MS-DOS QBASIC - Versión 1.1 (1993)
REFERENCIAS DEL LENGUAJE

PALABRAS CLAVES Y TEMAS SEGUN TAREAS

Tarea de programación	Palabras clave y temas incluidos
Controlar flujo del programa	DO...LOOP, END, EXIT, FOR...NEXT, IF...THEN...ELSE, GOSUB...RETURN, GOTO, ON...GOSUB, ON...GOTO, SELECT CASE, STOP, SYSTEM
Declarar constantes y variables	CONST, DATA, DIM, ERASE, OPTION BASE, READ, REDIM, REM, RESTORE, SWAP, TYPE...END TYPE Juego de caracteres del Qbasic Tipos de variables
Definir y llamar procedimientos de Basic	CALL, DECLARE, EXIT, FUNCTION, RUN, SHELL, SHARED, STATIC, SUB Línea de comandos
Entrada/salida de dispositivos	CLS, CSRLIN, INKEY$, INP, INPUT, KEY (asignación), LINE INPUT, LOCATE, LPOS, LPRINT, LPRINT USING, OPEN COM, OUT, POS, PRINT, PRINT USING, SPC, SCREEN (función), TAB, VIEW PRINT, WAIT, WIDTH Códigos de exploración de teclado
Presentar imágenes gráficas	CIRCLE, COLOR, GET (gráficos), LINE, PAINT, PALETTE, PCOPY, PMAP, POINT, PRESET, PSET, PUT (gráficos), SCREEN (instrucción), VIEW, WINDOW Atributos y valores de color Modos de pantalla
Comandos del sistema de archivo DOS	CHDIR, KILL, MKDIR, NAME, RMDIR
Entrada/salida de archivos	CLOSE, EOF, FILEATTR, FREEFILE GET (archivos), INPUT, INPUT$, LINE INPUT, LOC, LOCK, LOF, OPEN, PUT (archivos), SEEK (función), SEEK (instrucción), UNLOCK, WRITE
Manejo de la memoria	CLEAR, FRE, PEEK, POKE
Manipulación de cadenas	ASC, CHR$, HEX$, INSTR, LCASE$, LEFT$, LEN, LSET, LTRIM$, MID$ (función), MID$ (instrucción), OCT$, RIGHT$, RSET, RTRIM$, SPACE$, STR$, STRING$, UCASE$, VAL Códigos ASCII
Realizar cálculos matemáticos	ABS, ASC, ATN, CDBL, CINT, CLNG, COS, CSNG, CVDMBF, CVSMBF, EXP, INT, LOG, RANDOMIZE, RND, SGN, SIN, SQR, TAN, TIME$ (función)
Crear desvios para eventos y errores	COM, ERDEV, ERDEV$, ERL, ERR, ERROR, KEY (desviar eventos), ON COM, ON ERROR, ON KEY, ON PEN, ON PLAY, ON STRIG, ON TIMER, PEN, PLAY (desviar eventos), RESUME, RETURN, STRIG, TIMER (función), TIMER (instrucción) Códigos de errores de ejecución
Operadores Booleanos	AND, OR, NOT, XOR, EQV, IMP

Vea Límites del lenguaje para aplicar a cada tarea de programación.

INDICE ALFABETICO DE PALABRAS CLAVES Y TEMAS

PALABRAS CLAVE Y TEMAS DE QBASIC POR ORDEN ALFABETICO

* En el detalle de las palabras clave sigue las siguientes convenciones:

PALABRAS CLAVE	Elementos en mayúsculas son palabras clave de Basic. Estas constituyen una parte requerida de la sintaxis de una instrucción, a menos que estén entre corchetes.
marcadores	Elementos en minúsculas son marcadores para la información que debe suministrar en la instrucción (por ejemplo, archivo$). La sintaxis de QBasic usa sufijos indicando el tipo de dato para los marcadores que deben ser de un tipo específico. Los marcadores que pueden ser de más de un tipo de date no tienen esta clase de sufijo.
[elemento optativo]	Los elementos entre corchetes son optativos.
{opción1 | opción2}	Llaves y una barra vertical indican dos o más opciones de las que debe escoger y utilizar una de las instrucciónes, a menos que las llaves estén entre corchetes.
elemento, elemento...	Puntos suspensivos indican que se pueden usar más elementos de los que preceden los puntos suspensivos en una instrucción que ocupa una sola línea.
Palabra inicial . . . . Palabra final	Una serie de tres puntos en sentido vertical se usa para describir instrucciones que ocupan varias líneas (o instrucciones en bloque). Significa que se pueden usar otras instrucciones entre el principio y final del bloque.

* En el desarrollo de cada palabra clave se sigue el siguiente orden:

  - Palabra clave
  - Explicación general
  - Sintáxis
  - Aclaraciones
  - Ejemplo
  - Referencia a otras palabras claves y temas

ABS

ABS devuelve el valor absoluto de un número.
SGN devuelve un valor indicando el signo de una expresión numérica
(1 si es positivo, 0 si es cero o -1 si es negativo).

```
ABS(expresión-numérica)
SGN(expresión-numérica)
```

    * expresión-numérica   Cualquier expresión numérica.

```
Ejemplo:
    PRINT ABS(45.5 - 100!)           'Resultado:  54.5
    PRINT SGN(1), SGN(-1), SGN(0)    'Resultado:  1  -1  0
```

ABSOLUTE: Vea CALL ABSOLUTE

ACCESS: Vea OPEN

AND

Los operadores Booleanos realizan un manejo de bits, operaciones Booleanas,
o pruebas con respecto a múltiples relaciones. Generan un valor verdadero
(no cero) o falso (cero) que será utilizado al realizar una decisión.

```
resultado = expresión1 operador-Booleano expresión2
```

    * operador-Booleano   Cualquier operador Booleano de la lista
                          siguiente:

        NOT   Bit-wise complement
        AND   Conjunción

OR	Disjunción ("o" inclusivo)
XOR	"O" exclusivo
EQV	Equivalencia
IMP	Implicación

* Cada operador genera resultados así indicados en la tabla siguiente. V es verdadero (no cero); F es falso (no cero):

Expresión1	Expresión2	NOT	AND	OR	XOR	EQV	IMP
V	V	F	V	V	F	V	V
V	F	F	F	V	V	F	F
F	V	V	F	V	V	F	V
F	F	V	F	F	F	V	V

* Para NOT Expresión2 no es aplicable.
* Las operaciones Booleanas se realizan después de las operaciones aritméticas y relacionales, en el orden de precedencia.
* Las expresiones se convierten en enteros o enteros largos antes de realizar una operación booleana.
* Si las expresiones se evaluan como 0 ó -1, una operación Booleana generará un resultado 0 ó -1. Como los operadores Booleanos realizan cálculos bit-wise, el uso de valores que no sean 0 para falso y -1 para verdadero puede producir resultados inesperados.

ANY: Vea DECLARE

APPEND: Vea OPEN

AS

Realiza diferentes acciones como parte de varias instrucciones:

* En las instrucciones COMMON, DECLARE, DEF FN, DIM, FUNCTION, REDIM, SHARED, STATIC y SUB, especifica un tipo de variable.
* En la instrucción TYPE, especifica un tipo de elemento para un tipo de datos definido por el usuario.
* En la instrucción OPEN, asigna un número de archivo a un archivo o dispositivo.
* En la instrucción FIELD, especifica un nombre de campo.
* En la instrucción NAME, especifica un nuevo nombre de archivo.

Vea también: COMMON, DECLARE, DEF FN, DIM, REDIM, FIELD, FUNCTION, NAME, OPEN, SHARED, STATIC, SUB, TYPE

ASC

ASC devuelve el código ASCII correspondiente al primer carácter de una expresión de cadena.
CHR$ devuelve el carácter correspondiente al código ASCII especificado.

```
ASC(expresión-cadena$)
CHR$(código-ascii%)
```

* expresión-cadena$ Cualquier expresión de cadena.
* código-ascii% El código ASCII del carácter deseado.

Ejemplo:

```
PRINT ASC("Q")    'Resultado:  81
PRINT CHR$(65)    'Resultado:  A
```

ATRIBUTOS Y VALORES DE COLOR

	Monitor a color		Monitor monocromático	
Atributo de color	Valor pre-determinado(a)	Color pre-sentado	Valor pre-determinado	Color pre-sentado
Modos de pantalla (SCREEN) 0, 7, 8, 9(b), 12 y 13				
0	0	Negro	0(c)	Desactivado
1	1	Azul		Subrayado(d)
2	2	Verde	1(c)	Activado(d)
3	3	Azul-verdoso	1(c)	Activado(d)
4	4	Rojo	1(c)	Activado(d)
5	5	Magenta	1(c)	Activado(d)
6	6	Marrón	1(c)	Activado(d)
7	7	Blanco	1(c)	Activado(d)
8	8	Gris	0(c)	Desactivado
9	9	Azul claro		Alta intensidad Subrayado

10	10	Verde claro	2(c)	Alta intensidad
11	11	Azul-verdoso claro	2(c)	Alta intensidad
12	12	Rojo claro	2(c)	Alta intensidad
13	13	Magenta claro	2(c)	Alta intensidad
14	14	Amarillo	2(c)	Alta intensidad
15	15	Blanco de alta intensidad	0(c)	Desactivado

Modos de pantalla (SCREEN) 1 y 9(e)

0	0	Negro	0	Desactivado
1	11	Azul-verdoso claro	2	Alta intenidad
2	13	Magenta claro	2	Alta intensidad
3	15	Blanco de alta intensidad	0	Blanco desactivado

Modos de pantalla (SCREEN) 2 y 11

0	0	Negro	0	Desactivado
1	15	Blanco de alta intensidad	0	Blanco desactivado

(a) Números de color EGA. VGA y MCGA utilizan valores de presentación que reproducen colores visualmente equivalentes.
(b) Para VGA o EGA con memoria de video > 64Kb.
(c) Sólo para el modo 0.
(d) Desactivado cuando se utilice como color de fondo.
(e) EGA con memoria de video <= 64Kb.

Vea también: COLOR, PALETTE, PALETTE USING, SCREEN, Modos de pantalla

---

ATN

ATN devuelve la arcotangente de la expresión numérica especificada.
COS, SIN y TAN devuelven el coseno, el seno y la tangente del ángulo especificado.

```
ATN(expresión-numérica)
COS(ángulo)
SIN(ángulo)
TAN(ángulo)
```

* expresión-numérica La razón entre los lados de un triángulo recto.
* ángulo Un ángulo, expresado en radianes.
* La función ATN devuelve un ángulo en radianes.
* Para convertir de grados a radianes, multiplique grados por (PI / 180).

Ejemplo:

```
CONST PI=3.141592654
PRINT ATN(TAN(PI/4.0)), PI/4.0    'Resultado:  .7853981635  .7853981635
PRINT (COS(180 * (PI / 180)))     'Resultado:  -1
PRINT (SIN(90 * (PI / 180)))      'Resultado:  1
PRINT (TAN(45 * (PI / 180)))      'Resultado:  1.000000000205103
```

BASE: Vea OPTION BASE

BEEP

Genera una señal de advertencia desde el altavoz de la computadora.

```
BEEP
```

BINARY: Vea OPEN

BLOAD, BSAVE

BSAVE copia el contenido de un área de memoria a un archivo.
BLOAD carga un archivo creado mediante BSAVE en la memoria.

```
BSAVE archivo$, valor%, longitud&
BLOAD archivo$[,valor%]
```

* archivo$ Para BSAVE, el archivo en el que se copia una área de memoria (una imagen de memoria byte-por-byte). Para BLOAD, un archivo de imagen de memoria creado por una instrucción BSAVE anterior.
* valor% Para BSAVE, el valor de compensación de la dirección inicial del área de memoria que se está guardando. Para BLOAD, el valor de compensación de la dirección dónde se inicia el proceso de cargar.
* longitud& El número de bytes que se copiarán (de 0 a 65,535).

* La dirección de inicio del área de memoria guardada o cargada es determinada por el valor de compensación y la más reciente instrucción DEF SEG.

Vea también: DEF SEG, VARPTR, VARSEG

CALL

Transfiere el control a un procedimiento SUB.

[CALL] nombre [([listargumentos])]

* nombre — El nombre del procedimiento SUB que será llamado.
* listargumentos — Las variables o constantes que serán pasados al procedimiento SUB. Separe los argumentos con comas. Para especificar argumentos de matrices, use el nombre la matriz seguido de paréntesis vacíos.
* Si omite la palabra clave CALL, también deberá omitir los paréntesis alrededor de listargumentos. Declare el procedimiento en una instrucción DECLARE antes de llamarlo, o guarde el programa y QBasic generará automáticamente una instrucción DECLARE.
* Para especificar un argumento cuyo valor no será cambiado por el procedimiento, encierre el argumento entre paréntesis.

Ejemplo:

```
'Programa principal
DECLARE SUB linea (x%)
DECLARE FUNCTION porcent! (v!, t!)
DECLARE FUNCTION nombre$ (a$)

'Dibuja una línea de 10 guiones
linea 10

'Nombre que tiene la primera letra mayúscula y el resto en minúsculas
INPUT "Ingrese un nombre: ", n$
nom$ = nombre$(n$)
PRINT nom$

'Calculo del porcentaje
INPUT "Ingrese un valor: ", valor
INPUT "Ingrese el total: ", total
p = porcent(valor, total)
PRINT "El porcentaje es: "; p; "%"
END


'Procedimientos y funciones
SUB linea (x)
CLS
FOR n = 1 TO x
PRINT "-";
NEXT
PRINT
END SUB

FUNCTION nombre$ (a$)
nombre$ = UCASE$(LEFT$(a$, 1)) + LCASE$((RIGHT$(a$, LEN(a$) - 1)))
END FUNCTION

FUNCTION porcent (v, t)
porcent = v / t * 100
END FUNCTION
```

Vea también: CALL ABSOLUTE, DECLARE, SUB

CALL ABSOLUTE

Transfiere el control a un procedimiento en lenguaje de máquina.

CALL ABSOLUTE ([listargumentos,] desplazamiento%)

* listaargumentos — Argumentos pasados a un procedimiento en lenguaje de máquina como valor inicial desde el segmento de datos actual.
* valor% — La cantidad del valor inicial desde el segmento de código actual establecido mediante DEF SEG, hasta el punto de inicio del procedimiento.

Ejemplo:

```
    'Llama a una rutina para imprimir la pantalla en una impresora local.
    DIM a%(2)
    DEF SEG = VARSEG(a%(0))
    FOR i% = 0 TO 2
       READ d%
       POKE VARPTR(a%(0)) + i%, d%
    NEXT i%
    DATA 205, 5, 203  : ' int 5  retf  'Código en lenguaje de máquina
                                       'para imprimir la pantalla.
    CALL ABSOLUTE(VARPTR(a%(0)))
    DEF SEG
```

Vea también: CALL, VARPTR, VARSEG

CASE: Vea SELECT CASE

CDBL

CDBL convierte una expresión numérica en un valor de presición doble.
CSNG convierte una expresión numérica en un valor de precisión sencilla.

```
CDBL(expresión-numérica)
CSNG(expresión-numérica)
```

- expresión-numérica Cualquier expresión numérica.

```
Ejemplo:
    PRINT 1 / 3, CDBL(1 / 3)    'Resultado:  .3333333  .3333333333333333
    PRINT CSNG(975.3421515#)    'Resultado:  975.3422
```

Vea también: CINT, CLNG

CHAIN

Transfiere el control desde el programa actual a otro programa de Basic.

```
CHAIN archivo$
```

* archivo$ El nombre del programa al que se pasará el control.

```
Ejemplo:
    'Supone que el programa TEST.BAS está en el directorio \DOS.
    CHAIN "C:\DOS\TEST.BAS"
```

Vea también: CALL, COMMON, RUN

CHDIR

CHDIR cambia el directorio predeterminado de una unidad de disco. MKDIR crea un subdirectorio. MRDIR elimina un subdirectorio. FILES muestra el contenido del directorio en uso o del directorio especificado.

```
CHDIR ruta$
MKDIR ruta$
RMDIR ruta$
FILES (archivo$)
```

* ruta$ La ruta de acceso al nuevo directorio predeterminado, al subdirectorio que se va a crear, o eliminar.
* archivo$ Un nombre de archivo o ruta de acceso (puede incluir una unidad y comodines de DOS). Si no se especifica archivo$, FILES presentará todos los archivos que hay en el directorio en uso.

```
Ejemplo:
    MKDIR "C:\TEMP\TEST"
    CHDIR "C:\TEMP"
    FILES
    RMDIR "TEST"
```

CHR$: Vea ASC

CINT

CINT redondea una expresión numérica a entero.
CLNG redondea una expresión numérica a entero largo (de 4 bytes).

```
CINT(expresión-numérica)
CLNG(expresión-numérica)
```

* expresión-numérica Para CINT, cualquier expresión numérica entre -32,768 y 32,767.
Para CLNG, cualquier expresión numérica entre -2,147,483,648 y 2,147,483,647.

Ejemplo:
```
PRINT CINT(12.49), CINT(12.51)    'Resultado:  12  13
PRINT CLNG(338457.8)              'Resultado:  338458
```

Vea también: CDBL, CSNG, FIX, INT

CIRCLE

Traza un círculo o elipse en la pantalla.

CIRCLE [STEP] (x!,y!),radio![,[color%] [,[inicio!] [,[fin!] [,aspecto!]]]]

* STEP Especifica que las coordenadas estarán situadas con relación a la posición actual del cursor de gráficos.
* (x!,y!) Las coordenadas del centro del círculo o elipse.
* radio! El radio del círculo o elipse en unidades del sistema de coordenadas actual, determinado por las más recientes instrucciones SCREEN, VIEW y WINDOW.
* color% Un atributo de color que determina el color del círculo. Los atributos de color disponibles dependerán del adaptador de gráficos y del modo de pantalla establecido mediante la más reciente instrucción SCREEN.
* inicio! El ángulo inicial para el arco, en radianes.
* fin! El ángulo final para el arco, en radianes.
* aspecto! La dimensión entre la longitud del eje"y" la longitud del eje "x", utilizada para trazar elipses.
* Para convertir grados en radianes, multiplique grados por (PI / 180).

Ejemplo:
```
'Este ejemplo requiere un adaptador de gráficos a color.
SCREEN 2
CIRCLE (320, 100), 200
CIRCLE STEP(0, 0), 100
```

Vea también: COLOR, DRAW, LINE, SCREEN, VIEW, WINDOW, Atributos y valores de color, Modos de pantalla

---

CLEAR

Cierra todos los archivos, libera los búferes de archivos, borra las variables comunes, asigna el valor de cero a las variables numéricas y a las matrices, asigna el valor nulo a las variables de cadena e inicializa la pila. Si especifica la opción , CLEAR también puede cambiar el tamaño de la pila.

CLEAR [,,pila&]

* pila& Establece el tamaño (en bytes) de la pila para su programa.

Ejemplo:
```
CLEAR , , 2000
```

Vea también: ERASE

CLNG: Vea CINT

CLOSE

Cierra uno o más archivos o dispositivos abiertos.

CLOSE [[#]numarchivo%[,[#]numarchivo%]...]

* numarchivo% El número de un archivo o dispositivo abierto.
* CLOSE sin argumentos cerrará todos los archivos y dispositivos abiertos.

