinas se denominan también "subprogramas". Mientras el programa principal controla el progreso de todo el programa, las subrutinas realizan ciertas operaciones específicas.



Veamos un ejemplo práctico de la función que cumple una subrutina.

Ejemplo) Prepare un programa para obtener permutaciones y combinaciones.

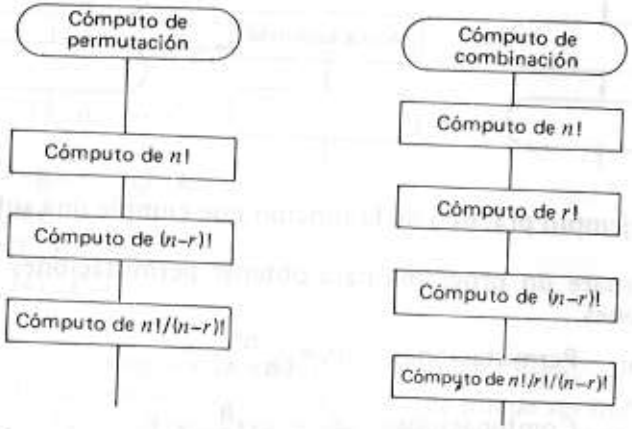
Expresión: Permutaciones $_nP_r = \frac{n!}{(n-r)!}$

Combinaciones $_nC_r = \frac{n!}{r!(n-r)!}$

Este programa permite calcular permutaciones y combinaciones mediante la entrada de dos datos: (n, r).

Prepare un diagrama de fiujo simple como el siguiente.

Preparemos a continuación un diagrama de flujo más detallado para los cálculos de permutación y combinación.



Como puede observarse, el cálculo de $n!$ y $(n-r)!$ se utilizan tanto en el cómputo de las permutaciones como en el de las combinaciones. Cualquier duda sobre el cómputo de factoriales, remítase al programa de la sección anterior, donde se trató el cálculo de factoriales. El programa se extendería en exceso si se llevara a cabo tres veces el cálculo de factoriales.

EJEMPLO (1)

```
 10 INPUT N,R
 20 A=1
 30 FOR B=1 TO N
 40 A=A*B
 50 NEXT B
 60 E=A
 70 A=1
 80 FOR B=1 TO N-R
 90 A=A*B
100 F=A
110 NEXT B
120 P=E/F
130 A=1
140 FOR B=1 TO R
150 A=A*B
160 NEXT B
170 G=A
180 C=E/G/F
190 PRINT P,C
200 END
```

Cómputo de $n!$ (líneas 10–60)

Cómputo de $(n-r)!$ (líneas 70–110)

120 ····· Cómputo de $\frac{n!}{(n-r)!}$

Cómputo de $r!$ (líneas 130–170)

180 ····· Cómputo de $\frac{n!}{r!(n-r)!}$

VARIABLES

N : n
R : r
P : Permutaciones
C : Combinaciones
A : Para factoriales
B : Para el lazo FOR-NEXT
E : $n!$
F : $(n-r)!$
G : $r!$

En este programa, los tres cómputos de factoriales utilizan un programa en común, excepto para el valor final en la sentencia FOR. Si este valor final pudiera controlarse por medio de una variable común, estas tres operaciones podrían llevarse a cabo por medio de una única subrutina. Es aquí donde las subrutinas son muy convenientes.
Por lo tanto, para los valores finales haga uso de la variable H, a fin de utilizar una única subrutina. Para ello, es necesario entrar por anticipado en la variable H los valores de n, $n-r$ y r.
De este modo, cada vez que sea necesario, se utiliza un mando para saltar a una subrutina, al final de la cual se retorna al programa principal por medio de otro mando. Los mandos para saltar a la subrutina y retornar desde la misma al programa principal son GOSUB y RETURN, respectivamente.

EJEMPLO (2)

```
 10 INPUT N,R
 20 H=N
 30 GOSUB 150
 40 E=A
 50 H=N-R
 60 GOSUB 150
 70 F=A
 80 P=E/F
 90 H=R
100 GOSUB 150
110 G=A
120 C=E/G/F
130 PRINT P,C
140 END
150 A=1                    }
160 FOR B=1 TO H           }
170 A=A*B                  } Subrutina
180 NEXT B                 }
190 RETURN                 }
```

De este modo, siempre que se pueda utilizar en un programa una parte en común, hágalo incorporando dicha porción en una subrutina.
Si bien en nuestro ejemplo el uso de la subrutina permite acortar el programa en sólo una línea, en programas más extensos el ahorro es mucho mayor.

[EJERCICIOS]

Prepare un programa para calcular desviaciones estándar. La entrada de datos, los cómputos de las sumatorias, la elevación al cuadrado de las mismas y la cuenta del número de datos se incorporan en una subrutina.

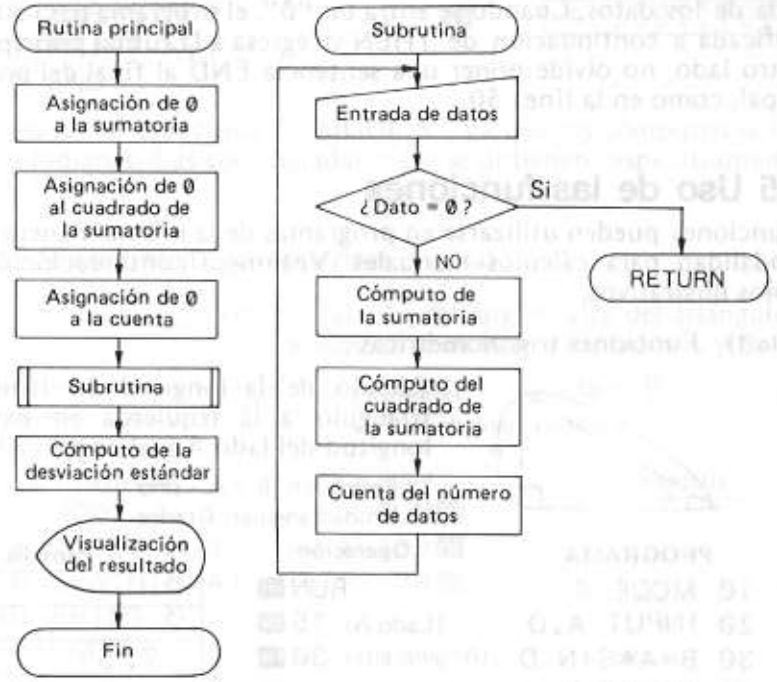
Expresión:

$$\sigma n = \sqrt{\frac{\Sigma x^2 - (\Sigma x)^2/n}{n}}$$

Σx : Sumatoria
Σx^2 : Cuadrado de la sumatoria
n : Número de datos

<PLANTEO>

Si se entra un cero para dar por terminada la entrada de los datos, una sentencia IF hace que se salte al programa principal para obtener la desviación estándar. Para el cálculo del cuadrado, se utiliza SQR.



PROGRAMA

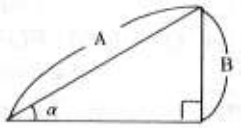
```
 10 B=0:C=0:D=0
 20 GOSUB 100 ..... Hacia la subrutina
 30 E=SQR((C-B*B/D)/D) ..... Desviación estándar
 40 PRINT E
 50 END
100 INPUT A                                    }
110 IF A=0 THEN RETURN                         }
120 B=B+A ..... Sumatoria                      } Subrutina
130 C=C+A*A ..... Sumatoria al cuadrado        }
140 D=D+1 ..... Número de datos                }
150 GOTO 100                                   }
```

En este programa, la línea 20 transfiere la ejecución a la subrutina. Luego, se entran los datos y se obtienen la sumatoria, el cuadrado de la sumatoria y el número de datos, todo ello mediante la subrutina que se encuentra a partir de la línea 100.
La sentencia IF en la línea 110 es la que determina si ha finalizado la entrada de los datos. Cuando se entra un "0", el programa pasa a la línea especificada a continuación de THEN y regresa a la rutina principal.
Por otro lado, no olvide poner una sentencia END al final del programa principal, como en la línea 50.

3-3-5 Uso de las funciones

Las funciones pueden utilizarse en programas de la misma manera que en la modalidad para cálculos manuales. Veamos a continuación algunos ejemplos ilustrativos.

Ejemplo 1) Funciones trigonométricas



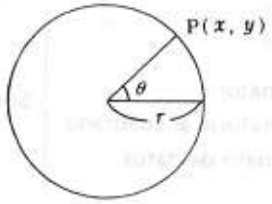
Cálculo de la longitud del lado B del triángulo a la izquierda en base a la longitud del lado A y al ángulo alfa.

Expresión: B = A · sinα
Unidad angular: Grados

PROGRAMA	Operación	Pantalla
10 MODE 4	RUN EXE	?
20 INPUT A,D	(Lado A) 15 EXE	?
30 B=A*SIN D	(Angulo alfa) 30 EXE	7.5
40 PRINT B		
50 END		

La unidad angular se especifica mediante la línea 10. Como los cálculos se llevan a cabo en grados, la unidad angular se especifica mediante "MODE 4". En la línea 30 se utilizan funciones trigonométricas para calcular la longitud del lado A.

Ejemplo 2) Función trigonométrica



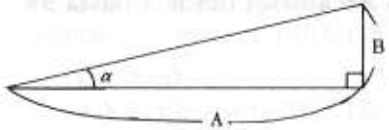
Cálculo de las coordenadas (x, y) correspondientes al punto P en la circunferencia de la izquierda en base al radio r y al ángulo θ.

Expresión : $x = r \cdot \cos\theta$
$y = r \cdot \sin\theta$
Unidad angular: Radianes

PROGRAMA	Operación	Pantalla
10 MODE 5	RUN EXE	?
20 INPUT R,T	(Radio) 5 EXE	?
30 X=R*COS T	(Angulo θ) π /3 EXE	2.5
40 Y=R*SIN T	EXE	4.330127019
50 PRINT X,Y		
60 END		

En la línea 10 se especifica el "MODE 5", ya que los cómputos se llevan a cabo en radianes. Las coordenadas x e y se obtienen respectivamente en las líneas 30 y 40.

Ejemplo 3) Funciones trigonométricas inversas



Cálculo del ángulo alfa del triángulo a la izquierda en base a los lados A y B.

Expresión: $\alpha = \tan^{-1}\frac{B}{A}$
Unidad angular: Grados

PROGRAMA	Operación	Pantalla
10 MODE 4	RUN EXE	?
20 INPUT A,B	(Lado A) 100 EXE	?
30 D=ATN(B/A)	20 EXE	11.30993247
40 PRINT D		
50 END		

Ejemplo 4) Conversión decimal ↔ sexagesimal

Prepare un programa para calcular la hora.

PROGRAMA	Operación	Pantalla
10 T=0	RUN EXE	?
20 INPUT D,E,G	(Horas) 1 EXE	?
30 S=SGN D	(Minutos) 25 EXE	?
40 D=ABS D	(Segundos) 36 EXE	1°25'36
50 T=T+S*DEG(D,E,G)	EXE	?
60 PRINT DMS$(T)	(Horas) 2 EXE	?
70 GOTO 20	(Minutos) 15 EXE	?
	(Segundos) 5 EXE	3°40'41

Entre la hora, los minutos y los segundos en las variables D, E y G respectivamente mediante la línea 20.
Las líneas 30 y 40 se utilizan para operaciones de resta en caso de que se entre un número negativo para la hora. En caso de ser un número positivo el entrado, no se lleva a cabo operación alguna.
El total se obtiene en la línea 50. Cuando se entra un 1 en la variable S, se lleva a cabo una suma; cuando a S se le asigna un −1, la operación realizada es una resta. La función DEG convierte las horas, minutos y segundos a una cifra decimal. El total se obtiene también en decimal.
La línea 60 incorpora la función DMS$, la cual convierte la cifra decimal en sexagesimal para su visualización.

Ejemplo 5) Generación de números aleatorios

Prepare un programa para generar números alteatorios desde 1 hasta 99.

PROGRAMA

```
10 R=INT(RAN#*99)+1
20 PRINT R
30 GOTO 10
```

Los número aleatorios se obtienen en la línea 10. Como el número aleatorio debe ser igual o mayor que 1 e igual o menor que 99, multiplique RAN# por 99 y súmele 1.

Para números aleatorios desde 5 hasta 9 → INT(RAN#*5)+5

Para números aleatorios desde 10 hasta 20 → INT(RAN#*11)+10

3-3-6 Uso de matrices

Las variables matriciales (o de matriz) son aquellas variables que adoptan un nombre único y se diferencian de las demás por un subíndice a continuación del nombre de la variable. Su uso, por otro lado, también es diferente al de las variables comunes.
Por lo general, como variables matriciales, se utilizan variables comunes, de la A hasta la Z, con el subíndice a continuación.

Ejemplo)

```
A(1)
 └──Subíndice
└────Nombre de la variable
```


Se recomienda aprender el uso de las variables de matriz, ya que las mismas permiten el tratamiento de información en grandes cantidades. Por ejemplo, para entrar 10 datos en una variable de matriz:

```
10 FOR A=1 TO 10
20 INPUT N(A)
30 NEXT A
```

En este caso, a cada elemento de la matriz se asignarían los datos del siguiente modo.

Variable común	O	P	Q	R	S	T	U	V	W	X
Variable matricial	N(1)	N(2)	N(3)	N(4)	N(5)	N(6)	N(7)	N(8)	N(9)	N(10)

Para, por ejemplo, tomar el dato cuyo valor sea el mayor entre toda la información, podría utilizarse el siguiente programa.

```
100 A=0
110 FOR B=1 TO 10
120 IF N(B)>A THEN A=N(B)
130 NEXT B
140 PRINT A
```

No obstante su conveniencia, deben tomarse precauciones en cuanto a la disposición de los elementos de la matriz. Ciertas posiciones de memoria utilizadas para las variables matriciales se utilizan también para las variables comunes. Por ejemplo, la porción de memoria para la variable común A es la misma que la utilizada para el elemento 0 de la variable de matriz A, o sea A (0). Lo mismo sucede con B y A (1). Es aquí donde una debe tener cuidado, de no superponer las variables comunes con las de matriz. La relación entre las porciones de memoria donde se almacena el contenido de uno y otro tipo de variables es la siguiente.

```
 A      B      C      D                    X       Y       Z
A(0)   A(1)   A(2)   A(3)·····           A(23)   A(24)   A(25)
       B(0)   B(1)   B(2)·····           B(22)   B(23)   B(24)
                                                  :
                                          X(0)    X(1)    X(2)
                                                  Y(0)    Y(1)
                                                          Z(0)
```

* Para mayores detalles, vea la página 166.

Llevemos a cabo las siguientes operaciones:

Asigne un 7 a la variable I.

```
I=7 EXE
```

Confirme el contenido de la variable I.

```
I EXE
 7
```

Asigne un 10 al 9º elemento de la variable de matriz A.

```
A(8)=10 EXE
```

Confirme el contenido de la variable I.

```
I EXE
 10
```

Al corroborar el contenido de la variable I por segunda vez, después de asignar 10 a la variable A (8), se puede observar que el mismo ha cambiado. Ello se debe a que la porción de memoria donde se almacena el valor asignado a estas dos variables en cuestión es la misma. Siempre que prepare un programa donde se utilicen variables de matriz y sea necesario usar variables comunes, hágalo cuidando que las mismas no se superpongan.

