I DUE BASIC DEL PC 128

Alla ricerca di nuovi comandi

9ª parte

LINE INPUT

TIPO: Istruzione
SINTASSI: LINE INPUTstr.;var.str
LINE INPUTstr.,var.str
COMMENTO:
Permette d'introdurre in una variabile stringa una sequenza di caratteri provenienti dalla tastiera. Il numero massimo di caratteri consentiti è 255.

La presenza della virgola, al posto del punto e virgola, non determina alcun cambiamento; in ogni caso il punto interrogativo non compare sullo schermo.

È da tener presente, che l'istruzione LINEINPUT, accetta solo variabili alfanumeriche in argomento e non variabili numeriche.

```
10 LINEINPUT "PRIVA1",A$
20 PRINT A$
```

LINE INPUT

TIPO: Istruzione
SINTASSI: LINE INPUT
#n.ca.,var.str
COMMENTO:
Legge il record successivo del file in una variabile stringa prendendo tutti i caratteri.

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

LIST

TIPO: Comando
SINTASSI: LISTdescrittore
file,r.r1,n.r2
COMMENTO:

L'uso del comando LIST permette l'accesso a uno o più numeri di linea di un eventuale programma Basic contenuto in memoria. Gli output del comando LIST possono essere: lo schermo; un file; oppure una stampante. Il tipo di periferica deve essere specificata in descrittore file: se quest'ultimo è un drive, il comando si comporta come l'istruzione SAVE A.

LIST n.1	Lista la riga n.1
LIST.	Lista la riga corrente.
LIST n.1-n.2	Lista una parte di programma compreso tra n.1 e n.2.
LISTn.1-	Lista una parte di programma da n.1 n.1 fino alla fine del listato.
LIST -n.2	Lista una parte di programma partendo dalla prima linea fino a n.2.
LIST	Lista tutto il programma sul video.

LIST"LPTR:(40)",100-200 Manda alla stampante la parte di programma compresa tra la riga 100 e la riga 200. È importante notare che la stampante scriverà su una larghezza di 40 caratteri, come specificato in argomento.
LIST"0:PROVA1.BAS" Invia il programma sul file "PROVA1.BAS", del dischetto contenuto nel drive 0.

LOAD

TIPO: Comando
SINTASSI: LOADdescrittore file,R
COMMENTO:
Dove R è opzionale e permette di lanciare automaticamente l'esecuzione del programma appena questo è caricato.

Per mezzo di LOAD si carica in memoria un file residente su memoria di massa esterna. Questa operazione provoca la perdita dei programmi Basic residenti in memoria, e chiude tutti i file aperti.

La periferica di default è il drive 0, se si desidera cambiare questo valore, è sufficiente usare il comando DEVICE.

```
LOAD"prova1" LOAD"1:prova2",R
```

LOADM

TIPO: Istruzione
SINTASSI: LOADMdescrittore
file,traslazione,R
COMMENTO:
Permette di caricare in memoria un file in linguaggio macchina, precedentemente salvato per mezzo del comando SAVEM o da un assemblatore.

Il caricamento viene effettuato nel banco corrente, ma se viene indicato uno spostamento, questo viene eseguito seguendo le indicazioni date in argomento.

Anche in LOADM, se R viene specificata, si avrà l'immediata esecuzione del programma, all'atto del caricamento.

LOADP

TIPO: Istruzione.
SINTASSI: LOADPdescrittore
file,elemento matrice

COMMENTO:

Permette di caricare in una matrice, l'immagine contenuta in un file.

All'interno di una matrice numerica si possono immagazzinare più disegni, questi infatti vengono compattati, basta che la matrice stessa sia sufficientemente estesa. A tale proposito, la matrice deve essere già stata dichiarata al momento dell'uso del comando LOADP, e deve essere del tipo intero.

Per mezzo di LOADP, si ottengono dei risultati simili a quelli ottenibili con il comando GET.

Se l'estensione non viene definita, questa sarà considerata per default come .MAP.

