I DUE BASIC DEL PC 128

Continua l'analisi del Basic del PC 128

8ª parte

Prosegue con questa nostra ottava parte l'analisi dei comandi dei due Basic del PC 128. Questa rubrica è con noi dall'inizio della nostra pubblicazione e con ogni probabilità ci terra compagnia per molti altri numeri ancora. Pertanto arrivederci al prossimo numero e buon lavoro.

GOTO

TIPO: Istruzione

SINTASSI: GOTOn.r

COMMENTO:

Dove n.r è un numero, indicante il numero di una linea di programa.

Per mezzo di questo comando di salto incondizionato, si può trasferire il comando del programma a qualsiasi linea, purché compresa nel programma stesso. Nel caso in cui il numero di linea in argomento non esista, verrà generato un messaggio d'errore del tipo "Undefined Line".

IL comando GOTO, a volte abusato da molti programmatori, consente di formare dei cicli di controllo del tipo:

```
10 CLS
20 INPUT"Scrivere un numero";A$
30 IF A$="10" THEN PRINT
   "BRAVO":END
40 GOTO 10
```

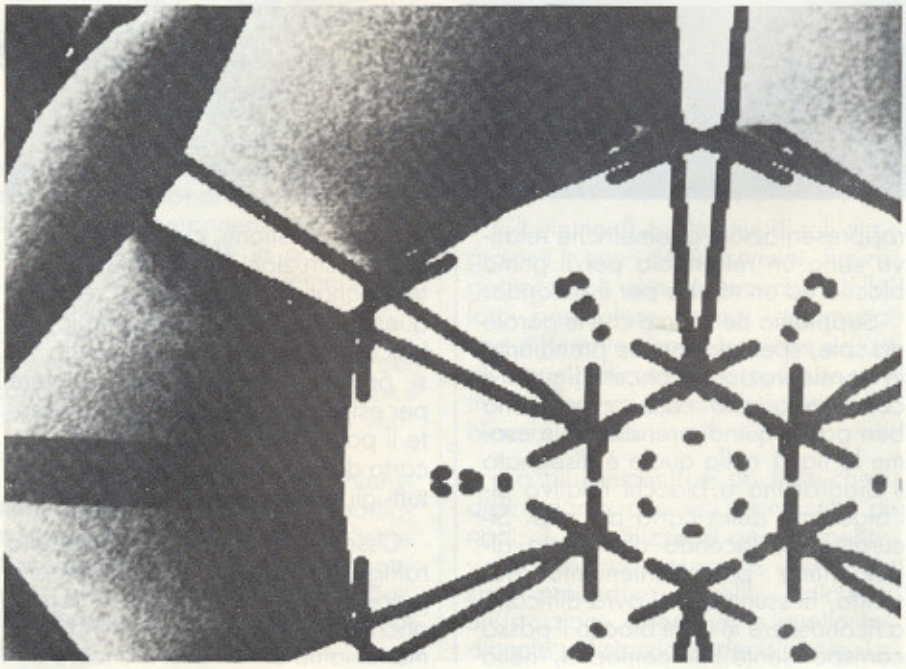
Questo tipo di struttura è più facile da gestire, anche dal punto di vista logico, dallo statement DO-...LOOP.

