olivetti
PRODEST

PC 128

Manuale per l'utente

Olivetti Prodest
Via Caldera, 21 - 20153 Milano (MI)
Telefono 02/452731

Guida di installazione

Introduzione

Questa sezione introduce al micro–computer OLIVETTI PRODEST PC 128.

Nella sua prima parte, essa permette di acquisire i principi fondamentali per l'utilizzo del PC 128 e, nella seconda parte, introduce alla programmazione in linguaggio BASIC.

Si constaterà che non vi è alcun bisogno di conoscere l'informatica per utilizzare il PC 128, questo grazie alla presentazione molto chiara delle scelte da effettuare nei "menu" che verranno di volta in volta visualizzati.

Disponendo di una penna ottica o di un mouse, le scelte si effettuano puntando il tasto su video o l'icona corrispondente.

Si apprezzerà la risposta meccanica della tastiera ed il fascino della tavolozza colori, che permette di scegliere 16 colori e 4096 sfumature possibili.

Grazie al registratore incorporato vengono eliminati i normali collegamenti volanti.

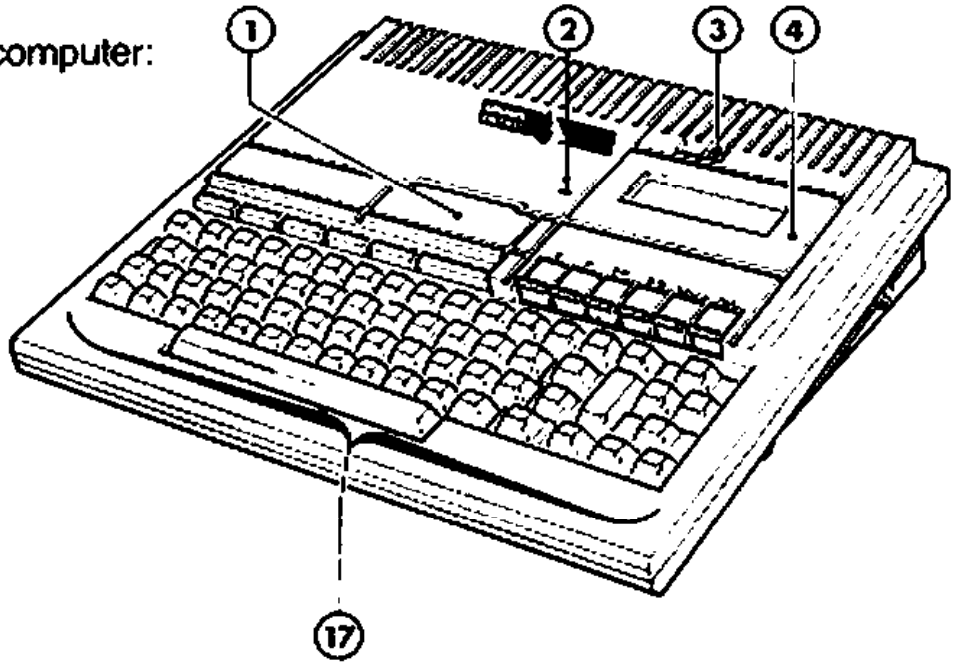
Gli amanti della programmazione scopriranno un linguaggio molto potente, il BASIC 128 MICROSOFT®, che sfrutta totalmente la capacità di memoria del PC 128, e offre un'ampia gamma di possibilità e di modalità grafiche.

Presentazione

Nella confezione del PC 128 sono presenti:

– l'unità centrale PC 128,
– il cavo d'allacciamento al televisore.

Descrizione del micro–computer:



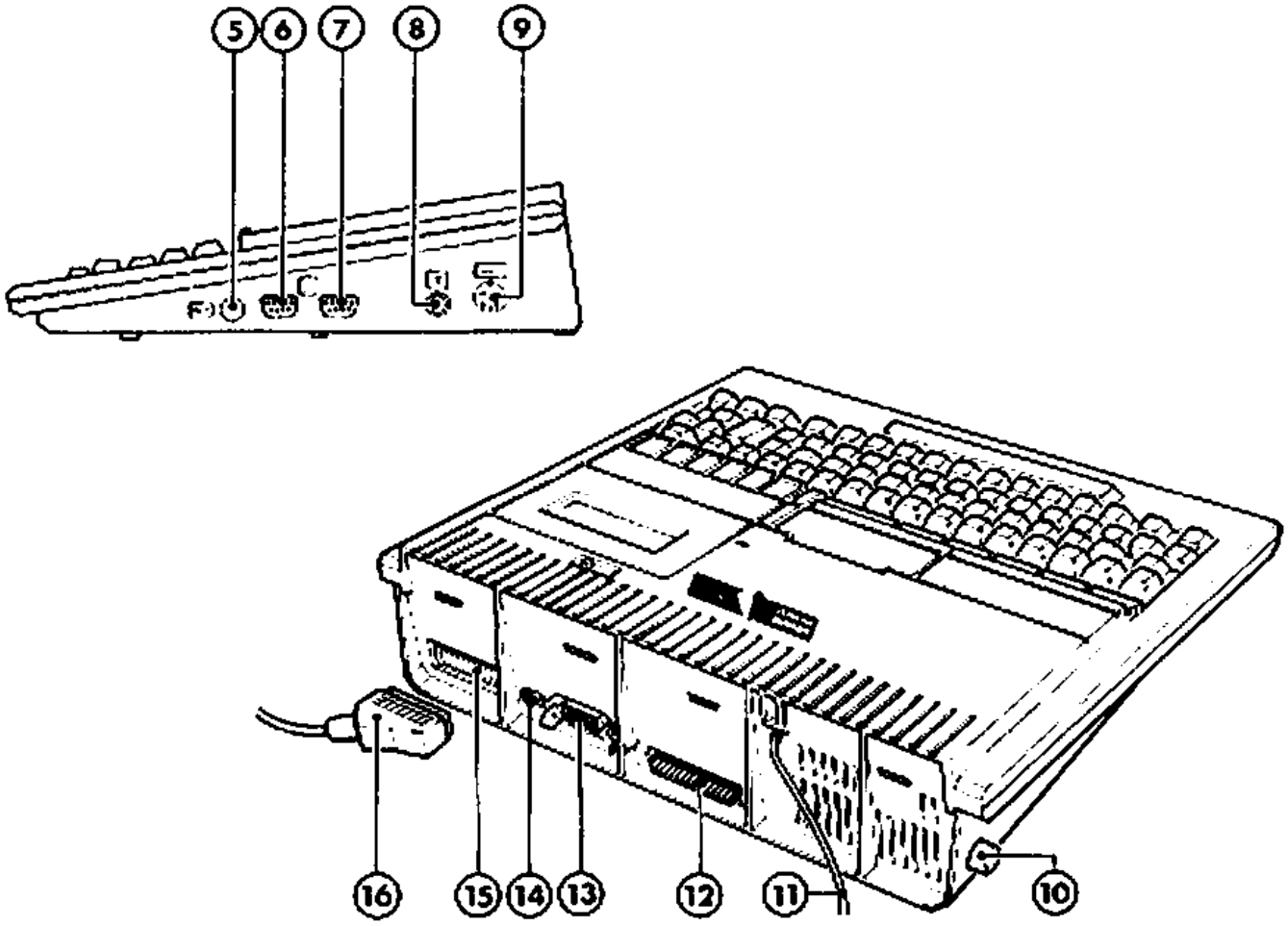


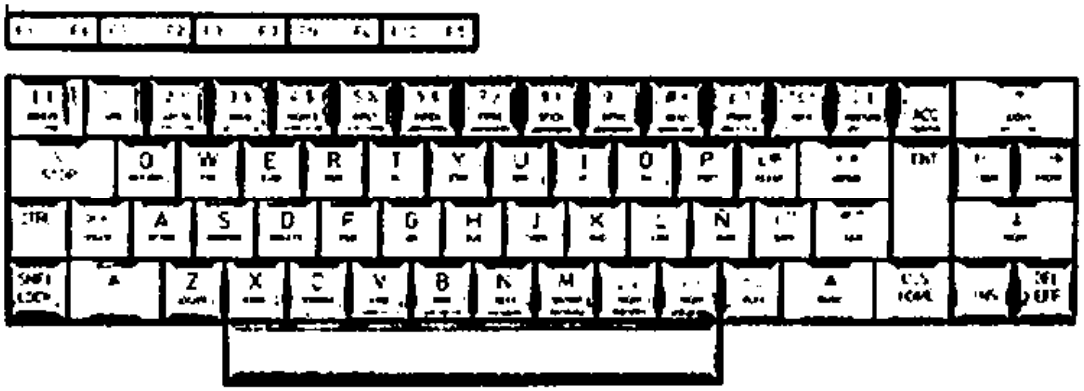
Fig.1

1 – Alloggiamento per la cartuccia
2 – Spia di acceso–spento
3 – Contagiri del registratore
4 – Registratore
5 – Tasto di reinizializzazione
6 – Connettore del primo joystick, numero 0 in BASIC, o del mouse
7 – Connettore del secondo joystick, numero 1 in BASIC
8 – Uscita antenna UHF PAL
9 – Presa per allacciamento penna ottica
10 – Tasto d'accensione
11 – Cavo d'alimentazione
12 – Connettore dell'espansione
13 – Uscita stampante
14 – Uscita AUDIO
15 – Uscita cavo per l'allacciamento televisore (presa SCART)
16 – Spina SCART
17 – Tastiera meccanica

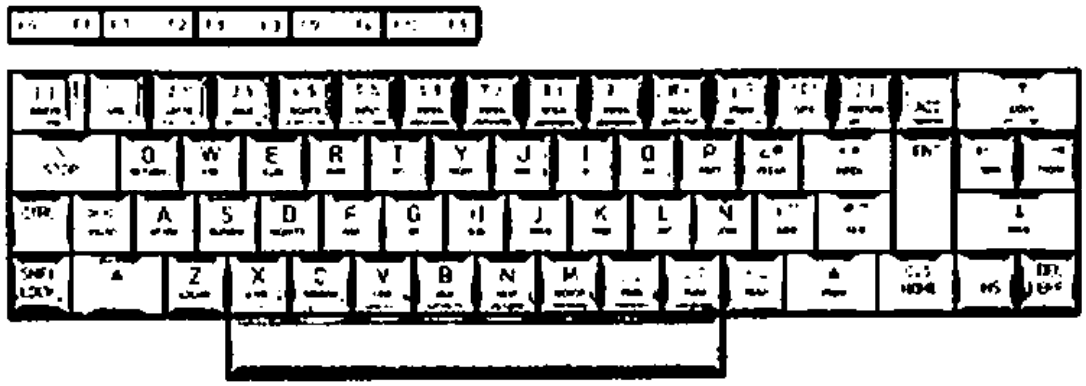
La tastiera

La tastiera si compone di quattro parti:

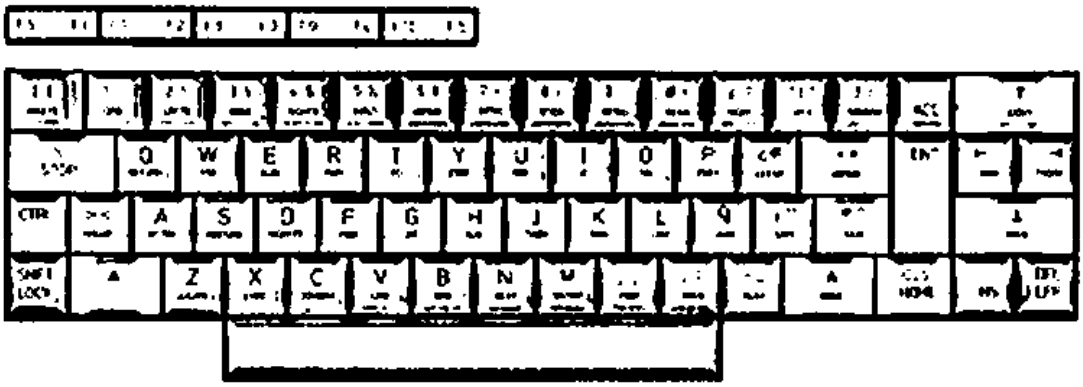
– La tastiera QWERTY, identica a quella di una macchina per scrivere, con il tasto SHIFT–LOCK munito di un led rosso e che permette di selezionare sia i caratteri maiuscoli e i numeri, quando il led rosso è acceso, sia i caratteri minuscoli.



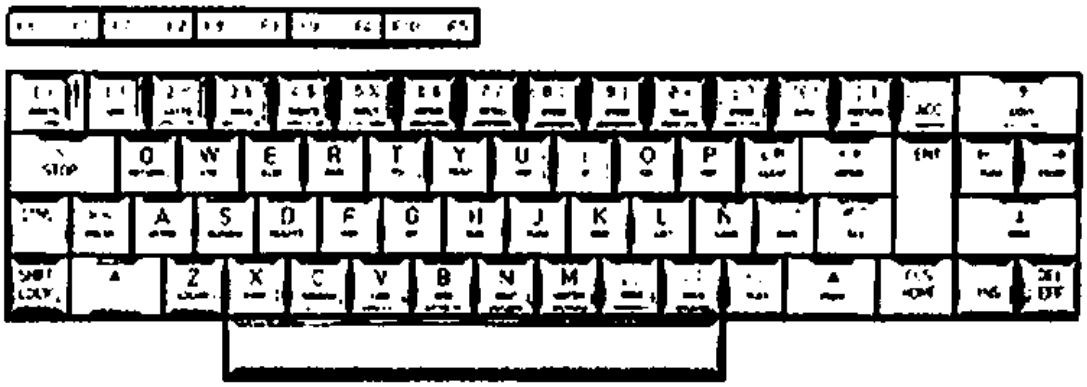
– I tasti di simboli e di accenti che non subiscono l'effetto del tasto SHIFT–LOCK. Per ottenere i caratteri o i comandi rappresentati sulla tastiera con un determinato colore, premere contemporaneamente il tasto ▲ dello stesso colore ed il tasto desiderato.



– I tasti speciali, come i tasti con le frecce, il tasto ENT, i tasti ACC, INS, EFF, CLS HOME, STOP, CTRL e i tasti funzione da F1 a F10, sono i tasti di comando di alcune funzioni del PC 128.



– I tasti recanti le istruzioni o le funzioni BASIC, direttamente accessibili premendo simultaneamente il tasto blu BASIC e il tasto corrispondente.



installazione e allacciamento

Il primo passo consiste nel collegare il PC 128 al televisore o al monitor tramite il cavo di allacciamento al televisore.

Per questa operazione è necessario che il televisore sia spento.

Collegare uno qualsiasi dei due capi del cavo al televisore e l'altro al PC 128, inserendoli nelle prese corrispondenti.

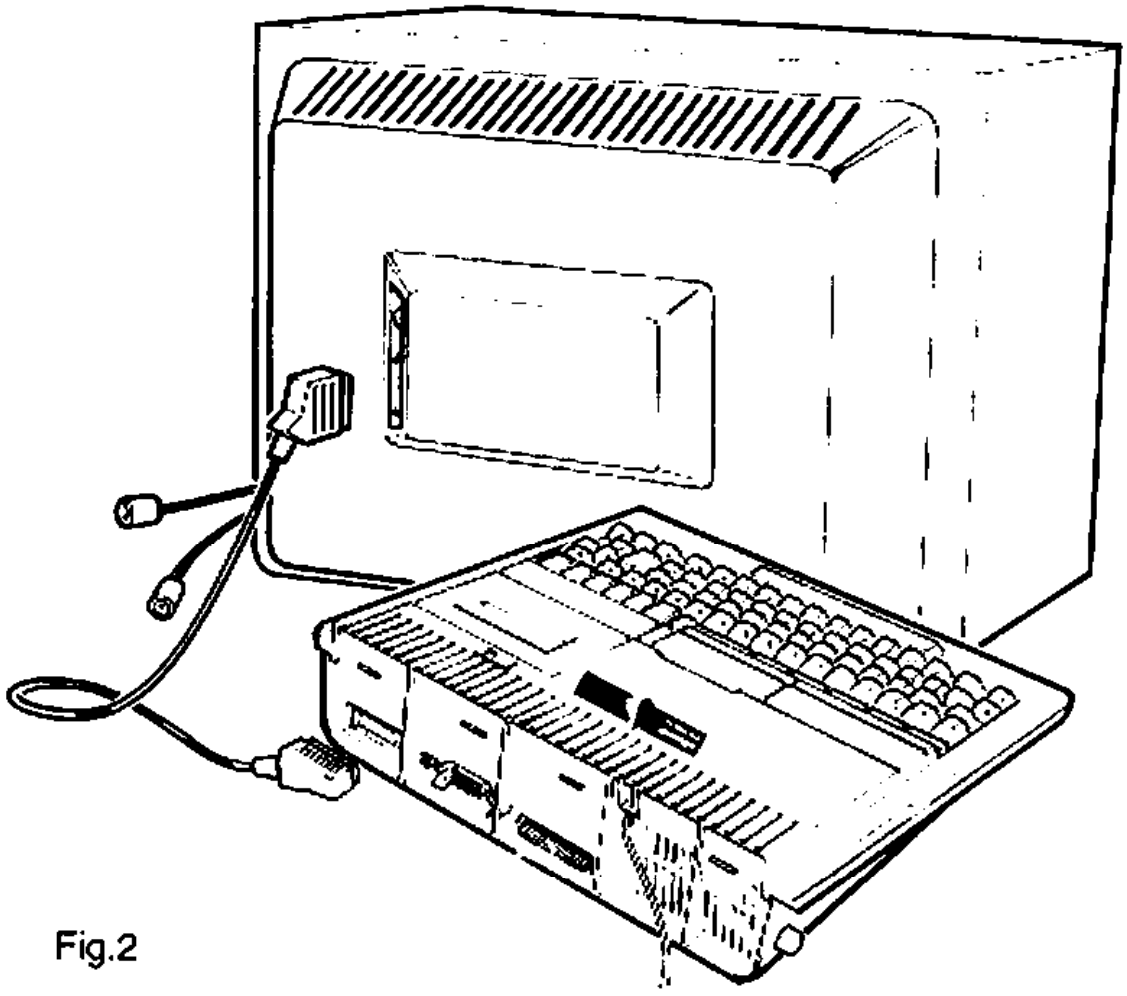


Fig.2

Allacciare quindi la spina d'alimentazione del PC 128 a 220V, 50Hz.

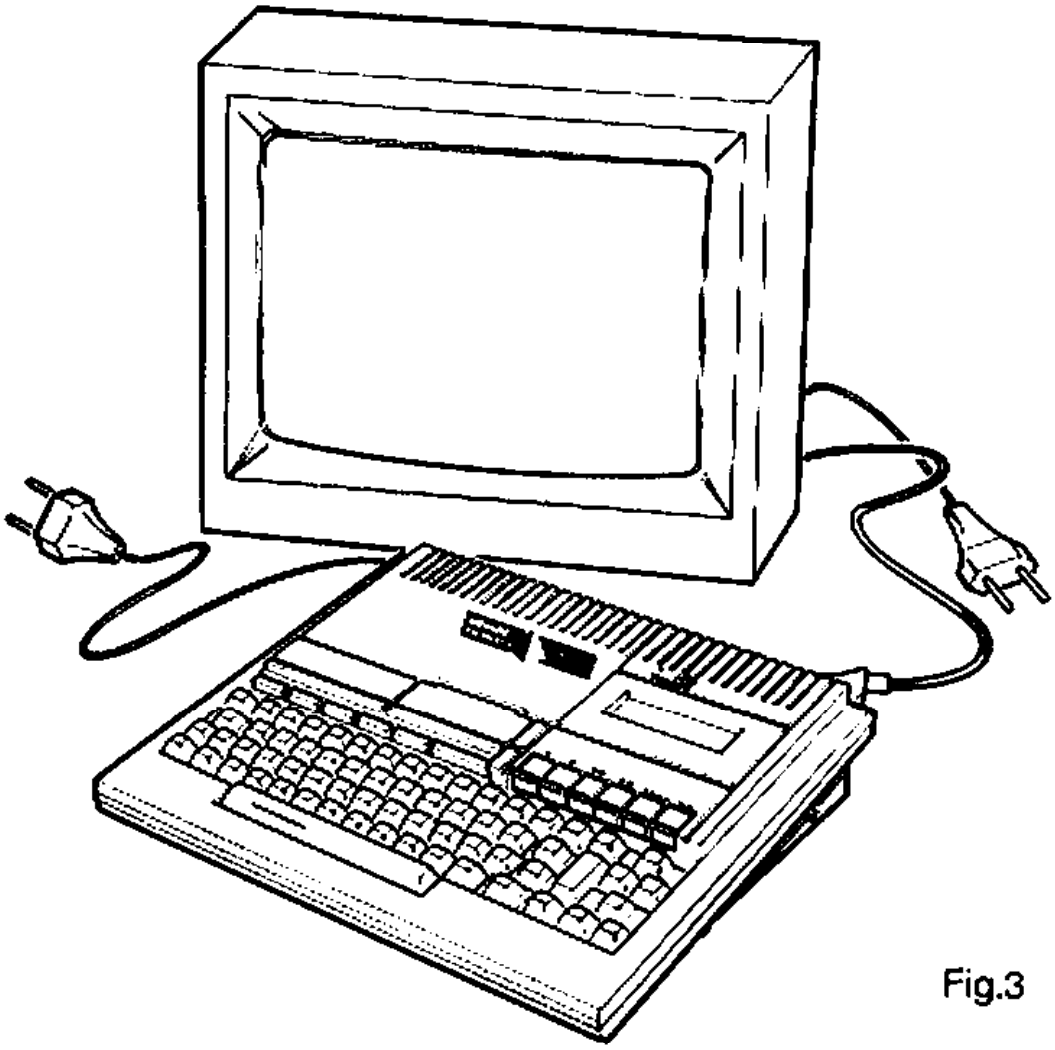


Fig.3

Se si possiede un televisore senza presa SCART, si dovrà utilizzare l'uscita antenna UHF PAL (pag 2, fig. 1, numero 8) e collegare il computer, tramite il cavo fornito nella confezione, alla presa antenna del televisore.

La sintonia ottimale si ottiene sul canale 36.

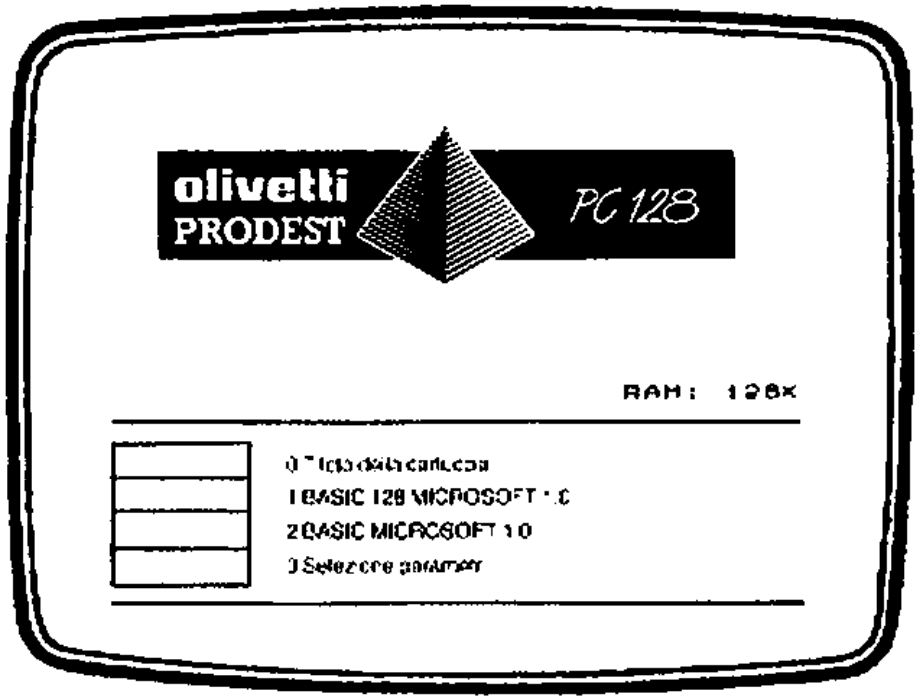
Se si possiedono delle unità periferiche (penna ottica, drive o stampante), collegarle ai connettori corrispondenti.

Se si possiede un solo joystick o un mouse, collegarlo alla presa 6 della figura numero 1. In seguito useremo la parola "puntare" per descrivere l'azione di indicare con la penna ottica e di appoggiare la sua punta sullo schermo, o di mirare con il mouse e premere il pulsante di sinistra.

Avviamento

Accendere nell'ordine il televisore su un qualsiasi canale, le periferiche, poi il PC 128 premendo il tasto acceso–spento; una spia verde si illuminerà a sinistra del registratore.

Dopo qualche secondo appare un'immagine: è la pagina di testa con il suo menu di selezione.



La pagina di testa

La pagina di testa indica:

- la quantità di memoria a disposizione: 128 K,
- il titolo del programma della cartuccia, se inserita, accessibile dal tasto 0 (zero) della tastiera, o puntando tramite una penna ottica o un mouse il rettangolo giallo corrispondente,
- il BASIC 128 MICROSOFT®, accessibile dal tasto 1 della tastiera, o puntando il rettangolo giallo corrispondente,
- il BASIC 1.0 MICROSOFT®, accessibile dal tasto 2 della tastiera, o per puntamento,
- lo schermo di regolazione e opzioni, accessibile dal tasto 3 della tastiera, o per puntamento.

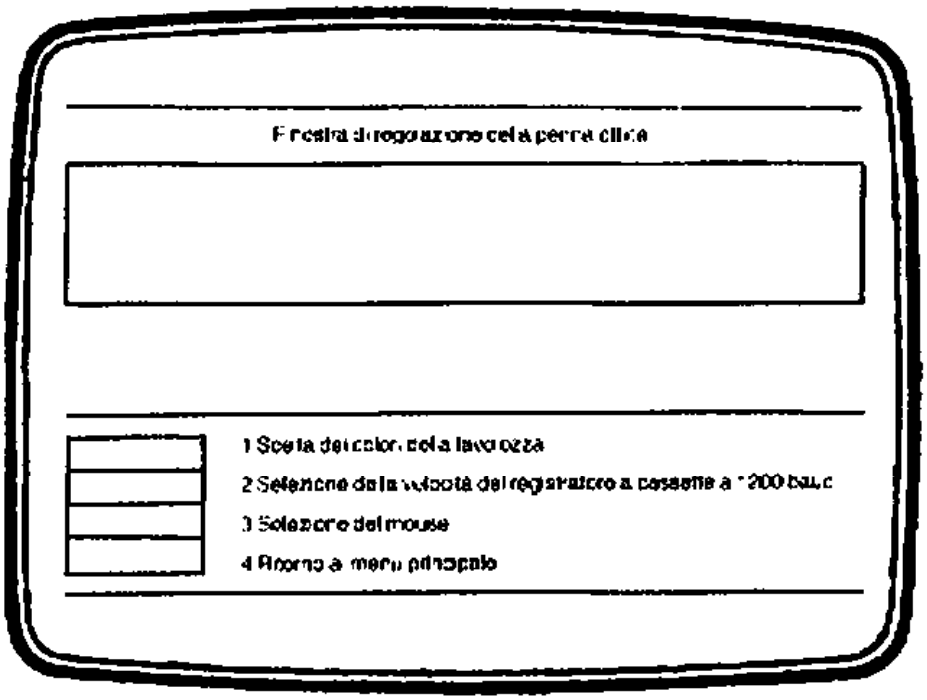
Con un drive, la scelta del BASIC 128 permette di caricare in memoria automaticamente un programma su dischetto il cui nome è AUTO.BAT.

La scelta del BASIC 1.0 implica, per poter utilizzare il drive, la presenza all'interno del lettore del dischetto DOS, disponibile presso il rivenditore. Allo stesso modo il

BASIC 1.0 potrà caricare automaticamente un programma presente sul dischetto DOS se il suo nome è: AUTO.BAT.

Regolazioni e opzioni

Selezionare sulla tastiera il tasto 3: regolazioni e opzioni.



Appare un nuovo schermo che comprende:

– nella sua parte superiore, una finestra di regolazione della penna ottica.

Per procedere alla regolazione di una penna ottica, sarà sufficiente appoggiarne la punta perpendicolarmente allo schermo. Apparirà una barra verticale.

Far coincidere con i tasti ← e → questa barra con la punta della penna; la regolazione è terminata.

– nella sua parte inferiore, un menu:

1 – Scelta dei colori della tavolozza
2 – Selezione della velocità del registratore a cassette a 1200 baud
3 – Selezione del mouse
4 – Ritorno al menu principale

Avendo un mouse il menu appare come segue:

1 – Scelta dei colori della tavolozza
2 – Selezione della velocità del registratore a cassette a 1200 baud
3 – Selezione della penna ottica
4 – Ritorno al menu principale

Il PC 128 dà la possibilità di collegare allo stesso tempo un mouse e una penna ottica, ma non è possibile utilizzarli contemporaneamente.
IL PC 128 ha dunque bisogno di sapere con quale periferica si desidera lavorare.
Una volta acceso, il PC 128 individua la presenza di un mouse e nel menu dà la possibilità di ulilizzare la penna ottica.

Scegliendo la penna ottica, il menu si modifica per dare la possibilità di scegliere nuovamente il mouse:

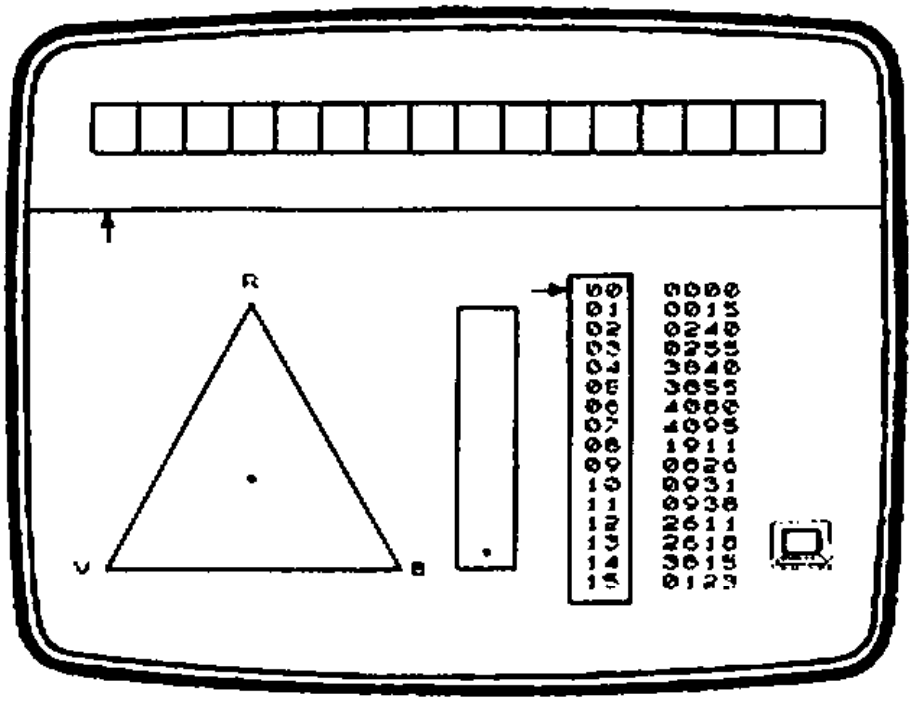
1 – Scelta dei colori della tavolozza
2 – Selezione della velocità del registratore a cassette a 1200 baud
3 – Selezione del mouse
4 – Ritorno al menu principale

L'ultima opzione del menu permette di ritornere alla pagina iniziale del PC 128.

La tavolozza dei colori

Selezionare la tavolozza dei colori tramite il tasto 1.

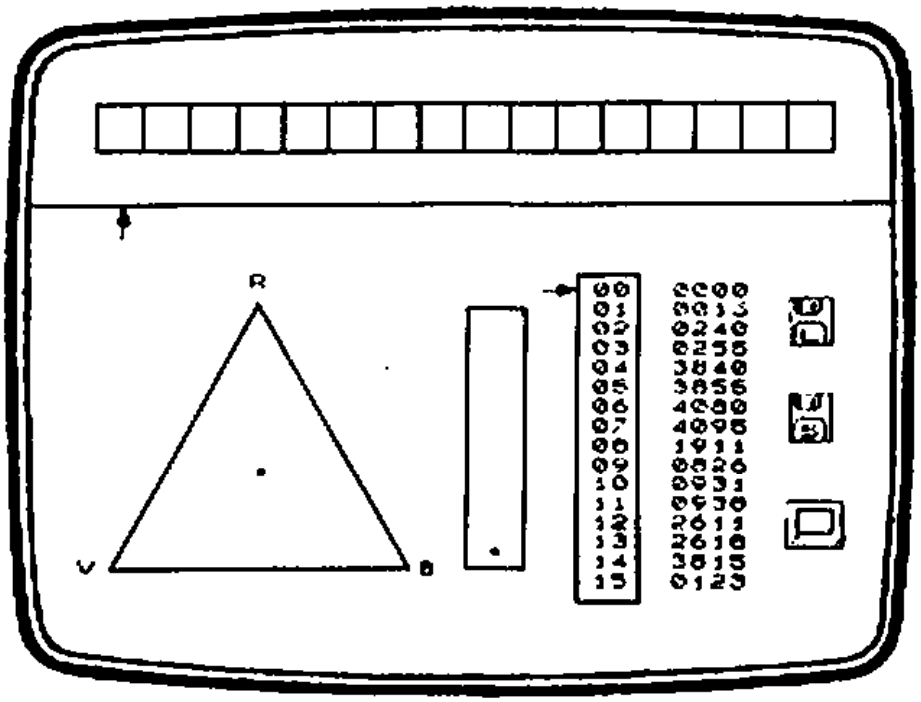
Appare un nuovo schermo:



Nella sua parte superiore lo schermo è costituito dalla tavolozza iniziale del PC 128, composta da 16 colori, di cui il primo, il nero, è indicato da una freccia e serve da colore di cornice alla tavolozza.

Sotto si trovano un triangolo di selezione del colore, un rettangolo verticale di selezione d'intensità di tate colore, più a destra i numeri dei colori con una freccia che indica il colore selezionato, e i numeri che rappresentano questo colore tra le 4096 sfumature possibili, di seguito un'icona rappresentante un microcomputer per ritornare alla pagina di testa.

Collegando un drive o un QDD, sono previste due icone supplementari che permettono di salvare e rileggere la propria tavolozza personalizzata.



Per selezionare un colore, è sufficiente spostare la freccia che indica i colori premendo il numero necessario di volte il tasto ENT. Si noterà che la freccia salta dal colore 15 al colore 0.

Contemporaneamente, si potrà osservare che il colore di cornice della tavolozza assume la tinta selezionata e che appare un punto in diverse posizioni del triangolo di selezione del colore e sul rettangolo d'intensità.

Per modificare un colore:
- premere il tasto [ENT] .
- poi, spostare il punto corrispondente nel triangolo di selezione per mezzo dei quattro tasti freccia della tastiera.

I tre vertici del triangolo corrispondono ai tre colori fondamentali Rosso, Verde, Blu che, per addizione, danno tutte le sfumature di colore, rappresentate ognuna da un punto del triangolo.

Per modificare l'intensità, utilizzare i tasti contrassegnati [+] e [−] .
- il tasto [+] intensifica la tinta,
- il tasto [−] scurisce la tinta fino al nero.

Con un mouse o una penna ottica, la regolazione è ancora più semplice:

- selezionare il colore da modificare puntando il mouse o ta penna ottica sul suo numero,
- spostare il punto rappresentante il colore nel triangolo puntando il mouse o la penna ottica,
- scegliere l'intensità di colore spostando nello stesso modo il punto rappresentativo nel rettangolo di regolazione.

Con un drive, sarà possibile salvare la tavolozza personale registrandola.

Per far ciò, sarà sufficiente premere sulla tastiera il tasto [S] per salvarla, e il tasto [L] per leggere la tavolozza registrata sul dischetto, o con una penna ottica o un mouse, puntare le icone corrispondenti.

Il file che contiene la tavolozza personalizzata si chiama TAVOLOZ.CFG. Per salvare la propria tavolozza su cassetta, vedere l'appendice dopo il capitolo 5.
Per ritornare alla pagina di testa, premere il tasto STOP, o puntare l'icona [⏏] .

Il registratore

Il registratore incorporato al PC 128 funziona con una velocità di trasferimento delle informazioni di 2400 Baud (un Baud=un Bit al secondo).

La seconda opzione del menu: "Selezione della velocità del registratore a cassette a 1200 baud", permette di registrare delle cassette a una velocità di trasferimento di 1200 Baud.

Durante la lettura, il registratore riconosce automaticamente la vetocità alla quale è stata registrata la cassetta, quindi non è necessario modificare questo parametro.

La tastiera del registratore è composta da sei tasti:

da sinistra a destra:

- tasto registrazione [●]
- tasto lettura [►]
- tasto arretramento rapido [◄◄]
- tasto avanzamento rapido [►►]
- tasto arresto nastro e fuoriuscita cassetta [STOP/EJECT]
- tasto arresto momentaneo [PAUSE]

Per inserire una cassetta, è necessario premere il tasto "STOP–EJECT" per aprire lo sportello porta–cassetta. Quindi, basta introdurre la cassetta con il nastro magnetico rivolto verso l'operatore, e richiudere lo sportello. Si consiglia di utilizzare cassette nuove o cancellate.

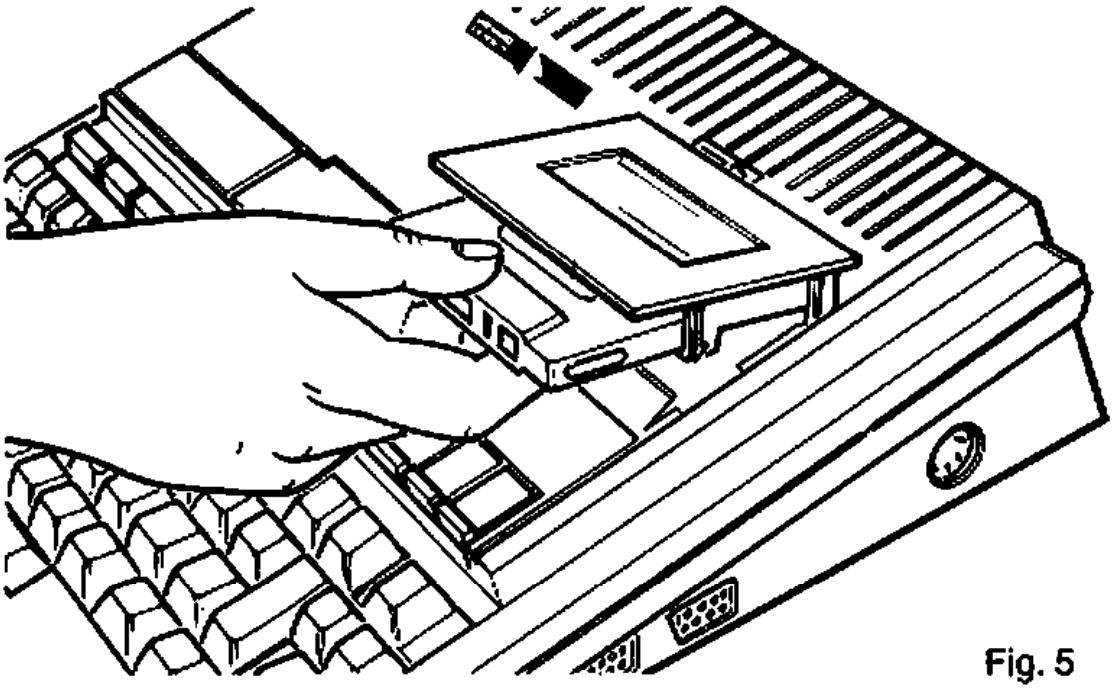


Fig. 5

Registrazione

La cassetta è utilizzabile su due lati.

Per registrare l'inizio di una cassetta, per prima cosa è necessario azzerare il contagiri premendo il pulsante che si trova alla sua destra, e far scorrere il nastro per un istante premendo il tasto avanzamento rapido poi il tasto STOP, in modo da evitare di cominciare la registrazione sulla parte iniziale non magnetizzata del nastro.

Premere quindi simultaneamente il tasto registrazione e il tasto lettura. La spia verde di accensione diventa rossa. La partenza del nastro è comandata dal PC 128.

Alla fine della registrazione, premere il tasto "STOP–EJECT" per rilasciare i tasti di lettura e registrazione, e annotare il numero indicato dal contagiri per poter registrare successivamente a partire da questa posizione un altro programma.

E' preferibile lasciare uno spazio sul nastro tra due registrazioni, in modo da facilitarne il reperimento.

Si possono proteggere delle registrazioni importanti da una cancellazione accidentale, eliminando la linguetta corrispondente al lato da proteggere.

Per poter registrare nuovamente su un lato protetto, occorre otturare l'apertura di protezione con del nastro adesivo.

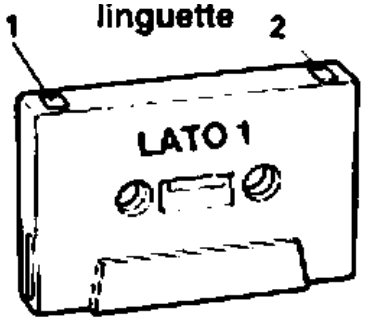


Fig.6 **Cassette con nastro a ossido di ferro**

Lettura

Utilizzando il riferimento del contagiri, annotato al momento della registrazione, si accelera la ricerca di una registrazione.

Per eseguire questa operazione, far scorrere il nastro tramite avanzamento e arretramento rapido per posizionarlo un po' prima del numero di riferimento, premere il tasto "STOP–EJECT" per arrestare lo scorrimento, quindi premere il tasto di lettura. La partenza è comandata dal PC 128.

Manutenzione

Per mantenere tutte le qualità del registratore, è necessario di tanto in tanto procedere alla pulizia delle testine, del capstan e del rullo premi–nastro con un bastoncino ovattato imbevuto di alcool.

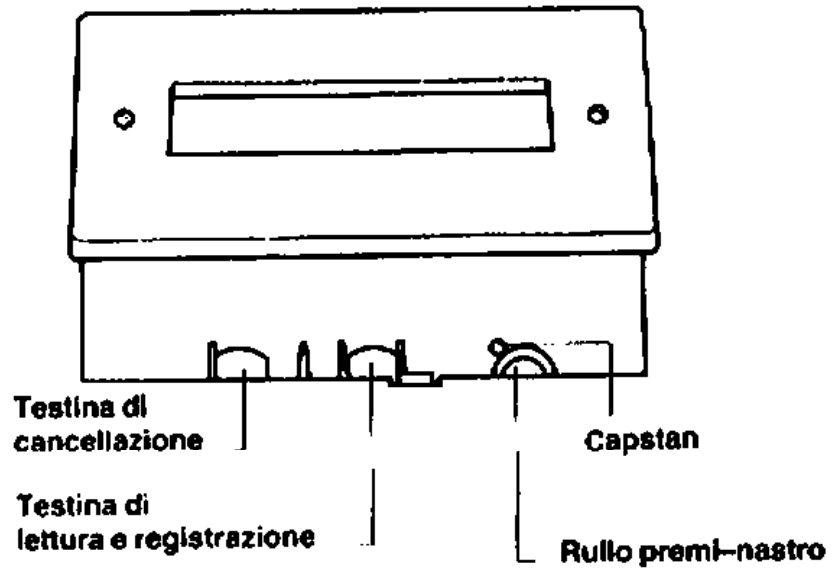


Fig.7

Per eseguire questa operazione, togliere la cassetta e premere il tasto di lettura.

Alla fine dell'operazione di pulizia, premere il tasto "STOP–EJECT".

Uso della cartuccia

Per utilizzare una cartuccia di linguaggio o di programma nel PC 128, inserirla nell'apposito alloggio situato al di sopra della tastiera, con l'etichetta rivolta verso l'operatore, e seguire le istruzioni fornite con la cartuccia.

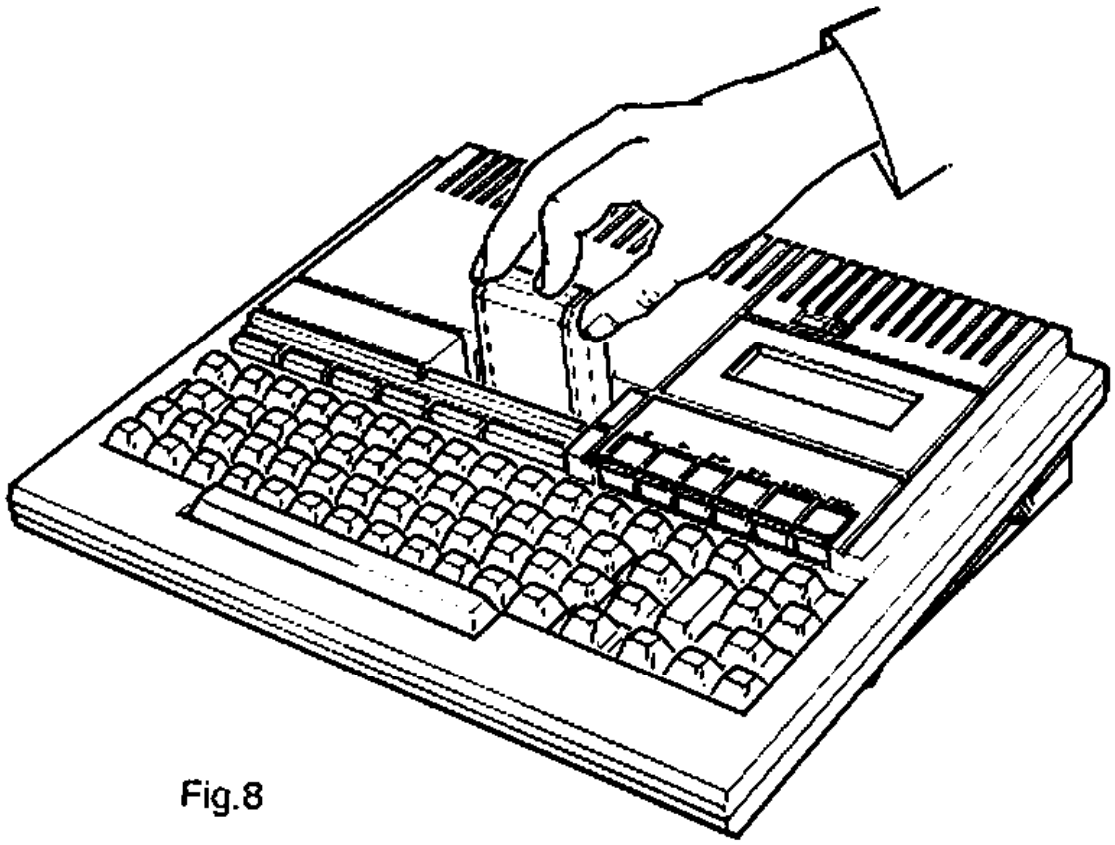


Fig.8

Le periferiche del PC 128

E' possibile collegare al PC 128 le seguenti periferiche:

- una penna ottica
- un mouse
- uno o due joystick
- una stampante: 40 colonne, modello PR 90 055
  80 colonne, modello PR 90 582 o PR 90 600

Al connettore per l'espansione, è possibile collegare una delle seguenti periferiche:

- un QDD con controltore
- un drive per dischetti da 3,5 pollici con controllore
- un drive per dischetti da 5,25 pollici, da 320 Kbyte, doppia densità o modello 80 Kbyte singola densità
- un MODEM modello MD 90 333
- un'interfaccia IEEE
- un'interfaccia seriale RS 232 (solo il modello RS 57 932)
- un'interfaccia video

Al contrario, non è possibile collegare le seguenti periferiche:

- la cartuccia di espansione memoria RAM di 64 Kbyte per utilizzo in rete (già integrata)
- l'espansione musica e giochi (già integrata)
- l'espansione uscite parallele (già integrata)
- l'espansione digitalizzazione
- l'espansione telematica

NOTA 1: Quando si seleziona il linguaggio BASIC del PC 128, avendo precedentemente collegato un drive per dischetti da 5,25 pollici, doppia densità, il controllore individua automaticamente la densità del dischetto inserito, e continua a lavorare in questa densità. Se, per errore, fosse stato lasciato inserito un dischetto registrato nella densità che non si desidera utilizzare, è necessario inserire un dischetto della giusta densità nel drive e premere il tasto RESET.

NOTA 2: Quando si utilizza il PC 128 contemporaneamente ad un mouse attivo ed una stampante, la pressione di uno dei tasti del mouse blocca la stampa di un documento. La stampa riprende rilasciando il tasto.

Caretteristiche principali

I – Specifiche elettriche

Corrente d'alimentazione: 220 V, 50 Hz
Consumo: 24 W
Fusibile: 160 mA

II – Descrizione

Esterno: materiale plastica stampata
Dimensioni: L=362, H=87, P=315 mm.
Peso: 3 Kg

Tastiera: QWERTY, 69 tasti meccanici, con spia per il carattere maiuscolo.

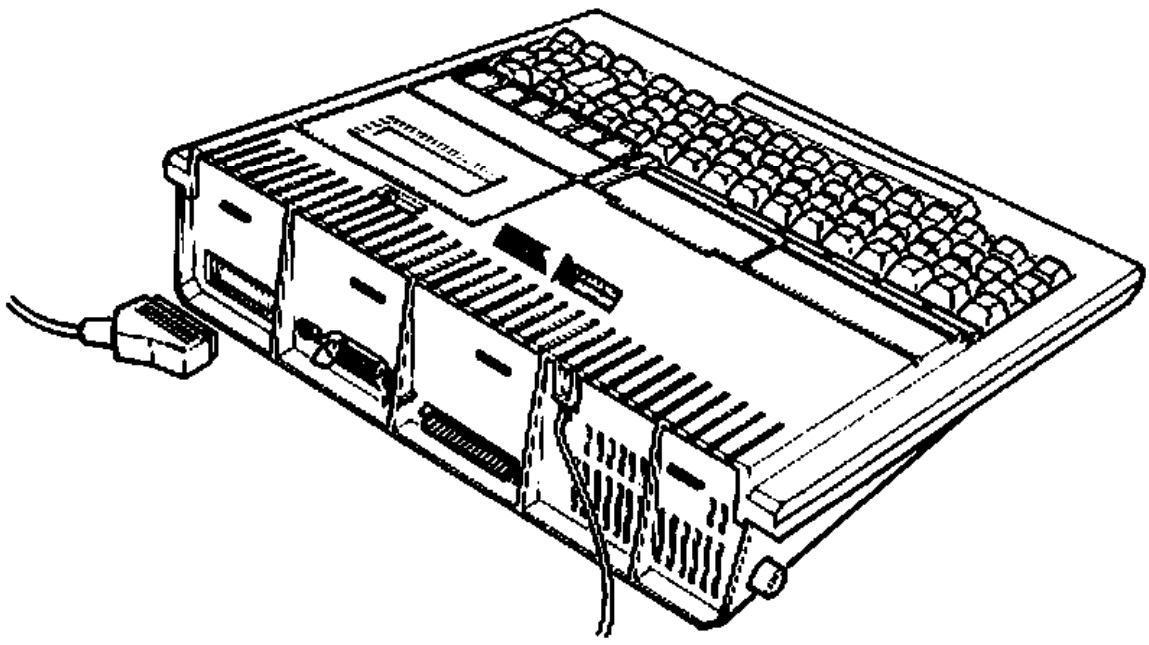
III – Specifiche tecniche

Microprocessore: 6809 E – 1 MHz
Memoria: ROM: 64 Kbyte ampliabile con integrati: BASIC 128, BASIC 1.0 MICROSOFT
RAM: 128 Kbyte non ampliabile

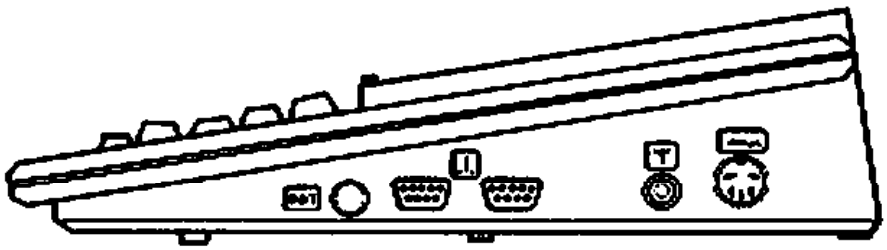
Schermo: uscita RGB+audio attraverso presa SCART
320 x 200 – 16 colori (40 colonne)
640 x 200 – 2 colori (80 cotonne)
320 x 200 – 4 colori punto per punto (40 colonne)
160 x 200 – 16 colori punto per punto
320 x 200 – 3 colori con un livello di trasparenza
320 x 200 – 2 colori con visualizzazione alternativa di 2 pagine
160 x 200 – 5 colori con 3 tivelli di trasparenza

Tutti i colori sono selezionabili in una tavolozza che propone 4096 sfumature.
Penna ottica: risoluzione 320 x 200 punti
Registratore integrato 1200 o 2400 baud a riconoscimento automatico durante lettura
Sintetizzatore musicale integrato a 4 voci su 7 ottave
Uscita audio: presa CINCH

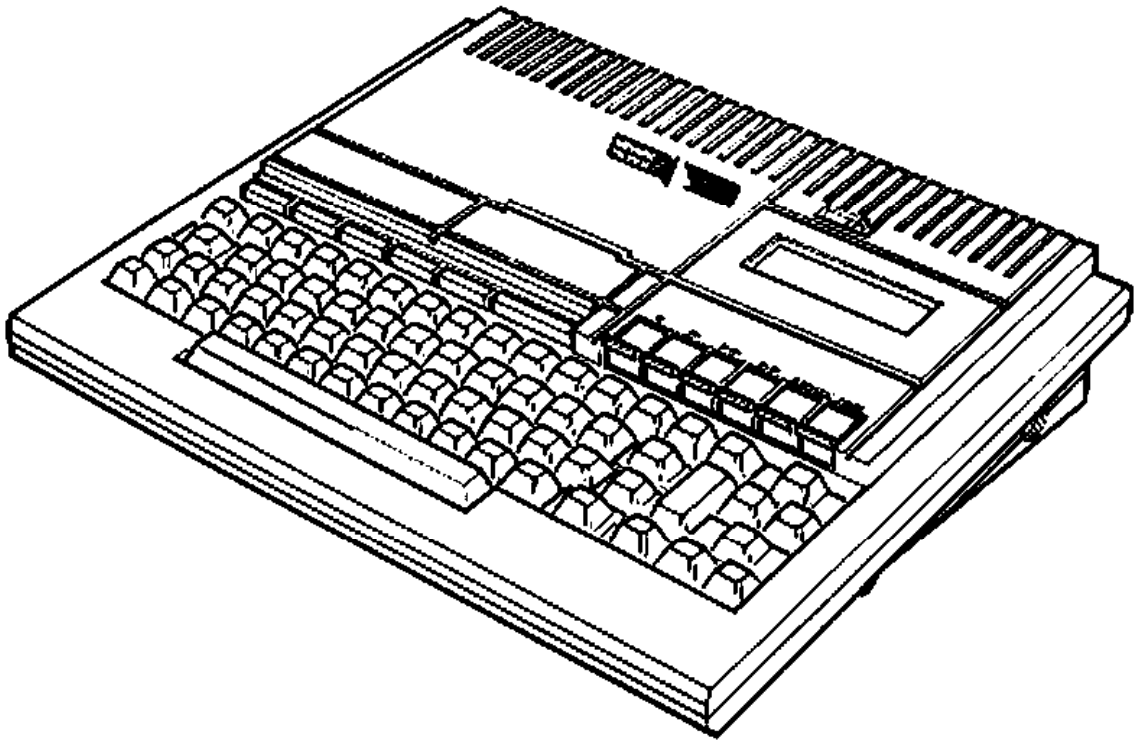
Connettori:
sul retro: un connettore polivalente per espansioni
una presa CENTRONICS 14 poli per stampante
una presa CINCH uscita suono
una uscita SCART (allacciamento TV)



sul lato destro: due prese Sub–D 9 poli per mouse, joystick
una presa CtNCH uscita UHF–PAL
una presa DtN per penna ottica



sulla parte superiore:
un alloggiamento per cartucce di programma o di linguaggio.



Guida al BASIC

Parte I: Introduzione

Dopo quest'iniziale approccio è venuto il momento di utilizzare tutte le possibilità operative del BASIC PC 128. Il computer PC 128 possiede infatti due linguaggi BASIC:
– con il tasto 1 si accede al BASIC 128 MICROSOFT®
– con il tasto 2 si accede al BASIC 1.0 MICROSOFT®

La presente guida al BASIC permetterà di scoprire le potenzialità e le grandi risorse del BASIC 128. Essa si articola in due parti:
– un'introduzione alla programmazione composta da 50 capitoli–scheda di facile comprensione,
– una guida di riferimento che definisce tutte le particolarità inerenti a ciascuno dei due linguaggi disponibili.