È di particolare interesse sapere che per mezzo di LOADP è possibile manipolare anche le schermate salvate con COLORPAINT.

```
DIM A%(2000)
LOAD"0:prova1",A%(2000)
PUT(0,0),A%(2000)
```

In questo programmino d'esempio si può notare che il file immagine "prova1.MAP", viene caricato dal drive 0 nella matrice A% a partire dall'elemento 999.

LOC

TIPO: Funzione
SINTASSI: LOC(n.c)
COMMENTO:

Dove n.c è un numero indicante un canale.

LOC indica in che punto avviene la scrittura (o la lettura), in un file aperto tramite il numero di canale in argomento.

Se il file è ad accesso sequenziale, tramite LOC sarà possibile avere di ritorno il numero di settori letti o scritti a partire dall'apertura del file.

Se invece il file è ad accesso diretto, LOC indica il numero del record che segue quello che è appena stato letto da GET# o scritto da PUT#.

LOC(1) Ritorna la posizione di lettura o scrittura in corso sul file 1.

LOCATE

TIPO: Istruzione
SINTASSI: LOCATEcl,r,cursor
COMMENTO:

Dove cl e r possono essere dei numeri o delle variabili numeriche consentite dal Basic che rappresentano le nuove coordinate, verticali e orizzontali del cursore.

I campi d'impiego delle due variabili sono rispettivamente 39 (oppure 79) e 24.

Cursor è un numero che determina, al cambiare del suo valore, la presenza o meno del cursore:

0 Cursore invisibile.

1 Cursore visibile.

Le istruzioni grafiche (PSET, BOX, BOXF, LINE), che indirizzano punti elementari del disegno (pixel), non modificano la posizione del cursore del testo.

```
10 CLS
20 FOR A=1 TO 200
30 LOCATE RND*39,RND*23
40 PRINT"."
50 NEXT A
60 END
```

LOF

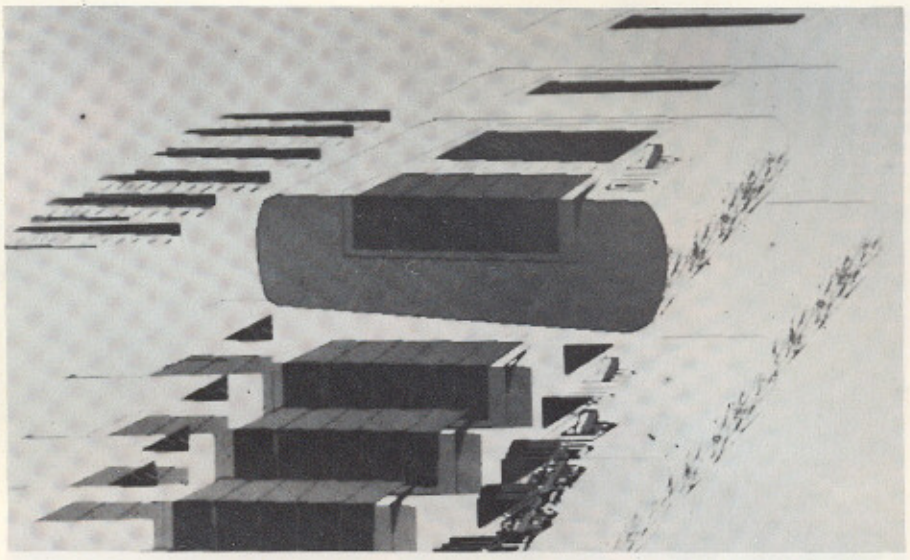
TIPO: Funzione
SINTASSI: LOF(n.canale)
COMMENTO:

Dà la posizione dell'ultimo file aperto tramite il canale in argomento.

Se il file è ad accesso sequenziale, LOF ritorna il numero dell'ultimo settore del file.

Se il file è ad accesso diretto, LOF ritorna il numero dell'ultimo record.

LOF(2) Ritorna la posizione dell'ultimo record del file 2.



LOG

TIPO: Funzione matematica
SINTASSI: LOG(n.)
COMMENTO:

Dove n. è un numero o una variabile numerica consentita.