Ejemplo 1) Cuando se use una matriz de 10 elementos:
Variables de matriz: A (0) hasta A (9)
Variables comunes: K hasta Z

Ejemplo 2) Cuando se necesiten una matriz de 50 elementos y 15 variables comunes:
Variables comunes: A hasta O
Variables de matriz: P (0) hasta P (49) [Entre DEFM 39 EXE .]

Ejemplo 3) Cuando se usa la variable B para lazos FOR-NEXT, la variable B para obtener el total y una matriz de 50 elementos:

```
10 DEFM 26          ..... Agregue 26 variables (52 total)
20 B=0
30 FOR A=0 TO 49
40 INPUT C(A)       ..... Pueden usarse las variables
50 B=B+C(A)               de matriz C (0) hasta C (49).
60 NEXT A
```


De este modo, se puede evitar la superposición de variables matriciales con las variables comunes siempre que sea necesario.

En el caso de que el número de elementos de la matriz más el número necesario de variables comunes exceda el número máximo de variables comunes (A hasta Z), será necesario ampliar la memoria. Esta expansión se lleva a cabo mediante el uso del mando DEFM, por medio del cual se especifica el número de variables (o elementos de matriz) que desean agregarse.
Por ejemplo, para agregar 10 variables:

Manualmente: DEFM 10 EXE
En un programa: Número de línea DEFM 10

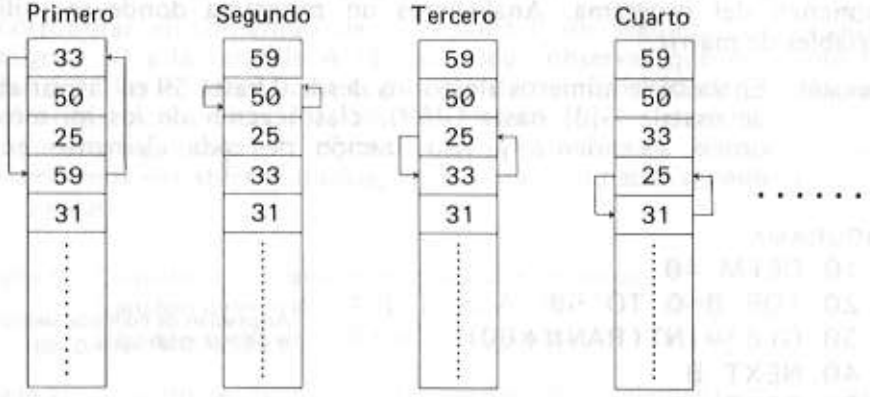
Si bien este mando pueden usarse en ambos modos manual y por programa, se recomienda, siempre que sea necesario, incorporarlo al comienzo del programa. Analicemos un programa donde se utilizan variables de matriz.

Ejemplo) Entrada de números aleatorios desde 0 hasta 59 en las variables de matriz G(0) hasta G(59), clasificación de los mismos en orden ascendente y visualización de cada elemento en el mismo orden.

PROGRAMA

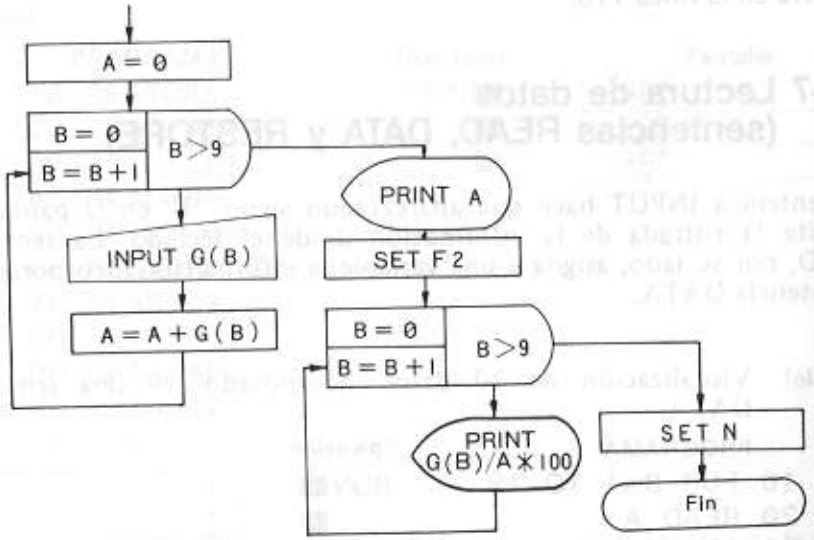
```
10 DEFM 40
20 FOR B=0 TO 59
30 G(B)=INT(RAN#*60)        } Asignación de números aleatorios 0 a 59 en G(0) hasta G (59).
40 NEXT B
50 BEEP 1
60 FOR B=0 TO 59
70 A=-1
80 FOR C=B TO 59
90 IF G(C)>A THEN A=G(C):D=C   } Ordenamiento en orden ascendente
100 NEXT C
110 E=G(B):G(B)=G(D):G(D)=E
120 NEXT B
130 BEEP 0
140 FOR B=0 TO 59
150 PRINT G(B)              } Visualización secuencial
160 NEXT B
170 END
```

Este programa se compone de tres partes principales. En la primera, se generan números aleatorios desde 0 hasta 59, asignándose cada uno de ellos a los elementos de una matriz, G(0) hasta G(59). En la segunda parte, se ordenan estos elementos en orden ascendente. En la tercera parte, se visualizan los datos en el orden que fueron clasificados.
Las sentencias BEEP en las líneas 50 y 130 se utilizan para generar un sonido. El sonido generado es agudo cuando se utiliza BEEP 1 y es grave cuando se usa BEEP 0. Como estas dos línea se utilizan sólo para generar un sonido a fin de indicar la finalización de los procesos de asignación de los datos y ordenamiento de los mismos, no son realmente necesarios.
Las líneas 60 hasta 120 se utilizan para clasificar los datos. Para ello, se repite un procedimiento en el cual se selecciona el dato mayor al primer elemento de las variables de matriz, el siguiente al segundo elemento, y así sucesivamente.



La sentencia DEFM de la línea 10 se utiliza para ampliar la memoria reservada para las variables. Como las variables de la G hasta la Z ocupan las posiciones de memoria correspondientes a los primeros 20 elementos de las variables matriciales G(0) hasta G(59), se agregan 40 variables.
Las variables matriciales son de gran conveniencia para trabajar con información en grandes cantidades.

[EJERCICIO]
Entre 10 datos y obtenga la relación de componentes para el total y para cada dato. La relación de componente será un porcentaje con dos posiciones decimales.



PROGRAMA

```
 10 A=0
 20 FOR B=0 TO 9
 30 INPUT G(B)
 40 A=A+G(B)
 50 NEXT B
 60 PRINT A
 70 SET F2
 80 FOR B=0 TO 9
 90 PRINT G(B)/A*100
100 NEXT B
110 SET N
120 END
```

Las líneas 20 hasta la 50 componen un lazo FOR-NEXT que se utiliza para la entrada de los 10 datos, cada uno de ellos en un elemento de las variables matriciales G(0) hasta G(9). "SET F2" en la línea 70 especifica la cantidad de posiciones decimales para el porcentaje para presentar la relación de componente. Las líneas 80 hasta 100 son las que se encargan de visualizar los porcentajes. La especificación de las posiciones decimales se libera en la línea 110.

3-3-7 Lectura de datos (sentencias READ, DATA y RESTORE)

La sentencia INPUT hace que aparezca un signo "?" en la pantalla y permite la entrada de la información desde el teclado. La sentencia READ, por su lado, asigna a una variable la información incorporada en la sentencia DATA.

Ejemplo) Visualización de 10 datos incorporados en una sentencia DATA.

PROGRAMA

```
10 FOR B=1 TO 10
20 READ A
30 PRINT A
40 NEXT B
50 DATA 1,2,3,4,5,
   6,7,8,9,10
60 END
```

Operación

RUN [EXE]
[EXE]
[EXE]
⋮
[EXE]

1
2
3
⋮
10

La sentencia DATA asigna a la variable que le sigue la información que lee de la sentencia DATA. La sentencia READ no puede usarse si no hay sentencia DATA alguna en el programa.
A continuación de READ puede colocarse más de una variable, separadas entre sí por una coma.

Ejemplo)

PROGRAMA

```
10 READ A$,B$
20 PRINT A$;B$
30 END
40 DATA CASIO,PB & FX
```

Operación

RUN [EXE]

Pantalla

CASIOPB&FX

La sentencia DATA puede colocarse en cualquier parte del programa. El programa, al ejecutarse, va leyendo la información en las sentencias por orden ascendente del número de línea y por orden de aparición en una misma línea. Para datos numéricos, use en la sentencia READ variables numéricas. Para información de caracteres, incorpore en la sentencia READ variables de carácteres.

Ejemplo)

PROGRAMA

```
10 RESTORE
20 READ A$
30 PRINT A$
40 READ B
50 PRINT B
60 RESTORE 90
70 READ C$
80 PRINT C$
90 DATA ABC
100 DATA 123456
110 END
```

Operación

RUN [EXE]
[EXE]
[EXE]

Pantalla

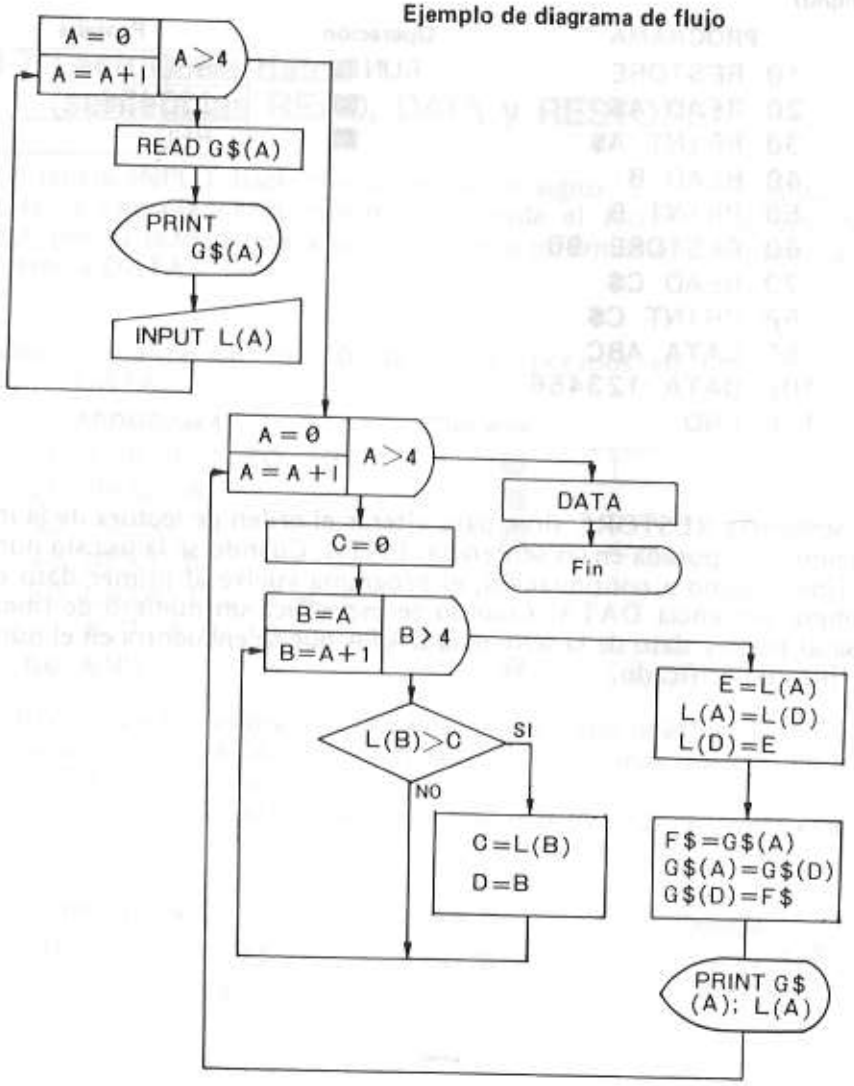
ABC
123456
ABC

La sentencia RESTORE sirve para alterar el orden de lectura de la información incorporada en las sentencias DATA. Cuando se la usa sin número de línea alguno a continuación, el programa vuelve al primer dato de la primera sentencia DATA. Cuando se especifica un número de línea, se pasa al primer dato de la sentencia DATA que se encuentra en el número de línea especificado.

[EJERCICIO]

Asigne los nombres de las ciudades CHICAGO, LONDON, PARIS, ROMM y TOKIO a cinco variables de matriz y clasifíquelas por orden alfabético, entrando para ello datos agregados.
Tenga en cuenta que los nombres de las ciudades se leen en G$(0) hasta G$(4), mientras que los datos se entran en L(0) hasta L(4).

Ejemplo de diagrama de flujo



EJEMPLO DE PROGRAMA

```
 10 FOR A=0 TO 4
 20 READ G$(A)
 30 PRINT G$(A);
 40 INPUT L(A)
 50 NEXT A
 60 FOR A=0 TO 4
 70 C=0
 80 FOR B=A TO 4
 90 IF L(B)>C THEN C=L(B):
    D=B
100 NEXT B
110 E=L(A):L(A)=L(D):L(D)
    =E
120 F$=G$(A):G$(A)=G$(D):
    G$(D)=F$
130 PRINT G$(A);L(A)
140 NEXT
150 DATA CHICAGO, LONDON, PARIS, ROME, TOKYO
```

VARIABLES

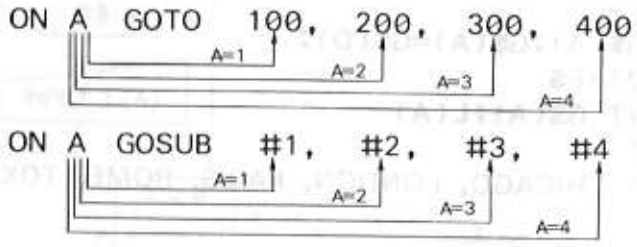
A : Para las sentencias FOR-NEXT
B : Para las sentencias FOR-NEXT
C : Valor máximo
D : Número de valores máximos
E : Para ordenamiento
F$: Para ordenamiento
G$(0)—G$(4) : Nombres de ciudades
L(0)—L(4) : Datos

Este programa se divide en dos secciones: la de entrada, compuesta por las líneas 10 hasta la 50, y la de ordenamiento, desde la línea 60 hasta la 140. En la primera, se leen los nombres de las ciudades incorporados en la sentencia DATA repitiendo 5 veces un lazo FOR-NEXT, mediante el cual también se entran los datos. La sentencia PRINT en la línea 30 visualiza los nombres de las ciudades como un mensage antes de entrar los datos por medio de la sentencia INPUT que se encuentra a continuación. Si bien debe ordenarse como en casos anteriores, este programa utiliza el método que puede observarse en las líneas 110 y 120, ya que deben clasificarse tanto los nombres como los datos.
Si bien la sentencia DATA de la línea 150 puede ubicarse en cualquier parte del programa, es más fácil identificarla si se la coloca al final del programa.