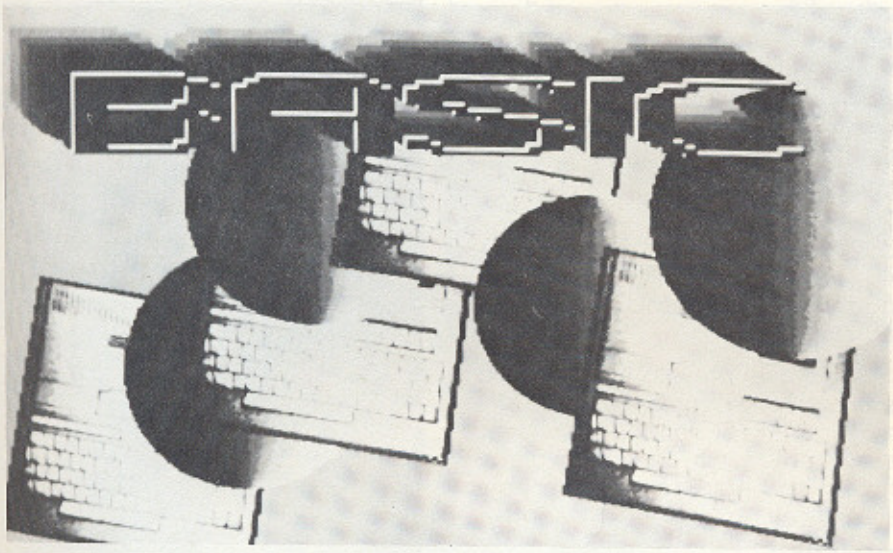
Si parla spesso di abuso del comando GOTO, perché normalmente viene utilizzato per risolvere dei problemi di impostazione. È infatti usato spesso come tappabuchi. Quando cioè il programma che si sta provando, è mancante in una qualche sua parte, la cosa più semplice da fare è di creare un'altra sezione di programma, e per arrivarci si usa appunto, un comando GOTO. È per questo motivo che i puristi della programmazione strutturata lo vorrebbero sopprimere, e non senza ragione. Infatti si trovano spesso dei programmi talmente aggrovigliati, che è quasi impossibile seguirne i concetti logici di formazione, e ciò che è peggio, è che nemmeno i programmatori, dopo qualche tempo, sono in grado di decifrarli.

Pertanto, è logico usare il comando GOTO, ma come per molte cose, con oculatezza.

```
10 CLS
20 GOTO 40
30 END
40 SCREEN4,1,1
50 PRINT"Prova GOTO"
60 SCREEN4,6,6
70 GOTO 30
```

In questo esempio, il comando GOTO viene usato come si trattasse di un GOSUB.





GR$

TIPO: Funzione

SINTASSI: GR$(n.)

COMMENTO:

Dove n. è un numero o una variabile numerica, indicante un carattere predefinito dall'utente.

Il comando GR$, visualizza il carattere utente il cui numero corrisponde all'argomento. Il carattere deve essere stato precedentemente dichiarato per mezzo del comando CLEAR (terzo parametro), e definito per mezzo del comando DEFGR$.

I carattere predefiniti dall'utente sono numerati da 0 a 127. Il comando GR$(a) equivale al comando CHR$(128+a).

```
10 PRINT GR$(10)
```

Visualizza il contenuto del carattere utente n.10

HEAD

TIPO: Funzione

SINTASSI: HEAD

COMMENTO:

Indica la direzione della tartaruga attiva.

Il numero ottenuto, sarà compreso tra 0 e 255.

La tartaruga attiva, è l'ultima indicata dal comando TURTLE.

```
10 PRINT HEAD
```

HEAD

TIPO: Istruzione

SINTASSI: HEAD TO n.

COMMENTO:

Dove n. è un numero compreso tra -255 e 255, ed indica la direzione della tartaruga. Se il numero è superiore al campo indicato, il comando HEAD assumerà come argomento il risultato della divisione n./256.

L'opzione TO, se presente, imposta il parametro in modo assoluto, in caso contrario questo sarà considerato relativo.

Un argomento positivo determina uno spostamento della tartaruga verso destra, al contrario, un numero negativo determina uno spostamento verso sinistra. Come sopra, la tartaruga attiva, è l'ultima designata per mezzo del comando TURTLE.

```
10 HEAD TO 600
```

HEX$

TIPO: Funzione

SINTASSI: HEX$(n.)

COMMENTO:

Dove n. è un numero o una variabile, compresi tra -65536 e 65536. Se l'argomento non è un numero intero, viene automaticamente privato della parte decimale.

Per mezzo di questa funzione, che ritorna una stringa di caratteri rappresentante un numero esadecimale (cioè in base 16), è possibile convertire qualsiasi numero in base 10, nel suo corrispondente in base 16.

```
PRINT HEX$(49152)
C000
OK
```

IF ... THEN

TIPO: Controllo

SINTASSI: IF espressione (THEN o GOTO) ELSE (n.lin o istr.)

COMMENTO:

Dove espressione è una qualsiasi espressione numerica o letterale che può essere VERA o FALSA.

Nel caso in cui l'espressione risulta falsa, e nella linea di istruzioni è presente il comando ELSE, il programma ignora tutte le istruzioni fino a ELSE ed esegue ciò che segue quest'ultimo comando. Se invece ELSE non è presente, il programma salta alla riga successiva.