La funzione LOG ritorna il logaritmo neperiano del numero in base e.

```
 10 CLS
 20 SCREEN1,6,6
 30 LINE (0,100)-(319,100),4
 40 LINE (0,75)-(319,75),0
 50 PSET (0,9)"1",0
 60 PSET (0,12)"0",4
 70 PSET (3,13)"1",4
 80 PSET (1,100-25*LOG(1/25)),4
 90 FOR I=2 TO 319
100 LINE-(I,100-25*LOG(I/25)),4
110 NEXT I
120 END
```

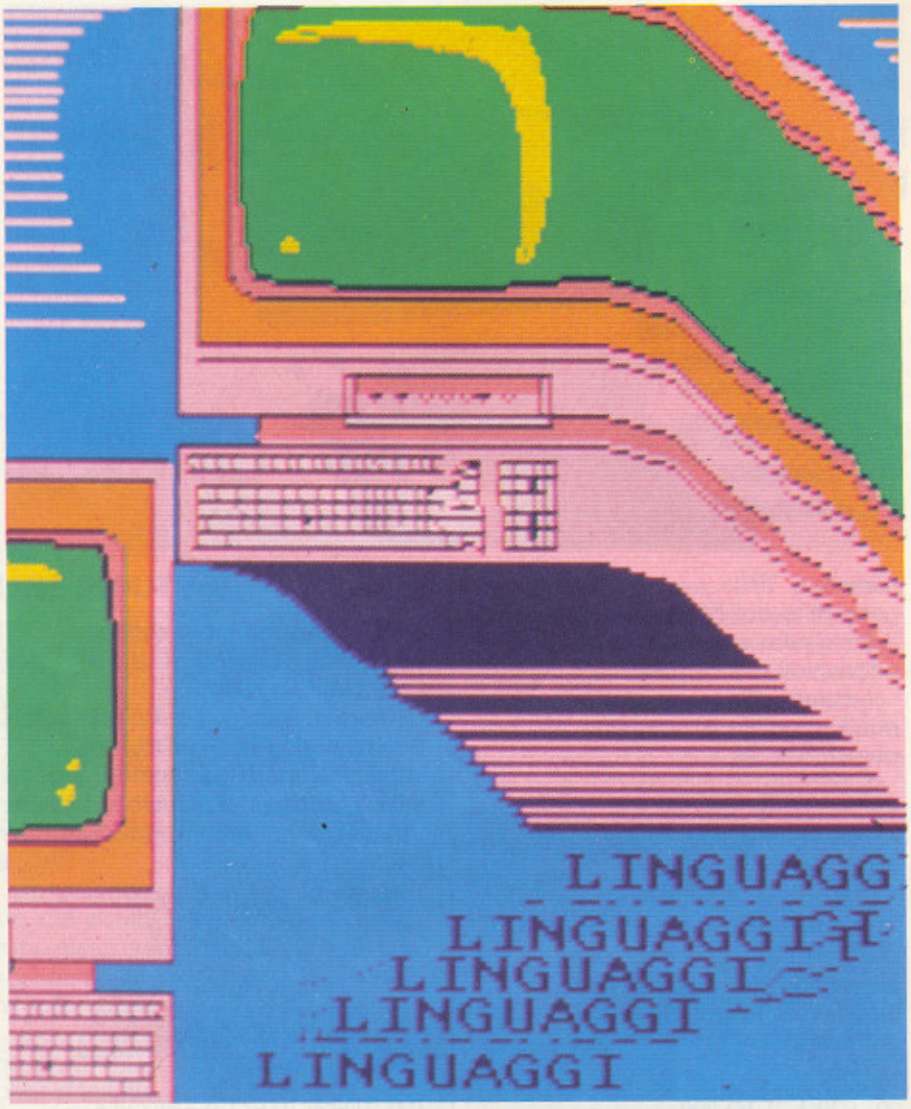
LSET

TIPO: Istruzione
SINTASSI: LSETvar.str=str.caratteri
COMMENTO:

Dove var.str è il nome di una variabile stringa che è stata in precedenza definita per mezzo di una specifica FIELD. Mentre str.caratteri è un'espressione stringa da porre nel campo identificato da var.str.

