IBM

Personal Computer

BASIC

Manuale di consultazione rapida

6834179

IBM *Personal Computer*

BASIC

Manuale di consultazione rapida

Come usare il manuale

La funzione del manuale di consultazione rapida è quella di ricordare lo scopo e la sintassi dei comandi, delle specifiche, delle funzioni e delle variabili BASIC.

Tali voci sono elencate in ordine di categoria e viene fornita la spiegazione di ogni esempio.

Per ulteriori dettagli fare riferimento al manuale BASIC.

Indice

Generali

ASC

```
10 X$="TEST"
20 PRINT ASC(X$)
```

Viene stampato il codice ASCII del primo carattere di X$. Per la lettera maiuscola T, il valore è 84.

BEEP

```
10 IF X<20 THEN BEEP
```

Se X è inferiore a 20 l'altoparlante emette un unico segnale acustico.

BLOAD

```
10 DEF SEG=&HB800
20 BLOAD "PICTURE",0
```

Carica un file di nome PICTURE precedentemente caricato in memoria alla posizione relativa 0 del segmento HB8000.

BSAVE

```
10 DEF SEG=&HB800
20 BSAVE "PICTURE",0,&H4000
```

Salva l'immagine dell'area di memoria nel segmento HB8000 dalla posizione relativa 0 alla 4000. L'immagine viene scritta su un file chiamato PICTURE.

CALL

```
10 DEF SEG=&H8000
20 OFS=0
30 CALL OFS(A,B$,C)
```

Richiama un sottoprogramma in linguaggio macchina che comincia alla posizione 0 del segmento H8000 e passa allo stesso le variabili A,B$ e C.

CHAIN

```
10 CHAIN "A:PROG1",10,ALL
```

Trasferisce il controllo a PROG1 che comincia l'esecuzione alla linea 10 e gli passa mediante ALL tutte le variabili.

CLEAR

```
10 CLEAR 32768,2000
```

Pulisce mediante CLEAR la memoria usata per i dati, senza cancellare il programma in uso; imposta l'area di lavoro a 32768 byte e la coda di lavoro a 2000 byte.

CLS

```
10 CLS
```

Pulisce lo schermo e posiziona il cursore in alto a sinistra.

COLOR (Text Mode)

```
10 COLOR 1,0
```

Il video monocromatico IBM produce caratteri verdi evidenziati su sfondo nero. Il video a colori produce caratteri azzurri su sfondo nero.

COMMON

```
100 COMMON A, B(), C$
110 CHAIN "A:PROG3"
```

Passa a PROG3 le variabili A,B(), e C$.

CSRLIN

```
10 Y=CSRLIN
```

Memorizza il valore del numero di riga del cursore in uso.

DATA

```
10 DATA 3.05, 5.89, 23.08
```

Memorizza una lista di costanti DATA a cui si accede mediante una specifica READ.

DATE$

```
10 DATE$="10/31/83"
20 PRINT DATE$
```

Può impostare e richiamare la data.

DEF FN

```
10 PI=3.141593
20 DEF FNAR(R)=PI*RA2
30 INPUT "Radius? ", RAD
40 PRINT "Area=";FNAR(RAD)
```

Definisce una funzione scritta per calcolare l'area di un cerchio.

DEF SEG

```
10 DEF SEG=&HB800
```

Definisce il segmento di memoria a cui si può accedere HB8000 è l'inizio della memoria di transito del video a colori.

DEFtype

```
10 DEFINT A-D
20 DEFSING F-L,X
30 DEFSTR M-W
```

Dichiara i tipi di variabile mediante la prima lettera del nome della variabile stessa.

DEF USR

```
10 CLEAR,&H8000
20 DEF SEG
30 DEF USR2=&H8000
40 X=USR2 (Y+2)
```

Definisce un sottoprogramma in linguaggio macchina chiamato USR(2) che comincia alla posizione relativa 24000 del segmento in uso.

DIM

```
10 DIM VAR(20)
```

Genera dello spazio in memoria per un massimo di 21 subscritti per la variabile VAR.

END

```
10 END
```

Finisce l'esecuzione del programma.