Ejemplo:
```
CLS
INPUT "Escriba el nombre del archivo: ", n$
OPEN n$ FOR OUTPUT AS #1
PRINT #1, "Esto se guarda en el archivo."
CLOSE
OPEN n$ FOR INPUT AS #1
INPUT #1, a$
PRINT "Leer el archivo: "; a$
CLOSE
```

Vea también: END, OPEN, RESET, STOP

CLS

Borra la pantalla.

CLS [{0 | 1 | 2}]

CLS	Borra la ventana gráfica o la de texto. Si se ha establecido una ventana gráfica (usando VIEW), se borrará sólo la ventana gráfica. De lo contrario, se borrará la ventana de texto o toda la pantalla.
CLS 0	Borra la pantalla, quitando todo el texto y los gráficos.
CLS 1	Borra la ventana de gráficos, o la pantalla completa si no se ha establecido una ventana para gráficos.
CLS 2	Borra la ventana de texto.

Vea también: VIEW, VIEW PRINT, WINDOW

CODIGOS DE CARACTERES ASCII

Códigos ASCII normales (códigos 0 - 127)

000	(nul)	016	(dle)	032 sp	048 0	064 @	080 P	096 `	112 p
001	(soh)	017	(dc1)	033 !	049 1	065 A	081 Q	097 a	113 q
002	(stx)	018	(dc2)	034 "	050 2	066 B	082 R	098 b	114 r
003	(etx)	019	(dc3)	035 #	051 3	067 C	083 S	099 c	115 s
004	(eot)	020	(dc4)	036 $	052 4	068 D	084 T	100 d	116 t
005	(enq)	021	(nak)	037 %	053 5	069 E	085 U	101 e	117 u
006	(ack)	022	(syn)	038 &	054 6	070 F	086 V	102 f	118 v
007	(bel)	023	(etb)	039 '	055 7	071 G	087 W	103 g	119 w
008	(bs)	024	(can)	040 (	056 8	072 H	088 X	104 h	120 x
009	(tab)	025	(em)	041)	057 9	073 I	089 Y	105 i	121 y
010	(lf)	026	(eof)	042 *	058 :	074 J	090 Z	106 j	122 z
011	(vt)	027	(esc)	043 +	059 ;	075 K	091 [	107 k	123 {
012	(np)	028	(fs)	044 ,	060 <	076 L	092 \	108 l	124 |
013	(cr)	029	(gs)	045 -	061 =	077 M	093]	109 m	125 }
014	(so)	030	(rs)	046 .	062 >	078 N	094 ^	110 n	126 ~
015	(si)	031	(us)	047 /	063 ?	079 O	095 _	111 o	127 □

Códigos ASCII extendidos (códigos 128 - 255)

128 Ç	143 Å	158 ×	172 ¼	186 ‖	200 ╚	214 Í	228 õ	242 ═
129 ü	144 É	159 ƒ	173 ¡	187 ╗	201 ╔	215 Î	229 Õ	243 ¾
130 é	145 æ	160 á	174 «	188 ╝	202 ╩	216 Ï	230 µ	244 ¶
131 â	146 Æ	161 í	175 »	189 ¢	203 ╦	217 ┘	231 þ	245 §
132 ä	147 ô	162 ó	176 ░	190 ¥	204 ╠	218 ┌	232 Þ	246 ÷
133 à	148 ö	163 ú	177 ▒	191 ┐	205 ═	219 █	233 Ú	247 ¸
134 å	149 ò	164 ñ	178 ▓	192 └	206 ╬	220 ▄	234 Û	248 °
135 ç	150 û	165 Ñ	179 │	193 ┴	207 ¤	221 ¦	235 Ù	249 ¨
136 ê	151 ù	166 ª	180 ┤	194 ┬	208 ð	222 Ì	236 ý	250 ·
137 ë	152 ÿ	167 º	181 Á	195 ├	209 Ð	223 ▀	237 Ý	251 ¹
138 è	153 Ö	168 ¿	182 Â	196 ─	210 Ê	224 Ó	238 ¯	252 ³
139 ï	154 Ü	169 ®	183 À	197 ┼	211 Ë	225 ß	239 ´	253 ²
140 î	155 ø	170 ¬	184 ©	198 ã	212 È	226 Ô	240 -	254 ■
141 ì	156 £	171 ½	185 ╣	199 Ã	213 ı	227 Ò	241 ±	255
142 Ä	157 Ø							

CODIGOS DE EXPLORACION DE TECLADO

Tecla	Código	Tecla	Código	Tecla	Código
Esc	1	A	30	Bloq Mayús	58
! ó 1	2	S	31	F1	59
@ ó 2	3	D	32	F2	60
# ó 3	4	F	33	F3	61
$ ó 4	5	G	34	F4	62
% ó 5	6	H	35	F5	63
^ ó 6	7	J	36	F6	64
& ó 7	8	K	37	F7	65
* ó 8	9	L	38	F8	66
(ó 9	10	: o ;	39	F9	67
) ó 0	11	" o '	40	F10	68
_ o -	12	~ o `	41	F11	133
+ o =	13	Mayús Izq	42	F12	134
Retroc	14	| o \	43	Bloq Num	69
Tab	15	Z	44	Bloq Despl	70
Q	16	X	45	Inicio ó 7	71
W	17	C	46	Arriba u 8	72
E	18	V	47	RePág ó 9	73
R	19	B	48	Gris -	74

T	20	N	49	Izq ó 4	75
Y	21	M	50	Centro ó 5	76
U	22	< o ,	51	Derecha ó 6	77
I	23	> o .	52	Gris +	78
O	24	? o /	53	Fin ó 1	79
P	25	Mayús Derech	54	Abajo ó 2	80
{ o [	26	Imp Pant o *	55	AvPág ó 3	81
} o]	27	Alt	56	Ins ó 0	82
Entrar	28	Barra Espac.	57	Supr o .	83
Ctrl	29				

CODIGOS DE ERRORES EN TIEMPO DE EJECUCION

Código	Mensaje	Código	Mensaje
1	NEXT sin FOR	37	Resultado-argto. no coinciden
2	Error de sintaxis	38	Matriz no definida
3	RETURN sin GOSUB	40	Variable requerida
4	Datos DATA agotados	50	Desborde de FIELD
5	Llamado a función no válido	51	Error interno
6	Desbordamiento	52	Nombre/num. archivo incorrecto
7	Memoria agotada	53	Archivo no se encontró
8	Etiqueta no definida	54	Modo de archivo incorrecto
9	Subíndice fuera de alcance	55	Archivo ya está abierto
10	Definición duplicada	56	Instrucción FIELD activa
11	División entre cero	57	Error en dispositivo E/S
12	No válido en modo directo	58	Archivo ya existe
13	Tipos no coinciden	59	Longitud de registro incorrecto
14	Espacio insuf.para cadenas	61	Disco lleno
16	Fórmula de cadena compleja	62	Info. excedió fin de archivo
17	No se puede continuar	63	Número de registro incorrecto
18	Función no definida	64	Nombre de archivo incorrecto
19	Falta RESUME	67	Demasiados archivos
20	RESUME sin error	68	Dispositivo no disponible
24	Límite de tiempo de dispos	69	Desborde de búfer de comunic.
25	Fallo en dispositivo	70	Permiso denegado
26	FOR sin NEXT	71	El disco no está listo
27	Falta papel	72	Error de disco/almacenamiento
29	WHILE sin WEND	73	Característica no disponible
30	WEND sin WHILE	74	Cambiar nombre entre discos
33	Etiqueta duplicada	75	Error de acceso en ruta/arch.
35	Subprograma no definido	76	Ruta de acceso no se encontró

COLOR

Establece los colores presentados en la pantalla.

```
COLOR [primerplano%] [,[fondo%] [,bordes%]]   Modo de pantalla 0 (sólo texto)
COLOR [fondo%] [,paleta%]                     Modo de pantalla 1
COLOR [primerplano%]                          Modos de pantalla 4, 12, 13
COLOR [primerplano%] [,fondo&]                Modos de pantalla 7-10
```

* primerplano%
  primerplano&
  Un número que establece el color de primer plano de la pantalla. En el modo de pantalla 0, primerplano% es un atributo de color que establece el color al texto. En otros modos de pantalla, primerplano% es un atributo de color o valor de color de 4 bytes (modo de pantalla 4 solamente) que establece el color para el texto y líneas dibujadas.
* fondo%
  fondo&
  Un número que establece el color de fondo en la pantalla. En el modo de pantalla 0, fondo% es un atributo de color. En el modo de pantalla 1, fondo% es un valor de color de 4 bits. En los modos de pantalla 7-10, fondo& es un valor de color.
* bordes%
  Un atributo de color que establece el color de los bordes de la pantalla.
* paleta%
  Un número (0 ó 1) que especifica el juego de atributos de color que será utilizado:

paleta%	Atributo 1	Atributo 2	Atributo 3
0	Verde	Rojo	Marrón
1	Azul-verdoso	Magenta	Blanco Intenso

* Los atributos y valores de color disponibles dependerán del adaptador de gráficos y del modo de pantalla establecido mediante la última instrucción SCREEN. Si su sistema tiene un adaptador EGA, VGA o MCGA, utilice la instrucción PALLETTE para cambiar las asignaciones de color correspondientes a los atributos de color.

Ejemplo:

```
    'Este ejemplo requiere un adaptador de gráficos a color.
    SCREEN 7
    FOR i% = 0 TO 15
        COLOR i%
        PRINT i%
    NEXT i%
```

Vea también: DRAW, PAINT, PALETTE, PALETTE USING, SCREEN, Atributos y valores de color, Modos de pantalla

---

COM: Vea ON COM

COMMON

Define variables globales que podrán ser compartidas en todo el programa o entre programas encadenados.

COMMON [SHARED] listavariables

* SHARED — Indica que las variables serán compartidas en todos los procedimientos SUB o FUNCTION.
* listavariables — Una o más variables que serán compartidas:
  variable[( )] [AS tipo] [, variable[( )] [AS tipo]]...
  - variable — Un nombre de variable Basic. Los nombres de variables pueden tener hasta 40 caracteres y deben comenzar con una letra. Los caracteres válidos son A-Z, 0-9 y el punto (.). No se pueden usar letras acentuadas ni la ñ.
  - tipo — El tipo de datos de la variable (INTEGER, LONG, SINGLE, DOUBLE, STRING, o un tipo de datos definido por el usuario).
* A menos que haya sido declarada como matriz estática en una instrucción DIM anterior, una variable de matriz en una instrucción COMMON será una matriz dinámica. Será necesario establecer sus dimensiones posteriormente a través de una instrucción DIM o REDIM.

Vea también: CHAIN, DIM, REDIM, FUNCTION, SHARED, STATIC, SUB

CONST

Declara una o más constantes simbólicas.

CONST nombreconst = expresión [,nombreconst = expresión]...

* nombreconst — El nombre de la constante. Este nombre puede tener hasta 40 caracteres y debe comenzar con una letra. Los caracteres válidos son A-Z, 0-9 y el punto (.). No se pueden usar letras acentuadas ni la ñ.
* expresión — Una expresión asignada a la constante. La expresión puede consistir de literales (tales como 1.0), otras constantes, cualquier operador aritmético o lógico excepto el de exponenciación (^), o una sola cadena literal.

Ejemplo:

```
    CONST PI = 3.141593
    INPUT "Radio del círculo: "; r
    PRINT "Area = "; PI * r ^ 2
```

COS: Vea ATN

CSNG: Vea CDBL

CSRLIN: Vea LOCATE

CVD

MKI$, MKL$, MKS$ y MKD$ convierten números en cadenas numéricas que pueden ser almacenadas en variables de cadena a través de la instrucción FIELD. CVI, CVL, CVS y CVD convierten esas cadenas nuevamente en números.

MKI$(expresión-entero%)
MKL$(expresión-entero-largo&)
MKS$(expresión-precisión-sencilla!)
MKD$(expresión-precisión-doble#)
CVI(cadena-numérica-de-2-bytes)
CVL(cadena-numérica-de-4-bytes)
CVS(cadena-numérica-de-4-bytes)

CVD(cadena-numérica-de-8-bytes)

Función	Devuelve	Función	Devuelve
MKI$	Cadena de 2 bytes	CVI	Entero
MKL$	Cadena de 4 bytes	CVL	Entero largo
MKS$	Cadena de 4 bytes	CVS	Número de precisión sencilla
MKD$	Cadena de 8 bytes	CVD	Número de precisión doble

Vea también: FIELD, MKSMBF$, MKDMBF$, CVSMBF, CVDMBF

CVDMBF

MKSMBF$ y MKDMBF$ convierten números con el formato IEEE en cadenas numéricas con el formato binario de Microsoft que pueden ser almacenadas en variables de cadena a través de la instrucción FIELD.
CVSMBF y CVDMBF convierten esas cadenas nuevamente a números con el formato IEEE.

MKSMBF$(expresión-precisión-sencilla!)
MKDMBF$(expresión-precisión-doble#)
CVSMBF (cadena-numérica-de-4-bytes)
CVDMBF (cadena-numérica-de-8-bytes)

Función	Devuelve
MKSMBF$	Una cadena de 4 bytes que contiene un número con el formato binario de Microsoft
MKDMBF$	Una cadena de 8 bytes que contiene un número con el formato binario de Microsoft
CVSMBF	Un número de precisión sencilla con el formato IEEE
CVDMBF	Un número de precisión doble con el formato IEEE

* Estas funciones son útiles para mantener archivos de datos creados con versiones antiguas de Basic.

Ejemplo:

```
TYPE Buffer
    SngNum AS STRING * 4
    DblNum AS STRING * 8
END TYPE
DIM RecBuffer AS Buffer
OPEN "TESTDAT.DAT" FOR RANDOM AS #1 LEN = 12
SNum = 98.9
DNum = 645.3235622#
RecBuffer.SngNum = MKSMBF$(SNum)    RecBuffer.DblNum = MKDMBF$(DNum)
PUT #1, 1, RecBuffer
GET #1, 1, RecBuffer
CLOSE #1
PRINT CVSMBF(RecBuffer.SngNum), CVDMBF(RecBuffer.DblNum)
```

Vea también: FIELD, MKn$, CVn

CVI, CVL, CVS: Vea CVD

CVSMBF: Vea CVDMBF

DATA

DATA especifica los valores que serán leídos por instrucciones READ subsecuentes.
READ lee esos valores y los asigna a variables.
RESTORE permite a READ volver a leer valores en las instrucciones DATA especificadas.

DATA constante[,constante]...
READ listavariables
RESTORE [línea]

* constante — Una o más constantes numéricas o de cadena que especifican los datos que serán leídos. Las constantes de cadena que contengan comas, dos puntos (:) o espacios antes o después serán puestas entre comillas (" ").
* listavariables — Una o más variables, separadas por comas, a las que se asignarán valores. Los nombres de variables pueden tener hasta 40 caracteres y deben comenzar con una letra. Los caracteres válidos son A-Z, 0-9 y el punto (.). No se pueden usar letras acentuadas ni la ñ.
* línea — La etiqueta o número de línea de una instrucción DATA.

Si se omite, la siguiente instrucción READ leerá los valores de la primera instrucción DATOS del programa.
* Sólo se pueden introducir instrucciones DATA al nivel de módulo. No pueden ser utilizados en procedimientos.

Ejemplo:
```
FOR i% = 1 TO 3
    READ a%, b$
    PRINT a%, b$
    RESTORE
NEXT i%
DATA 1, "Repetir"
```

DATE$

La función DATE$ function devuelve la fecha actual del sistema de su computadora.
La instrucción DATE$ define la fecha actual en el sistema de su computadora.

```
DATE$
DATE$ = expresión-cadena$
```

* expresión-cadena$ La fecha, en una de las siguientes formas: dd-mm-aa, dd-mm-aaaa, dd/mm/aa, dd/mm/aaaa.
* La función DATE$ devuelve una cadena en la forma mm-dd-aaaa.

Ejemplo:
```
PRINT DATE$
DATE$ = "01-01-90"    'Nota: La nueva fecha del sistema permanecerá
                      '      vigente hasta que usted la vuelva a cambiar.
PRINT "Fecha cambiada a "; DATE$
```

Vea también: TIME$

DECLARE

Declara una función FUNCTION o subprograma SUB e invoca la verificación de tipo de datos para los argumentos.

```
DECLARE {FUNCTION | SUB} nombre [([listaparámetros])]
```

* nombre — El nombre del procedimiento.
* listaparámetros — Una o más variables que especifican los parámetros que serán pasados al procedimiento cuando éste sea llamado:
  variable[( )] [AS tipo] [, variable[( )] [AS tipo]]...
  variable — El nombre de una variable Basic.
  tipo — El tipo de datos de la variable (INTEGER, LONG, SINGLE, DOUBLE, STRING o un tipo de datos definido por el usuario). ANY permite cualquier tipo de datos.
* DECLARE es requerido si se hace un llamado a un procedimiento SUB sin CALL. QBasic generará automáticamente instrucciones DECLARE cuando se guarde un programa.

Vea también: Ejemplo en CALL, FUNCTION, SUB

DEF FN

Define una función.

```
DEF FNnombre(listaparámetros)] = expresión
DEF FNnombre[(listaparámetros)]
    [bloqueinstrucciones]
  FNnombre = expresión
    [bloqueinstrucciones]
  EXIT DEF]
    [bloqueinstrucciones]
  END DEF
```

* listaparámetros — Uno o más argumentos con esta forma:
  variable[( )] [AS tipo] [, variable[( )] [AS tipo]]...
  variable — Un nombre de variable Basic.
  tipo — El tipo de datos de la variable (INTEGER, LONG, SINGLE, DOUBLE, STRING o un tipo de datos definido por el usuario).
* expresión — El valor devuelto por la función.
* La instrucción FUNCTION ofrece una mejor manera de definir una función.

Vea también: EXIT, FUNCTION, SHARED, STATIC

DEF SEG

Define la dirección del segmento actual.

```
DEF SEG [=dirección]
```

* dirección Una dirección de segmento utilizada por BLOAD, BSAVE, CALL ABSOLUTE, PEEK o POKE; un valor entre 0 y 65,535. Si se omite dirección, DEF SEG asignará el valor del segmento de datos predeterminado a la dirección de segmento actual.

Ejemplo:

```
DEF SEG = 0
Estado% = PEEK(&H417)              'Lee el estado del teclado.
POKE &H417, (Estado% XOR &H40)     'Cambia estado de Bloq Mayús, bit 6.
```

Vea también: BSAVE, BLOAD, CALL ABSOLUTE, PEEK, POKE

DEFDBL

Define el tipo de de datos predeterminado para variables, funciones DEF FN y procedimientos FUNCTION.

```
DEFINT letras [,letras]...
DEFLNG letras [,letras]...
DEFSNG letras [,letras]...
DEFDBL letras [,letras]...
DEFSTR letras [,letras]...
```

* letras Una letra o serie de letras (por ejemplo, A-M). QBasic establecerá el tipo de datos predeterminado para variables, funciones DEF FN y procedimientos FUNCTION cuyos nombres comienzan con la letra o letras especificadas, según se indica a continuación:

Instrucción	Tipo de datos predeterminado
DEFINT	Entero
DEFLNG	Entero largo
DEFSNG	Precisión sencilla
DEFDBL	Precisión doble
DEFSTR	Cadena

* Un sufijo indicando el tipo de datos (%, &, !, # o $) siempre tendrá prioridad sobre una instrucción DEFtipo.
* Precisión sencilla será el tipo de datos predeterminado si no se especifica una instrucción DEFtype.
* Después de especificar una instrucción DEFtipo en un programa, QBasic insertará automáticamente una instrucción DEFtipo correspondiente en cada procedimiento que usted haya creado.