Nel caso in cui l'espressione risulta vera, ed è presente il comando ELSE, vengono eseguiti i comandi posti di seguito all'istruzione THEN, fino al comando ELSE, quindi il programma passa alla linea successiva.

Vediamo alcune esemplificazioni:

```
IF A=B (vero) THEN PRINT "x" ELSE PRINT "y"
IF A=B (falso) THEN PRINT "x" ELSE PRINT"y"
IF A=B (vero) THEN 100
IF A=B (vero) GOTO 100
```

Nel primo caso, sul video apparirà la lettera "x", e il programma continuerà dalla linea successiva, ignorando i comandi posti di seguito all'istruzione ELSE.

Nel secondo caso, la lettera stampata sarà la "y" e pertanto saranno ignorate le istruzioni poste fra il comando THEN e il comando ELSE.

Il terzo e il quarto esempio danno lo stesso risultato di salto condizionato.

È da notare, che dopo il comando THEN possono essere presenti più istruzioni separate tra loro dai due punti e che queste devono per forza terminare con un GOTO. Il perché di tale affermazione è molto semplice: se così non fosse il programma eseguirebbe la linea successiva, in modo del tutto indifferente dal risultato dell'espressione.

Un altro particolare interessante è la possibilità di nidificare più istruzioni IF. In questo caso è bene tener presente che ogni ELSE è associato all'istruzione THEN più vicina, que-

sta però non deve essere a sua volta associata ad un altro ELSE.

L'importanza di questo comando nell'economia di un programma Basic, è fondamentale, infatti è solo grazie ad esso che noi siamo in grado di operare delle scelte. Per un uso ancora più potente di questa istruzione, è opportuno dare un'occhiata agli operatori logici AND, OR e NOT. I quali permettono di concatenare diverse espressioni legandole tra loro logicamente e per questa loro caratteristica sono spesso indispensabili nella redazione di un programma.

```
10 CLS 20 INPUT
   "Immettere un numero
   da 1 a 10"; A$
30 IF A$="0" THEN 80
40 IF A$="1"
   OR A$="5"
   OR A$="7" THEN 160
50 IF A$="6" THEN 140
60 IF A$="3"
   OR A$="9" THEN 100
70 IF A$="2"
   OR A$="4"
   OR A$="8"
   OR A$="10" THEN 120
80 PRINT "HO DETTO
   UN NUMERO COMPRESO
   TRA 1 E 10!!"
90 GOTO 20
100 PRINT A$;" È DIVISIBILE PER 3"
110 GOTO 20
120 PRINT A$;" È DIVISIBILE PER 2"
130 GOTO 20
140 PRINT A$; " È DIVISIBILE
   SIA PER 2 CHE PER 3"
150 GOTO 20
160 PRINT A$;" NON È DIVISIBILE"
170 GOTO 20
```

Come abbiamo spesso detto, i nostri programmi non puntano alla razionalità, ma a una esposizione prettamente didattica. Infatti, il programma appena descritto, è utile solo a mostrare l'uso dei comandi sopra esposti, ma non certo a risolvere il problema di trovare la divisibilità di un numero. Pertanto non utilizzatelo a questo scopo.

INKEY$

TIPO: Funzione

SINTASSI: INKEY$

COMMENTO:

INKEY$ è una variabile riservata, utilizzabile per poter ottenere il valore dell'ultimo carattere premuto.

Appena premuto un tasto qualsiasi, esclusi i tasti STOP e CTRL-C, la variabile INKEY$ assume il valore del tasto utilizzato.

```
10 CLS
20 A$=INKEY$
30 IF A$="X" THEN 50
40 GOTO 20
50 PRINT"HAI PREMUTO UNA X"
60 END
```

Dopo aver provato il programma scritto qui sopra, noterete che la funzione INKEY$ non ha eco sullo schermo. Ciò vuol dire che qualsiasi tasto viene premuto, non compare sullo schermo.

INMOUSE
INPUTMOUS

TIPO: Istruzione

SINTASSI: INMOUSEvar1,var2
INPUTMOUSEvar1,var2

COMMENTO:

Dove var1 e var2 sono le coordinate del mouse.

Per mezzo di INMOUSE è possibile leggere immediatamente la posizione del mouse, mentre utilizzando INPUTMOUSE, è possibile ottenere ciò solo tramite la pressione di uno dei due tasti.