Per mezzo di questa istruzione è possibile spostare dei dati nella memoria di transito di un file ad accesso casuale, in preparazione di una specifica PUT.

Se str.caratteri, richiede meno byte di quelli indicati per var.str nella spe-



cifica FIELD, LSET allinea a sinistra la stringa nel campo e RSET la allinea a destra (si usano degli spazi per riempire le posizioni in eccesso). Se str.caratteri è più lunga di var.str, vengono ommessi i caratteri a destra.

I valori numerici devono essere convertiti in stringhe, prima di LSET o RSET. Vedi quanto segue:

MKI$ ( ) Per un numero intero.
MKS$ ( ) Per un numero a precisione singola.
MKD$ ( ) Per un numero a precisione doppia.

LSET A$=PROVA1$+PROVA2$
Assegna ad A$ la concate- nazione di PROVA1$ e PROVA2$.

MAX

TIPO: Funzione
SINTASSI: MAX(elenco numeri)
COMMENTO:
Dato un elenco di numeri, per mezzo della funzione MAX(), è possibile ricavarne il numero più grande.

```
10 PRINT MAX(1,2,4,6,9,20) Ritorna il
                           numero 20
```

MERGE

TIPO: Comando
SINTASSI: MERGEdescrittore file,R
COMMENTO:
Il comando MERGE, permette di fondere un programma contenuto in memoria con uno residente su memoria di massa.

Il programma contenuto nella memoria di massa, deve essere stato salvato con SAVEA, e cioè in formato ASCII.

Se alcuni numeri di riga del programma caricato, corrispondono ai numeri di linea del programma residente, quet'ultime verranno cancellate e sostituite dalle nuove linee.

Se l'opzione R è presente, appena terminato di caricare il file, il nuovo programma verrà messo in esecuzione automaticamente.

MERGE"1:PROVA1"	Carica il programma PROVA1 dal dischetto contenuto nel drive 1, e lo accoda al programma in memoria.
MERGE"PROVA1"	Carica in coda al programma residente il programma PROVA1.

MID$

TIPO: Funzione
SINTASSI: MID$(str.,pos.,lungh.)
COMMENTO:
Dove str. è una stringa o una variabile stringa; pos. è un numero o una variabile numerica consentita, compresa tra 1 e 255; lungh. è un numero o una variabile numerica consentita, compresa tra 0 e 255.

La funzione MID$, permette di estrarre una parte di stringa, della dimensione specificata nell'argomento lungh., dalla stringa str., a partire dalla posizione pos.

Se il parametro lungh. è mancante, o troppo grande, la funzione analizza la stringa fino alla sua fine.

Se il parametro posizione è maggiore della lunghezza della stringa, la funzione ritorna una stringa nulla.

```
10 CLS
20 A$="ABCDEFGHILMNO"
30 PRINT A$
40 PRINT
50 FOR A=0 TO 12
60 B=INT(RND*12)+1
70 B$=MID$(A$,B,1)
80 PRINT B$
90 NEXT A
100 END
```

MID$

TIPO: Istruzione
SINTASSI: MID$(var. str., inizio, lungh.) = str.caratteri

COMMENTO:
Dove var.str. è una variabile stringa; inizio è un numero o una variabile numerica di tipo intero; lunghezza è un numero o una variabile numerica di tipo intero; str.caratteri può essere sia una stringa di caratteri che una variabile stringa.

L'istruzione MID permette di cambiare una stringa, sostituendone una sua parte con un'altra stringa.

Il parametro lungh. è facoltativo, e indica la lunghezza della parte da sostituire.

Se la stringa di sostituzione è più lunga della stringa da sostituire, di questa viene usata solo la parte iniziale.

```
10 CLS
20 A$="abcdefghijklmnopqr"
30 PRINT A$
40 MID$(A$,9,4)="1234"
50 PRINT A$
60 END
```

MIN

TIPO: Funzione
SINTASSI: MIN(elenco numeri)
COMMENTO:
MIN ritorna il numero dal valore più basso, contenuto in un elenco.