Apprendere a programmare può richiedere un po' di tempo, ma ottenere immediatamente dei risultati incoraggianti sullo schermo è questione di pochi minuti. Dal capitolo 1 al capitolo 17 sarà l'utente ad introdurre tutte le istruzioni che permetteranno di acquisire le basi per il funzionamento del BASIC. I risultati grafici e musicali saranno promettenti.

I capitoli 21 e 22 forniscono i principi generali della programmazione per guidare il programmatore ancora inesperto. Dopodiché, la strada è libera! Ciascun capitolo succeasivo arricchisce il vocabolario acquisito e risolve piccoli problemi: come accedere ai dati da tastiera, come utilizzare le funzioni grafiche, ecc.

Infine i capitoli 49 e 50 riservano ai curiosi molte soddisfazioni. In effetti, in poche righe di programma, essi permettono di creare degli effetti video assolutamente sorprendenti (49) e forniscono un metodo per produrre un'infinita quantità di Rock and Roll (50).

Par il momento, selezionare il BASIC 128 per mezzo del tasto 1 e seguire le istruzioni della guida.

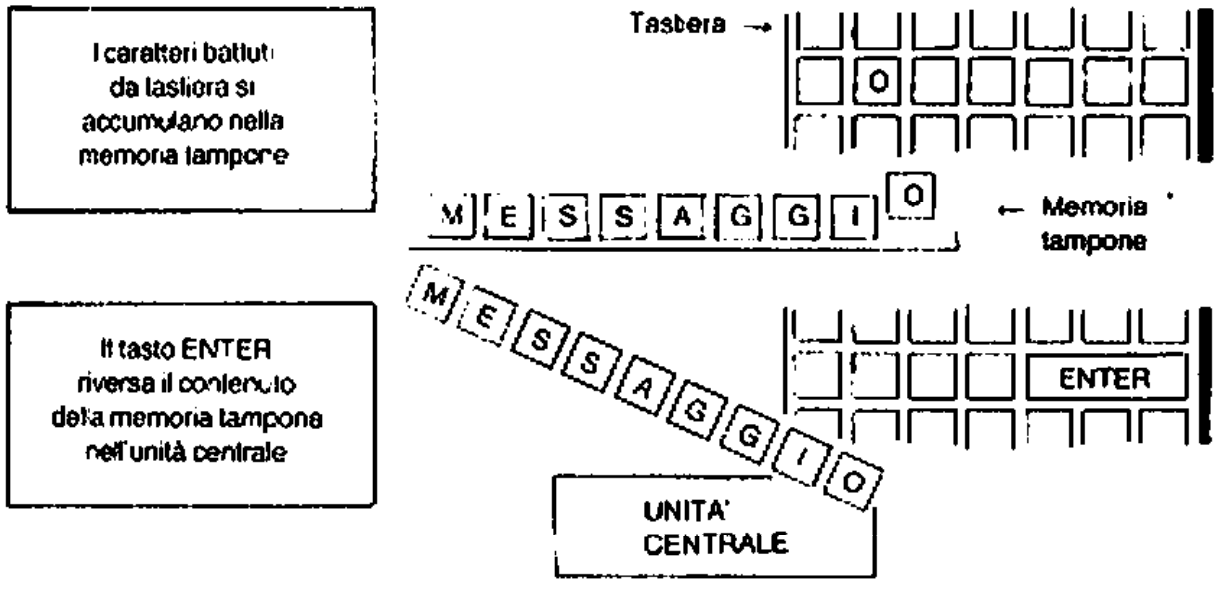
Capitolo 1

Il tasto ENTER ed altri tasti

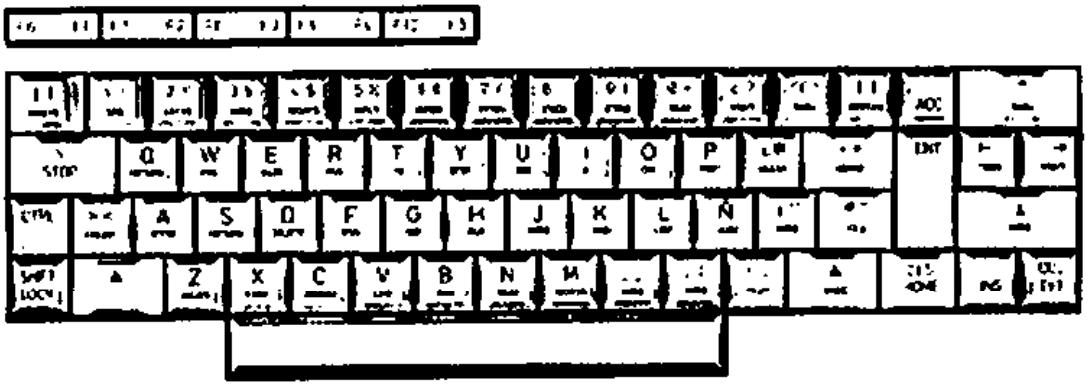
Prima di tutto è importante presentare la "scena". Per l'utente del PC 128 la "scena" è costituita dalla tastiera e dallo schermo. L'utente "comanda" il proprio PC 128 dalla tastiera. Le risposte della macchina, quando fornite, sono visibili sullo schermo. Il tasto chiave che permette di realizzare questo dialogo è ENTER. Questo tasto si nota al primo colpo d'occhio ed è facilmente accessibile, questo perchè è un tasto che deve essere utilizzato in continuazione.

Il tasto ENTER

Il tasto ENTER è chiamato anche tasto di ritorno carrello perchè la sua funzione è simile a quella di un ritorno carrello su di una macchina per scrivere tradizionale. Presenta tuttavia delle differenze sostanziali che gli conferiscono di fatto un ruolo preponderante. Quando si batte da tastiera un messaggio destinato at computer, i caratteri sono caricati in una memoria intermedia, detta memoria tampone, che registra passivamente il messaggio: quando si scrive una lettera, le parole si accumulano sul foglio senza che il destinatario ne conosca ancora il contenuto. Il gesto importante consiste nell'imbucare la lettera nell'apposita buca. Con un microelaboratore la spedizione del messaggio si attua invece semplicemente premendo il tasto ENTER: l'insieme dei caratteri battuti si riversa nella parte attiva della macchina che analizzerà ed eventualmente eseguirà le istruzioni in esso contenute.



La capacità della memoria tampone è di 255 caratteri. Ciò significa che il dialogo con la macchina è limitato a 255 caratteri per invio. Ciò permette quanto meno di comprendersi.



Un'occhiata alla tastiera

Questa tastiera è det tipo QWERTY (QWERTY sono le prime lettere della prima fila di tasti alfabetici).

Quando si programma in BASIC, occorre per prima cosa premere il tasto SHIFT–LOCK (vedere "Guida all'installazione"). La spia di questo tasto si accende. Prendiamo l'esempio del tasto 1 che permette di produrre tre scritture differenti.

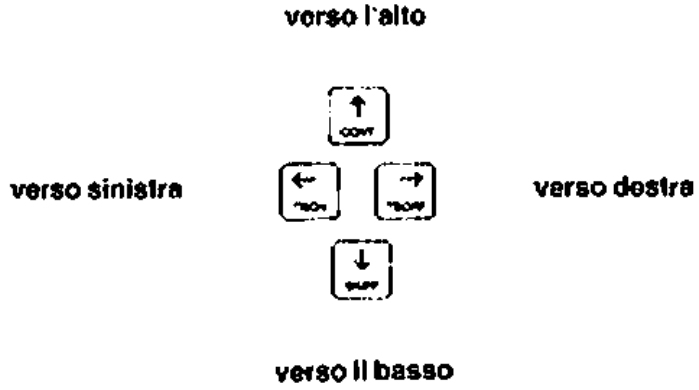
premere	1 !	produce la scrittura di	1
premere	1 !	produce la scrittura di	!
premere	▲ BASIC 1 !	produce la scrittura di	GRS

I tasti diversi da quelli alfanumerici non sono interessati dal tasto SHIFT–LOCK. Per questi, si otterrà il simbolo riportato in un determinato colore premendo dapprima il tasto ▲ dello stesso colore (vedere "Guida all'installazione"). Le istruzioni BASIC si ottengono come precedentemente descritto.

L'utilizzazione di questi tasti "preprogrammati" permette di evitare errori di battitura e di accelerare la scrittura. L'utente del PC 128 ha a disposizione le istruzioni BASIC più utilizzate. E' importante conoscere bene la propria tastiera per non perdere troppo tempo a cercare le parole già presenti o quelle non previste.

Tasti di movimento cursore

Il cursore e un carattere speciale sempre presente sut video e in genere lampeggiante. tndica all'utente il punto preciso in cui verrà visualizzato il prossimo carattere introdotto. I quattro tasti con le frecce permettono di spostare il cursore e la loro disposizione ricorda l'azione che svolgono:



L'uso di questi tasti deve diventare familiare. Si noti che in un altro punto della tastiera si trova un tasto che agisce sul cursore portandolo in alto a sinistra sul video, contrassegnato da CLS HOME.

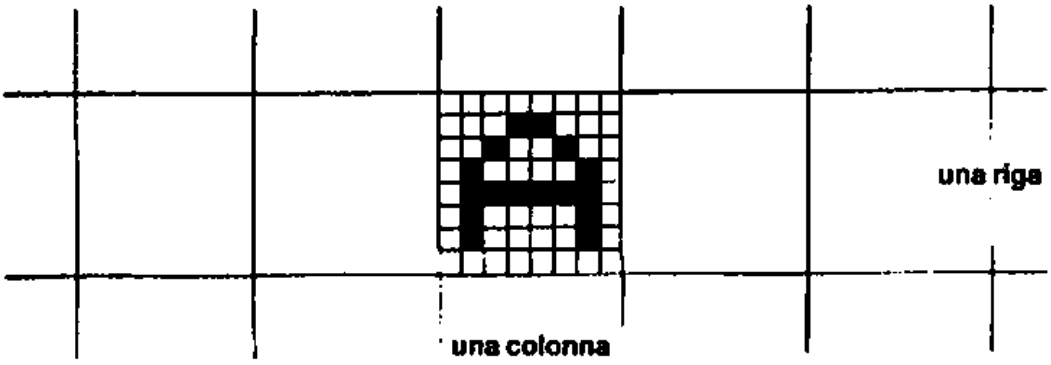
Attenzione: è molto importante distinguere lo zero "0" dalla tettera O.

Consultare la parte relativa all'instaltazione per l'accesso sia alle maiuscole/minuscole sia agli accenti.

Capitolo 2

Lo schermo e la tastiera

Consideriamo lo schermo come una griglia con colonne verticali e righe orizzontali le cui intersezioni definiscono delle celle in cui si inseriscono i caratteri. A sua volta ogni cella può essere vista come un'ulteriore griglia più piccola in cui alcune celle (pixel) sono accese o spente per permettere la visualizzazione di un carattere.



Premendo uno dei tasti di spostamento del cursore, è possibile passare da una cella a una delle quattro celle adiacenti. La rappresentazione a righe e a colonne non è in effetti una caratteristica dello schermo ma del microcomputer stesso.
Il PC 128 visualizza i caratteri su 40 colonne e 25 righe.

Cancellazione dello schermo

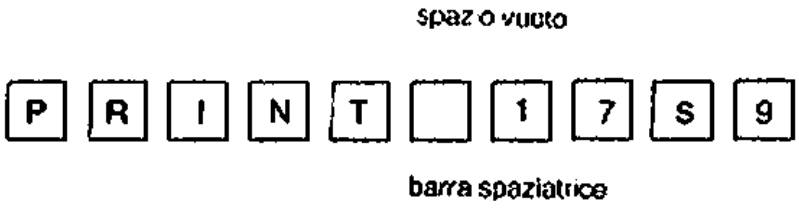
Il tasto CLS HOME cancella completamente ciò che è visualizzato sullo schermo in quel momento. In questo modo il cursore viene a trovarsi in alto a sinistra e il PC 128 è pronto per ricevere dei nuovi comandi. Lo schermo è vuoto: si può cominciare.

Visualizzazione

Dato che lo schermo è lo strumento che permette al computer di comunicare i risultati delle sue operazioni, è importante prendere in esame subito un comando di visualizzazione. E' meglio iniziare con i numeri perchè sono i caratteri che il computer gestisce più facilmente.
Il comando di visualizzazione che esaminiamo ora è PRINT.

```
PRINT 1789 [ENT]
1789
OK
```

Questo primo esempio è, almeno apparentemente, molto semplice. In realtà questo comando presuppone la memorizzazione da parte della memoria tampone dei dieci caratteri inseriti prima di premere il tasto ENTER.



Premendo il tasto ENTER, il contenuto della memoria tampone viene passato all'interprete BASIC che inizia immediatamente il suo lavoro di analisi.
Dopo il quinto carattere, l'interprete ha identificato una parola chiave del linguaggio BASIC e ha lanciato la procedura legata a questa parola, e cioè ignorare lo spazio vuoto e successivamente visualizzare sullo schermo tutto quello che precede il tasto ENTER.

Vediamo ora un uso più complesso del comando PRINT.

```
PRINT 17+35 [ENT]
52
OK
```

Notiamo che il PC 128 si comporta come se fosse una calcolatrice, anche se è necessario ordinare la visualizzazione del risultato mentre non è indispensabile farlo con una calcolatrice. Vedremo poi il perchè, anche se è possibile rispondere immediatamente a questo quesito. Se si vuole fare un calcolo abbastanza lungo, è probabile che interessi solo il risultato finale. Può non interessare la visualizzazione dei risultati intermedi. Quindi il computer può eseguire dei calcoli, ma per vedere i risultati è necessario chiedere di visualizzarli. Questo lo differenzia da una calcolatrice e costituisce al tempo stesso un grande vantaggio in quanto l'utente può decidere che cosa visualizzare.
Quindi, tutto sommato, anche per quanto riguarda i catcoli, il computer è meglio di una catcolatrice.
Ecco per esempio l'esecuzione di un calcolo complicato dove vengono usati:
– il punto (.) per separare nei numeri decimali la parte intera dalla parte decimale in base all'uso anglosassone.
– il segno più (+) e il segno meno (–)
– il segno di moltiplicazione (*) (da non confondere con la lettera X)
– il segno di divisione (/)
– le parentesi "(" e ")"; è importante assicurarsi che siano aperte e chiuse nell'ordine esatto.
E' anche importante sapere che le regole di priorità delle quattro operazioni seguono quelle usate abitualmente in matematica: la moltiplicazione e la divisione hanno la priorità sull'addizione e sulla sottrazione. Le parentesi impongono delle priorità assolute.

```
PRINT (5*3.4-2*5.6)/7+2 [ENT]
2.82857
OK
```

I principali vantaggi rispetto a una calcolatrice sono i seguenti:
- La permanenza del calcolo e del suo risultato
- La possibilità grazie all'editor di riga di modificare e correggere.

L'editor

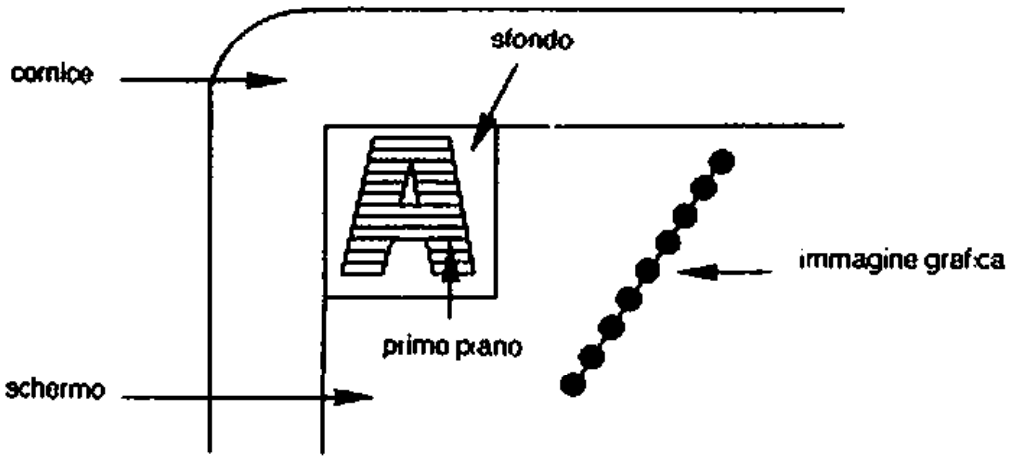
L'editor è un programma interno destinato a semplificare la visualizzazione e la correzione. E' molto importante avere una buona padronanza dell'editor. Quando è stato individuato un errore, i tasti di spostamento del cursore permettono di posizionarsi sulla zona in cui effettuare la correzione; questa correzione può essere di tre tipi:
- La sostituzione di un carattere: è sufficiente battere it nuovo carattere al posto di quello errato. Il risultato sul video è immediato.
- L'eliminazione: per eliminare un carattere non bisogna assolutamente premere la barra spaziatrice in quanto è necessario che allo stesso tempo il carattere scompaia dallo schermo e che non venga lasciato alcuno spazio. E' il tasto EFF che esegue le due operazioni contemporaneamente.
- L'inserimento: per aggiungere uno o più caratteri bisogna premere il tasto INS. Il carattere davanti al quale verrà fatto l'inserimento viene visualizzato in inversione video. I nuovi caratteri che vengono aggiunti si inseriscono spingendo il resto della riga (modalità inserimento). Per uscire dalla modalità inserimento è sufficiente premere di nuovo il tasto INS: il carattere davanti al quale vi è stato l'inserimento torna del suo colore standard.

Le quattro frecce di spostamento permettono di spostare il cursore ovunque sulto schermo e di scrivere qualsiasi cosa all'interno dello schermo stesso.
L'editor del PC 128 viene definito un editor a "schermo pieno".

Capitolo 3

I colori dello schermo

Abbiamo notato che in fase di inserimento il PC 128 usa l'inversione video. Questa è una delle possibilità offerte dal PC 128 per generare i colori. In generale, l'utente può far intervenire dei codici colore a cinque livelli:



Il PC 128 è dotato di sedici colori selezionabili tra 4.096 sfumature. Inizialmente la visualizzazione avviene in blu scuro su azzurro. Per cambiare le caratteristiche di visualizzazione, il comando di modifica dei colori dello schermo deve essere dato al computer nel modo seguente:

SCREEN 7,0,4

Dopo aver inserito questa istruzione, premere il tasto ENTER per farla "eseguire" dal computer. Ormai dovrebbe essere chiaro l'uso del tasto ENTER e quindi non comparirà più alla fine del comando.
Se il messaggio è stato trasmesso e ricevuto correttamente, l'istruzione viene eseguita e i caratteri diventano bianchi su sfondo nero, con il bordo esterno blu. Quindi il computer dà l'OK e il cursore torna all'inizio della riga seguente e attende una nuova istruzione. L'istruzione SCREEN (= schermo) è definita da tre numeri:
- il primo per il colore dei caratteri
- il secondo per il colore dello sfondo
- il terzo per il colore del contorno del rettangolo utile.

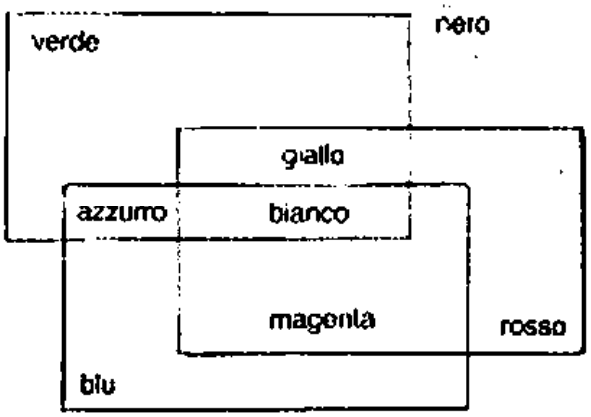
Nozioni di tecnica dei colori

Lo schermo di un televisore a colori è costellato di punti, ognuno dei quali è composto da tre colori: rosso, verde e blu.

Sullo schermo, contrariamente a quello che avviene sulla carta, l'occhio somma i colori. Partendo dai tre colori fondamentali, si ottengono cinque altri colori di sintesi, quindi un totale di otto colori (2^3) di base.
I colori vengono codificati nel computer per mezzo dei "bit". Il bit è la più piccola unità di memorizzazione delle informazioni usata dal microcomputer. Il suo valore è 1 o 0 e non può assumere altri valori che questi.

Esempio di codifica degli otto colori di base

Ognuno di questi otto colori può essere codificato su tre bit: ogni bit è a 0 o a 1 a seconda che sia presente o meno uno dei tre colori fondamentali. Lo schema e la tabella che seguono riassumono la sintesi e la codifica di questi otto colori.



	Blu	Verde	Rosso
nero	0	0	0
rosso	0	0	1
verde	0	1	0
giallo	0	1	1
blu	1	0	0
magenta	1	0	1
azzurro	1	1	0
bianco	1	1	1

Per aumentare il numero delle sfumature, si gioca sulla luminosità di ognuno dei tre colori fondamentali. Viene così codificato un quarto bit per ottenere o un colore scuro o un colore chiaro.

Attenzione: se si tenta di schiarire il bianco si ottiene l'arancione.

Tabella dei colori

0 nero
1 rosso
2 verde
3 giallo
4 blu
5 magenta
6 azzurro
7 bianco
8 grigio
9 rosso chiaro
10 verde chiaro
11 giallo chiaro
12 blu chiaro
13 magenta chiaro
14 azzurro chiaro
15 arancione

Questa numerazione è standard. L'opzione 1 del menu principale (scegliere la tavolozza dei colori) permette di cambiare i colori e i loro codici. In tutto il manuale verrà usata la tavolozza standard.

Sintassi dell'istruzione SCREEN

Un'istruzione può spesso essere scritta con formati diversi più o meno complessi. Quindi l'istruzione SCREEN non deve essere seguita necessariamente da tre numeri.

Esempi: per far diventare rosso solamente il colore dei caratteri, è sufficiente battere:

SCREEN 1

e per cambiare il cotore dei caratteri e dello sfondo

SCREEN 1,5

E' possibile usare anche un quarto perametro (o "argomento"):

SCREEN,,,1

Questo comando fa invertire i cotori dei caratteri e dello sfondo (inversione video). Se il comando viene eseguito una seconda volta:

SCREEN,,,1

avviene una nuova inversione che riporta la visualizzazione ai colori iniziali. Queste regole costituiscono la sintassi dell'istruzione SCREEN. In seguito verrà data solo la forma più corrente delle istruzioni oppure quella che si renderà necessaria. Le altre possibilità relative ad ogni istruzione sono contenute nella guida di riferimento.

Messaggi di errore del PC 128

Dopo aver dato istruzioni quali PRINT o SCREEN, può capitare che appaia il messaggio "Syntax error" (Errore di sintassi). Questo messaggio significa che il PC 128, o più precisamente il programma interno che deve intrepretare quanto viene inserilo da tastiera, non è in grado di riconoscere una data parola. Questo programma interno si chiama interprete BASIC.

L'interprete BASIC è responsabile della traduzione delle istruzioni che vengono inserite per permettere al microcomputer di comprenderle. Infatti il PC 128 e l'utente non parlano la stessa lingua. L'utente si sforza di parlare in BASIC; l'interprete si incarica di tradurre le istruzioni. Quando viene visualizzato il messaggio "Syntax Error", molto spesso significa che si è verificato un errore di battitura (per esempio PRIT al posto di PRINT) e che quindi la sintassi dell'istruzione risulta incomprensibile. L'utilizzo dell'editor a piena pagina permette di correggere l'errore. Un attro errore dato dall'istruzione SCREEN può essere dovuto at fatto che viene indicato un numero maggiore di 15 per un colore oppure che si è dimenticato di inserire la virgola tra i parametri (per esempio tra il parametro del colore dei caratteri e quelto dello sfondo). In questo caso appare il messaggio seguente:

Illegal Function Call
(Richiamo funzione illegale)

Questo significa che l'interprete ha capito che si volevano cambiare i colori della visualizzazione (e quindi che si trattava di un'istruzione SCREEN) ma i parametri proposti non corrispondono ai colori autorizzati.

Nota: l'elenco dei messaggi di errore è contenuto nell'appendice.

Capitolo 4

Stampare qualsiasi cosa, ovunque

... l'importante è sapere come.
Proviamo.

```
PRINT CASA
0
OK
```

Come si è visto, invece della stampa di CASA appare uno 0. Anche il comando PRINT COSA darebbe esattamente lo stesso risultato e quindi l'effetto del comando non dipende affatto dalla parola che viene usata.
La spiegazione di quanto sopra verrà data in dettaglio nel capitolo 6. Il modo corretto per visualizzare la parola CASA è di scriverla tra virgolette.

```
PRINT "CASA"
CASA
OK
```

Attenzione: Si usano le stesse virgolette sia all'inizio che alla fine della parola. Se la parola PRINT è seguita dalle virgolette, il computer visualizzerà tutto quello che è stato inserito tra le due virgolette. A questo punto sembrerebbe che sia impossibile visualizzare le virgolette. Ma la parola impossibile non fa parte del vocabolario BASIC. Si troverà una soluzione a questo problema alla fine del capitolo 10.

L'utente è libero di inserire tra virgolette tutti i caratteri che vuole, compresi degli spazi vuoti e dei numeri. L'insieme di questi caratteri viene definito stringa di caratteri. La lunghezza massima delle stringhe è di 255 caratteri, più di sei righe su video da quaranta colonne. Per visualizzare dei testi lunghi è necessario usare diverse stringhe. Se la stringa di caratteri supera i 40 caratteri, verrà visualizzata su più righe. Ricordarsi di non confondere una riga del video con una riga di istruzioni. La fine di una riga di istruzioni è data dal tasto ENTER che non è visibile su video. Se la stringa occupa più di una riga, verrà divisa in funzione della dimensione del video senza tener conto delle parole.

Esempio:

```
PRINT "OOMANI SARA' VERAMENTE UNA BELLIS
SIMA GIORNATA DI SOLE"
DOMANI SARA' VERAMENTE UNA BELLISIMA GIO
RNATA OI SOLE
OK
```

Precisazione: non confondere una stringa vuota (due virgolette succesive) con una stringa che contiene uno o più spazi vuoti ottenuti con la barra spaziatrice. Per distinguere i due risultati, bisogna semplicemente tener conto del fatto che tutti i caratteri occupano lo stesso spazio sia su video che su stampante.

Le istruzioni:

```
PRINT""
PRINT"  "
PRINT"    "
```

danno lo stesso risultato (e cioè niente) ma il computer non ha commesso un errore. Non accetterà per esempio una stringa con più di 255 spazi vuoti.
Osserviamo questi due nuovi esempi.

```
PRINT "12+3"
12+3
OK

PRINT 12+3
15
OK
```

Si risparmia tempo sostituendo la parola PRINT con un semplice punto di domanda ?. L'interprete BASIC tradurrà ? con l'istruzione PRINT facendo risparmiare del tempo.

Oppure è possibile premere insieme il tasto BASIC ed il tasto ? e visualizzare la parola PRINT con un solo comando.

Fino ad ora è stata data la possibilità di scrivere una sola istruzione. Il PC 128 non può fare diverse cose contemporaneamente ma permette di concatenare molto rapidamente le operazioni da svolgere.

Come concatenare le operazioni grazie ai separatori

I due punti

E' possibile scrivere diverse istruzioni sulla stessa riga a condizione di separarle con i due punti. Visto che per ora si è preso in esame solo l'istruzione PRINT, potremo concatenare solo dei comandi di visualizzazione.

```
PRINT 1914:PRINT 1918
  1914
  1918
OK
```

Si osservi che il secondo comando PRINT dà automaticamente il comando di a capo. La stessa cosa avviene con le stringhe di caratteri.

```
PRINT"BELLA":PRINT"CASA"
BELLA
CASA
OK
```

Il comando PRINT senza argomenti permette di far saltare una riga.

```
PRINT "ROSSI":PRINT:PRINT"BIANCHI"
ROSSI

BIANCHI
OK
```

Il punto e virgola

Per visualizzare più stringhe sulla stessa riga, o meglio, una stringa di seguito ad un'altra, si inserisce un punto e virgola dopo il numero o la stringa di caratteri da visualizzare.

```
PRINT 1914:1918
  1914      1918
OK
```

Lo spazio vuoto che separa i due numeri è riservato al segno, in questo caso il segno + che non viene visualizzato. L'esempio precedente potrebbe essere realizzato in modo diverso.

```
PRINT 1914;:PRINT 1918
  1914      1918
OK
```

Se quindi l'utente decide di scrivere due istruzioni di visualizzazione diverse, dovrà assolutamente separarle con un segno di due punti. In caso di omissione o di inversione apparirà un messaggio di errore.

Attenzione: Il video è composto da 25 righe da 40 colonne. Le righe possono essere individuate con un numero da 0 a 24 dall'alto in basso e le colonne da 0 a 39 da sinistra a destra. E' possibile chiedere una visualizzazione in un dato punto del video anteponendo al comando PRINT il comando LOCATE.

Esempio:

```
LOCATE 10,15:PRINT "BUONGIORNO"
```

La parola BUONGIORNO verrà visualizzata al centro del video.
La B di Buongiorno è alla colonna 10, riga 15.

Capitolo 5

Attributi di visualizzazione

La virgola

Nei comandi di stampa la virgola viene utilizzata allo stesso modo del punto e virgola ma serve per delle visualizzazioni sulla stessa riga in campi di tredici caratteri. Se si tratta di visualizzazione dei numeri, il primo viene visualizzato a partire dalla prima colonna, il secondo a partire dalla quattordicesima colonna, il terzo a partire dalla ventisettesima colonna, e così via. Questo segno viene usato per creare delle videate ben impostate anche quando la lunghezza dei numeri e delle stringhe di caratteri è variabile.

```
PRINT 1,2,3
1          2          3
OK
PRINT 10,100,1000
10         100        1000
OK
PRINT"BELLA","CASA","GRANDE"
BELLA      CASA       GRANDE
OK
```

Riepilogo della punteggiatura

: separa due istruzioni nella stessa riga di istruzione.

; se posto dopo un numero o una stringa di caratteri da stampare, il punto e virgola forza una visualizzazione sulla stessa riga subito dopo quella precedente.

, se posta tra due numeri o due stringhe di caratteri, la virgola provoca la visualizzazione nella riga ogni quattordici colonne a partire dalla prima.

? abbreviazione del comando PRINT.

. viene usato come virgola per i numeri decimali.

Scrittura grande o piccola

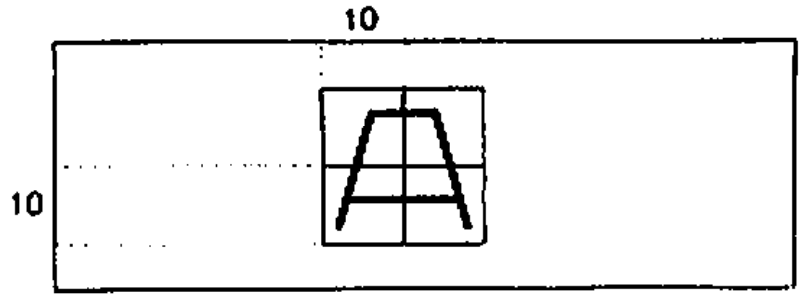
Sono disponibili quattro dimensioni diverse per visualizzare i caratteri. Vengono selezionate con il comando ATTRB.

	Altezza	Larghezza	Esempio
ATTRB 0,0	normale	normale	
ATTRB 1,0	normale	doppia	
ATTRB 0,1	doppia	normale	
ATTRB 1,1	doppia	doppia	

Se l'istruzione ATTRB non viene espressa, si sottintende ATTRB 0,0.
Quando viene data un'istruzione ATTRB, resta valida fino a nuovo ordine.
L'editor BASIC torna sempre alla modalità ATTRB 0,0.

Osservazione importante

In caso di stampa con carattere doppio, l'istruzione LOCATE è riferita alla posizione in basso a sinistra del carattere.



ATTRB 1,1:LOCATE 10,10:PRINT"A"

Si noterà quindi che la stampa di un carattere con altezza doppia sulla riga 0 lascerà apparire soltanto la parte inferiore del carattere.

Come cambiare il colore della visualizzazione: COLOR

Per mettere in rilievo una o più parole su una videata, è possibile cambiare il colore dei caratteri che compongono questa parola.
Per esempio con l'istruzione COLOR 1,4, la visualizzazione avverrà in rosso (1) su sfondo blu (4). L'istruzione COLOR ha una sintassi simile a quella del comando SCREEN dato che la prima cifra indica il colore dei caratteri mentre la seconda quella dello sfondo. Tuttavia questa istruzione agisce soltanto sulle visualizzazioni seguenti, mentre l'istruzione SCREEN agisce su tutto lo schermo. Queste prove hanno ulteriormente confermato le molteplici possibilità offerte dal microcomputer. Le due osservazioni che seguono riguardano alcune particolarità che possono essere utili:

Osservazione importante

Un'istruzione come:

LOCATE 35,10:PRINT"BELLA-CASA"

pone un problema che il PC 128 risolve da solo. Lanciando la stampa della stringa BELLA-CASA (10 caratteri) alla colonna 35, è impossibile non dividere la stringa. Il PC 128 decide di non scrivere la stringa nel punto richiesto ma all'inizio della riga seguente. Prima di eseguire ogni istruzione PRINT, il PC 128 valuta quindi se è possibile la stampa senza saltare una riga. In caso contrario, comincia la stampa all'inizio della riga seguente. Questo comportamento fornisce sia dei vantaggi sia degli inconvenienti. Se si vuole forzare la visualizzazione in un punto determinato, sostituire il comando PRINT con il comando PRINT USING "&" e consultare la "Guida di riferimento".

Perfezionamento della funzione LOCATE

L'istruzione LOCATE prevede un terzo argomento che decide l'eventuale presenza del cursore.

In alcuni video grafici, quando viene visualizzata una bella immagine, la presenza del cursore lampeggiante potrebbe rovinarne l'effetto.

LOCATE X,Y,0

In questo modo il cursore non apparirà sul video.

LOCATE X,Y,1

In questo modo il cursore riapparirà sul video.

Un problema particolare

```
PRINT CASA+5
 5
OK
PRINT CASA-3
-3
OK
PRINT CASA/2
0
OK
```

Ci si chiederà a che numero corrisponde CASA. Osservando con un po' più di attenzione, sarà facile indovinarlo. In caso contrario, la risposta è nel capitolo seguente... a condizione di essere predisposti per la storia.

Capitolo 6

Variabili

Avrà sicuramente sorpreso l'ignoranza palese dei microcomputer. Al di là delle tavole di moltiplicazione e di altre minime cognizioni, non sanno assolutamente niente. Ad esempio non conoscono la data della battaglia di Caporetto.

```
PRINT CAPORETTO
0
OK
```

Notiamo che il nostro allievo non ha studiato la lezione dalo che la battaglia di Caporetto non è avvenuta nell'anno 0 ma nel 1917. Quindi l'utente dovrà agire da insegnante per aiutare il povero computer a elevare un po' il suo livello culturale. Ecco come fare.

```
CAPORETTO=1917
OK
PRINT CAPORETTO
1917
OK
```

Per capire: l'istruzione CAPORETTO=1917 è quella che viene definita un'istruzione di *assegnazione*. In questo modo si ordina al computer di porre in una o più celle di memoria il numero 1917. Si potrà richiamare questo numero chiamandolo CAPORETTO. E' quello che avviene con la seconda istruzione.
L'utente chiede di stampare il contenuto delle celle di memoria CAPORETTO. Il computer lo va a cercare dove lo ha memorizzato e lo restituisce perfettamente: 1917. A questo punto l'utente potrà chiedere tutte le volte che vorrà il contenuto di CAPORETTO e il computer risponderà con precisione assoluta.

Variabili

Nell'esempio prscedente, CAPORETTO è una variabile. Questo nome non è molto appropriato in quanto il contenuto della variabile CAPORETTO non ha alcun motivo di variare. La variazione potrebbe avvenire solo in seguito ad una nuova istruzione di assegnazione. Ecco un esempio che usa le due variabili A e B ed offre la possibilità di riutilizzare i separatori (vedere capitolo 4).

```
A=25.7:B=140:PRINT A:B
25.7      140
OK
```

Prima di qualsiasi istruzione di assegnazione, il contenuto delle variabili è inizializzato a zero. Ecco perchè quindi quando lo si accende, il PC 128 è convinto che la battaglia di Caporetto si sia svolta nell'anno 0.

Trasferimento

Il contenuto di una variabile A può essere trasferito in una variabile B. In questo modo si ottiene una duplicazione.

```
A=75:B=0:PRINT A;B
75      0
OK
B=A:PRINT A;B
75          75
OK
```

All'inizio A vale 75 e B vale 0.
L'istruzione B=A trasferisce il valore di A in B (si può pensare come B ← A).
Non bisogna dimenticare che questa istruzione cancella il contenuto precedente di B di cui non resta alcuna traccia.

Permuta

Sono date due variabili, una si chiama X e contiene il numero 10, l'altra si chiama Y e contiene il numero 25.
Si vuole permutare il contenuto di queste due variabili, e cioè mettere il contenuto di X in Y e viceversa. Questo viene ottenuto tramite la seguente istruzione:

```
X=25:Y=10
```

Immaginiamo ora che non si conosca il contenuto di X e Y e che non possiamo scoprirlo usando l'istruzione PRINT X e PRINT Y. Questo piccolo enigma farà riflettere.
Prima di offrire una soluzione, non possiamo fare a meno di dire due parole su una pista sbagliata che verrebbe voglia di seguire immediatamente.
Dato che bisogna permutare X e Y, si potrebbe pensare di mettere X in Y e poi Y in X. E' una idea abbastanza buona ma nel momento in cui si assegnerà il valore di X a Y si perderà il contenuto di Y. Mettendo poi Y in X, in realtà è il contenuto di X che viene rimesso in X.
Da questa idea sbagliata ne nasce una migliore: prima di trasferire il contenuto di X in Y bisogna trasferire il contenuto di Y in una variabile accessoria U.
Da qui le seguenti istruzioni:

```
X=25:Y=10:PRINT X;Y
25     10
OK
U=Y:Y=X:X=U:PRINT X;Y
10      25
OK
```

Ricordare comunque questa procedura ma notare che l'istruzione SWAP dà lo stesso risultato:

```
A=10:B=20:PRINT A;B
10      20
OK
SWAP A,B:PRINT A;B
20      10
OK
```

Curiosità: per richiamare una variabile (in questo caso Caporetto), il PC 128 utilizza un indirizzo, cioè un numero. La memoria RAM viene quindi suddivisa in celle ognuna delle quali ha un suo riferimento.
Una cella di memoria può contenere un numero, un carattere, il contenuto di un'area del video, una parte dell'istruzione PRINT, ecc. Il numero 1917 è sistemato in una parte della memoria RAM. Una semplice istruzione BASIC permette di saperlo: VARPTR.

```
PRINT VARPTR(CAPORETTO)
```

Il numero che si otterrà dipende da quello che si è fatto prima di quest'istruzione ed indicherà l'indirizzo della prima cella di memoria in cui si trova il contenuto di Caporetto. Questa indagine potrebbe essere approfondita esplorando il contenuto della cella di memoria indicata. A questo punto si dovrà apprendere alcuni segreti contenuti nel capitolo 36. Il contenuto di una variabile generica viene di solito disposto in celle di memoria consecutive e viene sottoposto ad una complessa codifica matematico–informatica. Ricordiamo che il BASIC lascia il compito di organizzare la disposizione delle celle di memoria al microcomputer per evitare ogni preoccupazione all'utente.

Nomi delle variabili: Non devono cominciare con una parola chiave del BASIC (SCREEN, COLOR, ...). L'elenco di queste parole è contenuto nell'appendice 5. Quindi l'istruzione

```
VALORE=PREZZO*SCONTO
```

genera il messaggio "Syntax Error" (errore di sintassi) perchè VALORE comincia con VAL che è una delle parole chiavi del BASIC.

Capitolo 7

Stringhe e altre variabili

Torniamo al trattamento delle stringhe di caratteri (vedere capitolo 4). Anche per le stringhe è possibile utilizzare l'assegnazione. La sola precauzione da prendere è quella di fare seguire da un simbolo $ il nome della stringa che si vuole memorizzare. Il simbolo $ indicherà all'interprete che si tratta di una stringa di caratteri e non di un numero. L'esempio che segue prevede la memorizzazione della stringa "CAPORETTO" sotto il nome B$.

```
B$="CAPORETTO":PRINT B$:PRINT B$
CAPORETTO
CAPORETTO
OK
```

Tutto ciò che è stato detto precedentemente sulle regole dell'assegnazione è valido anche per le variabili stringa. Quello che le differenzia è la presenza del simbolo $ alla fine del nome della variabile e la presenza delle virgolette. E' facile capire che non bisogna confondere le variabili stringa e le variabile numeriche, poiché un microelaboratore non ammette confusioni.
Le istruzioni del tipo D="CAPORETTO" e B$=1917 determinano automaticamente un messaggio d'errore. Per la stessa ragione, non è possibile effettuare alcun trasferimento tra variabili stringa e variabili numeriche: A$=B o B=A$ determinano messaggi d'errore. E' invece possibile effettuare permutazioni fra stringhe di caratteri, così come è possibite fra variabiti numeriche.

```
A$="MO":B$="RA":PRINT AS;B$
MORA
OK
SWAP AS.B$:PRINT A$;BS
RAMO
OK
```

Da quanto visto, emerge una differenza importante fra variabili numeriche e variabili stringa di caratteri (o stringhe alfanumeriche). Queste variabili sono di tipo differente.
Per le variabili numeriche, esistono delle sotto-categorie che è opportuno conoscere: quasti tipi corrispondono a particolari tecniche di memorizzazione.

Variabili Intere

Possono essere utilizzate per gestire numeri interi con valore compreso fra -32768 e 32767. Per indicare che una variabile è di tipo intero, si possono utilizzare due diversi metodi:

1. Far seguire il nome della variabile dal segno %
Es.:
A%=1854

2. Far precedere l'assegnazione della dichiarazione DEFINT che dispensa dall'indicazione del segno %. Questo metodo permette la realizzazione di arrotondamenti automatici.

```
DEFINT A:A=9.86:PRINT A
10
OK
```

Variabili reali a singola precisione

Quando di una variabile non viene indicato il tipo, essa sarà considerata una variabile reale a precisione singola. Queste variabili prendono in considerazione un massimo di sette cifre significative. Il loro nome può essere seguito da un simbolo ! (facoltativo).

Variabili reali a doppia precisione

Questo tipo di variabili si utilizza come nel caso precedente, ma tiene in considerazione fino a quindici cifre significative. Il nome delle variabili deve essere seguito da un simbolo #, oppure il loro tipo deve essere definito precedentemente con l'istruzione DEFDBL. Questo tipo di variabile può essere utilizzato per aumentare la precisione di certi calcoli, come ad esempio la divisione; lo si può rilevare dall'esempio seguente:

```
DEFOBL X
B=10/7:X=10#/7#:PRINT B:PRINT X
1 42857
1.4285714285714 29
OK
```

Si può notare che nel caso di calcolo a doppia precisione, per effettuare operazioni a doppia precisione, le costanti 10 e 7 devono essere seguite dal segno #.

Un unico ordine di stampa può mescolare variabili numeriche e stringhe alfanumeriche.

```
A=48:B=1024
PRINT A;"X";B;"=";A*B
48 X 1024 =49152
OK
```

Operazioni numeriche

Ecco l'esempio più semplice.

```
A=15+16
OK
```

Dietro questa semplice istruzione, si trova il seguente concatenamento:
1. Effettuare l'operazione 15+16
2. Cercare una locazione libera in memoria
3. Memorizzarvi il risultato dell'operazione
4. Assegnare al risultato il nome A

Per assicurarsi della buona riuscita di questa manovra:

```
PRINT A
31
OK
```

L'interesse a mantenere in memoria i risultati è dato dalla possibilità di poterli richiamare e riutilizzare per altri calcoli.

```
X=15:Y=16:Z=X+Y
OK
```

Riassumendo:

1. Memorizzare il numero quindici nella variabile X.
2. Memorizzare il numero sedici nella variabile Y.
3. Prendere il contenuto di X e quello di Y.
4. Sommare i due numeri.
5. Memorizzare il risultato in Z.

La trasformazione del contenuto di una variabile può essere fatta a partire dal contenuto della variabile stessa. In questo caso si crea un contatore.

```
A=10:PRINT A
10
OK
A=A+1:PRINT A
11
OK
```

La seconda riga di istruzioni ha permesso di aumentare di una unità il contenuto della variabile A: prendere il contenuto di A, aggiungere 1, memorizzare il risultato in A. In questo caso si è avuto un incremento del contatore A. Nello stesso modo si può avere un decremento.

```
A=10:PRINT A
10
OK
A=A-1:PRINT A
9
OK
```

Capitolo 8

Operazioni su stringhe alfanumeriche

Concatenamento:+

Questa operazione consiste nell'affiancare due stringhe in modo che queste diventino un'unica stringa alfanumerica.

```
A$="ABCD":B$="EFG"
PRINT A$+B$
ABCDEFG
OK
```

Il risultato di un concatenamento può essere memorizzato. L'esempio che segue costruisce due slogan: S1$ e S2$. Il lettore si renderà presto conto che i nomi delle variabili alfanumeriche, come per quelle numeriche, possono prevedere delle lettere. Si dovrà prevedere, volendo, l'inserimento di spazi vuoti nelle stringhe da concatenare, per poter separare le singole parole.

```
A$="VIVA ":B$="ABBASSO ":M$="LA JUVE!"
S1$=A$+M$:S2$=B$+M$
PRINT S1$:PRINT S2$
VIVA LA JUVE
ABBASSO LA JUVE
OK
```

Parte sinistra: LEFT$

La funzione LEFT$ estrae la parte sinistra di una stringa, di cui bisogna precisare nome e contenuto e il numero di caratteri da considerare.

```
PRINT LEFT$("ABCDEF",4)
ABCD
OK
```

Nell'utilizzare le variabili con questa funzione bisogna fare attenzione nel non richiedere un numero di caratteri superiore alla lunghezza della stringa considerata.

```
A$="CASA"
L1$=LEFT$(A$,1):L2$=LEFT$(A$,2)
L3$=LEFT$(A$,3):L4$=LEFT$(A$,4)
PRINT L1$:PRINT L2$:PRINT L3$:PRINT L4$
C
CA
CAS
CASA
OK
```

Parte destra: RIGHT$

La funzione RIGHT$ estrae la parte destra di una stringa. Le modalità per il suo impiego sono le stesse indicate per la funzione LEFT$.

```
PRINT RIGHT$("ABCDEF",4)
COEF
OK
```

Parte centrale: MID$

Per estrarre una parte di stringa, bisogna precisare quanto segue:
– Di quale stringa si tratta.
– La posizione del primo carattere da considerare.
– Il numero di caratteri da considerare.
Quindi: MID$(A$,1,2) considererà, nella stringa A$, 2 caratteri a partire dal primo.

```
PRINT MID$("ABCDEF",3,2)
CD
OK
PRINT MID$("ABCDEF",2,3)
BCD
OK
```

Lunghezza di una stringa: LEN

La funzione LEN dà la lunghezza di una stringa. Questo dato può essere memorizzato in una variabile numerica: è sempre un numero intero compreso tra 0 e 255.

```
A$="ABCDE":L=LEN(A$):PRINT L
5
OK
```

Ripetizione di una sottostringa in una stringa: INSTR

X$ e Y$ sono due stringhe. La funzione INSTR viene utilizzata per fornire la posizione di Y$ in X$ (o viceversa). INSTR(X$, Y$) dà la posizione, in X$, nella quale Y$ è stato individuato.

```
PRINT INSTR("BASIC","S")
     3
OK
```

Si può dedurre che la S appare per la prima volta in BASIC in terza posizione. Se la stringa Y$ non è presente in X$, la funzione INSTR riporta 0. La funzione INSTR è molto utile nei programmi interattivi che necessitino di un'analisi della risposta.

Dopo una domanda posta dal programma, si memorizza la risposta in una variabile stringa, nella quale sia possibile verificare la presenza o meno delle parole chiave.

Cambiamenti di tipi

Le variabili previste in memoria sono di due tipi differenti, fra di loro incompatibili: numeriche o stringhe. Esistono due funzioni che permettono di passare da un tipo all'altro, permettono cioè di trasformare una variabile numerica in una stringa alfanumerica e viceversa.

Passaggio dal tipo numerico al tipo stringa: STR$

Se si inserisce il numero 123 nella variabile X con X=123, sarà possibile trasformare questo numero in una stringa alfanumerica con STR$(X).

```
X=123:A$=STR$(X)
PRINT A$
  123
OK
```

Apparentemente sul video non esiste alcuna differenza tra X e A$: in realtà X è un numero e A$ è una stringa di caratteri. A$ non potrà essere utilizzato come numero, non lo si potrà nè moltiplicare nè dividere. Al contrario con A$ si potranno utilizzare tutte le operazioni eseguibili su stringhe alfanumeriche.

```
A=123:B=45
X$=STR$(A):Y$=STR$(B)
PRINT A+B:PRINT X$+Y$
  168
  123  45
OK
```

In questo esempio si può facilmente notare la differenza nell'impiego dell'operatore +: addizione per le variabili numeriche, concatenazione per le stringhe alfanumeriche. Lo spazio vuoto che separa 123 e 45 evidenzia lo spazio del segno che non è visualizzato.

Passaggio dal tipo stringa al tipo numerico: VAL

La funzione VAL non trasforma le lettere in numeri, ma se un numero è presente all'inizio di una stringa, esso sarà estratto e sarà possibile memorizzarlo in una variabile numerica. Così VAL("1917 CAPORETTO") è uguale al numero 1917. Se la stringa contiene più numeri separati da caratteri, sarà considerato solo il primo di questi numeri.

VAL("14 LUGLIO 1789") è uguale a 14

Capitolo 9

Ulteriori informazioni sulle stringhe

Lunghezza di un numero

Associata a STR$, la funzione LEN permette di conoscere la lunghezza di un numero, cioè il numero di cifre di cui è composto.

```
X=1917:A$=STR$(X)
L=LEN(A$)-1:PRINT L
4
OK
```

Si è sottratto 1 a LEN(A$) tenendo conto della posizione del segno.

Carattere nullo e spazio

Nel capitolo 4, è stata sottolineata la differenza tra le stringhe nulle e le stringhe riempite di spazi. La funzione LEN permette di misurare con precisione questa differenza.

```
A1$="":A2S="  ":A3S="    "
PRINT LEN(A1$):LEN(A2$):LEN(A3$)
0  1  4
7
OK
```

La stringa A1$ non contiene caratteri, la stringa A2S ne contiene uno e la stringa A3$ ne contiene quattro.
Associando le funzioni VAL e STR$ è possibile isolare le singole cifre di un numero, componendone uno nuovo.

Esempio: per estrarre la prima cifra di un numero:

```
X=7259:A$=STR$(X)
B$=LEFT$(A$,2):PC=VAL(B$)
PRINT PC
7
OK
```

Commenti

1. Il vantaggio di questo metodo è quello di poter trattare un numero X qualunque, anche se l'utente non ne conosce la lunghezza e il valore.

2. In B$, sono stati isolati i primi due caratteri di A$ tenendo conto del carattere spazio introdotto in A$ per il segno.

3. Votendo, è possibile sintetizzare le istruzioni per il calcolo di X in una sola:

PRINT VAL(LEFT$(STR$(X),2))

Osservare la coerenza nell'uso delle parentesi.
A chiusura di questa parte dedicata alle stringhe alfanumeriche e alle variabili numeriche, si pone l'accento su alcune difficoltà che potrebbero sorgere. E' opportuno non dimenticare che in caso di inserimento di dati errati nel PC 128, questo può segnalare un errore non prevedibile. Si consiglia quindi di verificare la coerenza dei risultati ottenuti.

Precauzioni

1. VAL(75) e VAL(X) causano un errore di tipo Type Mismatch (Errore tipo). Per le stesse ragioni STR$("12") e STR$(A$) sono errori di tipo. Quando il primo carattere di una stringa non è né uno spazio, né una cifra, né un punto, il valore VAL di detta stringa è 0. Così VAL("A5") è uguale a 0. Attenzione: sul PC 128 si avrà VAL("2+3")=2 e non VAL("2+3")=5 come ci si potrebbe aspettare.
Al contrario una funzione EVAL (associata a CRUNCH$) effettua questa operazione. Consultare la guida di riferimento, ma fare attenzione poichè questa funzione non lavora in modo diretto.

2. Anche se il calcolatore accetta PRINT LEFT$(A$,0) e risponde "correttamente" a PRINT RtGHT$("123",5) dando "123", è preferibile evitare di porre simili problemi.

Ultima nota sulle assegnazioni

Riprendiamo gli esempi riguardanti le variabili numeriche. A partire da una o più variabili numeriche, è possibile effettuare su di esse qualsiasi tipo di operazione: calcoli e trasferimenti.

Ecco qualche esempio:

```
A=14:B=12:C=16
MEDIA=(A+B+C)/3:PRINT MEDIA
14
OK
```

```
A=3.5:B=A+A:C=B+B
PRINT A,B,C
3.5       7         14
OK
```

Vediamo di spiegare la differenza fra i due esempi che seguono.

```
A=1:B=2:C=A-B:B=C:A=B
PRINT A,B,C
3         3         3
OK
```

```
A=1:B=2:C=A+B:A=B:B=C
PRINT A,B,C
2         3         3
OK
```

Risulta evidente che l'ordine in cui sono scritte le istruzioni di assegnazione ha un ruolo molto importante. La riflessione e l'abitudine aiuteranno il lettore a superare gli inevitabili errori dei primi tentativi.

Capitolo 10

Il codice ASCII

ASCII è una parola formata dalle iniziali di *American Standard Code Information Interchange*.
Il codice ASCII dà una rappresentazione numerica dei caratteri: lettere alfabetiche sia maiuscole che minuscole, cifre, operatori aritmetici e simboli di punteggiatura, caratteri di controllo. L'elenco completo si trova nell'Appendice 4.
Si incontra di nuovo la nozione di codifica di una informazione che era già stata introdotta nel capitolo riguardante i colori del video.
Una particella elementare di informazione, equivalente a SI o NO, si chiama bit. Le lettere dell'alfabeto, le cifre, gli operatori, ecc., sono rappresentabili con una serie di bit. Il PC 128, o meglio, il microprocessore in esso presente, è in grado di interpretare una serie di "0" e di "1": sa come trattarli, memorizzarli e restituirli quando richiesto.
Quando l'utente preme il carattere A (maiuscolo) il PC 128 lo trasforma in bit nel modo seguente:

1	0	0	0	0	0	1

Più precisamente, il microprocessore tratta pacchetti di otto bit alla volta. Questi pacchetti di otto bit vengono denominati byte. Di conseguenza la codifica completa del carattere A sarà:

0	1	0	0	0	0	0	1

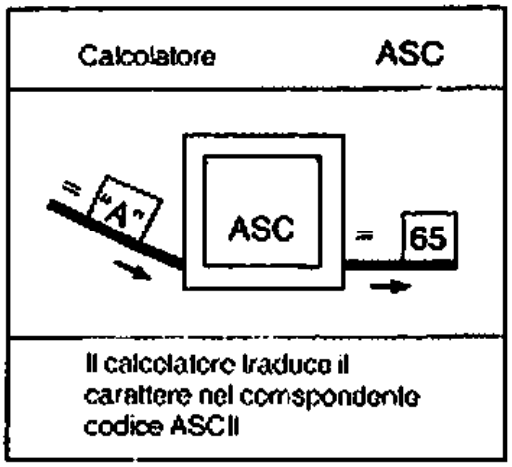
Il codice ASCII, tramite i byte, permette la rappresentazione di tutti i caratteri della tastiera. Chi conosce il principio della numerazione in base 2, può notare che il byte proposto per la codifica della lettera A è la rappresentazione del numero 65 in base 2.
E' in questo modo che il codice ASCII assegna un valore numerico a ciascun carattere presente sulla tastiera del PC 128 e anche a caratteri non presenti.
Il BASIC dà la possibilità, attraverso due sue funzioni, di conoscere immediatamente il codice ASCII di un carattere e di identificare un carattere di cui se ne conosce il codice. Queste due funzioni trasformano una variabile numerica in una stringa alfanumerica e viceversa, come avviene per VAL e STR$.

ASC

Il codice ASCII del carattere A è 65. La funzione ASC lo conferma.

```
PRINT ASC("A")
65
OK

PRINT ASC("7")
55
OK
```

In effetti 55 è il codice ASCII del carattere 7. E' possibile illustrare questa trasformazione.



La funzione ASC è applicabile anche a stringhe di più caratteri. In questo caso, darà il codice del primo carattere della stringa.