INMOUSEx,y

INPEN
INPUTPEN

TIPO: Istruzione

SINTASSI: INPENvar.c,var.r
INPUTPENvar.c,var.r

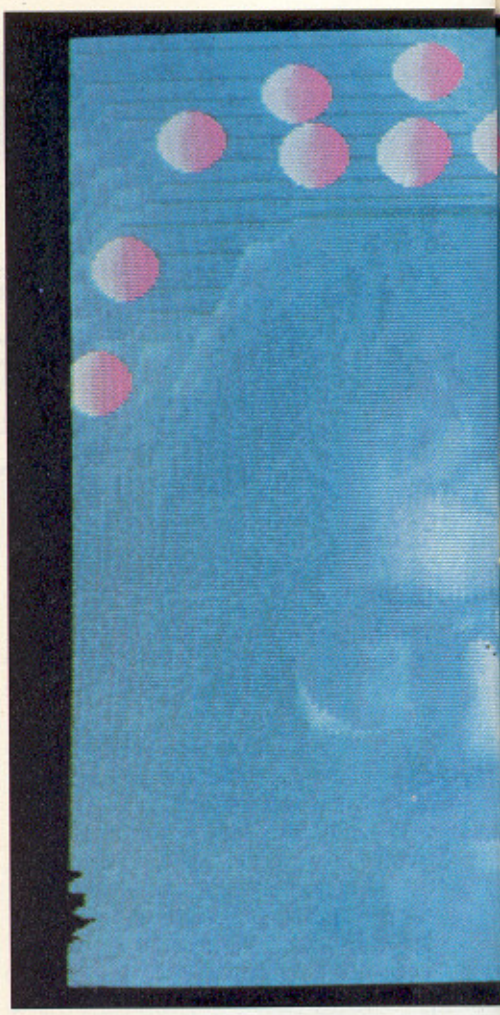
COMMENTO:

Dove var.c e var.r sono le variabili corrispondenti alle coordinate del punto indicato dalla penna ottica.

INPEN ritorna immediatamente il valore del punto indicato, mentre INPUTPEN necessita dell'interruzione del contatto per memorizzare i valori.

Il campo di valori in cui si può trovare var.c è da 0 a 319, in modo 40 colonne, e da 0 a 638 in 80 colonne. Il valore di var.r ha un campo compreso tra 0 e 199.

Se il segnale alla penna è insufficiente, il valore dato alle due variabili è pari a -1. Tale situazione si verifica nei seguenti casi:



— Cattiva regolazione della penna ottica.

— Insufficiente illuminazione del punto mirato; ciò spesso dipende dal colore usato.

— Puntamento esterno al campo utile dello schermo.

```
10 CLS
20 INPEN X,Y
30 IF X < 0 THEN 60
40 LINE (1,1)-(X,Y),RND*7.4
50 LINE -(1,199),RND*7.4
60 IF INKEY$=" " THEN END
70 GOTO 20
```

INPUT

TIPO: Istruzione

SINTASSI: INPUT"stringa car.",elenco var.
NPUT"stringa car.";elenco var.

COMMENTO:

Dove "stringa car." è una stringa di caratteri ed è facoltativa e elenco var. è un elenco di variabili consentite.

La differenza fra le due sintassi, è nella presenza di un punto e virgola o, in alternativa, di una virgola. Nel primo caso si avrà la presenza di un punto interrogativo sullo schermo, di seguito alla scritta contenuta nella stringa, mentre nel secondo caso no.

I dati dell'input devono corrispondere esattamente ai tipi di variabili elencate, in caso contrario, apparirà il messaggio d'errore "Redo from start". È da ricordarsi pure che se si usano immissioni multiple, ogni elemento deve essere separato dagli altri per mezzo di una virgola.

Le variabili possono essere delle parti di matrici, ma non delle matrici.

In caso di immissione di stringhe contenenti punteggiatura o spazi, al loro inizio o alla loro fine, è necessario includerle tra due virgolette.

```
10 CLS
20 INPUT
   "GIORNO, MESE, ANNO
   ",A$,B$,C
30 PRINT
40 PRINT A$,B$,C
50 END
```

INPUT #

TIPO: Istruzione

SINTASSI: INPUT # n.canale,elenco var.

COMMENTO:

Legge un file sequenziale del quale si precisa il canale e attribuisce i valori letti alle variabili in argomento.