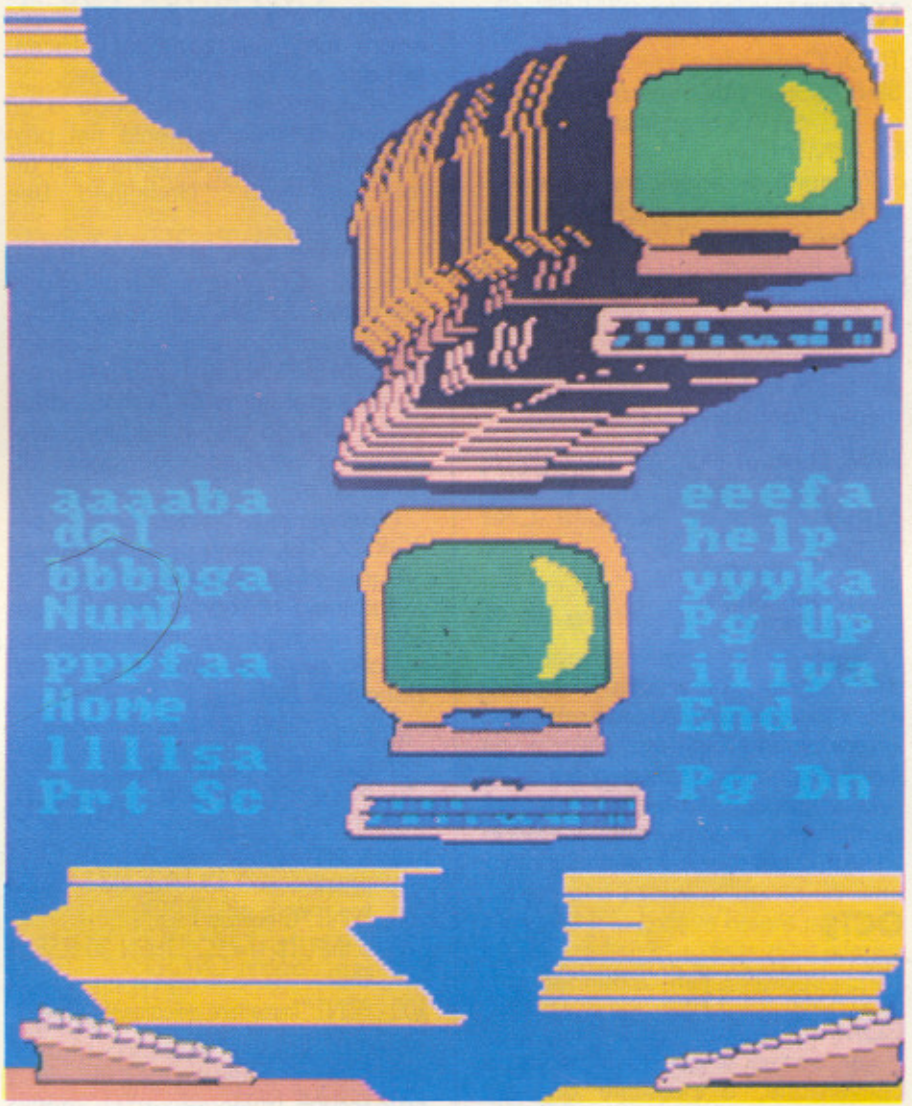
MKD$
MKI$
MKS$

TIPO: Funzioni
SINTASSI: MKD$(numero)
MKI$(numero)
MKS$(numero)
COMMENTO:
Per mezzo delle funzioni sopra elencate, è possibile convertire i valori di tipo numerico in valori di tipo stringa.

La scelta delle diverse funzioni, è determinato dal tipo di dato da convertire:

MKI$ espressione intera,
MKS$ espressione a precisione singola,
MKD$ espressione a precisione doppia.

Qualsiasi valore numerico posto nella memoria di transito di un file ad accesso casuale con una specifica LSET o RSET, deve essere convertito in stringa.



Per ulteriori approfondimenti, guardare le istruzioni LSET e RSET.

MOTOR

TIPO: Istruzione
SINTASSI: MOTORon MOTORoff
COMMENTO:
Permette di accendere e spegnere, da programma, il registratore a cassette.

MOTOR on Accende il motore.
MOTOR off Spegne il motore.