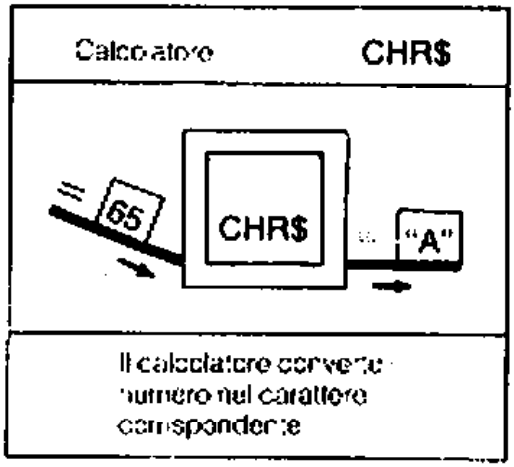
```
PRINT ASC("CAPORETTO")
67
OK

PRINT ASC(" ")
32
OK
```

Il primo carattere della stringa è uno spazio. Il suo codice ASCII è 32. In caso di richiesta di codice ASCII di una stringa nulla, si avrà un messaggio di errore in quanto non esiste codice.

CHR$

E' la funzione reciproca di ASC. Dà il carattere corrispondente al codice ASCII inserito.

```
PRINT CHR$(65)
A
OK
```



Attenzione!

```
PRINT CHR$(49);CHR$(43);CHR$(49);CHR$(61);CHR$(51)
1+1=3
OK
```

In questo caso il PC 128 ha semplicemente visualizzato le cifre, senza verificare il risultato del calcolo in quanto l'istruzione data era solo di scrivere le cifre. CHR$ restituisce sempre una stringa alfanumerica che può essere trattata come tutte le altre stringhe.

Esempio:

```
A$=CHR$(80)+CHR$(69):A$=A$+A$
PRINT A$
PEPE
OK
```

Precauzioni

CHR$ lavora su numeri e ASC su stringhe. Di conseguenza ASC(29), ASC(X), CHR$("A"), CHR$(M$) determinano automaticamente errori di tipo Type Mismatch.

Altri esempi:

```
PRINT ASC(CHR$(65))
65
OK

PRINT CHR$(ASC("A"))
A
OK
```

Caratteri di controllo

I caratteri presenti sulla tabella dei codici ASCII con codifica compresa tra 0 e 31 sono caratteri di controllo. Non sono stampabili ed agiscono direttamente sulla memoria centrale.

Esempio:

PRINT CHR$(7) provoca un segnale acustico
PRINT CHR$(12) cancella lo schermo
PRINT CHR$(30) porta il cursore in alto a sinistra

Virgolette

Il codice ASCII è formale: per visualizzare le virgolette, bisogna digitare:

```
PRINT CHRS(34)
```

Capitolo 11

Prima lezione di disegno

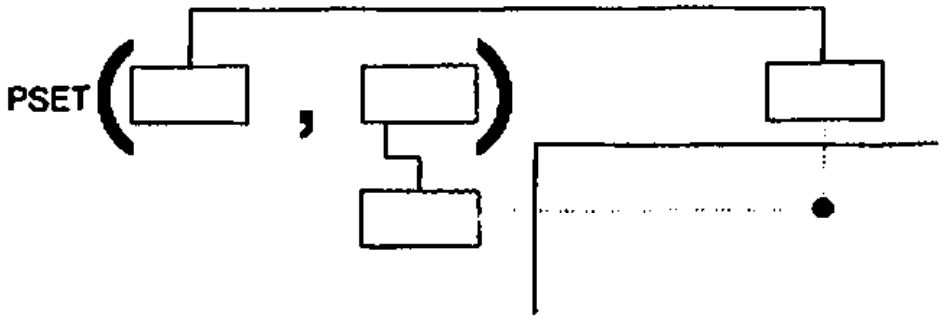
Una delle principali attrattive del PC 128 è la capacità che offre all'utente di poter creare delle immagini. Naturalmente, per poter utilizzare questo aspetto grafico, è necessario innanzitutto conoscere le funzioni grafiche del BASIC, ed inoltre ricordarsi (o imparare) i principi elementari della rappresentazione cartesiana dei punti. Vediamo di ricordarli rapidamente.
Ogni punto (pixel) dello schermo è reperibile con 2 numeri chiamati le coordinate del punto. Normalmente la prima coordinata è quella orizzontale (ascissa) e la seconda quella verticale (ordinata). Ogni coordinata è rappresentata da un numero che corrisponde ad una posizione su una scala graduata.

Per ta rappresentazione grafica si parla di schermo ad alta risoluzione: più il numero delle colonne e delle righe è elevato, migliore è la precisione e la ricchezza dell'immagine. Il PC 128 offre 3 modalità grafiche:

- 320 colonne e 200 righe
- 640 colonne e 200 righe
- 160 colonne e 200 righe

Attivazione di un punto

Un punto può eseere attivato con l'istruzione PSET(X,Y) dove X è un numero di colonna e Y è un numero di riga.



Per modificare il colore è sufficiente far seguire questa istruzione dal codice del colore desiderato.
Quindi: PSET(200,100),1 ← punto rosso
PSET(200,100),15 ← punto arancione

Tracciamento di una linse retta

Per tracciare una linea retta si usa l'istruzione LINE seguita dalle coordinate dei 2 punti che saranno le estremità della linea.
Quindi: LINE(50,50)–(250,150)
traccia una linea retta, che va dal punto di coordinate (50,50) a quello le cui coordinate sono (250,150).
Si noti così che l'istruzione LINE (250,150)–(50,50) porterà esattamente allo stesso risultato.

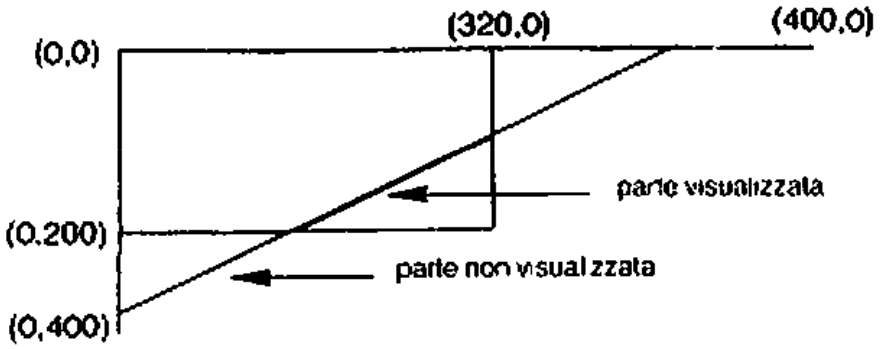
Al termine dell'istruzione si può precisare il colore esattamente come per l'istruzione PSET.

Quindi: LINE(0,0)–(319,199),4 traccia una diagonale blu.

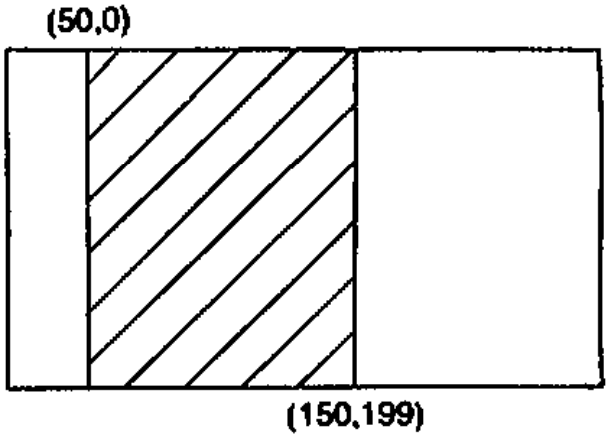
E' possibile omettere il punto di partenza della linea. In questo caso, viene sostituito dall'ultimo punto indicato. Quindi: LINE(10,20)–(50,100):LINE–(150,50) traccerà prima una linea dal punto (10,20) al punto (50,100) e poi un'altra dal punto (50.100) al punto (150,50).

Attenzione: l'istruzione LINE (come tutte le istruzioni grafiche) accetta numeri di colonne e di righe compresi tra –32768 e +32767. Si può quindi disegnare fuori dallo schermo visualizzato.

Esempio: LINE(400,0)–(0,400) darà

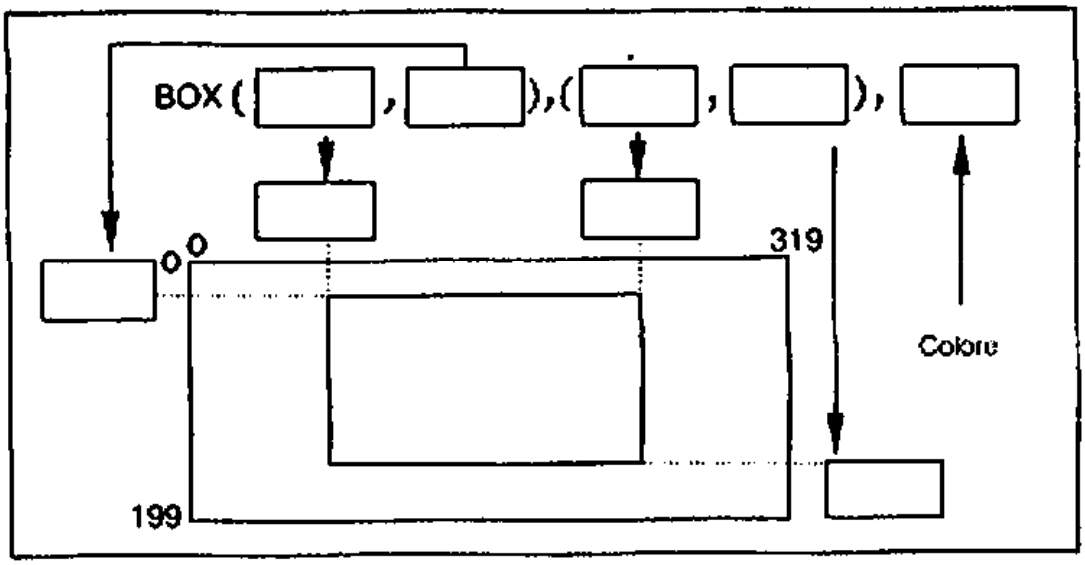


Essendo possibile disegnare fuori dallo schermo visualizzato, l'operazione sarà simile a quella effettuata su uno schermo grande (da –32768 a +32767 da entrambi i lati) tramite una piccola finestra le cui dimensioni corrispondono a quelle dello schermo (da 0 a 319 in colonne e da 0 a 199 in righe). L'istruzione WINDOW (finestra) permette di ridurre ulteriormente le dimensioni della finestra e di localizzarla dove si vuole all'interno della parte visualizzata. L'istruzione WINDOW (50,0)–(150,199) delimita un rettangolo–finestra all'interno del quale si disegneranno le immagini.



Per ritornare alla situazione iniziale, battere:
WINDOW(0,0)–(319,199)

Come disegnare un riquadro



Per ottenere un rettangolo, si utilizza l'istruzione BOX (riquadro) seguita dalle coordinate dei 2 punti che designano i 2 angoli opposti. I lati del rettangolo risulteranno paralleli ai bordi dello schermo.
Facendo qualche prova, ci si convincerà facilmente che dati due punti qualsiasi sullo schermo, esiste un solo rettangolo che ammette questi 2 punti come angoli opposti. Quindi non ha alcuna importanza l'ordine con cui si danno questi punti.
Come per PSET e LINE, l'istruzione BOX può essere seguita da un codice di colore.

Riquadri pieni

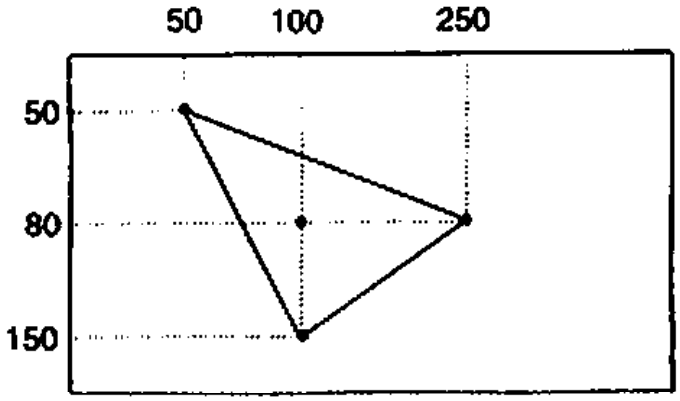
Se si sostituisce BOX con BOXF, il rettangolo verrà colorato completamente; la F sta per Fill che in inglese significa riempire.

Colorazione

E' possibile colorare l'interno di una figura chiusa con l'istruzione PAINT(X,Y) dove X e Y sono le coordinate di un punto interno alla figura.

Esempio:
Per colorare l'interno di un triangolo:

```
LINE(50,50)-(250,80),4:LINE-(100,150),4:
LINE-(50,50),4:PAINT(100,80),4
```



Capitolo 12

Seconda lezione di disegno

Tracciamento di un cerchio...

Il BASIC del PC 128 dispone dell'istruzione CIRCLE che permette di disegnare un cerchio di cui si forniscono le coordinate del centro e il raggio.

Esempio:

CIRCLE(200,150),30

disegna il cerchio il cui centro è (200,150) ed il cui raggio è 30.
Il suo colore può essere diverso da quello dei caratteri con la seguente precisazione:

CIRCLE(200,150),30,2

Sia il raggio, sia le coordinate del centro sia il colore possono essere definiti con delle variabili.

di un'ellisse...

La stessa istruzione CIRCLE permette di tracciare delle ellissi a patto che si precisino due parametri supplementari.
Se RH e RV designano i raggi orizzontali e verticali, si potrà ottenere l'ellisse con:

CIRCLE(C,L)RH,RV

Attenzione alla sintassi: niente virgola prima di RH; è proprio l'assenza della virgola che precisa che l'istruzione deve disegnare un'ellisse e non un cerchio.

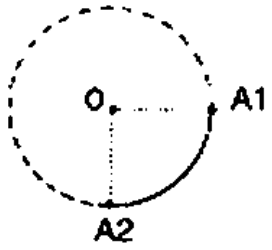
Esempio:

CIRCLE(160,100)70,50

traccia un'ellisse centrata in 160,100, il cui asse orizzontale è 70 e quello verticale è 50. Naturalmente è possibile determinare il colore tramite l'aggiunta dell'argomento relativo.
Si noterà che il tracciato dell'ellisse non utilizza i centri dell'ellisse stessa. Il loro utilizzo costituirebbe un buon tema di riflessione.

...e un arco di cerchio

E c'è di più; si può disegnare qualsiasi parte di cerchio in qualsiasi punto dello schermo.



Un arco, per esempio A1 A2, è limitato da due raggi OA1 OA2. La posizione di questi due raggi verrà precisata dagli angoli che formano con l'orizzontale: 0 gradi per OA1, 90 gradi per OA2.
Per il computer, gli angoli devono essere espressi in radianti. Sapendo che π =3,14 radianti corrispondenti a 180 gradi, non resta che applicare la regola del 3 semplice. Quindi 90 gradi = π /2 rd=1,57 rd
Se il cerchio è situato al centro dello schermo, si otterrà l'arco di cerchio A1 A2 con:

```
CIRCLE(160,100),50,0,1.57
```

I due angoli limite sono precisati dopo il raggio e separati da quest'ultimo tramite un punto e virgola. Si osservi che gli angoli vengono contati in senso orario a partire dal punto A1. Essendo l'asse delle y orientato verso il basso, si può ritrovare qui quel senso trigonometrico familiare ai matematici.

La scelta del modello: PATTERN

Un riquadro o un cerchio possono anche essere riempiti con un modello:

Esempio:

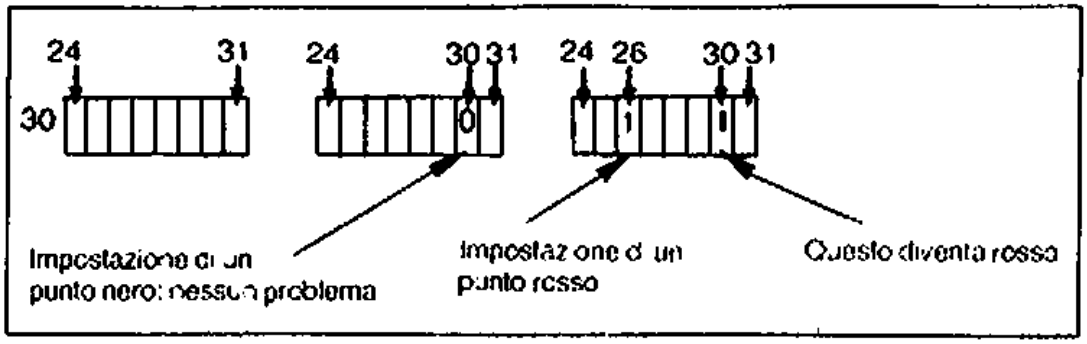
```
PATTERN "/"
BOXF(40,50)-(200,150)
```

riempirà il rettangolo designato con delle barre oblique.

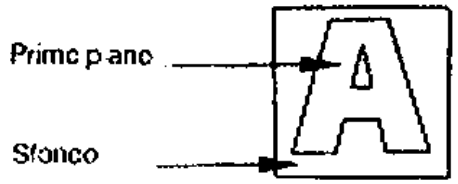
Attenzione: tutti i caratteri possono essere usati come modello.

Limiti dei colori

Supponiamo che lo schermo abbia un proprio colore iniziale (azzurro: 6). L'istruzione PSET(30,30),0 genera un punto nero a (30,30). L'istruzione PSET(26,30),1 genera un punto rosso a sinistra del precedente. Osservando attentamente il punto nero precedente ci si accorge che è diventato anch'esso rosso. Bisogna immaginare una suddivisione dello schermo in piccole barre orizzontali di 8 punti (ottetto). La ricezione dei colori è la seguente: ogni gruppo di 8 punti può ricevere solo 2 colori ed è l'ultimo punto impostato che influenza gli altri.



Si potrebbe pensare che il nero ed il rosso non siano in fondo altro che due colori e che il PC 128 avrebbe dovuto accettarli. Questo succede perchè non si tiene conto del colore dello sfondo dello schermo (azzurro): compreso quest'ultimo, vi sono ben tre colori presenti su questo ottetto e quindi ce ne è uno di troppo.
Anche qui, come per scrivere, occorrono due colori, uno per l'inchiostro (primo piano) ed uno per la carta (sfondo).



Tre delle modalità proposte risolvono il problema della limitazione dei colori e sono attivabili tramite l'istruzione CONSOLE (vedere "Guida di riferimento"). La prima modalità offre due colori, ma raddoppia il numero dei pixel; la seconda permette di utilizzare quattro colori senza limitazione e la terza offre sedici colori senza limitazione portando a 160 il numero delle colonne.

Passaggio di una linea attraverso un riquadro

Esempio 1:

```
SCREEN 4,11,11
BOXF(48,50)-(199,150),0
LINE(0,0)-(319,199),1
```

L'effetto osservato al momento del passaggio della linea gialla nel riquadro nero, dipende dalla limitazione dei colori. Ogni punto giallo attivato nel riquadro colora di giallo tutti i punti neri del segmento al quale appartiene.

Per ovviare a questo spiacevole inconveniente, basta attivare i punti del riquadro con un colore dello sfondo che mantenga visibile il passaggio dei punti gialli che sono del colore di primo piano.

Esempio 2:

```
SCREEN 4,11,11
BOXF(48,50)-(199,150),-1   <-- un codice colore negativo per un colore dello sfondo
LINE(0,0)-(319,199),1
```

Avvertenza: la colorazione con un colore di sfondo si ottiene dando un codice negativo ai colori: -(C+ 1) per il colore C.

Si ottiene quindi la seguente codifica:

Nero: -1
Rosso: -2
Verde: -3
ecc.

Capitolo 13

La tartaruga

La tartaruga è un personaggio celebre... del linguaggio LOGO. Anche il BASIC del PC 128 permette di utilizzare una tartaruga.
E' possibile
- farla apparire in ogni punto dello schermo
- chiederle di andare avanti o indietro
- farle cambiare le proprie dimensioni o farla sparire.

Inoltre, è possibile darle una forma strana e fare sì che diventi un oggetto animato, veloce e sorprendente. Lo scopo della tartaruga è quello di lasciare dietro di sé una traccia, creando così dei disegni particolarmente complessi.

Visualizzazione della tartaruga: TURTLE, SHOW, FWD

All'inizio, la tartaruga è rappresentata semplicemente da un triangolo isoscele. Chiamiamo la prima tartaruga in modalità diretta con l'istruzione TURTLE:

```
TURTLE 0
```

Ora la tartaruga è pronta a spostarsi, a disegnare, ma rimane invisibile. Per visualizzarla utilizzare l'istruzione

```
SHOW t
```

Il triangolo appare al centro dello schermo, posizione di partenza standard per le tartarughe. Se si cambia SHOW 1 con SHOW 0, la tartaruga torna ad essere invisibile.

L'istruzione FWD (che sta per FORWARD, cioè avanti), seguita da un numero fa avanzare la tartaruga dello spazio indicato. Se il numero precisato è negativo, la tartaruga indietreggia. Lo spostamento avviene verso destra, direzione del naso della tartaruga ed è anche piuttosto rapido.

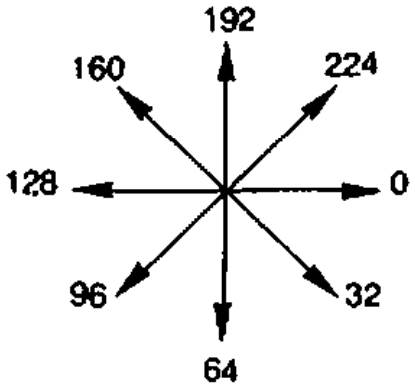
Come disegnare con la tartaruga: TRACE, HEAD

Mentre si sposta, la tartaruga può produrre una traccia che rende visibile la sua traiettoria. Il comando corrispondente è TRACE 1 (con TRACE 0 la traiettoria diventa invisibile). Ecco un comando che chiama la tartaruga, la fa apparire, fa visualizzare la sua traiettoria e la fa avanzare di 50 punti:

```
TURTLE 0:SHOW 1:TRACE 1:FWD 50
```

La tartaruga appare al centro dello schermo. Ricordare che il punto di tracciamento è posto al centro della tartaruga. Sostituendo TRACE 1 con TRACE 0, la tartaruga avanza senza disegnare.

La direzione del movimento può essere modificata con l'istruzione HEAD (testa) seguita da un numero che precisa di quanto bisogna girare: un giro completo (360 gradi) corrisponde al numero 256, metà giro (180 gradi) a 128, un quarto di giro (90 gradi) a 64, un ottavo di giro (45 gradi) a 32, e così via.



Quindi, HEAD 64 significa: prendere la direzione a 90 gradi verso destra e HEAD –64, prendere la direzione a 90 gradi verso sinistra.

Nell'esempio precedente, si può vedere che la tartaruga è apparsa in mezzo allo schermo. E' possibile spostare il punto di partenza della tartaruga, precisando le coordinate del punto di partenza desiderato nell'istruzione TURTLE dopo il numero della tartaruga:

TURTLE 0,160,20

Come fare ruotare su se stessa la tartaruga: ROT

Mentre si sposta, la tartaruga mantiene la testa nella direzione di destra, ma non il suo orientamento. L'istruzione HEAD modifica la direzione ma non l'orientamento. E' l'istruzione ROT che fa girare la tartaruga attorno al proprio centro. La suddetta istruzione è seguita da un numero che precisa di quanto debba girare la tartaruga rispetto alla sua posizione.

Quindi, affinchè la tartaruga segua l'orientamento della sua traiettoria, basta far seguire ogni istruzione HEAD da un'istruzione ROT di uguale valore.

Effetto ZOOM

Le proporzioni di ogni tartaruga possono variare continuamente su scala da 0 a 255:

– 16 corrisponde alla scala 1, cioè alla dimensione standard

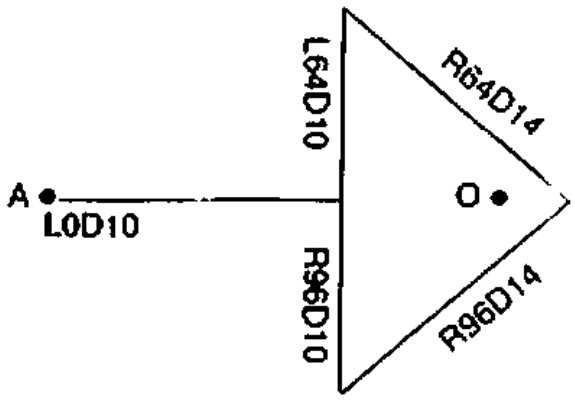
- 32 corrisponde alla scala 2
- 48 corrisponde alla scala 3
- ...
- 255 corrisponde alla scala 16

L'istruzione ZOOM, seguita da un numero, fissa le dimensioni della tartaruga.

Una nuova tartaruga

Creiamo una seconda tartaruga di forma diversa, che verrà definita nell'istruzione TURTLE da una stringa di caratteri indicanti la sequenza dei segmenti necessari per disegnarla. Al fine di distinguerla dalla prima, le si attribuisce un numero diverso. Questo numero viene indicato subito dopo TURTLE. Esempio:

TURTLE 2,,"L0D10L64D10R96D14R64D14R96D10":SHOW 1,1



La stringa che definisce una tartaruga è formata da una serie di coppie. Ogni coppia comprende due elementi: il primo indica di quanto bisogna girare, il secondo di quanto bisogna avanzare. I due elementi sono obbligatori.
L0D10 significa:
- gira a sinistra di 0
- avanza disegnando di 10.

R96D14 significa:
- gira a destra di 96
- avanza disegnando di 14.

Se si vuole far girare questa nuova tartaruga, la rotazione avverrà intorno al punto di partenza A del tracciato. Con il triangolo della tartaruga standard, la rotazione avveniva attorno al centro del triangolo. Per far ruotare su se stessa la nuova tartaruga, attorno ad un attro punto O più centrale, si aggiunge la coppia L128U15 che significa: – cambia l'orientamento di partenza da 0 in 128 – avanza senza disegnare di 15.

TURTLE 2,,"L128U15L128D10L64D10R96R64D14R96D10"

Con questa istruzione la tartaruga ruota attorno al punto O. Il punto di partenza del tracciato di una tartaruga è anche la posizione della sua "matita".

Capitolo 14

Musica

PLAY

L'istruzione PLAY permette di produrre note musicali. La successione delle note da suonare deve essere data prima di PLAY sotto forma di una stringa di caratteri. Questa stringa può essere una costante (con le note poste tra virgolette) o una variabile definita precedentemente. Le note vanno scritte senza separazioni e possono essere scelte tra:

DO RE MI FA SO LA SI

Va notato che tutte queste note sono scritte con due lettere, anche SOL che è stata abbreviata. Per esempio, per suonare le prime tre note della scala, si potrà battere:

PLAY"DOREMI"

oppure:

A$="DOREMI":PLAY A$

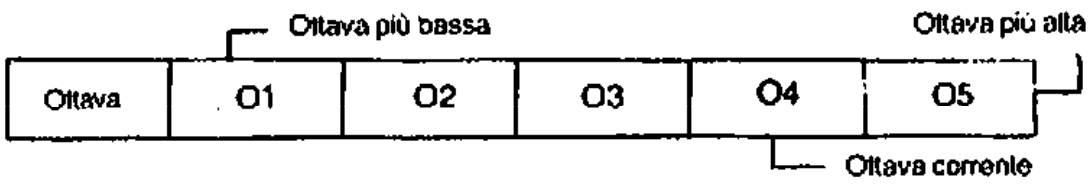
Pausa

Per produrre una pausa, cioè una nota che non si sente, si usa la lettera P. Per esempio, per suonare le note Do, Re, Mi separate da una pausa battere:

PLAY"DOPREPMIP"

Ottave

Le note possono essere suonate su cinque ottave diverse, cioè a cinque altezze diverse. L'ottava più bassa è codificata con 1 e la più alta con 5. Se non viene fatta alcuna precisazione, l'ottava è 4. Per scegliere l'ottava, si alterna nella stringa la lettera O seguita dal numero.

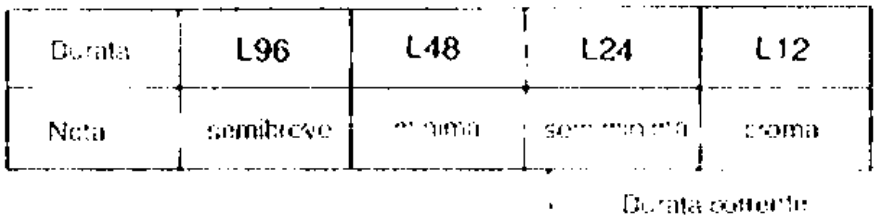


Per esempio, per suonare la nota LA sulle cinque ottave permesse, battere:

PLAY"O1LAO2LAO3LAO4LAO5LA"

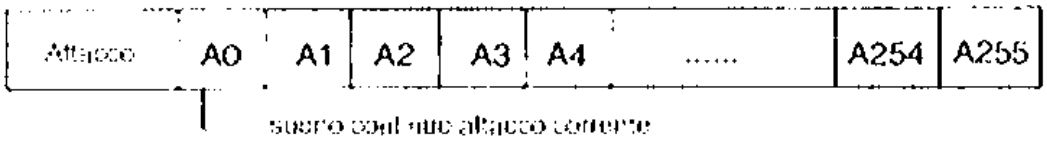
Durata

Le note possono essere suonate con durate diverse. La durata più breve è codificata con 1 e la più lunga con 96. Se non vi sono precisazioni, la durata è 24. Questa durata corrisponde a quella che i musicisti chiamano semiminima. La durata doppia (48) corrisponde quindi ad una minima. Per scegliere la durata si alterna nella stringa la lettera L seguita dal numero corrispondente alla durata prescelta.



Attacco

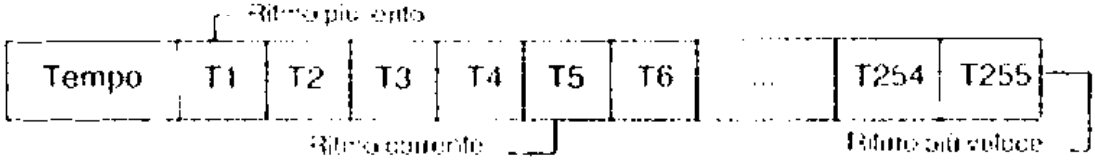
La nota può essere suonata in tutta la sua durata (suono continuo) o solo all'inizio (attacco). L'attacco è codificato da un numero compreso fra 0 e 255. Se non vi sono precisazioni, l'attacco è 0 e corrisponde ad un suono continuo. Per scegliere l'attacco, si alterna nella stringa la lettera A seguita dal numero corrispondente all'attacco prescelto.



Attenzione: è inutile scegliere attacchi oltre 100 perchè non sarà possibile sentire la nota suonata.

Tempo

Il tempo permette di regolare la velocità di esecuzione di una serie di note. Il tempo è fissato da un numero compreso fra 1 (più veloce) e 255 (più lento). Se non si precisa nulla, il valore utilizzato sarà 5. Per scegliere il tempo, si alterna nella stringa la lettera T seguita dal numero corrispondente al ritmo prescelto.



Diesis e bemolle

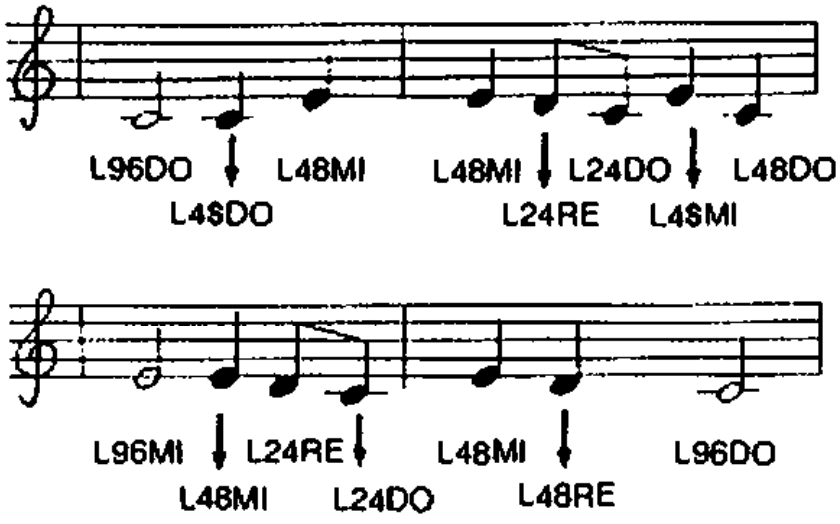
Per ottenere tutte le note di una gamma, si deve utilizzare il carattere# per diesis e il carattere b per bemolle, subito dopo il nome della nota. Con il PC 128 non utilizzeremo il bemolle: una nota seguita da un bemolle verrà rappresentata dalla precedente seguita da un diesis.

Claire fontaine

Un dettaglio importante: prima di comunicare al computer quale nota deve essere suonata, è necessario impostarne la durata. Mettersi nei panni della macchina per capire che bisogna assolutamente conoscere la durata prima della nota. Ecco qualche esempio di traduzioni:

Solfeggio				
Italiano	Semiminima-Sol	Minima-La	Croma-M	Semibreve-M
BASIC	L24SO	L48LA	L12MI	L96MI

Ecco una partitura semplificata del ritornello e la traduzione BASIC di ognuna delle sue note.



Quando un'ottava, una durata, un attacco, un tempo sono stati precisati, rimangono impostati fino ad indicazione contraria.

```
M1$="L96DO L48DOMI"
M2$="MIL24REDOL48MIDO"
M3$="L96MIL48MIL24REDO"
M4$="L48MIREL96DO"
PLAY M1$+M2$+M3$+M4$
```

La prima stringa definita (M1$) contiene le tre note della prima misura. La prima è una minima (L96), le due successive sono semiminime (L48, cioè metà di una minima). Noterete che la durata della terza nota (MI) non è precisata. Resterà quindi valida per quest'ultima, come per la prima nota della seconda misura (variabile M2$), la durata precedentemente impostata (48).

Capitolo 15

Primi cicli

Finora, al fine di mantenere vivo l'entusiasmo del lettore, si è badato a non richiedergli ta stesura di serie di istruzioni troppo impegnative. Ovviamente i risultati ottenuti sul video non sono spettacolari. E' arrivato il momento di dimostrare come, con poco, grazie al linguaggio BASIC, sia possibite visualizzare un'ampia serie di caratteri. Questo è possibile grazie ai cicli.

```
FOR K=1 TO 5:PRINT"PAROLA":NEXT K
PAROLA
PAROLA
PAROLA
PAROLA
PAROLA
OK
```

Le novità presenti in questa riga di istruzioni sono: FOR, TO e NEXT. Queste tre parole permettono la costruzione di cicli, permettono cioè le iterazioni. Su un'interazione i fattori importanti sono:

- ciò che bisogna ripetere
- quante volte lo si vuole ripetere.

Prima di descrivere le regole precise che devono essere osservate per la stesura dei cicli, ecco alcuni esempi; si invita il lettore a prepararne di propri, a provarli, a modificarli.

```
FOR K=1 TO 100:PRINT"A";:NEXT K
;\AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAA
OK
```

```
FOR N=0 TO 50:PRINT N::NEXT N
0  1  2  3  4  5  6  7  8  9  10  11
12  13  14  15  16  17  18  19  20  21
22  23  24  25  26  27  28  29  30  31
32  33  34  35  36  37  38  39  40  41
42  43  44  45  46  47  48  49  50
OK
```

```
FOR U=-10 TO 10:PRINT U^2;:NEXT U
-20 -18 -16 -14 -12 -10 -8 -6 -4 -2 0
2 4 6 8 10 12 14 16 18 20

D=25:F=50
FOR J=D TO F:PRINT J;:NEXT J
25 26 27 28 29 30 31 32 33 34
35 36 37 38 39 40 41 42 43 44
45 46 47 48 49 50
OK

FOR M=1 TO 5:PRINT M;"KM":NEXT M
1 KM
2 KM
3 KM
4 KM
5 KM
OK
```

Bisogna ora cercare di visualizzare delle strisce orizzontali di colore. Una striscia si otterrà inviando in stampa una serie di "spazi" il cui colore di sfondo venga modificato da un'istruzione COLOR. Se si vogliono visualizzare tutti i colori, la stessa azione dovrà essere ripetuta sedici volte.

Per esempio, per ottenere una striscia rossa:

```
COLOR,1:PRINT"
```

In questo caso l'istruzione COLOR modifica solo lo sfondo lasciando inalterato il colore dei caratteri; l'istruzione PRINT visualizza in seguito dodici spazi di colore rosso.

Esempio per visualizzare più colori:

```
FOR C=0 TO 15:COLOR,C:PRINT"        ":NEXT C
```

Con la tartaruga

Osservando la formazione di un disegno su video con l'aiuto della tartaruga si può comprendere come l'aspetto ripetitivo dei cicli determini la scelta di disegnare figure geometriche regolari, per esempio un ottagono. Questo sarà formato da segmenti di uguale lunghezza, ciascuno inclinato rispetto al precedente di 45 gradi.

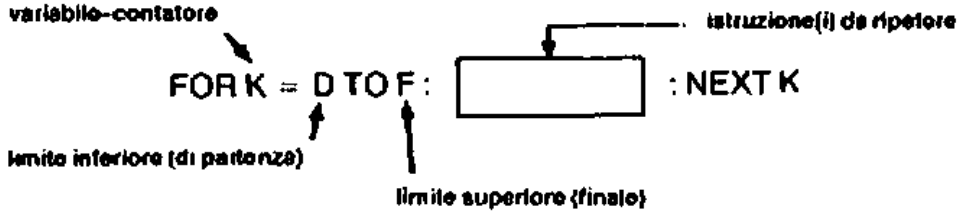
```
TURTLE 0:TRACE 1:FOR N=1 TO 8:FWD 30:HEAD 32:NEXT N
```

Sostituendo HEAD 32 con HEAD 16 ed il vatore massimo di N con 16, si ottiene un poligono di 16 lati; in questo caso però, una parte della figura rimarrà al di fuori del video. Bisogna dunque spostare il punto di partenza della tartaruga indicando le coordinate nell'istruzione TURTLE. Per esempio:

```
TURTLE 0,150,20
```

Precisazione

Un ciclo comincia con la parola FOR seguita dal nome di una variabile numerica, e termina con NEXT seguito dallo stesso nome di variabile. Fra queste due istruzioni bisogna scrivere la o le istruzioni da ripetere. Il limite inferiore e quello superiore sono due valori numerici che delimitano il TO.



Per l'esecuzione di questo ciclo, il calcolatore procede nel seguente modo:
1 – Confronta i contenuti di K e di F. Se K è inferiore a F, va al punto 2, altrimenti termina.
2 – come 1.
3 – come 2.
4 – Incrementa il contenuto di K di una unità (K=K+1). Va al punto 1.

Cambio di passo

Nei cicli precedenti il passo è 1: la variabile del contatore viene aumentata regolarmente di 1. E' però possibile aumentare il pesso, prevedendo dei cicli con passo 2, 10, 0,4. Poichè in inglese passo si dice STEP, si avranno le seguenti istruzioni:

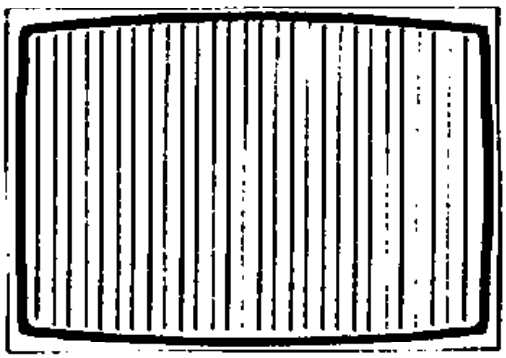
```
FOR K=0 TO 10 STEP 2:PRINT K;:NEXT K
2  4  6  8  10
OK

FOR I=4 TO 5 STEP 0.2:PRINT I;:NEXT I
4  4.2  4.4  4.6  4.8  5
OK

FOR J=5 TO 10 STEP 20:PRINT J:NEXT J
5
OK

FOR S =10 TO 0 STEP –1:PRINT S;:NEXT S
10  9  8  7  6  5  4  3  2  1  0
OK
```

Capitolo 16

Ancora sui cicli

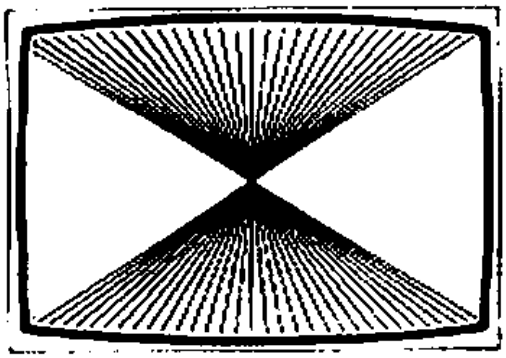
Video pieno di linee

FOR C=0 TO 320 STEP 10:LINE(C,0) – (C,199):NEXT C



Cambiando l'istruzione da iterare, si ottengono motivi differenti.

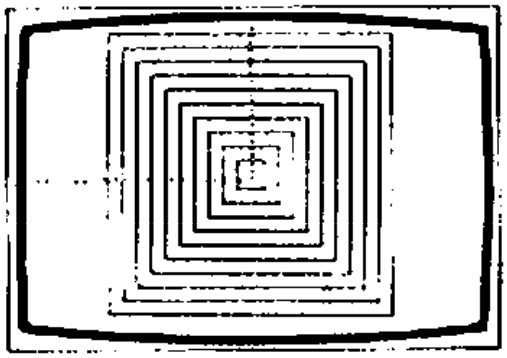
LINE(C,0) – (319–C,199)



BOX(160–C,100–C) – (160+C,100+C)

160

100



Attenzione: sia per i passi sia per i limiti, è possibile utilizzare nomi di variabili numeriche.

```
A=1:B=2
FOR K=B - A TO B+A STEP A:PRINT K:NEXT K
1
2
3
OK
```

Isolamento delle lettere di una parola

```
PAR$="BASIC":L=LEN(PAR$)
FOR K=1 TO L:PRINT MID$(PAR$,K,1):NEXT K

B
A
S
I
C
OK
```

Attenzione: MID$(PAR$,K,1) è in effetti la k–esima lettera di PAR$.

Piramide

```
PAR$="BASIC":L=LEN(PAR$)
FOR K=1 TO L:PRINT LEFT$(PAR$,K):NEXT K
B
BA
BAS
BASI
BASIC
OK
```

Pausa

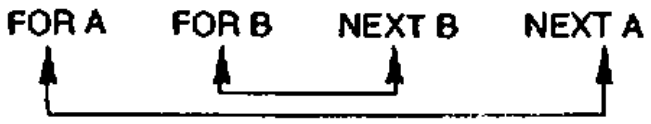
```
FOR K=1 TO 1000:NEXT K
OK
```

In realtà in questo ciclo non si fa nulla se non contare da 1 a 1000. L'utente quindi non vedrà nulla sul video, ma potrà verificare il tempo impiegato dal calcolatore per lo svolgimento di questa attività. Il tempo dipende ovviamente dalla differenza tra i limiti minimo e massimo.

Si sconsiglia:
– Un passo zero.
– La mancanza di coerenza tra limiti e passo. Se D è il limite inferiore, F il limite superiore e P il passo, si dovrà avere:
– se D < F allora P > 0
– se D > F allora P < 0
In tutti gli altri casi, il ciclo non viene svolto e si passa all'istruzione successiva a NEXT.
– Non si deve cambiare la variabile contatore nelle istruzioni da iterare. Per esempio, se questa variabile è K, tutte le istruzioni del tipo K=K+1 sono da evitare.

Cicli nidificati

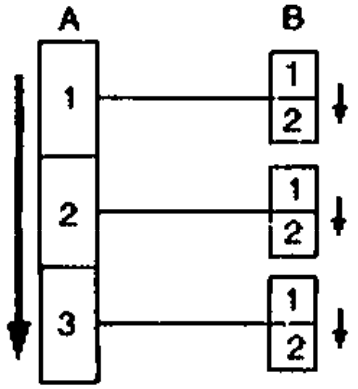
E' possibile prevedere due cicli, uno all'interno dell'altro, come per le scatole cinesi. La serie di istruzioni prevede allora due FOR, due NEXT e due contatori.



Questo schema riassume abbastanza bene il solo punto sul quale è necessario insistere: il programmatore deve assolutamente rispettare l'ordine di incastro dei cicli. L'errore più frequente si ha nella permutazione dell'ordine di NEXT B e NEXT A.
Nell'approccio diretto nel quale si sta lavorando, i cicli nidificati comportano delle righe di istruzioni piuttosto lunghe. Di questo non c'è da preoccuparsi. Non bisogna però dimenticare di premere il tasto ENTER solo dopo l'ultima istruzione, anche se sul video l'insieme della stessa occupa più righe.

```
FOR A=1 TO 3:FOR B=1 TO 2:PRINT"M";:
NEXT B:NEXT A
MMMMMM
OK
```

Si ottengono così sei lettere M allineate. Il ciclo B viene svolto tre volte all'interno del ciclo A. Questo significa che si stampa due volte la lettera M, per tre volte. Lo sviluppo dei contatori A e B può essere rappresentato da un albero:



Tavole di moltiplicazione

```
FOR A=1 TO 9:FOR B=1 TO 9:PRINT A*B;:
NEXT B:PRINT:NEXT A
1  2  3  4  5  6  7  8  9
2  4  6  8  10  12  14  16  18
3  6  9  12  15  18  21  24  27
4  8  12  16  20  24  28  32  36
5  10  15  20  25  30  35  40  45
6  12  18  24  30  36  42  48  54
7  14  21  28  35  42  49  56  65
8  16  24  32  40  48  56  64  72
9  18  27  36  45  54  63  72  81
OK
```

Attenzione: bisogna cominciare ad esaminare il ciclo B. Quando è terminato, l'istruzione PRINT che segue NEXT B determina l'allineamento a nuova riga. Poi si riprende il ciclo B. Questa serie di istruzioni stampa nove volte (ciclo A) una serie di nove numeri (ciclo B).
In questa tavola l'allineamento non è perfetto a causa della presenza di numeri di 1 o 2 cifre.

Capitolo 17

Casualità

Sia per gioco che per applicazioni più serie, è importante che il PC 128 possa effettuare delle manipolazioni casuali. Questa è un'applicazione per la quale un microelaboratore sembra inadeguato. Essendo ormai abituati ad un elaboratore che non lascia niente al caso, bisogna attendersi che la casualità prodotta sia particolare, una sorta di falsa casualità, una simulazione.
Ed è proprio di questo che si tratta: di una casualità calcolata, ma così ben calcolata da sembrare una casualità reale, imprevedibile. La funzione RND (dall'inglese random) fornisce un numero decimale aleatorio compreso fra 0 e 1.

```
PRINT RND:PRINT RND
.594972
.512679
OK
```

Questi numeri sono calcolati dal BASIC e si ripartiscono uniformemente fra 0 e 1.

Applicazioni

L'istruzione RND fornisce un numero decimale compreso fra 0 e 1. Grazie ad alcune semplici manipolazioni aritmetiche, si possono ricavare numeri di tipo diverso con intervalli diversi.

Decimali

Fra 0 e X : RND*X
Fra A e B : A+RND*(B–A)

Interi

Fra 0 e N : INT(RND*(N+1))
Fra P e Q : P+INT(RND*(Q–P+1))

Attenzione: la funzione INT appena introdotta fornisce l'intero più alto inferiore al numero indicato fra parentesi.

```
PRINT INT(256/7)
36
OK
```

dato che 256/7=36,571428...

Esempi

```
X=INT(RND*2)
PRINT"TESTA:1.CROCE:0":PRINT X
TESTA:1.CROCE:0
1
OK
```

E' lesla!

```
X=1+INT(RND*6)
PRINT"FACCIA DEL DADO:";X
FACCIA DEL DADO:4
OK
```

```
X=1+INT(RND*49)
PRINT"TOMBOLA VINCENTE:":X
TOMBOLA VINCENTE:22
OK
```

Cielo steilato

```
SCREEN 3,0,0:FOR K=1 TO 200:X= INT(RND*320):
Y=INT(RND*200):PSET(X,Y):NEXT K
```

Attenzione: questo comando illumina 200 punti te cui coordinate sono scelte casualmente all'interno di un rettangolo di 320 per 200, pari all'intero video.

Passeggiata di una particeiia

```
C=160:L=100:PSET(C,L):K=30:
FOR I=1 TO 1000:DC=K*(RND-.5):
DL=K*(RND-.5):C=C+DC:L=L+DL:LINE-(C,L):
NEXT I
```

Attenzione, confermare la riga di istruzione solo dopo NEXT I.

In esecuzione, si vedrà la traccia di una particella che si muove in tutte le direzioni.

Sfortunatamente, la particella tende ad uscire dal video, a ritornarvi e ad uscirne di nuovo. E' possibile comunque utilizzare le istruzioni di spostamento di una tartaruga (vedere capitolo 13). Soto l'angolo è scelto casualmente fra 0 e 256. Ogni spostamento elementare ha una di lunghezza 20.

```
CLS:TURTLE 0,160,100:TRACE1:FOR I=1 TO 1000:
A=1+INT(RND*256):HEAD A:FWD 20:NEXT I
```

Una brutta sorpresa

Esperimento: fare stampare qualche numero ricavato casualmente con la funzione RND e prenderne nota, poi spegnere il PC 128.

Ricominciando la stessa operazione, si avrà la brutta sorpresa di constatare che i numeri che verranno proposti saranno gli stessi precedentemente proposti e annotati.

In effetti, come già detto in precedenza, questi numeri aleatori vengono calcolati partendo dalla medesima formula che dà sempre i medesimi risultati. Ma poichè ogni RND viene riutilizzato nella formula per il calcolo del successivo, la serie prodotta è sempre la stessa. Per ottenere delle serie differenti, è sufficiente inizializzare la serie partendo da numeri diversi.

Nel capitolo 29 si imparerà a simulare, con il PC 128, una reale casualità.

Capitolo 18

Introduzione alla programmazione

Esistono due modi di far uso del linguaggio BASIC. Il primo modo si chiama modalità diretta o modalità immediata ed è quella che noi abbiamo utilizzato fino a questo punto. Gli ordini che noi davamo al PC 128 in questa maniera si chiamano comandi e cioè istruzioni che vengono eseguite subito dopo aver premuto il tasto ENTER.
Abbiamo molto insistito sulla possibilità offerta in modalità diretta di memorizzare dei dati (di tipo numerico o a stringa di caratteri) nella memoria RAM del PC 128 e di poterli recuperare in qualsiasi momento. Tutte queste operazioni ci hanno permesso di introdurre e di mettere in pratica i principali comandi del linguaggio BASIC.
Ora passeremo ad esaminare una fase successiva: il programma. Il principio di funzionamento di un programma si basa sulla possibilità offerta dal sistema di memorizzare i comandi per una esecuzione differita.
La memorizzazione dei comandi pone considerevoli problemi tecnici che non è il caso di ricordare qui. Al programmatore basta conoscere il processo che realizza questa funzione. Nel BASIC è molto semplice.
Per memorizzare un comando, basta farlo precedere da un numero intero. Il comando così memorizzato viene definito riga di istruzione e il numero che lo precede è il numero di questa riga.

Primo esempio di programma

1 PRINT 1917 [ENT]

L'unica conclusione che si può trarre da questo primo esempio è che se un comando è preceduto da un numero, l'esecuzione non ha luogo. Il numero 1917 non è stato stampato, l'OK non compare, la macchina aspetta qualcosa.

Bisogna anche osservare che tutti i comandi esaminati nei moduli precedenti cominciano con una lettera. Da questa prima esperienza potrebbero sorgere i seguenti quesiti:
- Come eseguire questa riga d'istruzione?
- Come modificarla?
- Come cancellarla?
- Come memorizzare altre istruzioni?

Le risposte a queste domande richiamano termini del linguaggio BASIC, comandi particolari che bisogna conoscere bene prima di eseguire il primo programma.

Non faremo al lettore il torto di eseguire davanti a lui un programma che comporti una sola istruzione. Ecco un ciclo ripartito su tre istruzioni. Sarebbe stato possibile memorizzarlo in una sola riga, ma è interessante vedere come la macchina gestisca più righe di programma. Queste tre righe, nel nostro programma, sono numerate 1, 2 e 3.

```
1 FOR I=50 TO 55
2 PRINT I;
3 NEXT I
```

Esecuzione
Per eseguire questo ciclo, bisogna digitare RUN (modalità diretta).

```
RUN
50  51  52  53  54  55
OK
```

RUN è dunque il comando di "lancio" del programma. E' qui che bisogna cogliere la differenza fondamentale fra la modalità diretta e la modalità programma. Essendo ogni riga di istruzione numerata e memorizzata ogni comando RUN lancia l'esecuzione del programma tutte te volte che l'utente lo desidera.

```
RUN
50  51  52  53  54  55
OK
RUN
50  51  52  53  54  55
OK
```

Visualizzazione del programma

Se si ripete questa operazione per un certo numero di volte, è possibile che per effetto dello scorrimento dell'immagine (*scrolling*), tutte le istruzioni del programma scompaiano dallo schermo. Tuttavia esse non sono scomparse dalla memoria. Il comando LIST permette di richiamarle.

```
LIST
1 FOR I=50 TO 55
2 PRINT I;
3 NEXT I
```

Particolari formati del comando LIST permettono di consultare determinate istruzioni.
– LIST: visualizza tutte le righe d'istruzione del programma.
– LIST I–F: visualizza le righe d'istruzione i cui numeri vanno da I a F.
– LIST I–: visualizza tutte le righe d'istruzione a partire dalla riga I.

Numerazione

Il programmatore può decidere di introdurre le sue istruzioni nell'ordine che desdera, ma solo l'ordine dei numeri sarà preso in considerazione nella memoria. Il comando LIST visualizza te righe di programma in ordine numerico.

```
2 A=578
1 B=609
LIST
1 B=609
2 A=578
```

Capitolo 19

Consigli preliminari

Numerazione

L'annotazione precedente concernente l'ordine delle istruzioni mostra che è possibile arricchire il programma in qualsiasi momento per mezzo di nuove righe. A questo proposito i programmatori hanno preso l'abitudine di numerare le righe di t0 in 10 al fine di inserire facilmente e in ogni luogo le nuove righe di programma. Il programmatore ha libera scelta. Uniche restrizioni:
- I numeri di riga devono essere numeri interi.
- Essi non devono essere maggiori di 63999.

Un programma può cominciare dalla riga 3142 e questa sarà la prima ad essere eseguita dopo RUN.

Modifica

Come si apprenderà programmando, la modifica occupa una grande parte del lavoro di programmazione. Il caso più semplice riguarda un errore di sintassi in una riga di programma. Se il programma contiene una riga di questo tipo:

640 PRIT"BUONGIORNO"

è probabile che il comando RUN sia seguito da un messaggio d'errore. L'interprete BASIC vi segnala il numero della riga in cui l'errore è stato individuato.

Syntax Error in 640
(Errore di sintassi riga 640)

Per correggere questo errore digitare il comando LIST seguito da un punto.

LIST.
640 PRIT ■ "BUONGIORNO"

Non si potrà non notare allora la dimenticanza della N di PRINT.

E' ancora più facile che il quadratino pieno indichi che l'errore è situato nell'istruzione che lo precede. Si porta allora il cursore nel punto richiesto (sulla T di PRIT) e si preme INS seguito da N. Si convalida questa riga con ENTER e l'errore è corretto. Bisogna segnalare che gli errori sono individuati uno per uno e che è solo il primo errore incontrato ad essere segnalato.

Cancellazione

Per cancellare una riga di programma, basta digitare il numero di questa riga seguito da ENTER. Questa nuova riga di istruzione vuota si sovrappone alla precedente, cancellandola. Il comando LIST non la visualizzerà più. Si può anche utilizzare il comando DELETE (cancellare), utile soprattutto per cancellare diverse righe. Esso segue gli stessi principi di LIST.

– DELETE I–F: cancella tutte le righe di istruzione i cui numeri vanno da I a F.

– DELETE I: cancella tutte le righe del programma a partire dalla riga I. Per cancellare tutto un programma, è sufficiente digitare DELETE 0–. Naturalmente DELETE deve essere utilizzato con molta più attenzione di LIST poichè una riga di programma cancellata con questo comando non è più recuperabile.

CLEAR

CLEAR è un'istruzione che inizializza le variabili. 0 viene utilizzato per le variabili numeriche e nessun carattere per le stringhe di caratteri. CLEAR può allo stesso modo servire a riservare in memoria dello spazio per le variabili, a seconda del loro tipo. E' talvolta utile porre questa istruzione all'inizio del programma ma, la maggior parte delle volte, il comando RUN ha lo stesso effetto (inizializzazione delle variabili).