Ejemplo:

```
DEFDBL A-Z
a = SQR(3)
PRINT "Raíz cuadrada de 3 = "; a
```

DEFINT, DEFLNG, DEFSNG, DEFSTR: Vea DEFDBL

DIM

DIM declara una matriz o especifica un tipo de datos para una variable que no sea una matriz.
REDIM declara o cambia el tamaño de una matriz dinámica, borrando los valores anteriores.

```
DIM [SHARED] variable[(subíndices)] [AS tipo]
             [,variable[(subíndices)] [AS tipo]]...
REDIM [SHARED] variable(subíndices) [AS tipo]
               [,variable(subíndices) [AS tipo]]...
```

* SHARED Especifica que las variables serán compartidas con todos los procedimientos SUB o FUNCTION existentes en el módulo.
* variable El nombre de una matriz o variable.
* subíndices Las dimensiones de la matriz, indicadas de esta forma: [inferior TO] superior [,[inferior TO] superior]...
  inferior El límite inferior de los subíndices de la matriz. El valor predeterminado es 0.
  superior El límite superior.

* AS tipo Declara el tipo de datos de la matriz o variable (INTEGER, LONG, SINGLE, DOUBLE, STRING o un tipo de datos definido por el usuario).
* DIM declara matrices estáticas o dinámicas. A menos que se haya determinado el almacenamiento de la matriz mediante $STATIC, $DYNAMIC o COMMON, las matrices cuyas dimensiones son definidas con números serán estáticas y las matrices cuyas dimensiones son definidas con variables serán dinámicas. REDIM siempre declara matrices dinámicas.
* El almacenamiento de matrices estáticas será asignado al iniciar el programa y permanecerá fijo. El almacenamiento de matrices dinámicas será asignado durante la ejecución del programa.

Ejemplo:

```
' $DYNAMIC
DIM A(49, 49)
REDIM A(19, 14)
```

Vea también: COMMON, ERASE, OPTION BASE, SHARED, STATIC, $STATIC, $DYNAMIC

DO

Repite un bloque de instrucciones mientras una condición tenga el estado verdadero, o hasta que una condición adquiera el estado verdadero.

```
DO [{WHILE | UNTIL} condición]
    [bloqueinstrucciones]
LOOP

DO
    [bloqueinstrucciones]
LOOP [{WHILE | UNTIL} condición]
```

* condición Una expresión numérica que Basic evalúa como verdadero (no cero) o falso (cero).

Ejemplo:

```
i% = 0
PRINT "El valor de i% al prinicipio del bucle es  "; i%
DO WHILE i% < 10
    i% = i% + 1
LOOP
PRINT "El valor de i% al final del bucle es  "; i%
```

Vea también: EXIT, FOR, WHILE

DOUBLE: Vea Tipos de datos

DRAW

Dibuja un objeto.

```
DRAW cadenacomando$
```

* cadenacomando$ Una expresión de cadena que contiene uno o más de los comandos DRAW.

  Comandos para trazar líneas y movimientos del cursor:

D[n%]	Mueve el cursor hacia abajo n% unidades.
E[n%]	Mueve el cursor hacia arriba y hacia la derecha n% unidades.
F[n%]	Mueve el cursor hacia abajo y hacia la derecha n% unidades.
G[n%]	Mueve el cursor hacia abajo y hacia la izquierda n% unidades.
H[n%]	Mueve el cursor hacia arriba y hacia la izquierda n% unidades.
L[n%]	Mueve el cursor hacia la izquierda n% unidades.
M[{+|-}]x%,y%	Mueve el cursor al punto x%,y%. Si x% va precedido de + ó -, moverá el cursor con relación al punto actual.
R[n%]	Mueve el cursor hacia la derecha n% unidades.
U[n%]	Mueve el cursor hacia arriba n% unidades.
[B]	Prefijo optativo que mueve el cursor sin dibujar.
[N]	Prefijo optativo que dibuja y devuelve el cursor a su posición original.

  Comandos de color, rotación y escala:

An%	Rota un objeto n% * 90 grados (n% puede ser 0, 1, 2 ó 3).
Cn%	Establece el color para dibujar (n% es un atributo de color).
Pn1%,n2%	Establece los colores de relleno y bordes de un

objeto (n1% es el atributo de color de relleno, n2% es el atributo de color de bordes).

Sn% Determina la escala de dibujo estableciendo la longitud de una unidad de movimiento del cursor. El valor predeterminado para n% es 4, el equivalente de un pixel.

TAn% Rota un ángulo n% grados (de -360 a 360).

* Si se omite n% de los comandos para trazar líneas y mover el cursor, el cursor se moverá 1 unidad.
* Para ejecutar una subcadena del comando DRAW desde una cadena del comando DRAW, utilice el comando "X":
  DRAW "X"+ VARPTR$(cadenacomando$)

Ejemplo:

```
'Este ejemplo requiere un adaptador de gráficos a color.
SCREEN 1
Triángulo$ = "F60 L120 E60"
DRAW "C2 X" + VARPTR$(Triángulo$)
DRAW "BD30 P1,2 C3 M-30,-30"
```

Vea también: PALETTE, PALETTE USING, SCREEN, VARPTR$, Atributos y valores de color

---

$DYNAMIC

Establece el almacenamiento predeterminado de matrices.

```
{REM | '} $STATIC
{REM | '} $DYNAMIC
```

* {REM | '} REM o un carácter de comentario (') debe preceder los metacomandos.
* $STATIC Especifica que las matrices declaradas en instrucciones DIM subsecuentes sean matrices estáticas (a menos que sean declaradas en un procedimiento SUB o FUNCTION no estática). El almacenamiento de matrices será asignado al iniciar el programa y permanecerá fijo.
* $DYNAMIC Especifica que las matrices declaradas en instrucciones DIM subsecuentes sean matrices dinámicas. El almacenamiento de matrices será asignado en forma dinámica durante la ejecución del programa.

* DIM y REDIM generalmente constituyen una mejor manera de especificar si las matrices deben ser dinámicas o estáticas.

Vea también: DIM, REDIM, REM, SHARED, STATIC

ELSE, ELSEIF: Vea IF

END

Pone fin a un programa, procedimiento, bloque o tipo de datos definido por el usuario.

```
END [{DEF | FUNCTION | IF | SELECT | SUB | TYPE}]
```

* DEF Termina la definición de una función DEF FIN que ocupe varias líneas.
* FUNCTION Termina la definición de un procedimiento FUNCTION.
* IF Termina un bloque de instrucciones IF...THEN...ELSE.
* SELECT Termina un bloque SELECT CASE.
* SUB Termina un procedimiento SUB.
* TYPE Termina la definición de un tipo de datos definido por el usuario.
* Si no se especifica ningún argumento, END pondrá fin al programa y cerrará todos los archivos.

Ejemplo:

```
PRINT "Juego terminado."
END
```

Vea también: DEF FN, FUNCTION, IF, SELECT CASE, STOP, SUB, SYSTEM, TYPE

ENVIRON

ENVIRON$ devuelve una cadena de ambiente DOS.
ENVIRON cambia o agrega una cadena de ambiente en la tabla de ambiente DOS.

```
ENVIRON$ (variable-ambiente$)
ENVIRON$ (n%)
```

ENVIRON expresióncadena$

* variable-ambiente$ El nombre de una variable de ambiente de DOS.
* n% Especifica que ENVIRON$ devolverá la enésima cadena de la tabla de cadenas de ambiente.
* expresióncadena$ El nombre y valor de una variable de ambiente de DOS (tal como PATH o PROMPT) en una de las siguientes formas:
  variable-ambiente$=cadena-ambiente$
  variable-ambiente$ cadena-ambiente$
* Los cambios realizados a través de la instrucción ENVIRON serán borrados cuando se termine el programa.

Ejemplo:

```
ENVIRON "PATH=TEST"
PRINT ENVIRON$("PATH")
```

EOF

Verifica si se ha llegado al final de un archivo. EOF da como resultado verdadero (no cero) si se ha llegado al final de un archivo.

EOF(numarchivo%)

* numarchivo% El número de un archivo abierto.

Ejemplo:

```
CLS
OPEN "TEST.DAT" FOR OUTPUT AS #1
FOR i% = 1 TO 10
    WRITE #1, i%, 2 * i%, 5 * i%
NEXT i%
CLOSE #1
OPEN "TEST.DAT" FOR INPUT AS #1
DO
    LINE INPUT #1, a$
    PRINT a$
LOOP UNTIL (EOF(1))
```

Vea también: CLOSE, LOC, LOF, OPEN

ERASE

Reinicializa los elementos de una matriz o libera espacio de almacenamiento para matrices dinámicas.

ERASE nombrematriz [,nombrematriz]...

* nombrematriz El nombre de una matriz.
* Para matrices estáticas, ERASE asigna cero a cada elemento de matrices numéricas y asigna nulo a cada elemento de matrices de cadena.
* Para matrices dinámicas, ERASE libera la memoria utilizada por la matriz. Hay que volver a declarar las dimensiones de la matriz usando REDIM o DIM antes de usarla.

Ejemplo:

```
DIM a%(0)
a%(0) = 6
PRINT "Antes: "; a%(0)
ERASE a%
PRINT "Después: "; a%(0)
```

Vea también: CLEAR, DIM, REDIM

ERDEV

ERDEV devuelve un código de error del último dispositivo que haya generado un error crítico. ERDEV$ da el nombre del dispositivo que generó el error.

ERDEV
ERDEV$

* El byte inferior del valor producido por ERDEV contiene el código de error de DOS. El byte superior contiene información sobre atributos del dispositivo.

Ejemplo

```
'Muestra el uso de ERDEV, ERDEV$, ERL, ERR, ERROR, ON ERROR y RESUME.
ON ERROR GOTO Manejador
```

```
CHDIR "a:\"                'Origina ERR 71 "El disquete no está listo"
                           'si no hay disquete en la Unidad A.
y% = 0
x% = 5 / y%                'ERR 11 "División entre cero."
PRINT "x% ="; x%
ERROR 57                   'ERR 57 "Error de dispositivo I/O."

Manejador:
  PRINT
  PRINT "Error "; ERR; " en la línea "; ERL
  SELECT CASE ERR
    CASE 71
      PRINT "Usando el dispositivo "; ERDEV$; " código de error  = "; ERDEV
      RESUME NEXT
    CASE 11
      INPUT "Escriba el valor por el que desee dividir"; y%
      RESUME               'Reintentar con el nuevo valor de y%.
    CASE ELSE
      PRINT "Error inesperado, terminando el programa."
      END
  END SELECT
```

Vea también: ERR, ERL, ON ERROR, Códigos de errores de ejecución

---

ERR, ERL

ERR devuelve el código de error de ejecución del error más reciente.
ERL devuelve el número de línea donde ocurrió el error, o el número de línea más cercano antes de la línea donde ocurrió el error.

```
ERR
ERL
```

* ERL no devuelve las etiquetas de líneas. Si no hay números de línea en el programa, ERL devolverá el valor 0.

Ejemplo

Vea también: ERDEV, ERDEV$, ERROR, ON ERROR, RESUME, Códigos de errores de ejecución

---

ERROR

Simula el acontecimiento de un error de Basic o un error definido por el usuario.

```
ERROR expresión%
```

* expresión% El código de error correspondiente a un error de Basic o un error definido por el usuario; un valor entre 1 y 255. Para definir su propio error, utilice un valor que no aparezca en la tabla de Códigos de errores de ejecución.

Vea también: ERDEV, ERDEV$, ERL, ERR, ON ERROR, RESUME, Códigos de errores de ejecución

---

EXIT

Sale de un bucle DO o FOR, un procedimiento FUNCTION o SUB, o una función DEF FN.

```
EXIT {DEF | DO | FOR | FUNCTION | SUB}
```

* DEF Sale de una función DEF FN.
* DO Sale de un bucle DO.
* FOR Sale de un bucle FOR.
* FUNCTION Sale de un procedimiento FUNCTION.
* SUB Sale de un procedimiento SUB.

Ejemplo:

```
    i% = 0
    DO
        i% = i% + 1
        IF i% = 500 THEN EXIT DO
    LOOP
    PRINT "SALE en"; i%
```

Vea también: DEF FN, DO, FOR, FUNCTION, SUB

EXP

EXP devuelve "e" elevado a una potencia especificada, donde "e" es la base del logaritmo natural. LOG devuelve el logaritmo natural de una expresión numérica.

```
EXP(expresión-numérica)
LOG(expresión-numérica)
```

- expresión-numérica Para EXP, un número menor o igual que 88.02969. Para LOG, cualquier expresión numérica positiva.

Ejemplo:

```
PRINT EXP(0), EXP(1)         'Resultado:  1  2.718282
PRINT LOG(1), LOG(EXP(1))    'Resultado:  0  1
```

FIELD

Asigna espacio para variables en un búfer de archivo de acceso aleatorio.

```
FIELD [#]numarchivo%, anchocampo% AS variablecadena$
                  [,anchocampo% AS variablecadena$] ...
```

* numarchivo% El número de un archivo abierto.
* anchocampo% El número de caracteres en el campo.
* variablecadena$ Una variable que identifica el campo y contiene datos para el campo.
* Las variables de registro generalmente constituyen una mejor manera de manejar los datos de registros.

Ejemplo:

```
OPEN "FILEDAT.DAT" FOR RANDOM AS #1 LEN = 80
FIELD #1, 30 AS nombre$, 50 AS dirección$
```

Vea también: GET, PUT, LSET, RSET, TYPE

FILEATTR

Devuelve información acerca de un archivo abierto.

```
FILEATTR(numarchivo%,atributo%)
```

* numarchivo% El número de un archivo abierto.
* atributo% Especifica el tipo de información que dará. Si abributo% es 1, FILEATTR devolverá un valor que indica el modo de acceso del archivo:

Valor	Modo
1	Entrada
2	Salida
4	Aleatorio
8	Anexado
32	Binario

Si atributo% es 2, FILEATTR devolverá los identificadores de archivos de DOS.

Ejemplo:

```
OPEN "TEST.DAT" FOR BINARY AS #1
PRINT FILEATTR(1, 1)
CLOSE
```

Vea también: OPEN

FILES: Vea CHDIR

FIX

FIX trunca una expresión de punto flotante, dejando la porción entera. INT devuelve el entero más grande que sea menor o igual que una expresión numérica especificada.

```
FIX(expresión-numérica)
INT(expresión-numérica)
```

- expresión-numérica Cualquier expresión numérica.

Ejemplo:

```
PRINT FIX(12.49), FIX(12.54)    'Resultado:  12  12
```

```
    PRINT INT(12.54), INT(-99.4)    'Resultado:  12  -100

Vea también: CINT, CLNG
```

FOR

Repite un bloque de instrucciones el número de veces especificado.

```
FOR contador = inicio TO fin [STEP incremento]
    [bloqueinstrucciones]
NEXT [contador [,contador]...]
```

* contador — Una variable numérica utilizada como contador de bucle.
* inicio y fin — Los valores inicial y final del contador.
* incremento — El incremento con el que se cambia el contador cada vez que se ejecute el bucle.

```
Ejemplo:
    FOR i% = 1 TO 15
        PRINT i%
    NEXT i%
    FOR i% = 7 to -6 STEP -3
        PRINT i%
    NEXT i%

Vea también: DO, EXIT, WHILE, OPEN
```

FRE

Devuelve la cantidad (en bytes) de memoria libre o disponible.

```
FRE(expresión-numérica)
FRE(expresión-cadena$)
```

* expresión-numérica — Un valor que especifica el tipo de memoria:

Valor	FRE devuelve
-1	El tamaño de la matriz más grande (no de cadena) que se pueda crear
-2	Espacio de pila disponible
Cualquier otro número	La cantidad de espacio disponible para cadenas

* expresión-cadena$ — Cualquier expresión de cadena. FRE compacta el espacio disponible para cadenas en un solo bloque, y después devuelve la cantidad de espacio disponible para cadenas.

```
Ejemplo:
    PRINT "Espacio para cadenas", FRE("")
    PRINT "Espacio disponible en pila", FRE(-2)
    PRINT "Espacio para matrices", FRE(-1)
```

FREEFILE

Devuelve el siguiente número de archivo válido que esté disponible.

```
Ejemplo:
    OPEN "TEST.DAT" FOR OUTPUT AS #1
    PRINT "Siguiente número de archivo: "; FREEFILE
    CLOSE

Vea también: OPEN
```

FUNCTION

Define un procedimiento FUNCTION.

```
FUNCTION nombre [(listaparámetros)] [STATIC]
    [bloqueinstrucciones]
  nombre = expresión
    [bloqueinstrucciones]
END FUNCTION
```

* nombre — El nombre de la función y tipo de datos que devuelve especificado por un sufijo de tipo de datos (%, &, !, #, o $).

* listaparámetros Una o más variables que especifican parámetros que serán pasados a la función cuando ésta sea llamada:
  variable[( )] [AS tipo] [, variable[( )] [AS tipo]]...
  variable Un nombre de variable Basic.
  tipo El tipo de datos de la variable (INTEGER, LONG, SINGLE, DOUBLE, STRING, o un tipo de datos definido por el usuario).
* STATIC Especifica que los valores de las variables locales de la función seán guardados entre llamados a la función.
* expresión El valor de la función devuelto.
* Cuando llama una función, podrá especificar que el valor de un argumento no sea cambiado por la función, poniendo el argumento entre paréntesis.

Vea también: Ejemplo en CALL, DECLARE, DEF FN, EXIT, SHARED, STATIC, SUB

GET (archivos)

GET lee información de un archivo colocándola en un búfer de acceso aleatorio o en una variable.
PUT escribe una variable o un búfer de acceso aleatorio en un archivo.

```
GET [#]numarchivo%[,[numregistro&][,variable]]
PUT [#]numarchivo%[,[numregistro&][,variable]]
```

* numarchivo% El número de un archivo abierto.
* numregistro& Para archivos de acceso aleatorio, es el número del registro que será leído o escrito. Para archivos binarios la posición de byte donde se inicia el proceso de leer o escribir.
* variable Para GET, es una variable utilizada para recibir información del archivo. Para PUT, una variable que contiene la información de salida que será escrita en el archivo. La variable generalmente tiene el tipo de datos definido por el usuario.