3-3-8 Saltos indirectos (sentencias ON-GOTO y ON-GOSUB)

Ya hemos visto anteriormente las funciones de las sentencias GOTO y GOSUB. Estas sentencias permiten saltar a diferentes partes del programa especificando el número de línea correspondiente a continuación de la sentencia misma. No obstante, en muchos casos, en aquellos casos donde es necesario decidir el destino del salto en base al resultado de determinada operación, la combinación de una de estas sentencias con la sentencia IF no es del todo conveniente.
Para determinar el destino de un salto en base a cierto resultado, se utilizan los denominados saltos indirectos, los cuales se realizan por medio de las sentencias ON-GOTO u ON-GOSUB. La función de estas dos sentencias es muy similar.

Ejemplo)



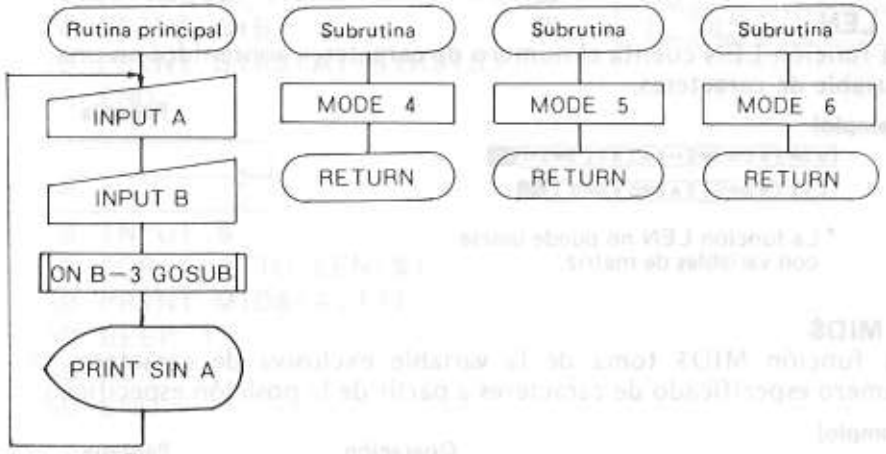
Cuando el contenido de la variable que se encuentra entre ON y GOTO o GOSUB (A en este caso) es igual a 1, se salta al primer destino especificado; cuando el contenido es igual a 2, se salta al segundo destino, y así sucesivamente. Cuando el contenido de la variable excede el número de destinos especificados, el programa avanza a la siguiente línea o mando (en el caso de una sentencia múltiple).

Ejemplo)

```
 10 INPUT A
 20 ON A GOTO 100,200,300,400,500
 30 PRINT " NO"
 40 END
100 PRINT "LINE 100" :END
200 PRINT "LINE 200" :END
300 PRINT "LINE 300" :END
400 PRINT "LINE 400" :END
500 PRINT "LINE 500" :END
```


[EJERCICIO]
Entre un ángulo y la unidad angular deseada mediante los valores numéricos 4 al 6, provoque un salto condicional a la subrutina que especifique la unidad angular y obtenga el seno del ángulo en la unidad especificada.

Diagrama de flujo



PROGRAMA

```
 10 INPUT "ANGLE",A
 20 INPUT "UNIT",B
 30 ON B-3 GOSUB 100,200,300
 40 PRINT SIN A
 50 GOTO 10
100 MODE 4
110 RETURN
200 MODE 5
210 RETURN
300 MODE 6
310 RETURN
```

Como en este programa se entran dos datos por vez, cada entrada se identifica por medio de un mensaje en cada sentencia INPUT. En la línea 10, se asigna el ángulo a la variable A; en la línea 20, se especifica la unidad angular (página 131) por medio de un número (4, 5 ó 6), el cual se asigna a la variable B. En la línea 30, se determina el destino del salto (ON-GOSUB) en base al valor numérico entrado para especificar la unidad angular.
En al subrutina, se especifica cada unidad angular y se regresa al punto de origen.

3-3-9 Funciones de caracteres (LEN, MID$, VAL y STR$)

Del mismo modo que hay funciones numéricas, como SIN y COS, también hay funciones de caracteres. Estas son: LEN, MID$, VAL y STR$.

• LEN

La función LEN cuenta el número de caracteres contenidos en una variable de caracteres.

Ejemplo)

[A][SHIFT][$][=][SHIFT]["][A][B][C][SHIFT]["][EXE]
[L][E][N][SHIFT][(][A][SHIFT][$][SHIFT][)][EXE]

Pantalla

```
_
3
```

* La función LEN no puede usarse con variables de matriz.

• MID$

La función MID$ toma de la variable exclusiva de caracteres ($) el número especificado de caracteres a partir de la posición especificada.

Ejemplo)

```
10 $="CASIO PB&FX"
20 PRINT $
30 PRINT MID$(1,5)
40 PRINT MID$(7,5)
50 END
```

Operación

RUN[EXE]
[EXE]
[EXE]

Pantalla

```
CASIO PB&FX
CASIO
PB&FX
```

• VAL

La función VAL convierte en valores numéricos los números contenidos en una variable de caracteres como caracteres.

Ejemplo)

```
10 A$="123":B$="456"
20 PRINT A$+B$
30 PRINT VAL(A$)
   +VAL(B$)
40 END
```

Operación

RUN[EXE]
[EXE]

Pantalla

```
123456
 579
```

* Las función VAL no puede usarse con variables de matriz.

• STR$

A la inversa de la función VAL, la función STR$ convierte valores numéricos almacenados en una variable numérica en números.

Ejemplo)

```
10 A=123:B=456
20 PRINT A+B
30 PRINT STR$(A)+STR$(B)
40 END
```

Operación

RUN[EXE]
[EXE]

Pantalla

```
 579
123456
```

Ejemplo)

```
10 INPUT $
20 FOR A=1 TO LEN($)
30 PRINT MID$(A,1);
40 BEEP 1
50 NEXT A
60 END
```

Este programa toma por medio de la función MID$ un carácter por vez entrado en la variable exclusiva de caracteres. Si bien la posición del carácter la especifica la sentencia FOR-NEXT, el valor final lo determina la función LEN.
Al final de la línea 30 se agrega un punto y coma (;) para que la visualización sea continua.

Ejemplo)

```
 10 A=1:B=0
 20 PRINT "<";STR$(A);">";
 30 INPUT $
 40 IF $="END" THEN 100
 50 B=B+VAL($)
 60 A=A+1
 70 GOTO 20
100 PRINT B/(A-1)
110 END
```

Este programa obtiene el promedio de un número desconocido. La entrada de los datos se da por terminada entrando "END" y el promedio se visualiza saltando a la línea 100.
La línea 20 proporciona un mensaje que facilita la identificación de la información que debe entrarse.
En la línea 50, como los datos se entran como caracteres en la variable exclusiva de caracteres $, el total se obtiene convirtiendo los datos entrados en valores numéricos.
Por otro lado, este programa provoca un error siempre que se intente la entrada de caracteres alfabéticos que no sean "END".

3-3-10 Convenientes funciones para entrada y salida (KEY$ y CSR)

Si bien la función KEY$ se utiliza como la sentencia INPUT para la entrada de datos desde el teclado, las mismas difieren en lo siguiente.

Sentencia INPUT	Función KEY$
• Mantisa de hasta 12 dígitos y exponente de hasta 2 dígitos (valores numéricos). • Variables de hasta 7 caracteres. Hasta 30 caracteres en el caso de la variable exclusiva $.	• Asigna el carácter correspondiente a la tecla pulsada a la variable de caracteres especificada.
• El signo "?" visualizado en la pantalla indica un modo de espera de entrada desde el teclado.	• Sin modo de espera de entrada desde el teclado.

Ejemplo)

```
10 INPUT A
20 PRINT A
30 B$=KEY$
40 IF B$="" THEN 30
50 PRINT B$
60 END
```


La línea 10 usa una sentencia INPUT mientras que las líneas 30 y 40 utilizan la función KEY$. Si bien esta función permite la entrada de un carácter desde el teclado, no se entra en un modo de espera de entrada, púlsese o no tecla alguna. Es por ello que se la combina con la sentencia IF, como en la línea 40, para ver si se ha pulsado alguna tecla o no. En caso de no haberse pulsado tecla alguna, el programa retorna a la línea 30.
De tal modo, la función KEY$ se utiliza en combinación con la sentencia IF del modo ilustrado en el programa anterior.

Ejemplo)

```
10 A$=KEY$
20 IF A$="1" THEN 100
30 IF A$="2" THEN 200
40 IF A$="3" THEN 300
50 GOTO 10
100 PRINT "LINE 100":END
200 PRINT "LINE 200":END
300 PRINT "LINE 300":END
```

Este programa también ejemplifica el uso de la función KEY$, sólo que en este caso se verifica si la entrada es un 1, un 2 o un 3. En caso contrario, el programa regresa a la línea donde se encuentra la función KEY$.
Siempre que se utilice la función KEY$ al comienzo de un programa, debe prestarse atención a lo siguiente. Para iniciar la ejecución de un programa, pueden utilizarse dos métodos. Es justamente cuando se utilizan las teclas [SHIFT] [P1] combinadas que si la tecla [1] no se suelta inmediatamente, la función KEY$ del programa podrá considerar dicha pulsación como válida.
Probemos, por ejemplo, almacenando este programa en el área P1 y ejecutándolo por medio de [SHIFT] [P1]. Siga pulsando la tecla [1] aún iniciado el programa y verá cómo el mismo visualiza el mensaje "LINEA 100".
Para evitar tal situación, podría ser conveniente agregar las líneas a continuación.

```
 5 A$=KEY$
 6 IF A$<>"" THEN 5
10 A$=KEY$
   :
   :
```

} Espera hasta que se suelte la tecla pulsada.

La función CSR se utiliza en combinación con la sentencia PRINT y especifica la posición de visualización dentro de la pantalla.

Ejemplo)

```
10 PRINT "A"
20 PRINT CSR 2;"A"
30 PRINT CSR 8;"A"
40 END
```

La función de CSR podrá comprenderse fácilmente ejecutando este programa.
Las posiciones especificadas por la función CSR se cuentan a partir del extremo izquierdo de la pantalla. Por ello, para comenzar a visualizar a partir de dicho extremo, esta función puede omitirse.

Ejemplo)

```
10 A=INT(RAN#*12)
20 PRINT CSR A;"↑"
30 GOTO 10
```

Este programa genera números aleatorios del 0 al 11 por medio de la función RAN # y visualiza el signo "↑" haciendo uso de la función CSR. Una vez iniciado el programa, cada pulsación de la tecla [EXE] hace que el signo "↑" se visualice en posiciones aleatorias de la pantalla. Con la combinación de las funciones CSR y KEY$ se podria preparar un juego bastante divertido.

Ejemplo)

```
 10 D=0:$=" 0123456789"
 20 FOR B=1 TO 10
 30 IF KEY$≠" " THEN 30
 40 A=INT(RAN#*10)
 50 PRINT MID$(1,A+1);"↑";MID$(A+3);
 60 FOR C=1 TO 30
 70 E$=KEY$
 80 IF E$≠"" THEN 100
 90 NEXT C
100 IF E$<"0" THEN 130
110 IF E$>"9" THEN 130
120 IF VAL(E$)=A THEN D=D+1:BEEP 1:GOTO 140
130 BEEP 0
140 PRINT
150 NEXT B
160 PRINT CSR 2;"RIGHT";D;
170 IF D≠10 THEN 210
180 FOR B=1 TO 10
190 BEEP 1:BEEP 0
200 NEXT B
210 END
```

Este es un juego en el cual se visualizan números del 0 hasta el 9, uno de los cuales aparece como "↑". Vd. debe pulsar la tecla numérica correspondiente a la posición que aparece el signo "↑". En la línea 40, se visualiza el

En la línea 30, se utiliza KEY$ para hacer que el programa siga su curso sólo si se suelta la tecla pulsada para iniciar su ejecución. Esta es otra aplicación de la función KEY$, la cual asigna a una variable el carácter correspondiente a la tecla pulsada.

Las líneas 60 en adelante se utilizan par decidir si la entrada es correcta, repitiéndose el lazo FOR-NEXT la cantidad de veces especificada. Las líneas 100 y 110 se utilizan para que el programa salte a la línea 140 y considere la respuesta incorrecta siempre que la entrada no sea un número del 0 al 9. Para la comparación de los caracteres que determinan si la respuesta es correcta o no, se utilizan los códigos que le corresponden a cada uno de ellos y que pueden observarse en la página 00. Para mayores detalles, remítase a esta tabla.

La línea 120 verifica si la respuesta es la correcta. Como la función KEY$ lee un carácter, la comparación se lleva a cabo convirtiendo dicho carácter en un valor numérico por medio de la función VAL.

Si bien el método que se observa en la línea 30 es adecuado para juzgar si se ha pulsado o no una tecla, cuando debe compararse la respuesta correcta entre muchas otras, deben tomarse decisiones como las que pueden verse en las líneas 70 hasta 120. Entre este programa desde el teclado.

Ejemplo de operación

Pantalla	Operación
01↑3456789	Pulse [2].
0123456↑↑9	Pulse [8].
0123↑56789	Pulse [4].
⋮	⋮

Si la velocidad a la que cambia lo visualizado en la pantalla es demasiado alta, asigne al valor final de la sentencia FOR en la línea 60 un número mayor que 30.

3-4. USO DE PROGRAMAS DE LA PB-100

Los programas preparados para las computadoras PB-100 y PB-300 pueden usarse también en esta computadora.

Si bien esto es cierto, como esta computadora posee más mandos que las dos mencionadas perviamente, el uso de tales programas no aprovecharía al máximo el potencial de su máquina.

Si bien el lenguaje BASIC utilizado en su computadora es casi igual al incorporado en las PB-300 y PB-100, el primero consta de más mandos.

- **Diferencias**

- **Mandos adicionales**

PASS (claves para programas)
BEEP (señal sonora)
READ (para asignar a variables la información en la sentencia DATA)
DATA (incorporación de datos en un programa)
RESTORE (especificación del orden de obtención de la información en las sentencias DATA)
ON-GOTO (Saltos indirectos por medio de GOTO)
ON-GUSUB (Saltos indirectos por medio de GOSUB)
REM (Sentencia para comentarios en un programa)

- **Funciones adicionales**

DEG (conversión sexagesimal → decimal)
DM$ (conversión decimal → sexagesimal)
STR$ (conversión de un valor numérico en una secuencia de caracteres)

- **Mandos modificados**

Su computadora	PB-100/PB-300
NEW (NEW ALL)	CLEAR (CLEAR A)
CLEAR	VAC
IF-THEN	IF-;
DEFM (puede incluirse en los programas)	DEFM (puede ejecutarse sólo manualmente)

- **Funciones modificadas**

Su computadora	PB-100/PB-300
KEY$	KEY
MID$	MID

- **Comandos borrados**

SAVE (SAVE ALL)
LOAD (LOAD ALL)
PUT
GET
MODE 7
MODE 8

No hay comandos de salida a una impresora ni para salida o entrada a una cinta de cassette. Por lo tanto los comandos PUT, GET, MODE 7 y MODE 8 de los programas de la serie PB-100 deberán ser borrados.