MTRIG

TIPO: Funzione
SINTASSI: MTRIG(n.)
COMMENTO:
Dove n. è un numero e può essere 0 oppure 1.

La funzione MTRIG verifica la condizione dei pulsanti del mouse:
0 pulsante sinistro,
1 pulsante destro.

Il risultato è uguale a -1 se il pulsante designato è premuto, altrimenti è uguale a 0.

```
10 CLS
20 DO
30 IF MTRIG(0) THEN A%=A%+1: CLS:
   PRINT A%
40 IF MTRIG(1) THEN A%=A%-1: CLS:
   PRINT A%
50 IF MTRIG(0) AND MTRIG(1) THEN
   END
60 LOOP
70 END
```

Notare l'uso della variabile intera A% in riga 30 e riga 40.

NAME

TIPO: Comando
SINTASSI: NAMEdescrittore file1 AS descrittore file2
COMMENTO:
Permette di modificare il nome di un file residente su dischetto.
Il descrittore file1 contiene le specifiche del file di cui si deve cambiare il nome, e deve assolutamente trovarsi sul dischetto.
Il descrittore file2 è il nuovo nome del file, e non deve essere assolutamente giàpresente sul dischetto.

```
NAME"1:prova1.BAS" AS "prova2.BAS"
```

NEW

TIPO: Comando
SINTASSI: NEW
COMMENTO:
Cancella il programma Basic residente in memoria e tutte le sue variabili, comprese quelle riservate con il l'istruzione COMMON.
NEW però non chiude i canali aperti e non cambia le specifiche impartite per mezzo del comando CLEAR.

OCT$

TIPO: Funzione di conversione
SINTASSI: OCT$(n.)
COMMENTO:
Per mezzo di questa funzione di conversione, è possibile ottenere una stringa rappresentante il valore ottale dell'argomento decimale.
In altre parole si può ottenere il valore in ottale di un qualsiasi numero decimale posto in argomento.
Per la conversione in esadecimale guardare la funzione HEX$.

```
10 PRINT OCT$(24) 30
```

ON ERROR

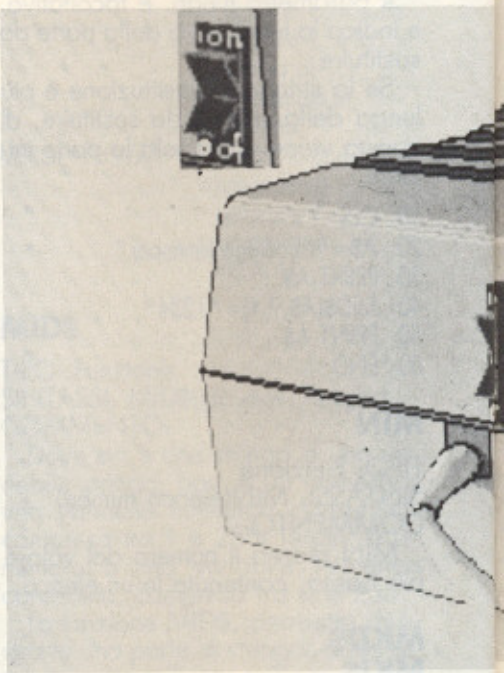
TIPO: Istruzione
SINTASSI: ON ERROR GOTO n.riga
COMMENTO:
Dove n.riga è un numero che permette l'intercettazione degli errori, e l'invio del programma alla prima riga costituente un sottoprogramma di trattamento degli errori stessi. Ciò permette di non far interrompere un programma a causa di un eventuale errore, ma di utilizzarlo come facente parte del programma stesso.
Se n.riga non corrisponde a un numero di linea esistente nel programma, il computer genera un errore del tipo "Undefined Line Number".
I comandi ON ERROR GOTO 0; RESUME; CLEAR; annullano l'effetto del comando ON ERROR GOTO.
È consigliabile utilizzare la specifica ON ERROR GOTO 0, a supporto del programma di gestione degli errori, proprio per individuare quel genere d'errori non correggibili direttamente.
Normalmente l'istruzione ON ERROR GOTO, viene posta in testa ai programmi, o all'inizio delle parti di programma da controllare. Per definire l'area di programma interessata, alla sua fine verra posta l'istruzione RESUME.

```
10 CLS
20 ON ERROR GOTO 100
30 INPUT"COLORI DELLO SFONDO -0--15-";C
40 SCREEN,C
50 PRINT
60 PRINT"C ontinua F ine";
70 IF INPUT$(1)="C" THEN 30 ELSE END
100 PRINT"Immettere unicamente -0- e -17-!"
110 PRINT
120 RESUME NEXT
```