Il numero del canale deve essere quello specificato nell'ordine OPEN #, il quale deve precedere tutte le prove di lettura o scrittura.

Le regole di immissione dei parametri, sono le stesse da seguire per l'immissione da tastiera.

Se si prova a leggere un file dopo che questo è terminato, si avrà un messaggio d'errore del tipo "Past End". Per evitare questo tipo d'errore si deve usare la funzione EOF.

```
10 CLS
20 OPEN"I", # 1,"FILE"
30 IF EOF(1) THEN 70
40 INPUT # 1,A$
50 PRINT A$
60 GOTO 30
70 PRINT"FINE DEL FILE"
80 END
```

INPUT$

TIPO: Funzione

SINTASSI: INPUT$(n. caratteri, # n. canale)

COMMENTO:

Ritorna una stringa di caratteri contenente il numero di caratteri letti in successione nel file indicato.

Per mezzo della funzione INPUT$, è possibile leggere un numero determinato di caratteri, su un canale prefissato, consentendoci in tale modo di conoscere esattamente il contenuto di un file.

Se non viene indicato il numero di canale, l'istruzione leggerà i dati dalla tastiera, senza che questa abbia un'eco sullo schermo. È importante tener presente che i tasti CTRL-C e STOP verranno letti come tutti gli altri e pertanto non potranno interrompere il programma in corso.

Per mezzo della tecnica appena descritta, è possibile costruire dei sottoprogrammi di input che permettono l'introduzione dei dati evitando di dover premere continuamente il tasto RETURN.

INPUT$(8, # 3)
Legge 8 caratteri dal file 1.

INPUTTURTLE

TIPO: Istruzione

SINTASSI: INPUTTURTLEvar.c,var.r

COMMENTO:

Dove var.c e var.r sono rispettivamente il valore corrispondente alla colonna e alla riga.

Per mezzo di questa istruzione, è possibile conoscere esattamente la posizione della tartaruga attiva tramite le sue coordinate.

```
10 CLS
20 DO
30 FWD 15:HEAD 20
40 INPUTTURTLE C,R: LOCATE0,0:
   PRINT C,R
50 LOOP
```

INPUTWAIT

TIPO: Istruzione

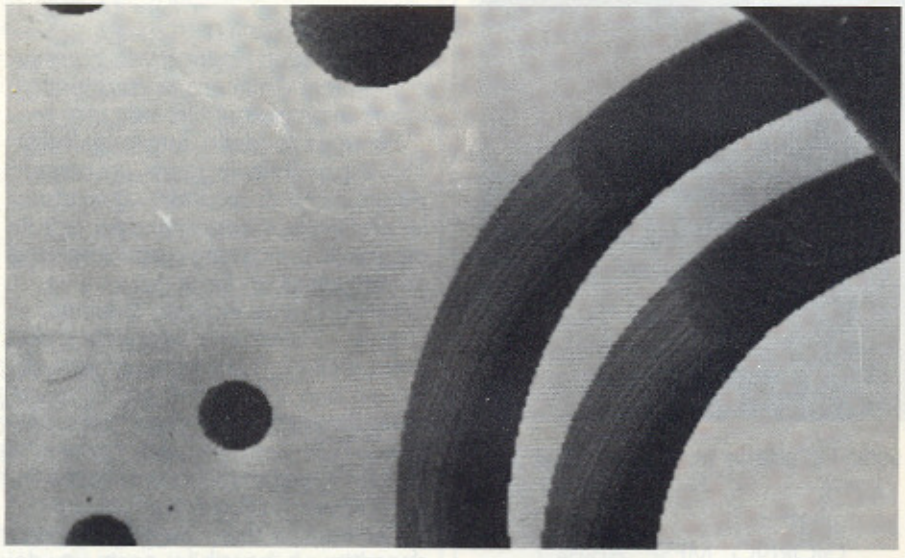
SINTASSI: INPUTWAITn.r; durata, "str. caratteri", var
INPUTWAITn.r;durata, "str.caratteri";var

COMMENTO:

Dove n.r è un numero e indica un numero di riga del programma corrente; durata è un numero e indica un tempo; str.caratteri è una stringa di caratteri e var può essere una va

```
50 PRINT"A=";A
60 INPUTWAIT
   100;5,"QUANTO FA 8*A";B
70 IF B=8*A THEN 80 ELSE CLS:
   PRINT"NON SAI
   CONTARE?":
   FOR X=0 TO 1000:
   NEXT X: GOTO 10
80 PRINT B
90 LOOP
100 END
110 CLS
120 PRINT
   "MI DISPIACE MA SEI
   TROPPO LENTO!!"
130 FOR C=0 TO 1000: NEXT C
140 GOTO 10
```

In questo piccolo gioco di riflessi si utilizza, in riga 60, l'istruzione INPU-



riabile o un elenco di variabili separate da una virgola.