No obstante las diferencias aquí citadas, todos los programas preparados para la PB-100/PB-300 pueden, básicamente, utilizarse en su computadora sin modificación alguna.
De todos modos, siempre que se utilicen tales programas, deberá prestarse atención a que los mismos no incorporen mandos, sentencias o formas de sintaxis incompatibles con el lenguaje BASIC de su máquina. Veamos a continuación algunos ejemplos.

Ejemplo 1)

Programa para la PB-100

```
10 VAC
20 FOR A=1 TO 20
30 INPUT Z(A)
40 IF Z(A)>80:B=B+1:GOTO 90
50 IF Z(A)<60:C=C+1:GOTO 90
60 IF Z(A)>40:D=D+1:GOTO 90
70 IF Z(A)>20:E=E+1:GOTO 90
80 F=F+1
90 NEXT A
    :
    :
```


Este ejemplo es parte de un programa que permite la entrada de información numérica, distribuyendo la misma según su valor. Para poder usar este programa correctamente en su computadora, es necesario realizar ciertas modificaciones que citaremos a continuación.
El mando VAC en la línea 10 debe cambiarse por un mando CLEAR.

```
10 CLEAR
```

El signo ":" en las líneas 40 hasta 70 debe cambiarse por la sentencia THEN.

```
40 IF Z(A)>80 THEN B=B+1:GOTO 90
    :
    :
```

Con estas modificaciones, esta porción de programa podrá utilizarse sin problemas.
Por otro lado, este programa requiere la ampliación de la memoria para variables. Para ello, incorpore el mando DEFM al comienzo del programa (en la PB-100/PB-300, este mando puede ejecutarse sólo manualmente).

```
5 DEFM 20
```

Ejemplo 2)

Programa para la PB-100

```
10 INPUT "I=1/O=2/P=3",N
20 IF N<1 THEN 10
30 IF N>3 THEN 10
40 GOTO N*100
    :
    :
```

Este programa determina los destinos de los saltos en base a las operaciones a realizarse. En el caso de su computadora, este programa podría simplificarse mediante el uso de la sentencia ON-GOTO del siguiente modo.

```
10 INPUT "I=1/O=2/P=3",N
20 ON N GOTO 100,200,300
30 GOTO 10
    :
    :
```

Usando la sentencia GOTO de esta manera se elimina la necesidad de verificar el valor de N.
Tomando en consideración los puntos recién mencionados, los programas para la serie PB-100 pueden ser utilizados en la PB-80. Cualquiera de los programas en el mercado para la serie PB-100 puede ser usado inmediatamente.

[PRECAUCIONES]

- No use los mandos READ #, WRITE # y RESTORE # en aquellos programas preparados en su computadora para usar en otras de la misma marca CASIO.
  Para los modelos PB-100, PB-300, FX-700P y FX-802P, las sentencias KEY$ y MID$ deben cambiarse por KEY y MID, respectivamente.
- Siempre que prepare programas para su computadora en otros modelos de bolsillo de CASIO, tenga en cuenta la diferencia en la sintaxis de la sentencia IF-THEN. Para su computadora, siempre que se especifique un salto a continuación de esta sentencia, es necesario agregar la sentencia GOTO, y no sólo el número de línea (el destino) como puede hacerse en otras computadoras.

CAPITULO 4

MANDOS Y SENTENCIAS

* Para la descripción del formato de sintaxis de los diferentes mandos y sentencias, se han utilizado en este capítulo los siguientes símbolos.

- $\begin{Bmatrix} \times\times\times\times \\ OOOO \end{Bmatrix}$ ..... Debe elegirse uno de los elementos entre { }.
- $\left[OOOO\right]$ ..... El elemento entre [ ] puede omitirse.
- OOOO* ..... Los elementos con un asterisco a su derecha pueden usarse repetidamente.
- Expresiones numéricas .....
  Valores numéricos, expresiones de cálculo y variables numéricas, como ser, 10, 2+3, A ó S*Q.
- Expresiones de caracteres .....
  Constantes, variables y variables de caracteres, como ser, "ABC", X$, N$+M$.
- Parámetros ..... Elementos que acompañan un mando.
- Ⓟ ..... Puede ejecutarse sólo en un programa.
- Ⓜ ..... Puede ejecutarse sólo manualmente.
- Ⓐ ..... Puede ejecutarse tanto manualmente como en un programa.
- Ⓕ ..... Función que puede ejecutarse tanto manualmente como en un programa.

< Ejemplo > DATA [dato] [,[dato]]*

Como todos los "datos" están entre corchetes, los mismos pueden omitirse. Es más, los "datos" pueden repetirse ya que se encuentra un asterisco [ ]* a la derecha del último [dato]. Si se omitiera el primer "dato", la sentencia quedaría así: "DATA, dato, dato, ...".

GOTO { **Número de línea** / **# Número de programa** }

Según esta estructura, esta sentencia podría escribirse de los dos siguientes modos.
(1) GOTO número de línea
(2) GOTO # número de programa

NEW [ALL] Ⓜ

Función

Borra programas y variables.

Parámetros

Cuando se especifica ALL, se borran todas las variables y todos los programas que se encuentren en las áreas de programa P0 hasta P9.

Explicación

(1) Si no se especifica ALL, se borra el programa almacenado en el área de programa en curso y las variables quedan intactas.
(2) Siempre que se especifique ALL, se borran tanto los programas almacenados en todas las áreas de programa, como la totalidad de las variables. Se cancela toda especificación por medio de DEFM y se lleva el número de variables al valor inicial de 26.
(3) No puede ejecutarse después de haberse especificado una clave.
(4) No puede usarse dentro de un programa.
(5) Puede ejecutarse sólo en el modo WRT.
* NEW ALL puede abreviarse del siguiente modo: NEW A.

Ejemplo SHIFT 1 NEW EXE

RUN [Línea de ejecución inicial] (Número de línea) Ⓜ

Función

Ejecución de programas.

Parámetros

Puede especificarse la línea a partir de la cual se desea ejecutar el programa.

Explicación

(1) Ejecuta programas a partir de la línea especificada o desde el comienzo cuando no se especifica número de línea alguno.
(2) Cuando el número de línea especificado no existe en el programa, la ejecución comienza desde la línea que le sigue a la especificada.
(3) Las variables quedan intactas.

Ejemplo

```
10 PRINT "LINE 10"
20 PRINT "LINE 20"
30 END
```

```
RUN EXE
RUN 20 EXE
```

```
LINE 10
LINE 20
```


LIST [{Número de línea / ALL}] Ⓜ

Función

Visualiza en la pantalla el listado de un programa.

Parámetros

Número de línea: De la primera línea que desea visualizarse.
ALL: Hace que se visualice en la pantalla los listados de todos los programas almacenados en las áreas de programa P0 hasta P9.

Explicación

I. En el modo RUN

(1) Visualiza en la pantalla cada línea que compone el programa, a partir de la especificada o desde la primera en caso de omitirse el número de línea.
(2) Las líneas aparecen una por una en la pantalla. Para interrumpir momentáneamente el avance de las línea en la pantalla, pulse la tecla [STOP]. La pulsación de la tecla EXE hace que aparezca en la pantalla la línea que sigue.

II. En el modo WRT

(1) Visualiza el listado de un programa, ya sea desde el número de línea especificado o desde el comienzo cuando no se especifica número de línea alguno.
(2) En este modo, cada línea se visualiza con fines de corrección. En caso de que la línea visualizada no requiera corrección alguna, podrá avanzar a la siguiente pulsando la tecla EXE. Para hacer que se visualice la línea anterior a la presentada, pulse EXE mientras presiona [SHIFT].

- Siempre que se especifique ALL, se visualizan los programas almacenados en P0 hasta P9, en este orden. En este caso, aún cuando se está en el modo WRT no se puede llevar a cabo corrección alguna.
- Este mando no puede usarse después de haberse especificado una clave.

* LIST ALL puede abreviarse del siguiente modo: LIST A.

Ejemplo

```
LIST EXE
LIST 30 EXE
```

PASS "Clave" / Secuencia de caracteres Ⓜ

Función

Especifica o cancela una clave de programa.

Parámetros

Clave: Puede tener de 1 hasta 8 caracteres.

Explicación

(1) Si este mando se ejecuta sin haber especificado un clave previamente, la misma se especifica para todas las áreas de programa.
(2) Si este mando se ejecuta con una clave de programa igual a la clave en curso, la misma se cancela. En caso de que la clave especificada sea diferente a la clave en curso, aparece un ERR8.
(3) En las claves de programa pueden utilizarse 1 hasta 8 caracteres alfabéticos, numéricos o símbolos especiales. El carácter (") no puede utilizarse.
(4) Los mandos LIST, LIST ALL, LIST #, NEW, NEW ALL y NEW # no pueden usarse mientras está especificada una clave de programa. Por otro lado, no se pueden hacer correcciones (modo WRT); en caso de intentarse correcciones, aparece un error (ERR8).
(5) No puede usarse en programas.
(6) Las claves se mantienen aún si se apaga la unidad.

Precaución

Cuando no recuerde la clave en curso, pulse el botón ALL RESET que se encuentra por detrás de la computadora para borrar todos los programas y la memoria.

Ejemplo PASS "CASIO" EXE

CLEAR Ⓐ

Función

Borra todas las variables.

Explicación

(1) Lleva todas las variables, ya sean numéricas 0 de caracteres, a su estado inicial.
(2) Puede usarse tanto manualmente como incorporado en un programa.
(3) No puede ejecutarse dentro de un lazo FOR-NEXT (ver página 123) ya que se borran también las variables de control de dicho lazo, provocándose un error en la ejecución de la sentencia NEXT.

* El mando CLEAR tiene la misma función que VAC.

END Ⓟ

Función

Dar por finalizado un programa.

Explicación

Da por finalizado un programa y hace que la computadora vuelva a la condición previa a la ejecución.

STOP Ⓟ

Función

Interrupción momentánea de la ejecución de un programa.

Explicación

(1) Interrumpe momentáneamente la ejecución de un programa y visualiza "STOP" en la pantalla.
(2) La pulsación de EXE permite reanudar la ejecución.
(3) Si se pulsa la tecla STOP un vez que se haya interrumpido momentáneamente el programa por medio de una sentencia STOP, aparecen en la pantalla el número del área de programa y el número de línea.
(4) Se pueden hacer cálculos manuales durante la interrupción momentánea por medio de STOP.

[LET] { Variable numérica = expresión numérica / Variable de caracteres = expresión de caracteres } Ⓟ

Función

Asigna a la variable de la izquierda el resultado de la expresión a la derecha.

Explicación

(1) Las expresiones numéricas sólo pueden asignarse a variables del mismo tipo. Lo mismo sucede con las expresiones de caracteres, sólo pueden asignarse a variables de caracteres.
(2) LET puede omitirse.

Ejemplo

```
10 LET X=12
20 LET Y=X↑2+2*X-1
30 PRINT Y
40 A$="CASIO"
50 B$="PB&FX"
60 PRINT A$;B$
70 END
```

REM Comentario (Secuencia de caracteres) Ⓟ

Función

Incorporación de comentarios dentro de un programa, sin afectar su ejecución.

Explicación

(1) Todo lo que se escriba a continuación de REM (dentro de un programa), se considera un comentario y no afecta en absoluto la ejecución del programa.
(2) Cuando se agrega a continuación de una sentencia, en la misma línea, coloque dos puntos (:) antes de REM.

Ejemplo

```
10 INPUT "R",R
20 S=π*R↑2:REM AREA
30 PRINT S
40 END
```


INPUT ["Mensaje" / Secuencia de caracteres ,] variable [, ["Mensaje" / Secuencia de caracteres ,] variable]* Ⓟ

Función

Asignación a una variable de la información entrada desde el teclado.

Parámetros

Mensaje: Secuencia de caracteres
Variable: Numérica o de caracteres.

Explicación

(1) Asignación a una variable especificada de la información entrada desde el teclado.
(2) Cuando se incluye un mensaje, se visualiza el signo "?" a continuación del mismo.
(3) Cuando no se incluye mensaje alguno, aparece sólo el signo "?".
(4) Exige la pulsación de la tecla EXE para finalizar su ejecución y asignar a la variable la información entrada desde el teclado.
(5) Cuando se intenta asignar caracteres a una variable numérica, aparece el mensaje de error (ERR2) y se vuelve a la condición de espera de entrada, con el signo "?" en pantalla, una vez pulsada la tecla AC. Si se entra una expresión numérica, se asigna a la variable especificada el resultado de dicha expresión. En caso de entrarse caracteres, los mismos se asignan a variables de caracteres.
(6) Si se pulsa la tecla EXE sin haber entrado información alguna desde el teclado, no se asigna nada a la variable especificada. En caso de ser ésta numérica, aparece el mensaje de error ERR2.

Ejemplo

```
10 INPUT A
20 INPUT "B$=",B$
30 INPUT "C$=",C$,"D$=",D$
```

KEY$ Ⓟ

Función

Permite la entrada de un carácter desde el teclado.

Explicación

(1) Acepta la entrada de un sólo carácter.
(2) Pueden entrarse caracteres numéricos y alfabéticos, y símbolos especiales.

(3) KEY$ no visualiza el signo "?" ni provoca una condición de espera de entrada. Es por ello que, casi siempre, se combina con la sentencia IF.

* KEY$ puede abreviarse del siguiente modo: KEY

Ejemplo

```
10 PRINT "INPUT<6>";
20 A$=""
30 K$=KEY$
40 IF K$="" THEN 30
50 A$=A$+K$
60 IF LEN(A$)<6 THEN 30
70 PRINT A$
80 END
```

* Se aceptan seis caracteres desde el teclado.

PRINT [Elemento a visualizarse][{ ; / , }[Elemento a visualizarse]]* Ⓟ

Función

Visualiza en la pantalla la información especificada.

Parámetros

Pueden usarse funciones para control de la visualización (CSR), expresiones numéricas y expresiones de caracteres.

Explicación

(1) Visualiza en la pantalla la información deseada. La posición de visualización puede especificarse por medio de la función para control de visualización.
(2) Se pueden visualizar expresiones numéricas y de caracteres.
(3) Siempre que se visualice una expresión numérica, se antepone una posición para el signo de dicho número (+ ó −). El signo positivo (+) no se visualiza en dicho caso.

- **Presentación de caracteres** — ␣␣␣·····␣ (␣␣␣·····␣: Elemento a visualizarse)
- **Presentación de números** — ␣␣␣␣·····␣ (␣: Signo; ␣␣␣·····␣: Elemento a visualizarse)

(4) Cuando se desea visualizar una expresión numérica cuya mantisa tenga más de 10 dígitos, se redondea a partir del undécimo dígito. Cuando la mantisa viene acompañada de un exponente, se visualizan el signo de exponenciación E y el exponente con dos dígitos.

(5) Los signos " , " y " ; " pueden usarse entre los elementos como delimitadores de los mismos. Cuando se utiliza la coma, aparece un elemento por vez, y es necesario pulsar la tecla EXE para hacer que se visualice el siguiente elemento. Cuando se utiliza el punto y coma, los elementos van apareciendo uno a continuación del otro, sin parar.
(6) Cuando no se especifica elemento alguno (o sea que se ejecuta sólo PRINT), se borra la pantalla.
(7) La sentencia SET permite establecer el formato de visualización de los números.