ON-GOSUB ON-GOTO

TIPO: Istruzioni
SINTASSI: ONn. GOSUBn.r1, n.r2, ... n.rn
ONn. GOTOn.r1, n.r2, ... n.rn
COMMENTO:
Dove n. è un numero o un espressione o una variabile e deve essere compreso tra 0 e 255. Se n. non è di tipo intero, il suo valore verrà arrotondato all'intero più vicino.
Il compito delle istruzioni in questione, è di permettere dei salti alle linee in argomento, dipendenti dal risultato dell'espressione.
Pertanto il valore di n. determinerà il salto ad una delle linee di programma indicate in argomento. Per essere individuata, questa dovrà avere come posizione, nella lista seguente il comando GOSUB o GOTO, lo stesso numero corrispondente al valore di n.
Se il valore di n. è 0 o maggiore del numero di elementi della lista (ma minore o uguale a 255), il Basic continua con la istruzione seguente.

```
10 CLS
20 FOR A=0 TO 5
```



```
30 B%=INT(RND*6)+1
40 ON B% GOSUB 100,110,120,130,140,150
60 NEXT A
70 END
100 PRINT"Riga 1"
101 RETURN 110 PRINT"Riga 2"
111 RETURN
120 PRINT"Riga 3"
121 RETURN
130 PRINT"Riga 4"
131 RETURN
140 PRINT"Riga 5"
141 RETURN
150 PRINT"Riga 6"
151 RETURN
```

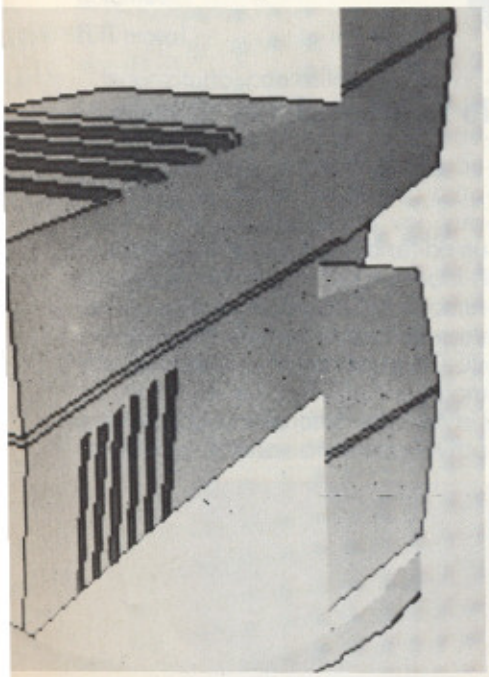
ON INTERVAL

TIPO: Istruzione
SINTASSI: ON INTERVAL=n. GOSUBn.riga
ON INTERVAL=n. GOTOn.riga

COMMENTO:

Dove n. è un numero indicante un tempo. Tale indicazione viene data in decimi di secondo.

Il comando INTERVAL ON ordina al clock interno di iniziare un conteggio fino al numero specificato in argomento. Quando viene raggiunto il valore determinato dal comando, il computer interrompe quanto sta facendo, e passa l'esecuzione



alla linea indicata di seguito all'istruzione GOSUB, oppure GOTO.

Mentre il clock sta contando, il programma viene svolto normalmente.

Il comando GOSUB, se utilizzato, deve corrispondere ad un comando RETURN posto al termine del sottoprogramma da eseguire.