NEW

Il comando NEW cancella tutto il programma in memoria ed è molto utile durante gli esercizi di programmazione, nel corso dei quali si cambia spesso progetto e quindi programma. Il PC 128 deve essere informato del cambio di attività al fine di non mescolare le righe dei diversi programmi. Quando si comincia un nuovo programma, come prima istruzione utilizzare dunque NEW.
Il BASIC del PC 128 accetta l'istruzione NEW in una riga di programma. Quando l'interprete passa sopra questa riga, cancella tutto. Questa istruzione viene utilizzata soprattutto nei programmi auto–distruttivi.

Interruzione

Può presentarsi la necessità di interrompere un programma in esecuzione, ad esempio parchè il programma non è interessante, non dà i risultati attesi o, peggio, si sa che non si fermerà mai da solo e che un intervento esterno è necessario.

Prendiamo l'esempio di un programma particolarmente noioso perchè si accinge a scrivere 10.000 volte NOIA.

```
10 FOR X=1 TO 10000
20 PRINT"NOIA"
30 NEXT X
```

Interruzione definitiva

Per interrompere il programma, premere il tasto di reinizializzazione RESET.

Selezionare il BASIC 128 nella pagina iniziale, oppure premere contemporaneamente i tasti CTRL e C. Quest'ultimo metodo è più consigliabile poichè CTRL–C interrompe lo svolgimento del programma ma, contrariamente al tasto RESET, lascia intatto il contenuto delle variabili.

Nell'esempio precedente, dopo aver lasciato girare il programma per qualche secondo, lo si interrompe con CTRL–C. Eseguendo successivamente il comando PRINT X, è possibile sapere quante volte il vocabolo NOIA è stato stampato. Quando un programma viene interrotto da CTRL–C, sul video appare il messsggio "Break in" (interruzione) seguito dal numero di riga in cui è avvenuta l'interruzione.

Interruzione provvisoria

Il tasto STOP interrompe provvisoriamente lo svolgimento del programma. Nessun messaggio particolare appare sullo schermo. Tutto resta in sospeso. Premendo un tasto qualsiasi si riavvia il programma. Le interruzioni di questo tipo possono essere ripetute. Si può così fermare più volte uno schermo che scorre troppo velocemente. L'istruzione STOP può essere utilizzata in un programma per interromperlo. Dopo il messaggio Break in, si può riprendere l'esecuzione digitando il comando CONT.

Capitolo 20

Salvataggio di un programma

Le istruzioni di un programma, come il contenuto delle variabili, sono registrate nella memoria RAM, una memoria volatile che si cancalla in assenza di alimentazione elettrica. Bisogna dunque provvedere a una memorizzazione esterna permanente di tipo magnetico.
Il PC 128 dispone di un lettore–registratore di programmi in grado di gestire delle cassette magnetiche.
Consigli per l'uso delle cassette:
– Riavvolgere sempre le cassette e rimettere a zero il contatore. Ciò permetterà di individuare i numeri di inizio e di fine registrazione.
– Attenzione: la parte iniziale del nastro è assolutamente inutilizzabile ai fini della registrazione.

Per registrare ("salvare") un programma

Si consiglia di effettuare una prova, che potrebbe essere la seguente:

```
10 PRINT "PROVA DI MEMORIZZAZIONE"
```

– Scegliere un nome da dare al programma, per esempio PROVA.
– Premere contemporaneamente i tasti [●] e [►] del registratore.
– Digitare sulla tastiera del PC 128:

```
SAVE"PROVA"
```

Il registratore fa avanzare il nastro della cassetta e il programma viene registrato. Quando il nastro si ferma, sul video appare il messaggio OK.
SAVE"PROVA" copia il contenuto della memoria RAM su nastro magnetico con il nome PROVA. Questo nome servirà a richiamare il programma.

```
SAVE"PROVA"
OK
```

Per caricare ("leggere") un programma

LOAD

Un programma viene registrato per poterlo richiamare successivamente. Per trasferire il contenuto della cassetta nella memoria RAM, digitare l'istruzione LOAD"PROVA". Questo nel caso in cui il programma sia stato salvato con SAVE"PROVA".

Attenzione: è compito dell'utente posizionare correttamente il nastro davanti alla testina di lettura. Da qui la necessità di annotare il numero di posizionamento all'atto della registrazione. Il registratore fa avanzare il nastro della cassetta e il PC 128 visualizza

SEARCHING

Se altri programmi sono presenti prima di PROVA, non vengono letti, ma il PC 128 segnala la loro presenza visualizzando, per esempio:

Skip: COMPITO .BAS

(*Skip* significa "salto" e BAS è l'abbreviazione di BASIC). Quando la testina di lettura riconosce il programma PROVA, sul video appare:

Found: PROVA .BAS

Il programma viene caricato in memoria. Quando il nastro si ferma, sul video appare OK. A questo punto è possibile verificare la presenza di PROVA in memoria, dando il comando LIST o RUN.

```
LOAD"PROVA
Skip COMPITO .BAS
Found PROVA .BAS
OK
_
```

Problema di caricamento

– Se in fase di lettura o di caricamento appare il messaggio

Device I/O Error
OK
(Errore I/O dispositivo)

bisogna riavvolgere il nastro all'inizio della registrazione, per tentare di nuovo il caricamento.

MOTORON/MOTOROFF

Il BASIC mette a disposizione due istruzioni per la gestione del registratore tramite PC 128.
– MOTORON che determina lo scorrimento del nastro, a condizione che il tasto ▶ sia premuto. Se il tasto di registrazione è a sua volta premuto, il nastro viene cancellato. Durante lo scorrimento, è possibile continuare ad impostare qualsiasi comando da tastiera.
– MOTOROFF che arresta lo scorrimento del nastro.

Attenzione: nel caso venga memorizzato un solo programma per cassetta, all'atto del caricamento non è più necessario indicare il nome del programma. E' sufficiente battere:

LOAD " [ENT]

In questo modo verrà caricato il primo programma presente sulla cassetta.

Nota: è possibile inserire dati in memoria tramite l'istruzione di assegnazione (=). Si possono memorizzare istruzioni facendole precedere da un numero di priorità. Si memorizzano programmi su cassetta tramite SAVE. Si possono memorizzare su cassetta anche dei dati: in questo caso avremo un archivio o file. Questo argomento verrà trattato nei capitoli 47 e 48.

Consigli per la gestione delle cassette

Si consiglia di essere estremamente metodici nella classificazione delle cassette per evitare errori quando cominciano ad essere numerose. Ecco alcune indicazioni alle quali fare eventualmente riferimento:

– Numerare o denominare le cassette su entrambi i lati. Attenzione: annotare il nome sulla cassetta (utilizzando un'etichetta autoadesiva) e non sul contenitore, che può essere intercambiabile.
– Di ogni cassetta (provvista di etichetta) annotare la successione delle registrazioni e il numero di partenza corrispondente rilevato dal contatore.
– Riavvolgere sempre le cassette prima di toglierle dal registratore.
– Eseguire sempre una copia di ogni cassetta.
– Riporre le cassette nei relativi contenitori.
– Con regolarità, recuperare le cassette contenenti registrazioni di lavoro di cui non ci si serve più e cancellarle.
– Per ulteriore sicurezza, registrare il programma due volte sulla stessa cassetta e tenerne una copia su un'altra cassetta da non utilizzare se non in caso di emergenza.

Capitolo 21

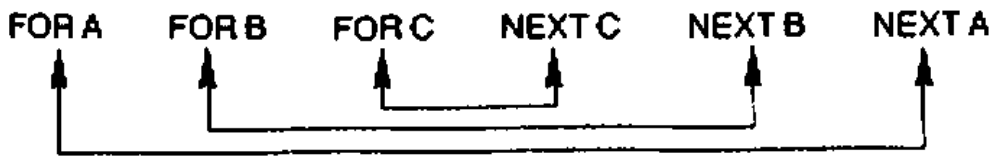
I primi programmi

Riprendiamo sotto forma di programmi gli esempi trattati in modalità diretta. La possibilità di scrivere su più righe permette di chiudere facilmente diversi cicli e di ottenere quindi dei risultati interessanti.

Esempio 1: la tartaruga disegna nuovamente un ottagono.

```
20 CLS
30 TURTLE 0
40 TRACE 1
50 FOR N=1 TO 8
60 FWD 30:HEAD 32
70 NEXT N
```

Esempio 2: è possibile chiudere tre, quattro o più cicli rispettando sempre scrupolosamente gli ordini di nidificazione.



Ricordiamo l'istruzione SCREEN
SCREEN X,Y,Z prepara un primo piano di colore X su uno schermo di colore Y con una cornice di colore Z. Per visualizzare più colori, si userà la seguente procedura:
1. Muovere il cursore su tutto lo schermo scrivendo qualsiasi cosa, anche senza senso.
2. Riportare il cursore in alto a sinistra con CLS HOME.
3. Introdurre il programma seguente premendo CTRL–X alla fine di ogni riga prima di ENTER per cancellare i caratteri che potrebbero essere ancora presenti sulla riga.

```
10 FOR X=0 TO 15
20 FOR Y=0 TO 15
30 FOR Z=0 TO 15
40 SCREEN X,Y,Z
50 NEXT Z,Y,X
60 SCREEN 4,6,6
```

4. Premere RUN. Il programma durerà abbastanza a lungo in quanto passa in rivista le 4.096 colorazioni diverse dello schermo. Di tanto in tanto si noterà che la scrittura sparisce quando il colore di primo piano e quello dello sfondo sono identici (X=Y).

La riga 60 è stata aggiunta per evitare che l'ultima combinazione di colori visualizzata sullo schermo dopo l'esecuzione del programma lasci sullo schermo una nebbia arancione.

Esempio 3: il PC 128 offre una gamma di 4.096 tinte diverse:

```
10 FOR X=4095 TO 0 STEP -1
20 FOR Y=1 TO 10:LOCATE 0,0:PRINT X
30 PALETTE 0,X
40 NEXT Y:NEXT X
```

L'istruzione PALETTE nella riga 30 assegna al colore 0 la tinta X. Dopo le 4.096 tinte riapparirà lo sfondo dello schermo, che all'inzio è nero.

Curve per esercizi matematici

Il tracciato delle curve costituisce una buona palestra per utilizzare l'istruzione DEFFN che permette all'utente di definire delle funzioni particolari.
Un buon lavoro di programmazione consiste nel creare un programma che visualizzi le curve scelte dall'utente in una finestra qualsiasi. E' anche un ottimo strumento di lavoro per gli studenti di matematica. Per tracciare la curva di

$$f(x) = \frac{2x^2+x-1}{x^2-x+1}$$

battere il seguente programma:

```
10 CLS
20 DEFFN S(X)=(2*X*X+X-1)/(X*X-X+1)
30 EX=50:TX=100:EY=25:TY=50
40 FOR X=-2 TO 2 STEP 1/EX
50 PSET (X*EX+TX,FNS(X)*EY+TY)
60 NEXT X
```

Il risultato ottenuto è il seguente:

In questo tipo di operazioni la difficoltà consiste nel calcolare i coefficienti moltiplicatori (EX e EY) in modo tale che la curva sia sufficientemente grande senza però uscire dallo schermo.

Le funzioni matematiche verranno riprese nel capitolo 34. Speriamo comunque di aver stimolato un certo interesse con questi primi esempi di programmi.

Attenzione: quello che viene visualizzato sullo schermo non è necessariamente memorizzato e viceversa. Spesso ci si dimentica di cancellare un'istruzione di cui si è modificato il numero. Data per esempio questa istruzione:

7 COLOR 1,2

se si vuole assegnare a questa istruzione il numero 9 (perchè deve seguire e non precedere l'istruzione 8), bisogna:
- portare il cursore sul numero 7 all'inizio della riga
- battere 9 e convalidare il comando premendo ENTER.

L'istruzione

9 COLOR 1,2

viene in questo modo memorizzata. L'istruzione 7 non è più visualizzata sullo schermo ma esiste sempre in memoria. Bisogna quindi cancellarla battendo 7 seguito da ENTER. Così non ci sarà più in memoria l'istruzione numero 7 ma una nuova istruzione 9.

Attenzione

Per memorizzare una riga di programma, è necessario premere il tasto ENTER. In caso contrario, il comando viene scritto sullo schermo ma non sarà mai eseguito. Si consiglia di non usare le frecce per spostare il cursore da una riga all'altra quando si scrive un programma, per evitare che questo provochi un errore di sintassi. Se non si riesce a scoprire l'errore segnalato da:

Syntax Error in 50
OK

bisogna assolutamente eliminare dalla memoria l'istruzione 50 e, se necessario, reintrodurre una nuova istruzione 50.

Commenti

Scrivere dei programmi, memorizzarli, riprenderli molto tempo dopo per migliorarli è l'attività classica del programmatore, dilettante o professionista. Per assicurare la manutenzione del software, e cioè la possibilità di intervenire nella scrittura di un programma dopo la sua realizzazione finale, è necessario che il programma sia documentato. Per i programmi brevi a diffusione limitata all'ambito familiare, si ammette che la documentazione venga scritta nelle istruzioni del programma stesso in modo che il listato del programma permetta a chiunque di capire la sua organizzazione in vista di eventuali interventi.
Un'istruzione BASIC permette di inserire dei commenti nei programmi. Si tratta dell'istruzione REM. Tutto ciò che segue l'istruzione REM in una riga di comando viene ignorato dall'interprete BASIC al momento dell'esecuzione, ma viene memorizzato. L'istruzione REM può essere posta all'inizio o all'interno di una riga di programma ma dopo l'istruzione REM non verrà eseguita alcuna istruzione che si trovi sulla stessa riga. Il BASIC ammette l'apostrofo (') come abbreviazione di REM.

Capitolo 22

L'arte di programmare

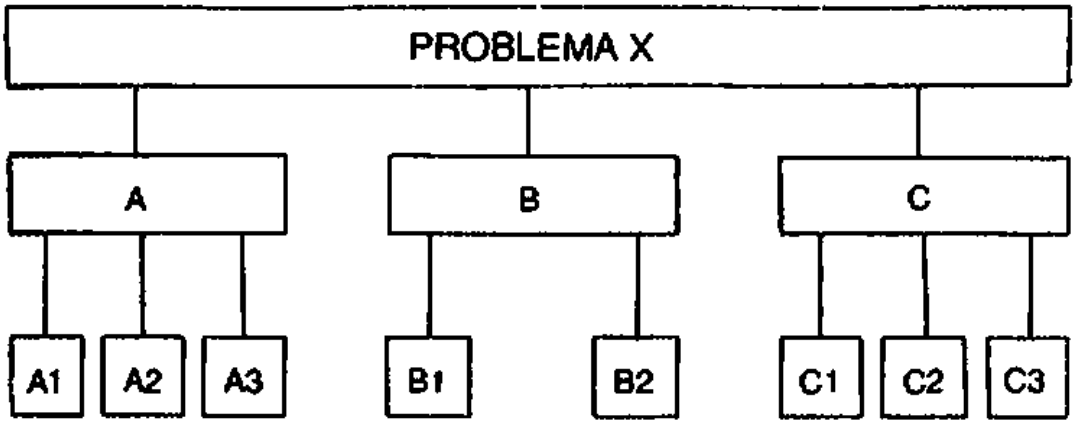
Il BASIC non è l'unico linguaggio di programmazione esistente. Imparare a programmare in BASIC, come del resto in qualsiasi altro linguaggio, dovrebbe prevedere due fasi distinte:
- Imparare a programmare.
- Imparare il BASIC.

Se si trascura la prima di queste due fasi, indubbiamente la più difficile ma anche la più importante, si rischia di diventare un finto programmatore.

L'analisi top–down

L'idea dell'analisi top–down è semplice ed efficace. Programmare significa risolvere un problema con un algoritmo, e cioè una serie di operazioni elementari perfettamente descritte nel loro contenuto e nella loro concatenazione. L'analisi top–down parte dalla definizione di un algoritmo rispettando i due principi seguenti:
- Scomporre il problema nelle sue fasi principali.
- Trattare ogni parte del problema allo stesso modo del problema generale.

Praticamente l'analisi top–down obbliga ad occuparsi sempre dell'essenziale rinviando a dopo quello che può essere considerato accessorio. Il termine anglossassone "top down analysis" significa appunto analisi dall'alto in basso. Nello schema seguente il problema X è stato scomposto nei tre compiti A, B, C, a loro volta suddivisi in alcuni sottocompiti. Si può procedere ad un'analisi sempre più spinta verso il basso fino a prendere in considerazione tutti i dettagli.



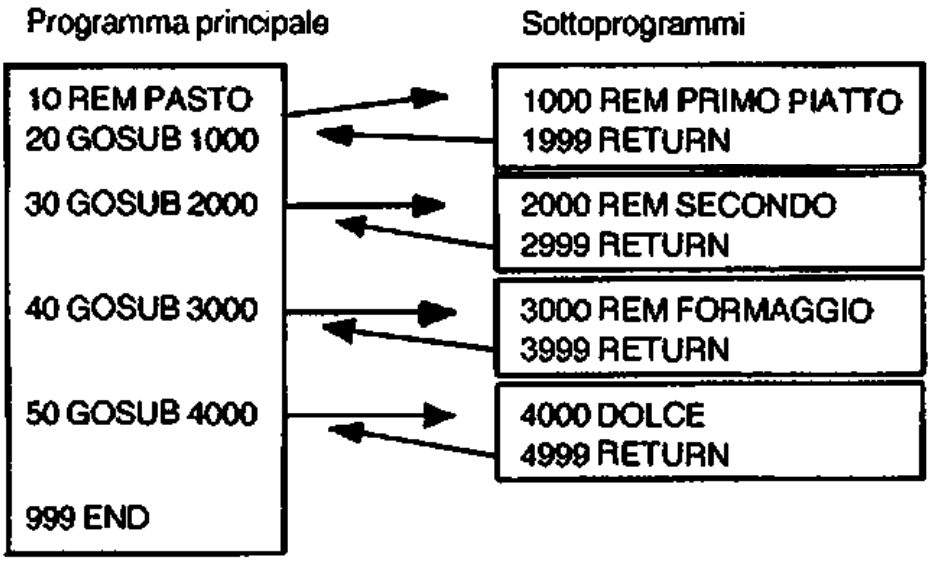
I principali vantaggi offerti dall'analisi top–down sono i seguenti:
– possibilità di dividere il programma tra più programmatori,
– esame delle difficoltà una per una,
– strutturazione automatica dei programmi,
– indipendenza nei confronti del linguaggio di programmazione.

Non è certamente necessario usare l'analisi top–down ogni volta che si devono scrivere poche righe di programma in BASIC. D'altra parte, bisogna dire che il BASIC è un linguaggio che non si adatta molto alla programmazione strutturata. Il PASCAL è stato ideato per adeguarsi a questo stile e i risultati sono stati ottimi. Comunque i principi dell'analisi top–down devono spingere il lettore a rispettare sempre questa massima

> Riflettere prima
> Programmare dopo

Sottoprogrammf

I principi dell'analisi top–down fanno dei sottoprogrammi uno degli strumenti essenziali della programmazione. I sottoprogrammi servono per rinviare a una fase successiva quello che non può essere fatto immediatamente per non perdersi nei dettagli.
Un sottoprogramma si presenta come una serie di istruzioni nel programma. Per evidenziare meglio un sottoprogramma nel listato, si consiglia di iniziare la numerazione con un numero "tondo" (100, 200, 1000, 2000, 10000, 20000, ecc.) e di inserire sulla prima di queste righe una REM che indichi il compito svolto dal sottoprogramma. La fine del sottoprogramma è indicata dall'istruzione RETURN. Per eseguire il sottoprogramma, utilizzare l'istruzione GOSUB seguita dal numero della prima riga del sottoprogramma che si desidera chiamare. In questo modo il listato risulta diviso in due parti:
– il programma principale in cui vengono chiamati i sottoprogrammi,
– l'insieme dei sottoprogrammi.
Tutte queste spiegazioni all'inizio possono sembrare un po' complesse per coloro che si avvicinano per la prima volta al mondo della programmazione, ma è molto importante capire sin dall'inizio la struttura di un programma BASIC. Il programma principale deve essere in testa al programma: comincia con una istruzione REM che contiene il nome (ed eventualmente l'autore) del programma e finisce con l'istruzione END che segna la fine del programma principale.
Per chiarire meglio quanto affermato, riportiamo uno schema che riassume la preparazione informatica di un pasto tradizionale. Il ruolo del capo cuoco è svolto dal programma principale che chiama a sua volta i sottoprogrammi che corrispondono a ogni portata.



Commento: t'esecuzione dei sottoprogrammi avviene in base alla modalità dell'andata e ritorno: GOSUB per andare, RETURN per tornare. Per la numerazione vengono riservate delle fasce complete. Un modo interessante di fissare i limiti della numerazione del programma e dei sottoprogrammi è il seguente: il programma principale resta compreso tra 1 e 999,i sottoprogrammi tra 1000 e 1999, tra 2000 e 2999, ecc. Se quindi si vuole scrivere un sottoprogramma 5000, si può iniziare a inserire

```
5000 REM CAFFE'
5999 RETURN
```

Capitolo 23

Interruzione di sequenza

Fino a quando non incontra un'istruzione GOSUB, il programma si svolge in modo sequenziale, cioè seguendo l'ordine dei numeri di riga. Non è fondamentale chiamare i sottoprogrammi seguendo un ordine preciso. Basta cambiare le istruzioni del programma principale.

```
20 GOSUB 3000
30 GOSUB 1000
40 GOSUB 4000
50 GOSUB 2000
```

L'istruzione END è l'ultima riga del programma principale prima della prima riga del primo sottoprogramma.

Il programmatore è comunque libero di usare un metodo personale che non è necessariamente il più razionale o il più rigido, ma che può rivelarsi più efficace e più pratico per lui.

L'autore di questo metodo, a cui spesso capita di dover sviluppare dei programmi in BASIC di una certa consistenza, segue le regole seguenti.

Scrive il programma su un foglio, prima a grandi linee e poi precisando sempre di più le varie fasi. Poi passa alla programmazione in BASIC. Alcuni compiti particolari possono essere programmati interamente in un sottoprogramma che può essere provato successivamente sul PC t2B. Iniziamo quindi a vedere com'è strutturato un sottoprogramma.

Esempi

Perimetro e superficie

```
1 '
2 'PERIMETRO E SUPERFICIE
3 '
20 LUN=5.6:LAR=2.8
30 GOSUB 1000
40 GOSUB 2000
999 END
1000 '
1001 'PERIMETRO
1002 '
1010 PER=(LUN+LAR)*2
1020 PRINT "PERIMETRO:";PER
1099 RETURN
2000 '
2002 ' SUPERFICIE
2003 '
2010 SUP=LUN*LAR
2020 PRINT "SUPERFICIE";":";SUP
2099 RETURN
```

```
RUN

PERIMETRO:16.8
SUPERFICIE:15.68
```

Commenti: evidentemente sarebbe possibile limitare il numero dei sottoprogrammi. I programmi risulterebbero più concisi ma si perderebbero le buone abitudini della programmazione.

Conversioni

Il vantaggio principale di un sottoprogramma risulta più evidente quando viene chiamato più volte all'interno dello stesso programma. Consideriamo un esempio di conversione da dollari in franchi, dove il dollaro sia pari a 6,90 franchi francesi.

```
1 '
2 'CONVERSIONE FRANCO-DOLLARO
3 '
20 DF=6.90
30 SD=15:GOSUB 1000
40 SD=3850.65:GOSUB 1000
50 SF=60:GOSUB 2000
60 SF=1000:GOSUB 2000
999 END
1000 '
1001 'DOLLARO-FRANCO
1002 '
1010 SF=DF*SD
1020 PRINT SD;"DOLLARI=";SF;"FRANCHI"
1099 RETURN
2000
2001 'FRANCO-DOLLARO
2002 '
2010 SD=INT((SF/DF*100)/100)
2020 PRINT SF;"FRANCHI=";SD;"DOLLARI"
2099 RETURN

RUN
15 DOLLARI=103.5 FRANCHI
3850.65 DOLLARI=26569.48 FRANCHI
60 FRANCHI=8.69 DOLLARI
1000 FRANCHI=144.92 DOLLARI
```

Commenti: Variabili:

- DF è il corso del dollaro in franchi
- SD è una somma in doltari
- SF è una somma in franchi

Il sottoprogramma 1000 converte i dollari in franchi. Dato SD, calcola SF in base a una formula data. Si dice che SD è una variabile di input e SF una variabile di output.

Il sottoprogramma è una macchina che svolge un certo compito. E' necessario fornire la materia prima (te variabili di input) perchè possa produrre dei risultati (le variabili di output).

Il sottoprogramma 2000 converte i franchi in dollari. Le variabili di input diventano variabili di output e viceversa.

La conversione dei franchi in dollari avviene operando una divisione. La formula della riga 2010 dà il risultato di questa divisione con due cifre significative. Se SF/DF=25.64387, allora 100*(SF/DF)=2464.387, quindi INT(100*SF/DF))=2564 e infine INT(100*(SF/DF))/100=25.64.

Arrotondamento

Il metodo esposto precedentemente può essere perfezionato per ottenere un vero e proprio arrotondamento, e cioè che 8,348 venga arrotondato a 8,35, con la seguente istruzione:

```
SD=INT((SF/DF)*100+0.5)/100
```

Il semplice fatto di aggiungere 0,5 permette di forzare l'arrotondamento. Anticipiamo che esiste un'istruzione di visualizzazione specifica, PRINTUSING, che consente tra l'altro di operare degli arrotondamenti (vedere capitolo 42).

Capitolo 24

Tre dadi

Progetto

Si vogliono visualizzare a caso le facce di tre dadi. Questa semplice routine potrà essere riutilizzata in un programma più complesso.

Analisi

A. Produrre a caso un numero intero compreso tra 1 e 6
B. A seconda del numero, rinviare il controllo a uno dei sei sottoprogrammi che visualizzeranno la faccia del dado.
C. Eseguire A e B tre volte

Istruzioni BASIC usate per questa procedura

Per A si userà l'istruzione RND che fornisce dei numeri aleatori (capitolo 17) e l'istruzione INT che dà dei valori interi. Per B è necessaria soltanto l'istruzione PRINT. Per i punti del dado si userà l'asterisco (*), il segno – per i lati orizzontali e il simbolo ! per i lati verticali. Per visualizzare la faccia 6 si potrebbero concatenare le istruzioni.

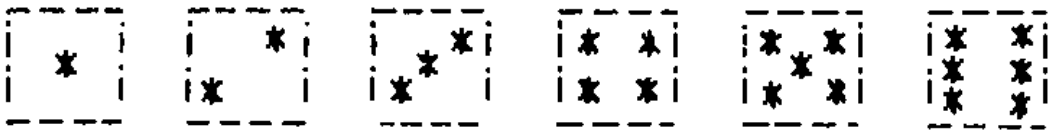
```
PRINT"-----"
PRINT"!* *!"
PRINT"!* *!"
PRINT"!* *!"
PRINT"-----"
```

Notiamo che questo risultato potrebbe essere ottenuto in modo più economico dato che la stessa stringa viene stampata più volte. Si stabilisce quindi una lista di cinque stringhe elementari che permettono di visualizzare le sei facce del dado.

Stringhe elementari

```
"!     !"; "!*    !"; "!    *!"; "!* *!"; "!  *  !";
```

Le sei facce



L'ordine e le stringhe scelte determineranno quale faccia verrà visualizzata. Bisogna ancora vedere come passare dal programma a uno dei sei sottoprogrammi di visualizzazione.

ON...GOSUB

Con un solo comando si ottiene l'effetto desiderato. Se K è un numero intero, l'istruzione ON K GOSUB 1000, 2000, 3000 passa il comando al sottoprogramma 1000 se K vale 1, al sottoprogramma 2000 se K vale 2, al sottoprogramma 3000 se K vale 3 e ignora questa istruzione se K è maggiore di 3. Il numero dei valori possibili per K, l'ordine e la numerazione dei sottoprogrammi vengono stabiliti dall'utente.

Per C si usa un loop FOR...NEXT.

Attenzione: per ottenere una vera e propria "casualità" (vedere capitolo 29), la riga 40 viene modificata nel modo seguente.

Programma

```
10 REM TRE DADI
20 A$="-----":B$="! *!":C$="! * !"
30 D$="!*!":E$="!* *!":FS="! !"
40 REM SPAZIO RISERVATO
50 REM TIRI
60 FOR I=1 TO 3
70 K=1+INT(RND*6):PRINT A$
80 ON K GOSUB 1000,2000,3000,4000,5000,6000
90 PRINT A$
100 NEXT I
199 END
1000 '
1001 'FACCIA 1
1002 '
1010 PRINT F$:PRINT C$:PRINT FS
1999 RETURN
2000 '
2001 'FACCIA 2
```

```
2002 '
2010 PRINT D$:PRINT F$:PRINT B$
2999 RETURN
3000 '
3001 'FACCIA 3
3002 '
3010 PRINT D$:PRINT C$:PRINT B$
3999 RETURN
4000 '
4001 'FACCIA 4
4002 '
4010 PRINT E$:PRINT F$:PRINT E$
4999 RETURN
5000 '
5001 'FACCIA 5
5002 '
5010 PRINT E$:PRINT C$:PRINT E$
5999 RETURN
6000 '
6001 'FACCIA 6
6002 '
6010 PRINT E$:PRINT E$:PRINT E$
6999 RETURN
```

La riga 40 è riservata per l'impostazione dei numeri aleatori (vedere capitolo 29).

Introduzione al test

Normalmente per distinguere un numero pari da un numero dispari è sufficiente guardare la cifra delle unità. Se è 0, 2, 4, 6, 8, allora il numero è pari, in caso contrario è dispari. Oppure si può dividere il numero per due e vedere se la divisione non ha resto. Il computer però non ha la possibilità di fare questo tipo di valutazione. L'istruzione del test serve a questo scopo.
La sintassi del test in BASIC utilizza le parole inglesi IF (se) e THEN (allora). Nella maggior parte dei nostri esempi, IF sarà seguito da una relazione di confronto tra i contenuti di due variabili dello stesso tipo o di una variabile e di una costante dello stesso tipo. Per quanto riguarda le variabili numeriche, la relazione di confronto può assumere sei forme diverse:

```
A=B; A<B; A>B; A<=B; A>=B; A<>B

<= significa minore di o uguale a
```

Attenzione

Il segno = di relazione di confronto tra i contenuti delle variabili A e B, posto dopo IF, non ha niente a che vedere con il segno = dell'istruzione di assegnazione in quanto le variabili sono precedute da IF, che indica un'operazione successiva relazionale.

Capitolo 25

Test

Questa è la struttura completa di un test in BASIC:

se [condizione] allora [fare questo] altrimenti [fare quest'altro]

e cioè:

IF [condizione] THEN [fare questo] ELSE [fare quest'altro]

In questo caso l'uso di ELSE è facoltativo mentre quello di THEN non lo è. Se vengono però usati sia THEN sia ELSE, dopo aver eseguito le istruzioni successive a queste parole, il programma prosegue dalla riga sottostante. L'uso migliore che si possa fare di questa sequenza è di indirizzare il programma verso due sottoprogrammi diversi in funzione del risultato del test. Per esempio:

```
IF A=B THEN GOSUB 1000 ELSE GOSUB 2000
```

che corrisponde esattamente all'istruzione:

```
IF A<>B THEN GOSUB 2000 ELSE GOSUB 1000
```

Gli operatori logici

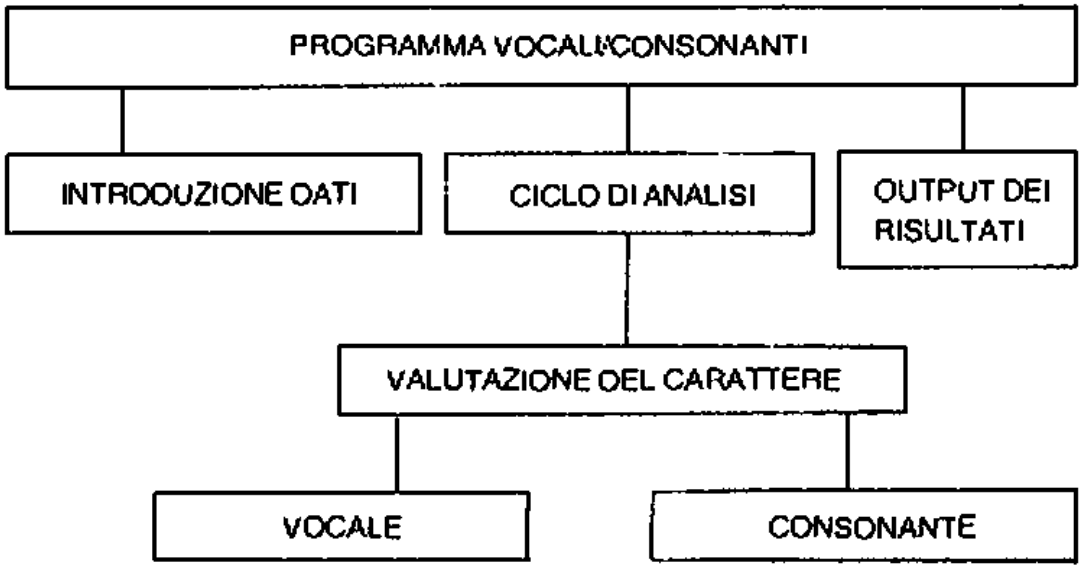
E' possibile raggruppare diverse condizioni nella stessa istruzione IF...THEN...ELSE servendosi degli operatori logici. Gli operatori logici a disposizione sono AND (E), OR (O) et XOR (O esclusivo). Quest'ultimo operatore significa "uno o l'altro" ma non i due insieme. E' possibile scrivere l'opposto di una condizione utilizzando NOT. Gli operatori logici verranno analizzati in dettaglio nel capitolo 35. Per il momento, è sufficiente un po' d'intuito per capire gli esempi che seguono.

Vocali e consonanti

Ecco un esempio che seleziona separatamente le vocali e le consonanti di una stringa di caratteri. Tutti i caratteri che non appartengono a una di queste categorie vengono ignorati.

Analisi

1. Prendere ogni carattere della stringa, uno dopo l'altro.
2. Controllare se si tratta di una lettera dell'alfabeto.
3. Controllare se si tratta di una vocale o di una consonante.
4. Ordinare le vocali e le consonanti in due stringhe di caratteri riservate a questo scopo.



Istruzioni BASIC usate

1. I caratteri vengono individuati con l'istruzione MID$ (vedere capitolo 8).
2. I caratteri vengono controllati con l'istruzione IF...THEN abbinata ai codici ASCII (vedere capitolo 10) che permettono di distinguere le lettere maiuscole (ELSE facoltativo).
3. Le vocali e le consonanti vengono raccolte per concatenazione (vedere capitolo 15) in due stringhe (V$ e C$).

Il programma

```
1'
2'VOCALI E CONSONANTI
20 AS="ALI-BABA E I 40 LADRONI"
30 V$="":CS="":L=LEN(A$)
40 REM CICLO DI ANALISI
50 FOR K=1 TO L
60 LA$=MIDS(AS,K,1)
70 IF LA$>="A" AND LA$<="Z" THEN GOSUB 1000
80 NEXT K
90 REM BILANCIO
100 PRINT A$
110 PRINT "VOCALI:";VS
120 PRINT "CONSONANTI:";C$
999 END
1000'
1001 'VALUTAZIONE DEL CARATTERE
1002'
1110 TLA=INSTR("AEIOU",LAS)
1120 IF TLA=0 THEN GOSUB 12000 ELSE GOSUB 11000
1999 RETURN
11000'
11001 'VOCALI
11002'
11010 V$=V$+LA$
11999 RETURN
12000'
12001 'CONSONANTI
12002'
12010 C$=C$+LA$
12999 RETURN

RUN

ALI-BABA E I QUARANTA LADRONI
VOCALI:AIAAEIUAAAAOI
CONSONANTI:LBBQRNTLORN
```

Osservazioni:

70 : Primo test. Determina se un carattere è una lettera maiuscola o minuscola.
1110 : L'istruzione INSTR permette di determinare la presenza di una stringa in un'altra stringa. In questo caso si vede se il carattere LA$ è presente nella stringa formata dalle vocali ("AEIOU"). Se è presente, la variabile TLA prende il valore della vocale individuata. Se TLA è uguale a 0, si può concludere che la lettera è una consonante.
1120 : I sottoprogrammi 11000 e 12000 consentono di inviare il carattere in una delle due stringhe, C$ o VS.

L'istruzione IF...THEN...ELSE apre ampie prospettive. Notiamo che è possibile legare tra loro diverse istruzioni IF:

```
IF (condizione 1) THEN (IF (condizione 2)...)
                       ELSE (IF (condizione 3)...)
```

Attenzione a non confondersi.
Per concludere e per dimostrare che è possibile trattare delle istruzioni in un IF...THEN...ELSE piuttosto che in sottoprogrammi, proviamo a risolvere il problema della determinazione dei numeri pari con la divisione senza resto.

Divisione senza resto

Per sapere se la divisione di A per 2 è senza resto, bisogna dividere A per 2. La divisione è esatta se non c'è parte decimale, e cioè se la parte intera del quoziente è uguale al quoziente. Può sembrare strano, ma nell'ottica del computer è molto logico. Se A/2 =INT (A/2), allora la divisione non ha resto.

```
1 '
2 'NUMERI PARI
3 '
20 X=2367
30 Q1=X/2:Q2=INT(Q1)
40 IF Q1=Q2 THEN P$="PARI" ELSE
   P$="DISPARI"
60 PRINT X;"E'";P$
100 END
```

```
RUN
2367 E' DISPARI
OK
```

Capitolo 26

Cicli senza indice

Abbiamo imparato come creare dei loop. Tuttavia l'istruzione FOR...NEXT che stiamo usando già da un po' di tempo presenta un inconveniente: bisogna sapere prima il numero di iterazioni, ovvero il numero di ripetizioni, da inserire. Questa limitazione può essere fastidiosa in alcuni casi. Vedremo che l'uso della coppia DO...LOOP risolve questo tipo di problema.
L'inizio di un ciclo è dato dall'istruzione DO (dall'inglese "fare").
La fine del ciclo è segnalata dall'istruzione LOOP (dall'inglese "ciclo"). Tutte le istruzioni comprese tra DO e LOOP verranno ripetute. Per evitare una ripetizione all'infinito, è necessario inserire un'istruzione di fine ciclo. Il formato di questa istruzione è il seguente:

IF (condizione di uscita) THEN EXIT

L'istruzione EXIT provoca il salto all'istruzione che segue l'istruzione LOOP.
Esempio: Si pensi di lanciare dei dadi e di fermarsi non appena le due facce sono uguali. La succesione dei tiri aleatori avviene in un ciclo DO...LOOP. Si stabilisce che si ha vinto se si riesce a fare una coppia con meno di sei tentativi.

Osservazione: dato che si usa una funzione di casualità, si ha comunque sempre il problema della "vera casualità". A questo proposito, fare riferimento al capitolo 29.

```
1 '
2 'COPPIA CON SEI LANCI
3 '
20 REM VERA CASUALITA'
30 CONTATORE=1
40 DO
50 F1=1+INT(RND*6)
60 F2=1+INT(RND*6)
70 IF F1=F2 THEN EXIT
80 CONTATORE=CONTATORE+1
90 LOOP
100 REM RISULTATO
110 IF CONTATORE<=6 THEN PRINT "VINTO"
    ELSE PRINT "PERSO"
120 PRINT "CONTATORE" CONTATORE
130 END
```

Uscita dal ciclo e flag

Confrontiamo questo programma con quello che segue che utilizza l'istruzione FOR...NEXT con una lunghezza di ciclo uguale a 6. Se la coppia esce al primo tiro, si può decidere che è inutile continuare e quindi si preferisce uscire dal ciclo senza che questo venga svolto interamente per non perdere tempo.

```
10 REM COPPIA CON 6 TIRI
20 FLAG=0
30 FOR K=1 TO 6
40 F1=1+INT(RND^6)
50 F2=1+INT(RND^6)
60 IF F1=F2 THEN FLAG=1:EXIT
70 NEXT K
80 REM RISULTATO
90 IF FLAG=1 THEN PRINT "VINTO"
   ELSE PRINT "PERSO"
100 END
```

Commenti

20. FLAG significa bandiera e abbiamo scelto questo nome di variabile perché svolge proprio il ruolo di una bandiera. All'inzio FLAG vale 0, ovvero la bandiera è abbassata.
50. E' il test per sapere se è uscita la coppia. Se è uscita, avvengono due cose importanti: si alza la bandiera (FLAG=1) e si provoca l'uscita dal ciclo con l'istruzione EXIT che restituisce il valore di K al momento dell'uscita.
90. Se la bandiera è alzata, significa che è uscita una coppia.

In BASIC il FLAG è una variabile che indica che un'istruzione è stata eseguita o per passaggio su una riga specifica o nell'analisi di un test.

Calcolo delle probabilità

E' interessante calcolare la probabilità di riuscita offerta dal gioco precedente: probabilità di far uscire una coppia con meno di sei tiri. Dato che si tratta di informatica, si può scrivere un programma di simulazione che esegue l'esperimento 100 volte e indica quante volte esce una coppia prima del sesto tiro. Ne approfitteremo per combinare tra loro i due cicli di loop.

```
1 '
2 'SIMULAZIONE
3 '
20 REM CASUALITA'
30 NC=0 CC=1
40 DO
50 FOR K=1 TO 6
60 F1=1+INT(RND*6)
70 F2=1+INT(RND*6)
80 IF F1=F2 THEN : NC=NC+1:EXIT
90 NEXT K
```

```
100 IF CC=100 THEN EXIT
110 CC=CC+1
120 LOOP
130 PRINT NC
140 END
```

Gli esperimenti fatti sembrano dimostrare una percentuale di riuscita di circa il 60%.

Commenti

Il FLAG è stato tolto in quanto inutile. NC è un contatore che rileva il numero delle coppie. CC è il contatore del numero di prove.

L'istruzione HEAD

Proviamo a far fare dei balzi alla tartaruga.

```
1 '
2 'BALZO
3 '
20 CLS
30 TURTLE 1.0,100
40 SHOW1,1
50 TRACE1
60 DO
70 FWD5:HEAD4:ROT4
80 IF HEAD=68 THEN HEAD 124
90 LOOP
```

Commenti

Per fermare la tartaruga quando è in posizione verticale, si utilizza la funzione HEAD (che ha lo stesso nome dell'istruzione) che dà il valore della direzione della tartaruga (come avviene con l'istruzione ZOOM che ne definisce la dimensione).
80. Quando la direzione raggiunge 68, la tartaruga deve girare di 124.
70. La tartaruga ha appena tracciato un segmento verticale, la sua direzione è già 68 dato che l'incremento tramite l'istruzione HEAD avviene dopo il tracciamento con FWD. Quindi l'ultimo segmento tracciato corrisponde a HEAD=68. A questo punto è sufficiente aggiungere 124 alla direzione della tartaruga in modo che si ritrovi in verticale nell'altro senso.

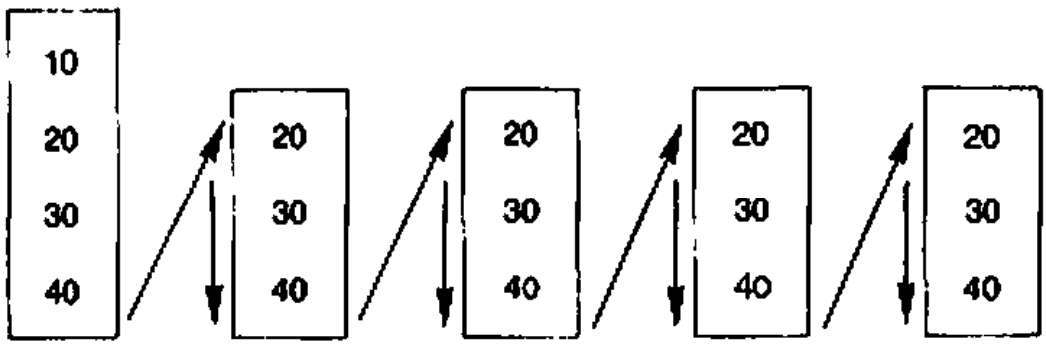
Capitolo 27

Salto condizionato

E' giunto il momento di prendere in considerazione una delle più famose istruzioni BASIC, e cioè l'istruzione GOTO che significa ANDARE A e che permette, come l'istruzione GOSUB, di indirizzare lo svolgimento del programma su un'altra serie di istruzioni. Ma a differenza di GOSUB, l'istruzione GOTO non assicura il ritorno al punto di partenza, e questo rappresenta sicuramente uno svantaggio.
Comunque GOTO è un'istruzione molto pratica che piace molto ai principianti perchè consente delle realizzazioni spettacolari con poca fatica. Il primo esempio consiste nella visualizzazione di molti numeri su video. Inoltre questo programma non si ferma da solo (è inutile inserire un'istruzione END) e consigliamo all'utente di consultare il capitolo 19 prima di tentare questo esperimento perchè lo svolgimento di questo programma si interromperà solo con un intervento dall'esterno.

```
10 A=0
20 PRINT A;
30 A=A+1
40 GOTO 20
```

L'istruzione GOTO 20 interrompe lo svolgimento sequenziale del programma indirizzandolo alla riga 20. Dopo l'esecuzione della riga 20 e in mancanza di un'altra istruzione di salto condizionato, riprende lo svolgimento del programma fino alla nga 40, poi ritorna alla riga 20, ecc. Ecco come si presentano le righe di programma nell'ordine di esecuzione:



Il passaggio alla riga 20 stampa il contenuto della variabile A, il passaggio alla riga 30 aumenta di una unità il contenuto di questa variabile. Ecco il risultato:

```
RUN
0  1  2  3  4  5  6  7  8  9  10  11
12  13  14  15  16  17  18  19  20  21
22  23  24  25  26  27  28  29  30  31
32  33  34  35  36  37  38  39  40  41
42  43  44  45  46  47  48  49  50  51
52  53  54  55  56  57  58  59  60  61
62  63  64  65  66  67  68  69  70  71
```

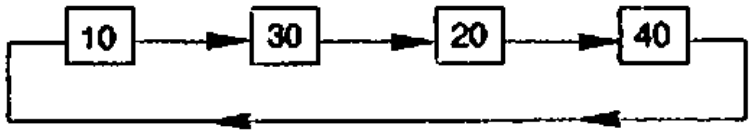
Per interrompere lo svolgimento del programma premere CTRL–C. Vediamo ancora altri esempi dell'istruzione GOTO con due programmi assurdi che non producono alcun risultato e durano moltissimo tempo.

```
10 GOTO 10
```

E' un programma veramente assurdo che si può interrompere premendo CTRL-C.

```
10 GOTO 30
20 GOTO 40
30 GOTO 20
40 GOTO 10
```

Non è un programma molto intelligente ma è interessante seguire l'ordine delle esecuzioni



Se l'istruzione GOSUB tende a strutturare i programmi, l'istruzione GOTO tende a confonderli.

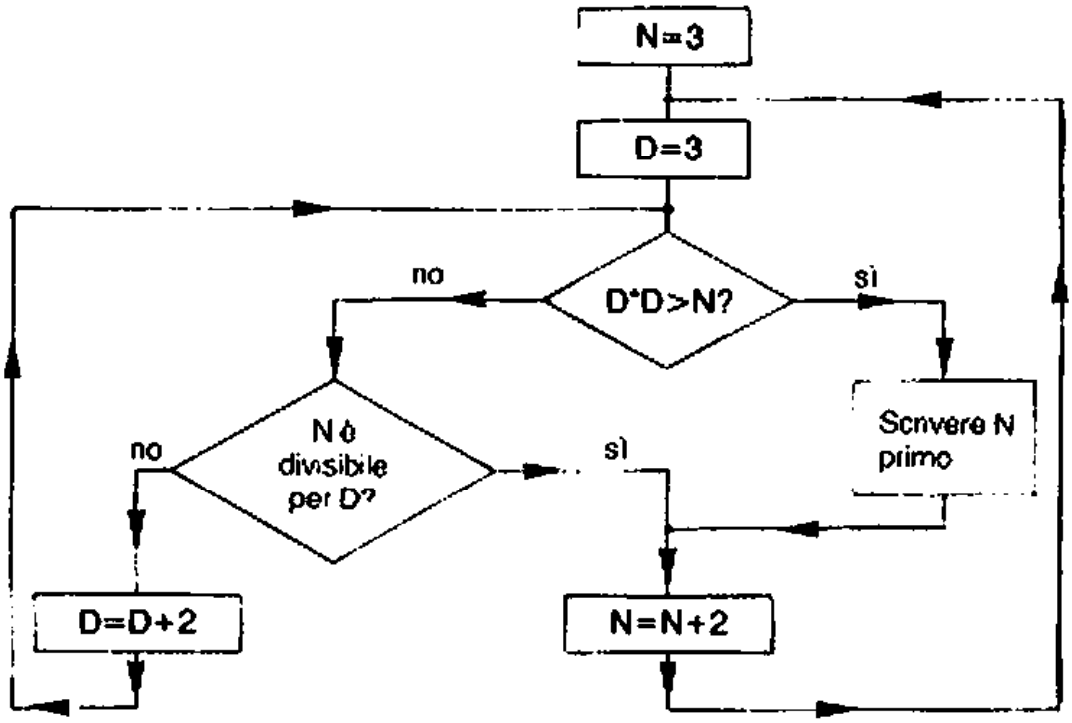
Esempio: i numeri primi

Da sempre i numeri primi esercitano sui matematici professionisti e dilettanti un fascino che non si smentisce mai.
Ricordiamo che un numero primo è un numero intero divisibile solo per se stesso e per 1. 19 è un numero primo, 20 non lo è. I numeri primi sono molto numerosi e sono distribuiti rispetto agli altri numeri in modo molto misterioso.

E' possibile scrivere un programma molto semplice che produce l'elenco dei numeri primi. Il metodo utilizzato può essere uno qualsiasi. Il peggiore consiste nel dividere il numero esaminato per tutti quelli minori del numero stesso. Riflettendo è possibile perfezionarlo nel modo seguente:
1. Si esamineranno solo i numeri dispari a partire da 3 dato che i numeri pari non sono chiaramente dei numeri primi.
2. Per un motivo analogo, si cercheranno dei divisori solo tra i numeri dispari: un numero dispari non è divisibile per un numero pari.
3. Inoltre: per sapere se N è un numero primo, basta cercare un divisore eventuale inferiore alla radice quadrata di N.

Flow–chart

Un flow–chart è uno schema che riassume l'algoritmo di un programma. Vi sono pareri contrastanti sul flow–chart. Noi riteniano che troppo spesso i flow–chart vengono definiti a posteriori. Ecco quindi il flow–chart del programma per la ricerca dei numeri primi che permette di capire meglio i diversi salti del programma.



Il programma segue i vari salti grazie alle istruzioni GOTO.

```
10 REM NUMERI PRIMI
20 N=3
30 D=3
40 IF D*D>N THEN GOTO 70
50 Q1=N/D:Q2=INT(Q1)
60 IF Q1=Q2 THEN GOTO 80 ELSE D=D+2:GOTO 40
70 PRINT N;
80 N=N+2:GOTO 30
```

Attenzione: nell'istruzione IF...THEN, il GOTO è facoltativo; quindi nella riga 40 si potrà scrivere: IF D*D>N THEN 70.

Capitolo 28

Introduzione dati

ON...GOTO

Questa sintassi funziona in modo simile a ON...GOSUB (vedere capitolo 24) a differenza del fatto che GOTO non assicura il ritorno al punto di partenza del programma, tuttavia permette di limitare il numero di istruzioni in caso di salti multipli in funzione dei valori sssunti da una variabile.

```
100 IF K=1 THEN GOTO 150
110 IF K=2 THEN GOTO 200
120 IF K=3 THEN GOTO 250
```

è equivalente a

```
100 ON K GOTO 150,200,250
```

Personalizzazione

L'istruzione INPUT comporta nel programma le seguenti conseguenze:
1. Interrompe lo svolgimento del programma.
2. Mette anche il PC128 "all'ascolto" della tastiera e visualizza un punto interrogativo.
3. Tutti i tasti che vengono battuti vengono memorizzati in una memoria tampone.
4. Il tasto ENTER invia il contenuto della memoria tampone nella variabile il cui nome segue obbligatoriamente la parola INPUT (in questo caso la variabile numerica A).
5. Se il contenuto della memoria tampone non corrisponde al tipo di variabile, appare il messaggio di errore REDO (rifare). La memoria tampone si vuota e il programma ritorna all'istruzione INPUT.
6. Se è stato rispettato il tipo della variabile, il programma continua in sequenza con il nuovo contenuto di A.

Esempi

```
10 INPUT A
RUN
?1917 [ENT]
OK
PRINT A
1917
OK
RUN

? ACCIDENT' [ENT]
REDO
? 1848 [ENT]
OK
PRINT A
1848
OK
```

il tipo di variabile (numerica) non corrisponde ai caratteri inseriti (stringa di caratteri)

```
10 PRINT "COME TI CHIAMI:";
20 INPUT N$
30 PRINT "CHE BEL NOME,";N$;"!"
40 END
RUN
COME TI CHIAMI:? PAOLO [ENT]
CHE BEL NOME. PAOLO!
OK
```

Ottimizzazioni

1. Le righe 10 e 20 del programma precedente potevano essere risssunte in una sola riga.

```
10 INPUT "COME TI CHIAMI:";N$
```

2. Si possono memorizzare più variabili con una sola istruzione INPUT separandole con delle virgole.

```
10 INPUT A,B,C
20 PRINT A+B+C
RUN
?3,4,5
12
OK

10 INPUT "NOME,ETA':";N$,E
20 PRINT N$:"HA";E:"ANNI."
RUN
NOME,ETA':? PAOLO,34
PAOLO HA 34 ANNI
OK
```

Se si vuole avere una sola risposta con INPUT, attenzione a non inserire delle virgole, in quanto questo comporterebbe un messaggio di errore di questo genere: "Redo from start" (ricominciare dall'inizio).
Questo è comprensibile in quanto l'interprete BASIC considera che le risposte all'istruzione INPUT siano separate da delle virgole. Quindi se c'è una virgola nella risposta, crede che vi siano due risposte e, dato che ne aspettava solo una, appare il messaggio di errore.
Se è necessario mantenere la virgola, sarà necessario mettere la risposta tra virgolette. Questo potrebbe risultare fastidioso per la comprensione dei testi. Se si vuole che la risposta contenga veramente qualsiasi carattere alfanumerico usare l'istruzione LINE INPUT.

```
LINE INPUT "Qualsiasi parola";A$
```

permette di attribuire alla stringa A$ una serie qualsiasi di caratteri.

Osservazione e conclusione:
L'istruzione LINE INPUT non visualizza il punto interrogativo e ignora le virgole.

Capitolo 29

Casualità

L'istruzione INPUT (capitolo 28) è riservata ad un uso semplice dell'interattività. Quando incontra l'istruzione INPUT, il programma si ferma e attende che l'utente prema il tasto ENTER. Questa procedura è evidentemente incompatibile con un uso rapido o un'analisi approfondita dei caratteri introdotti. Esiste un'istruzione BASIC che permette di verificare lo stato della tastiera ed eventualmente di continuare il programma. Si tratta dell'istruzione INKEY$. Quando il programma incontra l'istruzione INKEY$, apre una memoria tampone che è in grado di ricevere *solo* un carattere e vi registra il carattere inserito da tastiera. Questo carattere può essere posto in una stringa di caratteri variabile con una normale sssegnazione (A$=INKEY$) in modo tale che sia facilmente analizzabile. Le differenze essenziali tra INPUT e INKEY$ sono le seguenti:

1. INKEY$ memorizza solo un carattere.

2. Questo carattere è di tipo stringa.

3. Non è necessario premere il tasto ENTER con l'istruzione INKEY$.

4. INKEY$ non interrompe il programma.

5. Il carattere inserito con INKEY$ non viene visualizzato.

In genere l'istruzione tNKEYS è subito seguita da dei test che permettono di far saltare il programma in funzione dei tasti premuti. Di solito si comincia esaminando il contenuto di INKEY$.

```
10 PRINT "PRESTO, PREMI UN TASTO!"
20 IF INKEYS=""THEN GOTO 10
30 PRINT "UFFA...GRAZIE!"
40 END
```

Nessun commento.
Ecco una versione migliorata.

```
10 PRINT "PRESTO, PREMI A!"
20 A$=INKEY$
30 IF A$<>"A" THEN GOTO 10
40 PRINT "UFFA...GRAZIE!"
50 END
```

Dato che sicuramente A$ contiene una stringa lunga al massimo 1 carattere, si può effettuare una verifica solo sul codice ASCII di questo carattere, indispensabile per controllare se sono stati premuti i tasti di spostamento cursore o il tasto ENTER.