Questa istruzione permette di temporizzare un input e di saltare al numero di linea in argomento, nel caso in cui non si risponda entro il tempo stabilito.

La sintassi è simile a quella dell'istruzione INPUT: anche qui abbiamo la possibilità di eliminare o di mantenere il punto di domanda dopo la stringa, semplicemente ponendo una virgola o un punto e virgola, fra la stringa di caratteri e l'elenco di variabili.

```
10 CLS
20 PRINT"PROVA DI VELOCITA"
30 DO
40 A=INT(RND*10)+1
```

TWAIT, proprio per valutare i vostri riflessi. In riga 40 l'istruzione RND viene usata in questo particolare formato per creare una generazione casuale di numeri compresi tra 1 e 10.

INSTR

TIPO: Funzione

SINTASSI: INSTR(pos.,str.1,str.2)

COMMENTO:

Dove pos. è un numero indicante la posizione di partenza ; str.1 e str.2 sono due stringhe. Se il primo parametro è mancante, il test avrà inizio dalla prima lettera.

Per mezzo di questa funzione, si può verificare se una stringa chiamata str.2 è contenuta nella stringa str.1, iniziando la comparazione dalla posizione pos.

Il risultato della funzione INSTR, sarà un numero indicante la posizione della stringa cercata. Nel caso in cui la str.2 è nulla, si avrà per risultato il numero 1, l'equivalente della posizione di partenza. Se invece il risultato è lo 0, cio ha i seguenti significai possibili:

— La str.2 non figura nella str.1.
— La str.1 è vuota.
— La posizione di partenza specificata è superiore alla lunghezza della str.1.

```
10 CLS
20 PRINT INSTR("DOVE VAI","D")
30 PRINT INSTR(4,"
   DOVE VAI", "O")
40 PRINT INSTR(6,"
   DOVE VAI", "")
50 PRINT INSTR(4,"
   DOVE VAI", "VAI")
60 END
```

Vediamo ora un altro piccolo esempio di come è possibile utilizzare la funzione INSTR.

```
10 CLS
20 PRINT "-U- menu, -P- load,
   -T- save"
30 A$=INPUT$(1)
40 PRINT A$
50 A%=INSTR("UPT",A$)
60 ON A% GOTO 100,200,300
70 GOTO 10
 .
 .
100 ...
200 ...
300 ...
```

INT

TIPO: Funzione

SINTASSI: INT(n.)

COMMENTO:

Dove n. è un numero, una variabile numerica o un'espressione, e può essere: intero, reale, o a doppia precisione.

Il risultato di questa funzione è la parte intera dell'argomento, se questo è positivo, per i numeri negativi invece, viene fornito il numero negativo immediatamente inferiore.

```
10 PRINT INT( 56.456)
   Il risultato è 56
20 PRINT INT(-56.456)
   Il risultato è -57
```

INTERVAL ON
INTERVAL OFF

TIPO: Istruzioni

SINTASSI: INTERVAL ON
INTERVAL OFF

COMMENTO:

Permette di attivare o disattivare il temporizzatore precedentemente definito per mezzo dell'istruzione ON INTERVAL.

KILL

TIPO: Comando

SINTASSI: KILL"descrittore file"

COMMENTO:

Dove descrittore file deve avere il solito formato: "n.drive:nomefile.estensione".

Cancella il file in argomento dal dischetto, e ne lascia libero e disponibile lo spazio.

Il nome della periferica non è obbligatorio, ma è necessaria l'estensione. Se non viene specificato il numero del drive, questo sarà, per default, lo 0, oppure quello impostato tramite il comando DEVICE.

KILL"1:PROVA1.BAS" Cancella il file "PROVA1.BAS dal dischetto contenuto nel drive 1.