Ejemplo:

```
TYPE RegistroPrueba
    Alumno AS STRING * 20
    Nota AS SINGLE
END TYPE
DIM MiClase AS RegistroPrueba
OPEN "FINAL.DAT" FOR RANDOM AS #1 LEN = LEN(MiClase)
MiClase.Estudiante = "SabineCse"
MiClase.Nota = 99
PUT #1, 1, MiClase
CLOSE #1
OPEN "FINAL.DAT" FOR RANDOM AS #1 LEN = LEN(MiClase)
GET #1, 1, MiClase
PRINT "ESTUDIANTE:", MiClase.Alumna
PRINT "NOTA:", MiClase.Nota
CLOSE #1
KILL "FINAL.DAT"
```

Vea también: FIELD, GET (gráficos), PUT (gráficos), LSET, RSET, MKn$, CVn, TYPE

GET (gráficos)

GET captura una imagen de pantalla en gráficos. PUT presenta una imagen capturada mediante GET.

```
GET [STEP](x1!,y1!)-[STEP](x2!,y2!), matriz[(índice%)]
PUT [STEP] (x1!,y1!), matriz[(índice%)] [,palabra]
```

* STEP Especifica que las coordenadas estarán situadas con relación a la posición actual del cursor de gráficos.
* (x1!,y1!) Las coordenadas superior izquierda de la imagen capturada mediante GET o la posición en la pantalla donde PUT colocará la imagen.
* (x2!,y2!) Las coordenadas inferior derecha de la imagen capturada.
* matriz Nombre de la matriz donde será almacenada la imagen. Vea Matrices para imágenes de pantalla y compatibilidad (más adelante) para averiguar el tamaño requerido para la matriz.
* índice% El índice de la matriz donde comenzará el almacenamiento de la imagen.
* palabra Una palabra clave indicando la forma en que se presentará

la imagen:

Palabra	Efecto
AND	Combina la imagen almacenada con una imagen existente.
OR	Superpone la imagen almacenada en una imagen existente.
PSET	Traza una imagen almacenada, borrando una imagen existente.
PRESET	Traza una imagen almacenada en colores inversos, borrando una imagen existente.
XOR	Traza una imagen almacenada o borra una imagen trazada anteriormente y conserva el fondo, produciendo efectos de animación.

* Siempre se debe ejecutar una instrucción PUT en el mismo modo de pantalla que la instrucción GET utilizada para capturar la imagen, o en un modo compatible.
* Matrices para Imágenes de Pantalla y Compatibilidad.
  Utilice valores de bits por pixel por plano y valores de plano para determinar el tamaño requerido de la matriz que almacena una imagen gráfica de pantalla. Los valores de bits por pixel por plano y valores de plano, junto con la resolución horizontal, también determinan los modos de pantalla compatibles.

Modo de pantalla	Bits por pixel por plano	Planos	Resolución horizontal (en pixel)
1	2	1	320
2, 4, 11	1	1	640
3	1	1	720
7	1	4	320
8, 9(> 64Kb memoria de video), 12	1	4	640
9(64Kb memoria de video), 10	1	2	640
13	8	1	320

La siguiente fórmula da el tamaño requerido, en bytes, de una matriz utilizada para almacenar una imagen capturada:

```
tamaño% = 4 + INT(((PMAP (x2!, 0) - PMAP (x1!, 0) + 1) *
          (bits-por-pixel-por-plano%) + 7) / 8) * planos% *
          (PMAP (y2!, 1) - PMAP (y1!, 1) + 1)
```

Las operaciones GET y PUT son compatibles en modos de pantalla que tengan los mismos valores de resolución horizontal, bits por pixel por plano y plano. Por ejemplo, los modos de pantalla 2, 4 y 11 son compatibles y los modos de pantalla 8 y 12 son compatibles.

Ejemplo:

```
'Este ejemplo requiere un adaptador de gráficos a color.
SCREEN 1
DIM Cuadro%(1 TO 200)
x1% = 0: x2% = 10: y1% = 0: y2% = 10
LINE (x1%, y1%)-(x2%, y2%), 2, BF
GET (x1%, y1%)-(x2%, y2%), Cuadro%
DO
    PUT (x1%, y1%), Cuadro%, XOR
    x1% = RND * 300
    y1% = RND * 180
    PUT (x1%, y1%), Cuadro%
LOOP WHILE INKEY$ = ""
```

Vea también: SCREEN, Modos de Pantalla

GOSUB

Se bifurca a una subrutina y regresa desde la misma.

```
GOSUB línea1
.
.
.
RETURN [línea2]
```

* línea1 La etiqueta o el número de línea correspondiente a la primera línea de la subrutina.
* línea2 La etiqueta o el número de línea al que regresará la subrutina.
* Si no indica una etiqueta o número de línea en RETURN, el programa continuará su ejecución a partir de la instrucción que le sigue a GOSUB

(para llamadas a subrutinas) o a partir del lugar donde ocurrió un evento (para identificación de eventos). Consulte la información de Ayuda sobre la palabra clave ON para aprender más sobre instrucciones relacionadas con la identificación de eventos.
* Las instrucciones SUB y CALL constituyen una mejor alternativa para el uso de subrutinas GOSUB.

Ejemplo:
```
FOR i% = 1 TO 20
    GOSUB Cuadro
NEXT i%
END

Cuadro:
PRINT i%, i% * i%
RETURN
```

Vea también: CALL, ON...GOTO, ON...GOSUB, SUB

GOTO

Realiza una bifurcación a una línea especificada.

GOTO línea

* línea La etiqueta o el número de la siguiente línea que será ejecutada.
* DO...LOOP, SELECT CASE, IF...THEN...ELSE, SUB y FUNCTION ofrecen mejores formas de controlar el flujo de los programas.
* GOTO también se utiliza como palabra clave en la instrucción ON ERROR.

Ejemplo

Vea también: DO, FUNCTION, IF, ON ERROR, ON...GOTO, SELECT CASE, SUB

HEX$

HEX$ devuelve una representación hexadecimal de cadena de un número.
OCT$ devuelve una representación octal de cadena de un número.

```
HEX$(expresión-numérica&)
OCT$(expresión-numérica&)
```

* expresión-numérica& Cualquier expresión numérica. La expresión será redondeada a un entero o entero largo antes de ser evaluada.

Ejemplo:
```
INPUT x
a$ = HEX$ (x)
b$ = OCT$ (x)
PRINT x; "decimal es "; a$; " hexadecimal y "; b$; " en octal."
```

IF

Ejecuta una instrucción o bloque de instrucciones según las condiciones especificadas.

```
IF condición1 THEN
    [bloqueinstrucciones-1]
[ELSEIF condición2 THEN
    [bloqueinstrucciones-2]]...
[ELSE
    [bloqueinstrucciones-n]]
END IF
```

IF condición THEN instrucciones [ELSE instrucciones]

* condición1, condición2 Cualquier expresión que puede ser evaluada como verdadera (no cero) o falsa (cero).
* bloqueinstrucciones-1, bloqueinstrucciones-2, bloqueinstrucciones-n Una o más instrucciones en una o más líneas.
* instrucciones Una o más instrucciones, separadas con el signo de dos puntos.

Ejemplo:
```
INPUT "¿1 ó 2? ", i%
IF i% = 1 OR i% = 2 THEN
    PRINT "OK"
```

```
    ELSE
        PRINT "Valor no válido"
    END IF

Vea también: ON...GOSUB, ON...GOTO, SELECT CASE
```

IMP: Vea AND

INKEY$

Lee un carácter desde el teclado.

```
INKEY$
```

* INKEY$ genera una cadena nula si no hay ningún carácter para devolver.
* Para las teclas estándar, INKEY$ genera una cadena de 1 byte que contiene el carácter leído.
* Para las teclas extendidas, INKEY$ genera una cadena de 2 bytes que contienen el carácter nulo (ASCII 0) y el código de teclado.

```
Ejemplo:
    PRINT "Presione Esc para salir..."
    DO
    LOOP UNTIL INKEY$ = CHR$(27)    '27 es el código ASCII para Esc.

Vea también: INPUT, Códigos de exploración de teclado
```

INP

INP genera un byte leído del puerto (hardware) de Entrada y Salida (E/S).
OUT envía un byte a un puerto (hardware) de E/S.

```
INP(puerto%)
OUT puerto%, datos%
```

* puerto% Un número entre 0 y 65,535 que identifica el puerto.
* datos% Una expresión numérica entre 0 y 255 que será enviada al puerto.

```
Ejemplo:
    x% = INP(&H3FC)          'Leer COM1 Modem Control Register.
    OUT &H3FC, (x% XOR 1)    'Cambiar bit de Data Terminal Ready.

Vea también: WAIT
```

INPUT

INPUT lee información desde el teclado o desde un archivo. LINE INPUT lee una línea de hasta 255 caracteres desde el teclado o desde un archivo.

```
INPUT [;] ["mensaje"{; | ,}] listavariables
LINE INPUT [;] ["mensaje";] variable$
INPUT #numarchivo%, listavariables
LINE INPUT #numarchivo%, variable$
```

* mensaje Una cadena literal optativa que será presentada antes de que el usuario introduzca datos. Un punto y coma después del mensaje agregará un signo de interrogación al texto del mensaje.
* listavariables Una o más variables, separadas con comas, en las que serán almacenados los datos introducidos desde el teclado o leídos desde un archivo. Los nombres de variables pueden tener hasta 40 caracteres y deben comenzar con una letra. Los caracteres válidos son A-Z, 0-9 y el punto (.). No se pueden usar letras acentuadas ni la ñ.
* variable$ Almacena una línea de caracteres introducidos desde el teclado o leídos desde un archivo.
* numarchivo% El número de un archivo abierto.
* INPUT utiliza la coma como separador entre entradas. LINE INPUT lee todos los caracteres hasta encontrar un retorno de carro.
* Para datos introducidos desde el teclado, un punto y coma inmediatamente después de INPUT mantendrá el cursor en la misma línea después que el usuario presione la tecla Entrar.

```
Ejemplo:
    CLS
    OPEN "LISTA" FOR OUTPUT AS #1
    DO
```

```
        INPUT "   NOMBRE:       ", Nombre$  'Lee datos desde el teclado.
        INPUT "   EDAD:         ", Edad$
        WRITE #1, Nombre$, Edad$
        INPUT "¿Desea ingresar otros datos"; R$
    LOOP WHILE UCASE$(R$) = "S"
    CLOSE #1
    'Mostrar el archivo en la pantalla.
    OPEN "LISTA" FOR INPUT AS #1
    CLS
    PRINT "Datos en el archivo:": PRINT
    DO WHILE NOT EOF(1)
        LINE INPUT #1, REC$  'Leer datos del archivo.
        PRINT REC$           'Imprimir los datos en la pantalla.
    LOOP
    CLOSE #1
    KILL "LISTA"
```

Vea también: INKEY$, INPUT$, APPEND

INPUT$

Devuelve una cadena de caracteres leída desde el archivo especificado.

```
INPUT$(n[,[#]numarchivo%])
```

* n — El número de caracteres (bytes) que serán leídos.
* numarchivo% — El número de un archivo abierto. Si numarchivo% se omite, INPUT$ leerá desde el teclado.

Ejemplo:

```
    OPEN "TEST.DAT" FOR OUTPUT AS #1
    PRINT #1, "El texto"
    CLOSE
    OPEN "TEST.DAT" FOR INPUT AS #1
    PRINT INPUT$(3, 1)        'Imprime los 3 primeros caracteres.
    CLOSE
```

Vea también: INPUT, LINE INPUT

INSTR

Devuelve la posición de la primera aparición de una cadena dentro de otra.

```
INSTR([inicio%,]expresión-cadena1$,expresión-cadena2$)
```

* inicio% — Establece la posición de carácter donde se iniciará la búsqueda. Si se omite inicio%, INSTR se iniciará en la posición 1.
* expresión-cadena1$ — La cadena en la que se realizará la búsqueda.
* expresión-cadena2$ — La cadena que será buscada.

Ejemplo:

```
    a$ = "Microsoft QBasic"
    PRINT "Posición de cadena ="; INSTR(1, a$, "QBasic")
```

Vea también: LEFT$, RIGHT$, LEN, MID$

INT: Vea FIX

INTEGER: Vea Tipos de variables

IOCTL

IOCTL transmite una cadena de control a un controlador de dispositivo. IOCTL$ devuelve la información sobre el estado actual de un controlador de dispositivo.

```
IOCTL [#]numarchivo%, cadena$
IOCTL$([#]numarchivo%)
```

* numarchivo% — El número de un dispositivo abierto.
* cadena$ — La cadena de control que se enviará al dispositivo.
* Las cadenas de control IOCTL y la información producida por IOCTL$ dependen del controlador de dispositivo. Consulte la documentación del controlador de dispositivo para obtener información sobre las cadenas de control y los resultados de la función IOCTL$.

IS: Vea SELECT CASE

JUEGO DE CARACTERES DE QBASIC

El juego de caracteres de Microsoft Basic incluye las letras del alfabeto (A-Z, a-z), caracteres numéricos (0-9 y A-F o a-f para números hexadecimales) y caracteres especiales. Algunos caracteres tienen un significado especial en Basic:

Sufijos indicando el tipo de dato

!	Precisión sencilla	%	Entero
#	Precisión doble	&	Entero largo
$	Cadena		

Operadores matemáticos

*	Multiplicación
-	Menos (restar)
/	División
=	Operador relacional o símbolo de asignación
>	Mayor que
+	Más (sumar)
.	Punto decimal
<	Menor que
\	Símbolo de división de enteros (barra invertida)
^	Exponente (flecha arriba o circumflejo)

Especial

'	Comentario (comilla sencilla)
;	Controla salida de instrucciones PRINT e INPUT
,	Controla salida de instrucciones PRINT e INPUT
:	Separa varias instrucciones en una sola línea
?	indicador de instrucción INPUT
_	Subrayado para continuación de línea (reservado para una compatibilidad con otras versiones de Basic pero no reconocido por QBasic)

KEY (asignación)

Asigna valores de cadena a teclas de función y, si se especifica la opción, presenta los valores de las teclas.

```
KEY tecla%, expresióncadena$
KEY LIST
KEY ON
KEY OFF
```

* tecla% — El número de una tecla de función. Utilice 1 a 10 para las teclas de función F1 a F10. Utilice 30 y 31 para F11 y F12 en teclados extendidos.
* expresióncadena$ — Una expresión de cadena de hasta 15 caracteres generados al presionar la tecla de función.
* LIST — Presenta las asignaciones para cada tecla.
* ON — Activa la línea que muestra las teclas de función.
* OFF — Desactiva la línea que muestra las teclas de funciones.

Ejemplo:

```
KEY 4, "MENU" + CHR$ (13)
KEY LIST
KEY 4, ""
KEY LIST
```

Vea también: KEY, ON KEY (intercepción de eventos)

KEY (intercepción de eventos)

KEY activa, desactiva o suspende la intercepción de eventos de una tecla. Si está activada la intercepción de eventos, ON KEY irá a una subrutina cada vez que se presione la tecla.

```
KEY(n%) ON
KEY(n%) OFF
KEY(n%) STOP
ON KEY(n%) GOSUB línea
```

* n% — Un valor que especifica una tecla de funciones, tecla de dirección o tecla definida por el usuario:

n%	Tecla
0	Todas las teclas indicadas aquí (KEY(0) ON, KEY(0) OFF y KEY(0) STOP solamente)
1-10	Teclas de funciones F1-F10
11	Tecla Flecha Arriba
12	Tecla Flecha Izquierda
13	Tecla Flecha Derecha
14	Tecla Flecha Abajo
15-25	Teclas definidas por el usuario. Para obtener más información al respecto, vea

	Declarar teclas definidas por el usuario.
30, 31	Teclas de funciones F11 y F12.

* KEY(n%) ON — Activa intercepción de eventos para la tecla especificada.
* KEY(n%) OFF — Desactiva la intercepción de eventos para la tecla especificada.
* KEY(n%) STOP — Suspende la intercepción de eventos de teclas. Los eventos serán procesados una vez que se active la intercepción de eventos a través de KEY ON.
* línea — La etiqueta o el número de la primera linea de la subrutina para intercepción de errores.

Ejemplo:

```
'Este ejemplo requiere las teclas Bloq Mayús y Bloq Num desactivadas.
CONST ESC = 27
KEY 15, CHR$(&H4) + CHR$(&H1F)              'Define Ctrl+S como KEY 15.
ON KEY(15) GOSUB PauseHandler
KEY(15) ON
WHILE INKEY$ <> CHR$(ESC)
    PRINT "Presione Esc para parar, Ctrl+S para hacer una pausa."
    PRINT
WEND
END

PauseHandler:
    SLEEP 1
    RETURN
```

Para declarar una tecla definida por el usuario, utilice la siguiente variación de la instrucción KEY:

```
KEY n%, CHR$(banderateclado%) + CHR$(códigoteclado%)
```

- n% — Un valor entre 15 y 25 que identifica la tecla.
- banderateclado% — Uno de los siguientes valores, o una suma de valores, especificando si la tecla definida por el usuario será utilizada en combinación con las teclas Mayús, Ctrl, Alt, BloqNum o BloqMayús, o con teclas extendidas:

Valor	Tecla
0	No hay bandera de teclado
1 through 3	Cualquiera de las teclas Mayús
4	Tecla Ctrl
8	Tecla Alt
32	Tecla BloqNum
64	Tecla BloqMayús
128	Teclas extendidas en un teclado con 101 teclas

Para especificar una combinación de varias teclas, sume los valores. Por ejemplo, el valor de 12 especifica que la tecla definida por el usuario será utilizada en combinación con las teclas Ctrl y Alt.

- códigoexploración% — El código de exploración para tecla declarada

Vea también: KEY (asignación), KEY (intercepción de eventos), ON KEY, Códigos de exploración de teclado

---

KILL

Elimina archivos del disco.

```
KILL archivo$
```

* archivo$ — Identifica el archivo o los archivos que serán eliminados. Puede incluir una ruta de acceso y los comodines ? y * DOS.

Ejemplo:

```
INPUT "Archivo a eliminar: "; f$
KILL f$
```

Vea también: FILES

LBOUND

Devuelven el límite inferior y superior (el subíndice menor o mayor disponible), para la dimensión de matriz especificada.

```
LBOUND(matriz[,dimensión%])
UBOUND(matriz[,dimensión%])

    * matriz        El nombre de la matriz.
    * dimensión%    Indica la dimensión de la matriz cuyo límite inferior
                    o superior será dado como resultado. Utilice 1 para la
                    primera dimensión, 2 para la segunda, etc. El valor
                    predeterminado es 1.

Ejemplo:
    DIM a%(1 TO 3, 2 TO 7)
    PRINT LBOUND(a%, 1), UBOUND(a%, 2)

Vea también: DIM, REDIM
```

LCASE$

```
Convierte cadenas en letras minúsculas o letras mayúsculas.

LCASE$(expresión-cadena$)
UCASE$(expresión-cadena$)

    * expresión-cadena$    Cualquier expresión de cadena.

Ejemplo:
    Test$ = "LA cadena"
    PRINT Test$
    PRINT LCASE$(Test$); " en minúsculas"
    PRINT UCASE$(Test$); " EN MAYUSCULAS"
```

LEFT$

```
Devuelven un número específico de caracteres al extremo izquierdo o derecho
en una cadena.

LEFT$(expresión-cadena$,n%)
RIGHT$(expresión-cadena$,n%)

    ■ expresión-cadena$    Cualquier expresión de cadena.
    ■ n%                   El número de caracteres que devolverá como
                           resultado, comenzando con el último carácter de
                           la izquierda o de la derecha de la cadena.

Ejemplo:
    a$ = "Microsoft QBasic"
    PRINT LEFT$(a$, 5)     'Resultado:  Micro
    PRINT RIGHT$(a$, 5)    'Resultado:  Basic


Vea también: MID$
```

LEN

```
Devuelve el número de caracteres en una cadena, o el número de bytes
requeridos para almacenar una variable.

LEN(expresión-cadena$)
LEN(variable)

    * expresión-cadena$    Cualquier expresión de cadena.
    * variable             Cualquier variable que no sea de cadena.

Ejemplo:
    a$ = "Microsoft QBasic"
    PRINT LEN(a$)

Vea también: OPEN
```

LET

```
Asigna el valor de una expresión a una variable.

[LET] variable=expresión

    * variable      Cualquier variable. Los nombres de variables pueden
                    tener hasta 40 caracteres y deben comenzar con una letra.
                    Los caracteres válidos son A-Z, 0-9 y el punto (.). No se
                    pueden usar letras acentuadas ni la ñ.
    * expresión     Cualquier expresión que produzca un valor que puede ser
```

asignada.
* No es recomendable usar la palabra clave optativa LET. La instrucción de asignación variable=expresión realiza la misma acción con o sin LET.