SI e NO

Questa è un'alternativa che spesso viene proposta all'utente dal programma. Se una domanda comporta una risposta del tipo Sì e No, la verifica verrà fatta su un singolo tasto per snellire l'interattività. Il programma torna a INKEY$ quando il tasto premuto è diverso da S e N. Questa precauzione impedisce che l'utente risponda in modo sbagliato.

```
10 PRINT"STAI BENE?"
20 DO
30 A$=INKEY$
40 IF A$="S" THEN GOSUB 100:EXIT
50 IF A$="N" THEN GOSUB 200:EXIT
60 LOOP
99 END
100 '
101 'SI'
102 '
110 PRINT "SONO CONTENTO!"
199 RETURN
200 '
201 'NO
202 '
210 PRINT "PECCATO!"
299 RETURN
```

La vera casualità

L'istruzione INKEY$ consente di effettuare operazioni veramente casuali.

```
10 PRINT "PREMI UN TASTO"
20 X=RND:IF INKEY$=""THEN GOTO 20
30 PRINT RND
40 END
```

La riga 20 permette di prolungare l'effetto della pressione di tasti anche quando questi non sono stati effettivamente premuti. Alla variabile X viene assegnato ogni volta un numero aleatorio. Dato che il tempo di attesa varia necessariamente ad ogni uso, si ottiene ad ogni svolgimento del programma una inizializzazione diversa dalla sequenza utilizzata.
A partire da questo programma, lanciare il programma più volte (istruzione RUN) e confrontare i risultati.

Verifica di diversi caratteri da tastiera: INPUT$

Nel dialogo con il programma, può essere interessante avere un controllo automatico sul numero dei caratteri introdotti dall'utente, indipendentemente dai tasti premuti.
L'istruzione INPUT$ è stata prevista a questo scopo.
Attenzione: non si tratta di un'istruzione come INPUT ma di una funzione come RIGHT$ quindi dà un risultato che deve essere assegnato ad una variabile oppure elaborato direttamente. Nella funzione INPUT$ si indica il numero di caratteri attesi e la funzione dà i caratteri inseriti da tastiera. Il test S o N può allora essere riscritto usando la funzione INPUT$(1). La possibilità di verificare una risposta di più caratteri può servire per inserire delle parole chiave per proteggere il software.

Esempio:

```
10 REM DIMOSTRAZIONE
20 PRINT "PAROLA CHIAVE"
30 REP$=INPUT$(5)
40 IF REP$<>"MARCO" THEN NEW
   ELSE PRINT "BENVENUTO"
50 REM INIZIO PROGRAMMA
999 FINE
```

Commenti

Nella riga 30, l'interprete non passa alle altre istruzioni fino a quando l'utente ha inserito cinque caratteri. Per verificare la differenza delle due stringhe, si utilizza il segno <>, come per i numeri. Nella riga 40, l'istruzione NEW annulta tutto quello che era presente in memoria. Questa istruzione deve quindi essere usata con molta attenzione.

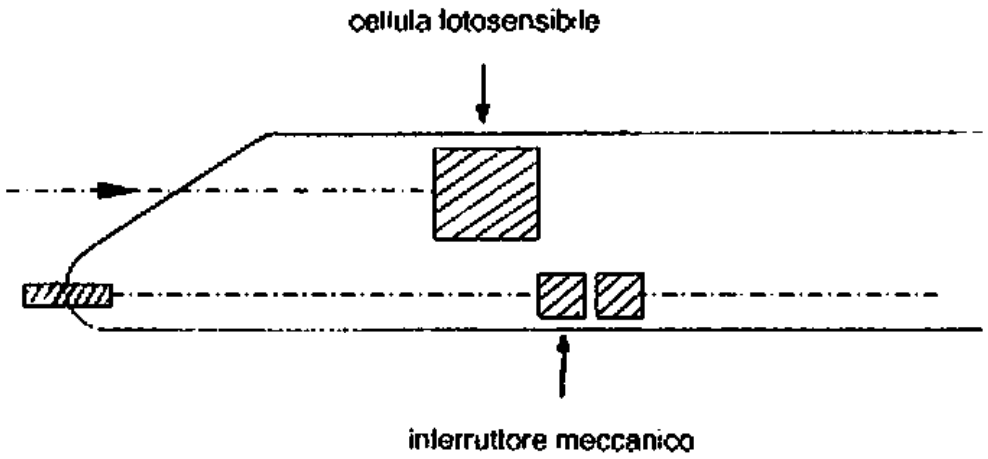
Attenzione:

Si potrebbe affermare che questo metodo non è molto sicuro. Infatti basta che l'utente dia il comando LIST ed ecco che le prime righe del programma appaiono sul video rivelando la parola chiave.

Capitolo 30

La penna ottica

La penna ottica è uno strumento di interattività molto potente usato da numerosi programmi. I comandi che controllano te funzioni della penna ottica sono molto facili. Lo stato della penna è determinato da due parametri: lo stato di un interruttore meccanico e lo stato di una celtula fotosensibite.



Esistono quindi delle istruzioni che informano sulto stato della penna in quel momento.

1. *L'interruttore da solo.* PTRIG vale –1 se l'interruttore è premuto. PTRIG vale 0 se l'interruttore è rilasciato.

2. *La cellula da sola.* INPEN X,Y carica un numero di colonna grafica in X e un numero di riga in Y. La penna deve essere ben orientata e a meno di 10 cm dallo schermo. Il rosso e il nero non vengono rilevati. Quando la cellula non è attivata, carica –1 in X e in Y.

3. *L'interruttore e la cellula.* INPUTPEN X,Y funziona come INPEN dopo ta chiusura dell'interruttore, in genere appoggiando ta penna stessa sul video.

Programma

Visualizza le coordinate di un punto letto con la penna ottica.

```
10'PUNTO LETTO
20 CLS
30 DO
40 INPEN C,L
50 LOCATE 0,0:PRINT "COLONNA";C,"RIGA";L
60 LOOP
```

Commenti

Dopo aver lanciato l'esecuzione del programma, vengono visualizzate le coordinate –1 e –1. Poi, avvicinando la penna ottica allo schermo, i numeri scorrono rapidamente. Verificare che il numero della colonna vari da 0 a sinistra a 319 a destra (non è facile leggere i punti sul bordo dello schermo) e che il numero della riga vari da 0 a 199 verso il basso.
Se le coordinate continuano a mantenere i valori –1, significa che si è verificata una delle condizioni seguenti:
– si stanno leggendo punti al di fuori della zona utile,
– il colore dello sfondo è troppo scuro (rosso o nero),
– il televisore non invia luce sufficiente alla penna ottica; bisogna aumentare la luminosità del televisore.

Varlazionl

Si può visualizzare il punto mirato aggiungendo un'istruzione PSET. Lo schermo si riempie allora di piccoli puntini che corrispondono ad ogni punto letto. Per visualizzare unicamente i punti di coordinate determinate, e non dei punti letti, si può utilizzare l'istruzione PTRIG che funziona con l'interruttore della penna ottica. Per esempio, l'istruzione tF PTRtG THEN PSET (C,L) significa: se è stata premuta la penna ottica, visualizzare il punto C,L. Per capire meglio il funzionamento di questo test, vedere il capitolo 35.

Applicazioni

Le applicazioni della penna ottica sono numerosissime dal momento che questa permette all'utente di liberarsi dai vincoli della tastiera. Con la penna ottica si può scegliere un menu, designare con precisione il punto dello schermo senza dover spostare il cursore e senza danneggiare l'immagine.
Si pensi soltanto al modo in cui si sceglie una cella in una griglia di un carattere. Con la sola lastiera bisogna psssare da un sistema di coordinate, con tutti i problemi che questa operazione comporta.
L'aspetto più spettacolare della penna ottica è costituito dalla possibilità di poter disegnare sul video. Anche un utente che non ha ancora familiarizzato molto con il BASIC, dovrà ammettere che le tre istruzioni descritte precedentemente semplificano estremamente la realizzazione di questo progetto.
Per tracciare il disegno, si usa l'istruzione PSET (X,Y) e le istruzioni della tartaruga.

```
5 CLS
10 TURTLE 0
20 DO
30 DO
40 INPEN X,Y
50 IF X<>-1 THEN EXIT
60 SHOW O
70 LOOP
80 TRACE PTRIG:TURTLE O,X.Y:SHOW 1
90 LOOP
```

Commenti

40. Registra una posizione.
80. A seconda che l'interruttore sia premuto o meno, la tartaruga segna o meno lo spostamento.

Ecco un altro programma che simula un elastico. Inserire il programma e poi lanciarlo con RUN. Iniziare a mirare un qualsiasi punto dello schermo. Allontanare la penna ottica. Si vedrà un tratto elastico seguire il gesto della mano. Questo programma utilizza l'istruzione INPEN che funziona come INPUTPEN ma senza che la penna ottica sia appoggiata sullo schermo. E' possibile allontanare la punta della penna fino a 10 cm dal video. Oltre i 10 cm. la luminosità risulta troppo debole.

```
10 SCREEN 0,7,7:CLS
20 X0=160:Y0=100:X1=160:Y1=100
30 GOSUB 1000
40 X0=X2:Y0=Y2:GOTO 30
999 END
1000 '
1001 'SPOSTAMENTO ELASTICO
1002 '
1010 INPEN X2,Y2:IF X2<0 OR Y2<0 THEN 1010
1020 LINE (X0,Y0)-(X1-Y1),-8
1030 LINE (X0-Y0)-(X2-Y2),0
1040 X1=X2:Y1=Y2
1050 IF PTRIG=0 THEN 1010
1999 RETURN
```

Commenti

Il programma utilizza tre coppie di coordinate:
- X0 e Y0: estremità fissa dell'elastico
- X1 e Y1: estremità mobile precedente
- X2 e Y2: nuova estremità mobile

Il sottoprogramma 1000 controlla la posizione della penna ottica (1010), cancella il tratto elastico precedente (1020) attivando il colore dello sfondo (-8), traccia il nuovo elastico (1030) in nero, sostituisce il tratto precedente con quello nuovo (1040). Quando l'interruttore è premuto (1050), esce dal sottoprogramma. La riga 40 del programma principale ridefinisce le coordinate della nuova estremità fissa dell'elsstico.

Per coloro che possiedono un mouse.
Esistono tre istruzioni di programmazione del mouse che si rifanno a quelle della penna ottica e in cui PEN è sostituito da MOUSE.
INPEN → INMOUSE
INPUTPEN → INPUTMOUSE
PTRIG → MTRIG

Capitolo 31

Vettori e matrici

Le matrici permettono di memorizzare sotto uno stesso nome dati dello stesso tipo destinati per lo più allo stesso uso; il singolo dato è selezionabile per mezzo di uno o più numeri interi detti indici.

Vettori

Le matrici ad una dimensione (vettori) sono le più semplici: si indica il nome della variabile seguito tra parentesi da un indice che serve a identificare il singolo elemento. La dimensione del vettore rappresenta anche l'indice massimo. In un vettore A di tipo numerico di dimensione 10, si possono registrare dieci numeri chiamati rispettivamente A(1), A(2), A(3), fino ad A(10).
Il programmatore dovrà prevedere fin dall'inizio la dimensione massima del vettore e dovrà dichiararla nell'istruzione DIM, per esempio DIM A(10). E' indispensabile dimensionare il vettore in modo adeguato e sufficientemente ampio poichè l'istruzione DIM riserva dello spazio in memoria e una volta dimensionato non è più modificabile.

> E' vietato dimensionare più volte lo stesso vettore

E' possibile definire vettori numerici o di stringhe alfanumeriche.

Nota: E' possibile utilizzare vettori senza dichiararli. In questo caso il vatore massimo degli indici è 10. Questo significa che ad un vettore non dimensionato da DIM vengono riservati per default undici spazi (da 0 a 10). Si consiglia, per una maggiore chiarezza del programma, di dimensionare sempre i vettori utilizzati. Di seguito viene riportato un programma che genera un vettore di numeri interi casuali compresi fra 0 e 100, e un vettore di caratteri alfanumerici casuali compresi fra A e Z. Per seguire lo svolgimento del lavoro, viene effettuata la stampa degli elementi del vettore.

```
10 REM VETTORI ALEATORI
20 DIM A(10),B$(10)
30 FOR I=1 TO 10
40 A(I)=INT(RND*100)
50 B$(I)=CHR$(65+INT(RND*26))
60 NEXT I
```

```
RUN
OK
PRINT A(4),B$(7)
4          C
```

Attenzione: il nome di un vettore stringa deve terminare con $, e la sua dimensione deve essere dichiarata.

Dimensioni varie

Le matrici possono avere varie dimensioni, ma praticamente vengono utilizzate solo matrici unidimensionali (vettori) e bidimensionali. Queste ultime sono interessanti per i giochi su scacchiera, molto praticati in microinformatica a causa della bidimensionalità del video.
Di seguito viene riportato un breve programma che calcola il prodotto di due matrici i cui elementi sono numeri interi casuali. Utilizzando le istruzioni di inserimento (INPUT) lo si può rendere un comodo programma di utilità (vedere capitolo 28).

Programma

```
10 REM PRODOTTO DI MATRICI
20 N=3:P=4:Q=5
30 DIM A(N,P),B(P,Q),C(N,Q)
40 PRINT"MATRICE A"
50 FOR I=1 TO N:FOR J=1 TO P
60 A(I,J)=5-INT(RND*10):PRINT A(I,J);
70 NEXT J:PRINT:NEXT I
80 PRINT"MATRICE B"
90 FOR I=1 TO P:FOR J=1 TO Q
100 B(I,J)=5-INT(RND*10):PRINT B(I,J);
110 NEXT J:PRINT:NEXT I
120 REM PRODOTTO
130 PRINT"MATRICE AXB"
140 FOR I=1 TO N:FOR J=1 TO Q
150 FOR K=1 TO P
160 C(I,J)=C(I,J)+A(I,K)*B(K,J)
170 NEXT K:PRINT C(I,J);
180 NEXT J:PRINT:NEXT I
200 END
```

```
MATRICE A
0  0  5  1
4 -1 -4  1
5  5  3  0

MATRICE B
 4  5 -4  5  3
-4  1  3  3 -3
 3  3  0 -4  5
-2 -3  0  5  0

MATRICE AXB
13  12  0 -15  25
6  4 -19  38  -5
9  39  -5  28  15
```

Commenti

I cicli nidificati 50–70 definiscono e stampano la matrice A. I cicli nidificati 90–100 definiscono e stampano la matrice B. Fra le righe 140 e 180 sono presenti almeno tre cicli nidificati. Quelli in I e J generano la matrice prodotto. Il ciclo in K contiene il calcolo dei termini del prodotto. La formula è rintracciabile alla riga 160. Le matrici sono molto usate in informatica. In effetti permettono di operare sotto un unico nome su una quantità rilevante di variabili. Si noterà che tutte le variabili inserite in una matrice sono dello stesso tipo. Non è possibile all'interno della stessa matrice avere delle variabili a singola precisione e a doppia precisione, e tantomeno delle variabili stringa. Questa limitazione dipende dal metodo di memorizzazione usato dal calcolatore che esige che tutti gli elementi della matrice occupino lo stesso numero di byte.

La particolarità di una matrice sta anche nel metodo di accesso ai suoi elementi. Nel caso del PC 128, non esiste differenza nei tempi di accesso al primo elemento della matrice o all'ultimo (sia esso anche il ventottesimo). Diverso peso avrà la cosa nel momento in cui si affronterà il metodo di accesso alle informazioni contenute in un file sequenziale (vedere capitoli 46–47). Le matrici sono residenti in memoria centrale, ecco perchè sono così agevolmente accessibili, ma tendono ad occupare ampi spazi di memoria. Inserire molte informazioni in una matrice può portare al blocco di un programma per mancanza di spazio disponibile.

Da quanto sopra emerge uno dei problemi fondamentali presenti in informatica: se è meglio privilegiare i tempi di accesso alle informazioni (e di conseguenza i tempi di lavoro) o gli spazi di memoria occupati, dato che le due cose non possono essere ottenute contemporaneamente.

Capitolo 32

I dati

Esistono tre differenti modi per memorizzare il numero 1917 sotto il nome di variabile CAPORETTO. Due sono già stati presentati nei capitoli precedenti. Il terzo (DATA–READ) viene introdotto in questo capitolo.

```
10 CAPORETTO=1917
20 PRINT CAPORETTO
RUN
1917
OK      1
```

```
10 DATA
20 READ CAPORETTO
30 PRINT CAPORETTO
RUN
1917
OK      3
```

```
10 INPUT CAPORETTO
20 PRINT CAPORETTO
RUN
? 1917
1917
OK      2
```

Commento

- Il primo metodo è il più breve e naturale.
- Il secondo metodo fa gestire tutto dall'utente, cosa che è sempre opportuno evitare.
- Il terzo metodo utilizza due nuove istruzioni DATA e READ di cui ora si vedrà il funzionamento.

Quando l'interprete BASIC riconosce una riga DATA, finge di ignorarla come se si trattasse di una REM. In un primo momento i numeri o le stringhe relative a DATA vengono ignorate.
Quando l'interprete riconosce una istruzione READ seguita obbligatoriamente dal nome di una variabile, l'interprete stesso legge il puntatore dei dati, legge il dato corrispondente e lo memorizza sotto il nome di variabile indicato nell'istruzione, poi incrementa il contatore dei dati in previsione di una successiva READ. Le righe DATA possono essere introdotte in qualsiasi punto del programma, prevedono un numero variabile di dati, di tutti i tipi, separati dalla virgola. In questo modo l'utente ha la possibilità di associare una gran quantità di dati ad un gran numero di variabili. Di seguito viene riportato un esempio di memorizzazione e stampa di una matrice numerica. Si può notare come la suddivisione dei DATA all'interno del programma ricalchi la struttura della matrice in modo da facilitare la correzione degli errori.

```
10 DIM A(5,3)
20 DATA 1,8,5.3,2
30 DATA 4,6,7.2,9
40 DATA 1,0,6.6,3
50 FOR J=1 TO 3:FOR I=1 TO 5
60 READ A(I,J):PRINT A(I,J);
70 NEXT I:PRINT:NEXT J
RUN
1  8  5  3  2
4  6  7  2  9
1  0  6  6  3
```

RESTORE

Questa istruzione ripristina il valore iniziale del puntatore dei dati di modo che la prima READ possa leggere il primo dato relativo al primo DATA. Viene usata quando all'interno del programma si desidera leggere più volte gli stessi dati.

Tipi di variabile

Una stessa riga DATA può prevedere variabili di tipo diverso a condizione che il comando di lettura tenga conto di queste differenze.

```
10 DATA CAPORETTO, 1917
20 READ B$,D
```

Fine lettura

Quando il numero di dati nell'istruzione DATA non è prevedibile a priori, si può prevedere in coda ai dati stessi, un dato supplementare fittizio che diventerà il segnale di fine lettura. Ecco un programma che calcola la media su un numero qualsiasi di dati presenti in DATA, con numerazione compresa tra 0 e 20. Il segnale di fine lettura è il numero 999.

```
10 REM MEDIA
20 DATA 11,14,13,8,10,5,999
30 N=0:T=0:RESTORE
35 DO
40 READ A
50 IF A>20 THEN EXIT
60 T=T+A:N=N+1
70 LOOP
80 PRINT"MEDIA:":T/N
100 END
```

Esempio di utilizzo:

1. Leggere trentun valori rappresentanti le differenti temperature rilevate a Milano nel mese di luglio.
2. Calcolare la temperatura minima e la massima del mese.
3. Tracciare la curva di evoluzione delle temperature (scala = 5 righe per 1 grado).

Per calcolare le minime e le massime del punto 2 si utilizzeranno le funzioni matematiche MIN e MAX.
MIN (A,B) = minore dei due numeri A e B
MAX (A,B) = maggiore dei due numeri A e B

Programma

```
10 'METEO
20 DIM TEMP(31)
30 CLS
40 FOR I=1 TO 31
50 READ TEMP(I)
60 NEXT I
99 DATA 25,27,26,30,29,33,28,32,31,33,29,28,31,30,26,29,
   32,30,28,25,28,26,31,30,27,29,28,29,27,26,29
100 'MINIMA-MASSIMA
110 TMIN=100:TMAX=-50
120 FOR I=1 TO 31
130 TMIN=MIN(TEMP(I),TMIN)
140 TMAX=MAX(TEMP(I),TMAX)
150 NEXT I
160 PRINT"MASSIMA";TMAX
170 PRINT"MINIMA";TMIN
200 'CURVA
210 PSET(0,200-TEMP(I)*5)
220 FOR I=2 TO 31
230 LINE-((I-1)*10,200-TEMP(1)*5)
240 NEXT I
999 END
```

Commenti

- L'istruzione DATA è usata per l'inizializzazione di una tabella: TEMP.
- La temperatura minima è definita TMIN e la massima TMAX; ricordarsi che i nomi delle variabili non possono cominciare per MIN e MAX in quanto queste sono parole chiave.
- 110. Si inizializzano queste due variabili con valori improbabili (100 e -50).
- 130 e 140. Si ricicla sui numeri interessati controllando ogni volta il minore ed il maggiore.

– 130 e 140. Si ricicla sui numeri interessati controllando ogni volta il minore ed il maggiore.

Capitolo 33

Ordinamento dei numeri (SORT)

L'ordinamento dei numeri è da sempre uno dei maggiori problemi di chi si occupa di informatica. Ordinare una quantità ridoita di numeri è relativamente semplice: il tempo nscesssrio per farlo è sempre poco. Diverso è quando si traita di migliaia di numeri. Ecco allora che diventa interessante trovare soluzioni che riducano al massimo il numero di operazioni da effettuare.
In due pagine non si potrà affrontare adeguatamente il problema. Verranno però suggeriti due metodi etementari per l'ordinamento di due gruppi di numeri.

Metodo del minimo

Si esaminano tutti i numeri e si sceglie il più piccolo. Si ricomincia con quelli che restano finchè non ne rimangono più.
Il programma ordina N numeri casuali compresi tra 0 e 100. Nell'esempio si è fissato N a 10, ma nulla impedisce di cambiarlo.

```
10 REM ORDINAMENTO DI NUMERI
20 N=10:DIM A(N)
30 FOR I=1 TO N
40 A(I)=INT(RND*100):PRINT A(I);
50 NEXT I:PRINT
60 REM INIZIO
70 FOR D=1 TO N-1
80 X=D:INF=A(D)
90 FOR K=D+1 TO N
100 IF A(K)<INF THEN X=K:INF=A(K)
110 NEXT K
120 SWAP A(X),A(D)
130 NEXT D
140 REM STAMPA DELLA SERIE NUMERICA ORDINATA
150 FOR I=1 TO N:PRINT A(I);:NEXT I
200 END
RUN
59 51 1 42 16 62 90 43 4 9
1 4 9 16 42 43 51 59 62 90
```

Commenti

Sono presenti due cicli nidificati. Nel ciclo interno si cerca al di là di A(D) il numero più piccolo (INF). Notare il relativo indice (X). All'uscita dal ciclo si permuta A(D) con A(K) (riga 120), poi si incrementa D di 1. Questo tipo di ordinamento è molto lungo: è facile rendersene conto introducendo N=100.

Ordinamento a bolle (bubble sort)

Qui il procedimento è molto differente e molto più efficace. Si considerano i numeri nel loro ordine e a due a due. Se all'interno di ogni coppia i due numeri non rispettano l'ordine voluto, i numeri verranno scambiati di posizione. Si procede così per tutte le coppie. Quando queste sono terminate, anche l'ordinamento è terminato. Questo programma utilizza una flag (bolla) che risale lentamente verso l'alto della serie di numeri.

```
10 REM ORDINAMENTO A BOLLE
20 N=10.DIM A(N)
30 FOR I=1 TO N
40 A(I)=INT(RND*100):PRINT A(I);
50 NEXT I
60 REM INIZIO
70 FLAG=0:FOR I=1 TO N-1
80 IF A(I)<=A(I+1)THEN GOTO 100
90 SWAP A(I),A(I+1):FLAG=1
100 NEXT I
110 IF FLAG=1 THEN GOTO 70
120 REM FINE
130 PRINT:FOR I=1 TO N
140 PRINT A(I);:NEXT I
RUN
66  40  59  94  27  65  3  93  38  57
3  27  38  40  57  59  65  66  93  94
```

Commenti

Nella riga 70 il flag viene azzerato. Ogni numero della serie viene letto e confrontato con il successivo. Se sono ordinati (80) si prosegue, altrimenti vengono scambiati (90) e si mette il flag a 1.
Se è stato effettuato almeno uno scambio, il flag è a 1 e bisogna ricominciare. Quando tutta la serie viene scorsa senza che si verifichino scambi (quindi senza che il flag venga messo a 1), l'ordinamento è terminato. Lo si può verificare effettuando la stampa della serie numerica ordinata.

Ordinamento alfabetico

L'ordinamento di dati alfabetici potrebbe essere difficoltoso se non si potesse contare sui codici ASCII che considerano ogni lettera maiuscola dell'alfabeto come un numero intero compreso tra 65 e 90. Come i numeri, le strighe possono essere confrontate tramite gli operatori relazionali <, > o =. Dire che una stringa A$ è minore di una stringa B$, significa che A$ precede B$ nell'ordine ASCII.
Esempio:
"ab" < "bc"
"rosso" < "verde"
"abc" < "bc"

I metodi di ordinamento precedentemente esaminati possono quindi essere applicati a tabelle di stringhe alfanumeriche. E' sufficiente inserire il simbolo $ alla fine dei nomi delle variabili. Ovviamente la definizione della tabella non è semplice come i numeri: ogni elemento dovrà essere dichiarato all'inizio del programma. Una difficoltà può derivare dalla presenza di lettere minuscole, codificate nella tabella ASCII da 97 a 122. Ne deriva che "B" < "a". Poichè i confronti su stringhe si basano sui codici ASCII, l'unico modo per aggirare l'ostacolo è quello di prevedere una routine che trasformi le lettere minuscole in lettere maiuscole.

```
10 A$="Ali-Baba":LA=LEN(A$):AA$=""
20 FOR K=1 TO LA
30 CL=ASC(MID$(A$,K,1))
40 IF CL>96 AND CL<123 THEN CL=CL-32
50 AA$=AA$+CHR$(CL)
60 NEXT K
70 PRINT A$:PRINT AA$
80 END
```

Commenti

10. AA$ è una stringa nulla nella quale si copierà A$, carattere per carattere, cambiando le minuscole in maiuscole.
30. CL è il codice ASCII del K-esimo carattere di A$.
40. Se CL contiene un codice di lettera minuscola, gli si toglierà 32 per portarlo al codice dell'equivalente lettera maiuscola: il codice di "A" è 65, quello di "a" è 97; 97-65=32.
50. Si inseriscono i caratteri in AA$.
70. Si noterà che in stampa solo le minuscole sono state trasformate in maiuscole e non viceversa.

Capitolo 34

Applicazioni matematiche

Funzioni matematiche

Per coloro che sono interessati anche marginalmente alle applicazioni matematiche del PC 128, di seguito viene riportata la lista delle funzioni matematiche con il relativo formato BASIC.

Funzione	Simbolo	Esempio	BASIC
Elevamento a potenza	↑	$y=x^2$	Y=↑2
Radice quadrata	SQR	$y=\sqrt{x}$	Y=SQR(X)
Logaritmo neperiano	LOG	y=Log(x)	Y=LOG(X)
Esponenziale	EXP	$y=e^x$	Y=EXP(X)
Seno (angolo in radianti)	SIN	y=sinx	Y=SIN(X)
Coseno (angolo in radianti)	COS	y=cosx	Y=COS(X)
Tangente (angolo in radianti)	TAN	y=tgx	Y=TAN(X)
Arcotangente (risultato in radianti)	ATN	x=Arc tg(y)	X=ATN(Y)
Parte Intera	INT	n=E(x)	N=INT(X)
Valore assoluto	ABS	y=|x|	Y=ABS(X)
Valore del segno	SGN	n=sign(x)	N=SGN(X)
Troncamento dei decimali	FIX	x_1=tronc(x)	X1=FIX(X)
Minimo	MIN	z=min(x,y)	Z=MIN(X,Y)
Massimo	MAX	z=max(x,y)	Z=MAX(X,Y)

Esempio

L'operatore di divisione / effettua la divisione dando anche gli eventuali decimali. Per fare una divisione intera (divisione euclidea) si può usare l'istruzione MOD che dà il resto della divisione. Esempio:

```
PRINT 256 MOD 7
4
OK
```

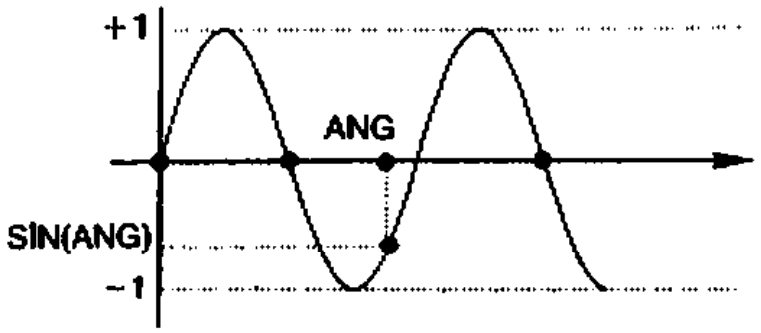
```
256 | 7
----|---
 46 | 36
[4]
```

```
10 CLS:SCREEN 3,0,0
20 FOR X=0 TO 319
30 C=X MOD 16
40 LINE(0,0)-(X,199),C
50 NEXT X
60 END
```

Vengono tracciate 320 linee in uno dei sedici colori disponibili. L'argomento verrà ripreso nel capitolo 36.

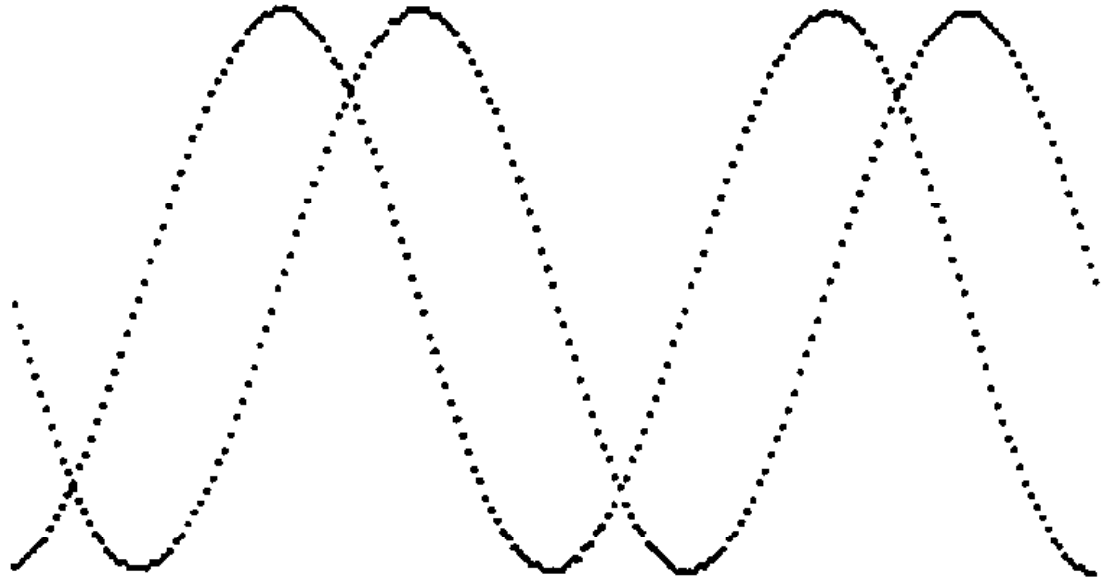
Sinusoide

Se la variabile ANG rappresenta un angolo espresso in radianti, SIN(ANG) è un numero variabile in modo sinusoidale tra –1 e +1.



In questo caso i valori di ANG e SIN(ANG) sullo schermo sono impercettibili, dato che l'unità di misura è il pixel. E' conveniente utilizzare un coefficiente moltiplicatore, una scala, che permetta una visualizzazione più o meno lunga e più o meno larga.
Di seguito viene riportato un programma che utilizza solo l'istruzione grafica e visuatizza due archi di sinusoide nelle dimensioni compatibili con lo schermo ad alta risoluzione. E' stata inoltre aggiunta una curva della funzione coseno.

```
10 CLS
20 PI=3.14:EX=15:EY=50
30 FOR ANG=0 TO 4*PI STEP 1/EX
40 PSET(ANG*EX,SIN(ANG)*EY+EY
50 PSET(ANG*EX,COS(ANG)*EY+EY
60 NEXT ANG
```



Operatori logici

La logica è un ramo della matematica i cui elementi di base sono vero e falso. Il matematico Boole già nel secolo scorso aveva sostituito vero e falso con dei numeri (0 e 1) a partire dai quali ha costruito un'algebra detta appunto "booleana". L'informatica ha naturalmente sfruttato l'algebra booleana poichè essa era in grado di dare risposte soddisfacenti ad alcuni suoi problemi.

L'operatore AND

AND significa E. Se A e B sono due proposizioni, la proposizione "A E B" è vera quando A e B sono entrambe verificate. Si costruisce così una tavola di verità dell'operatore AND.

AND	F	V
F	F	F
V	F	V

F AND F = F
F AND V = F
V AND F = F
V AND V = V

Capitolo 35

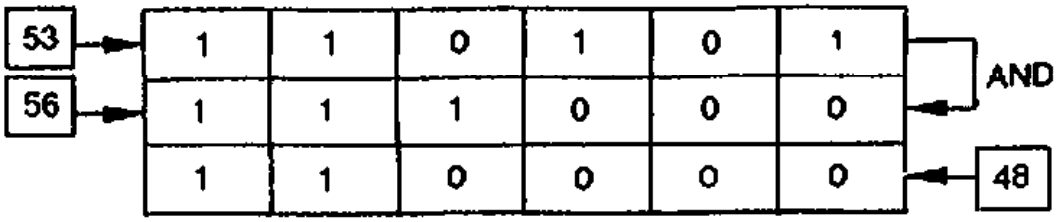
Operatori logici (seguito)

Il PC 128 rappresenta Vero con –1 e Falso con 0. Si può applicare l'operatore logico AND su due numeri qualsiasi.

Un'operazione AND tra due numeri qualunque si fa in modo molto semplice. Si può provare innanzitutto in modalità diretta.

```
PRINT –1 AND 0
0
OK
PRINT –1 AND –1
–1
OK
PRINT 53 AND 56
48
OK
```

I due primi calcoli confermano le regole operative di AND. Il terzo sembra contravvenire a queste regole. Osservando però la rappresentazione binaria dei numeri 53 e 56, ci si renderà conto che l'operazione binaria AND è stata effettuata su ciascuna cifra dei due numeri (vedere appendice 3); viene quindi confermata la sua coerenza alle regole operative di AND.



Altre operazioni logiche

Esistono altre operazioni logiche costruite su regole diverse ma che obbediscono comunque alle stesse leggi di generalizzazione dei numeri diversi da 0 e +1.

A	B	A AND B	A OR B	A IMP B	A XOR B	A EQV B
0	0	0	0	–1	0	–1
0	–1	0	–1	–1	–1	0
–1	0	0	–1	0	–1	0
–1	–1	–1	–1	–1	0	–1

AND : e
OR : o
IMP : implica
XOR : o esclusivo
EQV : equivalente a

Test logici

Le operazioni precedenti possono essere utilizzate nei test e per semplificare la programmazione.

Espressioni

In BASIC un'espressione è una serie di caratteri in cui possono essere presenti delle costanti, dei nomi di variabile e dei segni di operazione. L'interprete valuta queste espressioni e l'istruzione PRINT visualizza il risultato della valutazione.

Questa nozione di espressione può essere estesa a scritture del tipo A=B o A>B. In un test che utilizza simili espressioni, l'interprete provvede ad una valutazione numerica che permetterà di decidere se il test è verificato o meno. A vero viene associato un numero, a falso un altro.

VERO	FALSO
–1	0

```
A=5:B=7
PRINT A=B
0
OK
PRINT A<B
–1
OK
PRINT A>=B
0
OK
```

La valutazione numerica delle espressioni che usano una relazione di confronto viene fatta dall'interprete BASIC senza che la cosa sia evidente per l'utente: è però facile immaginare che venga fatto confrontando i contenuti delle variabili, bit per bit.

A partire da ciò l'utilizzo degli operatori logici (AND, OR, NOT) diventa un problema di calcoli tra 0 e –1.

```
A=5:B=6:C=7
PRINT A<B AND B<C
-1
OK
PRINT A<B AND B>C
0
OK
PRINT A<B OR B>C
-1
OK
PRINT A>B OR B>C
0
OK
```

Test logici

L'utente potrà quindi utilizzare gli operatori AND, OR, NOT, ecc. per verificare più condizioni all'interno di un unico IF. A priori, non esiste alcuna limitazione nella lunghezza delle espressioni e nel numero di operatori, se non quella che limita il numero di caratteri di una riga di istruzioni.

Assegnamenti logici

Il BASIC è estremamente esigente per ciò che concerne la sintassi. Una punteggiatura assente o introdotta erroneamente, la minima regola non rispettata, provocano messaggi d'errore e l'interruzione del programma. Riguardo all'assegnazione (=): un'istruzione come A=3 è corretta, ma 3=A non lo è. Per quanto riguarda A=B=C, questa è un'espressione corretta, contrariamente a quello che si potrebbe credere ad uno sguardo superficiale. Il segno=dopo A significa assegnazione. L'espressione a destra del segno deve essere valutata e il suo valore deve essere messo nella variabile A. Questa espressione è B=C, il suo valore è 0 o –1 a seconda che B sia o meno uguate a C. Il valore di A darà quindi una valutazione dell'uguaglianza di B e C.

```
B=5:C=6
A=B=C:PRINT A
0
OK
B=10:C=10
A=B=C:PRINT A
-1
OK
```

Capitolo 36

Dimensioni della memoria

La CPU di ogni microelaboratore ha una propria capacità di memoria misurata in Kbyte. Ogni istruzione di programma e ogni variabile occupa dello spazio in memoria. Via via che un programma viene ampliato, la memoria disponibite diminuisce. Si può arrivare al punto in cui non c'è più memoria disponibile.
Il comando PRINT FRE(0) può segnalare in qualsiasi momento la capacità di memoria disponibile.
Nell'esempio seguente viene rilevato it numero di byte disponibiti inizialmente, poi il numero dei byte dopo la scrittura di un programma senza commenti, quello dopo l'inssrimento dei commenti ed infine dopo aver dato il comando NEW che cancella l'intero programma.

```
PRINT FRE(0)
110120
OK
20 A=19:B=38
30 C=A+B
40 PRINT C
RUN
57
OK
PRINT FRE(0)
110089
OK
10 REM SOMMA
35 REM VISUALIZZ SOMMA
RUN
57
OK
PRINT FRE(0)
110055
OK
NEW
PRINT FRE(0)
110120
OK
```

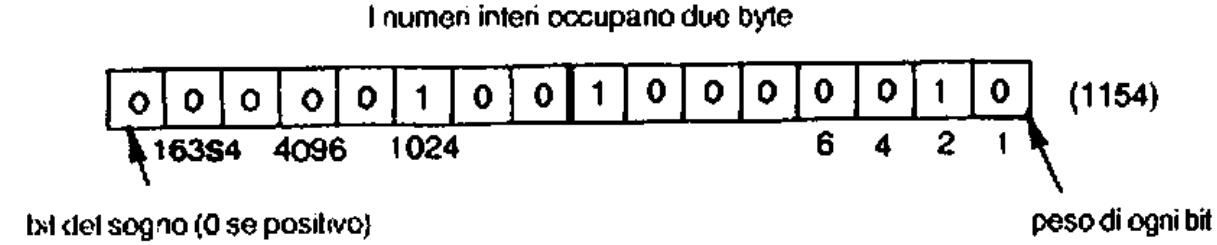
Conclusioni

1. All'inizio la disponibilità è di 110120 byte.
2. Dopo l'inserimento del programma, senza commenti, sono disponibili 110089 byte. Gli altri 31 byte sono stati utilizzati dal programma.
3. Dopo l'inserimento dei commenti al programma, sono disponibili 110055 byte. Gli altri 65 byte sono stati utilizzati dal programma (31 byte) e dai commenti (34 byte).
4. Dopo NEW, sono di nuovo disponibili 110120 byte.

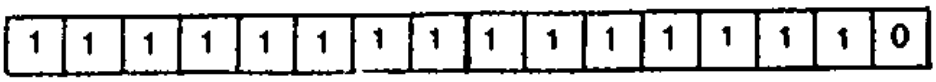
Si può concludere che anche i commenti occupano spazio. Da qui la necessità di usare in modo intelligente le REM, evitando di usare REM inutili, con commenti insignificanti o troppo lunghi. Usare quindi REM corte, nei punti in cui sono effettivamente necessarie.

Memoria occupeta da variabili

Per memorizzare una variabile, il BASIC occupa un certo numero di byte: per il nome della variabile, per il tipo, per l'indirizzo e per il valore.



Quindici bit permettono di codificare un numero compreso tra 0 e 32767 (cioè $2^{15}-1$). Il sedicesimo bit serve per il segno. Il numero che è stato appena codificato corrisponde a 1154 (1024+128+2); il bit del segno (positivo) è 0. Se il bit del segno è 1, il numero rappresentato è negativo; il suo valore assoluto si ottiene cambiando tutti i bit (0 in 1 e 1 in 0) e aggiungendo 1 al risuttato (questa operazione viene detta complemento a 2).
Ecco una rappresentazione di –2.



I numeri reali occupano quattro byte e si rappresentano nel formato:

La mantissa viene memorizzata in tre byte dove uno dei bit è usato dal segno. Si ottengono così le sette cifre decimali significative, a precisione singola. L'esponente occupa un byte, di cui un bit per il segno, e può arrivare fino a 127. Il limite di rappresentazione di un intero positivo è dunque di 2^{127} e di circa 10^{38}.

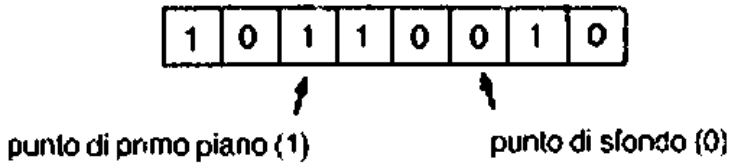
Le stringhe alfanumeriche occupano un byte per ogni carattere, più tre byte per la gestione della stringa.

Per le matrici, ogni elemento occupa il posto previsto dal suo tipo; bisogna aggiungere alcuni byte per te caratteristiche della matrice.

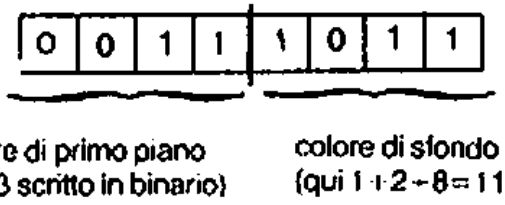
Memoria di schermo

La memoria di schermo può essere modificata in qualsiasi momento sia dall'utente sia dal calcolatore stesso. Il video è composto da 64000 punti o pixel (200x320). Il video riserva un'area di memoria per ogni pixel di schermo in modo tale che si può immaginare la memoria del video come una copia del video stesso, dove i pixel accesi sono sostituiti da 1 e i pixel spenti da 0.
Se ognuno dei 64000 punti potesse essere acceso in un colore previsto nella gamma dei sedici disponibili, occorrerebbero 32 Kbyte per memorizzare tutte queste informazioni. Per risparmiare spazio in memoria, si è portata questa memorizzazione a 16 Kbyte. In questo caso non è possibile trattare ogni punto separatamente dagli attri: la conseguenza è l'"effetto sbavatura" già rilevata negli esempi grafici. Lo schermo è dunque memorizzato sotto forma di segmenti di otto punti ciascuno.

Per ogni segmento:
– Un byte indica l'appartenenza di ogni punto alla categoria "primo piano" o alla categoria "sfondo".



– Un byte, diviso in 2x4, definisce il colore dello sfondo e il colore del primo piano.



Su 4 bit, si possono individuare 2^4, cioè 16 colori.

Capitolo 37

ASCII

Nel capitolo precedente, si è visto come un carattere inserito in memoria occupi un byte. Un byte corrisponde a otto bit, ed ha quindi la possibilità di codificare $2^8 = 256$ caratteri differenti. In effetti sugli otto bit disponibili solo sette sono effettivamente utilizzabili per la codifica. Ciò rende disponibili $2^7 = 128$ caratteri, codificati dalla tabella ASCII di cui per il momento si conoscono solo le lettere alfabetiche (vedere capitolo 10):

A è codificato con 65 in decimale, e con 01000001

Z è codificato con 90 in decimale, e con 01011010 in un byte

E' possibile esaminare la codifica ASCII grazie a CHR$. Si consiglia di effettuare questo esame in due tempi.
- Primo: non scendere al di sotto del codice 32.

```
FOR I=32 TO 127:PRINT CHR$(I):NEXT I
```

Si vedranno apparire tutti i caratteri rappresentabili, disponibili sul PC 128.

- Secondo: scendendo al di sotto del codice 32 il calcolatore si può bloccare. In effetti i codici da 0 a 31 sono riservati al calcolatore. Vi si trovano i codici di ritorno carrello, dei tasti di spostamento del cursore, del lampeggiamento e del segnale acustico, ecc. Questi caratteri non sono rappresentabili su video e possono provocare un blocco del calcolatore. Nel capitolo 43 verrà illustrata un'applicazione dell'utilizzo dei codici ASCII per gestire lo spostamento del cursore.

PEEK e POKE

Il lavoro di un programmatore o di un utente di microelaboratore consiste nella lettura e modifica dei contenuti di determinate celle di memoria, nella memorizzazione, nel richiamo di informazioni.

Memorizzazione

Nel BASIC è prevista un'istruzione molto semplice che permette di inserire in una calla di memoria un numero intero compreso tra 0 e 255. Per poterlo fare è necessario sapere due cose: cosa inserire e dove inserirlo cioè quale byte e a quale indirizzo. L'ordine di memorizzazione è POKE seguito dall'indirizzo e dal numero da memorizzare.

POKE indirizzo, numero da memorizzare

```
POKE 17400,178
OK
```

Questo OK assicura che la memorizzazione è andata a buon fine, ma poiché non è stato dato alcun ordine di visualizzazione, nulla è stato visualizzato.

Richiamo

Per sapere se l'ordine dato è stato eseguito correttamente, si può usare l'istruzione PEEK che va a leggere la cella di memoria specificata. Per avere la visualizzazione della cella di memoria all'indirizzo 17400, occorre dare ordine di stampa (PRINT) del contenuto (PEEK) dell'indirizzo 17400.

```
PRINT PEEK(17400)
178
OK
```

PEEK legge senza modificare il contenuto della cella di memoria interessata. Per poter apportare modifiche occorre utilizzare l'istruzione POKE.

I due esempi seguenti permettono di cogliere la differenza tra memoria disponibile e memoria protetta.

Primo esempio

```
PRINT PEEK(41000)
76
OK
POKE 1000,42
OK
PRINT PEEK(1000)
76
OK
```

Riassumendo:

Nell'area di memoria 41000, è stato trovato il numero 76. E' stato fatto un tentativo di sostituirlo con 42, usando POKE. Però l'istruzione POKE è stata ignorata e il contenuto è rimasto inalterato. Bisogna quindi dedurre che quella cella di memoria è protetta e quindi non è modificabile: il suo contenuto non può essere alterato dall'utente.

Secondo esempio

```
PRINT PEEK(17400)
0
OK
POKE 17400,42
OK
PRINT PEEK(17400)
42
OK
```

17400 è sicuramente una cella di memoria disponibile. Inizialmente tutte le celle di memoria disponibili, fatta eccezione per la memoria dello schermo, contengono 0. Sarà l'utente che le riempirà progressivamente.

Buona parte della programmazione potrebbe essere fatta usando PEEK e POKE, almeno per quanto riguarda la memorizzazione e il richiamo delle informazioni. Fortunatamente il BASIC offre maggiori possibilità. Alcune delle facilitazioni offerte dal BASIC consistono nel lasciar decidere al calcolatore stesso in quali celle di memoria inserire le informazioni.

Attenzione

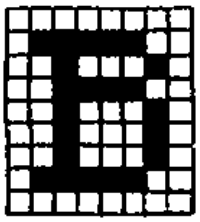
Inserendo delle istruzioni POKE a caso nella memoria del PC 128 ci si espone ad un rischio: quello di bloccare il PC 128. Premendo RESET si ripristina la situazione precedente.

Capitolo 38

Visualizzazione di un carattere

Quando l'utente preme un tasto della tastiera, sul video appare il carattere corrispondente. E' interessante stabilire cosa accade dal momento in cui si preme il tasto a quello dell'apparizione del carattere ad esso associato su video. Premendo ad esempio la lettera "B", è possibile visualizzarla in diversi modi, ma il principio è sempre lo stesso, cioè la selezione di punti scelti all'interno di una griglia (o matrice di punti).

visualizzazione di B
su una matrice 8x8



E' possibile osservare che la cornice esterna della matrice non viene mai riempita. Questa precauzione tende a permettere una buona visualizzazione dei caratteri adiacenti: due caratteri vicini sono sempre separati da almeno due linee o due colonne di pixel. Per capire i principi che governano la trasmissione dell'informazione tra tastiera e video è sufficiente sostituire nella matrice del carattere B i punti accesi con degli 1 e i punti spenti con degli 0.

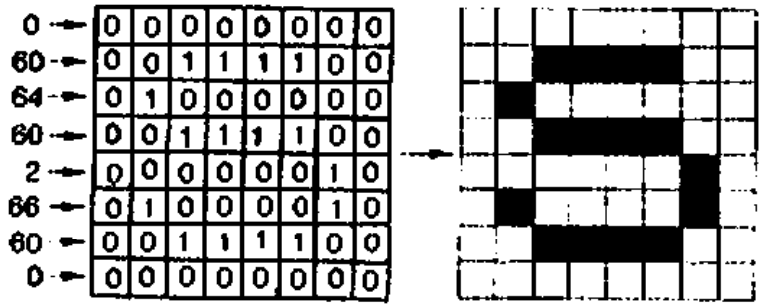
rappresentazione binaria
di una matrice 8x8 per la
visualizzazione di una B

0	0	0	0	0	0	0	0
0	1	1	1	1	1	0	0
0	0	1	0	0	0	1	0
0	0	1	1	1	1	0	0
0	0	1	0	0	0	1	0
0	0	1	0	0	0	1	0
0	1	1	1	1	1	0	0
0	0	0	0	0	0	0	0

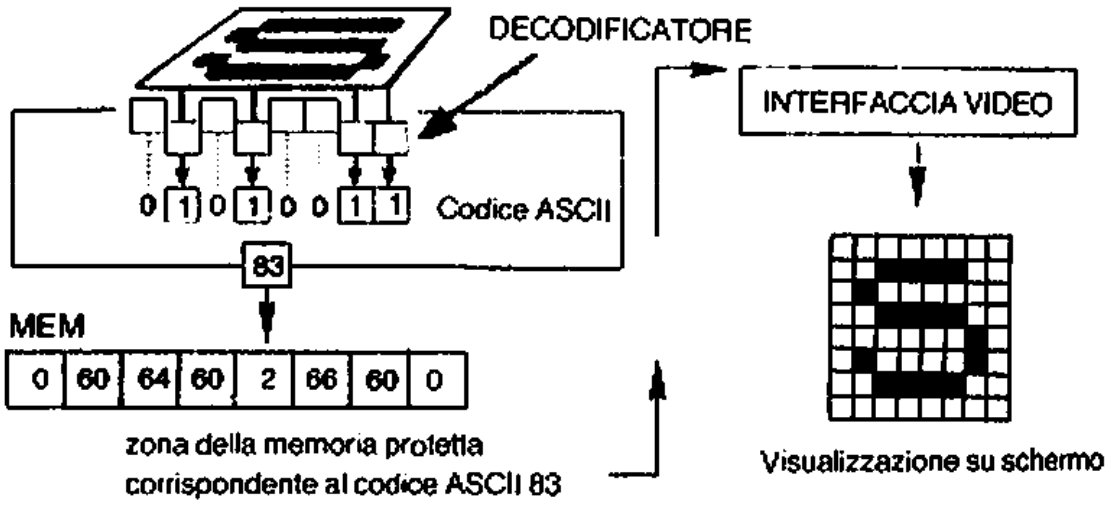
Ogni riga della matrice può essere considerata come un byte e, di conseguenza, l'intera matrice è codificata come un insieme di otto byte.
Per il carattere B avremo, nell'ordine, i seguenti byte:
00000000,01111100,00100010,00111100,00100010,00100010,01111100,
00000000.
E' possibile dare anche una rappresentazione decimale di ognuno di questi byte: la relativa codifica sarà un insieme di otto numeri compresi tra 0 e 255. La visualizzazione del carattere sulla matrice determina in modo univoco l'insieme degli otto numeri e la conoscenza degli otto numeri permette di rintracciare esattamente la matrice stessa.

Esempio:

Troviamo ora il carattere codificato con i numeri: 0,60,64,60,2,66, 60,0. Ognuno di questi numeri ha una sola possibile trascrizione binaria che permette di rintracciare il carattere.



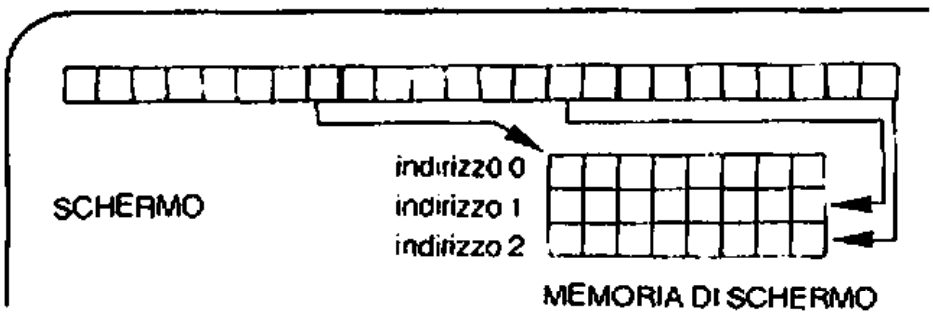
Per permettere all'elaboratore una corretta visualizzazione di tutti i caratteri presenti sulla tastiera, occorre ottimizzare nelle celle di memoria protette a gruppi di otto i byte corrispondenti a ciascun carattere. Per recuperare la zona di memoria protetta interessata dal tasto premuto, esiste un decodificatore ASCII dei tasti. Lo schema seguente riassume l'intervento da svolgere. Viene simulato un decodificatore meccanico per semplificare la situazione ma il lettore deve sapere che in realtà si tratta di un meccanismo elettronico. Il codice ASCII del carattere S è 83.



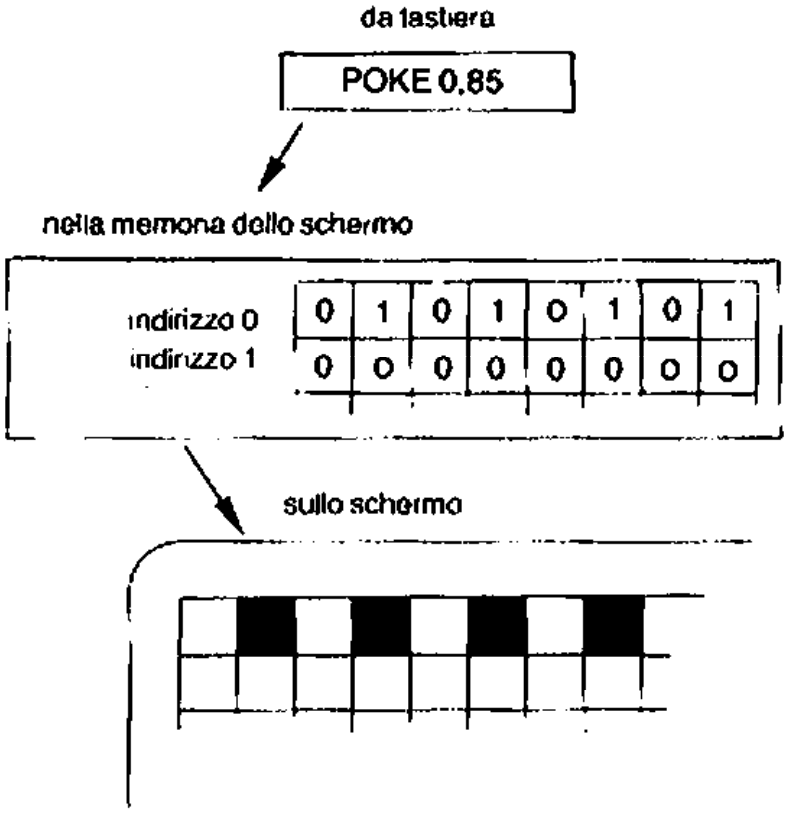
Come agire sulla memoria di schermo

La memoria di schermo è una parte di memoria disponibile. Nulla impedisce all'utente di modificare il contenuto di determinate celle di memoria per vederne i risul-

tati sullo schermo. E' sufficiente conoscere gli indirizzi. Per il PC 128 il primo indirizzo è 0, l'ultimo 7999. Il primo byte corrisponde ai primi otto punti orizzontali in alto a sinistra, il secondo (indirizzo 1) agli otto successivi, ecc. Quando si arriva alla fine di una riga, si passa semplicemente all'inizio della successiva.