ENVIRON

```
10 ENVIRON "PATH=C:\"
```

Imposta un percorso nella tabella in ambiente BASIC nell'indirizzario radice sull'unità C:.

ENVIRON$

```
10 PRINT ENVIRON$("PATH")
```

Visualizza il contenuto attuale del parametro PATH nella tabella di ambiente BASIC.

EOF

```
10 IF EOF THEN END
```

Effettua un controllo di fine file durante la lettura da un file sequenziale.

ERASE

```
10 ERASE TEST1,TEST2
```

Cancella le schiere TEST1 e TEST2 dalla memoria.

ERDEV

```
10 PRINT ERDEV
```

Visualizza un numero contenente un codice di errore di unità.

ERDEV$

```
10 PRINT ERDEV$
```

Visualizza il nome dell'unità che ha causato l'errore.

ERR

```
10 IF ERR=27 THEN GOTO 100
```

Se il codice di errore è uguale a 27 (fine carta) allora GOTO al programma di gestione degli errori.

ERL

```
10 IF ERL=250 THEN RESUME
```

Se l'errore si verifica alla riga 250, ignora l'errore.

ERROR

```
20 ON ERROR GOTO 40
30 IF B>500 THEN ERROR 210
40 IF ERR=210 THEN END
```

Definisce un nuovo codice di errore da utilizzare nel programma.

FOR...NEXT

```
10 FOR I=1 TO 500 STEP 2
20 NEXT I
```

Aumenta di 2 il contatore ogni volta che viene eseguita una NEXT sino al raggiungimento di 500.

FRE

```
PRINT FRE(0)
```

Visualizza il numero di byte inutilizzati della memoria.

GOSUB...RETURN

```
20 GOSUB 40
30 PRINT"CHECK":END
40 PRINT"DOUBLE":RETURN
```

Va al sottoprogramma di cui alla riga 40 quindi ritorna alla specifica che segue la GOSUB.

GOTO

```
10 GOTO 30
20 PRINT"ERROR"
30 PRINT"CONTINUE"
40 PRINT"PROGRAM
```

Va a riga 30 e continua l'esecuzione.

IF

```
10 IF A=20 THEN GOTO 30 ELSE STOP
```

Se A=20 è vero, salta a riga 30; altrimenti ferma l'esecuzione.

INKEY$

```
10 PRINT"PRESS ANY KEY"
20 A$=INKEY$
30 IF A$="" THEN 10
```

Legge un carattere dalla tastiera e lo assegna alla variabile chiamata A$. Se non viene premuto nessun tasto l'elaborazione torna a riga 10 e continua sino alla pressione di un tasto.

INP

```
10 A=INP(255)
```

Legge un dato dalla porta n° 255 e lo assegna alla variabile A.

INPUT

```
10 INPUT "WHAT RADIUS";R
```

Attende l'immissione da tastiera; visualizza la frase fra apici; assegna l'input da tastiera alla variabile R.

INPUT$

```
10 X$=INPUT$(5)
```

Legge una stringa di 5 caratteri della tastiera e li assegna a X$.

IOCTL

```
10 OPEN "O",#1,"LPT1:"
20 IOCTL #1, "PL56"
```

Passa il dato di controllo (PL56) che imposta la lunghezza della pagina dell'unità aperta #1.

IOCTL$

```
10 IF IOCTL$(1)="56" THEN PRINT "OK"
```

Legge la stringa dei dati di controllo nell'unità aperta #1.

KEY

```
10 KEY 6, "PRINT"
```

Programma il tasto funzionale F6 a visualizzare la parola PRINT.

KEY(n)

```
10 KEY(10) STOP
```

Disabilita il tasto funzionale 10.

LET

```
10 LET Y=35
```

Assegna il valore 35 alla variabile Y.

LINE INPUT

```
10 LINE INPUT "Address? ";C$
```

Legge un'intera linea di immissione inclusi i delimitatori nella variabile C$.

LOCATE

```
10 LOCATE 10,15
```

Posiziona il cursore a riga 10, colonna 15.

LPOS

```
10 PRINT LPOS(0)
```

Visualizza la posizione della testina di stampa nella memoria di transito della stampa.