LEFT$

TIPO: Istruzione

SINTASSI: LEFT$(str.,I)

COMMENTO:

Dove str. è una stringa o una variabile stringa; I è un numero o una variabile numerica di valore compreso tra 0 e 255.

La funzione ritorna una parte della stringa, lunga quanto indicato da I, partendo da sinistra.

```
10 PRINT LEFT$("ABCDEFGH",5)
   sarà ABCDE
20 PRINT LEFT$("ABCDEFGH",8)
   sarà ABCDEFGH
30 A=7
40 A$="ABCDEFGH"
50 PRINT LEFT$("A$",A)
   sarà ABCDEFG
```

LEN

TIPO: Funzione

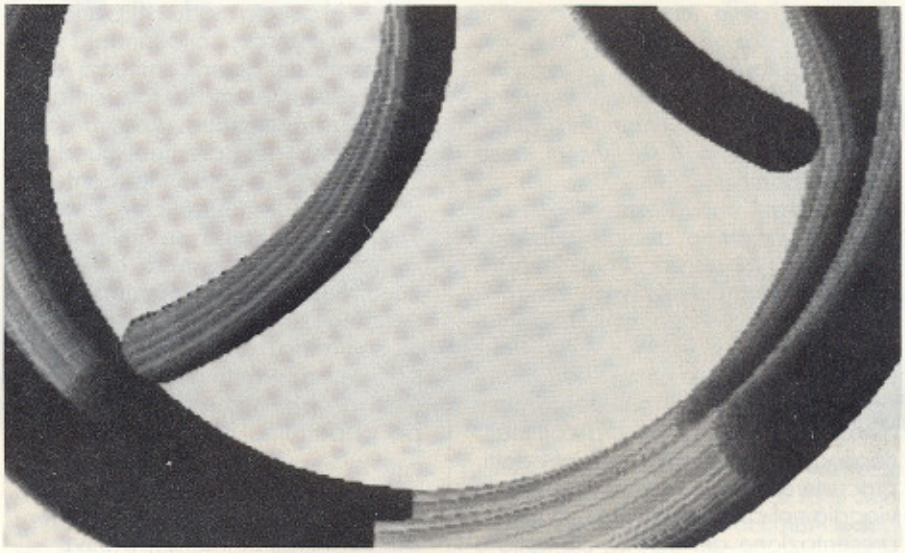
SINTASSI: LEN(str.)

COMMENTO:

Dove str. può essere sia una stringa che una variabile stringa.

Questa funzione ritorna il numero di caratteri componente la stringa.

```
10 CLS
20 INPUT"Introdurre una frase
   o una parola! ",A$
30 CLS
40 PRINT A$;
   " ha";LEN(A$);
   "caratteri!"
50 END
```



LET

TIPO: Istruzione

SINTASSI: LETvar.=espress.

COMMENTO:

Dove var. è una variabile consentita dal Basic; espress. è una espressione compatibile con il genere di variabile che compare alla sinistra dell'uguale.

La parola chiave LET può essere omessa.

Questa è una istruzione di assegnazione, serve cioè ad assegnare dei dati a delle variabili.

```
10 A$="CIAO PIPPO"
20 A=12*18/126
30 A$=B$+C$+D$
40 A=B+C+D
```

LINE

TIPO: Istruzione

SINTASSI: LINE (c1,r1)-(c2,r2) carattere, colore, sfondo, inv
LINE (c1,r1)-(c2,r2), colore

COMMENTO:

Dove c1,r1 sono le coordinate del punto di partenza e c2,r2 le coordinate del punto di arrivo. Se l'argomento carattere è presente, l'istruzione funzionerà in modo carattere, mentre se è assente funziona in modo grafico.

I campi entra i quali devono trovarsi i valori delle coordinate dipendono dal modo grafico e sono i seguenti:

Modo grafico

$0 =< c1 =< 319$ $0 =< r1 =< 199$
$0 =< c2 =< 319$ $0 =< r2 =< 199$

Modo carattere

$0 =< c1 =< 39$ $0 =< r1 =< 24$
$0 =< c2 =< 39$ $0 =< r2 =< 24$

Per mezzo dell'istruzione LINE è possibile tracciare una linea tra due punti.

```
10 CLS
20 LINE(1,9)-(15,15)"-",2,0
30 LINE-(20,35)"O",3,0
40 LINE(100,60)-(163,95),2
50 LINE-(200,200)
60 LINE-(130,190)
```