Il principio di corrispondenza tra i bit della memoria e i pixel dello schermo è molto semplice: 0 per un pixel spento, 1 per un pixel acceso. Inserire il numero 85 nella cella di memoria con indirizzo 0. In binario 85 si esprime 01010101.



Per conoscere la locazione di una cella di memoria con indirizzo X sullo schermo, digitare l'istruzione:

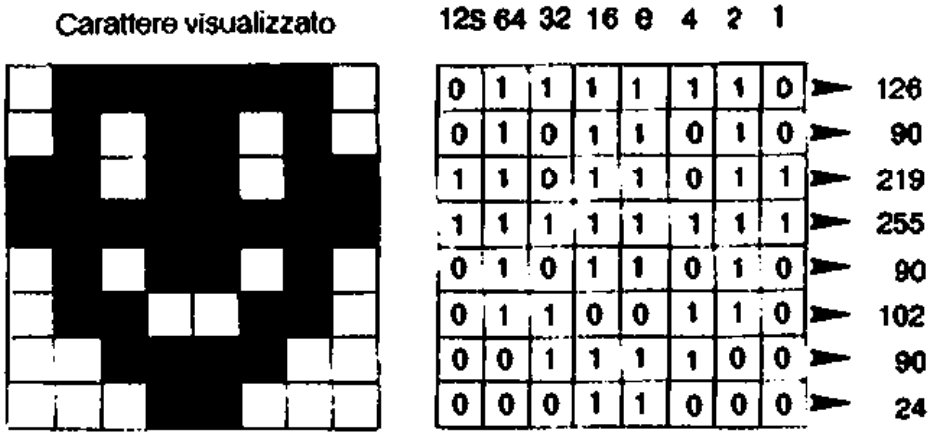
POKE X,255

ed osservare in quale posizione dello schermo appare la barra di otto pixel. Per cancellare i punti accesi, inserire un'istruzione POKE per la cella di memoria interessata, prevedendo l'inserimento di 0.

Capitolo 39

Definizione di caratteri

I caratteri correnti (quelli della tabella dei codici ASCII) sono inseriti nella memoria protetta e non sono modificabili. Però l'utente dispone di un'area di memoria nella quale può memorizzare propri caratteri. L'istruzione CLEAR gli permette di cancellare una determinata cella di memoria per la creazione dei propri caratteri, mentre l'istruzione DEFGR$ gli darà la possibilità di riempire questa cella con i dati desiderati.
Come per i caratteri standard si utilizzerà una matrice 8x8 per rappresentare un carattere con gli elementi a 1 in corrispondenza dei pixel illuminati.



La memorizzazione di questo carattere speciale si effettua in tre tempi.

CLEAR,,1
Questa istruzione informa il calcolatore che gli verrà proposto un nuovo carattere (CLEAR,,5 per 5 caratteri, ecc.).

DEFGR$(0)=126,90,219,255.90,102,60.24
Ecco l'istruzione che definisce numericamente il disegno sopra illustrato.

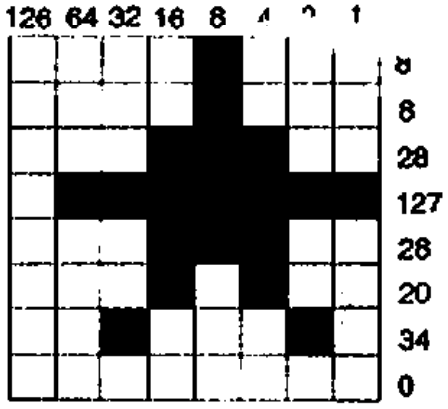
- DEF per definire,
- GR per grafico,
- $ per carattere alfanumerico,
- 0 è il numero del disegno.

Indipendentemente dal numero di caratteri definiti, la numerazione deve obbligatoriamente cominciare da 0 per proseguire secondo l'ordine dei numeri naturali interi fino ad un massimo di 127.
Sono facilmente riconoscibili, dopo il segno =, gli otto numeri decimali che costituiscono la codifica numerica del disegno sopra illustrato.

PRINT GR$(0)

Questa istruzione provocherà la visualizzazione del carattere GR$(0). E' possibile, come per tutti gli altri caratteri, definire la sua posizione con LOCATE, la sua dimensione con ATTRB ed il suo colore con COLOR. E' anche possibile modificare il suo nome: A$=GR$(0) portando così in A$ il contenuto del disegno. B$=GR$(0)+GR$(0) provoca la memorizzazione in B$ di due immagini identiche.

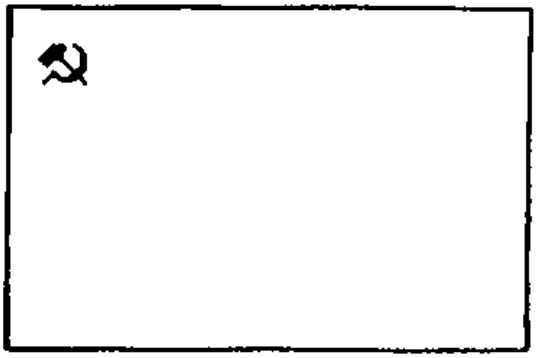
Viene ora proposto un programma che disegna la bandiera americana. Questo programma usa un carattere definito dall'utente, la stella.

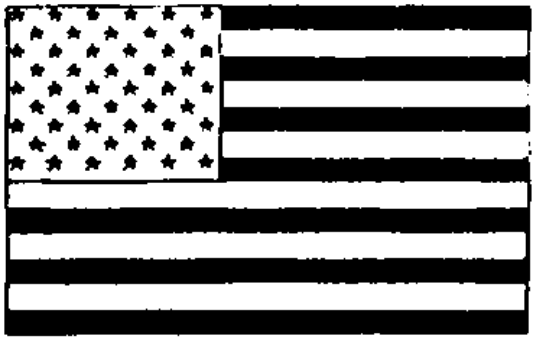



```
10 'BANDIERA AMERICANA
20 SCREEN 7,4,0:CLS:LOCATE 0,0,0
30 CLEAR,,1
40 DEFGR$(0)=8,8,28,127,28,20,34,0
50 FOR X=0 TO 10
60 FOR Y=0 TO 8
70 IF (X+Y)MOD 2 = 0 THEN LOCATE X,Y:PRINT GR$(0)
80 NEXT Y
90 NEXT X
100 X=90
110 FOR Y=0 TO 132 STEP 11
120 IF Y>72 THEN X=0
130 C=1+6*(Y MOD 2)
140 BOXF(X,Y)-(220,Y+10),C
150 NEXT Y
160 LOCATE 0,20:END
```

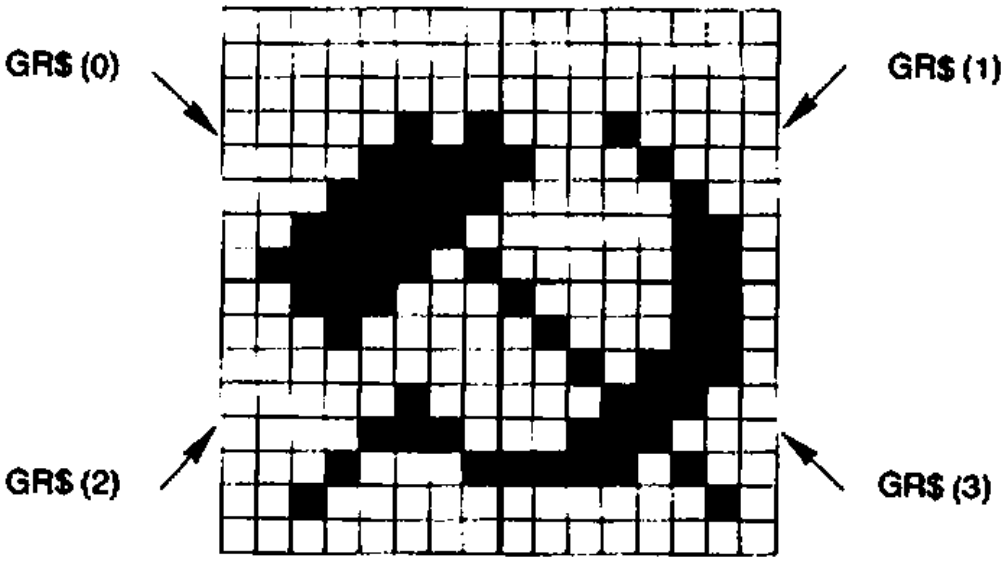
Commenti

50–90. Per la stampa delle 50 stelle vengono utilizzati due cicli nidificati. Il test di valore di (X+Y)MOD 2 permette di riconoscere le celle nelle quali devono essere presenti le stelle.
130. Formula per la scelta del colore 1 o 7.





Ora viene proposto un programma per disegnare la bandiera sovietica. Innanzitutto bisogna tentare di definire il carattere per ottenere la falce ed il martello. Per ottenere una rappresentazione realistica, occorre usare quattro caratteri.




```
10 'BANDIERA SOVIETICA
20 SCREEN 3,0,0:CLS
30 CLEAR,,4
40 DEFGR$(0)=0,0,0,5,15,31,63,127
50 DEFGR$(1)=0,0,0,16,136,4,6,6
60 DEFGR$(2)=56,16,0,4,14,17,32,0
70 DEFGR$(3)=134,70,46,28,56,244,2,0
80 FM$=GR$(0)+GR$(1)+CHR$(10)+CHR$(8)+CHR$(8)+GR$(2)+GR$(3)
90 BOXF(0,0)-(150,100),-2
100 LOCATE 1,1,0:COLOR 3,1:PRINT FM$
110 LOCATE 0,20:END
```

Commenti

Per stampare il motivo in una sola istruzione, si sono utilizzati i codici per lo spostamento del cursore:
- PRINT CHR$(10) determina uno spostamento verso il basso di una riga.
- PRINT CHR$(8) determina uno spostamento verso sinistra di una colonna.

(Vedere anche il capitolo 43)

Capitolo 40

Controllo dell'input

Se il programmatore ha intenzione di scrivere un programma interattivo, utilizzerà senz'altro le istruzioni INPUT e INKEY$. Scegliendo INPUT non ssrà possibile controllare i dati introdotti prima che venga premuto il tasto ENTER: l'utente potrà impostare tutti i caratteri che vorrà. L'uso di INKEY$ dà margini di sicurezza decisamente superiori poichè non si potrà mai verificare quanto detto prima per INPUT. Però anche INKEY$ presenta degli inconvenienti: il primo è dato dall'assenza di visualizzazione dei dati; si può facilmente aggirare l'ostacolo prevedendo un'istruzione PRINT subito dopo. Il secondo è la mancanza dell'editor, attivo invece con INPUT. Con INPUT l'utente ha la possibilità di spostare il cursore all'indietro per correggere eventuali errori, prima di premere ENTER.
Di seguito viene proposta una procedura per la registrazione di una serie di cifre, che contemporaneamente controlla che si tratti effettivamente di cifre. L'utente avrà la possibilità di correggere una cifra già impostata, spostando il cursore all'indietro. Tenere presente che lo scorrimento del cursore sulle cifre comporta la loro cancellazione. Il numero massimo di cifre può essere fissato a priori.

```
1'
2' EDITOR INKEY$
3'
10 LMAX=10
20 PRINT"Inserire un numero intero inferiore a";LMAX;"cifre:";
30 A$="":L=0:GOSUB 1000
999 END
1000'
1001' Attesa di battitura
1002'
1010 R$=INKEY$
1020 IF R$="" "THEN 1010
1030 C=ASC(R$)
1040 IF C=13 THEN 1999
1050 IF C=8 AND L>0 THEN A$= LEFT$(A$,L-1):PRINT
     CHR$(8)+""+CHR$(8);:L=L-1:GOTO 1010
1060 IF R$<"0" OR R$>"9" THEN 1010
1070 IF L=LMAX THEN 1010
1080 PRINT R$;:A$=A$+R$:L=L+1
1090 GOTO 1010
1999 RETURN
```

Variabili

A$ è la stringa che riceverà il numero. L è la lunghezza. LMAX è la lunghezza massima della stringa. R$ è il tasto eventualmente premuto sulla tastiera. C è il codice ASCII di R$.

Commenti

1020 Se nessun tasto è stato premuto, bisogna aspettare.
1040 Se il tasto premuto è ENTER, si esce dal sottoprogramma.
1050 Se viene premuto il tasto per lo spostamento all'indietro del cursore, e se A$ non è vuoto, si elimina l'ultimo carattere in A$ (correzione) quindi si arretra il cursore, si inserisce uno spazio vuoto e si arretra di nuovo il cursore.
1060 Se il tasto premuto non è un numero, si ritorna alla registrazione.
1070 Se A$ è pieno si ritorna alla registrazione.
1060 Si stampa la cifra scelta, la si accoda a A$ e si ritorna alla registrazione.

ONKEY=...GOTO...oppure
ONKEY=...GOSUB...

Il BASIC dà la possibilità all'utente di effettuare alcune scelte da tastiera. Questo significa che premendo un tasto verrà avviata una procedura associata a quel tasto. Il programma deve essere ovviamente protetto da interventi di questo tipo: questo è il ruolo che compete alle istruzioni ONKEY=...GOTO... e ONKEY=...GOSUB...
Per impostare una tastiera musicale occorre scegliere, per esempio, di far corrispondere le cifre da 1 a 0 alle sette note dell'ottava 5 e alle ultime tre note dell'ottava 4 (sulla tastiera 0 è il tasto situato a destra di 9).

1 corrisponde a O4 SO,..., 9 corrisponde a O5 LA, 0 corrisponde a O5 SI

Dalle istruzioni iniziali si può dedurre l'effetto dei tasti. Ad ogni tasto è associato un GOTO.

Programma

```
5' TASTIERA PIANO
10 ONKEY="1" GOTO 210
20 ONKEY="2" GOTO 220
30 ONKEY="3" GOTO 230
40 ONKEY="4" GOTO 240
50 ONKEY="5" GOTO 250
60 ONKEY="6" GOTO 260
70 ONKEY="7" GOTO 270
80 ONKEY="8" GOTO 280
90 ONKEY="9" GOTO 290
100 ONKEY="0" GOTO 300
200 GOTO 200
210 PLAY"O4 SO":GOTO 200
220 PLAY"O4 LA":GOTO 200
230 PLAY"O4 SI":GOTO 200
240 PLAY"O5 DO":GOTO 200
250 PLAY"O5 RE":GOTO 200
260 PLAY"O5 MI":GOTO 200
270 PLAY"O5 FA":GOTO 200
280 PLAY"O5 SO":GOTO 200
290 PLAY"O5 LA":GOTO 200
300 PLAY"O5 LA":GOTO 200
```

Commento

L'istruzione di attesa è un ciclo infinito 200 GOTO 200

Attenzione: questo tipo di programma permette di apprezzare la possibilità di duplicare le righe cambiando il loro numero.

Con ONKEY...GOSUB è possibile saltare ad un sottoprogramma tramite l'uso della tastiera.

Capitolo 41

Menu

La maggior parte dei programmi ben fatti cominciano e finiscono con un menu. Il menu di apertura propone delle opzioni (livello, scelta in un elenco) mentre il menu di chiusura offre un'alternativa tra il proseguimento e la fine del lavoro.
Generalmente, questi due menu si trovano in sottoprogrammi, alla fine del listato, per tenerli svincolati dal programma principale e per facilitarne la consultazione. Una delle prime istruzioni del programma principale rinvia al menu iniziale e una delle ultime al menu di chiusura. E' opportuno cominciare a scrivere i programmi dal menu, in modo da poterli provare, e a scrivere l'insieme dei sottoprogrammi richiamati dal menu, dotandoli di una numerazione di riga appropriata.
Ecco, per esempio, un programma che propone da menu quattro giochi con le carte: briscola, scopa, poker e bridge. Solo i menu di apertura e di chiusura sono scritti interamente. I sottoprogrammi sono fatti in modo che il programma possa essere verificato.

```
1'
2' GIOCHI CON LE CARTE
3'
20 GOSUB 10000
30 ON SCELTA GOSUB 1000,2000,3000,4000
40 GOSUB 15000
50 ON ZZ GOTO 20,999
999 END
1000'
1001 'BRISCOLA
1002'
1010 CLS:PRINT"BRISCOLA"
1020 GOSUB 20000
1999 RETURN
2000'
2001' SCOPA
2002'
2010 CLS:PRINT"SCOPA"
2020 GOSUB 20000
2999 RETURN
3000'
3001' BRIDGE
3002'
3010 CLS:PRINT"BRIDGE"
3020 GOSUB 20000
3999 RETURN
4000'
4001 'POKER
4002'
4010 CLS:PRINT"POKER"
```

```
4020 GOSUB 20000
4999 RETURN
10000'
10001'MENU
10002'
10010 CLS:PRINT"GIOCHI CON LE CARTE":PRINT
10020 PRINT"1:BRISCOLA"
10030 PRINT"2:SCOPA"
10040 PRINT"3:BRIDGE"
10050 PRINT"4:POKER"
10060 GOSUB 20000
10070 IF ZR<49 OR ZR>52 THEN 10060
10080 SCELTA=ZR-48
10999 RETURN
15000'
15001'FINE
15002'
15010 CLS:PRINT"VUOI CONTINUARE?"
15020 GOSUB 20000
15030 IF ZR=79 THEN ZZ=1:GOTO 15999
15040 IF ZR=78 THEN ZZ=2:GOTO 15999
15050 GOTO 15020
15999 RETURN
20000'
20001'ATTESA DI BATTITURA DI UN TASTO
20002'
20010 ZR$=INKEY$
20020 IF ZR$=""THEN 20010
20030 ZR=ASC(ZR$)
20999 RETURN
```

Input da penna ottica

Un menu interattivo può proporre sullo schermo una serie di risposte sotto forma di zone da indicare per mezzo di una penna ottica.
La coppia di istruzioni PEN e ON PEN... consente:
– di definire sullo schermo fino a otto differenti celle,
– di richiedere l'esecuzione di una riga differente per ogni cella, quando una cella è indicata dalla penna ottica. Dopo ON PEN, si può trovare un GOTO ad un'istruzione o un GOSUB a un sottoprogramma. La logica di funzionamento è la stessa di ON...GOTO e ON...GOSUB.

PEN

Questa istruzione mette in relazione un numero compreso tra 0 e 7 con una zona dello schermo ben definita.

Esempio di programma:
– Un primo sottoprogramma visualizza le scelte proposte e le celle destinate a ricevere le risposte della penna ottica.
– Ogni risposta è elaborata da un sottoprogramma apposito.

Attenzione: se la penna ottica tocca una zona esterna ai riquadri definiti dal primo sottoprogramma, l'istruzione ON PEN determinerà il passaggio all'istruzione successiva.

Programma

```
10'
20 CLS:SCREEN 0,6,6
30 LOCATE 3,11,0
40 GOSUB 1000
50 ON PEN GOSUB 2000,3000,4000,5000,6000,7000,8000,9000
60 GOTO 50
999 END
1000 REM MENU PRINCIPALE
1010 'Scelte proposte
1100 FOR N=0 TO 7
1110 COL=20+N*24
1120 BOX(COL,80)-(COL+24,95)
1130 PEN N:(COL,80)-(COL+23,95)
1140 NEXT N
1999 RETURN
2000 REM
2999 RETURN
3000 REM
3999 RETURN
4000 REM
4999 RETURN
5000 REM
5999 RETURN
6000 REM
6999 RETURN
7000 REM
7999 RETURN
8000 REM
8999 RETURN
9000 REM
9999 RETURN
```

Non resta che scrivere i sottoprogrammi.

Capitolo 42

Stampe di tabelle

La rappresentazione di tabelle numeriche i cui elementi sono risultati da calcoli, pone dei problemi di allineamento se si vuole che i numeri rientrino nelle celle previste e se si vuole che i numeri con decimali siano allineati. Esiste un'istruzione BASIC che si occupa di tutto questo: PRINT USING.
L'istruzione PRINT USING deve essere seguita da una stringa alfanumerica con la definizione del formato e con il numero che deve essere stampato.

Formato

Il formato "###.##" permette la visualizzazione di tutti i numeri con tre cifre prima della virgola (o del punto decimale) e due cifre dopo la virgola.

```
A$="###.##"
PRINT USING A$;453.2
453.20
PRINT USING A$;27.438
  27.44
PRINT USING A$;5845.3
%5845.30
OK
```

Commenti

- Se nscessario, la perte decimale viene riempita di zeri.
- Se le cifre della parte decimale non sono tutte visualizzabili, si ha un arrotondamento ai decimali previsti dal formato, con troncamento dei decimali successivi.
- Se la parte intera è troppo grande, il numero viene visualizzato preceduto dal segno % che indica un'incoerenza tra dato e formato.

Esempio: Questo programma crea una tabella nella quale figurano un prezzo imponibile (IMP), l'ammontare dell'imposta IVA e il totale comprensivo di IVA (TOT). L'acquisizione dei dati viene ottenuta da un sottoprogramma (INPUT).

```
1'
2'IVA
3'
20 A$="####.##"
30 GOSUB 1000:CLS
40 PRINT TAB(2);"IMP";TAB(12);"IVA";TAB(22);"TOT":PRINT
50 FOR K=1 TO NA
60 PRINT TAB(2);:PRINT USING A$;IMP(K);
70 PRINT TAB(12);:PRINT USING A$;IVA(K);
80 PRINT TAB(22);:PRINT USING A$;TOT(K)
100 NEXT K
999 END
1000'
1001'INTRODUZIONE DATI
1002'
1010 CLS
1020 INPUT"NUMERO DEGLI ARTICOLI:";NA
1030 INPUT"PERCENTUALE IVA:";PRC
1040 DIM IMP(NA),IVA(NA),TOT(NA)
1050 FOR K=1 TO NA
1060 PRINT"ARTICOLO:";K;
1070 INPUT"IMPONIBILE:";IMP(K)
1080 IVA(K)=IMP(K)*PRC/100
1090 TOT(K)=IMP(K)+IVA(K)
1100 NEXT K
1999 RETURN
```

Svolgimento

IMP	IVA	TOT
78.90	14.20	93.10
175.00	31.50	206.50
567.42	102.14	669.56
89.44	16.10	105.54
715.00	128.70	843.70

Per l'approfondimento delle caratteristiche di questa istruzione si consiglia la consultazione della guida di riferimento.

Nel formato dell'istruzione PRINT USING è possibile richiedere:
– La visualizzazione del segno o di un altro carattere davanti al numero.
– La visualizzazione di asterischi negli spazi non stampati.
– La visualizzazione della virgola come delimitatore di gruppo di tre cifre (notazione anglosassone).
– La visualizzazione in notazione esponenziale.
PRINT USING può anche essere utilizzato per modificare stringhe alfanumeriche. Il formato "!" visualizza solamente il primo carattere.

```
PRINT USING "!";"AL1"
A
OK
```

Il formato "@@" (o "%%") riserva tanti spazi quanti ne intercorrono fra i due @ (o i due %) più due.

Tabulazione

L'istruzione PRINT TAB (X) permette di spostare di X caratteri la visualizzazione di una riga, a partire dalla prima colonna a sinistra sullo schermo. E' molto utile per visualizzare testi al centro dello schermo o per allineare più colonne di stringhe alfanumeriche.
La prima colonna ha il numero 1.
Ecco una routine che visualizza delle stringhe A$ al centro dello schermo. Alla riga 10 viene fissato il numero di colonne NC=40.

```
10 NC=40
20 A$="BASIC":GOSUB 100
30 A$="TABULAZIONE":GOSUB 100
99 END
100 REM CENTRATURA
110 LUNG=LEN(A$)
120 IF LUNG>NC THEN X=0 ELSE X=INT((NC-LUNG)/2)
130 PRINT TAB(X);A$
199 RETURN RUN
```

```
BASIC
TABULAZIONE
```

Capitolo 43

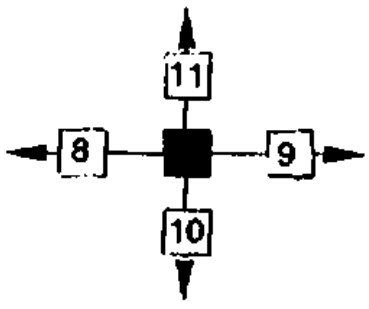
Spostamenti del cursore da tastiera

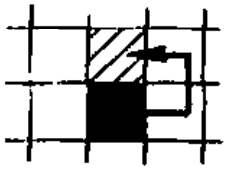
In questo capitolo viene proposto un programma interattivo che permette all'utente di spostare a piacere un carattere sullo schermo. Questa routine è indispensabile per i giochi d'azione. Lo spostamento viene realizzato utilizzando i quattro tasti freccia.

L'idea di questo programma è la seguente:
1 – Leggere il tasto premuto tramite INKEY$.
2 – Controllare che lo spostamento richiesto non determini l'uscita dallo schermo.
3 – Cancellare il carattere, poi riscriverlo a sinistra, a destra, sopra o sotto.

Programma

Sono autorizzati solo i quattro tasti di spostamento del cursore. I relativi codici ASCII sono 8, 9, 10 e 11. Le righe 110, 210, 310 e 410 contengono dei controlli per evitare di superare i bordi dello schermo.
Per capire il significato di D$ bisogna consultare la riga di visualizzazione 500.
CHR$(32) corrisponde allo spazio vuoto usato per la cancellazione del carattere.
D$ permette di posizionare il cursore nel punto corrispondente al tasto premuto. Così, per il tasto di spostamento verso l'alto, D$ farà sì che il cursore si sposti di un passo verso l'alto e di un passo a sinistra.





```
1'
2'SPOSTAMENTO DA TASTIERA
3'
20 CLS
30 X$=CHR$(8):SX=20:SY=10:X= 0:Y=0
40 J$=CHR$(127)+X$:PRINT J$
50 R$=INKEY$
60 IF R$=""THEN GOTO 50
70 A=ASC(RS)
80 IF A<8 OR A>11 THEN GOTO 50
90 ON A-7 GOTO 200,100,400,300
100'
101'DESTRA
102'
110 IF X>SX THEN GOTO 50
120 D$="":X=X+1:GOTO 500
200'
201'SINISTRA
202'
210 IF X<1 THEN GOTO 50
220 D$=X$+X$:X=X-1:GOTO 500
300'
301'ALTO
302'
310 IF Y<1 THEN GOTO 50
320 D$=CHR$(11)+X$:Y=Y-1:GOTO 500
400'
401'BASSO
402'
410 IF Y>SY THEN GOTO 50
420 D$=CHR$(10)+XS:Y=Y+1:GOTO 500
430 REM VISUALIZZAZIONE
500 PRINT CHRS(32);D$;J$:
510 GOTO 50
```

Commenti

Variabili

X$: abbrevia la scrittura di CHR$(8) che è lunga da scrivere e che compare più volte.
SX e SY: limiti dello spostamento (colonne e righe).
X e Y: posizione attuale del carattere. Inizialmente in alto a sinistra.
J$: stringa formata dal carattere pieno e dal carattere di spostamento all'indietro del cursore.
R$: carattere battuto da tastiera.
A: codice ASCII di R$.
D$: carattere di controllo per lo spostamento del cursore.

Utilizzo di un cronometro

La finalità dell'esempio successivo è quello di realizzare un gioco e di visualizzare un cronometro su schermo.

Programma

Il cronometro è inserito in un sottoprogramma. L'istruzione ON INTERVAL... fa riferimento all'orologio interno del PC 128, quello su cui si basa il funzionamento del microprocessore. Questo orologio ha intervalli di tempo di 1/10 di secondo. Per questo motivo l'istruzione ON INTERVAL= 10 GOSUB prevede, a ogni secondo, un salto al sottoprogramma definito da GOSUB; questo si verifica qualunque sia l'indirizzo del programma nel quale ci si trova. Avviare il cronometro significa passare dall'istruzione INTERVAL ON.

```
1'
2' Gioco
3'
10 SCREEN 3,0,0:CLS
20 ON INTERVAL=10 GOSUB 1000
30 INTERVAL ON
40 DRA=0
50 GOSUB 10000
60 IF DRA=0 THEN 50
999 END
1000'
1001' Cronometro
1002'
1010 SEC=(SEC+1)MOD 60:ATTRB 0,0
1020 LOCATE 35,0:PRINT SEC
1999 RETURN
10000'
10001' Gioco
10002'
10010 ATTRB 1,1:COLOR 0,3
10020 LOCATE 10,10:PRINT CHR$(49+RND*7)
10030 FOR I=1 TO 60
10040 IF INKEY$<>""THEN DRA=1:EXIT
10050 NEXT I
10999 RETURN
```

Commenti

Il sottoprogramma di cronometro visualizza lo scorrere dei secondi sullo schermo, in alto a destra. Il sottoprogramma 10000 visualizza nel centro dello schermo un numero casuale a intervalli regolari. Si ferma quando l'utente preme un tasto.

Capitolo 44

Principi di animazione

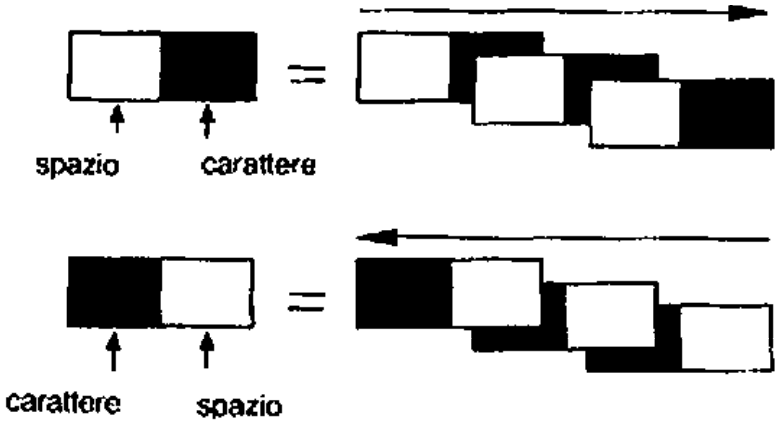
Sia per gli esperti in informatica che per i principianti, i principi fondamentali dell'animazione sono gli stessi. Per spostare un punto o un gruppo di punti, un carattere o un gruppo di caratteri, bisogna visualizzarli, rivisualizzarli in un'altra posizione e cancellare la vecchia posizione. Il più delle volte questi spostamenti vengono gestiti all'interno di cicli di FOR...NEXT.

La velocità dello spostamento corrisponde alla velocità di visualizzazione e di cancellazione: il BASIC è reputato un linguaggio particolarmente lento; è però opportuno chiarire cosa si intende per lento. Se si scrive un ciclo anche molto lungo per la visualizzazione di numeri, lo schermo si riempie molto rapidamente. La visualizzazione appare lenta quando all'interno del programma sono previsti molti calcoli e controlli, oppure quando il programma prevede numerose visualizzazioni e numerosi spostamenti simultanei. Un esempio tipico è un gioco interattivo del tipo guerre stellari. Si può quindi affermare che la programmazione in BASIC è incompatibile con una gestione dello schermo molto mobile.

In questi casi, gli esperti utilizzano il linguaggio macchina, la cui velocità di esecuzione è enormemente superiore a quella del BASIC. In ogni caso il BASIC è in grado di effettuare in modo accettabile animazioni non troppo complesse. In certi casi, quando si tratta di animazioni molto semplici, la velocità risulta abbastanza elevata e gli occhi riescono a malapena a seguirne il movimento.

Oscillazioni orizzontali

Lo scopo di questo programma è quello di far oscillare un carattere su una linea orizzontale. Per il movimento da sinistra a destra, si visualizza un carattere preceduto da uno spazio vuoto che provvede a cancellare la posizione precedente. Per il movimento da destra a sinistra, si visualizza il carattere seguito da uno spazio vuoto che provvede a cancellare la posizione precedente.



Per gli spostamenti da sinistra a destra, bisogna spostare il cursore di una colonna verso sinistra. Per gli spostamenti da destra a sinistra, occorre spostare il cursore di tre colonne verso sinistra (X$ e XX$).

Lo stesso sottoprogramma gestisce entrambi gli spostamenti. Le variabili di input sono: .
– A e B sono le colonne all'estremità dello schermo utile.
– D$ è il carattere di ritorno cursore.
– P è il passo.
– C$ è la stringa da stampare.

```
1'
2' OSCILLAZIONI
3'
20 CLS
30 LOCATE 0,10,0
40 X$=CHR$(8):XX$=X$+X$+X$
45 OO
50 REM SINISTRA DESTRA
60 A=0:B=20:D$=X$:P=1:C$="^"
70 GOSUB 1000
80 REM DESTRA SINISTRA
90 A=20:B=0:D$=XX$:P=-1:C$="*"
100 GOSUB 1000
110 LOOP
120 END
1000'
1001'SPOSTAMENTO
1002'
1010 FOR K=A TO B STEP P
1020 PRINT C$;O$;
1030 FOR N=1 TO 10:NEXT N
1040 NEXT K
1099 RETURN
```

Commenti

All'utente la scelta per il test di fine del ciclo DO...LOOP. All'interno del ciclo di visualizzazione è previsto un ciclo di temporizzazione per rallentare lo spostamento.

Scacchiera

E' molto raro trovare un programma di giochi o un programma educativo che non preveda al suo interno l'uso di una scacchiera. Di seguito viene proposto un sottoprogramma in grado di realizzare qualsiasi tipo di scacchiera.

Programma

Variabili principali:
– Coordinate dell'angolo superiore sinistro: SX e SY
– Numero colonne: NC
– Numero righe: NR
– Lunghezza di una colonna: LC
– Altezza di una riga: AR
– Colore: C

```
1'
2' Realizzazione di qualsiasi tipo di scacchiera
3'
10 CLS:SCREEN 3,0,0
20 SX=50:SY=20
30 NC=10:NR=8
40 LC=20:AR=12
50 C=3
60 GOSUB 1000
999 END
1000'
1001' Scacchiera
1002'
1010 TX=SX+NC*LC:TY=SY+NR*AR
1020 FOR X=SX TO TX STEP LC
1030 LINE(X,SY)-(X,TY),C
1040 NEXT X
1050 FOR Y=SY TO TY STEP AR
1060 LINE(SX,Y)-(TX,Y),C
1070 NEXT Y
1999 RETURN
```

Caratteristiche della scacchiera

Commenti: Per utilizzare questo sottoprogramma, è sufficiente modificare le 7 variabili di input definite nelle righe da 20 a 50.
– TX e TY sono le coordinate dell'angolo inferiore destro.

Capitolo 45

Ultimi consigli di programmazione

Scrivere un piccolo programma non pone grossi problemi. Il programmatore può ritrovare facilmente il filo conduttore di ciò che ha fatto e del perchè lo ha fatto. Diverso è quando il programma è più impegnativo: in quel caso diventa pericoloso contare sulla memoria del programmatore, magari a distanza di mesi, per capire l'impostazione del programma. Da questo punto di vista il BASIC è un linguaggio molto pericoloso perchè permette una programmazione immediata e non obbliga il programmatore a strutturare il proprio programma. Il metodo di programmazione proposto nel capitolo 22 invitava a scomporre il programma in moduli elementari. Seguendo questo metodo i programmi risulteranno più chiari e più facilmente leggibili.
I consigli che vengono dati di seguito riguardano alcuni punti particolarmente delicati: le uscite dai cicli.

FOR...NEXT e DO...LOOP

Per entrare in un ciclo FOR...NEXT, bisogna obbligatoriamente iniziare da una riga dove sia presente FOR, e uscire da una riga dove sia presente NEXT.
Allo stesso modo, è vietato entrare in un ciclo DO...LOOP, se non passando attraverso DO.
Ignorare o dimenticare questi due principi significa crearsi notevoli fastidi. Non bisogna mai dimenticarlo, poichè le istruzioni GOTO permettono di rientrare e di uscire dai cicli senza controllo determinando un'enorme confusione. Ad esempio, ss un ciclo è presente dalla riga 100 alla 150, nessun GOTO esterno al ciclo dovrà indirizzarsi ad un numero di riga compreso fra 100 e 150. Inoltre, nelle righe comprese tra 100 e 150, non dovrà essere presente alcun GOTO che si indirizzi ad un numero di riga inferiore a 100 o superiore a 150. Al contrario, i GOSUB all'interno di un ciclo sono ammessi poichè il rientro è assicurato.
Per entrare in un ciclo non vengono fatte eccezioni a quanto detto. L'uscita da un ciclo, invece, può avvenire senza che il ciclo stesso sia stato completamente sviluppato.
Per uscire dai cicli FOR...NEXT e DO...LOOP senza attenderne la fine, la migliore soluzione è l'utilizzo dell'istruzione EXIT. Tutti gli attri metodi (GOTO compreso) devono essere evitati poiché comportano la perdita della strutturazione del programma. In presenza di cicli nidificati, si consiglia di prevedere una riga di fine ciclo per ogni ciclo aperto, anche se il BASIC permette di chiudere tutti i cicli sulla medesima riga. Questo metodo facilita la lettura dei programmi. Se durante l'esecuzione di un programma dovessero sorgere difficoltà, il calcolatore è in grado di aiutare il programmatore nella procedura di messa a punto. Le istruzioni a disposizione sono TRON e TROFF.

TRON–TROFF

L'istruzione TRON permette di visualizzare, durante l'esecuzione di un programma, una lista di numeri di riga di tutte le istruzioni eseguite, racchiusi tra parentesi quadre.

```
TRON
OK
RUN
[10][30][20][40][10][30][20][40][10][30]
[20][40][10][30][20][40][10][30][20][40]
[10][30][20][40][10][30][20][40][10][30]
[20][40][10][30][20][40][10][30][20][40]
[10][30][20][40][10][30][20][40][10][30]
```

Il comando TROFF annulla il comando TRON e interrompe la funzione.
Parlando di cicli, è opportuno segnalare gli inconvenienti che si possono più facilmente verificare. Il problema maggiore è quello del programma che "entra in loop", cioè quel programma che non arriva all'istruzione END. Questo problema si verifica spesso nei controlli in cui è richiesta un'accurata precisione numerica. Bisogna ricordare che le cifre usate dal calcolatore sono codificate e che la precisione della rappresentazione provoca a volte delle sorprese.

Esempio:

Il calcolatore ha due rappresentazioni dello zero: una per i numeri interi (due byte) e una per i numeri reali. Controllare se un numero reale è uguale a zero scrivendo IF X=0, può comportare dei problemi. E' preferibile usare IF ABS(X)<1.E–7. Poiché la precisione di rappresentazione dei numeri reali è fino alla settima cifra significativa, questo test è sufficiente. Ciò implica che per il calcolatore 0,0000001 e 0 sono uguali. Muovendosi nella stessa logica, viene ora riportato un programma che gestisce una grande quantità di numeri. E' opportuno ricordare che il limite dei numeri utilizzabili in semplice precisione è di 10^{-38}. Se si supera questo valore si verifica un errore di "overflow".

Esempio:

Volendo calcolare:

$$\frac{A^n}{n!} = \frac{AxAx..xA}{1x2x3x...xn}$$

La prima ipotesi consiste nella stesura di un ciclo per il calcolo del numeratore e la seconda nella valutazione del denominatore che con la crescita di n porterà inevitabilmente ad un overflow. Bisogna quindi calcolare il rapporto fra le due espressioni via via che le si valuta in modo da rimanere entro i limiti ammessi.

Programma

```
10 REM NUMERI
20 N=100:A=50
30 S=A
40 FOR I=2 TO N
50 S=S*A/I
60 NEXT I
99 END
```

Commento: viene lasciato al programmatore il compito di realizzare l'altro metodo e di confrontare i risultati.

Attenzione: gli errori che si presentano nei programmi possono essere gestiti dal programmatore. E' sufficiente utilizzare le istruzioni ON ERROR... e RESUME (vedere la parte "Guida di riferimento").

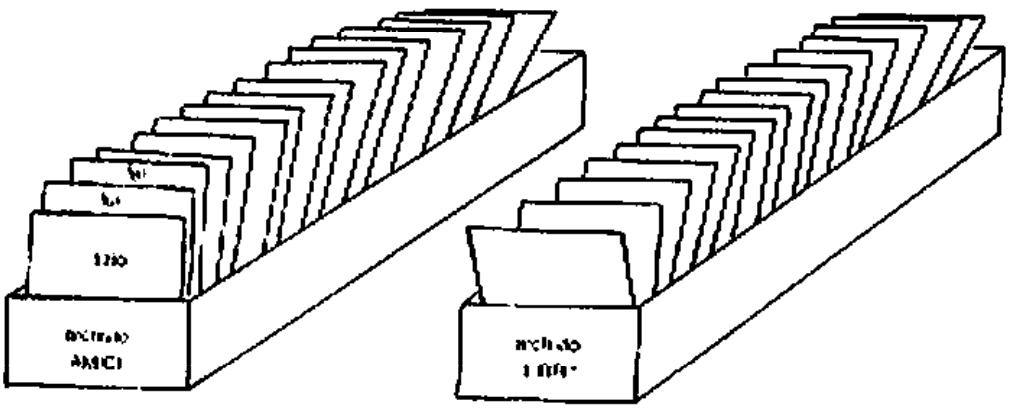
Capitolo 46

File sequenziali

Nel capitolo 20 si è affrontato il problema della memorizzazione di programmi su cessetta e del loro riutilizzo. In questo modo è stato creato un archivio (o file) di programma.

Cos'è un file?

Un file è una serie di dati memorizzati uno dopo l'altro proprio come delle schede in uno schedario.



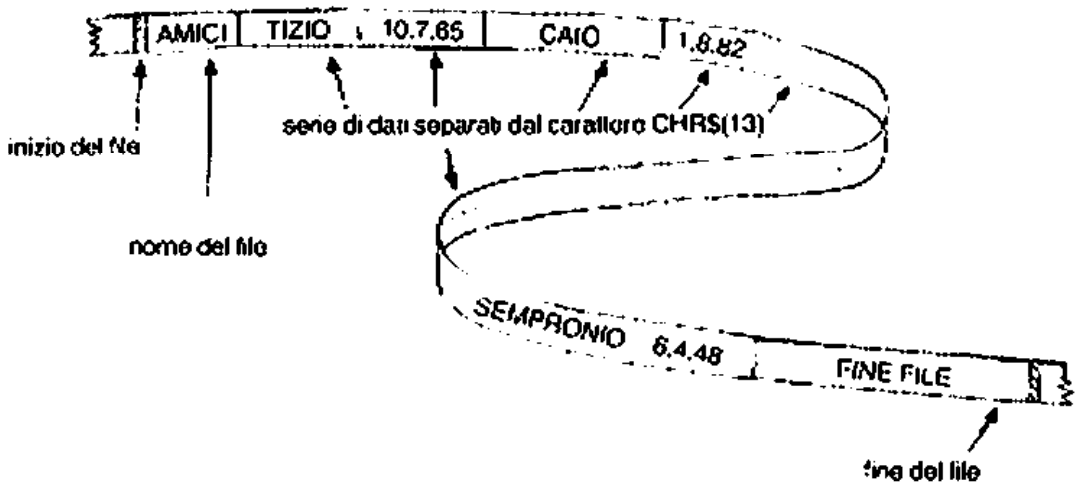
Ogni scheda contiene un certo numero di informazioni relative alla corrispondente "registrazione".
Nel caso di un file di programma, ogni scheda contiene un'istruzione numerata del programma registrato. Esiste un altro tipo di file: un file di dati. Al posto delle istruzioni sono registrati dei dati, cioè delle liste di parole che comprendono stringhe alfanumeriche. Potrebbe trattarsi, per esempio, del file dei propri amici; ogni scheda (record) conterrebbe il numero di telefono e la data di nascita; oppure del file dei clienti di un'azienda con ciascun record contenente le sue caratteristiche, la situazione degli ordini e la situazione contabile; oppure del file dei libri di una biblioteca.
Inserire dei dati in un file è spesso più vantaggioso che prevederli all'interno di un programma: innanzitutto su cassetta si possono memorizzare molti più dati, poi gli stessi dati presenti su file possono essere usati da programmi differenti.
A differenza dei programmi, con i dati non è possibile effettuare registrazioni, caricamenti o semplicemente gestire un file di dati con una sola istruzione (es. SAVE, LOAD, RUN).

Gestione di un file

Un insieme di record può essere letto e consultato oppure modificato (aggiunta o eliminazione di un record). Per consultare un file o per scrivere al suo interno è necessario:
– Trovare il file (per esempio, grazie al suo "nome") e aprirlo.
– Leggere i record finchè non si trova quello desiderato.
– Prendere il record e farlo arrivare a chi lo deve trattare (utilizzando il canale adeguato).
– Trattare le informazioni contenute nel record o modificarle, quindi reinserirle nel file.
– Chiudere il file.

File su cassetta

In pratica un file su cassetta è costituito da:
– un record iniziale che contiene il nome del file,
– una lista dei dati alfanumerici o numerici,
– un codice che segnala la fine dei record relativi a quel file.



Questo tipo di file viene definito sequenziale. Ciò significa che non è possibile inserire dati nuovi nel file senza aver fatto scorrere tutta la serie dei dati già esistenti: per inserire un nuovo dato bisogna "accodarlo" ai precedenti. Nessun dato ha un proprio nome, per cui è impossibile poterlo richiamare. Per consultare un file sequenziale bisogna avere presente la struttura dei record al momento della sua creazione in modo da poter "riconoscere" ogni dato via via che si presenta. Nell'esempio illustrato sopra, ogni record del file AMICI è costituito da un nome seguito da cifre che indicano la data di nascita.

Come effettuare una registrazione

Un file di dati è registrato su cassetta, cioè su una memoria di massa. Il calcolatore può elaborare solamente i dati presenti in memoria centrale (CPU). Effettuare una registrazione su file significa dunque trasferire dati da una memoria (CPU) ad un'altra memoria (quella di massa). Per farlo, bisogna innanzitutto scegliere un numero di canale.
In realtà, i dati presenti in memoria centrale passano prima attraverso una "zona tampone" (detta *buffer*), poi vengono trasferiti su cassetta.

OPEN, WRITE# e CLOSE

L'istruzione

OPEN"0",#1,"AMICI"

apre il file "AMICI" in output (dove cioè verrà fatta una scrittura). Il canale riservato alla sua gestione è il numero 1.

Esempio: si vuole creare un file contenente il nome e il numero di telefono di alcuni amici. Le informazioni da registrare sono inserite da tastiera.

```
5 CLS
10'REGISTRAZIONE RUBRICA
20 OPEN"0",#1,"RUBRICA"
40 DO
50 INPUT"NOME:";NOME$
60 IF NOME$=""THEN EXIT
70 INPUT"TELEFONO",TEL$
80 WRITE#1,NOME$,TEL$
90 LOOP
100 CLOSE#1
110 PRINT"REGISTRAZIONE TERMINATA"
999 END
```

Commenti

Le righe da 50 a 80 contengono un ciclo di registrazione. Il cicto termina quando si inserisce un nome nullo, cioè quando si preme direttamente ENTER in risposta alla domanda della riga 50.
100: Non dimenticare di chiudere il canale dopo l'uso. L'istruzione CLOSE#1 effettua la chiusura richiesta. Naturalmente prima dell'esecuzione del programma bisogna che il registratore sia impostato per la registrazione, con la cassetta inserita. E' nscessario posizionare esattamente l'inizio della registrazione che sta per essere effettuata.

Capitolo 47

Lettura/Scritture

PRINT#

Viene ora illustrato un altro esempio di scrittura di dati in un file su cassetta. Dall'esempio ci si può rendere conto che una serie di istruzioni DATA non è altro che un file sequenziale. Questo file viene consultato con l'istruzione READ. Il puntatore a DATA è assimilabile alla testina di lettura del registratore: nel momento in cui ne riceve l'ordine esso è in grado di effettuare la lettura dei dati successivi.

Programma

```
10 DATA TIZIO,34512633.CAIO,23542647,
   IO,40103748
11 DATA LEI,72091410,NESSUNO,45667890
19 DATA Z,0
100'
101'REGISTRAZIONE
102'
110 OPEN"0",#1,"TELEFONO"
115 DO
120 READ A$,T$
130 IF A$="Z"THEN EXIT
140 PRINT#1,A$,T$
150 LOOP
190 CLOSE#1
199 END
```

Commenti

- I dati dell'istruzione 19 permettono di identificare la fine della lista delle informazioni da registrare.

- L'istruzione

PRINT#1,A$,T$

ordina l'invio delle variabili A$ e T$ attraverso il canale 1 (che è stato assegnato ad uno specifico nome di file dall'istruzione OPEN).

Nota: l'istruzione PRINT# scrive i dati nello stesso formato nel quale l'istruzione PRINT li visualizza.

Lettura

Per leggere un file appena registrato, è necessario:
– Assicurarsi che il registratore sia posizionato sulla lettura e che la testina di lettura del nastro sia posizionata all'inizio del file registrato.
– Aprire la comunicazione tra la memoria centrale e la cassetta tramite un canale che può essere diverso da quello utilizzato per la scrittura.
– Trasferire in memoria centrale il contenuto del file ed eventualmente visualizzarlo sullo schermo.
– Chiudere la comunicazione tra la memoria centrale e la cassetta.

Esempio:

```
1 REM***LETTURA***
10 CLEAR 600
20 DIM A$(30),T$(30)
100 OPEN"I",#1,"TELEFONO"
110 I=1
120 DO
130 IF EOF(1)THEN EXIT
140 INPUT#1,A$(I),T$(I)
150 I=I+1
160 LOOP
200 CLOSE#1
999 END
```

Commenti

– L'istruzione

OPEN"I",#1,"TELEFONO"

apre il file "TELEFONO" (in lettura). Il canale interessato è il numero 1.
– La lettura dei dati si effettua tramite l'istruzione INPUT#. La sintassi è uguale a quella dell'istruzione INPUT. Ai dati vengono associati dei nomi e i dati stessi vengono memorizzati come elementi di due matrici: ciò permetterà in seguito di richiamarli (l'istruzione 20 riserva trenta zone di memoria per ogni matrice).

EOF

Nell'esempio precedente, il test per l'uscita dal ciclo introduce una variabile EOF(1). Il numero 1 indica che il canale interessato è il numero 1.
EOF è l'abbreviazione di *end of file* (fine file). Quando si legge un file è indispensabile controllare dopo ogni lettura se è stato letto un EOF; se il controllo non viene effettuato e si presenta un EOF si avrà il seguente messaggio di errore: "Input Past End" cioè tentativo di lettura dopo la fine del file.
La variabile EOF(1) è di tipo logico ed assume il valore VERO quando si legge l'ultimo dato del file trattato sul canale 1. Il test è quindi un test logico (non si scriverà quindi IF EOF(1)=0...).

Attenzione: come avviene per i programmi, i nomi dei file di dati prevedono un massimo di otto caratteri senza virgole nè punti.

Periferiche utilizzabili con un file

Se non esiste altra indicazione, l'apertura del canale 1 riguarda il registratore. Se oltre alla cassetta è presente un drive, questo diventerà prioritario rispetto al registratore. Ecco le diverse periferiche che possono essere utilizzate.

in scrittura

OPEN"O",#1,"CASS:RUBRICA"
(CASS è facoltativo nel caso in cui non sia presente un drive)
per la scrittura su cassetta,

OPEN"O",#1,"LPRT:"
per la scrittura su stampante,

OPEN"O",#1,"SCRN:"
per la scrittura su schermo,

OPEN"O",#1,"COMM:"
per la scrittura su interfaccia seriale.

La descrizione del file sarà "CASS:RUBRICA". Per le ultime tre periferiche è inutile precisare il nome del file dato che viene solo trasmesso e non memorizzato in lettura.

in lettura

OPEN"I",#1,"1:RUBRICA"
in lettura, dal lettore 1,

OPEN"I",#1,"CASS:RUBRICA"
in lettura, da cassetta,

OPEN"I",#1,"KYBO:"
in lettura, da tastiera (KEYBOARD),

OPEN"I",#1,"COMM:"
in lettura, da interfaccia seriale.

E' ovviamente impossibile leggere un file visualizzato su schermo o inviato alla stampante: queste due periferiche sono esclusivamente di output. Soto i drive, i registratori e l'interfaccia seriale possono essere usati sia in input sia in output. Due periferiche presentano caratteristiche differenti rispetto alle altre: lo schermo e la tastiera. Le due istruzioni BASIC INPUT e PRINT consentono al catcolatore di leggere dalla tastiera (input dei dati) o di scrivere su video (visualizzazione) senza usare canali. Per mettere in parallelo queste due periferiche con le altre, al programmatore viene offerta la possibilità di aprire un file in lettura da tastiera (KYBD:) e in scrittura su video (SCRN:).

Capitolo 48

Esempio di output su stampante

```
10 REM*LETTURA*SCRITTURA*
20 CLEAR 600
30 DIM A$(30),T$(39)
100 OPEN"I",#1,"TELEFONO"
110 GOSUB 1000
120 CLOSE#1
200 OPEN"O",#2,"LPRT"
210 GOSUB 2000
220 CLOSE#2
999 END
1000 REM LETTURA
1010 I=1
1020 DO
1030 IF EOF(1)THEN EXIT
1040 INPUT#1,A$(I),TS(I)
1050 I=I+1
1060 LOOP
1999 RETURN
2000 REM SCRITTURA
2010 FOR N=1 TO I
2020 PRINT#2,"NOME:";A$(N),"TELEFONO:";T$(N)
2030 NEXT N
2040 RETURN
```

Commenti

Il sottoprogramma 1000 è identico a quello descritto per la lettura. Rispetto all'altro fornisce anche il numero I di record letti. I è inferiore a 30, dimensione prevista per la lettura.
Il sottoprogramma 2000 prevede un ciclo di stampa che utilizza la variabile I fornita dal primo sottoprogramma.

Variazioni

E' possibile prevedere un solo ciclo leggendo un record sul canale 1 e riscrivendolo subito dopo sul canale 2. In questo caso si possono eliminare le due matrici definite all'inizio del programma. Tenere però presente che questo metodo non permette di trattare nuovamente i dati letti poichè questi vengono via via cancellati e sostituiti dai nuovi dati.

Attenzione: per effettuare stampe più accurate, è possibile utilizzare l'istruzione PRINT#USING allo stesso modo di PRINT USING.

Un file di numeri: punti di un disegno

Le coordinate dei punti di un disegno possono essere previste nell'istruzione DATA. Però se esse vengono memorizzate in un file, potranno essere richiamate da qualsiasi programma, utilizzando il disegno senza dovere riscrivere la lista delle coordinate.

Esempio

```
10*DISEGNO*
20 OPEN"O",#1,"DISEGNO"
30 DO
40 INPUT"COL.RIG";COL,RIG
50 IF COL=-1 THEN EXIT
60 WRITE#1,COL.RIG
70 LOOP
80 CLOSE#1
99 END
```

Commenti

Il test per l'uscita dal ciclo utilizza come indice per le colonne il valore –1 (per RIG un valore qualsiasi). Questo programma impone l'inserimento delle coordinate una alla volta: il lettore sarà senz'altro in grado di trovare un metodo migliore per l'inserimento dei dati. E' possibile, per esempio, impostare il disegno con la penna ottica semplicemente sostituendo INPUT nella riga 40 con INPUTPEN. Attenzione però che l'impostazione sequenziale del file non permette ripensamenti o correzioni. L'utilizzo di punti memorizzati può essere fatto nel seguente modo:

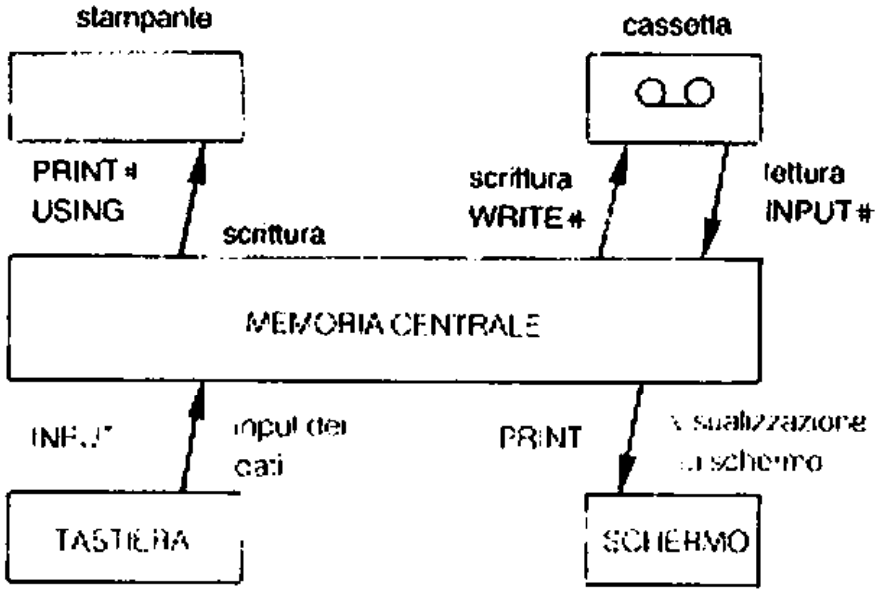
Programme

```
10 'DIMOSTRAZIONE
20 CLS
30 OPEN"I",#1,"DISEGNO"
40 INPUT#1,C,R
50 PSET(C,R)
60 DO
70 IF EOF(1)THEN EXIT
80 INPUT#1,COL,RIG
90 LINE-(COL,RIG)
100 LOOP
110 CLOSE
999 END
```

Commenti

- Il primo punto letto rappresenterà il punto di partenza del disegno.
- Il programma utilizza direttamente i valori contenuti nel file, senza visualizzarli.