LPRINT

```
10 LPRINT "TESTING"
```

Stampa la parola TESTING.

LPRINT USING

```
10 LPRINT USING "###.##";456.7832
```

Invia l'espressione 456.7832 sulla stampatrice usando il formato tra apici. La stampa sarà 456.78.

MERGE

```
10 MERGE "B:PROG2"
```

Fonde le linee del programma PROG2 con le linee dell'attuale programma in memoria. I numeri di linea duplicati vengono rimpiazzati da quelli del programma PROG2.

MOTOR

```
1Ø MOTOR 1
```

Mette in moto il registratore a cassette.

ON ERROR

```
1Ø ON ERROR GOTO 5ØØ
```

Ad ogni errore salta alla riga 500.

ON...GOSUB

```
1Ø ON NUMBER GOSUB 22Ø, 45Ø, 13Ø
```

If NUMBER=1. GOSUB 220; If NUMBER=2. GOSUB 450; If NUMBER=3, GOSUB 130.

ON... GOTO

```
1Ø ON NUMBER GOTO 22Ø, 45Ø, 13Ø
```

Uguale a ON...GOSUB

ON KEY(n)

```
1Ø ON KEY(1Ø) GOSUB 1ØØØ
```

Quando viene premuto il tasto funzionale 10 va alla riga 1000.

ON PEN

```
1Ø ON PEN GOSUB 2ØØØ
```

Quando la penna luminosa è attivata, va alla riga 2000.

ON PLAY (n)

```
1Ø ON PLAY(5) GOSUB 5ØØ
```

Quando la musica suona in sottofondo e rimangono 5 note nella memoria di transito, va alla riga 500.

ON STRIG(n)

```
1Ø ON STRIG(2) GOSUB 1ØØØ
```

Va alla riga 1000 quando viene premuto il pulsante del comando giochi.

ON TIMER

```
1Ø ON TIMER(3Ø) GOSUB 5ØØ
```

Trascorsi 30 secondi passa a riga 500.

OPTION BASE

```
1Ø OPTION BASE 1
```

Imposta ad 1 il subscritto dell'elemento della schiera di numero più basso.

OUT

```
10 OUT 980,2
```

Invia un valore di 2 alla porta 980.

PEEK

```
10 PEEK (&H410)
```

Legge il valore di memoria all'indirizzo H410.

PEN

```
10 PRINT PEN(6)
```

Visualizza la riga su cui si trovava la penna luminosa mentre era attiva.

PLAY

```
10 PLAY "XA$;"
```

Suona della musica secondo quanto contenuto nella stringa XA$.

PLAY(n)

```
10 PRINT PLAY(0)
```

Visualizza il numero di note restanti nella memoria di transito della musica di sottofondo.

POKE

```
10 SCREEN 1
20 DEF SEG
30 POKE &HFE,2
```

Scrive il valore 2 nella posizione di memoria HFE.

POS

```
10 IF POS(0)>60
   THEN PRINT CHR$(13)
```

Se la posizione della colonna del cursore è oltre 60, viene eseguito un ritorno di carrello.

PRINT

```
10 PRINT "ANYTHING"
```

Visualizza sullo schermo la parola ANYTHING.

PRINT USING

```
10 PRINT USING "###.###";456.2341
```

Visualizza l'espressione 456.2341 sullo schermo usando il formato tra apici. Sullo schermo apparirà 456.23.

RANDOMIZE

```
10 RANDOMIZE (20000)
```

Ripristina il generatore di numeri casuali con il numero 20000 per produrre una nuova sequenza di numeri.

READ

```
10 READ A(I)
```

Legge un valore da una specifica DATA e lo assegna alla variabile A (I).

RESTORE

```
10 RESTORE
```

Causa la lettura da parte della successiva specifica READ, della prima specifica DATA nel programma.

RESUME

```
10 RESUME 120
```

Conseguentemente al verificarsi di un errore l'esecuzione riprende dalla linea 120.

RETURN

```
10 RETURN
```

Il programma ritorna alla riga successiva la GOSUB che ha generato il salto ad un sottoprogramma.