Ejemplo

```
10 PRINT 1/3
20 PRINT "A=";A
30 PRINT "SIN 30",SIN 30
40 PRINT "END";
50 PRINT
60 END
```

CSR Posición de visualización (Expresión numérica) Ⓕ

Función

Especifica la posición en la pantalla donde la sentencia PRINT visualizará el elemento especificado.

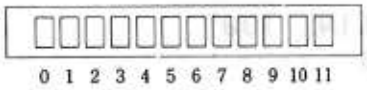
Parámetros

Posición en la pantalla: Expresión numérica. La porción decimal se descarta.

0 ≦ posición < 12

Explicación

(1) Se combina con la sentencia PRINT para especificar la posición de la pantalla a partir de donde se desea visualizar la información.
(2) La primera posición de la izquierda es la número 0.



Ejemplo

```
10 FOR I=0 TO 11
20 PRINT CSRI;"A";CSR 11-I;"B"
30 NEXT I
40 END
```

- Los caracteres A y B se desplazan hacia la izquierda y derecha, respectivamente, por cada pulsación de la tecla EXE .

GOTO { Destino del salto / Número de línea / # Número de area de programa (Desde 0 hasta 9) } Ⓟ

Función

Provoca un salto hacia la posición o destino especificado.

Parámetros

Número de línea: Desde 1 hasta 9999.
Número de área de programa: Desde 0 hasta 9.

Explicación

(1) Provoca un salto hacia el destino especificado.
(2) Cuando se especifica un número de línea, se salta hacia la línea del programa en curso con dicho número. Siempre que la línea especificada no exista, aparece el mensaje de error ERR4.
(3) Cuando se especifica un área de programa, se salta hacia el área especificada y se ejecuta desde el comienzo el programa en ella almacenado.

* Como especificación del destino del salto, se puede utilizar una expresión numérica.

Ejemplo

```
 10 PRINT "START";
 20 GOTO 100
 30 PRINT "LINE 30"
 40 END
100 PRINT "LINE 100"
110 GOTO 30
```


ON Condición de salto (Expresión numérica) GOTO [Destino del salto] [, [Destino del salto]]* Ⓟ

* Destino del salto { Número de línea / # Número de área de programa }

Función

Provoca un salto hacia la posición o destino especificado en base a ciertas condiciones de salto.

Parámetros

Condición de salto: Expresión numérica. La porción decimal se descarta.
Número de línea: Desde 1 hasta 9999.
Número de área de programa: Desde 0 hasta 9.

Explicación

(1) Provoca un salto en base a la parte entera del valor obtenido en la expresión de salto. Cada destino de salto se distribuye por orden ascendente del siguiente modo.

ON A GOTO 100 (A=1), 200 (A=2), 300 (A=3), ...

(2) Cuando el valor de la expresión es menor que 1 o cuando la posición de salto especificada no existe, se ejecuta la sentencia a continuación, sin producirse salto alguno.
(3) En esta sentencia se pueden incluir cuantos destinos permita la longitud de la línea.

Ejemplo

```
 10 INPUT A
 20 ON A GOTO 100,200,300
 30 PRINT "OTHER"
 40 GOTO 10
100 PRINT "LINE 100":GOTO 10
200 PRINT "LINE 200":GOTO 10
300 PRINT "LINE 300":GOTO 10
```

- Cuando se sentra un número del 1 al 3, se salta a los destinos correspondientes; en caso contrario, aparece en la pantalla el mensaje "OTHER".

IF Condición de salto / Expresión condicional THEN {Sentencia [: Sentencia]* / Destino del salto} Ⓟ

* Destino del salto { Número de línea / # Número de área de programa }

Función

Hace que se ejecuten las sentencias a continuación de THEN siempre que se cumpla la condición a continuación de IF. A continuación de THEN puede especificarse el destino de un salto.

Parámetros

Condición de salto: Expresión numerica. La porción decimal se descarta.
Número de línea: Desde 1 hasta 9999.
Número de area de programa: Desde 0 hasta 9.

Explicación

(1) Cuando se cumple la condición de salto, se ejecutan las sentencias a continuación de THEN o se provoca un salto hacia el destino especificado a continuación de THEN.
(2) Cuando no se cumple la condición de salto, se pasa a la ejecución de la línea que sigue.
(3) La condición de salto se juzga mediante los siguientes símbolos: =, ≠, <, >, ≦ y ≧.
   = Lo que está a la derecha es igual a lo que está a la izquierda.
   ≠ Lo que está a la izquierda no es igual a lo que está a la derecha.
   < Lo que está a la derecha es mayor que lo que está a la izquierda.
   > Lo que está a la derecha es menor que lo que está a la izquierda.
   ≦ Lo que está a la derecha es igual o mayor que lo que está a la izquierda.
   ≧ Lo que está a la derecha es igual o menor que lo que está a la izquierda.
(4) Cuando para cierta operación es necesario el cumplimiento de más de una condición, se pueden combinar las sentencias IF-THEN cuantas veces sea necesario.

IF — THEN IF — THEN . . . .

* Cuando después de THEN viene una sentencia, se puede utilizar un punto y coma en lugar del primero.

Ejemplo

```
10 N=6
20 PRINT CSR N;"↑";
30 K$=KEY$
40 IF K$="4"THEN N=N-1:IF N<0THEN N=0
50 IF K$="6"THEN N=N+1:IF N>11THEN N=11
60 PRINT
70 GOTO 20
```

● "↑" se desplaza hacia la izquierda cuando se presiona la tecla [4] y hacia la derecha cuando se presiona la tecla [6].

FOR Variable de control = Valor inicial / Expresión numérica TO Valor final / Expresión numérica Ⓟ
[STEP Incremento / Expresión numérica] NEXT Variable de control

Función

Repite el número de veces especificado por la variable de control las operaciones que se encuentran entre las sentencias FOR y NEXT.

Parámetros

Variable de control: Variable común. No pueden usarse variables de matriz.
Valor inicial: Expresión numérica
Valor final: Expresión numérica
Incremento: Expresión numérica.
Es igual a 1 cuando no se lo especifica.

Explicación

(1) Repite las operaciones que se encuentran entre las sentencias FOR y NEXT las veces especificadas por la variable de control. Esta variable va aumentando desde su valor inicial hasta el final un incremento por cada repetición. La repetición se da por terminada una vez que la variable de control excede el valor final especificado.
(2) Cuando el valor inicial es mayor que el final, la ejecución de las operaciones entre FOR y NEXT se lleva a cabo sólo una vez.
(3) Como incremento puede usarse también un número negativo.
(4) Cada sentencia FOR debe tener su correspondiente sentencia NEXT más adelante en alguna parte del programa.
(5) Es posible incluir un lazo FOR-NEXT dentro de otro, del siguiente modo.

```
10 FOR I=1 TO 10
20 FOR J=11 TO 20
30 PRINT I;":";J
40 NEXT J
50 NEXT I
60 END
```

(6) Se pueden incluir hasta 4 lazos FOR-NEXT dentro de otros.
(7) Cuando el lazo FOR-NEXT finaliza, el valor de la variable de control excede el valor final por el valor del incremento.

(8) Se pueden realizar saltos desde dentro de un lazo FOR-NEXT. Sin embargo, un salto hacia un lazo provoca la aparición un mensaje de error.

GOSUB { Número de línea / Expresión numérica / Número de área de programa / Número de 0 hasta 9 } Ⓟ

Función

Provoca un salto a la subrutina que comienza a partir del número de línea especificado.

Parámetros

Número de línea: Desde 1 hasta 9999.
Número de área de programa: Desde 0 hasta 9.

Explicación

(1) Provoca un salto hacia la subrutina deseada. De esta subrutina se vuelve por medio de la sentencia RETURN.
(2) El salto desde una subrutina hacia otra por medio de GOSUB es lo que se denomina "inclusión de subrutinas". Se permiten inclusiones de hasta 8 niveles.
(3) La sentencia RETURN hace que el programa vuelva a la sentencia que sigue al GOSUB que provocó el salto a la subrutina en curso.
(4) Para volver a la rutina principal, es imposible hacerlo mediante las sentencias GOTO o IF. Sólo puede hacerse por medio de RETURN.
(5) El mensaje de error (ERR4) aparece cuando el número de línea especificado no existe.

* Para la especificación del destino del salto, sea un número de línea o de área de programa, se pueden usar también expresiones numéricas.

Ejemplo

```
 10 PRINT "MAIN 10"
 20 GOSUB 100
 30 PRINT "MAIN 30"
 40 END
100 PRINT "SUB 100"
110 GOSUB 200
120 RETURN
200 PRINT "SUB 200"
210 RETURN
```

RETURN Ⓟ

Función

Hace que el programa regrese de una subrutina a la rutina principal.

Explicación

Hace que el programa regrese a la sentencia que sigue al GOSUB que provocó el salto a la subrutina en curso.

ON Condición de salto / Expresión numérica GOSUB [Destino del salto] [, [Destino del salto]]* Ⓟ

* Destino del salto { Número de línea / # Número de área de programa }

Función

Determina el destino del salto en base a ciertas condiciones.

Parámetros

Condición de salto: Expresión numérica. Se descarta toda porción decimal.
Número de línea: Desde 1 hasta 9999.
Número de área de programa: Desde 0 hasta 9.

Explicación

(1) Determina a qué subrutina saltar en base al valor de una expresión condicional. Los destinos de cada subrutina se colocan en orden, del siguiente modo.

ON B GOSUB 1000, 2000, 3000·····
B=1 B=2 B=3

(2) Cuando el valor de la expresión es menor que 1 o si el destino especificado no existe, se ejecuta la sentencia a continuación, sin provocarse salto alguno.
(3) Se pueden especificar cuantos destinos de salto como la longitud de la línea lo permita.

Ejemplo

```
 10 INPUT A
 20 ON A GOSUB 100,200,300
 30 GOTO 10
100 PRINT "SUB 100":RETURN
200 PRINT "SUB 200":RETURN
300 PRINT "SUB 300":RETURN
```

- Cuando se entre un número del 1 al 3, se provoca un salto hacia la subrutina correspondiente.

DATA [Dato] [, [Dato]]* Ⓟ
Constante Constante

Función

Para incorporar información dentro de un programa.

Parámetros

Dato: Constantes de caracteres o numéricas.

Explicación

(1) Se utiliza para incorporar datos que serán asignados a variables por medio de la sentencia READ.
(2) Puede incorporarse más de un dato por línea. Deben ir separados por el signo " , ".
(3) La presencia de una sentencia DATA no afecta la ejecución del programa si los datos en ella incorporados no se asignan a variables por medio de la sentencia READ.
(4) Cuando una constante de caracteres incluye una coma, póngase la primera entre comillas.

DATA ABC, DEF, "GHI,JKL",·····

(5) Cuando se omite un dato, se lo considera una secuencia de caracteres nula. (o sea de 0 caracteres).

DATA A, ,B → DATA A,"",B
DATA , → DATA "",""
DATA → DATA ""

READ Variable [, [variable]]* Ⓟ

Función

Asigna a la variable especificado la información contenida en la sentencia DATA.

Parámetros

Variable: Numérica o de caracteres. Puede ser de matriz.

Explicación

(1) Asigna a la variable especificada la información que lee secuencialmente de la sentencia DATA.
(2) A variables numéricas puede asignarse sólo información numérica.
(3) Las sentencia DATA se leen por orden ascendente de su número de línea y por orden de aparición dentro de una misma línea.
(4) Una sentencia READ asigna a la variable especificada un dato por vez.
(5) Los datos se van asignando secuencialmente a la o las variables especificadas por cada ejecución de la sentencia READ.
(6) La sentencia RESTORE permite alterar el orden de lectura de los datos.
(7) Si al llegarse al último dato en la sentencia DATA del programa se intentara asignar un datos más a cierta variable por medio de READ, aparecería un mensaje de error (ERR4).
(8) Los espacios al comienzo de los datos se saltan.

Ejemplo

```
10 DATA 1,2,3
20 READ A,B
30 PRINT A;B
40 DATA 4,5
50 READ C,D,E
60 PRINT C;D;E
70 END
```

- Lee los datos en la sentencia DATA por orden y los visualiza en la pantalla.

RESTORE [Número de línea / Expresión numérica] Ⓟ

Función

Especifica la ubicación del siguiente dato a asignarse a una variable por medio de READ.

Parámetros

Número de línea: Expresión numérica. Se descarta toda porción decimal.

1 ≦ número de línea ≦ 9999

Explicación

(1) Especifica el número de línea de una sentencia DATA de la cual READ tomará información para asignarla a la variable especificada.
(2) Cuando se omite el número de línea, se cancela la especificación del dato. Luego de ello, la siguiente ejecución de una sentencia READ hará que se comience desde el primer dato de la primera sentencia READ en dicho programa.
(3) Cuando se especifica el número de línea, la siguiente ejecución de READ hará que se asigne a la variable especificada el primer dato de la sentencia DATA que se encuentra en el número de línea especificado.
(4) Siempre que el número de línea especificado no exista, o no haya ninguna sentencia DATA en el mismo, aparece el mensaje de error (ERR4).

Ejemplo

```
10 DATA 1,2,3
20 DATA 4,5
30 READ A,B,C,D,E
40 RESTORE 10
50 READ F,G
60 RESTORE 20
70 READ H,I
80 PRINT A;B;C;D;E;F;G;H;I
90 END
```


BEEP [{ 0 / 1 }] Ⓐ

Función

Genera un tono de corta duración.

Parámetros

0: Agudo
1: Grave

Explicación

(1) Genera un tono grave o agudo.
(2) Puede también usarse manualmente.

Ejemplo

```
10 $="ABCDEFGHIJKLMNOPQRSTUVWXYZ":N=0
20 FOR I=1 TO 10
30 A$=MID$(RAN#*26+1,1)
40 PRINT CSR4;"<":A$;">";
50 FOR J=1 TO 30
60 K$=KEY$:IF K$<>""THEN80
70 NEXT J
80 IF K$=A$ THEN BEEP 1:N=N+1:GOTO 100
90 BEEP 0
100 PRINT:NEXT I
110 PRINT N;
120 IF N>10 THEN END
130 FOR I=1 TO 10
140 BEEP 0:BEEP 1
150 NEXT I
```

- Pulse la tecla alfabética que corresponda al carácter que aparece en la pantalla.

DEFM [Ampliación]

Expresión numérica Ⓐ

Función

Ampliación de la memoria para variables.

Parámetros

Ampliación: Expresión numérica. Se descarta toda parte decimal. Puede omitirse.

0 ≦ Ampliación < 69

Explicación

(1) Amplía la capacidad de memoria destinada para las variables.
(2) Se puede especificar arbitrariamente según el número de pasos de programa que resten.
(3) Cada variable equivale a 8 pasos de programa.
(4) Cuando se omite la ampliación, sólo se visualiza el número de variables disponibles en el momento.
(5) Puede ejecutarse tanto manualmente como desde un programa. En el primer caso, se visualiza en la pantalla el número total de variables (las agregadas + las 26 originales). No ocurre lo mismo cuando se ejecuta desde un programa.
(6) El mensaje de error ERR1 aparece siempre que se intente agregar más variables que el número de pasos de programa disponibles.
(7) Para cancelar toda expansión previa y volver a las 26 variables originales, use la sentencia DEFM 0.