Di seguito vengono riassunte le diverse possibilità offerte dai file sequenziali:



MERGE

A chiusura del capitolo sulla gestione dei file è opportuno segnalare l'esistenza di operazioni eseguibili sui file. Una di queste è molto importante, soprattutto quando agisce sui file di programma.
L'istruzione MERGE permette il caricamento di un programma registrato su una memoria di massa (in questo caso la cassetta). Il programma viene portato nella memoria centrale dell'elaboratore senza cancellare il programma già presente. In questo modo si possono concatenare i due programmi e renderli operativi con una sola istruzione RUN. La prima precauzione da prendere riguarda il numero delle righe di istruzione: i due programmi devono avere assolutamente numeri di righe diversi.
La seconda precauzione riguarda il tipo di salvataggio adottato per ta memorizzazione del programma. Il programma richiamato da cassetta deve essere registrato nel formato ASCII.
Per fare questo, è sufficiente aggiungere l'estensione A all'istruzione SAVE:

```
SAVE"PROVA",A
```

effettua una memorizzazione del programma nel formato ASCII.

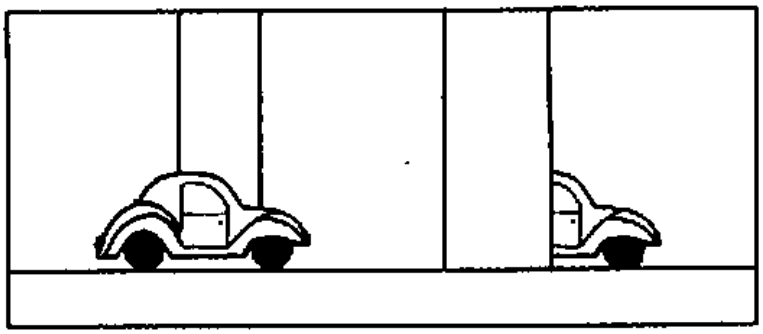
Capitolo 49

L'effetto camaleonte

In questo capitolo verrà affrontato l'uso dell'istruzione CONSOLE.
L'istruzione CONSOLE può essere seguita da quattro argomenti. I primi due definiscono una finestra di scorrimento orizzontate: questa istruzione è l'equivalente orizzontale dell'istruzione WINDOW.
Il terzo gestisce effetti speciali di animazione. Il quarto controlla la velocità di scorrimento: 0 corrisponde alla velocità normale, piuttosto rapida, 1 corrisponde a una velocità molto più contenuta.

Gestione del colore

Il terzo argomento permette sorprendenti effetti di animazione. Per sfruttare questa possibilità bisogna assegnargli il valore 1. Pertanto CONSOLE 0,24,1,0 estenderà l'effetto su tutto lo schermo. Il principio è il seguente: nel momento in cui viene incontrata questa istruzione, tutti i colori attualmente presenti sullo schermo vengono fissati fino all'esecuzione di una successiva istruzione CONSOLE. E' possibile stampare e spostare i caratteri su tutto lo schermo, ma nelle posizioni dove i caratteri sono stampati i colori di primo piano e di sfondo sono quelli fissati prima dell'istruzione CONSOLE. Dal momento che l'effetto si estende su tutto lo schermo, è inutile utilizzare l'istruzione COLOR. Prima di utilizzare questa modalità, il programmatore dovrà quindi prevedere esattamente ciò che avverrà in seguito. Nelle pagina che segue è illustrata un'automobilina rossa che viaggia su una strada fiancheggiata da alti platani. Visti dal basso, gli alberi in primo piano sembrano più larghi degli altri. Quando l'auto pessa dietro ad un albero in primo piano, sparisce, mentre quando passa davanti ad un albero in secondo piano, resta visibile.



Per realizzare scene di questo tipo in grado di dare il senso di profondità, bisogna innanzitutto creare un'auto con l'aiuto di due caratteri.
Quindi si fisserà un colore per il primo piano e uno o più colori per lo sfondo: nel nostro caso un colore per ogni zona del paesaggio interessato dallo spostamento dell'auto, cioè il cielo e gli alberi.
Gli alberi sono disegnati con caratteri pieni. Dopo l'istruzione CONSOLE 0,24,1 tutti i caratteri stampati sul cielo appariranno in rosso; tutti i caratteri stampati su un albero grande appariranno in arancione su sfondo arancione, per cui saranno invisibili; i caratteri stampati su un albero sottile appariranno in rosso su sfondo arancione.

Programma

```
1'
2'AUTO
3'
20 CLEAR,,2
30 DEFGR$(0)=15,25,49,127,255,255,60,24
40 DEFGR$(1)=224,48,16,254,255,255,30,12
50 AU$=GR$(0)+GR$(1)
60'PAESAGGIO
70 SCREEN 1,12,0:CLS
80 ATTRB 1,1:LOCATE 0,0,0
90 BOXF(0,120)-(319,199),2
100'ALBERI
110 FOR X=6 TO 38 STEP 8
120 BOXF(X,0)-(X+1,14)" ",15,15
130 NEXT X
140 FOR X=2 TO 34 STEP 8
150 BOXF(X,0)-(X,14)" ",1,15
160 NEXT X
170'EFFETTO CAMALEONTE
180 CONSOLE 0,24,1
190 X=0
200 LOCATE X,14:PRINT" "+AU$+" "
210 DO
220 R$=INPUT$(1):R=ASC(R$)
230 IF R=13 THEN EXIT
240 IF R=8 AND X>0 THEN X=X-1 ELSE IF R=9 AND X<33 THEN X=X+1
  ELSE GOTO 220
245 LOCATE X,14:PRINT"$"+AU$+"$"
250 LOOP
260 ATTRB 0,0:LOCATE 0,0:CONSOLE 0,24,0
999 END
```

Commenti

20–50 Definizione dell'auto, AU$ a partire dai due caratteri GR$(0) e GR$(1).
70 Definizione del colore del cielo: rosso su sfondo celeste.
90 Disegno dei prati (senza importanza).
110–130 Disegno degli alberi in primo piano. Riquadri con caratteri "spazio" color arancione su sfondo arancione.
140–160 Disegno degli alberi sullo sfondo. Riquadri con caratteri "spazio" rossi su sfondo arancione. Gli alberi appaiono tutti in arancione.
180 Effetto camaleonte.
200 Visualizzazione dell'auto sulla sinistra.
210 Memorizzazione del codice ASCII del tasto premuto.
220 La pressione di ENTER comporta il ritorno alle condizioni normali.
230 Movimento all'indietro (primo test)
Movimento in avanti (secondo test)
e individuazione di errori nella battitura dei tasti.

Modalità d'uso

1 – Introdurre il programma e battere RUN.
2 – L'auto si sposta con ← e →.
3 – ENTER interrompe l'esecuzione del programma.

Attenzione: sulla riga 240, si ritrova un esempio di due test nidificati IF...THEN...ELSE IF...THEN...ELSE. E' possibile effettuare un test nidificato sia sotto THEN che sotto ELSE. Fare attenzione ai livelli di nidificazione per evitare spiacevoli sorprese in fase di esecuzione del programma. Se il programma si interrompe prima di arrivare alla riga 999, occorre ripristinare i colori dello schermo nel seguente modo:

```
CONSOLE 0,24,0
```

Capitolo 50

Rock and roll

Il rock and roll si basa su tre accordi appartenenti ad una categoria denominata "settima maggiore". In un accordo non è importante solo l'altezza delle note, ma anche l'intervallo che le separa. Così un accordo di settima maggiore può essere rappresentato mediante i numeri 0,4,7,10 che segnalano le distanze, in semitoni, tra le note che lo compongono.

Ecco per esempio l'accordo DO maggiore settima rappresentato generalmente con DO7:

Do	Do#	Re	Re#	Mi	Fa	Fa#	Sol	Sol#	La	La#	Si
0	1	2	3	4	5	6	7	8	9	10	11

Un rock and roll è dato da una successione di dodici misure, ciascuna corrispondente ad un accordo.

Do7	Do7	Do7	Do7	Fa7	Fa7	Do7	Do7	Sol7	Fa7	Do7	Do7

Ogni misura è composta da otto crome. Per ottenere una misura, è sufficiente scegliere casualmente otto note nel corrispondente accordo. Si può notare infine che se si considera l'altezza del primo accordo come base, la successione dei dodici accordi può essere a sua volta codificata in una serie di dodici numeri.

Do7	Do7	Do7	Do7	Fa7	Fa7	Do7	Do7	Sol7	Fa7	Do7	Do7
0	0	0	0	5	5	0	0	7	5	0	0

Per avere un buon risultato, bisogna che la serie di note sia la stessa per ogni accordo. Supponiamo per esempio che il caso determini la serie
4,4,7,0,0,10,7
per le otto note dell'accordo di DO7. Saranno questi stessi intervalli che verranno ripresi per tutti gli altri accordi del pezzo.

Programma

```
1'
2'Rock
3'
10 DATA DO,DO#,RE,RE#,MI,FA,FA# ,SO,SO#,LA,LA#,SI
20 DIM NT$(60)
30 GOSUB 1000
40'
50 DATA 0,4,7,10
60 DATA 0,0,0,0,5,5,0,0,7,5,0,0,99
70 GOSUB 2000:GOSUB 3000
80 PLAY"L12":RESTORE 50
90 FOR K=1 TO 4:READ AN(K):NEXT K
100 GOSUB 4000
110 RESTORE 60
120 READ ACC
130 IF ACC=99 THEN 110
140 FOR X=1 TO 8
150 PLAY NT$(TN+ACC+SM(X))
160 NEXT X
170 GOTO 120
999 END
1000'
1001'Creazione della matrice
1002'
1010 FOR J=0 TO 4
1020 O$="O"+CHR$(49+J)
1030 RESTORE 10
1040 FOR K=1 TO 12
1050 READ T$:NT$(12*J+K)=O$+T$
1060 NEXT K
1070 NEXT J
1999 RETURN
2000'
2001'Inizializzazione della serie di numeri casuali
2002'
2010 PRINT"PREMERE UN TASTO"
2020 Z=RND
2030 IF INKEY$=""THEN 2020
2999 RETURN
3000'
3001'Tono
3002'
3010 PRINT"SCEGLIERE IL TONO TRA 0 E 43:";
3030 INPUT TN
3999 RETURN
4000'
4001'Serie melodica
4002'
```

```
4010 FOR X=1 TO 8
4020 K=1+INT(RND)
4030 SM(X)=AN(K)
4040 NEXT X
4999 RETURN
```

Commenti

- Il sottoprogramma 1000 crea una matrice che contiene le 60 note generabili dal PC128:

Ottave	1	DO	DO#	RE	RE#	MI	FA	FA#	SO	SO#	LA	LA#	SI
	2	DO	DO#	RE	RE#	MI	FA	FA#	SO	SO#	LA	LA#	SI
	3	DO	DO#	RE	RE#	MI	FA	FA#	SO	SO#	LA	LA#	SI
	4	DO	DO#	RE	RE#	MI	FA	FA#	SO	SO#	LA	LA#	SI
	5	DO	DO#	RE	RE#	MI	FA	FA#	SO	SO#	LA	LA#	SI

- Il sottoprogramma 2000 inizializza la successione di numeri casuali.

Il programma principale comincia dalle righe DATA 50 e 60 nelle quali è facile riconoscere le liste di numeri create precedentemente. La serie della riga 60 termina con il numero 99 che indica la fine del pezzo. Il sottoprogramma 3000 richiamato dalla riga 70, dopo il sottoprogramma 2000, permette di scegliere la tonalità del pezzo. Non bisogna superare 43 perchè nell'accordo più alto del pezzo (So17) la nota più alta si trova a 17 intervalli dalla nota più bassa, considerata come punto di partenza: 43=60 – 17.

La riga 80 fissa la lunghezza di tutte le note (crome). E' possibile modificarla a proprio gusto per accelerare o rallentare l'esecuzione.

La riga 100 rimanda al sottoprogramma 4000 che genera la melodia del pezzo. La tabella SM(X) contiene otto numeri scelti a caso tra 0, 4, 7 e 10.
Il ciclo 140–160 è la parte principale del programma: da qui parte l'esecuzione del pezzo. Il ciclo corrisponde alle otto note di un accordo. Il numero di nota della matrice NT$ è calcolato a partire:
- dal tono (TN) scelto dall'utente nel sottoprogramma 3000;
- dal numero dell'accordo (ACC) letto alla riga 120;
- dal numero della X–esima nota della melodia.

Variazioni

Per ottenere altre melodie è necessario modificare i dati inseriti nel programma. Una modifica può consistere nella variazione dei dati relativi agli accordi. Sostituendo, per esempio, l'accordo di settima maggiore con un accordo minore, si ottiene un rock triste. L'accordo minore si ottiene con le note 0, 3, 7, 15. E' sufficiente digitare:

```
50 DATA 0,3,7,15
```

senza modificare alcuna riga di programma.

Appendice

Seguono due programmi BASIC che permettono di salvare e di rileggere la tavolozza colori sul registratore.

Battere quindi sotto il BASIC il seguente programma:

```
10'Lettura del file tavolozza
20'
30 NOME$="CASS:TAVOLOZ.CFG"
40'
50 OPN"I",#1,NOME$
60 FOR COLORE=0 TO 15
70 TAVOLOZZA COLORE,CVI(INPUT$(2,#1))
80 NEXT
90 CLOSE
100 END
```

e salvarlo con SAVE"LETTAV.

Si introduca quindi il seguente programma:

```
10'Scrittura del file tavolozza
20'
30 FOR COLORE=0 TO 15
40 A$=A$+MKI$(TAVOLOZZA(COLORE))
50 NEXT COLORE
60'
70 OPEN"O",#1,"CASS:TAVOLOZ.CFG
80 PRINT #1,A$
90 CLOSE #1
100 END
```

e salvarlo con SAVE"SALVTAV.

Par salvare la tavolozza, riavvolgere la cassetta e battere: RUN"SALVTAV. La tavolozza verrà salvata in un file chiamato TAVOLOZ.CFG. Per rileggere la tavolozza, riavvolgere la cassetta e battere RUN"LETTAV.

Generalità

Parte II: Guida di Riferimento

1. Avviamento

Per utilizzare l'interprete del Basic 128, basta accendere il PC 128 e rispondere 1 al menu visualizzato. Appare il messaggio:

```
BASIC 128 v1.0 (c) Microsoft 1985
110127 bytes free
OK
```

l'interprete Basic è attivo.

L'interprete del Basic 128 è interamente residente. Sono disponibili tutti i comandi, le istruzioni e le funzioni, compresi quelli che permettono di gestire dei file su dischetti.
Le risposte 1 o 2 al menu comportano il caricamento, a partire da un drive, del programma AUTO.BAT, se esiste, e la relativa esecuzione.

2. Tastiera e editor integrato

L'interprete Basic è fornito di un editor a schermo pieno che semplifica notevolmente la scrittura e la modifica dei programmi. E' possibile modificare una riga visualizzata senza doverla ribattere. Dopo la modifica, la pressione del tasto ENTER, reintroduce la nuova riga nel programma.

Tastiera

I tasti specifici della tastiera sono i seguenti:

STOP	Ferma l'esecuzione del programma in corso. Niente viene modificato sullo schermo. La ripresa viene effettuata, premendo un tasto qualsiasi, tranne CTRL , . e STOP.
CTRL	E' utilizzato in combinazione con un altro tasto:
CTRL–C	Interrompe l'esecuzione del programma o del comando in corso. Tale interruzione interviene solamente alla fine dell'istruzione in corso, che può durare per un certo periodo nel caso di un'istruzione PLAY. Interrompe la numerazione automatica comandata con AUTO. La riga in corso non viene memorizzata.
CTRL–X	Cancella i caratteri, a partire dalla posizione del cursore fino alla fine della riga.

CTRL–W	Crea un legame logico tra la riga sulla quale si trova il cursore e quella successiva. Tale agevolazione viene utilizzata per riunire due righe di programma in una sola.
	Spostamento del cursore.
CLS HOME	Pone il cursore in alto a sinistra.
INS	Agisce sulla modalità di inserimento. In modalità inserimento, il carattere che si trova sopra il cursore viene rappresentato in inversione video. I caratteri battuti in seguito vengono visualizzati immediatamente a sinistra del cursore. Una seconda pressione del tasto INS riporta alla modalità normale.
EFF	Elimina il carattere alla posizione del cursore.
DEL	Elimina il carattere posto a sinistra del cursore.

Modifica di una riga

Una riga di programma o una riga di comandi o di istruzioni in modalità diretta può essere composta da un massimo di 255 caratteri. Durante la battitura dei caratteri, quando il cursore arriva al bordo destro dello schermo automaticamente ritorna all'inizio della riga successiva. Per modificare una riga presente per intero sullo schermo, si posizioni il cursore sulla parte da correggere e battere le correzioni sia scrivendo sui caratteri da modificare, sia eliminando o inserendone altri o anche eseguendo tutte e tre le operazioni.
La nuova riga viene memorizzata premendo il tasto ENTER senza dover riportare il cursore alla fine della riga.

Creazione e modifica di un programma

Ogni riga che inizi con un numero compreso fra 0 e 63999 è considerata una riga di programma. La sua introduzione nel programma viene effettuata premendo il tasto ENTER. Se esiste già una riga con lo stesso numero, questa verrà sostituita dalla nuova riga.
Se quest'ultima è data solo dal numero seguito o meno dal carattere "spazio" la riga viene eliminata.

Al momento dell'introduzione di un programma è possibile ottenere una numerazione automatica delle righe, con il comando AUTO.

La visualizzazione di tutto o di una parte del programma presente in memoria è ottenuta tramite il comando LIST. Si può effettuare la modifica di una riga preesistente riscrivendola o modificandola direttamente nel caso sia visualizzata.
L'eliminazione di una parte delle righe del programma è possibile con il comando DELETE.
La soppressione di tutto il programma viene eseguita dal comando NEW.

3. Modalità di funzionamento

Quando l'interprete Basic è attivo, appare il messaggio"OK". In tale situazione, si può utilizzare il Basic in due modi diversi:

• Modalità di esecuzione diretta

Tale modalità permette di effettuare calcoli, di assegnare dei valori alle variabili, di richiamare delle funzioni oppure di eseguire istruzioni e comandi.

Si termina ogni riga con ENTER e si ottiene subito il risultato.

Questa modalità è utile nei calcoli quanto una moderna calcolatrice, nella prova di alcune istruzioni o nella consultazione di determinati valori, quando si verifica un arresto di programma.

Esempio:

```
PRINT 12*12
144
OK
P=21.95
OK
PRINT P*1.1
24.145
OK
```

• Modalità di programma

Per introdurre in un programma o per modificarlo, è necessario iniziare ogni riga con un numero. Il numero di riga deve essere fra 0 e 63999, compresi. In questo caso, la riga viene memorizzata e inserita nel programma presente in memoria.

Si ottiene l'esecuzione con il comando RUN.

Una riga di programma può contenere istruzioni, comandi o commenti. Tuttavia l'esecuzione di alcuni comandi interrompe lo svolgimento del programma.

Esempio:

```
10 PRINT 31*24
RUN
744
OK
```

4. Struttura di una riga

Una riga di programma Basic deve avere il formato seguente:

numero istruzione: istruzione istruzione . .

Il numero di riga è un valore intero compreso fra 0 e 63999. La riga di programma deve comprendere almeno un'istruzione. Se vi sono molteplici istruzioni, i 2 punti (:) separano le une dalle altre.
La lunghezza della riga non deve superare i 255 caratteri. E' opportuno tenere presente che le lettere minuscole accentate vengono codificate su tre caratteri.
I caratteri che seguono l'istruzione REM o il carattere apostrofo (') che è il suo equivalente, non sono considerati istruzioni e possono essere utilizzati come commenti. E' possibile aggiungere dei commenti alla fine di una riga dopo il carattere apostrofo (') o dopo l'istruzione REM. Gli spazi sono facoltativi tranne quando una variabile precede una parola chiave del linguaggio. Può esistere un numero qualsiasi di spazi fra una parola chiave, una costante o una variabile. Le parole chiave e le variabili possono essere scritte indifferentemente maiuscole o minuscole senza accenti. L'interprete Basic trasforma automaticamente le minuscole in maiuscole.

Esempio:

```
10 REM Conto da 1 a 10
20 FOR I=1 to 10
30 PRINT:'visualizza il numero
40 next I
50 PRINT"fine":END
```

5. Alfabeto usato

L'interprete Basic usa il set di caratteri ASCII che è costituito da 96 caratteri visualizzati, il cui codice è compreso fra 32 a 127. I primi 32 caratteri possono essere impiegati come dati introducendoli direttamente da tastiera (tasto CTRL + lettera) o con la funzione CHR$.

I codici 128 e 255 sono destinati a ricevere i caratteri definiti dall'utente per mezzo dell'istruzione DEFGR$ e inoltre possono essere chiamati dalla funzione GR$(n) oppure CHR$(128+n) con n variante da 0 a 127.

6. Le costanti

Esistono due tipi di costanti:
– le costanti numeriche
– le costanti stringa.

Costanti numeriche

Le costanti numeriche sono numeri positivi o negativi. Vi sono tre tipi di costanti diverse a seconda della precisione.

Interi

Tutti i numeri interi positivi o negativi compresi fra –32768 e +32767 esempi:
13
–5273

Numeri raali a precisione singola

Sono tutti quei numeri positivi o negativi rappresentati nel formato m*10^e dove la mantissa m ha sette cifre significative e l'esponente è compreso fra –38 e +38. In pratica la precisione esatta del numero varia fra 6 e 7 cifre. Si può precisare, ma non è obbligatorio, che una costante è a precisione singola introducendo il segno ! dopo la costante.
esempi:
–1.6 E–19
6.02 E23
2.7828
12345!

Numeri reali a doppia precisione

Sono identici ai numeri reali a precisione singola ma con una mantissa di diciassette cifre significative. Nella notazione di un numero a precisione doppia la lettera che segna l'esponente è D invece di E. Quando non c'è l'esponente, è possibile precisare che una costante è a precisione doppia introducendo il segno # dopo la costante.
esempi:
3.1415926535#
.5432: D–27

Notazione delle costanti numeriche

Le costanti numeriche si scrivono sia sotto forma decimale (12.3456) sia sotto forma "scientifica" (.123456E2). I numeri interi possono essere scritti usando una base non decimale.

Notazione esadecimale
Una costante esadecimale è un numero espresso in base 16. Deve essere preceduta dai caratteri &H. Le cifre e i caratteri utilizzati sono: 0,1,2,3,...,9,A,B,C,D,E,F.
esempi:
&HFFFF equivalente a 65535
&H32F

Notazione ottale
Una costante ottale è un numero espresso in base 8. Deve essere preceduta dai caratteri &O oppure solo da &. Le cifre utilizzate sono 0,1,2,3,4,5,6,7.
esempi:
&O622
&1234

Notazione binaria
Una costante binaria è un numero espresso in base 2, rappresentato quindi solo da 0 e da 1. Deve essere preceduta dai caratteri &B. Questa notazione è particolarmente interessante per la rappresentazione dei bit di uno o due byte, specialmente allorché si utilizzino operatori logici (NOT, AND, OR, XOR, IMP, EQV).
esempi:
&B10101010
&B100000001

Costanti stringa

Una costante stringa è un insieme di caratteri racchiusi fra virgolette ("). Il numero dei caratteri della stringa, chiamato lunghezza della stringa deve essere compreso fra 0 e 255. Se una stringa non contiene alcun carattere è una stringa nulla o vuota. Tutti i caratteri (dal codice 0 al codice 255) possono essere inclusi in una stringa. E' bene osservare che le minuscole accentate occupano lo spazio di tre caratteri; è necessario tenerne conto per quanto riguarda il calcolo della lunghezza.
esempi:
"COME VORREI ANDARE AL MARE"
"La portinaia è sulle scale"

7. Variabili

Le variabili sono posizioni di memoria che posseggono un nome proprio e che sono destinate a ricevere i diversi dati utili allo svolgimento di un programma. Una variabile può ricevere il valore di una costante o il risultato di un calcolo o di una trasformazione di costanti e di variabili. Vi sono quattro tipi di variabili corrispondenti ai tipi di costanti:
– variabili numeriche intere
– variabili numeriche a precisione singola
– variabili numeriche a doppia precisione
– variabili stringa

Nome delle variabili

I nomi delle variabili sono formati da almeno una lettera, seguita eventualmente da lettere e da cifre. Il nome di una variabile può essere composto da un massimo di

255 caratteri. Tuttavia solo i primi sedici sono significativi e serviranno a distinguere tra di loro le variabili. Quindi due variabili i cui nomi hanno i primi sedici caratteri identici, sono considerati uguali. Il nome di una variabile non può cominciare con una parola chiave del Basic (queste parole la cui lista è data nell'appendice 5, sono dette parole riservate). Quindi TOTALE, FORZA, VALORE non possono essere nomi di variabili perchè iniziano con TO, FOR, VAL.

Esempio:

SOMMA
A2
RESOCONTODELLESPESEDELMESE

Il nome di una variabile deve essere seguito da uno dei caratteri %, #, $ per precisarne il tipo (intera, doppia precisione o stringa). Il carattere ! può essere usato per indicare una variabile a precisione singola. Ogni variabile che non termina con %, #, $ viene considerata a precisione singola. Il significato di questi caratteri è il seguente:

%	variabile numerica intera
!	variabile numerica a precisione singola
#	variabile numerica a precisione doppia
$	variabile stringa

esempi:

QUOZIENTE	variabile precisione singola
A3!	variabile precisione singola
PI#	variabile precisione doppia
NOBJET%	variabile intera
AOR$	variabile stringa

Si può precisare il tipo di una o più variabili anche con una delle istruzioni DEFINT, DEFSNG, DEFDBL, DEFSTR.

Matrici

Il Basic permette di raggruppare un gran numero di variabili dello stesso tipo sotto lo stesso nome all'interno di una matrice. Quest'ultima può avere una o più dimensioni. Ogni variabile è accessibile, attribuendo un valore a ciascun indice e costituisce un elemento della matrice. A volte si utilizza il termine variabile "indexata". Tutte le matrici che hanno più di 10 elementi vanno dichiarate prima di tutto con l'istruzione DIM.

esempio:

DIM T(12)

Tale istruzione crea una matrice numerica di tredici elementi a una dimensione. Infatti, l'indice varia dal valore 0 al massimo indicato da DIM. T(3) indica il quarto elemento della matrice T.

DIM PI'S(2,10)

Questa istruzione crea una matrice stringa a due dimensioni. tl primo indice varia da 0 a 2, il secondo da 0 a 10. Quindi comprende 33 elementi.
La sintassi del Basic accetta matrici che hanno fino a 255 indici, ognuno dei quali può variare da 0 a 32767.
Se la valutazione di un indice porta ad un valore non intero, questo viene arrotondato al numero intero più vicino.

esempi:

A(3,5) è equivalente a A(4)
A(3.49) è equivalente a A(3)

I nomi delle matrici obbediscono alle medesime regole delle variabili dello stesso tipo:

%	intero
!	precisione singola
#	precisione doppia
$	stringa.

Spazio occupato da una variabile o da una matrice

Ogni tipo di variabile corrisponde ad una precisa modalità di rappresentare un valore in memoria. Lo spazio occupato in funzione del tipo della variabile è il seguente:

intero	2 byte
precisione singola	4 byte
doppia precisione	8 byte
stringa	lunghezza della stringa in byte

Quindi, per una matrice, bisogna moltiplicare lo spazio occupato da un elemento per il numero totale di elementi della matrice.
esempio:
DIM A#(19) fissa una matrice numerica a precisione doppia i cui valori occuperanno 20*8=160 byte in memoria. Le dimensioni di una matrice sono limitate a 64 Kbyte.

Conversione

Non bisogna assolutamente mescolare numeri e stringhe in un'espressione.
Il Basic accetta espressioni numeriche "miste", cioè comprendenti variabili e costanti numeriche di diverso tipo. Dunque la valutazione di un'espressione viene eseguita, se c'è spazio, convertendo i valori nel tipo più preciso.
esempi:

?12/23.5
.510638

Il numero intero 12 verrà convertito a singola precisione e la divisione verrà eseguita a singola precisione.

```
?2*3.141592653#
6.283185306
```

Dopo la conversione del 2 a precisione doppia, il calcolo ed il risultato sono a doppia precisione.

Le operazioni aritmetiche vengono anch'esse eseguite a doppia precisione se uno degli operandi è formato da più di sette cifre significative oppure se è annotato a precisione doppia.

Le funzioni matematiche del linguaggio (SIN, ATN, LOG,...) vengono calcolate a precisione singola se l'argomento relativo è a precisione singola, e a precisione doppia, allorchè il loro argomento sia a doppia precisione.
esempio:

```
?LOG(2.0001)
.693197
?LOG(2.0001#)
.6931471793099987
```

E' necessario considerare molteplici casi, quando si attribuisce un valore ad una variabile di diversa precisione:

- Il valore è di precisione superiore
Viene convertito alla precisione della variabile per arrotondamento (non per troncamento) alla cifra intera.
esempio:

```
A%=251.72
?A%
252
```

-il valore è di precisione inferiore
Nel caso di un numero intero, viene convertito pur mantenendo intatto il suo valore. Invece, un numero a singola precisione attribuito ad una variabile a doppia precisione viene completato da cifre non significative ma tuttavia non nulle.
esempio:

```
R#=.12345678E1
?R#
1.2345672845840045
```

Il valore non subisce assolutamente alcun cambiamento. E' possibile riempire le ultime cifre significative con degli 0 effettuando la conversione dei numeri in stringhe di caratteri.
esempio:

```
R#=VAL(STR$(.12345678E1))
?R#
1.234567
```

8. Espressioni

Espressioni numeriche

Un'espressione numerica può essere una costante, una variabile oppure una combinazione di costanti, variabili e funzioni legate fra di loro per mezzo di operatori.
Gli operatori numerici vengono classificati in tre categorie:
– operatori artimetici
– operatori relazionali
– operatori logici.

Operatori arltmetici

Ecco la lista degli operatori aritmetici classificati in ordine di priorità decrescente:

^	elevamento a potenza	X^Y
–	negazione	–X
*/	moltiplicazione, divisione	X*X,X/Y
@	divisione intera	X@Y
MOD	modulo o resto della divisione intera	X MOD Y
+–	addizione, sottrazione	

La priorità fra gli operatori indica che in un'espressione dove sono presenti due operatori in assenza di parentesi, quello prioritario viene considerato per primo. A uguale priorità, la valutazione comincia da quello di sinistra.
esempio:

```
?4*3^2
36
?(4*3)^2
144
```

Divisione Intera @

Tale operazione può essere effettuata solamente se gli operandi sono compresi fra –32768 e 32767. Prima della divisione, i due operandi vengono arrotondati. Il risultato è il quoziente troncato della divisione.
esempio:

```
?10@3
3
?20@4.5
4
?99@50
1
```

Alcuni casi di errori

- Se un calcolo conduce al superamento della capacità nel tipo considerato (valore intero inferiore a –32768 o superiore a 32767, valore reale inferiore a 10 –38 o superiore a 10^38) appare il messaggio di errore Overflow.
- Una divisione per zero viene segnalata da:
Division by zero
In entrambi i casi, l'esecuzione viene interrotta.

Operatori relazionali

Gli operatori relazionali sono utilizzati per confrontare due valori. Il risultato del confronto è un valore "booleano" VERO o FALSO. Per convenzione, VERO ha come valore –1 e FALSO 0.

Gli operatori relazionali sono i seguenti:

=	uguale a	X=Y
<>	diverso da	X<>Y
<	minore di	X<Y
>	maggiore di	X>Y
<=	minore o uguale a	X<=Y
>=	maggiore o uguale a	X>=Y

Si può utilizzare il risultato di una relazione in una istruzione di collegamento:

```
IF X=Y<A*B THEN GOSUB 200
```

E' di basilare importanza distinguere l'operatore relazionale destinato a controllare l'uguaglianza di due valori, dall'istruzione = che attribuisce un valore ad una variabile.
esempio:

```
IF A=3 THEN B=2
```

significa:
Se il valore di A è 3 allora attribuire 2 a B.

Poichè il risultato di un confronto è un valore "booleano" rappresentato da un numero intero (0 o –1), può essere attribuito ad una variabile numerica.
esempio:

```
RIS=2^3<3^2
?RIS
-1
IF RES THEN?"VERO"
VERO
```

Operatori logici

Gli operatori logici offrono la possibilità di effettuare operazioni booleane e quindi di combinare molteplici relazioni in una condizione.
Sono disponibili sei operatori logici:

NOT	NON logico
AND	E logico
OR	O logico
XOR	O esclusivo
IMP	implicazione
EQV	equivalenza

NOT è l'unico operatore che può essere applicato ad un solo operando. Tenendo conto che V sta per VERO e F per FALSO, la tabella degli operatori è la seguente:

X	Y	NOT X	X AND Y	X OR Y	X XOR Y	X IMP Y	X EQV Y
V	V	F	V	V	F	V	V
V	F	F	F	V	V	F	F
F	V	V	F	V	V	V	F
F	F	V	F	F	F	V	V

Notare che:
– l'espressione X IMP Y è equivalente a (NOT X) OR Y
– l'espressione X EQV Y è equivalente a NOT(X XOR Y)
esempio:

```
IF A=B AND C=D THEN 500
IF NOT(A OR B)THEN 400
```

Funzionamento degli operatori logici

In pratica, gli operatori logici lavorano su numeri interi (compresi fra –32768 e 32767), cioè su serie di sedici bit. Le operazioni vengono effettuate sui bit la cui posizione è la stessa (un bit a 1 è VERO, un bit a 0 è FALSO).
esempio:

```
?127 AND 16
16
```

127 è rappresentato in binario da 1111111
16 è rappresentato in binario da 10000
il risultato vale dunque 10000

Questa particolarità può essere sfruttata per ricavare contemporaneamente il valore di un bit in uno o due byte grazie ad una maschera.
La maschera &B11110000 o 240 in decimale permette di ricavare con AND i quattro bit significativi di un byte. Per sapere se un numero X% è pari, si può battere:
X%AND&B1111111111111110=X%
Il risultato vale –1 (VERO) se X% è pari, 0 (FALSO) se X% è dispari.

Priorità delle operazioni

Quando si deve valutare un'espressione numerica, l'ordine di priorità di valutazione è il seguente:
1 - funzioni (ABS, SIN, INT,...)
2 - elevamento a potenza ^
3 - negazione -
4 - moltiplicazione, divisione *,/
5 - divisione intera @
6 - modulo MOD
7 - somma, differenza +, -
8 - relazioni <,>,=,<=,>=,<>
9 - operazioni logiche NOT, AND, OR, XOR, EQV, IMP
E' possibile modificare l'ordine di priorità con delle parentesi; le operazioni poste nelle parentesi più interne vengono valutate per prime.

Funzioni numeriche

Le funzioni numeriche disponibili (argomento numerico, risultato numerico) sono le seguenti:

ABS(x)	valore assoluto di x
ATN(x)	arcotangente di x
CDBL(x)	conversione a doppia precisione
CINT(x)	conversione in intero
COS(x)	coseno di x
CSNG(x)	conversione a precisione singola
EXP(x)	esponenziale di x
FIX(x)	parte intera di x
INT(x)	numero intero più grande minore o uguale a x
LOG(x)	logaritmo neperiano di x
MAX(x,y,...)	massimo di x,y,...
MIN(x,y,...)	minimo di x,y,...
RND(x)	generatore di numeri aleatori
SGN(x)	segno di x
SIN(x)	seno di x
SQR(x)	radice quadrata di x
TAN(x)	tangente di x

Espressioni stringa

Un'espressione stringa può essere una costante, una variabile oppure una combinazione di costanti, variabili e funzioni legate fra di loro per mezzo di operatori.
Esiste un solo operatore sulla stringa di caratteri:
+concatenazione
Questo operatore permette di collegare l'una all'altra due stringhe per formarne una terza.

Esempi:

```
?"AUTO"+"STRADA"
AUTOSTRADA
A$="ESSERE":B$="O":C$="NON"
?A$+B$+C$+A$
ESSERE O NON ESSERE
```

La concatenazione non introduce spazi tra le stringhe. Si dovranno quindi prevedere degli spazi alla fine delle stringhe singole per comporre una frase.

Confronto di stringhe di caratteri

Gli operatori relazionali possono essere utili per confrontare stringhe di caratteri, ma il risultato del confronto è logico, quindi espresso sotto forma numerica: –1 (VERO) o 0 (FALSO). Il confronto viene eseguito carattere per carattere, a partire dall'inizio di ogni stringa. Un carattere viene considerato inferiore ad un altro se il suo codice nel set di caratteri ASCII è inferiore. Finchè i caratteri sono uguali, il confronto può proseguire. La stringa che ha il primo carattere inferiore al carattere corrispondente nell'altra stringa è considerata la più piccola. Se si raggiunge la fine di una delle due stringhe prima che sia apparsa una differenza, la stringa più corta è considerata la più piccola. In un'operazione di confronto, si prendono in considerazione tutti i caratteri, compresi gli spazi ed i caratteri non visualizzabili.
esempi:

"AB"<"FG" perchè A è inferiore a F
"ABCD"<"ABCDEF" perchè "ABCD" è più corta
"GRANDE"<"GROSSO" perchè A è inferiore a O
"LUIGI"<"Luigi" perchè U è inferiore a u

Le espressioni stringhe possono anche includere funzioni il cui risultato è una stringa (LEFT$, MtD$,...).

9. Input–output e file

Input–output generatizzati

La gestione degli scambi con le principali periferiche (cassette, dischetti, tastiera, schermo, stampante e comunicazione seriale) viene esplicata dallo stesso set di istruzioni ad uso generale. Tali istruzioni permettono di gestire i trasferimenti riguardanti programmi, dati sotto forma di codici ASCII e dati in formato binario. L'orientamento delle informazioni verso l'una o l'altra periferica viene indirizzato scegliendo il nome della periferica, le sue opzioni di funzionamento e il nome del file da trasferire. Questo insieme costituisce il descrittore di file.

Descrittore di file

Il descrittore di file è una stringa che comprende tre parti organizzate nel modo seguente:

Nome della periferica: (opzioni) nome del file

A seconda dei casi, ognuna di queste tre parti può essere facoltativa.

Nome della periferica e opzioni

I nomi delle periferiche sono i seguenti:

0	1° drive o QDD
1	2° drive
2	3° drive
3	4° drive

CASS	registratore
COMM	dispositivo di trasmissione seriale
KYBD	tastiera
LPRT	stampante parallela
SCRN	schermo

Il nome della periferica, quando quest'ultima è presente, deve sempre essere seguito da due punti (:). Se non si precisa il nome della periferica, il suo valore è per default:

SCRN:	con il comando LIST e TRON.
0:	se è collegato un drive, per tutti gli altri comandi e istruzioni
CASS:	se è collegato solo il registratore.

L'istruzione DEVICE permette di modificare la periferica di default in un'istruzione di input e output.

Opzioni

Quattro periferiche accettano opzioni: la stampante parallela (LPTR:), il dispositivo di trasmissione seriale (COMM:), i drive (0: 1: 2: 3: 4:) e il registratore.

Stampante parallela:
L'opzione indica il numero di caratteri per riga: fra 0 e 255.
Il numero di default è 40. 0 indica una riga infinita, esempio:
LPTR:(80) indica una stampante a 80 colonne.

Dispositivo di trasmissione seriale:
L'opzione deve essere composta obbligatoriamente da tre parti.
- la prima cifra indica la velocità di trasmissione

1	110 baud
2	300 baud
3	600 baud
4	1200 baud
5	2400 baud

6 4800 baud
7 9600 baud
8 19200 baud

– la seconda cifra indica il numero di bit trasmessi

7 trasmissione a 7 bit
8 trasmissione a 8 bit

la parità non viene gestita quando si effettua una trasmissione seriale; deve essere quindi calcolata a partire dal numero di bit trasmessi.
– la terza cifra e la successiva indicano il numero di caratteri per riga (da 0 a 255) come per la stampante paralleta, tenendo conto che 0 corrisponde ad una riga infinita. Un codice di ritorno a capo viene emesso all'inizio del numero di caratteri indicati. La copia dello schermo si deve effettuare con una riga infinita (valore 0).
esempio:
COMM:(47120) indica una trasmissione a 1200 baud (4) a 7 bit (7) con 120 caratteri per riga (120).

Drive
L'opzione è un commento associato al file, quando quest'ultimo viene scritto sul disco. Tale commento è una stringa composta da un massimo 8 caratteri e viene visualizzato nella parte destra dello schermo, quando si richiede il catalogo del disco (DIR).
esempio:
SAVE"(13 maggio)TEST"
il commento 13 maggio è associato al file TEST.BAS.

Registratore:
CASS:(1) imposta la velocità di scrittura a 1200 baud.
CASS:(2) imposta la velocità di scrittura a 2400 baud.

Nome dei file

Il nome di un file è formato da due parti separate da un punto (.):
nome.estensione
Il nome comprende al massimo 8 caratteri: sono permessi tutti i caratteri, tranne il codice 0, il codice 255, le due parentesi ed il punto. L'estensione comprende al massimo tre caratteri che hanno le stesse limitazioni del nome. L'estensione non è obbligatoria.
esempi:
SCACCHI.GIO
Euclide.tst
PROVA
Se non vi è alcuna estensione, l'interprete Basic aggiunge il seguente suffisso:

.BAS per un programma Basic
.DAT per un file di dati
.BIN per un file binario (area memoria o programma in linguaggio macchina)
.MAP per un'immagine dello schermo
.TRA per una traccia di esecuzione del programma

Esiste un solo nome del file che ha un uso particolare:
AUTO.BAT
Questo nome è riservato al programma Basic che si esegue automaticamente al caricamento del Basic (scelta 1 del menu iniziale).

Altre periferiche

Dispositivi audio

La generazione di un suono può essere programmata in Basic, all'interno di una gamma di cinque ottave.
I suoni trasmessi all'altoparlante del televisore costituiscono le note della gamma (con diesis e bemolle) e la pausa.
Per ogni nota che viene emessa, è possibile regolare: ta durata (dalla semibreve alla biscroma puntata), il tempo (su una gamma di 256 valori) e l'attacco che rende il suono più o meno smorzato.
I suoni vengono definiti all'interno di stringhe di caratteri. L'ordine di emissione verso l'altoparlante è dato dall'istruzione PLAY.

La penna ottica

La penna ottica consiste in un lettore luminoso e in un interruttore che fissa il momento della lettura.
Il tempo trascorso fra l'inizio dell'emissione luminosa e la percezione dell'irradiamento permette di determinare con la massima precisione il punto letto, che è uguale al punto elementare dell'immagine.
La rilevazione delle coordinate del punto letto dalla penna ottica viene eseguita con due istruzioni:

INPEN	rilevato a distanza
INPUTPEN	rilevato nel momento di contatto dell'interruttore

La funzione PTRIG può rilevare la posizione dell'interruttore. Mirando una zona con la penna ottica, è possibile effettuare il controllo di una sequenza di istruzioni, se questa zona è stata definita con l'istruzione PEN e se vengono eseguiti i relativi collegamenti tramite ONPEN...GOTO o ONPEN...GOSUB.
I colori nero e rosso non permettono di leggere un punto con precisione. Di norma, la luminosità deve raggiungere un valore minimo, perchè la penna ottica possa funzionare. Per ottenere una misurazione precisa, è necessario regolare la penna ottica all'avviamento.

Joystick

Al controllore di gioco possono essere collegati due joystick. La posizione di ciascun joystick è data da due funzioni:

STICK	posizione del joystick in una delle 8 direzioni
STRIG	stato del pulsante di attivazione del joystick

Mouse

Il mouse viene collegato alla presa sinistra del joystick.
MTRIG stato del pulsante

10. Visualizzazione: modalità carattere e modalità grafica

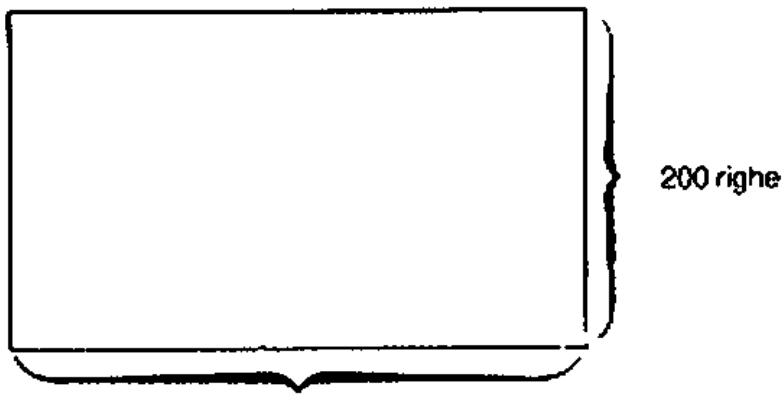
La visualizzazione dei dati sullo schermo può essere effettuata in due modi diversi:
– *modalità carattere:* per visualizzare testo o simboli prestabiliti delle dimensioni di un carattere,
– *modalità grafica:* per visualizzare righe, punti, riquadri o oggetti qualsiasi.
Le due modalità funzionano simultaneamente, ed è quindi possibile visualizzare sullo stesso schermo una parte di testo e una di grafica.

Funzionamento della visualizzazione

A seconda della precisione e del numero dei colori desiderati, è possibile visualizzare caratteri e immagini grafiche seguendo tre modalità distinte:
1 – modalità standard 40 colonne
2 – modalità 80 colonne
3 – modalità matrice di punti a 4 o 16 colori
La modalità standard a 40 colonne corrisponde a quella dell'avviamento. La scelta di una particolare modalità viene fatta con l'istruzione CONSOLE (quinto parametro).

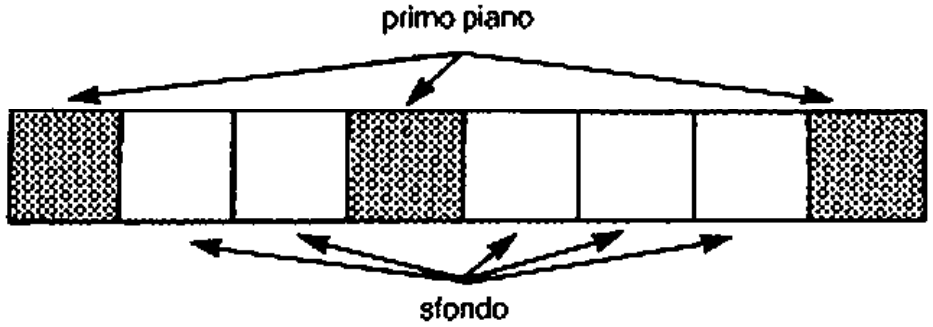
Modalità standard a 40 colonne

La parte utile dello schermo è composta da 200 righe distinte. Ogni riga è divisa in 40 segmenti di 8 punti ciascuna.



All'interno di un segmento, ogni punto è riconducibile ad una delle due categorie seguenti:
- primo piano: si tratta di un punto che appartiene ad un carattere, ad una riga, ad un riquadro o ad un oggetto.
- sfondo: è un punto dello sfondo dello schermo, che non è stato cambiato nè da alcuna riga nè da alcuna visualizzazione, o che è stato impostato come sfondo.

Esempio di un segmento con 3 punti di primo piano



Si possono attribuire solo due colori ad uno stesso segmento:
- un colore per i punti del primo piano,
- un colore per i punti dello sfondo.

I due colori di un segmento (primo piano e sfondo) sono quelli che sono stati stabiliti dall'ultima programmazione di un punto di primo piano o di sfondo del segmento.

Se nessun punto del segmento è stato modificato da un'istruzione di visualizzazione o di tracciamento, tutto il segmento appartiene allo sfondo dello schermo.

L'appartenenza di un punto al primo piano o allo sfondo durante un'operazione di tracciamento è fissata dal codice usato per il colore:
- codice positivo: il punto appartiene al primo piano,
- codice negativo: il punto appartiene allo sfondo.

Se la programmazione di un punto dello schermo viene eseguita con un'istruzione nella quale non è precisato il colore, il punto viene considerato come un punto di primo piano ed il colore di default è quello scelto precedentemente per i caratteri (istruzioni SCREEN e COLOR).

Da un segmento di 8 punti all'altro, la scelta dei due colori è completamente libera. I segmenti sono indipendenti. E' possibile visualizzare e tracciare senza modificare il colore di primo piano dei segmenti realizzati con il tracciato (terzo parametro dell'istruzione CONSOLE).

E' possibile effettuare una inversione simultanea dei colori di primo piano e di sfondo su tutto lo schermo (quarto parametro dell'istruzione SCREEN).

Inoltre è possibile un'inversione della parte rimanente da visualizzare (terzo parametro dell'istruzione COLOR).

Modalità a 80 colonne

In questa modalità, la precisione orizzontale dello schermo è doppia: ogni riga è divisa in 80 segmenti di 8 punti ciascuno. Sono disponibili solo due colori; una linea viene quindi tracciata senza tener conto del colore.

Modalità a matrice di punti a 4 e 16 colori

In questa modalità la definizione dello schermo è di 200 righe di 320 punti in 4 colori e di 200 righe di 160 punti in 16 colori.

Non esiste più alcuna nozione di segmento ed essendo ogni punto completamente indipendente da quelli vicini può assumere qualsiasi colore.

Scelta dei colori

Seguendo la modalità di funzionamento, il numero dei colori utilizzabili simultaneamente è sedici (modalità a 40 colonne) due, quattro o sedici (modalità a matrice di punti).

Questi colori possono essere scelti fra una varietà di 4096 tinte reali diverse.

L'attribuzione di un colore reale ad un codice di colore viene eseguita dall'istruzione PALETTE.

All'avviamento il codice dei colori è attribuito ai seguenti colori reali:

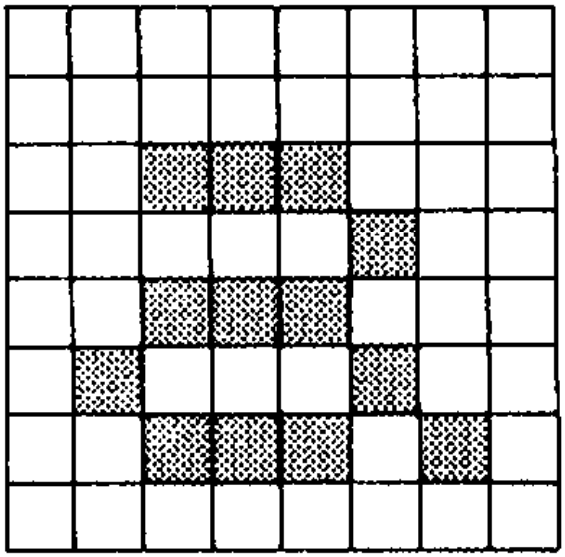
colore	codice primo piano	codice sfondo
nero	0	– 1
rosso	1	– 2
verde	2	– 3
giallo	3	– 4
blu	4	– 5
magenta	5	– 6
azzurro	6	– 7
bianco	7	– 8
grigio	8	– 9
rosso chiaro	9	–10
verde chiaro	10	–11
giallo chiaro	11	–12
blu chiaro	12	–13
magenta chiaro	13	–14
azzurro chiaro	14	–15
arancione	15	–16

I colori chiari (da 8 a 14) sono i primi 7 con l'aggiunta di un po' di bianco.

Il codice di sfondo di un colore è facilmente ottenibile con la relazione:
codice sfondo=– (codice primo piano+1).

Modalità carattere

Le istruzioni sulla modalità carattere prevedono uno schermo di 25 righe di 40 (o 80) caratteri ciascuna. Ogni carattere è rappresentato a sua volta sullo schermo da una matrice di 8*8 punti, e cioè ancora otto segmenti.



matrice del carattere a

Nella modalità carattere, le righe si contano da 0 (riga superiore) a 24 (riga inferiore dello schermo) e le colonne si contano da 0 (la colonna più a sinistra) a 39 o 79 (la colonna più a destra).

L'istruzione LOCATE permette di posizionare una stringa di caratteri in qualsiasi punto dello schermo.

La posizione del cursore che indica il punto in cui avverrà la prossima visualizzazione è data da due funzioni: CSRLIN per il numero di riga e POS per il numero di colonna.

Il codice ASCII del carattere situato in una posizione qualsiasi dello schermo è dato dalla funzione SCREEN (da non confondere con l'istruzione che porta lo stesso nome).

La finestra di visualizzazione può essere limitata in altezza da una linea superiore e da una linea inferiore con l'istruzione CONSOLE (primo e secondo parametro).

I caratteri possono essere visualizzati con altezza doppia e con larghezza doppia servendosi dell'istruzione ATTRB.

Alcune istruzioni grafiche funzionano in modalità carattere, e cioè compiono lo stesso tracciato ma assumono un carattere al posto di un punto con le coordinate dalla modalità carattere (colonne da 0 a 39 o 79 e righe da 0 a 24). Si tratta delle istruzioni PSET, LINE, BOX, BOXF. I caratteri possono essere usati per riempire un'area dello schermo in modalità grafica. Il carattere è allora considerato come un modello di riempimento (istruzione PATTERN).

Caratteri definiti dall'utente

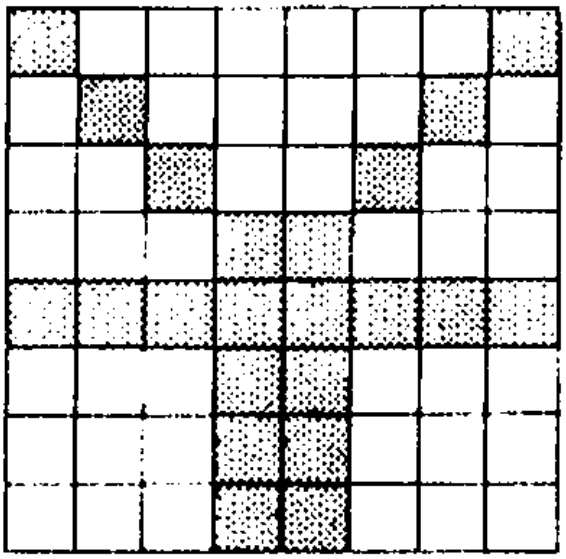
E' possibile definire 128 caratteri con forme diverse che possono essere utilizzati in seguito come caratteri normali.

Questi caratteri sono definiti dall'utente con l'istruzione DEFGR$. Il carattere i è designato con GR$(i) oppure con CHR$(128+i). I numeri dei caratteri utente sono compresi da 0 a 127.
Vengono visualizzati con l'istruzione PRINT oppure PRINT USING ed è possibile raddoppiare la loro altezza e la loro larghezza con l'istruzione ATTRB.

Il numero massimo di caratteri da definire viene fissato con un'istruzione CLEAR (terzo parametro) generalmente all'inizio del programma.
Se il numero di caratteri riservati è n, si intende il numero di caratteri da 0 a n-1.

La definizione di un carattere viene fatta con otto numeri interi compresi tra 0 e 255. Ogni numero intero rappresenta uno degli otto segmenti di otto punti che compongono il carattere.

Per ogni segmento, il valore associato viene calcolato in binario: è 1 se il punto è illuminato, e cioè la parte del disegno del carattere, è 0 se il punto fa parte dello sfondo.



matrice di punti del carattere numero i

binario	decimale
1000 0001	129
0100 0010	66
0010 0100	36
0001 1000	24
1111 1111	255
0001 1000	24
0001 1000	24
0001 1000	24

valori dei segmenti del carattere i

Ogni carattere può essere ridefinito nel programma.

Esempio

```
10 'DEFINIZIONE DI UN CARATTERE
20 CLEAR,,1
30 DEFGR$(0)=129,66,36,24,255,24,24,24
40 PRINT GR$(0)
50 PRINT
60 ATTRB 1,1:PRINT CHR$(128)
```

Modalità grafica

In modalità grafica, lo schermo è composto da 200 righe da 320 o 640 punti ciascuna.

Le coordinate di un punto grafico visibile sono comprese:
- per le colonne tra 0 e 319 o 639
- per le righe tra 0 e 199.