RND

```
10 Y=INT(RND*7)
```

Fornisce i numeri casuali tra 0 e 6.

SCREEN

```
10 PRINT SCREEN(5,10)
```

Visualizza il codice ASCII per il carattere che si trova alla riga 5, colonna 10 dello schermo.

SHELL

```
10 SHELL
```

Carica il DOS dal BASIC; il programma in uso rimane in memoria.

SOUND

```
10 SOUND 220.000, 18
```

Produce un suono di 220 cps per la durata di 18 battiti di orologio (1 secondo).

SPC

```
10 PRINT SPC(20)
```

Lascia in stampa 20 spazi.

STICK

```
1Ø P=STICK(Ø)
```

Ritorna la coordinata X della barra di comando A e l'assegna alla variabile P.

STRIG

```
1Ø STRIG ON
```

Permette la lettura dello stato del comando giochi.

STRIG(n)

```
1Ø STRIG(2) ON
```

Permette l'intercettazione del comando giochi B1 mediante la specifica ON STRIG(n).

SWAP

```
1Ø SWAP X$,Y$
```

Scambia tra loro i valori delle variabili X$ e Y$.

SYSTEM

```
1Ø SYSTEM
```

Ritorna al DOS.

TAB

```
1Ø PRINT TAB(25)
```

Muove il cursore alla posizione 25.

TIMES

```
1Ø TIME$="1Ø:23:ØØ"
2Ø PRINT TIME$
```

Imposta e recupera l'ora attuale.

TIMER

```
1Ø PRINT TIMER
```

Visualizza il numero di secondi trascorsi dalla mezzanotte o dal ripristino del sistema.

USR

```
1Ø DEF USRØ=&HFØØØ
2Ø C=USRØ(B/2)
```

Richiama il sottoprogramma in linguaggio macchina USRØ e fornisce l'argomento (B/2).

VARPTR

```
1Ø PRINT VARPTR(X)
```

Visualizza l'indirizzo in memoria della variabile X.

VARPTR$

```
10 PLAY "X"+VARPTR$(A$)
```

Ritorna una stringa di 3 byte dell'indirizzo di memoria della variabile A$. E' equivalente a PLAY"XA$;".

WAIT

```
10 WAIT 32,2
```

Il programma rimane in attesa fino a quando la porta 32 non riceve un bit di valore 1 nella posizione del secondo bit.

WHILE and WEND

```
10 X=0
20 WHILE X=0
30 INPUT X
40 S=S+X
50 WEND
60 PRINT "SUM=";S
```

Le specifiche tra WHILE e WEND vengono continuamente ripetute fino a quando non viene immesso un valore per X.

WIDTH

```
10 WIDTH 40
```

Imposta l'ampiezza dello schermo a 40 caratteri per riga.

WRITE

```
10 A=80:B=90:C$="THE END"
20 WRITE A,B,C$
```

Uguale a PRINT con l'inserimento di virgole e apici. L'esempio visualizzerà 80,90, THE END.

Comunicazioni

COM(n)

```
10 COM(1) ON
```

Abilita l'intercettazione dell'adattatore per comunicazioni #1.

ON COM(n)

```
10 ON COM(1) GOSUB 1000
```

Quando un'attività viene individuata nell'adattatore per comunicazioni #1, l'elaboratore passa alla riga 1000.

OPEN "COM...

```
10 OPEN "COM:" AS 1
```

Apre l'adattatore per comunicazioni #1.

Conversioni

CDBL

```
10 PRINT CDBL(A)
```

Converte il valore di A in un numero a precisione doppia.

CHR$

```
10 PRINT CHR$(66)
```

Converte il valore 66 nell'equivalente carattere ASCII che corrisponde alla lettera maiuscola B.

CINT

```
10 PRINT CINT(45.67)
```

Converte il valore 45.67 nel valore intero 46.

CSNG

```
10 PRINT CSNG(45.34536789)
```

Converte il valore a precisione doppia 45.34536789 nel valore a precisione singola 45.3453.

CVD

```
10 Y=CVD(N$)
```

Converte la variabile a stringa N$ di 8 byte in variabile numerica a precisione doppia.