Ejemplo

```
DEFM 10 [EXE]    ***VAR:36
DEFM [EXE]       ***VAR:36
```

```
10 DEFM 10
20 FOR I=1 TO 10
30 INPUT Z(I)
40 NEXT I
   ⋮
```


MODE

Expresión numérica Ⓟ

Función

Establece el estado de funcionamiento de la computadora.

Parámetros

Expresión numérica: Se descarta toda porción decimal.

4 ≦ expresión numérica < 9

Explicación

(1) Sirve para especificar la unidad angular y establecer o cancelar el modo de impresión.
(2) Sus valores son los siguientes.

MODE4 ····· Grados como unidad angular.
MODE5 ····· Radianes como unidad angular.
MODE6 ····· Gradientes como unidad angular.
MODE7 ····· Visualiza "PRT" y establece el modo de impresión.
MODE8 ····· Libera el modo de impresión.

(3) Cumple la misma función que la tecla [MODE] . Sólo que este mando no permite entrar en los modos RUN y WRT. Es más, su entrada es posible sólo por medio de las teclas [M][O][D][E] , y no mediante la tecla [MODE] .

Ejemplo

```
10 MODE 4
20 A=SIN 30
30 MODE 7
40 PRINT A
50 MODE 8
60 END
```

SET { Fn / En / N } Ⓐ

★ *n* es un entero desde 0 hasta 9.

Función

Especifica el formato de visualización e impresión de la información numérica.

Parámetros

F*n*: Especifica el número de posiciones decimales.
E*n*: Especifica el número de dígitos significativos.
N: Cancela toda especificación previa.

Explicación

(1) Especifica el número de posiciones decimales y de dígitos significativos.
(2) El número de posiciones decimales (F*n*), puede ser desde 0 hasta 9.
(3) En la especificación del número de dígitos significativos (E*n*), se puede usar un valor desde 0 hasta 9. Por otro lado, "SET E0" especifica 10 dígitos.
(4) "SET N" cancela ambas especificaciones.
(5) Puede ejecutarse tanto manualmente como desde un programa.

Ejemplo

```
10 INPUT N
20 SET F5:PRINT N
30 SET E5:PRINT N
40 SET N:GOTO 10
```


FUNCIONES DE CARACTERES

LEN (Variable de caracteres común) Ⓕ

Función

Determina la cantidad de caracteres almacenados en una variable de caracteres común.

Parámetros

Variable de caracteres común: No pueden usarse variables de matriz.

Explicación

(1) Cuenta el número de caracteres contenidos en una variable de caracteres común.
(2) Sólo pueden usarse variables de caracteres comunes (A$, Y$, etc.) y no variables de matriz (B$ (3), etc.).

Ejemplo

```
10 INPUT A$
20 PRINT LEN(A$)
30 GOTO 10
```

MID$ (Posición [, Número de caracteres]) Ⓕ

Posición: Expresión numérica — Número de caracteres: Expresión numérica

Función

Toma de la variable exclusiva de caracteres el número especificado de caracteres a partir de la posición especificada.

Parámetros

Posición: Expresión numérica. Se descarta toda parte decimal.

1 ≦ posición < 101

Número de caracteres: Expresión numérica. Se descarta toda parte decimal.

1 ≦ número de caracteres < 101

Cuando se omite, se toman todos los caracteres a partir de la posición especificada.

Explicación

(1) Toma de la variable exclusiva de caracteres ($) el número de caracteres a partir de la posición especificados.
(2) No se toma nada cuando la posición especificada excede el número de caracteres de la secuencia.
(3) Cuando el número de caracteres a partir de la posición especificada es menor que el número de caracteres especificado, se toman todos los caracteres hasta el final.

* MID$ puede abreviarse del siguiente modo: MID.

Ejemplo

```
10 $="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
20 INPUT M,N
30 PRINT MID$(M,N)
40 END
```

VAL (Variable de caracteres común) Ⓕ

Función

Convierte en un valor numérico una variable de caracteres que contenga una serie de caracteres numéricos.

Parámetros

Variable de caracteres común: No pueden usarse variables de matriz.

Explicación

(1) Convierte en un valor numérico una variable de caracteres que contenga una serie de caracteres numéricos.
(2) Cuando la variable de caracteres incluye signos como ser +, −, •, E ó E⁻, se convierte en un valor numérico como está.

Cuando A$ = "−12.3", VAL(A$) → −12.3

(3) El mensaje de error (ERR4) aparece siempre que el contenido de la variable comience con un carácter que no sea numérico o que no sea un signo matemático.

Cuando A$ = "A45", VAL(A$) → − error (ERR2)

(4) Cuando en medio de la variable se encuentra un carácter que no sea numérico, esta función hace la conversión hasta el carácter anterior al mismo.

Cuando A$ = "78A9", VAL(A$) → 78

Ejemplo

```
10 INPUT A$
20 PRINT VAL(A$)
30 END
```


STR$ (Expresión numérica) Ⓕ

Función

Convierte el valor de una expresión numérica en una secuencia de caracteres.

Parámetros

Expresión numérica: Valores numéricos, expresiones de cálculo y variables numéricas comunes o de matriz.

Explicación

(1) Convierte el valor de una expresión numérica en una secuencia de caracteres.
(2) Cuando la expresión numérica es una expresión de cálculo, lo que se convierte es el resultado de dicha expresión.
(3) Cuando la expresión numérica es positiva, se suprime el dígito correspondiente al signo y se convierte solamente la parte numérica.

Ejemplo

```
10 PRINT STR$(123)
20 PRINT STR$(45+78)
30 A=963
40 PRINT STR$(A)
50 END
```

FUNCIONES NUMERICAS

SIN Argumento (Expresión numérica) COS Argumento (Expresión numérica) (F)

TAN Argumento (Expresión numérica)

Función

Obtiene el valor de la función trigonométrica correspondiente para el argumento dado.

Parámetros

Argumento: Expresión numérica
$-1440° <$ argumento $< 1440°$ (grados)
$-8\pi <$ argumento $< 8\pi$ (radianes)
$-1600 <$ argumento < 1600 (gradientes)
Sin embargo, para TAN, no se incluyen "Argumento = $(2n-1) * 1$ ángulo recto ".

1 ángulo recto $= 90° = \frac{\pi}{2}$ rad $= 100$ grad.

Explicación

(1) Obtiene el valor de la función trigonométrica correspondiente para el argumento dado.
(2) El valor depende de la unidad angular en curso (especificada por medio de la tecla MODE o el mando MODE).

ASN Argumento (Expresión numérica) ACS Argumento (Expresión numérica) (F)

ATN Argumento (Expresión numérica)

Función

Funciones trigonométricas inversas que obtienen el ángulo correspondiente al argumento dado.

Parámetros

Argumento: Expresión numérica
Para ASN, ACS, $-1 \leqq$ argumento $\leqq 1$.

Explicación

(1) Funciones trigonométricas inversas que obtienen el ángulo correspondiente al argumento dado.
(2) El valor depende de la unidad angular en curso (especificada mediante la tecla MODE o el mando MODE).
(3) Los valores de estas funciones se dan dentro de los siguientes límites.

$-90° \leqq$ ASN X $\leqq 90°$
$0° \leqq$ ACS X $\leqq 180°$
$-90° \leqq$ ATN X $\leqq 90°$

LOG Argumento (Expresión numérica) LN Argumento (Expresión numérica) (F)

Función

Da el valor de una función logarítmica.

Parámetros

Argumento: Expresión numérica.
0 < argumento

Explicación

Da el valor de una función logarítmica.

- **LOG** **Logaritmo común** $\log_{10} x$, $\log x$
- **LN** **Logaritmo natural** $\log_e x$, $\ln x$

EXP

Argumento
Expresión numérica

Ⓕ

Función

Da el valor de una función exponencial.

Parámetros

Argumento: Expresión numérica.

$-227 \leqq$ argumento $\leqq 230$

Explicación

Da el valor de una función exponencial.

EXP e^x

SQR

Argumento
Expresión numérica

Ⓕ

Función

Da la raíz cuadrada de un argumento.

Parámetros

Argumento: Expresión numérica.

$0 \leqq$ argumento

Explicación

Da la raíz cuadrada de un argumento.

SQR $\sqrt{x}$

ABS

Argumento
Expresión numérica

Ⓕ

Función

Da el valor absoluto de un argumento.

Parámetros

Argumento: Expresión numérica.

Explicación

Da el valor absoluto de un argumento.

ABS $|x|$

SGN

Argumento
Expresión numérica

Ⓕ

Función

Da un valor correspondiente al signo de un argumento.

Parámetros

Argumento: Expresión numérica.

Explicación

Da un valor correspondiente al signo de un argumento.
Cuando el argumento es positivo: 1
Cuando el argumento es igual a 0: 0
Cuando el argumento es negativo: −1

INT

Argumento
Expresión numérica

Ⓕ

Función

Da el entero máximo que no exceda el argumento.

Parámetros

Argumento: Expresión numérica.

Explicación

Da el entero máximo que no exceda el argumento.

INT 12.56 → 12
INT −78.1 → −79

FRAC (Argumento: Expresión numérica) Ⓕ

Función

Da la parte decimal del argumento.

Parámetros

Argumento: Expresión numérica.

Da la parte decimal del argumento. El signo queda igual que el del argumento mismo.

RND (Argumento: Expresión numérica, Posición: Expresión numérica) Ⓕ

Función

Da el valor de un argumento redondeado al dígito especificado.

Parámetros

Argumento: Expresión numérica.
Posición: Expresión numérica. Se descarta toda porción decimal.

−100 < posición < 100

Explicación

(1) Da el valor de un argumento redondeado al dígito especificado.
(2) Para redondear el argumento a la tercera posición decimal (10^{-3}).

→ RND (x, −3)

Para redondear el argumento a las centenas (10^{2}).

→ RND (x, 2)

RAN # Ⓕ

Función

Da un número aleatorio entre 0 y 1.

Explicación

(1) Da un número aleatorio entre 0 y 1.

0 < número aleatorio < 1

(2) El número aleatorio dado tiene 10 dígitos.

Ejemplo

Para obtener un número aleatorio con 1 dígito de 0 a 9.

INT (RAN# * 10)

Para obtener un número aleatorio con 1 dígito de 1 a 5.

INT (RAN# * 5) + 1

Para obtener un número aleatorio con 2 dígitos de 10 a 99.

INT (RAN# * 90) + 10

DEG (Grados [, Minutos [, Segundos]]) Ⓕ

Función

Convierte números sexagesimales en decimales.

Parámetros

Grados: Expresión numérica.
Minutos: Expresión numérica.
Segundos: Expresión numérica.

DEG (grados, minutos, segundos) | < 10^{100}

Explicación

Convierte en decimal números sexagesimales expresados en grados, minutos y segundos.

Ejemplo

```
DEG(12,34,56) EXE          12.58222222

10 INPUT A,B,C
20 PRINT DEG(A,B,C)
30 END
```

DMS$ Argumento (Expresión numérica) Ⓕ

Función

Convierte números decimales en sexagesimales.

Parámetros

Argumento: Expresión numérica.

| **expresión numérica** | $< 10^{100}$

Explicación

(1) Convierte números decimales en sexagesimales.
(2) El resultado de la conversión se obtiene en la forma de una secuencia de caracteres.

Ejemplo

```
DMS$(45.678) EXE          45°40'40.8

10 INPUT A
20 $=DMS$(A)
30 PRINT$
40 END
```


MANDOS PARA EL BANCO DE DATOS

NEW # Ⓜ

Función

Borra los datos del Banco de Datos.

Explicación

(1) Borra todos los datos.
(2) No puede ejecutarse después de haber especificado una clave de programa.
(3) Puede ejecutarse sólo en el modo WRT.

Ejemplo

```
MODE 1
NEW # EXE
MODE 2
```

LIST # Ⓜ

Función

Lleva a la pantalla todos los datos del Banco de Datos.

Explicación

(1) Visualiza en la pantalla los datos almacenados, uno por uno.
(2) Cada dato aparece en la pantalla con su número correspondiente.
(3) Para mantener la visualización de determinada información, presione la tecla [STOP] al aparecer la misma en la pantalla. Para reanudar el avance de la información por la pantalla, pulse la tecla [EXE] .
(4) No puede utilizarse después de haber especificado una clave de programa.
(5) No puede ejecutarse en el modo de entrada para la función del Banco de Datos (MODE 9).

Ejemplo `LIST# EXE`

READ# Variable name [, Variable name]* Ⓟ

Función

Asigna a las variables especificadas información para el Banco de Datos.

Parámetros

Variable: Puede ser numérica o de caracteres.
Pueden usarse también variables de matriz.

Explicación

(1) Asigna secuencialmente a las variables especificadas los datos grabados en cinta de cassette.
(2) A las variables numéricas se les puede asignar solamente información numérica. El mensaje de error ERR2 aparece siempre que se intente asignar datos de caracteres a este tipo de variables.
(3) Por cada ejecución de la sentencia READ#, se asigna un dato.
(4) Las variables deben escribirse separadas por una coma. Cada dato se asigna ordenadamente por orden a aparición.

Ejemplo)

DATA
Nº. 1 A,X,Y
Nº. 2 B,Z
Nº. 3 C

⬇

Orden de asignación
A→X→Y→B→Z→C

(5) El mensaje de error ERR4 aparece siempre que se intente asignar cierta información que en realidad no existe.
(6) La sentencia RESTORE# permite modificar el orden de asignación de la información (remitase a la página 145 para mayores detalles).
(7) Todo espacio al comienzo de un dato no se tiene en cuenta en la asignación.
(8) La información de caracteres debe escribirse entre comillas.

Ejemplo

〈Dato〉
```
Nº. 1  1,2,3
Nº. 2  4,5,6
Nº. 3  7,8,9
Nº. 4  10,
```

〈Programa〉
```
10 A=0
20 READ#$
30 IF $="" THEN 60
40 A=A+VAL($)
50 GOTO 20
60 PRINT "Σx=";A
70 END
```

● Asigna datos numéricos para obtener la suma de los mismos.

RESTORE# ["Secuencia de caracteres a buscarse"/Expresión de caracteres [,{0/1}]] Ⓟ [,{Número de línea / Número de área de programa}]]]

Función

Modifica el orden de asignación de la información para la sentencia READ# localizando para ello la secuencia de caracteres especificada.

Parámetros

Secuencia de caracteres a buscarse: Expresión de caracteres.
Debe estar entre comillas.

Número de línea: Expresión numérica.

0 < número de línea < 10000

Número de área de programa: Expresión numérica.