Vea también: LSET, RSET

LIMITES DEL LENGUAJE

Limites para nombres, cadenas y numeros

	Máximo	Mínimo
Nombre de variable	40 caracteres	1 carácter
Cadena	32,767 caracteres	0 caracteres
Enteros	32,767	-32,768
Enteros largos	2,147,483,647	-2,147,483,648
Números de precisión sencilla:		
Positivo	3.402823E+38	2.802597E-45
Negativo	-2.802597E-45	-3.402823E+38
Números de precisión doble:		
Positivo	1.79769313486231D+308	4.940656458412465D-324
Negativo	-4.940656458412465D-324	-1.79769313486231D+308

Limites para matrices

	Máximo	Mínimo
Tamaño de matriz (todos los elementos):		
Estática	65,535 bytes (64Kb)	1 byte
Dinámica	65,535 bytes (64Kb)	
Número de dimensiones permitidas	60	1
Dimensiones permitidas sin especificar	8	1
Valor de subíndice para matriz	32,767	-32,768

Nota: La variación máxima entre valores de subíndice para matrices es de 32,767.

Limites para procedimientos y archivos

	Máximo	Mínimo
Tamaño de procedimiento	64Kb	0
Número de argumentos pasados	60	0
Números de archivos de datos	255	1
Número de registros en archivo de datos	2,147,483,647	1
Tamaño de registros en archivo de datos	32Kb	1 byte
Tamaño del archivo de datos	Espacio en disco	0
Rutas de acceso	127 caracteres	1 carácter
Números de mensajes de error	255	1

LINE (gráficos)

Traza una línea o rectángulo en la pantalla.

LINE [[STEP](x1!,y1!)]-[STEP](x2!,y2!) [,[color%] [,[B | BF] [,estilo%]]]

* STEP — Especifica que las coordenadas estarán situadas con relación a la posición actual del cursor de gráficos.
* (x1!,y1!), (x2!,y2!) — Las coordenadas de pantalla del principio de la línea y EL final de la línea.
* color% — Un atributo de color que establece el color de la línea o rectángulo. Los atributos de color disponibles dependerán del adaptador de gráficos y del modo de pantalla establecido mediante la más reciente instrucción SCREEN.
* B — Traza un rectángulo en lugar de una línea.
* BF — Traza un cuadro con relleno de color.
* estilo% — Un valor de 16 bits los cualles determinan si se trazan o no los pixeles. Utilizado para trazar líneas punteadas.

Ejemplo:

```
'Este ejemplo requiere un adaptador de gráficos a color.
SCREEN 1
LINE (110, 70)-(190, 120), , B
LINE (0, 0)-(320, 200), 3, , &HFF00
```

Vea también: CIRCLE, INPUT, LINE INPUT, PRESET, PSET, SCREEN,

LINE INPUT: Vea INPUT

LINEA DE COMANDOS

Se pueden escribir estas opciones en la línea de comando DOS siguiendo el comando QBasic:

QBASIC [/B] [/EDITOR] [/G] [/H] [/MBF] [/NOHI] [[/RUN] archivo]

Opción	Descripción
/B	Permite el uso de un monitor compuesto (monocromo) con una tarjeta de gráficos a color. La opción /B presenta QBasic en monocromo si tiene un monitor a color.
/EDITOR	Activa el editor de texto MS-DOS. Se puede abreviar como /ED.
/G	Hace que QBasic actualice una pantalla CGA lo más rápido posible (sólo para máquinas con monitor CGA). Si ve "nieve" (puntos intermitentes en la pantalla) cuando QBasic actualiza la pantalla, significa que su hardware no es compatible con esta opción. Si prefiere una pantalla limpia reinicie QBasic sin la opción /G.
/H	Presenta el número máximo de líneas que su hardware es capaz de producir.
/MBF	Hace que las funciones de conversión de QBasic (CVS, CVD MKS$, MKD$) interpreten los números con formato IEEE como números con el formato Microsoft-Binary.
/NOHI	Permite el uso de un monitor no compatible con alta intensidad. No se debe utilizar con computadoras portátiles "laptop" de Compaq.
archivo	Indica el nombre del archivo que será cargado al iniciar QBasic. Para cargar un archivo creado con GW-BASIC o BASICA, el archivo debe guardarse desde GW-BASIC o BASICA utilizando la opción ,A.
/RUN archivo	Hace que QBasic cargue y ejecute un archivo de un programa antes de presentarlo.

LIST: Vea KEY (asignación)

LOC

Devuelve la posición actual dentro de un archivo.

LOC(numarchivo%)

* numarchivo% El número de un archivo o dispositivo abierto.
* Para archivos binarios, LOC devuelve la posición del último byte leído o escrito.
* Para archivos de acceso aleatorio, LOC devuelve el número del último registro leído o escrito en el archivo.
* Para archivos secuenciales, LOC devuelve la posición actual del byte en el archivo, dividido por 128.

Ejemplo:

```
OPEN "TEST.DAT" FOR RANDOM AS #1
FOR i% = 1 TO 10
   PUT #1, , i%
NEXT i%
SEEK #1, 2
GET #1, , i%
PRINT "Datos: "; i%; " Registro actual:"; LOC(1); " Siguiente:"; SEEK(1)
```

Vea también: EOF, SEEK

LOCATE

LOCATE mueve el cursor en la pantalla a la posición especificada.
CSRLIN devuelve la posición actual de la fila donde se encuentra el cursor.
POS devuelve la posición actual de la columna donde se encuentre el cursor.

LOCATE [fila%] [,[columna%] [,[cursor%] [,inicio% [,fin%]]]]

```
CSRLIN
POS(expresión)
```

* fila% y columna% — El número de la fila y columna a la que se moverá el cursor.
* cursor% — Especifica si el cursor está visible: 0 = invisible, 1 = visible
* inicio% y fin% — Expresiones de enteros entre 0 y 31 que especifican la primera y última línea de exploración del cursor. Podrá cambiar el tamaño del cursor modificando las líneas de exploración.
* expresión — Cualquier expresión.

Ejemplo:

```
CLS
LOCATE 5, 5
MiLin% = CSRLIN
MiCol% = POS(0)
PRINT "Posición 1 (Presione cualquier tecla)"
DO
LOOP WHILE INKEY$ = ""
LOCATE (MiLin% + 2), (MiCol% + 2)
PRINT "Posición 2"
```

LOOK

LOCK limita o impide el acceso a un archivo mediante un proceso de red. UNLOCK cancela las limitaciones impuestas por la última instrucción LOCK.

```
LOCK [#]numarchivo% [,{registro& | [inicio&] TO fin&}]
UNLOCK [#]numarchivo% [,{registro& | [inicio&] TO fin&}]
```

* numarchivo% — El número de un archivo abierto.
* registro& — Para archivos de acceso aleatorio, el número del registro que será bloqueado, con relación al primer registro del archivo. Para archivos binarios, el número del byte que será bloqueado, con relación al primer byte del archivo..
* inicio& y fin& — Los números del primer y último registro o byte en una serie de registros o bytes que serán bloqueados o desbloqueados.

* Para archivos secuenciales, LOCK y UNLOCK afectan a todo el archivo.

Ejemplo:

```
'Este ejemplo sólo funcionará en una red.
OPEN "TEST.DAT" FOR RANDOM AS #1
FOR i% = 1 TO 10
    PUT #1, , i%
NEXT i%
LOCK #1, 2          'Bloquear registro 2.
GET #1, 2, i%
UNLOCK #1, 2        'Desbloquear registro 2.
```

Vea también: OPEN

LOF

Devuelve la longitud de un archivo en bytes.

```
LOF(numarchivo%)
```

* numarchivo% — El número de un archivo abierto.

Ejemplo:

```
INPUT "Escriba el nombre del archivo: "; f$
OPEN f$ FOR BINARY AS #1
PRINT "Longitud del archivo = "; LOF(1)
CLOSE
```

LOG: Vea EXP

LONG: Vea Tipos de variables

LOOP: Vea DO

LPOS

Devuelve el número de caracteres enviados a la impresora desde el último envio de retorno de carro.

```
LPOS(n%)

    * n%    Indica uno de los puertos de impresora:
            0 = LPT1, 1 = LPT1, 2 = LPT2, 3 = LPT3
Ejemplo:
    'Este ejemplo requiere una impresora.
    LPRINT
    FOR i% = 1 TO 20
        LPRINT i%;
        IF LPOS(1) >= 10 THEN LPRINT     'Iniciar una nueva línea.
    NEXT i%
```

LPRINT: Vea PRINT

LPRINT USING: Vea PRINT USING

LSET

LSET y RSET mueven datos, colocándolos en un búfer de archivo de acceso aleatorio (en preparación para una instrucción PUT) y alinean a la izquierda o a la derecha el valor de una variable de cadena. LSET también copia el contenido de una variable de registro en otra.

```
LSET variablecadena$=expresióncadena$
RSET variablecadena$=expresióncadena$
LSET variableregistro1=variableregistro2
```

    * variablecadena$ Cualquier variable de cadena o un campo de un archivo de acceso aleatorio definido en una instrucción FIELD.
    * expresióncadena$ Para LSET, la versión alineada a la izquierda de variablecadena$. Para RSET, la versión alineada a la derecha de variablecadena$.
    * variableregistro1   Variables de registro de cualquier tipo de datos definido por el usuario.
      variableregistro2   Use LSET para asignar una variable de registro de un tipo de datos a otro tipo de datos definido por el usuario.

```
Ejemplo:
    OPEN "FILEDAT.DAT" FOR RANDOM AS #1 LEN = 10
    FIELD #1, 5 AS Ls1$, 5 AS Rs1$
    LSET Ls1$ = "LSET"
    RSET Rs1$ = "RSET"
    PUT #1, 1
    CLOSE #1
    OPEN "FILEDAT.DAT" FOR RANDOM AS #1 LEN = 10
    FIELD #1, 5 AS Ls2$, 5 AS Rs2$
    GET #1, 1
    PRINT "*" + Ls2$ + "*", "*" + Rs2$ + "*"
    CLOSE #1
```

Vea también: FIELD, GET, PUT

LTRIM$

Eliminan los espacios delante o detrás de una cadena.

```
LTRIM$(expresión-cadena$)
RTRIM$(expresión-cadena$)
```

    * expresión-cadena$ Cualquier expresión de cadena.

```
Ejemplo:
    a$ = "    Basic    "
    PRINT "*" + a$ + "*"            'Resultado:  *    Basic    *
    PRINT "*" + LTRIM$(a$) + "*"    'Resultado:  *Basic    *
    PRINT "*" + RTRIM$(a$) + "*"    'Resultado:  *    Basic*
```

MID$

La función MID$ devuelve una porción de una cadena (una subcadena).
La instrucción MID$ reemplaza una porción de una variable de cadena con otra cadena.

```
MID$(expresión-cadena$,inicio%[,longitud%])
MID$(variable-cadena$,inicio%[,longitud%])=expresión-cadena$
```

    * expresión-cadena$ La cadena de la cual función MID$ toma una

subcadena, o la cadena de reemplazo utilizada por la instrucción MID$. Puede ser cualquier expresión de cadena.

* inicio% — La posición del primer carácter de la subcadena que será devuelta o reemplazada.
* longitud% — El número de caracteres en la subcadena. Si se omite, MID$ devolverá o reemplazará todos los caracteres situados a la derecha de la posición de inicio.
* variable-cadena$ — La variable de cadena que será modificada por la instrucción MID$.

Ejemplo:

```
a$ = "¿Dónde está Paris?"
PRINT MID$(a$, 10, 5)       'Resultado:  Paris
Texto$ = "Paris, Francia"
PRINT Texto$                 'Resultado:  Paris, Francia
MID$(Texto$, 8) = "Texas "
PRINT Texot$                 'Resultado:  Paris, Texas
```

Vea también: LEFT$, RIGHT$, LEN

MKD$: Vea CVD

MKDIR: Vea CHDIR

MKDMBF$: Vea CVDMBF

MKI$, MKL$, MKS$: Vea CVD

MKSMBF$: Vea CVDMBF

MOD

Divide un número por otro y devuelve el residuo.

expresión-numérica1 MOD expresión-numérica2

* expresión-numérica1, expresión-numérica2 — Cualquier expresión numérica. Los números reales serán redondeados a enteros.

Ejemplo:

```
PRINT 19 MOD 6.7    'QBasic redondea 6.7 a 7, luego hace la división.
                    'Resultado:  5
```

MODOS DE PANTALLA

La tabla siguiente presenta un resumen de los modos de pantalla:

Adaptadores MDPA, CGA, Hércules, Olivetti, EGA, VGA o MCGA
SCREEN 0: Modo de texto solamente

- Formato de texto 40 x 25, 40 x 43, 40 x 50, 80 x 25, 80 x 43 u 80 x 50, cuadro de carácter 8 x 8 (8 x 14, 9 x 14 ó 9 x 16 con EGA o VGA)
- 16 colores asignados a cualquiera de 16 atributos (con CGA o EGA)
- 64 colores asignados a cualquiera de 16 atributos (con EGA o VGA)
- Según la resolución de texto y el adaptador, 8 páginas de memoria de video (0-7), 4 páginas (0-3), 2 páginas (0-1) o 1 página (0)

Adaptadores CGA, EGA, VGA o MCGA
SCREEN 1: Gráficos 320 x 200

- Formato de texto 40 x 25, cuadro de carácter 8 x 8
- 16 colores de fondo y uno de dos juegos de 3 colores de primer plano asignado a través de una instrucción COLOR con CGA
- 16 colores asignados a 4 atributos con EGA o VGA
- 1 página de memoria de video (0)

SCREEN 2: Gráficos 640 x 200

- Formato de texto 80 x 25, cuadro de carácter 8 x 8
- 16 colores asignados a 2 atributos con EGA o VGA
- 1 página de memoria de video (0)

Adaptadores Hercules, Olivetti o AT&T
SCREEN 3: Se requiere adaptador Hercules, sólo para monitores monocromáticos

- Gráficos 720 x 348
- Formato de texto 80 x 25, cuadro de carácter 9 x 14
- Normalmente 2 páginas de memoria de video (0-1); 1 página (0) si está instalado un segundo adaptador de pantalla
- Incompatible con la instrucción PALETTE
- Invoque el controlador Hercules MSHERC.COM antes de usar el modo de pantalla 3

SCREEN 4:

- Compatible con computadoras personales Olivetti, modelos M24, M240, M28, M280, M380, M380/C y M380/T y con la serie 6300 de computadoras personales AT&T
- Gráficos 640 x 400
- Formato de texto 80 x 25, cuadro de carácter 8 x 16
- 1 de 16 colores asignado como color de primer plano (seleccionado mediante la instrucción COLOR); el color del fondo es negro fijo
- 1 página de memoria de video (0)
- Incompatible con la instrucción PALETTE

Adaptadores EGA o VGA
SCREEN 7: Gráficos 320 x 200
- Formato de texto 40 x 25, cuadro de carácter 8 x 8
- Asignación de 16 colores a cualquiera de 16 atributos
- Si hay 64Kb de memoria en adaptador EGA, 2 páginas de memoria de video (0-1); de lo contrario, 8 páginas (0-7)

SCREEN 8: Gráficos 640 x 200
- Formato de texto 80 x 25, cuadro de carácter 8 x 8
- Asignación de 16 colores a cualquiera de 16 atributos
- Si hay 64Kb de memoria en adaptador EGA, 1 página de memoria de video (0); de lo contrario, 4 páginas (0-3)

SCREEN 9: Gráficos 640 x 350
- Formato de texto 80 x 25 u 80 x 43, cuadro de carácter 8 x 14 u 8 x 8
- 16 colores asignados a 4 atributos (memoria de adaptador de 64Kb), o 64 colores asignados a 16 atributos (más de 64Kb de memoria de adaptador)
- Si hay 64Kb de memoria en adaptador EGA, 1 página de memoria de video (0); de lo contrario, 2 páginas (0-1)

Adaptadores EGA o VGA, Monitor monocromático solamente
SCREEN 10: Gráficos 640 x 350, monitor monocromático solamente
- Formato de texto 80 x 25 u 80 x 43, cuadro de carácter 8 x 14 u 8 x 8
- Hasta 9 seudocolores asignados a 4 atributos
- 2 páginas de memoria de video (0-1), se requieren 256Kb de memoria de adaptador

Adaptadores VGA o MCGA
SCREEN 11 (VGA o MCGA)
- Gráficos 640 x 480
- Formato de texto 80 x 30 u 80 x 60, cuadro de carácter 8 x 16 u 8 x 8
- Asignación de hasta 256Kb de colores a 2 atributos
- 1 página de memoria de video (0)

SCREEN 12 (VGA)
- Gráficos 640 x 480 graphics
- Formato de texto 80 x 30 u 80 x 60, cuadro de carácter 8 x 16 u 8 x 8
- Asignación de hasta 256Kb de colores a 16 atributos
- 1 página de memoria de video(0)

SCREEN 13 (VGA o MCGA)
- Gráficos 320 x 200
- Formato de texto 40 x 25, cuadro de carácter 8 x 8
- Asignación de hasta 256Kb de colores a 256 atributos
- 1 página de memoria de video(0)

Vea también: SCREEN

NAME

Cambia el nombre de un archivo o directorio.

```
NAME viejo$ AS nuevo$
```

* viejo$ y nuevo$	El nombre de un archivo existente y el nombre nuevo del archivo. Cada nombre puede incluir una ruta de acceso.

Ejemplo:
```
INPUT "Nombre existente: "; ViejoFN$
INPUT "Nombre nuevo: "; NuevoFN$
NAME ViejoFN$ AS NuevoFN$
```

Vea también: FILES

NEXT

Incrementa y verifica el contador en un bucle FOR...NEXT o, cuando se use con RESUME, continúa la ejecución desde un identificador de intercepción de errores.

Vea también: FOR, RESUME

NOT: Vea AND

OCT$: Vea HEX$

OFF

Desactiva la presentación de asignaciones de teclas de funciones cuando se utilice con la instrucción KEY (Asignación), o desactiva la intercepción de eventos cuando se utilice con instrucciones del tipo "evento OFF" (COM OFF, KEY OFF, PEN OFF, PLAY OFF, STRIG OFF y TIMER OFF).

Vea también: COM, ON COM, KEY, ON KEY, KEY (asignación), PEN, ON PEN, PLAY, ON PLAY, STRIG, ON STRIG, TIMER, ON TIMER

ON COM

COM activa, desactiva o suspende la intercepción de eventos en un puerto de comunicaciones. Si está activada la intercepción de eventos, ON COM bifurcará a una subrutina cada vez que se reciban caracteres en el puerto.

```
COM(n%) ON
COM(n%) OFF
COM(n%) STOP
ON COM(n%) GOSUB línea
```

* n% — El número de un puerto COM (en serie) (1 ó 2).
* COM(n%) ON — Activa la intercepción de eventos de comunicación.
* COM(n%) OFF — Desactiva la intercepción de eventos de comunicación.
* COM(n%) STOP — Suspende la intercepción de eventos de comunicación. Los eventos serán procesados una vez que se active la intercepción de eventos a través de COM ON.
* línea — La etiqueta o el número de la primera línea de la subrutina para intercepción de eventos.