CVI

```
10 X=CVI(X$)
```

Converte la variabile a stringa X$ di 2 byte in una variabile intera.

CVS

```
10 Y=CVS(Y$)
```

Converte una variabile a stringa Y$ di 4 byte in una variabile numerica a precisione singola.

HEX$

```
10 H$=HEX$(16)
```

Converte il valore decimale 16 nel valore esadecimale 10.

MKD$

```
1Ø D$=MKD$(AMT)
```

Converte il valore della variabile a precisione singola AMT in una variabile a stringa.

MKI$

```
1Ø R$=MKI$(STEP)
```

Converte il valore della variabile intera STEP in variabile a stringa.

MKS$

```
1Ø K$=MKS$(BALANCE)
```

Converte la variabile a precisione singola BALANCE in una variabile a stringa.

OCT$

```
1Ø PRINT OCT$(24)
```

Converte il valore decimale (24) nell'equivalente valore ottale (30).

Sviluppo

AUTO

```
1Ø AUTO 1ØØ,5Ø
```

Numera automaticamente ogni riga di programma di 5Ø in 5Ø iniziando dalla riga 1ØØ.

CONT

```
1Ø CONT
```

Continua l'esecuzione di un programma dopo una pausa.

DELETE

```
1Ø DELETE 4Ø-1ØØ
```

Cancella le righe 4Ø-1ØØ.

EDIT

```
1Ø EDIT 35
```

Visualizza per l'edizione la riga 35.

LIST

```
1Ø LIST 2Ø-2ØØ
```

Visualizza una lista delle righe 2Ø-2ØØ del programma in memoria.

LLIST

```
10 LLIST
```

Stampa la lista di un intero programma.

LOAD

```
10 LOAD "PROG3"
```

Carica il programma PROG3 in memoria.

NAME

```
10 NAME "ACCTS.BAS" AS "LEDGER.BAS"
```

Ridenomina un file.

NEW

```
10 NEW
```

Cancella dalla memoria il programma in uso e tutte le variabili.

REM

```
10 REM ignora questa specifica
```

Inserisce una nota non eseguibile.

RENUM

```
10 RENUM 1000,10
```

Rinumera l'intero programma partendo dalla riga 1000 ed incrementando di 10 ogni riga.

RUN

```
10 RUN
```

Esegue il programma in memoria.

SAVE

```
10 SAVE "PROG4",A
```

Salva il programma in memoria come un file ASCII usando il nome PROG4.

STOP

```
10 STOP
```

Ferma l'esecuzione del programma.

TRON e TROFF

```
10 TRON
20 REM traccia del programma
30 TROFF
```

Attiva e disattiva la traccia del programma.

File

CHDIR

```
10 CHDIR "\"
```

Modifica l'indirizzario dell'indirizzario radice.

CLOSE

```
10 CLOSE
```

Chiude tutti i file e le unità aperte.

FIELD

```
10 FIELD 1, 20 AS N$, 30 AS A$
```

Genera dello spazio per variabili nella memoria di transito del file ad accesso casuale # 1.

FILES

```
10 FILES "B:*.*"
```

Visualizza tutti i file presenti nell'unità B.

GET

```
10 OPEN "A:CUST" AS #1
20 GET 1
```

Legge un record dal file ad accesso casuale # 1.

INPUT #

```
10 OPEN "O",#1,"DATA"
20 INPUT#1,INFO$
```

Legge la variabile INFO$ dal file aperto.

KILL

```
10 KILL"B:DATA.BAS"
```

Cancella sull'unità B il file chiamato DATA.BAS.

LINE INPUT#

```
10 LINE INPUT#1, ADDRESS$
```

Legge un'intera riga del file aperto e lo assegna alla variabile ADDRESS$.

LOC

```
10 PRINT LOC(1)
```

Visualizza la posizione attuale nel file aperto come #1.

LOF

```
10 PRINT LOF(1)
```

Visualizza la lunghezza del file aperto come #1.

LSET e RSET

```
10 LSET N$=NA$
```

Muove il contenuto di NA$ nella memoria di transito di un file ad accesso casuale nominato N$ e lo allinea a sinistra.