0 ≦ Nº de área de programa < 10

Explicación

(1) Busca la información especificada y determina el orden de asignación de información para la siguiente sentencia READ#.
(2) La relación entre los parámetros y la información que desea localizarse es la siguiente.
① RESTORE#
Siempre que se omitan los parámetros, la asignación por medio de la sentencia READ# se lleva a cabo por orden de aparición de los datos, a partir del primero de ellos.
② RESTORE# "secuencia de caracteres a buscarse"
Localiza la secuencia de caracteres especificada, para que la siguiente sentencia READ# la asigne a la variable deseada.
③ RESTORE# "secuencia de caracteres a buscarse", {0/1}

Cuando se especifica 0, cumple la misma función que ②.
Cuando se especifica un 1, la siguiente sentencia READ# asigna el primer dato que se encuentra en la línea donde está la secuencia de caracteres a buscarse.

④ RESTORE# "secuencia de caracteres a buscarse", [{0/1}],
{número de línea / Nº de área de programa}

En este caso, siempre que la secuencia de caracteres especificada no exista, hace que el programa salte a la línea o área de programa especificado.

* En ② y ③, el mensaje de error ERR4 aparece en pantalla siempre que la información que desea localizarse no exista.
* El mismo mensaje de error ERR4 aparece siempre que en ④ el número de línea o de área de programa especificado no exista.

Ejemplo

```
〈Datos〉
№1 FOSTER,347-4811,NEW YORK
№2 SMITH,045-211-0821,CHICAGO
№3 JONES,06-314-2681,SAN FRANCISCO
№4 BROWN,075-351-1161,LOS ANGELES

 10 RESTORE#                    } Se visualiza la información almacenada al comienzo.
 20 READ# $
 30 PRINT $
 40 RESTORE#"S"                 } Se visualizan aquellos datos que comienzan con la letra S.
 50 READ# $
 60 PRINT $
 70 RESTORE#"LO",1              } Busca los datos cuyas dos letras iniciales son LO, y presenta el primer dato en la línea que incluye los datos.
 80 READ# $
 90 PRINT $
100 RESTORE#"AA",1,200          } Salto a la línea 200, al verse que no hay dato alguno que comience con dos letras A.
110 READ# $
120 PRINT $
130 END
200 PRINT"END"
210 END

RUN EXE
   EXE
   EXE
   EXE
```

FOSTER
SMITH
BROWN
END

WRITE# [Dato [, Dato] *] (P)

expresión expresión

Función

Almacena o borra,información del Banco de Datos.

Parámetros

Dato: Expresión numérica o de caracteres. Las de caracteres deben estar entre comillas.

Explicación

(1) Almacena información en el área del registro especificado por la sentencia RESTORE#.
(2) La información se almacena sin tenerse en cuenta si hay o no algún dato en el área de registro especificada.
(3) Cuando no se especifica información alguna, se borra la información que pueda estar almacenada en el área de registro especificada.
(4) Cuando haya más de un dato, los mismos pueden almacenarse en el mismo registro siempre que se los separe por una coma.
(5) Sólo puede almacenarse un dato por sentencia WRITE#.

Ejemplo

```
 10 REM WRITE
 20 RESTORE#                    } Se almacena información nueva.
 30 WRITE#"A,B,C"
 40 RESTORE#
 50 FOR I=1 TO 3
 60 READ# $:PRINT $;
 70 NEXT I
 80 PRINT" "
 90 REM CHANGE
100 RESTORE#                    } Se vuelve a almacenar la misma información.
110 FOR I=1 TO 3
120 WRITE# STR$(I)
130 NEXT I
140 RESTORE#
150 FOR I=1 TO 3
```

```
160 READ# $: PRINT $;
170 NEXT I
180 PRINT " "
190 REM CLEAR
200 RESTORE#
210 WRITE#
220 RESTORE#
230 READ# $
```

Borrado de la información.

Operación	Pantalla
RUN EXE	ABC
EXE	123
EXE	ERR4 P0-230

↓

Insuficiencia de información ya que se borraron los datos.

CAPITULO 5

PROGRAMOTECA

1. Conversión de números decimales en números hexadecimales
2. Cálculos para préstamos (Cuotas mensuales iguales)
3. Ejercicios aritméticos
4. Análisis de regresión cuadrática
5. Juego de reordenamiento

1. CONVERSION DE NUMEROS DECIMALES EN NUMEROS HEXADECIMALES

Este programa sirve para convertir números decimales en hexadecimales y viceversa. Es muy útil para determinar direcciones de memoria especificadas en uno y otro sistema numérico.

Ejemplo)

1. ¿Equivalente decimal del hexadecimal 14AF?
2. ¿Equivalente decimal del hexadecimal A540?
3. ¿Equivalente hexadecimal del decimal 4582?
4. ¿Equivalente hexadecimal del decimal 18657030?

Operación	Pantalla
• Inicie primeramente el programa.	
RUN [EXE]	10÷1-0-2÷16
• Al hacerlo, aparece en la pantalla lo que se denomina "menú". El mismo sirve para escoger el tipo de conversión que desea realizarse. Para convertir de hexadecimal a decimal, pulse la tecla [1]. En caso contrario, pulse la tecla [2]. Aquí, la pulsación de la tecla [2] hace que finalice el programa.	
[1]	HEX?
• Entre el primer número hexadecimal.	
14AF [EXE]	5295
• Entre el siguiente número hexadecimal.	
[EXE]	HEX?
A540 [EXE]	42304
• Para hacer que aparezca nuevamente el menú en la pantalla después de obtenido el resultado de la conversión, pulse la tecla [EXE] en lugar de entrar otro número.	
[EXE]	HEX?
[EXE]	10÷1-0-2÷16
• Una vez que se haya visualizado el menú en la pantalla, probemos hacer conversiones de decimal a hexadecimal pulsando la tecla [2].	
[EXE]	DEC?

Operación	Pantalla
• Entre el primer número hexadecimal.	
4582 [EXE]	11E6
• Entre el siguiente número.	
[EXE]	DEC?
18657030 [EXE]	11CAF06
• Para dar por terminado el programa, vuelva al menú y pulse la tecla [2].	
[EXE]	DEC?
[EXE]	10÷1-0-2÷16
[2]	----END----

■ **Lista del programa**

```
10 FOR I=0 TO 5: R
   EAD A$(I): NEXT
    I
20 DATA A,B,C,D,E,
   F
30 PRINT "10÷1-0-2
   ÷16";
40 $= KEY$: IF $="
   " THEN 40
50 BEEP : PRINT
60 ON VAL($) GOTO
   80,220
70 PRINT "----END-
   ---": END
80 INPUT "HEX";$
90 IF $="" THEN 30
100 Q= LEN($):R=0
110 FOR I=0 TO Q-1
120 P$= MID$(Q-I,1)
130 IF P$<"9" THEN
   A= VAL(P$): GOT
   O 180
140 FOR J=0 TO 5
150 IF A$(J)=P$ THE
   N A=J+10: GOTO
    180
160 NEXT J
170 BEEP : GOTO 100
180 R=R+16↑I*A
190 NEXT I
200 BEEP 1: PRINT R
210 GOTO 60
220 INPUT "DEC";$
230 IF $="" THEN 30
240 S= VAL($):P=S:Q
   =0:$=""
250 Q= FRAC(S/16)*1
   6:S=S-Q
260 IF P<16 THEN 29
   0
270 P= INT(P/16):Q=
   Q+1
280 GOTO 260
290 IF Q=0 THEN 350
300 FOR I=Q TO 1 ST
   EP -1
310 P= INT(S/16↑I)
320 Z=P: GOSUB 500
330 S= INT( FRAC(S/
   16↑I)*16↑I+.5)
340 NEXT I
350 Z=0: GOSUB 500
360 BEEP 1: PRINT $
370 GOTO 220
500 IF Z>9 THEN $=$
   +A$(Z-10): RETU
   RN
510 $=$+ STR$(Z): R
   ETURN
   Total: 507 pasos
```

2. CALCULOS PARA PRESTAMOS (CUOTAS MENSUALES IGUALES)

Este programa sirve para calcular las cuotas mensuales, el monto del préstamo y el número de cuotas teniendo como datos la tasa de interés y dos de los tres datos citados.

Datos conocidos	Resultado
• Tasa de interés anual, cuotas mensuales y número de cuotas	→ Monto del préstamo
• Tasa de interés anual, cuota mensual y monto del préstamo	→ Número de cuotas
• Tasa de interés anual, monto del préstamo y número de cuotas	→ Cuota mensual

La tasa de interés anual se entra en porciento.
La cuota mensual y el monto del préstamo se calcula en dólares, mientras que el número de cuotas se da en meses. Los resultados de los cómputos se redondean.

Fórmulas)

P: cuota mensual PV: Monto
i: Tasa de interés mensual (se entra la tasa anual)
n: Número de cuotas

$$P = PV\frac{i}{1-(1+i)^{-n}}$$

$$PV = P\frac{1-(1+i)^{-n}}{i}$$

$$n = -\frac{In\left(1-\frac{i\cdot PV}{P}\right)}{In(1+i)}$$

Ejemplos)

1. ¿Cuál sería el monto de un préstamo a pagarse con 24 cuotas mensuales de 200 dólares, si la tasa de interés anual es del 7,5%?
2. ¿De cuánto serían las cuotas para un préstamo de 5.000 dólares a pagarse en 36 meses si la tasa de interés anual es del 6,5%?
3. ¿Cuántas cuotas de 250 dólares se pagarían para un préstamo de 25.000 dólares si la tasa de interés anual es del 5,8%?

Operación	Pantalla
• Inicie el programa. RUN EXE	Payment?
• Entre la cuota mensual. En nuestro Ejemplo 1, la misma es de 200 dólares. 200 EXE	Loan?
• En este ejemplo, se averigua el monto del préstamo. Pulse entonces la tecla EXE . EXE	Rate?
• Entre la tasa de interés anual. 7.5 EXE	Months?
• Entre el número de cuotas. 24 EXE	Loan 4444 * Monto del préstamo
• Ejecutemos a continuación el Ejemplo 2. RUN EXE	Payment?
EXE	Loan?
• El monto del préstamo es de $5.000. 5000 EXE	Rate?
• La tasa de interés es del 6,5%. 6.5 EXE	Months?
• El número de cuotas es de 36. 36 EXE	Payment 153 *Cuota mensual
• ¿Veamos entonces el Ejemplo 3? RUN EXE	Payment?

- La cuota mensual es de $250.

250 EXE

```
Loan?
```

- El monto del préstamo es de $25.000.

25000 EXE

```
Rate?
```

- La tasa de interés es del 5,8%.

5.8 EXE

```
Months?
```

- Pulse la tecla EXE para obtener el número de cuotas.

EXE

```
Months 137
```

*Número de cuotas mensuales

■ Lista del programa

```
10 CLEAR
20 RESTORE
30 DATA Payment,Lo
   an,Rate,Months
40 FOR I=0 TO 3
50 READ E$(I): PRI
   NT E$(I);
60 INPUT $
70 IF $<>"" THEN A(
   I)= VAL($):Y=Y+
   1
80 NEXT I
90 IF Y<3 THEN BEE
   P 0: GOTO 10
100 K=C/1200
110 FOR I=0 TO 3
120 IF A(I)=0 THEN
   150
130 NEXT I
140 BEEP 0: GOTO 10
150 ON I GOTO 170,1
   40,180
160 B= INT(B*K/(1-(
   1+K)↑-D)+.5): G
   OTO 190
170 B= INT(A*(1-(1+
   K)↑-D)/K+.5): G
   OTO 190
180 D= INT(- LN(1-K
   *B/A)/ LN(1+K)+
   .5)
190 BEEP 1
200 PRINT E$(I);A(I
   )
```

Total: 287 pasos

3. EJERCICIOS ARITMETICOS

Este programa presenta problemas de suma, resta, multiplicación y división. En todos los casos, es posible especificar el número de dígitos (1 ó 2).
Cada juego consiste en diez problemas, por cada uno de los cuales se asigna un máximo de 10 puntos. El puntaje para los cocientes se determina si el mismo es correcto o no, con dos dígitos decimales.

Operación	Pantalla	
RUN EXE	`Push + - * /`	..... Indicación de las teclas usadas para suma, resta, multiplicación y división, respectivamente.
✚	`1 or 2`	..... Elección del número de dígitos.
1	`<1>3+9=?`	..... Problema 1
12 EXE	`<2>2+7=?`	..... Problema 2
9 EXE	`<3>2+6=?`	..... Problema 3
	⋮	
	`<10>6+7=?`	..... Problema 10
13 EXE	`SCORE:100`	..... Puntaje

El programa hace sonar un tono cada vez que la respuesta dada es correcta. Cuando se obtiene el puntaje máximo posible, se escucha una melodía musical.

■ **Lista del programa**

```
10 E=0
20 PRINT "Push + -
   * /";
30 K$= KEY$
40 IF K$="+" THEN
   M=1: GOTO 90
50 IF K$="-" THEN
   M=2: GOTO 90
60 IF K$="*" THEN
   M=3: GOTO 90
70 IF K$="/" THEN
   M=4: GOTO 90
80 GOTO 30
90 BEEP 1
100 PRINT : PRINT "
   1 or 2";
110 K$= KEY$
120 IF K$="1" THEN
   N=9: GOTO 150
130 IF K$="2" THEN
   N=99: GOTO 150
140 GOTO 110
150 BEEP 1
160 FOR I=1 TO 10
170 PRINT
180 A= INT( RAN#*N+
   1)
190 B= INT( RAN#*N+
   1)
200 ON M GOSUB 300,
   400,500,600
210 PRINT "(";STR$
   (I);")";$;: INP
   UT D
220 BEEP 0:D= INT(D
   *100)/100
230 IF C=D THEN E=E
   +1: BEEP 1: BEE
   P 1
240 NEXT I
250 PRINT "SCORE:";
   E*10;
260 IF E=10 THEN FO
   R I=1 TO 10: BE
   EP 1: BEEP 0: N
   EXT I
270 END
300 $= STR$(A)+"+"+
    STR$(B)+"="
310 C=A+B
320 RETURN
400 IF A<B THEN F=A
   :A=B:B=F
410 $= STR$(A)+"-"+
    STR$(B)+"="
420 C=A-B
430 RETURN
500 $= STR$(A)+"x"+
    STR$(B)+"="
510 C=A*B
520 RETURN
600 IF A<B THEN F=A
   :A=B:B=F
610 $= STR$(A)+"÷"+
    STR$(B)+"="
620 C= INT(A/B*100)
   /100
630 RETURN
```

Total: 540 pasos

4. ANALISIS DE REGRESION CUADRATICA

Este programa sirve para estimar el valor de y mediante un análisis de regresión cuadrática de los datos x e y.

$$a=\frac{\Sigma(x^2\cdot y)\cdot\Sigma(x\cdot x)-\Sigma(x\cdot y)\cdot\Sigma(x\cdot x^2)}{\Sigma(x\cdot x)\Sigma(x^2\cdot x^2)-\{\Sigma(x\cdot x^2)\}^2}$$

$$b=\frac{\Sigma(x\cdot y)\Sigma(x^2\cdot x^2)-\Sigma(x^2\cdot y)\Sigma(x\cdot x^2)}{\Sigma(x\cdot x)\cdot\Sigma(x^2\cdot x^2)-\{\Sigma(x\cdot x^2)\}^2}$$

$$c=\frac{\Sigma y_i}{n}-b\frac{\Sigma x_i}{n}-a\frac{\Sigma x_i{}^2}{n}$$

El programa en P0 se utiliza para la entrada de la información. Entre mediante el mismo los datos x e y para la obtención de Σx, Σy, Σx^2, Σxy, Σx^3, Σx^4, y $\Sigma x^2 y$.