Ejemplo:

```
    COM(1) ON       'Activa intercepción de eventos en puerto 1.
    ON COM(1) GOSUB ComHandler
    DO: LOOP WHILE INKEY$ = ""
    COM(1) OFF
    END

ComHandler:
        PRINT "Se ha escrito algo en el terminal conectado a COM1."
        RETURN
```

Vea también: OPEN COM

ON ERROR

Activa el identificador de errores, y cuando ocurra un error de ejecución, envia el programa a una rutina de identificador de errores o a que reanude su ejecución.

```
ON ERROR {GOTO línea | RESUME NEXT}
```

* GOTO línea — Se bifurca a la primera línea de la rutina de manejo de errores, especificado por una etiqueta o número de línea. Para desactivar el manejo de errores, especifique GOTO 0.
* RESUME NEXT — Reanuda la ejecución a partir de la instrucción que de la instrucción que causó el error de ejecución. Use la función ERR para obtener el código del error.
* Si no se utiliza ON ERROR, cualquier error de ejecución terminará el programa.

Vea también: ERDEV, ERDEV$, ERR, ERL, ERROR, GOTO, RESUME

ON

Realiza acciones diferentes como parte de varias instrucciones:

* En la instrucción ON ERROR, activa la intercepción de errores.
* En instrucciones del tipo "evento ON" (COM ON, KEY ON, PEN ON, PLAY ON, STRIG ON y TIMER), activa la intercepción de eventos.
* En instrucciones del tipo "ON evento" (ON COM, ON KEY, ON PEN, ON PLAY, ON STRING y ON TIMER), especifica un evento que será interceptado.
* En las instrucciones ON...GOSUB y ON...GOTO, especifica una expresión que será evaluada.

Vea también: COM, ON COM, KEY, ON KEY, ON ERROR, ON...GOSUB, ON...GOTO, PEN,

---

ON KEY: Vea KEY (intercepción de eventos)

ON PEN: Vea PEN (intercepción de eventos)

ON PLAY: Vea PLAY (Intercepción de eventos)

ON STRIG: Vea STRIG (intercepción de eventos)

ON TIMER: Vea TIMER (intercepción de eventos)

ON GOSUB, ON GOTO

Bifurca a una de varias posiciones, según el valor de una expresión.

```
ON expresión% GOSUB lista-líneas
ON expresión% GOTO lista-líneas
```

* expresión% Una expresión entre 0 y 255.
* lista-líneas Una serie de etiquetas o números de línea. Si el valor de la expresión es 1, el programa se bifurca a la primera línea de la lista; si la expresión es 2, se bifurca a la segunda línea, etc.
* SELECT CASE ofrece una mejor forma de realizar bifurcaciones múltiples.

Ejemplo:

```
    FOR i% = 1 TO 2
        ON i% GOSUB Uno, Dos
    NEXT i%
    END

Uno: PRINT "Uno"
    RETURN
Dos: PRINT "Dos"
    RETURN
```

Vea también: ON, SELECT CASE

OPEN

Abre un archivo o dispositivo.

```
OPEN archiv$ [FOR modo] [ACCESS acceso] [bloqueo] AS [#]numarch% [LEN=longreg
```

* archiv$ El nombre del archivo o dispositivo. El nombre puede incluir una unidad de disco y ruta de acceso.
* modo Uno de los siguientes modos de archivo: APPEND, BINARY INPUT, OUTPUT o RANDOM. INPUT, OUTPUT y RANDOM también se utilizan en la instrucción OPEN COM.
  - APPEND especifica que el archivo será abierto para dar información de salida secuencial y coloca el puntero de archivo al final del archivo. Una instrucción PRINT # o WRITE # luego anexa información al archivo.
  - BINARY especifica el modo de archivo binario. En este modo, es posible leer o escribir información en cualquier posición de byte del archivo usando instrucciones GET o PUT.
  - INPUT especifica que el archivo será abierto para recibir información de entrada secuencial.
  - OUTPUT especifica que el archivo será abierto para dar información de salida secuencial.
  - RANDOM especifica que el archivo será abierto en el modo de acceso aleatorio. RANDOM es el modo de archivo predeterminado.
* acceso En una red, especifica si el archivo será abierto con el tipo de acceso READ, WRITE o READ WRITE. ACCESS {READ | WRITE | READ WRITE}
  - READ Abre un archivo para lectura solamente.
  - WRITE Abre un archivo para escritura solamente.
  - READ WRITE Abre un archivo para lectura y escritura. El modo READ WRITE es válido sólo para archivos que tengan el modo binario o de acceso aleatorio, y para archivos abiertos con el modo APPEND (acceso secuencial).
* bloqueo Especifica el estado de bloqueo de archivos en una red: SHARED, LOCK READ, LOCK WRITE, LOCK READ WRITE.
* numarch% Un número entre 1 y 255 que identifica el archivo mientras está abierto.
* longreg% Para archivos de acceso aleatorio, la longitud de

registro (el valor predeterminado es 128 bytes). Para archivos secuenciales, el número de caracteres en búfer (el valor predeterminado es 512 bytes).

Ejemplo:

```
INPUT "Escriba el nombre del archivo: "; n$
OPEN n$ FOR OUTPUT AS #1
PRINT #1, "Esto se guarda en el archivo."
CLOSE
OPEN n$ FOR INPUT AS #1
INPUT #1, a$
PRINT "Leer del archivo: "; a$
CLOSE
```

Sintaxis diferente para OPEN:

OPEN modo2$,[#]numarch%,archivo$[,longreg%]

* modo2$ Una expresión de cadena que comienza con uno de los caracteres siguientes y especifica el modo del archivo:
  - O Modo de salida secuencial
  - I Modo de entrada secuencial
  - R Modo de entrada/salida para archivos de acceso aleatorio
  - B Modo de archivo binario
  - A Modo de salida secuencial. Fija el puntero de archivo al final del archivo y el número de registro en el número del último registro del archivo. Una instrucción PRINT # o WRITE # extiende (anexa) información al archivo.
* numarch% Un número entre 1 y 255 que identifica el archivo mientras está abierto.
* archivo$ El nombre del archivo (puede incluir la unidad de disquete y ruta de acceso).
* longreg% Para archivos de acceso aleatorio, la longitud del registro en bytes. Para archivos secuenciales, el número de caracteres en el búfer.
* QBasic reconoce esta sintaxis para ser compatible con programas escritos en versiones anteriores de Basic.

Vea también: CLOSE, FREEFILE, OPEN COM, TYPE, Sintaxis diferente para OPEN, PRINT, LPRINT, WRITE

OPEN COM

Abre e inicializa un canal de comunicación es para información de entrada y salida (E/S). Debe ejecutar la instrucción OPEN COM antes de usar un dispositivo para comunicaciones que utilice una interfaz RS232.

OPEN "COMn: listaop1 listaop2" [FOR modo] AS [#]numarch% [LEN=reclen%]

* n El puerto de comunicaciones que será abierto (1 = COM1, 2 = COM2).
* listaop1 Los parámetros de comunicación que se utilizan con mayor frecuencia:
  [baudios] [,[paridad] [,[datos] [,[paro]]]]
  baudios es la velocidad de transmisión en baudios para el dispositivo que será abierto:
  75, 110, 150, 300, 600, 1200, 2400, 4800, 9600
  paridad es el método para verificar la paridad:
  N (ninguno) E (par) O (impar)
  S (espacio) M (marca) PE (activar verif/errores
  datos es el número de bits de datos por byte:
  5, 6, 7, 8
  paro es el número de bits de paro:
  1, 1.5, 2
  Los valores predeterminados son: baudios 300, paridad par, 7 bits de datos, 1 bit de paro.
* listaop2 Lista de parámetros usados con menor frecuencia, separados con comas:

Opción	Descripción
ASC	Abre el dispositivo en modo ASCII.
BIN	Abre el dispositivo en modo binario.
CD[m]	Define el tiempo de espera (en milisegundos) en la línea Data Carrier Detect (DCD).
CS[m]	Define el tiempo de espera (en milisegundos) en la línea Clear to Send (CTS).
DS[m]	Define el tiempo de espera (en milisegundos) en la línea Data Set Ready (DS).

LF	Envía un carácter de salto de línea después de un retorno de carro.
OP[m]	Especifica cuánto tiempo (en milisegundos) OPEN COM esperará la apertura de todas las líneas de comunicaciones.
RB[n]	Define el tamaño (en bytes) del búfer receptor.
RS	Suprime la detección de Petición de Envio (RTS).
TB[n]	Define el tamaño (en bytes) del búfer de transmisión.

* modo — INPUT, OUTPUT o RANDOM (predeterminado). Vea Modos de archivo para OPEN.
* numarch% — Un número entre 1 y 255 que identifica el canal de comunicaciones mientras esté abierto.
* reclen% — Tamaño del búfer de modo de acceso aleatorio (el valor predeterminado es de 128 bytes).

Ejemplo:

```
'Use este ejemplo para detectar errores de comunicaciones en serie.
'Velocidad de transmisión lenta, handshaking de hardware desactivado,
'búferes agrandados.
OPEN "COM1:300,N,8,1,CD0,CS0,DS0,OP0,RS,TB2048,RB2048" FOR RANDOM AS #1
```

Vea también: OPEN

OPTION BASE

Establece el límite inferior predeterminado para subíndices de matrices.

OPTION BASE {0 | 1}

* La instrucción DIM con la claúsula TO constituye una mejor forma de definir el límite inferior de un subíndice de matriz.

Vea también: DIM, REDIM, LBOUND, UBOUND

OR: Vea AND

OUT: Vea INP

OUTPUT: Vea OPEN

PAINT

Rellena una área con el color o el diseño especificado.

PAINT [STEP] (x!,y!)[,[{color% | diseño$}] [,[colorbordes%] [,fondo$]]]

* STEP	Especifica que las coordenadas se situen con relación a la posición actual del cursor de gráficos.
* (x!,y!)	Las coordenadas de pantalla donde se iniciará el relleno de color.
* color%	Un atributo de color que establece el color de relleno.
* diseño$	Un diseño de relleno con un ancho de 8 bits y una longitud de hasta 64 bytes, definido así: diseño$ = CHR$(arg1) + CHR$(arg2) + ... + CHR$(argn%) Los argumentos de CHR$ son números entre 0 y 255. Cada CHR$(argn%) define un sector del diseño basado en la forma binaria del número.
* colorbordes%	Un atributo de color que especifica el color de los bordes del área rellena. PAINT deja de rellenar el área cuando encuentre un borde del color especificado.
* fondo$	Un sector del diseño de fondo de 1 byte, 8 pixeles. Al especificar un sector del diseño de fondo, podrá rellenar encima de una área previamente rellenada.

* Los atributos de color disponibles dependerán del adaptador de gráficos y del modo de pantalla establecido mediante la última instrucción SCREEN.

Ejemplo:

```
'Este ejemplo requiere un adaptador de gráficos a color.
SCREEN 1
CIRCLE (106, 100), 75, 1
LINE (138, 35)-(288, 165), 1, B
PAINT (160, 100), 2, 1
```

Vea también: ASC, CHR$, CIRCLE, DRAW, LINE, SCREEN, Modos de pantalla, Atributos y valores de color

---

PALETTE

Cambian las asignaciones de color correspondientes a los atributos de color en el modo de pantalla actual. PALETTE y PALETTE USING funcionan únicamente en sistemas que utilicen adaptadores EGA, VGA o MCGA.

PALETTE [atributo%,color&]
PALETTE USING matriz#[(índice%)]

* atributo% El atributo de color que será cambiado.
* color& Un valor de color que será asignado a un atributo.
* matriz# Una matriz de valores de color que será asignada al juego de atributos del modo de pantalla actual. La matriz debe ser suficientemente grande para asignar colores a todos los atributos.
* índice% El índice del primer elemento de la matriz que será asignado a un atributo.
* Los atributos y valores de color disponibles dependerán del adapatador de gráficos y del modo de pantalla actual establecido por la más reciente instrucción SCREEN.

Ejemplo:

```
'Este ejemplo requiere un adaptador de gráficos a color.
PALETTE 0, 1
SCREEN 1
FOR i% = 0 TO 3: a%(i%) = i%: NEXT i%
LINE (138, 35)-(288, 165), 3, BF
LINE (20, 10)-(160, 100), 2, BF
DO
    FOR i% = 0 TO 3
        a%(i%) = (a%(i%) + 1) MOD 16
    NEXT i%
    PALETTE USING a%(0)
LOOP WHILE INKEY$ = ""
```

Vea también: COLOR, SCREEN, Atributos y valores de color, Modos de pantalla

---

PCOPY

Copia una página de memoria de video en otra.

PCOPY páginaorigen%,páginadestino%

* páginaorigen% El número de la página de memoria de video que será copiada.
* páginadestino% El número de la página de memoria de video en la que será insertada la copia.
* El valor que identifica la página de video es determinado por el tamaño de la memoria de video y el modo de pantalla actual.

Ejemplo:

```
PCOPY 1, 3
```

Vea también: SCREEN, Modos de pantalla

PEEK

PEEK devuelve un valor de byte almacenado en una posición de memoria específica.
POKE escribe un valor de byte en una posición de memoria específica.

PEEK(dirección)
POKE dirección,byte%

* dirección Una posición de byte relativa a la dirección del segmento actual establecida mediante DEF SEG; un valor entre 0 y 65,535.
* byte% Un valor de byte que será escrito en la dirección de memoria especificada; un valor entre 0 y 255.

Ejemplo:

```
DEF SEG = 0
estado% = PEEK(&H417)              'Lee el estado del teclado.
POKE &H417, (estado% XOR &H40)     'Cambia estado de Bloq Mayús, bit 6.
```

Vea también: DEF SEG

PEN (función)

Devuelve el estado de un lápiz fotosensible.

PEN(n%)

* n% Especifica la información que será devuelta acerca del estado del lápiz fotosensible:

n%	Da como resultado
0	Si se ha utilizado el lápiz desde el último llamado (0 si No, -1 si Sí)
1	La coordenada de pantalla "x" del último contacto de lápiz
2	La coordenada de pantalla "y" del último contacto de lápiz
3	El estado actual de intercambio de lápiz (-1 si abajo, 0 si arriba)
4	La coordenada de pantalla "x" del último lugar donde el lápiz rompió contacto con la pantalla
5	La coordenada de pantalla "y" del último lugar donde el lápiz rompió contacto con la pantalla
6	La fila (caracteres) del último contacto de lápiz
7	La columna (caracteres) del último contacto de lápiz
8	La fila (caracteres) del último lugar donde el lápiz rompió contacto con la pantalla
9	La columna (caracteres) del último lugar donde el lápiz rompió contacto con la pantalla

Ejemplo:
```
DO
  P = PEN(3)
  LOCATE 1, 1: PRINT "El lápiz está ";
  IF P THEN PRINT "abajo" ELSE PRINT "arriba  "
  PRINT "X ="; PEN(4), " Y ="; PEN(5); "  "
LOOP
```

Vea también: PEN, ON PEN, instrucciones, SCREEN, Modos de pantalla

---

PEN (intercepción de eventos)

PEN activa, desactiva o suspende la intercepción de eventos de lápiz fotosensible. Si está activada la intercepción de eventos, ON PEN se bifurca a una subrutina cada vez que se active el lápiz fotosensible.

```
PEN ON
PEN OFF
PEN STOP
ON PEN GOSUB línea
```

* PEN ON Activa la intercepción de eventos de lápiz fotosensible.
* PEN OFF Desactiva la intercepción de eventos de lápiz fotosensible.
* PEN STOP Suspende la intercepción de eventos de lápiz fotosensible. Los eventos serán procesados una vez que se active la de errores usando PEN ON.
* línea La etiqueta o el número de la primera línea de la subrutina para intercepción de eventos.

Ejemplo:
```
'Este ejemplo requiere un lápiz fotosensible.
ON PEN GOSUB Identificador
PEN ON
PRINT "Presione Esc para salir."
DO UNTIL INKEY$ = CHR$(27): LOOP
END

Identificador:
PRINT "El lápiz está en la fila"; PEN(6); ", columna"; PEN(7)
RETURN
```

Vea también: PEN (función)

PLAY (función)

Devuelve el número de notas en la cola de fondo musical.

PLAY (n)

* n Cualquier expresión numérica.

Ejemplo:
```
Música$ = "MBT180o2P2P8L8GGGL2E-P24P8L8FFFL2D"
PLAY Música$
WHILE PLAY(0) > 5: WEND
```

```
    PRINT "¡Casi terminado!"
```

Vea también: PLAY (música), ON PLAY (intercepción de eventos)

PLAY (música)

Toca notas musicales.

```
PLAY cadenacomando$
```

    * cadenacomando$ Una expresión de cadena que contiene uno o más
                        de los siguientes comandos PLAY:

      Comandos de octava y tono:
        Ooctava   Establece la octava actual (0 - 6).
        < o >   Va arriba o abajo una octava.
        A - G   Toca la nota especificada en la octava actual.
        Nnota   Toca la nota especificada (0 - 84) en la gama de
                   siete octavas (0 es pausa).
      Comandos de duración y tempo:
        Lduración Define la duración de cada nota (1 - 64).
                   L1 es redonda, L2 es blanca, etc..
        ML   Música legato.
        MN   Música normal.
        MS   Música staccato.
        Ppausa   Hace una pausa (1 - 64). P1 es una pausa de redonda,
                   P2 es una pausa de blanca.
        Ttempo   Establece el tempo en negras por minuto (32 - 255).
      Comandos de modo:
        MF   Toca la música en primer plano.
        MB   Toca la música de fondo.
      Comandos de sufijo:
        # o +   Eleva la nota anterior a sostenida.
        -           Baja la nota anterior a bemol.
        .   Toca la nota anterior 3/2 en la duración especificada.
    * Para ejecutar una subcadena del comando PLAY desde una cadena de
      comando PLAY, utilice el comando "X":

      PLAY "X" + VARPTR$(cadenacomando$)

Ejemplo:

```
    'Tocar la escala en 7 octavas diferentes
    escala$ = "CDEFGAB"
    PLAY "L16"
    FOR i% = 0 TO 6
        PLAY "O" + STR$(i%)
        PLAY "X" + VARPTR$(escala$)
    NEXT i%
```

Vea también: BEEP, PLAY (función), ON PLAY (intercepción de eventos),
SOUND, VARPTR$

PLAY (intercepción de eventos)

PLAY activa, desesactiva o suspende la intercepción de eventos de PLAY.
Si está activada la intercepción de eventos, ON PLAY irá a una subrutina
cada vez que el búfer de música contenga menos notas que las especificadas.

```
PLAY ON
PLAY OFF
PLAY STOP
ON PLAY(límitecola%) GOSUB línea
```

    * PLAY ON   Activa la intercepción de eventos de PLAY.
    * PLAY OFF   Desactiva la intercepción de eventos de PLAY.
    * PLAY STOP   Suspende la intercepción de eventos de PLAY. Los
                     eventos serán procesados una vez que se active la
                     intercepción de eventos a través de PLAY ON.
    * límitecola%   Un número entre 1 y 32. ON PLAY bifurcará a una
                     subrutina cuando hayan menos notas que límitecola% en
                     el búfer de música.
    * línea   La etiqueta o el número de la primera línea de la
                     subrutina para intercepción de eventos.