MKDIR

```
10 MKDIR "SALES"
```

Crea un indirizzo chiamato SALES.

OPEN

```
10 OPEN "O",#1,"DATA"
```

Apre una unità o un file chiamato DATA per ricevere dati in emissione come file #1.

PRINT#

```
10 PRINT#1,DATE$;TIME$
```

Scrive le variabili DATE$ e TIME$ nel file sequenziale aperto come #1.

PRINT #USING

```
10 PRINT# 1 USING "###.##";AMOUNT
```

Scrive il valore della variabile AMOUNT nel file sequenziale aperto #1 usando il formato tra apici.

PUT

```
10 PUT#1
```

Scrive in un file a ricerca casuale un record immesso nella memoria di transito da LSET o RSET.

RESET

```
10 RESET
```

Chiude tutti i file aperti sul disco.

RMDIR

```
10 RMDIR "SALES"
```

Rimuove l'indirizzario chiamato SALES.

WRITE #

```
10 WRITE# 1,NAME$,AGE$
```

Scrive le variabili NAME$ e AGE$ nel file sequenziale aperto #1. Automaticamente inserisce delle virgole fra i vari passi e gli apici a delimitare le stringhe.

Grafici

CIRCLE

```
10 CIRCLE (160, 100),50
```

Disegna un cerchio con il centro a X=160, y=100 e il raggio=50.

COLOR

```
10 COLOR 9,0
```

Imposta il colore di fondo ad azzurro chiaro e seleziona la tavolozza 0 in SCREEN 1.

DRAW

```
10 SCREEN 1
20 DRAW "U20 R20 D20 L20
```

Traccia un quadrato di 20 unità di lato.

GET

```
10 GET (10,10)-(20,20), "PICTURE"
```

Salva il contenuto dello schermo nel rettangolo i cui angoli opposti sono (10,10), e (20,20) in una schiera chiamata PICTURE.

LINE

```
10 LINE (1,1)-(50,50),2,B
```

Disegna un quadrato usando il colore 2 i cui opposti angoli sono (1,1) e (50,50).

PAINT

```
10 PAINT (15,15),2
```

Riempie l'interno del quadrato riportato nell'esempio di LINE con il colore 2 iniziando dal punto (15,15).

PMAP

```
10 WINDOW (-1,-1)-(1,1)
20 PMAP(1,0):PMAP(0,1)
```

Trasforma il punto centrale di WINDOW da coordinate spaziali di (0,0) in coordinate fisiche di (160,100).

POINT

```
10 C=POINT(15,15)
```

Legge il colore del punto dello schermo (15,15) e lo assegna alla variabile C.

PSET and PRESET

```
10 PSET (15,15)
20 PRESET (15,15)
```

PSET traccia un punto alle coordinate (15,15) nel colore di primo piano. PRESET rimuove lo stesso punto.

PUT

```
10 PUT (40,40),"PICTURE"
```

Prende l'immagine caricata con GET nella schiera chiamata PICTURE e la porta sullo schermo con l'angolo superiore sinistro all'indirizzo (40,40).

SCREEN

```
10 SCREEN 1,0
```

Imposta lo schermo in modo grafici a media risoluzione ed abilita l'uso del colore.

VIEW

```
10 VIEW (5,5)-(100,100)
```

Definisce a finestra una sezione rettangolare dello schermo i cui opposti angoli sono (5,5) e (100,100) e vi immette il contenuto di WINDOW.

WINDOW

```
10 WINDOW (-6,-6)-(6,6)
20 CIRCLE (4,4),5,1
```

Crea un cerchio di raggio=5 che riempie la maggior parte dello schermo in quanto WINDOW ridefinisce le coordinate nell'ambito da +6 a –6.

Funzioni matematiche

ABS

```
10 PRINT ABS(7*(-5))
```

Visualizza il valore assoluto (35) dell'espressione stabilita.

ATN

```
10 PRINT ATN(5)
```

Visualizza l'arco tangente in radianti del numero 5.

COS

```
10 PRINT COS(3.14)
```

Visualizza il valore trigonometrico del coseno dell'angolo uguale a radianti 3,14.