Per chiudere il contatore usare l'istruzione INTERVAL OFF.

```
10  CLS
20  ON INTERVAL =20 GOSUB 380
30  Print "Passo 1"
40  FOR A=0 TO 150 :NEXT A
50  Print "Passo 2"
60  FOR A=0 TO 150 :NEXT A
70  Print "Passo 3"
80  FOR A=0 TO 150 :NEXT A
90  Print "Passo 4"
100 FOR A=0 TO 150 :NEXT A
110 Print "Passo 5"
120 FOR A=0 TO 150 :NEXT A
130 Print "Passo 1"
140 FOR A=0 TO 150 :NEXT A
150 Print "Passo 6"
160 FOR A=0 TO 150 :NEXT A
170 Print "Passo 7"
180 FOR A=0 TO 150 :NEXT A
190 Print "Passo 8"
200 FOR A=0 TO 150 :NEXT A
210 Print "Passo 9"
220 FOR A=0 TO 150 :NEXT A
230 Print "Passo 10"
240 FOR A=0 TO 150 :NEXT A
250 Print "Passo 11"
260 FOR A=0 TO 150 :NEXT A
270 Print "Passo 12"
280 FOR A=0 TO 150 :NEXT A
290 Print "Passo 13"
300 FOR A=0 TO 150 :NEXT A
310 Print "Passo 14"
320 FOR A=0 TO 150 :NEXT A
330 Print "Passo 15"
340 FOR A=0 TO 150 :NEXT A
350 Print "Passo 16"
360 FOR A=0 TO 150 :NEXT A
370 END
380 PRINT"Interruzione causata dal
    comando ON INTERVAL"
390 RETURN
```

ON KEY

TIPO: Istruzione
SINTASSI: ON KEY = carattere GOSUBn.riga
ON KEY = carattere GOTO n.riga

COMMENTO:

Dove carattere è una costante di un solo carattere, oppure, se composto da più caratteri, il primo carattere della sequenza; n.riga è un numero corrispondente ad una linea *del programma corrente*.

Permettono delle interruzioni logiche di salto, determinate dalla pressione di uno specifico tasto della console.

Il comando GOSUB, se utilizzato, deve corrispondere ad un comando RETURN posto al termine del sottoprogramma da eseguire.

Per mezzo di questa istruzione è possibile indicare fino a dieci tasti diversi: questi però non possono a loro volta essere letti dall'istruzione INKEY$.

```
10  CLS
20  PRINT"Per sospendere l'esecuzione
    premere S"
30  PRINT"Per riattivare l'esecuzione
    premere C"
40  PRINT"Per uscire dall'esecuzione
    premere Q"
50  ON KEY="S" GOTO 130
60  ON KEY="C" GOTO 80
70  ON KEY="Q" GOTO 140
80  DO
90  A%=A%+1
100 LOCATE15,10
110 PRINT A%
120 LOOP
130 GOTO 130
140 END
```

Notare il loop infinito in riga 130, e il contatore in linea 90.

ONPEN

TIPO: Istruzione
SINTASSI: ONPEN GOSUBn.1, n.2, ... n.n

COMMENTO:

Dove n.1,...n.n è una lista di numeri di linea del programma corrente.

Le istruzioni considerate, permettono un salto condizionato al numero di linea in argomento. Dove il numero di linea deve occupare, nella sequenza, il posto relativo al valore numerico assunto dall'area selezionata.

Per poter essere utilizzata, la istruzione ONPEN, deve essere preceduta dalla definizione delle aree interessate per mezzo dell'istruzione PEN.

Se la selezione viene fatta al di fuori delle aree selezionate il programma non causerà nessun salto.

```
10  CLS
20  PEN0;(10,15)-(30,35)
30  BOXF(10,15)-(30,35),3
40  PEN1;(40,15)-(60,35)
50  BOXF(40,15)-(60,35),4
60  PEN2;(70,15)-(90,35)
70  BOXF(70,15)-(90,35),2
80  DO
90  ONPEN GOSUB120,150,180
100 LOOP
110 END
120 LOCATE18,12
130 PRINT"AREA 1"
140 RETURN
150 LOCATE18,12
160 PRINT"AREA 2"
170 RETURN
180 LOCATE18,12
190 PRINT"AREA 3"
200 RETURN
```