Ejemplo:

```
    ON PLAY(3) GOSUB Fondo
    PLAY ON
    Música$ = "MBo3L8ED+ED+Eo2Bo3DCL2o2A"
    PLAY Música$
    LOCATE 2, 1: PRINT "Presione cualquier tecla para parar.";
```

```
    DO WHILE INKEY$ = "": LOOP
    END

    Fondo:
        i% = i% + 1
        LOCATE 1, 1: PRINT "Fondo llamado "; i%; "veces";
        PLAY Música$
        RETURN
```

Vea también: PLAY (música), PLAY (función)

PMAP

Devuelve la coordenada de ventana lógica equivalente a una coordenada de ventana física, según la definición en la instrucción WINDOW, o viceversa.

PMAP (coordenadainicio#, n%)

* coordenadainicio# Una coordenada de ventana lógica o un marco de visualización.
* n% Un valor indicando la coordenada que será devuelta como resultado:

coordenadainicio#	n%	Da como resulado
Coordenada x lógica	0	Coordenada x física
Coordenada y lógica	1	Coordenada y física
Coordenada x física	2	Coordenada x lógica
Coordenada y física	3	Coordenada y lógica

Ejemplo:

```
    'Este ejemplo requiere un adaptador de gráficos que sea compatible
    'con el modo de pantalla 1.
    SCREEN 1
    WINDOW SCREEN (0, 0)-(100, 100)
    PRINT "x lógica=50, x física="; PMAP(50, 0)
    PRINT "y lógica=50, y física="; PMAP(50, 1)
```

Vea también: POINT, VIEW, WINDOW

POINT

Devuelve las coordenadas actuales del cursor de gráficos o el atributo de color del pixel especificado.

POINT {(n%) | (x%,y%)}

* (n%) Indica el tipo de coordenadas que serán devueltas como resultado:

n%	Da como resultado
0	La coordenada x física actual (marco de visualización)
1	La coordenada y física actual (marco de visualización)
2	La coordenada x lógica actual
3	La coordenada y lógica actual

* (x%,y%) Las coordenadas del pixel cuyos colores serán verificados por POINT.
  Si las coordenadas están fuera de la ventana actual, POINT regresará como resultado -1.

Ejemplo:

```
    'Este ejemplo requiere un adaptador de gráficos a color.
    SCREEN 1
    LINE (0, 0)-(100, 100), 2
    LOCATE 14, 1
    FOR y% = 1 TO 10
        FOR x% = 1 TO 10
            PRINT POINT(x%, y%);
        NEXT x%
        PRINT
    NEXT y%
```

Vea también: COLOR, PMAP, SCREEN, VIEW, WINDOW, Atributos y valores de color

---

POKE: Vea PEEK

POS: Vea LOCATE

PRESET

Trazan el punto especificado en la pantalla.

```
PRESET [STEP] (x!,y!) [,color%]
PSET [STEP] (x!,y!) [,color%]
```

* STEP — Especifica que x! e y! serán expresados con relación a la posición actual del cursor de gráficos.
* (x!,y!) — Las coordenadas de pantalla del pixel que será establecido.
* color% — Un atributo de color que establece el color del pixel. Si se omite color%, PRESET usará el color de fondo actual y PSET usará el color de primer plano actual.
* Los atributos de color disponibles dependerán del adaptador de gráficos y del modo de pantalla. Los valores de las coordenadas dependerán del adaptador de gráficos, el modo de pantalla y las instrucciones VIEW y WINDOW más recientes.

Ejemplo:

```
'Este ejemplo requiere un adaptador de gráficos a color.
SCREEN 1
FOR i% = 0 TO 320
    PSET (i%, 100)
    FOR delay% = 1 TO 100: NEXT delay%
    PRESET (i%, 100)
NEXT i%
```

Vea también: SCREEN, VIEW, WINDOW, Atributos y valores de color, Modos de pantalla

---

PRINT

PRINT escribe datos en la pantalla o en un archivo.
LPRINT imprime los datos en la impresora LPT1.

```
PRINT [#numarchivo%,] [listaexpresiones] [{; | ,}]
LPRINT [listaexpresiones] [{; | ,}]
```

* numarchivo% — El número de un archivo abierto. Si no especifica un número de archivo, PRINT escribirá la información en la pantalla.
* listaexpresiones — Una lista de una o más expresiones numéricas o de cadena que serán impresas.
* {; | ,} — Determina el lugar dónde empieza la información de salida siguiente:
  - ; imprimirá inmediatamente después del último valor.
  - , imprimirá al inicio de la siguiente zona de impresión. Las zonas de impresión tienen un ancho de 14 caracteres.

Ejemplo:

```
OPEN "TEST.DAT" FOR OUTPUT AS #1
PRINT #1, USING "##.###  "; 12.12345
CLOSE
OPEN "TEST.DAT" FOR INPUT AS #1
INPUT #1, a$
PRINT a$
LPRINT "Esta es una línea"; 1
LPRINT "Esta es una línea",
LPRINT 2
```

Vea también: PRINT USING, LPRINT USING, WIDTH, WRITE

PRINT USING

PRINT USING escribe información de salida con formato en la pantalla o en un archivo.
LPRINT USING imprime información de salida con formato en la impresora LPT1.

```
PRINT [#numarchivo%,] USING cadenaformato$; listaexpresiones [{; | ,}]
LPRINT USING cadenaformato$; listaexpresiones [{; | ,}]
```

* numarchivo% — El número de un archivo secuencial abierto.
* formatocadena$; — Una expresión de cadena que contiene uno o más especificadores de formato.
* listaexpresiones — Una lista de una o más expresiones numéricas o de cadena que serán impresas; separadas con comas, punto y comas, espacios o tabulaciones.
* {; | ,} — Determina dónde se iniciará la siguiente

información de salida:
; imprimirá inmediatamente después del último valor.
, imprimirá al inicio de la siguiente zona de impresión. Las zonas de impresión tienen un ancho de 14 caracteres.

* Caracteres que dan formato a una expresión numérica

#	Posición de dígito.	-	Después del dígito, imprime el signo para los números negativos.
.	Posición de punto decimal.		
,	A la izquierda del punto decimal, imprime una coma cada 3 dígitos.	$$	Imprime $ adelante.
+	Posición del signo de número.	**	Llena espacios adelante con *.
^^^^	Imprime con formato exponencial.	**$	Combina ** y $$.

* Caracteres que dan formato a una expresión de cadena

&	Imprime la cadena completa.	\ \	Imprime los primeros n caracteres, donde n es el número de espacios entre barras + 2.
!	Imprime sólo el primer carácter de la cadena.		

* Caracteres utilizados para imprimir caracteres literales

_	Imprime el carácter de formato siguiente como literal.		Cualquier carácter que no esté en la tabla será impreso como literal.

Ejemplo:
```
a = 123.4567
PRINT USING "###.##"; a
LPRINT USING "+###.####"; a
a$ = "ABCDEFG"
PRINT USING "!"; a$
LPRINT USING "\ \"; a$
```

Vea también: PRINT, LPRINT, WIDTH

PSET: Vea PRESET

PUT (archivos): Vea GET (archivos)

PUT (gráficos): Vea GET (gráficos)

RANDOM: Vea OPEN

RANDOMIZE

RANDOMIZE inicializa el generador de números aleatorios.
RND devuelve un número aleatorio de precisión sencilla entre 0 y 1.

```
RANDOMIZE (semilla%)
RND[(n#)]
```

* semilla% Un número utilizado para inicializar el generador de números aleatorios.
Si se omite, RANDOMIZE le pedirá dicho argumento.
* n# Un valor que especifica la forma en que RND generará el siguiente número aleatorio:

n#	RND devuelve
Menor que 0	El mismo número para cualquier n#
Mayor que 0 (u omitido)	El siguiente número aleatorio
0	El último número generado

Ejemplo:
```
RANDOMIZE TIMER
x% = INT(RND * 6) + 1
y% = INT(RND * 6) + 1
PRINT "Tirar dos dados: dado 1 ="; x%; "y dado 2 ="; y%
```

READ: Vea DATA

REDIM: Vea DIM

REM

Le permite insertar comentarios y explicaciones en un programa.

REM comentario
' comentario

* comentario Cualquier texto.
* Se hará caso omiso a comentarios al ejecutar el programa a menos que contengan metacomandos.
* Se puede insertar un comentario en la misma línea después de una instrucción ejecutable si va precedido de una forma de REM que utilice la comilla sencilla (') o si REM va precedida de dos puntos (:).

Ejemplo:

```
REM    Este es un comentario.
'      Este también es un comentario.
PRINT "Prueba1"       'Un comentario después de la instrucción PRINT.
PRINT "Prueba2":   REM Otro comentario después de la instrucción PRINT.
```

Vea también: $STATIC, $DYNAMIC

RESET

Cierra todos los archivos y dispositivos abiertos.

RESET

Vea también: CLOSE, END, OPEN, STOP

RESTORE: Vea DATA

RESUME

Reanuda la ejecución del programa después de una rutina de identificación de errores.

RESUME [{línea | NEXT}]

* línea La etiqueta o el número de la línea donde se reanudará la ejecución. Si línea es 0 o se omite, la ejecución se reanudará a partir de la instrucción que causó el error.
* NEXT Reanuda la ejecución a partir de la instrucción que sigue a la instrucción que causó el error.

Ejemplo

Vea también: ERROR, ON ERROR

RETURN: Vea GOSUB

**RIGHT$**: Vea LEFT$

RMDIR: Vea CHDIR

RND: Vea RANDOMIZE

RSET: Vea LSET

**RTRIM$**: Vea LTRIM$

RUN

Ejecuta el programa actual o el programa especificado.

RUN [{númerolínea | archivo$}]

* númerolínea El número de línea en el program actual donde la ejecución deberá comenzar. Si no se especifica un número de línea, la ejecución comenzará en la primera línea ejecutable.
* archivo$ El nombre de un archivo de origen de Basic. QBasic supone que tendrá la extensión .BAS.
* RUN cierra todos los archivos y borra la memoria del programa antes de cargar un programa. Utilice la instrucción CHAIN si desea ejecutar un programa sin cerrar los archivos abiertos.

Ejemplo:

```
'Supone que el programa TEST.BAS está en el directorio \DOS.
RUN "C:\DOS\TEST.BAS"
```

Vea también: CHAIN

SCREEN (función)

Devuelve el valor ASCII o el atributo de color de un carácter en la posición de pantalla especificada.

SCREEN (línea%,columna% [,indicolor%])

* línea% La coordenada indicando la línea de un carácter.
* columna% La coordenada indicando la columna de un carácter.
* indicolor% Un valor (0 ó 1) que especifica el tipo de resultado producido.

Valor	Resultado
0 (u omitido)	El código ASCII del carácter
1	El atributo de color del carácter

Ejemplo:

```
CLS
PRINT "Hola
PRINT "El valor ASCII del carácter situado en 1,1 es"; SCREEN(1, 1)
```

Vea también: POINT, SCREEN (instrucción), Códigos de caracteres ASCII, Atributos y valores de color

SCREEN (instrucción)

Establece el modo de pantalla y otras características de la pantalla.

SCREEN modo% [,[cambiocolor%] [,[páginactiva%] [,páginavisual%]]]

* modo% Establece el modo de pantalla. Vea Modos de pantalla.
* cambiocolor% Un valor (0 ó 1) que cambia entre el modo de colores y el modo monocromático (modos 0 y 1 solamente):

Modo	Valor	Efecto
0	0	Desactiva colores
0	No cero	Activa color
1	0	Activa color
1	No cero	Desactiva color

* páginactiva% La página de pantalla en la que se escribe la información de salida de texto o gráficos.
* páginavisual% La página de pantalla presentada actualmente en la pantalla.

Ejemplo:

```
'Este ejemplo requiere un adaptador de gráficos a color.
SCREEN 1        'gráficos 320 x 200
LINE (110, 70)-(190, 120), , B
LINE (0, 0)-(320, 200), 3, , &HFF00
```

Vea también: CIRCLE, COLOR, DRAW, LINE, PAINT, SCREEN (función), VIEW, WINDOW, Modos de pantalla

SEEK

La función SEEK devuelve la posición actual del archivo.
La instrucción SEEK establece la posición del archivo para la siguiente acción de leer o escribir.

SEEK(numarchivo%)
SEEK [#]numarchivo%, posición&

* numarchivo% El número de un archivo abierto.
* posición& La posición donde ocurrirá la siguiente acción de leer o escribir. Para archivos de acceso aleatorio, un número de registro. Para otros archivos, la posición de byte con relación al principio del archivo. El primer byte ocupa la posición 1.

Ejemplo:

```
OPEN "TEST.DAT" FOR RANDOM AS #1
FOR i% = 1 TO 10
    PUT #1, , i%
NEXT i%
```

```
    SEEK #1, 2
    GET #1, , i%
    PRINT "Datos: "; i%; " Registro actual:"; LOC(1); " Siguiente:"; SEEK(1)
```

Vea también: GET, PUT, OPEN

SELECT CASE

Ejecuta uno de los bloques de instrucciones, según el valor de una expresión.

```
SELECT CASE expresiónaprobar
CASE listaexpresiones1
  [bloqueinstrucciones-1]
[CASE listaexpresiones2
  [bloqueinstrucciones-2]]...
[CASE ELSE
  [bloqueinstrucciones-n]]
END SELECT
```

*	expresiónaprobar	Cualquier expresión numérica o de cadena.
*	listaexpresiones1	Una o más expresiones para comparar con expresiónaprobar.
	listaexpresiones2	La palabra clave IS debe preceder a cualquier operador relacional que haya en una expresión.
*	bloqueinstrucciones-1 bloqueinstrucciones-2 bloqueinstrucciones-n	Una o más instrucciones en una o más líneas.
*	El argumento listaexpresiones puede tener cualquiera de las siguientes formas o una combinación, separadas por comas: expresión[,expresión]... expresión TO expresión IS operador-relacional expresión	
	expresión	Cualquier expresión numérica o de cadena que sea compatible con expresiónaprobar.
	operador-relacional	Uno de los siguientes operadores relacionales: <, <=, >, >=, <> o =.

Ejemplo:

```
    INPUT "Escriba nivel de riesgo aceptable (1-5): ", Total
    SELECT CASE Total

        CASE IS >= 5
            PRINT "Riesgo y ganancia máximos."
            PRINT "Seleccione plan de inversiones en bolsa de valores."

        CASE 2 TO 4
            PRINT "Riesgo y ganancia de moderados a altos."
            PRINT "Seleccione fondo mutuo o bonos de corporaciones."

        CASE 1
            PRINT "Sin riesgo, pocas ganancias."
            PRINT "Seleccione plan de pensión individual."

    END SELECT
```

Vea también: IF

SGN: Vea ABS

SHARED

SHARED permite que los procedimientos tengan acceso a variables de nivel de módulo.
STATIC convierte una variable local en una función o procedimiento y conserva su valor entre llamados.

```
SHARED variable[()] [AS tipo] [,variable[()] [AS tipo]]...
STATIC variable[()] [AS tipo] [,variable[()] [AS tipo]]...
```

*	variable	El nombre de la variable de nivel de módulo que será compartida o la variable que se hará estática. Los nombres de variables pueden tener hasta 40 caracteres y deben comenzar con una letra. Los caracteres válidos son A-Z, 0-9 y el punto (.). No se pueden usar letras acentuadas ni ñ.
*	AS tipo	Declara el tipo de datos de la variable (INTEGER, LONG, SINGLE, DOUBLE, STRING o un tipo de datos definido por el usuario).

Vea también: COMMON, DIM, REDIM, SUB, FUNCTION

SHELL

Suspende la ejecución de un programa Basic para ejecutar un comando de DOS o un archivo de procesamiento por lotes (batch).

SHELL [cadenacomando$]

* cadenacomando$ El nombre de un comando de DOS o archivo de procesamiento por lotes.
* Se reanudará la ejecución del programa cuando se termine de ejecutar el comando de DOS o el archivo de procesamiento por lotes.
* Si omite la cadena de comando, SHELL invocará DOS shell y presentará el símbolo del sistema DOS. Utilice el comando EXIT para reanudar al programa.

Ejemplo:

```
SHELL "DIR"
```

SIN: Vea ATN

SINGLE: Vea Tipos de variables

SLEEP

Suspende temporalmente la ejecución del programa.

SLEEP [segundos&]

* segundos& El número de segundos durante los que se suspenderá la ejecución del programa.
* Si segundos& es 0 ó se omite, el programa será suspendido hasta que se presione una tecla o hasta que ocurra un evento interceptado.

Ejemplo:

```
PRINT "Tomando una siesta de 10 segundos..."
SLEEP 10
PRINT "¡Despiértate!"
```

Vea también: WAIT

SOUND

Genera un sonido a través del altavoz de la computadora.

SOUND frecuencia, duración

* frecuencia La frecuencia del sonido en hertz; un valor entre 37 y 32,767.
* duración El número de pulsaciones del reloj del sistema que durará el sonido; un valor entre 0 y 65,535. Hay 18.2 pulsaciones de reloj por segundo.

Ejemplo:

```
FOR i% = 440 TO 1000 STEP 5
SOUND i%, i% / 1000
NEXT i%
```

Vea también: PLAY

SPACE$

Devuelve una cadena de espacios.

SPACE$(n%)

* n% El número de espacios que tendrá la cadena.

Ejemplo:

```
FOR i% = 1 TO 5
    x$ = SPACE$ (i%)
    PRINT x$; i%
NEXT i%
```

Vea también: SPC, STRING$

SPC

Omite un número de espacios especificados en una instrucción PRINT o LPRINT.

SPC(n%)

    * n%   El número de espacios que serán omitidos; un valor
            entre 0 y 32,767.

Ejemplo:
    PRINT "Texto1"; SPC(10); "Texto2"

Vea también: PRINT, LPRINT, PRINT USING, LPRINT USING, SPACE$, TAB

SQR

Devuelve la raíz cuadrada de una expresión numérica.

SQR(expresión-numérica)

    * expresión-numérica   Un valor mayor o igual que cero.

Ejemplo:
    PRINT SQR(25), SQR(2)   'Resultado: 5 1.414214

STATIC: Vea SHARED

$STATIC: Vea $DYNAMIC

STEP

En un bucle FOR...NEXT, especifica el incremento del contador en cada iteración. En instrucciones para gráficos, especifica que la posición gráfica actual del cursor sea relativa a las coordenadas de pixeles gráficos.

Vea también: CIRCLE, FOR...NEXT, GET, PUT, LINE, PAINT, PRESET, PSET

STICK

Devuelve las coordenadas de una palanca de mando.

STICK(n%)

    * n%   Indica las coordenadas que serán devueltas como resultado:

n%	Devuelve
0	coordenada x de palanca A
1	coordenada y de palanca A
2	coordenada x de palanca B
3	coordenada y de palanca B

    * Llamar STICK(0) antes de STICK(1), STICK (2) o STICK(3).
      STICK(0) guarda un registro de las coordenadas actuales.

Ejemplo:
    Temp% = STICK(0)
    PRINT STICK(2), STICK(3)

Vea también: STRIG (función), STRIG, ON STRIG

STOP

Detiene la ejecución de un programa.