El programa en P1 se utiliza para calcular a, b y c de los valores Σx, Σy, Σx^2, Σxy, Σx^3, Σx^4, y $\Sigma x^2 y$ obtenidos mediante el programa en P0. Luego, obtenga la estimación de y correspondiente a la x entrada.

Ejemplo)

	1	2	3	4	5	6	7
x	1	5	8	11	15	18	22
y	21	30	41	54	70	①	②

Las estimaciones de ① y ② se obtienen por medio de una análisis de regresión de los datos arriba ilustrados.

Operación	Pantalla
SHIFT P0	x 1?
1 EXE	y 1?
21 EXE	x 2?
5 EXE	y 2?
Entre los datos que siguen.	⋮
	y 5?
70 EXE	x 6?

Ejecute el programa almacenado en P1 después de entrar todos los datos.

Operación	Pantalla	
SHIFT P1	a= 0.08967	·····a
EXE	b= 2.14288	·····b
EXE	c= 18.23781	·····c
EXE	x=?	
18 EXE	y= 85.86245	·····Estimación de y
EXE	x=?	
22 EXE	y= 108.78103	·····Estimación de y

Usando el programa almacenado en P1 de este modo, se pueden obtener los valores de a, b y c y la estimación de y.
En nuestro ejemplo, se han redondeado los resultados a la quinta posición decimal. Para alterar el número de posiciones decimales (mediante la función 8ET), cambie la línea 90 del programa en P1 según sea necesario.

■ **Lista del programa**

```
P0
 10 CLEAR
 20 PRINT "x";M+1;:
    INPUT X
 30 PRINT "y";M+1;:
    INPUT Y
 40 N=N+X:O=O+Y
 50 P=P+X↑2:R=R+X*Y
 60 A=A+X↑3:B=B+X↑4
    :C=C+X↑2*Y
 70 M=M+1
 80 GOTO 20
```

```
P1
 10 D=P-N↑2/M
 20 E=R-N*O/M
 30 F=A-N*P/M
 40 G=C-P*O/M
 50 H=B-P↑2/M
 60 I=(G*D-E*F)/(D*
    H-F↑2)
 70 J=(E*H-G*F)/(D*
    H-F↑2)
 80 K=O/M-J*N/M-I*P
    /M
 90 SET F5
100 PRINT "a=";I,"b
    =";J,"c=";K
110 INPUT "x=";X
120 Y=I*X↑2+J*X+K
130 PRINT "y=";Y
140 GOTO 110
```

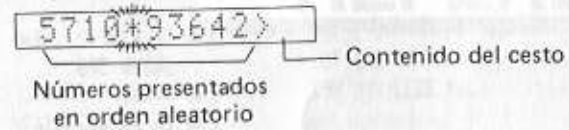
Total 309 pasos

Contenido de las variables	
A	Σx^3
B	Σx^4
C	$x^2 \cdot y$
I	a
J	b
K	c
N	Σx
M	Número de datos
O	Σy
P	Σx^2
R	$\Sigma x \cdot y$

5. JUEGO DE REORDENAMIENTO

Este entretenido juego consiste en ordenar en orden ascendente y en el menor tiempo posible los números del 0 al 9 visualizados en la pantalla en un orden aleatorio.
En el mismo, se utilizan las teclas 4 , 6 , y + . La pulsación de la tecla 4 desplaza el asterisco parpadeante una posición hacia la izquierda, mientras que la tecla 6 hace lo propio en sentido contrario. La pulsación de la tecla + hace que el número sobre el asterisco parpadeante pase al cesto y que el número que se encuentra en el cesto pase a la posición donde se encuentra el asterisco.

Lectura de la pantalla



El puntaje representa el tiempo que lleva reordenar todos los números; por lo tanto, cuanto menos tiempo se tarde mejor. Siempre que juegue este juego por primera vez o después de haber ejecutado otro programa, no olvide entrar CLEAR EXE .

Operación	Pantalla	
CLEAR EXE		
RUN EXE	Hi-Sc:0	... Mejor puntaje hasta el momento
	<<WAIT!>>	... Preparación del juego
	3721*94650>	
+	3721*94650>8	... Lleve el 8 al cesto
6666	3721 946*0>8	... Desplace el asterisco hasta la posición que corresponda.
+	3721 946*0>5	... Cambie el 5 por el 8.
444	3721 *4680>5	... Lleve el asterisco a la posición del 9.
+	3721 *4680>9	... Cambie el 5 por el 9.
	⋮	
+	0123*56789>4	... Lleve el 4 a la posición que le corresponde.
	[0123456789]	... Fin del juego
	SCORE 132	... Puntaje

Para volver a jugar, entre RUN EXE .

*** Habilidad según el puntaje obtenido.**

Puntaje	Evaluación
Menos de 50	Vd. es un genio o ha tenido mucha suerte.
50–69	Sus reflejos son excelentes.
70–99	Nivel promedio. Con un poco de práctica podrá alcanzar un mejor nivel.
100–149	Necesita practicar.
150 o más	Algo le pasa a Vd., Intente nuevamente.

■ **Lista del programa**

```
10 PRINT "Hi-Sc:";
   H;: FOR I=1 TO
   10: BEEP 1: NEX
   T I
20 $=""
30 PRINT : PRINT "
   << WAIT ! >>";
40 A$= STR$( INT(
   RAN#*10))
50 FOR I=1 TO LEN(
   $)
60 IF A$= MID$(I,1
   ) THEN 40
70 NEXT I
80 $=$+A$
90 IF LEN($)<10 TH
   EN 40
100 PRINT
110 X=5:B$=" ":N=0
120 PRINT CSR0;$;"
   ";B$;
130 PRINT CSRX;"*";
140 K$= KEY$
150 IF K$="4" THEN
   BEEP 0:X=X-1: I
   F X<0 THEN X=0
160 IF K$="6" THEN
   BEEP 0:X=X+1: I
   F X>9 THEN X=9
170 IF K$="+" THEN
   GOSUB 500
180 N=N+1
190 IF $*"012345678
   9" THEN 120
200 IF H=0 THEN H=N
210 IF H>N THEN H=N
220 PRINT : PRINT "
   [";$;"]";
230 FOR I=1 TO 10
240 BEEP 0: BEEP 1
250 NEXT I
260 PRINT
270 PRINT "SCORE";N
   ;
280 END
500 BEEP 1
510 IF X=0 THEN C$=
    MID$(1,1):$=B$
   + MID$(2): GOTO
    540
520 C$= MID$(X+1,1)
530 $= MID$(1,X)+B$
   + MID$(X+2)
540 B$=C$
550 RETURN
```

Total: 444 pasos

CAPITULO 6

MATERIAL DE REFERENCIA

6-1. TABLA DE MENSAJES DE ERROR

Código de error	Significado	Causa	Medida correctiva
1	Saturación de la memoria o saturación del nivel de operadores	• El número de pasos es insuficiente. El programa no se puede escribir. • Saturación del nivel de operadores	• Elimine los programas innecesarios o reduzca el número de memorias. • Divida las fórmulas y hágalas más sencillas.
2	Error de sintáxis	• Error en el formato del programa, etc. • Los formatos del lado izquierdo y del derecho difieren en una sentencia de asignación, etc.	• Corrija el error en el programa entrado, etc.
3	Error matemático	• El resultado del cálculo de una expresión numérica se excede de $\pm 1 \times 10^{100}$. • El argumento de la función numérica está fuera de la gama de entrada. • El resultado es infinito o imposible.	• Corrija las fórmulas o los datos de cálculo. • Verifique los datos.
4	Error por falta de definición	• No se designó el número de línea para una sentencia GOTO o GOSUB. • Se ejecutó una sentencia READ o READ# cuando no hay datos que recuperar.	• Designe el número de línea. • Verifique la relación entre las sentencias READ y DATA. Introduzca los datos en la sentencia para asignarlos a la(s) variable(s).
5	Error en el argumento	• El argumento requerido para un mando o una función está fuera gama de entrada.	• Corrija el argumento.
6	Error de variable(s)	• Se intentó usar una memoria que no había sido expandida. • Se intentó usar la misma memoria para una variable numérica y una variable de caracteres al mismo tiempo.	• Expanda adecuadamente la memoria. • No use la misma memoria para una variable numérica y una variable de caracteres al mismo tiempo.
7	Error de inclusión	• Se ejecutó la sentencia RETURN sin haberse ejecutado la subrutina. • Se ejecutó la sentencia NEXT fuera del lazo de FOR. • Los niveles de inclusión de subrutinas se exceden de 8. • Los niveles de inclusión de lazos FOR-NEXT se exceden de 4.	• Elimine las sentencias RETURN o NEXT innecesarias. • Mantenga las subrutinas o los circuitos FOR-NEXT dentro de los niveles adecuados.

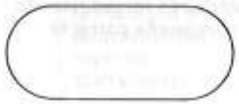
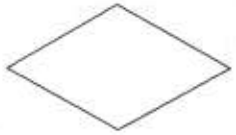
Código de error	Significado	Causa	Medida correctiva
8	Contraseña	• Cuando se ha especificado una contraseña y a) se especifica otra contraseña. b) se ejecuta un mando como LIST o NEW, los cuales no pueden usarse.	• Cancele el error de contraseña introduciendo la contraseña correcta.

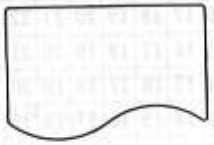
6-2. TABLA DE CODIGOS DE CARACTERES

	espacio	+	−	*	/	↑	⁄	"	#	$	>	≥	=	≤	<	≠
Números	0	1	2	3	4	5	6	7	8	9	.	π	)	(	Ē	E
Letras en mayúscula	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
	Q	R	S	T	U	V	W	X	Y	Z						
Letras en minúscula	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p
	q	r	s	t	u	v	w	x	y	z						
Símbolos	?	,	;	:												
Símbolos gráficos	○	Σ	◦	△	@	×	÷	♠	←	♥	◆	♣	μ	Ω	↓	→
	%	¥	□	[	&	_	'	•	]	■	\					

Los caracteres y símbolos de la tabla están ordenados en secuencia, siendo SPACE (Barra de espacio) el menor y "\" el mayor. ("\" aparece en la pantalla oprimiendo GRAPH P.)

6-3. SIMBOLOS PARA LOS DIAGRAMAS FLUJO

Símbolo	Significado
	Punto de entrada o salida (inicio, cambio de línea, fin, etc.)
	Introducción de datos desde el teclado
	Función de salida de información
	Procesamiento general
	Procesamiento de una subrutina
	Prueba (condición)

Símbolo	Significado
	Salida a un impresor
	Línea de flujo
	Punto de transferencia o de continuación

6-4. TABLA DE VARIABLES DE MATRIZ

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
B	–	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
C	–	–	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
D	–	–	–	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
E	–	–	–	–	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
F	–	–	–	–	–	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
G	–	–	–	–	–	–	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
H	–	–	–	–	–	–	–	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
I	–	–	–	–	–	–	–	–	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
J	–	–	–	–	–	–	–	–	–	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
K	–	–	–	–	–	–	–	–	–	–	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
L	–	–	–	–	–	–	–	–	–	–	–	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
M	–	–	–	–	–	–	–	–	–	–	–	–	0	1	2	3	4	5	6	7	8	9	10	11	12	13
N	–	–	–	–	–	–	–	–	–	–	–	–	–	0	1	2	3	4	5	6	7	8	9	10	11	12
O	–	–	–	–	–	–	–	–	–	–	–	–	–	–	0	1	2	3	4	5	6	7	8	9	10	11
P	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	0	1	2	3	4	5	6	7	8	9	10
Q	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	0	1	2	3	4	5	6	7	8	9
R	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	0	1	2	3	4	5	6	7	8
S	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	0	1	2	3	4	5	6	7
T	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	0	1	2	3	4	5	6
U	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	0	1	2	3	4	5
V	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	0	1	2	3	4
W	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	0	1	2	3
X	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	0	1	2
Y	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	0	1
Z	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	0
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

Esta tabla indica la relación existente entre las variables.

Ejemplo) H(0)~H(9)→H~Q

INDICE DE MANDOS/FUNCIONES

ESPECIFICACIONES

- **Tipo**
  PB-80
- **Funciones de cálculos fundamentales**
  Números negativos; exponentes; suma, resta, multiplicación y división (con función de evaluación de secuencia de prioridad (por lógica algebraica real).
- **Funciones incorporadas**
  Funciones trigonométricas/trigonométricas inversas (unidades angulares – grados/radianes/gradianes), funciones logarítmicas/exponenciales, raíces cuadradas, potencias, conversión a enteros, eliminación de la fracción de enteros, valor absoluto, simbolización, designación del número de dígitos significativos, designación del número de dígitos decimales, números aleatorios, π, conversión decimal ↔ sexagesimal.
- **Mandos**
  INPUT, PRINT, GOTO, ON-GOTO, FOR-NEXT, IF-THEN, GOSUB, ON-GOSUB, RETURN, READ, DATA, RESTORE, STOP, END, REM, LET, BEEP, PASS, RUN, LIST, LIST ALL, MODE, SET, CLEAR, NEW, NEW ALL, DEFM, NEW #, LIST #, READ #, WRITE #, RESTORE #.
- **Funciones de programa** KEY$, CSR, LEN, MID$, VAL, STR$
- **Gama de cálculo**
  De $\pm 1 \times 10^{-99}$ a $\pm 9.999999999 \times 10^{99}$ y 0 (los cálculos internos usan una mantisa de 12 dígitos).
- **Sistema de programa** Sistema incorporado
- **Número de pasos**
  544 pasos máximo (hasta 1.568 cuando se usa el módulo de expansión de la memoria RAM opcional)
- **Capacidad de programas** 10 programas, máximo (de P0 a P9).
- **Número de variables**
  26 originalmente; ampliable hasta 94 (máximo 222 cuando se usa el módulo para expansión de la memoria RAM opcional) y la variable exclusiva de caracteres.
- **Inclusiones** Subrutinas – 8 niveles
  Lazos FOR-NEXT – 4 niveles
  Valores numéricos – 6 niveles
  Operadores – 12 niveles
- **Sistema de presentación en la pantalla**
  Mantisa de 10 dígitos (incluyendo el signo de menos) o mantisa de 8 dígitos (7 dígitos para números negativos) y exponente de 2 dígitos.
- **Pantalla** De matriz de puntos para 12 dígitos (cristal líquido).
- **Componentes principales** C-MOS LVSI y otros.
- **Consumo de energía** 0,03 W máximo.
- **Duración de las pilas (uso continuo)** 150 horas aproximadamente
- **Función de apagado automático**
  La corriente se apaga automáticamente aproximadamente 6 minutos después de la última operación.
- **Temperatura ambiente** De 0°C a 40°C (32°F a 104°F).
- **Dimensiones** 14,5Al. x 146,5An. x 78Pr. mm
- **Peso** 117,5 g incluyendo las baterías

CASIO

SA0701064A Printed in Japan