EXP

```
10 PRINT EXP(1)
```

Visualizza il valore del numero e lo eleva alla prima potenza.

FIX

```
10 PRINT FIX(45.67)
```

Visualizza la parte intera (45) di 45,67.

INT

```
10 PRINT INT(-2.89)
```

Visualizza l'intero più grande (−3) inferiore o uguale a (−2.89).

LOG

```
10 PRINT LOG(45/7)
```

Visualizza il logaritmo naturale (1.860752) dell'espressione (45/7).

SGN

```
10 PRINT SGN(X)
```

Visualizza il segno della variabile X.

SIN

```
10 PRINT SIN(3.14)
```

Visualizza il valore trigonometrico del seno dell'angolo uguale a radianti 3,14.

SQR

```
10 PRINT SQR(81)
```

Visualizza la radice quadrata (9) di 81.

TAN

```
10 PRINT TAN(3.14)
```

Stampa il valore trigonometrico della tangente dell'angolo uguale a radianti 3,14.

Funzioni a stringa

INSTR

```
10 A$="ABCDEFG"
20 B$="B"
PRINT INSTR(A$,B$)
```

Confronta le due stringhe B$ e A$ e visualizza la posizione (2) di corrispondenza.

LEFT$

```
10 A$="BASIC EXAMPLE"
20 PRINT LEFT$(A$,2)
```

Visualizza i due caratteri di sinistra (BA) della stringa A$.

LEN

```
10 X$="IBM PC"
20 PRINT LEN(A$)
```

Visualizza la lunghezza (6) della stringa X$.

MID$

```
10 EX$="QRSTUVW"
20 PRINT MID$(EX$,2,3)
```

Partendo dal secondo carattere visualizza i successivi 3 caratteri (RST).

RIGHT$

```
10 SAM$="SAMPLE"
20 PRINT RIGHT$(SAM$,3)
```

Visualizza i 3 caratteri di estrema destra (PLE) della stringa SAMPLE.

SPACE$

```
10 SP$=SPACE$(25)
20 PRINT SP$;"TEST"
```

Stampa una stringa di 25 spazi; quindi stampa la parola TEST.

STR$

```
10 PRINT LEN(STR$(321))
```

Tratta l'espressione numerica 321 come una espressione a stringa e ne visualizza la lunghezza.

STRING$

```
10 X$=STRING$(10,45)
20 PRINT X$;"TEST";X$
```

Visualizza una stringa consistente nel carattere ASCII 45 (−) ripetuto 10 volte; quindi stampa la parola TEST e 10 ulteriori linette.

VAL

```
10 PRINT VAL("29 EAST AVE.")
```

Visualizza il valore numerico della stringa (29).

Messaggi di errore

Numero	Messaggio
1	NEXT without FOR
2	Syntax error
3	RETURN without GOSUB
4	Out of data
5	Illegal function call
6	Overflow
7	Out of memory
8	Undefined line number
9	Subscript out of range
10	Duplicate definition
11	Division by zero
12	Illegal direct
13	Type mismatch
14	Out of string space
15	String too long
16	String formula too complex
17	Can't continue
18	Undefined user function
19	No RESUME
20	RESUME without error
22	Missing operand
23	Line buffer overflow
24	Device timeout
25	Device fault
26	FOR without NEXT
27	Out of paper
29	WHILE without WEND
30	WEND without WHILE
50	Field overflow
51	Internal error
52	Bad file number
53	File not found
54	Bad file mode
55	File already open
57	Device I/O error
58	File already exists
61	Disk full
62	Input past end
63	Bad record number
64	Bad file name
66	Direct statement in file
67	Too many files
68	Device unavailable
69	Communication buffer overflow
70	Disk write protect
71	Disk not ready
72	Disk media error

73	Advanced feature
74	Rename across disks
75	Path/file access error
76	Path not found
—	Unprintable error
—	Incorrect DOS Version
—	You cannot SHELL to BASIC
—	Can't continue after SHELL

Note:

IBM United Kingdom International Products Limited
PO Box 41, North Harbour, Portsmouth PO6 3AU, England

Printed in Great Britain by Ben Johnson & Co. Ltd., York.