STOP

    * La palabra clave STOP también detiene la intercepción de eventos
      en las instrucciones siguientes:

      COM, ON COM   KEY, ON KEY   PEN, ON PEN
      PLAY, ON PLAY   STRIG, ON STRIG   TIMER, ON TIMER

Ejemplo:
    FOR i% = 1 TO 10
        PRINT i%
        IF i% = 5 THEN STOP   'STOP hace una pausa; F5 continúa.
    NEXT i%

Vea también: END, SYSTEM

STR$

STR$ devuelve una representación de un número como cadena.
VAL convierte una representación de número como cadena en un número.

```
STR$(expresión-numérica)
VAL(expresión-cadena$)
```

* expresión-numérica Cualquier expresión numérica.
* expresión-cadena$ Una representación de número en forma de cadena.

Ejemplo:

```
PRINT "65 decimal es representado en hexadecimal como ";
PRINT "&H" + LTRIM$(STR$(41))
PRINT VAL(RIGHT$("Microsoft 1990", 4))
```

STRIG (función)

Devuelve el estado del disparador de una palanca de mando.

```
STRIG(n%)
```

* n% Un valor que especifica el estado de la palanca:

n%	Condición
0	Disparador inferior de palanca A ha sido presionado desde la última STRIG(0)
1	Disparador inferior de palanca A está presionado actualmente
2	Disparador inferior de palanca B ha sido presionado desde la última STRIG(2)
3	Disparador inferior de palanca B está presionado actualmente
4	Disparador superior de palanca A ha sido presionado desde la última STRIG(4)
5	Disparador superior de palanca A está presionado actualmente
6	Disparador superior de palanca B ha sido presionado desde la última STRIG(6)
7	Disparador superior de palanca B está presionado actualmente

* STRIG devolverá -1 si la condición tiene el estado verdadero, de lo contrario devolverá 0.

Ejemplo:

```
PRINT "Presione Esc para salir."
DO
    IF STRIG(0) OR INKEY$ = CHR$(27) THEN EXIT DO
LOOP
DO
    BEEP               'BEEP mientras esté presionado el disparador A.
LOOP WHILE STRIG(1)
```

Vea también: STICK, STRIG (instruccción), ON STRIG

STRIG (instruccción)

STRIG activa, desactiva o suspende la intercepción de eventos de palanca de mando. Si está activada la intercepción de eventos, ON STRIG irá a una subrutina cada vez que se presione el disparador de palanca especificada.

```
STRIG(n%) ON
STRIG(n%) OFF
STRIG(n%) STOP
ON STRIG(n%) GOSUB línea
```

* n% Un valor que especifica un disparador de la palanca de mando:

n%	Disparador
0	Disparador inferior, palanca A
2	Disparador inferior, palanca B
4	Disparador superior, palanca A
6	Disparador superior, palanca B

* STRIG(n%) ON Activa la intercepción de eventos de palanca de mando.
* STRIG(n%) OFF Desactiva la intercepción de eventos de palanca de mando.
* STRIG(n%) STOP Suspende la intercepción de eventos de palanca de mando. Los eventos serán procesados un avez que se active la intercepción de errores usando SREIG ON.

* línea La etiqueta o el número de la primera línea de la subrutina para intercepción de eventos.

Ejemplo:

```
'Este ejemplo requiere una palanca de mando.
ON STRIG(0) GOSUB Identificador
STRIG(0) ON
PRINT "Presione Esc para salir."
DO UNTIL INKEY$ = CHR$(27): LOOP
END

Identificador:
PRINT "Está presionado el disparador de la palanca."
RETURN
```

Vea también: STICK, STRIG (función)

STRING: Vea Tipos de variables

STRING$

Devuelve una cadena de longitud específica constituida por un carácter repetido.

```
STRING$(longitud%,{código-ascii% | expresión-cadena$})
```

* longtitud% La longitud de la cadena.
* ascii-code% El código ASCII del carácter repetido.
* expresión-cadena$ Cualquier expresión de cadena. STRING$ llena la cadena con el primer carácter de expresión-cadena$.

Ejemplo:

```
PRINT STRING$(4, "-");
PRINT "Hola";
PRINT STRING$(4, "-")
```

Vea también: SPACE$, Códigos de caracteres ASCII

SUB

Define un procedimiento SUB.

```
SUB nombre[(listaparámetros)] [STATIC]
  [bloqueinstrucciones]
END SUB
```

* nombre El nombre de un procedimiento SUB, de hasta 40 caracteres, sin sufijo indicando el tipo de datos.
* listaparámetros Una o más variables que especifican los parámetros que serán pasados a un procedimiento SUB cuando es llamado:

  variable[( )] [AS tipo] [, variable[( )] [AS tipo]]...

  variable Un nombre de variable Basic.

  tipo El tipo de datos de la variable (INTEGER, LONG, SINGLE, DOUBLE, STRING, o un tipo de datos definido por el usuario).
* STATIC Especifica que los valores de las variables locales del procedimiento SUB sean guardados entre llamados a la función.
* Cuando hace el llamado a un procedimiento SUB, podrá especificar que el valor de un argumento no sea cambiado por el procedimiento, poniendo el argumento entre paréntesis.

Vea también: Ejemplo en CALL, DECLARE, EXIT, FUNCTION, SHARED, STATIC

SWAP

Intercambia los valores de dos variables.

```
SWAP variable1, variable2
```

* variable1 y variable2 Dos variables del mismo tipo de datos.

Ejemplo:

```
a% = 1: b% = 2
PRINT "Antes: "; a%, b%
SWAP a%, b%
PRINT "Después: "; a%, b%
```

SYSTEM

Cierra los archivos abiertos y devuelve el control al sistema operativo.

SYSTEM

Vea también: END, STOP

TAB

Mueve el cursor de texto a la posición de impresión especificada.

TAB(columna%)

* columna% El número de columna de la nueva posición de impresión.

Ejemplo:
```
PRINT TAB(25); "Texto"
```

Vea también: PRINT, LPRINT, PRINT USING, LPRINT USING, SPC, SPACE$

TAN: Vea ATN

THEN: Vea IF

TIME$

La función TIME$ devuelve la hora actual según el sistema de su computadora. La instrucción TIME$ define la hora actual en el sistema de la computadora.

TIME$
TIME$ = expresión-cadena$

* expresión-cadena$ La hora, en una de las siguientes formas:
  - hh Define la hora; los minutos y segundos cambian automáticamente a 00.
  - hh:mm Define la hora y los minutos; los segundos cambian automáticamente a 00.
  - hh:mm:ss Define la hora, los minutos y los segundos.
* La función TIME$ devuelve una cadena en la forma hh:mm:ss.

Ejemplo:
```
PRINT TIME$
TIME$ = "08:00:58"    'Nota: La nueva hora del sistema permanecerá
                      '      vigente hasta que la vuelva a cambiar.
PRINT "Hora cambiada a "; TIME$
```

Vea también: DATE$

TIMER (función)

Devuelve el número de segundos transcurridos desde la medianoche.

TIMER

* Utilice TIMER para medir el tiempo de programas o porciones de programas, o úsela junto con la instrucción RANDOMIZE para inicializar el generador de números aleatorios.

Ejemplo:
```
RANDOMIZE TIMER
```

Vea también: RANDOMIZE, RND, TIMER (instrucción), ON TIMER

TIMER (intercepción de eventos)

TIMER activa, desactiva o suspende la intercepción de eventos de TIMER. Si está activada la intercepción de eventos, ON TIMER irá a una subrutina cada vez que transcurra el número de segundos especificado.

TIMER ON
TIMER OFF
TIMER STOP
ON TIMER(n%) GOSUB línea

* TIMER ON Activa la intercepción de eventos de TIMER.
* TIMER OFF Desactiva la intercepción de eventos de TIMER.
* TIMER STOP Suspende la intercepción de eventos de TIMER. Los eventos serán procesados una vez que se active la intercepción de eventos a través de TIMER ON.

* n% El número de segundos que deberán transcurrir antes de que ON TIMER se bifurque a una subrutina de intercepción de eventos; un valor entre 1 y 86,400 (24 horas).
* línea La etiqueta o el número de la primera línea de la subrutina para intercepción de eventos.

Ejemplo:

```
ON TIMER(1) GOSUB HoraActualizada
TIMER ON
CLS
PRINT "Hora: "; TIME$
HoraInicio = TIMER
WHILE TiempoTranscurrido < 10
    TiempoTranscurrido = TIMER - HoraInicio
WEND
END

HoraActualizada:
    LOCATE 1, 7: PRINT TIME$
    RETURN
```

Vea también: TIMER (función)

TO

Especifica una gama de valores para:

* Una cláusula CASE de una instrucción SELECT CASE.
* Un contador de bucle en un bucle FOR...NEXT.
* Registros para bloqueo o desbloqueo en una instrucción LOCK...UNLOCK.
* Los límites superior e inferior en una instrucción DIM o REDIM.

Vea también: DIM, REDIM, FOR...NEXT, LOCK, UNLOCK, SELECT CASE

TIPOS DE VARIABLES

Las siguientes palabras clave especifican el tipo de datos de una variable en una instrucción declarativa o lista de parámetros:

* INTEGER Una variable de entero de 16 bits con signo.
* LONG Una variable de entero de 32 bits con signo.
* SINGLE Una variable de precisión sencilla y de punto flotante, de 32 bits.
* DOUBLE Una variable de precisión doble y de coma flotante, de 64 bits.
* STRING * n% Una variable de cadena de longitud fija con una longitud de n% bytes.
* STRING Una variable de cadena de longitud variable.

Vea también: AS, Juego de caracteres Basic, COMMON, DECLARE, DEF FN, DIM, REDIM, FUNCTION, SHARED, STATIC, SUB, TYPE

TROFF, TRON

TRON y TROFF activan y desactivan el rastreo de instrucciones del programa.

```
TRON
TROFF
```

* Las características de depuración de QBasic hacen innecesarias estas instrucciones al utilizar las teclas para ejecutar y depurar.

TYPE

Define un tipo de datos que contiene uno o más elementos.

```
TYPE tipousuario
  elemento AS tipo
  [elemento AS tipo]
.
.
.
END TYPE
```

* tipousuario El nombre del tipo de datos que será definido. El nombre puede tener hasta 40 caracteres y debe comenzar con una letra. Los caracteres válidos son A-Z, 0-9 y el punto (.). No se pueden usar letras acentuadas ni la ñ.
* elemento Un elemento del tipo de datos definido por el usuario.
* tipo El tipo de datos del elemento (INTEGER, LONG, SINGLE,

DOUBLE, STRING o un tipo de datos definido por el usuario).
* Use DIM, REDIM, COMMON, STATIC o SHARED para crear una variable con un tipo de datos definido por el usuario.

Ejemplo:
```
TYPE Cartas
    Palo AS STRING * 9
    Valor AS INTEGER
END TYPE
DIM Juego(1 TO 52) AS Carta
Juego(1).Palo = "Bastos"
Juego(1).Valor = 2
PRINT Juego(1).Palo, Juego(1).Valor
```

Vea también: COMMON, DIM, REDIM, SHARED, STATIC, Tipos de variables

---

UBOUND: Vea LBOUND

**UCASE$**: Vea LCASE$

UNLOCK: Vea LOCK

UNTIL: Vea DO

USING

Especifica el formato para las instrucciones PRINT USING y LPRINT USING y las asignaciones de paleta para la instrucción PALETTE USING.

Vea también: PALETTE, PALETTE USING, PRINT USING, LPRINT USING

VAL: Vea STR$

VARPRT

VARPTR devuelve la dirección compensada de una variable.
VARSEG devuelve la dirección de segmento de una variable.

```
VARPTR(nombre-variable)
VARSEG(nombre-variable)
```

* nombre-variable Cualquier nombre de variable Basic.

Vea también: CALL ABSOLUTE, DEF SEG, PEEK, POKE, VARPTR$

VARPTR$

Devuelve una representación de la dirección de una variable en forma de una cadena para ser utilizada en instrucciones DRAW y PLAY.

```
VARPTR$(cadena-comando$)
```

* cadena-comando$ Una variable de cadena que contiene comandos DRAW o PLAY.

Ejemplo:
```
Escala$ = "CDEFGAB"
PLAY "L16"
FOR i% = 0 TO 6
    PLAY "O" + STR$(i%)
    PLAY "X" + VARPTR$(Escala$)
NEXT i%
```

Vea también: DRAW, PLAY (música), VARPTR, VARSEG

VARSEG: Vea VARPTR

VIEW

Define el tamaño y la posición de un marco de visualización (viewport) en donde se pueden presentar gráficos en la pantalla.

```
VIEW [[SCREEN] (x1!,y1!)-(x2!,y2!) [,[color%] [,bordes%]]]
```

* SCREEN Especifica que las coordenadas se situen con relación a la pantalla y no con el marco de visualización (viewport).
* (x1!,y1!)-(x2!,y2!) Las coordenadas de las esquinas diagonalmente opuestas del marco de visualización (viewport).
* color% Un atributo de color que establece el color de

relleno del marco de visualización (viewport).
    * bordes% Un atributo de color que establece el color de los bordes del marco de visualización (viewport).
    * Si se omiten todos los argumentos, la pantalla completa será el marco de visualización (viewport).
    * Los atributos de color disponibles dependerán del adaptador de gráficos y del modo de pantalla establecido mediante la más reciente instrucción SCREEN.

Ejemplo:
```
    'Este ejemplo requiere un adaptador de gráficos a color.
    SCREEN 1
    VIEW (10, 10)-(300, 180), , 1
    LOCATE 1, 11: PRINT "Un marco de visualización de gráficos grande";
    VIEW SCREEN (80, 80)-(200, 125), , 1
    LOCATE 11, 11: PRINT "Un marco de visualización de gráficos pequeño";
```

Vea también: CLS, SCREEN, VIEW PRINT, WINDOW, Atributos y valores de color, Modos de pantalla

---

VIEW PRINT

Establece los límites de margen para el texto en la pantalla del marco de visualización.

```
VIEW PRINT [filasup% TO filainf%]
```

    * filaasup% El número de la fila superior del texto en la ventana.
    * filainf% El número de la fila inferior del texto en la ventana.
    * Si omite los argumentos filasup% y filainf%, VIEW PRINT establecerá la pantalla completa como pantalla de texto.
    * La gama de valores para filasup% y filainf% depende del modo de pantalla.

Ejemplo:
```
    VIEW PRINT 10 TO 15
    FOR i% = 1 TO 100      'Información de salida se desplazará.
        PRINT i%
    NEXT i%
```

Vea también: CLS, LOCATE, PRINT, LPRINT, SCREEN, WIDTH, Modos de pantalla

---

WAIT

Suspende la ejecución del programa hasta que la configuración de bits especificada se envie desde un puerto de entrada.

```
WAIT numpuerto%, expresión-AND% [,expresión-XOR%]
```

    * numpuerto% El número de un puerto de entrada.
    * expresión-AND% Una expresión en entero que WAIT combina con el valor de la configuración de bits usando un operador AND. Cuando el resultado no es cero, WAIT dejará de controlar el puerto.
    * expresión-XOR% Puede ser utilizado para activar y desactivar bits de línea en la configuración de bits antes de aplicar la operación AND.

Ejemplo:
```
    'Leer la dirección del puerto del controlador de interrupción &H20.
    'Presione cualquier tecla para continuar.
    WAIT &H20, 1
```

Vea también: INP, OUT, AND

WHILE ... WEND

Ejecuta una serie de instrucciones siempre que la condición especificada tenga el estado verdadero.

```
WHILE condición
.
.
.
WEND
```

    * condición Una expresión numérica que Basic evalúa como verdadera (no cero) o falsa (cero).

* DO...LOOP ofrece una mejor forma de ejecutar instrucciones en un bucle dentro del programa.

Vea también: DO, FOR...NEXT

WIDTH

Asignan un ancho a la línea de salida enviado a un dispositivo como una impresora o archivo, o cambian el número de columnas o filas presentadas en pantalla.

```
WIDTH [columnas%] [,filas%]
WIDTH {#numarchivo% | dispositivo$}, columnas%
WIDTH LPRINT columnas%
```

* columnas% — El ancho deseado, en columnas. El ancho presentado en la pantalla debe ser 40 u 80 columnas.
* filas% — La altura deseada, en filas. La altura presentada en la pantalla puede ser 25, 30, 43, 50 ó 60, según el adaptador de video y el modo de pantalla.
* #numarchivo% — El número de un archivo o dispositivo abierto.
* dispositivo$ — El nombre de un dispositivo: SCRN:, COM1:, COM2:, LPT1:, LPT2:, LPT3:

Ejemplo:

```
OPEN "LPT1:" PARA SALIDA COMO #1
WIDTH #1, 132
```

Vea también: PRINT, LPRINT, SCREEN, VIEW PRINT

WINDOW

Define las dimensiones lógicas de la ventana de gráficos actual. Utilice la instrucción WINDOW para definir su propio sistema de coordenadas para el marco de visualización.

```
WINDOW [[SCREEN] (x1!,y1!)-(x2!,y2!)]
```

* SCREEN — Invierte la dirección cartesiana normal de las coordenadas de pantalla de manera que los valores aumentan desde la parte superior de la pantalla hasta la parte inferior.
* (x1!,y1!) — Coordenadas lógicas que corresponden a las coordenadas de pantalla superior-izquierda del viewport.
* (x2!,y2!) — Coordenadas lógicas que corresponden a las coordenadas de pantalla inferior-derecha del marco de visualización.
* WINDOW sin argumentos desactiva el sistema de coordenadas lógicas.
* Utilice la instrucción VIEW si desea cambiar el tamaño de la ventana.

Ejemplo:

```
'Este ejemplo requiere un adaptador de gráficos a color.
SCREEN 1
FOR i% = 1 TO 10 STEP 2
    WINDOW (-160 / i%, -100 / i%)-(160 / i%, 100 / i%)
    CIRCLE (0, 0), 10
NEXT i%
```

Vea también: CLS, PMAP, POINT, SCREEN, VIEW, WIDTH

WRITE

Escribe datos en la pantalla o en un archivo secuencial.

```
WRITE [[#]numarchivo%,] listaexpresiones
```

* numarchivo% — El número de un archivo secuencial abierto. Si se omite el número de archivo, WRITE escribirá en la pantalla.
* listaexpresiones — Una o más variables o expresiones, separadas con comas, cuyos valores serán escritos en la pantalla o en un archivo.
* WRITE inserta comas entre los datos y comillas alrededor de cadenas mientras usted los escribe. WRITE escribe valores en un archivo en una forma que puede ser leída por la instrucción INPUT.

Ejemplo:

```
CLS
OPEN "LISTA" FOR OUTPUT AS #1
DO
    INPUT "   NOMBRE:       ", Nombre$
    INPUT "   EDAD:        ", Edad$
```

```
        WRITE #1, Nombre$, Edad$
        INPUT "Agregue otros datos"; R$
    LOOP WHILE UCASE$(R$) = "Y"
    CLOSE #1
    'Imprime el archivo en la pantalla.
    OPEN "LISTA" FOR INPUT AS #1
    CLS
    PRINT "Datos en archivo:": PRINT
    DO WHILE NOT EOF(1)
        INPUT #1, Reg1$, Reg2$   'Lee datos del archivo.
        PRINT Reg1$, Reg2$       'Imprime los datos en la pantalla.
    LOOP
    CLOSE #1
    KILL "LISTA"
```

Vea también: INPUT, LINE INPUT, OPEN, PRINT, LPRINT

XOR: Vea AND