Le coordinate di un punto vengono sempre indicate con la seguente sintassi:
(colonna, riga)

Il punto (0,0) è situato in alto a sinistra e il punto (319,199) o (639,199) in basso a destra.

Tutte le istruzioni grafiche accettano delle coordinate al di fuori della zona visibile. I valori delle coordinate di un punto possono variare da −32768 a 32767.

La parte visibile dei tracciati grafici può essere ridotta a una dimensione inferiore rispetto a quella dello schermo. La finestra visibile è definita dall'istruzione WINDOW.

Se il valore di una coordinata non è intero, viene arrotondato all'unità più vicina.

Le istruzioni del tracciato sono le seguenti:

PSET	punto
LINE	segmento
BOX	riquadro
BOXF	riquadro pieno
CIRCLE	cerchio o ellisse
CIRCLEF	cerchio o ellisse pieni

Il colore del tracciato è l'ultimo fissato da SCREEN o COLOR oppure può essere precisato nell'istruzione.

Se il codice del colore è negativo, i punti tracciati fanno parte dello sfondo.

Il tracciato può essere effettuato in quattro modi diversi:

- per cancellazione: il tracciato cancella quello che si trovava precedentemente nello stesso punto.

– in trasparenza: il tracciato si sovrappone al tracciato precedente, i punti del tracciato precedente non sono cancellati dai punti dello sfondo del nuovo tracciato.

– per inversione: ogni punto del tracciato è invertito, da sfondo a primo piano e viceversa. Due tracciati identici nello stesso punto riportano lo schermo allo stato iniziale.

– in modalità ET: il tracciato appare solamente sui punti impostati come primo piano.

Inoltre, il tracciato può essere visualizzato con o senza modifica del colore dei punti.

Tutte queste modalità vengono stabilite dal terzo parametro dell'istruzione CONSOLE.

La funzione POINT permette di sapere qual'è il colore e lo stato di un punto. Il codice prodotto è positivo se il punto appartiene al primo piano e negativo se appartiene allo sfondo.

E' possibile riempire una superficie di colore omogeneo con un colore qualsiasi per mezzo dell'istruzione PAINT. L'area viene colorata per default in modo uniforme. Il modello di riempimento (carattere ASCII o carattere definito dall'utente) viene stabilito con l'istruzione PATTERN.

E' possibile memorizzare diverse aree dello schermo in una tabella di numeri interi salvandola in un file per poterla utilizzare successivamente (istruzioni GET, SAVEP, LOADP e e PUT). Le coordinate di ogni area da memorizzare devono corrispondere alla divisione dello schermo in caratteri (colonne e righe contate in caratteri).

11. Tartarughe

Si possono definire un massimo di 10 tartarughe (numerate da 0 a 9) il cui comportamento è simile a quello della tartaruga del Logo.

Ogni tartaruga è caratterizzata dalla sua:
– forma
– dimensione
– posizione
– direzione di spostamento
– orientamento
– traccia (attivata o disattivata)
– presenza (visibile o invisibile).

Ogni tartaruga può essere spostata seguendo una direzione, o facendola ruotare su se stessa. E' possibile modificare la sua direzione di spostamento.

La forma di una tartaruga è definita da dei segmenti visibili o invisibili con l'istruzione TURTLE.

La dimensione di una tartaruga viene modificata dall'istruzione ZOOM sia in senso assoluto sia relativo.

La posizione di una tartaruga quando non è in movimento è impostata da coordinate assolute con l'istruzione TURTLE.

La direzione dello spostamento è impostata o modificata in modo assoluto o relativo dall'istruzione HEAD.

Lo spostamento in avanti o all'indietro di una tartaruga viene provocato dall'istruzione FWD.

La tartaruga può essere introdotta e spostata in un schermo virtuale le cui coordinate vanno da –32768 a 32767.

Quando la tartaruga raggiunge un lato dello schermo riappare sul lato opposto.

L'orientamento della tartaruga stessa viene impostato o modificato dall'istruzione ROT, in modo assoluto o relativo.

L'istruzione TRACE determina la visualizzazione del percorso della tartaruga.

L'istruzione SHOW indica da un lato se la tartaruga è visibile o invisibile e dall'altro se le modifiche della dimensione e dell'orientamento hanno un effetto immediato oppure rimandato al prossimo spostamento (FWD) o posizionamento (TURTLE).

I valori degli angoli nelle istruzioni HEAD e ROT sono dati in 1/256° di cerchio. Il valore 0 corrisponde a 0 gradi, 128 a 180 gradi e 255 a (360*255)/256 gradi.

Le caratteristiche di default di una tartaruga sono le seguenti:

forma	triangolo isoscele
dimensione	scala normale (16)
posizione	al centro dello schermo
direzione	verso il lato destro dello schermo (0 gradi)
orientamento	identico alla direzione (0)
tracciamento	attivato
visibilità	disattivata
evoluzione	modifiche differite

E' possibile sapere lo stato di una tartaruga per mezzo di un'istruzione

INPUTTURTLE	posizione

e cinque funzioni:

HEAD	direzione
ROT	orientamento

ZOOM	dimensione
SHOW	visibilità
TRACE	tracciamento

Le istruzioni e le funzioni si applicano alla tartaruga attiva; una tartaruga viene attivata con l'istruzione TURTLE.

12. Gestione degli errori

Gli errori vengono rilevati dall'interprete BASIC al momento dell'esecuzione delle istruzioni. La rilevazione di un errore provoca l'arresto dell'esecuzione.
Ogni errore comporta l'apparizione di un messaggio su video.

Esempi:

Syntax Error	(Errore di sintassi)
Division By Zero	(Divisione per zero)
Missing Operand	(Operando mancante)

Se l'errore appare in un programma, durante l'esecuzione viene indicato il numero della riga in cui è presente l'errore.

Type Mismatch in 150
(Errore tipo in 150)

La visualizzazione della riga in cui si è verificato l'errore può essere ottenuta con l'istruzione

LIST

Dopo il punto della riga in cui è stato rilevato l'errore viene visualizzato automaticamente un riquadro in inversione video.

Per evitare che alcuni errori prevedibili influenzino lo svolgimento di un programma e per rimediare eventualmente agli stessi, un'istruzione ON ERROR...GOTO permette di dichiarare l'inizio di una sequenza di trattamento d'errori. Nella sequenza di trattamento di errori, il numero dell'errore è contenuto nella variabile ERR e il numero di riga in cui si è prodotto l'errore è contenuto nella variabile ERL.

La lista dei messaggi di errore e il numero relativo sono contenuti nell'appendice 7.
Dopo il trattamento dell'errore, l'istruzione RESUME permette di riprendere l'esecuzione della parte rimanente del programma sia dal punto in cui si è verificato l'errore sia da un un punto diverso.

Esempio:

```
10 DIVISIONE PER 0
20 ON ERROR GOTO 100
30 DO
40 INPUT X
50 PRINT 1/X
80 LOOP
90 END
100 IF ERR=11 THEN PRINT "Introdurre un numero non nullo"
110 RESUME 40

RUN
?4
.25
?0
Introdurre un numero non nullo
```

Indice alfabetico dei comandi, delle istruzioni e delle funzioni

Il segno * indica che si tratta di un'istruzione specifica del Basic 128. Il segno ◆ indica che si tratta di un'istruzione specifica del Basic 1.

Prima di entrare nei dettagli di ogni singolo comando, istruzione o funzione, ricordiamo brevemente le convenzioni utilizzate nelle descrizioni.

- Gli elementi facoltativi di un'istruzione vengono stampati in corsivo sullo sfondo colorato.
- I principali parametri che cambiano da una operazione all'altra vengono indicati da:

Numero: rappresenta una costante numerica (3*55) o una variabile numerica (Z) oppure un'espressione il cui risultato è numerico (Z*RND)

Stringa: rappresenta una costante ("GIULIO") o una variabile (P$) oppure un'espressione il cui risultato è una stringa (MID$("GIULIO",1,1)).

Dato: rappresenta un numero oppure una stringa.

Variabile: rappresenta una variabile numerica (X) o una variabile stringa (YS).

- I diversi esempi forniti insieme ad ogni istruzione mostrano la parte opzionale dell'istruzione. Generalmente, le opzioni vengono precisate durante la spiegazione.

ABS(numero)
funzione

Dà il valore assoluto del numero.

Esempio:

```
?ABS(5*3)
15
```

ASC(stringa)
funzione

Dà il codice ASCII del primo carattere della stringa.

Nel caso la stringa sia vuota, appare il messaggio di errore Illegal Function Call (Richiamo di funzione non valida).

Attenzione: le lettere minuscole accentate vengono codificate da una sequenza di tre caratteri, di cui il primo ha sempre il codice 22. Quindi, il carattere viene codificato dalla sequenza di codici 22,66,101. E' possibile ottenere ciascuno di questi codici utilizzando la funzione MID$.

ATN(numero)*
funzione

Dà l'arcotangente del numero.

Il risultato della funzione è un angolo, espresso in radianti, compreso tra $-\pi/2$ e $\pi/2$.
Se il numero è a precisione doppia, il risultato viene dato a doppia precisione.

Esempio:

```
?ATN(3)
1.24905
```

ATTRB *larghezza, altezza*
istruzione

Definisce te proporzioni dei caratteri visualizzati sullo schermo.

Il primo parametro indica la larghezza dei caratteri:
  larghezza=0, targhezza normale
  larghezza=1, larghezza doppia

Il secondo parametro indica l'altezza dei caratteri:
  altezza=0, altezza normale
  altezza=1, altezza doppia

Nessun parametro è obbligatorio. Se manca uno dei due parametri, l'attributo corrispondente non viene modificato.

Specificando "doppia altezza" l'istruzione PRINT visualizza il carattere sulla riga in corso e su quella precedente; il ritorno a capo dà origine ad una doppia interlinea. Si raccomanda di saltare una riga vuota con un PRINT, prima di effettuare la prima visualizzazione a doppia altezza.

L'esecuzione di INPUT o di LINE tNPUT, dopo l'eventuale visualizzazione del testo associato, riporta il carattere alle dimensioni normali. Le funzioni INKEY$ e INPUT$ non modificano la modatità di visualizzazione.

Esempio:

```
100 PRINT"INIZIO DEL TESTO"
110 PRINT
120 ATTRB 1,1
130 PRINT"Pazienza..."
```

AUTO *numero, passo*
comando

Propone automaticamente i numeri di riga per la scrittura dei programmi.

Il primo numero indica il numero della prima riga da scrivere (10 per default).
Il secondo numero indica lo scarto fra le linee (10 per default).

I due parametri sono facoltativi.

Nel caso la riga proposta esista già, prima del numero di riga viene visualizzato il messaggio "Une not empty" (riga non vuota).

Il numero di riga proposto può essere modificato, il cursore può essere spostato su un'altra riga che può, a sua volta, essere modificata e poi convalidata: la numerazione automatica riparte quindi sul nuovo numero. E' possibile escludere la numerazione automatica con CTRL–C oppure convalidando una riga senza numero.

Esempio:

```
AUTO 1000,5
AUTO
```

BACKUP n. drive 1 *TO n. drive 2*
comando

Copia per intero il contenuto di un dischetto su un altro dischetto.

Tale comando copia integralmente il dischetto che si trova nel primo drive sul dischetto che si trova nel secondo drive.

Con un solo drive, BACKUP n. copia un dischetto su un altro, ponendo nel drive alternativamente il dischetto "sorgente" ed il dischetto "destinazione".

In Basic 1, BACKUP distrugge sempre il programma e le variabili.

Esempio:

BACKUP 0 TO 1	copia il dischetto 0 sul dischetto 1.
BACKUP 0	solo drive n. 0.
BACKUP 1	solo drive n. 1.

BANK *
funzione

Dà il numero del banco di memoria corrente.

Questo sarà un numero compreso fra 1 e 6.
All'inizializzazione viene selezionato il banco di grado più elevato.

Esempio:

```
?BANK
2
```

BANK numero *
istruzione

Seleziona il banco di memoria di cui viene precisato il numero.

Tale numero deve essere compreso fra 0 ed il più alto numero dei banchi di memoria disponibili (6).
Se il numero è uguale a 0, viene selezionato l'ultimo banco di memoria esistente, come avviene all'inizializzazione. Questa istruzione riguarda solamente i banchi di memoria posti fra gli indirizzi $H6000 e $H9FFF. Il suo scopo è quello di assegnare tutte le istruzioni che comprendono dei riferimenti diretti alla memoria: DEF USR, EXEC, CLEAR, LOADM, SAVEM, PEEK, POKE,...

Esempio:

```
BANK 1
```

BEEP
istruzione

Produce un suono breve.

Questa istruzione è equivalente a PRINT CHR$(7);.

Esempio:

```
FOR I=1 TO 100:BEEP:NEXT I
```

BOX *(c1,r1)*–**(c2,r2)** *carattere, colore, sfondo, inversione*
BOX (c1,r1)–(c2,r2), *colore*
istruzione

Traccia un rettangolo i cui lati sono paralleli ai bordi dello schermo.

Tale istruzione funziona in modalità carattere quando è presente l'argomento carattere. In questo caso, le coordinate di colonna (c1,c2) vengono contate da 0 a 39 (o 79), e di riga (r1,r2) da 0 a 24. Inoltre funziona in modalità grafica quando non è presente l'argomento carattere. In questo caso, le coordinate di colonna e di riga sono comprese fra –32768 e 32767 in Basic 128. In Basic 1 le coordinate di colonna sono comprese fra 0 e 319 e le coordinate di riga fra 0 e 199. Il rettangolo tracciato ha per vertici opposti i punti situati in (c1,r1) e (c2,r2).

Se viene omesso il primo punto (c1,r1), questo viene sostituito dall'ultimo punto dell'ultimo tracciato e cioè:
- il secondo vertice in BOX, BOXF o LINE,
- il punto tracciato da PSET,
- il punto (0,0) se non vi è stato alcun tracciato.

L'argomento colore è un numero. In modalità carattere, l'argomento carattere è una stringa (variabile o costante) il cui primo carattere è utilizzato per il tracciato. I tre parametri colore, sfondo, inversione sono gli stessi dell'istruzione COLOR ed il loro effetto è identico. In modalità grafica, il tracciato si verifica solo all'interno della finestra definita da WINDOW (solo Basic 128).
In modalità carattere e grafica, il tipo di tracciato è fissato dal terzo parametro dell'istruzione CONSOLE. Le coordinate del primo punto e gli argomenti colore, sfondo e inversione sono facoltativi.

Esempio:

```
BOX(1,9)-(12,15)"^",3,0          modalità carattere
BOX(104,8)-(216,48),2            modalità grafica
BOX-(300,70)
```

Vedere anche STEP.

BOXF *(c1,r1)*-**(c2,r2)** *carattere, colore, sfondo, inversione*
BOXF *(c1,r1)*-**(c2,r2)**, *colore*
istruzione

Traccia un rettangolo i cui lati sono paralleli ai bordi dello schermo e lo riempie del modello indicato.

In modalità carattere, il riempimento viene eseguito con il carattere indicato.
In modalità grafica, il modello di riempimento è quello fissato dall'istruzione PATTERN, altrimenti il riempimento è uniforme.
La sintassi dell'istruzione BOXF è identica a quella di BOX, sia in modalità carattere, sia in modalità grafica.

Esempio:

```
BOXF(2,2)-(14,14)"+",4,0         modalità carattere
BOXF(104,8)-(216,48),2           modalità grafica
```

Vedere anche STEP.

CDBL(numero)
funzione

Converte il numero in precisione doppia.

Il numero convertito in precisione doppia non viene completato solamente con degli zeri, cosa che potrebbe portare a piccole variazioni.
Se l'argomento non è un numero, appare il messaggio di errore "Illegal Function Call" (Richiamo di funzione non valida).

Esempio:

```
?CDBL(454.67)
454.6700134277344
```

CHAIN descrittore di file, *r1–r2,r3**
istruzione

Elimina una parte del programma residente in memoria, unisce il programma indicato e continua l'esecuzione alla riga indicata.

Tale istruzione permette di "fondere" una parte di programma e di eseguirla senza perdere il contenuto di tutte le variabili. La sezione di programma compresa fra le righe r1 e r2 viene eliminata (stessa sintassi di DELETE). Questa opzione di eliminazione non è obbligatoria. In seguito il programma indicato viene fuso al programma residente. Al momento della fusione le righe del nuovo programma sostituiscono le righe (eventuali) del programma residente. Il programma da unire deve essere salvato in binario (e non in ASCII come in MERGE).
Se è presente l'opzione r3, l'esecuzione prosegue alla riga r3, altrimenti l'esecuzione riprende a partire dalla prima riga del programma. Vengono considerate solamente le variabili presenti nella lista dell'istruzione COMMON.

Esempio:

CHAIN"1:SEGUITO",500–900,100 sopprime le righe da 500 a 900, unisce il programma SEGUITO del dischetto 1 e continua l'esecuzione dalla riga 100.
CHAIN"PG3945",500– sopprime la sezione del programma a partire dalla riga 500, l'esecuzione riprende dall'inizio.
CHAIN"PASS2" nessuna riga viene soppressa, l'esecuzione riprende dal punto di fusione.

CHR$(numero)
funzione

Dà il carattere del codice ASCII corrispondente al numero indicato.

Il numero deve essere compreso fra 0 e 255.
Questa funzione permette di formare delle stringhe contenenti codici di controllo. Quindi per inserire un segnale acustico (codice ASCII 7) si introdurrà CHR$(7).
I caratteri, il cui codice risulta essere superiore a 127, sono definiti dall'utente con l'istruzione DEFGR$. CHR$(128+numero) è equivalente a GR$(numero). Le lettere minuscole accentate possono essere introdotte da CHR$.
La funzione inversa di CHR$ è ASC.

Esempio:

```
CHR$(65)                              rappresenta A
?CHR$(12)                             cancella lo schermo
CHR$(22)+CHR$(66)+CHR$(101)           rappresenta é
```

CINT(numero)
funzione

Trasforma un numero reale in un numero intero.

La trasformazione viene effettuata per arrotondamento. Il numero deve essere compreso fra –32768 e 32767 altrimenti viene visualizzato il messaggio d'errore "Overflow".

Esempio:

```
?CINT(45.5):CINT(45.4)
 46 45
```

CIRCLE (colonna, riga) *ro,* **rv,** *colore; inizio, fine*
Istruzione

Traccia una porzione di cerchio o di ellisse il cui centro è il punto determinato dalla coordinata (colonna, riga).
Se ro è presente, tale istruzione traccia un'ellisse di raggio orizzontale ro e di raggio verticale rv. Altrimenti traccia un cerchio di raggio rv.
Se il parametro colore non è presente, il tracciato viene eseguito con il colore usato per i caratteri.
Se sono presenti i due parametri inizio e fine, il tracciato si limiterà unicamente alla parte di cerchio o di ellisse corrispondente. Gli angoli vengono calcolati in radianti; la direzione positiva è in senso orario.
Il tracciato viene eseguito con la modalità definita dal terzo parametro di CONSOLE.
Nel Basic 1, il tracciato di ellisse o di una parte di cerchio o di ellisse è impossibile.

Esempio:

CIRCLE(100,100)80,40,5;0,3.1416 semi–ellisse inferiore centrata in (100,100) di raggio orizzontale 80, di raggio verticale 40, di colore magenta.

CIRCLE(160,100),90,2 cerchio centrato nel mezzo dello schermo, di raggio 90, di colore verde.

CIRCLEF (colonna, riga) *ro*, **rv**, *colore; inizio, fine*
istruzione *

Traccia una sezione di cerchio o di ellisse il cui centro è il punto della coordinata (colonna, riga) e la riempie del modello corrente.

Il modello di riempimento è quello fissato da PATTERN. Per default, il riempimento è uniforme. Se viene tracciata sofamente una sezione di cerchio o di ellisse, verrà riempito solo il settore corrispondente.
La sintassi di CIRCLEF è identica a quella di CIRCLE.

Esempio:

PATTERN"*":CIRCLEF(100.100)80,40,5:0,3.1416 semi–ellisse inferiore centrata in (100,100) di raggio orizzontale 80, di raggio verticale 40, di colore magenta e riempita di caratteri "*".

CLEAR *volume,indirizzo1,numero,indirizzo2*
istruzione

Fissa molteplici parametri dello spazio di memoria utente e riporta a 0 tutte le variabili.

Questa istruzione elimina tutte le variabili e tutte le matrici esistenti ed annulla inoltre l'effetto dell'istruzione ON ERROR GOTO/GOSUB, ON KEY..., ON INTERVAL. Il primo parametro (volume) è un numero che fissa il volume riservato alle stringhe. Tale volume è fissato per default a 300 caratteri.

Esempio:

CLEAR 2000 riserva uno spazio di 2000 caratteri

Il secondo parametro (indirizzo1) permette di riservare una zona memoria nello spazio corrispondente ai banchi di memoria commutabili. Questa assegnazione riguarda il banco selezionato dall'istruzione BANK ed i banchi di livello superiore. L'indirizzo precisato è un numero che deve essere inferiore a &HDFFF. Nel Basic 1 è possibile riservare solo un banco.

Esempio:

BANK 4:CLEAR,&HBFFF riserva lo spazio di memoria da &HC000 a &HDFFF del banco 4 e i banchi 5 e 6 completamente.
BANK 4:CLEAR,&H8FFF riserva lo spazio di memoria da &H9000 a &H9FFF della zona non commutabite e i banchi 4, 5 e 6 completamente.
Il terzo parametro (numero) fissa il numero massimo di caratteri definiti dall'utente. Tale numero non viene modificato fino ad una successiva istruzione CLEAR.

Esempio:

CLEAR.,10 fissa a 10 il numero dei caratteri utente

Il quarto parametro (indirizzo2) permette di riservare una zona memoria all'interno dello spazio non commutabile, cioè fuori dai banchi. L'indirizzo precisato corrisponde ad un numero inferiore a &HA000. L'operazione di assegnazione riguarda esclusivamente la zona di memoria compresa fra l'indirizzo2+1 e &H9FFF e non riguarda i banchi di memoria. Tale parametro non esiste nel Basic 1.

Esempio:

CLEAR,,&H8FFF riserva lo spazio di memoria da &H9000 a &H9FFF

Nessuno dei parametri di CLEAR è obbligatorio. Le assegnazioni dello spazio di memoria effettuate con l'istruzione CLEAR possono essere modificate solo da un altro CLEAR.
CLEAR da solo annulla tutte le variabili, ma non influenza lo spazio riservato in memoria.

CLOSE #n. canale, #n. canale
Istruzione

Chiude il (o i) file, di cui si indica il n. di canale.

Tale operazione contrassegna la fine del file e scrive nel catalogo tutte le informazioni necessarie. Se non viene precisato alcun numero di canale, CLOSE chiude tutti i file tranne nel caso in cui si verifichi un errore durante l'esecuzione della chiusura.
Non bisogna dimenticare che un'interruzione effettuata con CTRL–C durante la scrittura o la lettura non chiude i file.

Esempio:

CLOSE#1,#3 chiude i file 1 e 3
CLOSE chiude tutti i file aperti

CLS
Istruzione

Cancella la finestra di visualizzazione definita da CONSOLE e pone il cursore in alto alla sinistra di questa finestra.

La cancellazione viene effettuata, riportando tutti i punti al colore dello sfondo. Lo stesso effetto può essere ottenuto con PRINT CHR$(12).

COLOR *primo piano,sfondo,inversione*
Istruzione

Fissa il colore di primo piano ed il colore di sfondo per le visualizzazioni e i tracciati che seguono.

I due primi parametri sono dei numeri. Il primo fissa il colore di primo piano, il secondo il colore di sfondo. Il terzo parametro è un numero (0 o 1). Quando è presente, dà origine ad un'inversione di colori del primo piano e dello sfondo.
Nessuno dei tre parametri è obbligatorio.

Esempio:

```
COLOR 2,0   caratteri e tracciati verdi su sfondo nero
COLOR,7     sfondo bianco
COLOR,,1    inverte i colori precedenti
```

COMMON elenco di variabili *
istruzione

Permette di trasmettere delle variabili da un programma all'altro durante l'operazione di concatenamento di tali programmi con CHAIN.

Le variabili dell'elenco che segue COMMON sono protette durante l'esecuzione CHAIN. Il programma unito potrà così utilizzarle.
L'elenco delle variabili può contenere delle matrici con le relative dimensioni; le matrici corrispondenti vengono quindi dichiarate, come avviene con DIM. Le variabili dell'elenco vengono annullate da CLEAR, RUN, NEW, LOAD, ecc.
Durante l'esecuzione di COMMON, vengono distrutte tutte le variabili esistenti, tranne le variabili già definite da COMMON. Quindi è preferibile porre le istruzioni COMMON all'inizio del programma, dopo un eventuale CLEAR.

Esempio:

```
COMMON A.B$,Q(1000)        protegge A,B$, dichiara e
C=3                        protegge la matrice C

COMMON C.D                 protegge C e D
?C;D
00                         il contenuto di C è perso
```

CONSOLE *riga superiore, riga inferiore, tracciato, scorrimento, modalità*
istruzione

Fissa molteplici parametri del tracciato e della visualizzazione.

Tutti i parametri sono opzionali, tuttavia tale istruzione non deve concludersi con una virgola e deve comprendere almeno un parametro.
I primi due parametri fissano la finestra di visualizzazione in modalità carattere:
– il primo indica la riga superiore della finestra,
– il secondo indica la riga inferiore.
Questi due parametri sono numeri compresi fra 0 e 24. Il terzo parametro precisa la modalità di tracciamento grafico:
– se è pari, il colore viene modificato, insieme al primo piano,

-se è dispari, il colore non viene modificato,
-se vale 0 o 1, il tracciato cancella i caratteri che si trovano sul suo percorso (i punti dello sfondo del modello vengono cancellati),
-se vale 2 o 3 il tracciato non cancella i caratteri che si trovano sul suo percorso. Un modello potrà essere sovrapposto ad un altro,
-se vale 4 o 5, il tracciato inverte i punti del primo piano esistente (OR esclusivo). Due tracciati successivi identici riportano allo stato iniziale;
-se vale 6 o 7, il tracciato appare unicamente sui punti il cui colore è quello del primo piano (AND). I valori 2, 3, 4 e 5 del terzo parametro non possono essere utilizzati nel Basic 1.
Il quarto parametro fissa la modalità di scorrimento:
-se vale 0, lo scorrimento avviene a velocità normale,
-se vale 1, lo scorrimento avviene a velocità lenta,
-se vale 2, lo scorrimento avviene in modalità pagina: quando la finestra è piena la nuova riga viene visualizzata nella parte alta della finestra.
Il quinto parametro definisce la modalità grafica:
-se vale 0, si tratta della modalità a 40 colonne, sedici colori;
-se vale 1, lo schermo sarà a 80 colonne, con solo due colori, il tracciato viene quindi eseguito senza tener conto del colore;
-se vale 2, lo schermo sarà a quattro colori su tutti i punti. In questa modalità, ogni punto può assumere una delle quattro colorazioni, indipendentemente da quella dei punti vicini;
-se vale 3, lo schermo sarà a sedici colori su tutti i punti. In questa modalità, ogni punto dello schermo, le cui dimensioni sono 160 x 200, può assumere una delle sedici colorazioni, indipendentemente da quella dei punti vicini. Il quinto parametro non esiste nel Basic 1.
-se vale 4, si attiva la modalità di visualizzazione a pagine, con la pagina 1 che utilizza il colore 1 e la scrittura tramite PRINT sulla pagina 2 con il colore 2;
-se vale 5, si attiva la visualizzazione a pagine con la pagina 2 che utilizza il colore 2 e la scrittura tramite PRINT sulla pagina 1 con il colore 1;
-se vale 6, si attiva la modalità di sovrapposizione delle due pagine con la scrittura tramite PRINT sulla pagina 2 con il colore 2;
-se vale 7, si attiva la modalità di sovrapposizione di due pagine con scrittura tramite PRINT sulla pagina 1 con il colore 1. La pagina 1 ha la priorità sulla pagina 2;
-se vale 8, 9, 10, 11 si attiva la modalità di tripla sovrapposizione con 8=primo piano, 9=secondo piano, 10=terzo piano, 11=quarto piano. Ogni piano utilizza il colore corrispondente al suo numero. Il quarto piano ha la priorità sul terzo piano, ecc. Nelle modalità da 4 a 11 il colore 0 viene utilizzato come colore di sfondo.
In modalità 3 la visualizzazione dei caratteri non viene gestita.
Nelle modalità da 8 a 11 la visualizzazione dei caratteri e della grafica non viene gestita.
Nelle modalità da 4 a 7 la selezione della pagina del tracciato grafico si effettua tramite il comando COLOR con il parametro da 0 a 3.

Esempio:

CONSOLE 10,20,,2	limita la visualizzazione dalla riga 10 alla riga 20 incluse, con modalità pagina
CONSOLE ,,1	traccia un'immagine grafica senza tener conto del colore

CONSOLE.,4	traccia un'immagine grafica con inversione e colore
CONSOLE,,,,2	modalità bitmap a 4 colori

CONT
comando

Riprende l'esecuzione di un programma dopo un'interruzione.

L'interruzione di un programma può essere stata determinata da CTRL–C, oppure dall'esecuzione di STOP o END, o da un errore. L'esecuzione riprende dal punto in cui è stata interrotta. Nel caso sia stato premuto CTRL–C durante la risposta ad un'istruzione INPUT, al momento della ripresa, il messaggio associato appare di nuovo ed è necesssrio fornire la risposta una seconda votta.
Il comando CONT è utile per l'esecuzione di programmi collegati all'istruzione STOP. Quando l'esecuzione è sospesa, il contenuto delle variabili può essere visualizzato ed anche modificato con una nuova assegnazione. Il comando CONT permette di riprendere l'esecuzione dal punto in cui era stata interrotta.
Il comando CONT non funziona se il programma è stato modificato (messaggio di errore Can't Continue).

COPY descrittore di file 1 *TO descrittore di file 2*
comando

Permette la copia integrafe di un file (cambiando eventualmente il nome) sullo stesso supporto o su un altro. Con due drive, COPY permette di copiare un file da un drive all'altro.
Con un solo drive, COPY permette di copiare un file:
– sullo stesso dischetto, cambiando il nome,
– su un altro dischetto, senza cambiare il nome.
In questo caso si pone altemativamente nel drive il dischetto sorgente ed il dischetto destinazione. Di conseguenza, nel comando si indica solo il nome del file da copiare (COPY nome file).
Nel Basic 128, il nome del file può comprendere un commento. COPY non offre la possibilità di trasferire un file da un dischetto ad un'altra periferica.

Esempio:

```
COPY"0:DESS.DAT"TO"1:(4 nov)DESS.DAT"
COPY"MENU.BAS"TO"SCELTA.BAS"
COPY"BUDGET.DAT"
```

COS(numero)
funzione

Dà il coseno dell'angolo espresso in radianti.

Nel Basic 128, il risultato è a doppia precisione se l'angolo è a doppia precisione.

Esempio:

```
?COS(0.2)
.980067
```

CRUNCH$(stringa) *
funzione

Analizza la stringa data e ritorna una stringa di codice valutabile con la funzione EVAL.

Questa funzione permette di codificare una stringa che rispetta la sintassi di un'espressione del linguaggio Basic. Questa espressione, una volta che è stata codificata da CRUNCH$, può essere valutata con EVAL.
La stringa da codificare deve essere un'espressione numerica o un'espressione stringa.

Esempio:

```
10 DO
20 LINE INPUT"Espressione:";RS
30 PRINT EVAL(CRUNCH$(R$))
40 LOOP
RUN
Espressione:3+4+5
12
Espressione:RIGHTS("DOMANI",4)
MANI
```

CSNG(numero)
funzione

Converte il numero a precisione singola.

La conversione viene effettuata dopo arrotondamento. Le altre funzioni di conversione sono CINT e CDBL.

Esempio:

```
?CSNG(975.341817#)
975.342
```

CSRLIN
funzione

Dà il numero della riga sulla quale si trova il cursore.

Il risultato è un numero intero compreso fra 0 (riga superiore) e 24 (riga inferiore). La funzione POS permette di conoscere la posizione del cursore sulla riga.

Esempio:

LOCATE 5,8:?CSRLIN visualizza 8 alla colonna 5, riga 8

CVD (variabile atringa)
funzione

Assicura la conversione del contenuto della variabile (stringa) in un numero a precisione doppia.

La variabile deve contenere dei dati che siano stati registrati con una conversione, tramite MKD$(). Questa funzione è applicata generalmente ad una variabile di campo definita con FIELD, dopo la lettura di una registrazione con GET#.

Esempio:

A#=CVD(PREZZOS)

CVI (variabile stringa)
funzione

Questa funzione analoga a CVD() assicura la conversione del contenuto della variabile in un numero intero.

La variabile deve contenere dei dati registrati con una conversione tramite MKI$().

Esempio:

I%=CVI(Q$)

CVS (variabile stringa)
funzione

Questa funzione analoga a CVD() assicura la conversione del contenuto della variabile in un numero a precisione singola.

La variabile deve contenere dei dati registrati con una conversione tramite MKS$().

Esempio:

```
J=CVS(TASSA$)
```

DATA elenco di costanti
istruzione

Questa istruzione permette di dichiarare un elenco di costanti che potranno essere lette dall'istruzione READ.

Le istruzioni DATA non sono eseguibili e possono trovarsi dappertutto nel programma. L'istruzione DATA può contenere tante costanti quante ne permette la lunghezza della riga e non esiste limite di numero per le istruzioni DATA contenute in un programma.
Le costanti della riga possono essere di tipo intero, reale, a precisione doppia o stringa. Le espressioni non possono figurare in questo elenco.
Le stringhe devono essere definite dalle virgolette se è presente un separatore (virgola, due punti, o spazi significativi all'inizio o alla fine). Negli altri casi, le virgolette possono essere omesse.
Le diverse costanti dell'elenco vengono separate le une dalle altre da una virgola.
L'insieme delle costanti contenute nell'istruzione DATA di un programma possono essere lette da istruzioni READ, in modo sequenziale. Il tipo di costanti lette deve corrispondere al tipo di variabili delle istruzioni READ, che ricevono i valori letti.
I dati di una stessa istruzione DATA possono essere letti più volte se l'istruzione RESTORE permette di tornare alla riga DATA corrispondente.

Esempio:

```
100 READ A,B,C#,A$
110 PRINT A,B,C#,A$
...
200 DATA 1,2,3.456789012,"I SUOI BEGLI OCCHI"

RUN
1 2 3.456789012 I SUOI BEGLI OCCHI
```

DEF FN nome (variabile, *variabile...)* **=espressione**
istruzione

Definisce una nuova funzione che riporta alle variabili indicate tra parentesi.

Il nome della funzione segue esattamente le stesse regole di un nome di variabile. La parte a destra del segno = è l'espressione che definisce la funzione. La funzione utilizza le stesse variabili di quelle indicate a sinistra dell'uguale. I nomi delle variabili possono essere identici a quelli già utilizzati altrove, senza che vi sia alcuna interazione.
La definizione della funzione deve essere eseguita prima del suo utilizzo. Le funzioni così definite vengono utilizzate come le funzioni ordinarie del Basic e vengono chiamate con il loro nome: FN nome (variabile, variabile,...). Le funzioni definite possono dare un risultato numerico o un risultato stringa.

Esempio:

```
DEF FN DIST(X,Y)=SQR(X2+Y2)
A=FNDIST(3,5)
DEF FNCH$(1)=MID$(STR$(1),2)
A$=FNCHS(DS)
```

DEFDBL *lettera–lettera, lettera–lettera,...* *
DEFINT *lettera–lettera, lettera–lettera,...*
DEFSNG *lettera–lettera, lettera–lettera,...*
DEFSTR *lettera–lettera, lettera–lettera,...*
istruzione

Dichiara il tipo di un gruppo di variabili che iniziano con la stessa lettera.

Queste istruzioni di dichiarazione riguardano tutte le variabili che non comprendono alcun tipo alla fine del nome, cioè che non terminano con %, !, # o $ ed il cui nome comincia con una lettera che si trova in una delle aree indicate.

DEFDBL	dichiara il tipo a precisione doppia
DEFINT	dichiara il tipo intero
DEFSNG	dichiara il tipo reale a precisione singola
DEFSTR	dichiara il tipo stringa

In mancanza di istruzioni di dichiarazione di tipo, o di carattere che contraddistingue il tipo, ogni variabile viene considerata come numerica a precisione singola. Le istruzioni di dichiarazione tipo devono essere eseguite prima dell'utilizzo delle variabili a cui si riferiscono. E' preferibile porle all'inizio del programma.

Esempio:

```
DEFDBL A–C,W–Z    tutte le variabili il cui nome inizia con A,B,C,W,X,Y,Z sono di tipo a precisione doppia
DEFINT I,J    le variabili il cui nome comincia con I o J sono intere
```

DEFGR$(n.)=elenco di otto numeri
istruzione

Definisce il carattere utente del numero indicato.

Il numero del carattere è una cifra compresa fra 0 e 127. E' possibile definire i caratteri utente solo se sono stati riservati da una precedente istruzione CLEAR (terzo parametro).
Il numero del carattere non può essere superiore al numero di caratteri riservati in CLEAR meno uno. La definizione dei caratteri viene eseguita da otto segmenti composti da otto punti ciascuno.
Ad ogni punto del segmento si fa corrispondere il numero binario:
1 se è attivato,
0 se non è attivato.

A ogni segmento, si fa corrispondere il numero binario così ottenuto. Il suo valore, in decimali, è quindi compreso fra 0 e 255.
Nella definizione di un carattere si indica alla destra del segno = l'elenco degli otto numeri corrispondente agli otto segmenti che compongono il carattere, cominciando dal segmento superiore. Questi otto numeri possono essere dati sotto forma di costanti, variabili o elementi di matrice. Il carattere così definito è indicato con GR$(n.) o CHR$(128+n.).

Esempio:

```
10 CLEAR,,2
20 DEFGR$(0)=128,64,32,16,24,36,66,129
30 DEFGR$(1)=&B1111.11. &B10000001, &B10000001, &B10000001,
   &B10000001, &B10000001, &B10000001, &B111111111
40 PRINT GR$(0):
50 PRINT GR$(1)
```

DEF USRn = indirizzo *
istruzione

Definisce la funzione utente in linguaggio macchina, di numero n, a partire dall'indirizzo di memoria indicato.

Vi possono essere 10 funzioni utente simultanee, numerate da 0 a 9 per default. L'indirizzo indica l'inizio della funzione in linguaggio macchina. Una zona di memoria destinata alle funzioni utente può essere riservata con: CLEAR,indirizzo o CLEAR,,,indirizzo. Il modulo in linguaggio macchina deve terminare con l'istruzione RTS del 6809 e non deve modificare il contenuto dei registri S e DP.
Il passaggio dei parametri usa, sia in chiamata sia al ritorno, i seguenti registri:
Accumulatore A: 2 numero intero
(tipo della variabile) 3 stringa
4 precisione singola
8 precisione doppia
Indice X: 2,X numero intero
(puntatore sul valore) 0,X numero reale a precisione singola o doppia 0,X descrittore di stringa
Accumulatore B: lunghezza della stringa
Registro U: indirizzo del primo byte della stringa

I numeri interi vengono codificati su due byte (notazione in complemento a due), i numeri reali a precisione singola su quattro byte (primo byte: esponente, resto: mantissa), i numeri reali a precisione doppia su otto byte (come sopra).

Il descrittore di stringa comprende tre byte: il primo contiene la lunghezza della stringa, i due successivi l'indirizzo del primo carattere.

Esempio:

DEF USR2=&H9F80	definisce la funzione numero 2 all'indirizzo &H9F80
W=USR2(V)	chiamata della funzione

DELETE zona di n. riga, *riga*
comando o istruzione

Elimina le righe del programma e prosegue l'esecuzione alla riga indicata.

Le righe del programma da eliminare vengono indicate nel modo seguente:

DELETE riga	elimina la riga
DELETE.	elimina la riga corrente
DELETE riga1–riga2	elimina tutte le righe dalla riga1 alla riga2 comprese
DELETE riga1–	elimina tutte le righe dalla 1 alla fine
DELETE –riga2	elimina tutte le righe dall'inizio alla riga2
DELETE –	elimina tutto il programma

Quando il secondo parametro è presente, esso indica la riga da eseguire dopo l'eliminazione. Questa opportunità può essere utile per l'eliminazione di una parte già eseguita in modo tale da liberare lo spazio necessario alle variabili che seguono. Il secondo parametro è vietato nel Basic 1.

Esempio:

DELETE 100–490.50	elimina le righe da 100 a 490 e prosegue l'esecuzione alla riga 50
DELETE 80	elimina la riga 80
DELETE –	elimina tutto il programma

DENSITY n. drive, densità ◆
istruzione

Fissa la densità di registrazione per il drive designato. Il numero del drive è compreso fra 0 e 4.
La densità è indicata con:
– 1 per densità singola,
– 2 per densità doppia.

Esempio:

DENSITY 0,2	imposta il drive 0 su densità doppia

Attenzione: Durante l'attivazione, con un drive a doppia densità, il PC 128 cambia automaticamente la sua densità, assumendo quella del dischetto presente nel drive. Per cambiare la densità nel Basic 128, è necessario reinizializzare con un dischetto che abbia la giusta densità.

DEVICE "nome periferica"
istruzione

Determina il nome della periferica che verrà assunto per default in un descrittore di file che non la indica.

Durante l'inizializzazione la periferica implicita è il drive 0 (Basic 128) o il registratore (Basic 1 senza DOS).

Esempio:

```
DEVICE"1:"          il drive 1 diventa la periferica assunta per default
DEVICE"CASS:"
```

DIM elenco di matrici
con matrice=nome di variabile (elenco di indici)
istruzione

Riserva lo spazio necessario alle matrici a una o più dimensioni.

Gli indici sono numeri arrotondati al numero intero più vicino che fissano il valore massimo di ogni indice di matrice. Il valore minimo che un indice può avere è 0. L'istruzione DIM è eseguibile e, durante la sua esecuzione, gli elementi di una matrice numerica vengono inizializzati al valore 0 e gli elementi di una matrice di stringa vengono inizializzati al valore "stringa vuota".
Se, durante l'esecuzione, un indice è al di fuori dei limiti specificati in DIM, viene visualizzato il messaggio di errore "Bad Subscript" (indice errato).
E' possibile utilizzare le matrici senza dichiararle. In questo caso, il valore numerico degli indici non deve essere superiore a 10.

Esempio:

```
DIM A(15,12)   matrice a due dimensioni il cui primo indice può variare da 0 a 15
ed il secondo da 0 a 12
DIM C$(14)     matrice di stringhe di 15 elementi
```

DIR *"n. drive: nome, estensione"*
istruzione

Elenca l'indice del dischetto relativo ai file precisati.

Se il n. di drive non viene precisato, si può ottenere l'indice del dischetto del drive di default ("0:" salvo modifica con DEVICE). Se il nome non viene precisato, l'elenco visualizza tutti i file che hanno l'estensione indicata. Al contrario, se l'estensione non viene precisata, l'elenco visualizza tutti i file che hanno questo nome. Se il nome è incompleto, l'elenco visualizza tutti i file che iniziano con la radice indicata. L'intestazione dell'indice dà la densità del dischetto, il n. del drive, il suo nome se esistente ed il numero di Kbyte liberi.

Nel Basic 1, è possibile indicare solo il numero del drive. Per ogni file elencato, le registrazioni dell'indice sono formate da sei parti:

1. Nome del file (otto caratteri)

2. Estensione
BAS: programma Basic
DAT: dati
BIN: binario
MAP: immagine
TRA: tracciato

3. Tipo di file
B: programma Basic
D: dati Basic
A: programma Assembler
M: programma in linguaggio macchina

4. Tipo di dati
A: ASCII
B: Binario

5. Dimensione del file espressa in Kbyte

6. Commento (8 caratteri), se esistente

L'esecuzione di DIR per una periferica che non sia un drive visualizza il messaggio di errore "Illegal Function Call" (richiamo funzione non valida).

Esempio:

DIR"0:PROVA.BAS"	elenca il file PROVA.BAS nell'elenco del drive 0
DIR"1:.DAT"	elenca tutti i fife di dati del drive 1
DIR"T"	elenca tutti i file che iniziano per T sul drive corrente
DIR	elenca l'indice completo del drive corrente

DIRP *"n. drive: nome.estensione"* *
istruzione

Elenca l'indice relativo ai file precisati sulla stampante parallela.

La sintassi è identica a quella di DIR.
Il listato viene inviato alla stampante parallela.

Esempio:

DIRP"2:"	invia alla stampante parallela l'elenco dei file del drive 2

DO...LOOP *
Istruzione generale d'iterazione

Tutte le istruzioni comprese tra DO e LOOP vengono ripetute. Normalmente non si può uscire da un ciclo d'iterazione se non attraverso l'istruzione EXIT. L'esecuzione di EXIT fa uscire dal ciclo d'iterazione; l'esecuzione continua quindi a partire dalla prima istruzione che segue **LOOP**.

Le istruzioni di collegamento (GOTO, ON...GOTO e THEN n. di riga) permettono un'uscita a un numero di righe diverso da quello dell'uscita normale. Questo tipo di uscita è utile solo in casi eccezionali (errore, arresto di programma,...) poiché fa perdere i vantaggi della strutturazione del programma.
Non è possibile inserirsi nel mezzo di un ciclo DO...LOOP tramite l'istruzione GOTO. L'interprete Basic, eseguendo LOOP, cerca il DO corrispondente. Quando il DO corrispondente non è stato eseguito, appare il messaggio d'errore "Loop Without DO" (Loop senza DO).
E' possibile costruire dei cicli nidificati avendo cura di chiudere il ciclo interno prima di quello esterno. Nel caso di più cicli nidificati, si può uscire da più cicli con una sola istruzione EXIT. Il numero che segue EXIT indica il numero dei cicli dai quali si deve uscire.

Esempi:

```
DO:PRINT"BUONGIORNO":LOOP          iterazione infinita
10 DO
20 INPUT A
30 IF A=0 THEN EXIT
40 PRINT A
50 LOOP
60 PRINT"FINITO"
```

Iterazione dalla riga 20 alla 40. Quando la risposta vale 0, si esce dal ciclo.

DOS ◆
comando

Ritorna alla pagina di testa perdendo il contenuto della memoria corrente per poter caricare il DOS del Basic 1.

DRAW stringa di caratteri ◆
istruzione

Traccia il disegno definito nella stringa di caratteri. Gli ordini elementari del linguaggio grafico eseguiti da DRAW sono i seguenti:

U (*su*), D (*giù*), R (*destra*), L (*sinistra*), E, F, G, H seguiti da un numero: spostamento del numero indicato nella direzione fissata. Se il numero vale 1, è possibile ometterlo.

D 30
Move: M colonna, riga
spostamento al punto situato in (colonna, riga)

M 100,1

M +/– C, +/– R
spostamento relativo di +/– C in colonne e di +/– R in righe. Il segno in questo caso è obbligatorio.

M+10, –10

Scala: S numero
cambiamento di scala (da 1 a 63). La scala normale è 4 (rapporto 1:1).

S 16

Arrow: A numero
fissa l'orientamento generale degli spostamenti ulteriori, effettua una rotazione sugli spostamenti ulteriori. Il numero da 0 a 3 corrisponde a: su (0), destra (1), giù (2), sinistra (3).

A 2

Cotor: C colore
fissa il colore del tracciato da –16 a 15.

C 3

Blank: B
posto davanti a un comando effettua lo spostamento senza tracciare.

BM 100, 150

No Update: N
posto davanti a un comando permette di tornare al punto di partenza dopo uno spostamento.

N U20

Execute: X variabile stringa
effettua il tracciato definito nella variabile stringa.

XAS;

Si può introdurre il valore di una variabile numerica mediante:

=variabile;
U=NB;

I comandi elementari possono essere separati da un punto e virgola o da uno spazio. Si può introdurre uno spazio anche tra un comando e il suo argomento.

```
DRAW"C1;BM150,150U20R20G20"
DRAW"U=NB1;L=NB2;"
DRAW"U30R30G30"
DRAW"U30"
```

DSKF (n. drive)
funzione

Dà il numero di Kbyte liberi sul dischetto inserito nel drive indicato.

Il numero di drive è obbligatorio.

Esempio:

DSKF(0) spazio libero sul dischetto nel drive 0

DSKI$ (n. drive, traccia, settore)
funzione

Effettua la lettura di un settore del quale viene indicato il numero, il numero della traccia e quello del drive dove esso si trova.

Il risultato è una stringa di caratteri di 128 byte a densità singola o 255 byte a densità doppia. Il numero di settore è compreso tra 0 e 4, il numero di traccia tra 0 e 79, il numero di settore tra 1 e 16 compresi. Qualsiasi tentativo di lettura al di fuori di questi valori visualizza il messaggio di errore "Illegal Function Call" (richiamo funzione non valida).

Esempio:

OSKI$(0,20,2) lettura del settore 2 della traccia 20 del drive 0

DSKINI n. drive, *fattore d'intreccio, nome*

Formatta il dischetto che si trova nel drive indicato.

Il dischetto non deve essere protetto in scrittura.

Durante la formattazione di un dischetto a due facce da 160K non bisogna spostare il mouse altrimenti la formattazione non avrà luogo correttamente.
La formattazione di un dischetto già registrato provoca la sua cancellazione integrale e tutti i dati in esso contenuti sono persi.
Se il dischetto è difettoso, appare il messaggio "Input/Output Error" (errore input/output).
Il fattore d'intreccio, che non è né obbligatorio né molto utile, permette di scegliere il numero di settore compreso tra due settori di numero consecutivo (implicitamente fissato a 7 in assenza di una precisazione).
Il numero di drive è obbligatorio. Il nome del dischetto, che non è obbligatorio, è una stringa di massimo otto caratteri.
In Basic 1 il nome del dischetto è vietato.
Se prima di tutto si esegue VERIFY ON, l'inizializzazione ha luogo con verifica e ha una durata maggiore.

Esempio:

DSKINI 1,6,"LAVORO" formatta il dischetto che si trova nel drive 1 con un intreccio di 6 e chiama questo dischetto LAVORO
DSKINI 0 formatta il dischetto che si trova nel drive 0

DSKO$ n. drive, traccia, settore, stringa caratteri
Istruzione

Deposita la stringa di caratteri nel settore del quale si specifica il numero, il numero della traccia e il numero del drive su cui questo si trova.

Il numero di drive deve essere compreso tra 0 e 4, il numero della traccia tra 0 e 79, il numero del settore tra 1 e 16. La stringa di caratteri deve avere una lunghezza massima di 128 caratteri a densità singola o 255 a densità doppia. Se la lunghezza è inferiore, il settore sarà riempito da caratteri di codice ASCII 0 nella parte vuota dopo l'ultimo carattere.

Esempio:

DSKO$ 0,18,3,HA$
DSKO$ 0,20,1,"GIOCHI" dà il nome GIOCHt al dischetto nel drive 0

END
Istruzione

Termina l'esecuzione di un programma.

Immediatamente al momento della sua esecuzione provoca dapprima la chiusura di tutti i file aperti, poi il passaggio alla modalità comando che viene indicata dal messaggio OK sullo schermo.
L'istruzione END non modifica né il contenuto delle variabili né le opzioni di CLEAR.
Può essere posta ovunque nel programma per terminare l'esecuzione. Non è indispensabile dopo l'ultima riga del programma.

EOF (n. canale)
funzione

Ritorna a –1 (VERO) se viene raggiunta la fine del file, altrimenti ritorna a 0 (FALSO).

Questa funzione evita il messaggio di errore "Input Past End" (input dopo la fine) nella lettura di un file.

Esempio:

```
10'LETTURA
20 OPEN"I",#1,"DATI"
30 DO
40 IF EOF(1) THEN EXIT
50 INPUT#1,A:PRINT A
60 LOOP
```

Variabili ERL e ERR

ERL e ERR sono delle variabili particolari che permettono, quando viene individuato un errore durante l'esecuzione di un programma, di conoscere il numero della riga dell'errore individuato e il numero dell'errore per poterlo correggere. Queste variabili sono riservate a questo uso e ogni tentativo di assegnare dei valori ad esse tramite un'istruzione d'assegnazione o una istruzione di lettura è vietato e individuato come un errore di sintassi.
Quando viene individuato un errore, si presentano due casi:
- Non viene indicato alcun trattamento dell'errore; l'esecuzione del programma è interrotta e il messaggio d'errore appare sullo schermo.
- Un trattamento d'errore viene attivato per mezzo dell'istruzione ON ERROR. In questo caso, la parte di programma incaricata del trattamento dell'errore viene eseguita. Il numero della riga dove si è prodotto l'errore è contenuto nella variabile ERL e il numero dell'errore è contenuto in ERR. Così le istruzioni del modulo di trattamento dell'errore possono conoscere il tipo d'errore. La lista dei codici d'errore è fornita nell'Appendice 7.

Esempio:

```
5 ON ERROR GOTO 100
10 DO
20 INPUT X
30 A=1/X
40 PRINT"L'inverso è";A
50 LOOP
100' modulo di trattamento d'errore
110 IF ERR=11 THEN PRINT"Dare un numero più grande"
120 IF ERR=6 THEN PRINT"Dare un numero più piccolo"
130 RESUME 20
```

ERROR numero
istruzione

Simula un errore di codice esistente o meno.

Il numero è convertito in intero per arrotondamento e dà il numero dell'errore. Questo errore simulato potrà sia provocare l'emissione del messaggio d'errore, sia essere trattato dal modulo designato da ON ERROR. Se non è eseguita alcuna

istruzione ON ERROR, appare il messaggio d'errore o, se si tratta di un codice sconosciuto all'interprete Basic, il messaggio "Undefined Error" (errore non definito).

Esempio:

```
10 S=10
20 T=5
30 ERROR S+T'simulazione dell'errore 15
RUN
String Too Long in 30
```

EVAL (stringa di caratteri) *
funzione

Rende il valore fornito dalla valutazione della stringa codificata da CRUNCH$.

Il risultato della valutazione è uguale a quello ottenuto dalla stessa espressione in un programma. Se la stringa non è un'espressione codificata da CRUNCH$, appare il messaggio "Syntax Error".

Esempio:

```
10 LINE INPUT A$
20 ? EVAL (CRUNCH$ (A$))
RUN
? 3^2+4^2
25
```

EXEC indirizzo, *elenco di parametri*
istruzione

Permette di eseguire un modulo scritto in linguaggio macchina residente nella memoria centrale.

Se l'indirizzo non è precisato, viene utilizzato quello dell'ultimo EXEC o l'indirizzo di esecuzione dell'ultimo LOADM che viene preso per default. Se l'indirizzo è compreso tra &H6000 e &H9FFF, viene indirizzato il banco di memoria fissato da BANK o per default il banco di numero più elevato.
Al modulo possono essere trasmessi dei parametri, introdotti dopo l'indirizzo e separati da virgole.
I parametri possono essere valutati chiamando, nel programma binario, i moduli di analisi dell'interprete Basic. Il richiamo si effettua mediante JSR indirizzo. Ogni richiamo valuta un parametro.
L'elenco degli indirizzi dei moduli di analisi viene fornito nell'Appendice 4.
Il passaggio dei parametri è vietato in Basic 1.
Il modulo in linguaggio macchina deve terminare con l'istruzione RTS per provocare il ritorno al programma Basic. I registri S (puntatore di stack) e DP (pagina diretta) non devono essere stati modificati.

EXIT numero *
Istruzione

Istruzione di uscita da un ciclo DO...LOOP o FOR...NEXT.

L'esecuzione di questa istruzione provoca il salto alla prima istruzione che segue LOOP o EXIT.
Il numero indica da quanti cicli si deve uscire. Per default, questo valore è 1. Un valore 0 non permette di uscire dal ciclo.

Esempio:

```
10 I=0
20 DO
30 I=I+1:PRINT I
40 IF INKEY$<:>""THEN EXIT
50 LOOP
```

Il ciclo (incremento di una variabile, visualizzazione del suo valore) termina quando si preme un tasto.

EXP(numero)
funzione

Ritorna l'esponenziale del numero (dando l'espressione e^ numero).

Ricordiamo che e vale 2.712828.
In Basic 128, il risultato è a doppia precisione se l'argomento è a doppia precisione.
Si oltrepassa la capacità quando il numero supera 58.02969.

Esempio:

```
?EXP(2)
7.38906
```

FIELD #n. canale, lunghezza 1 AS variabile di campo 1,
lunghezza 2 AS variabile di campo 2...
Istruzione

Definisce l'organizzazione in campi di record di un file ad accesso diretto del quale è dato il numero di canale.

Per ogni campo, si specifica la sua lunghezza in caratteri e il nome della variabile di campo che lo designa. Questa variabile è sempre una variabile stringa. La somma delle lunghezze non deve superare la lunghezza del record del file, che è stata fissata al momento dell'apertura (OPEN). Se la lunghezza non è stata fissata in OPEN, è di 255 caratteri (o 128 a densità singola). L'istruzione FIELD viene ge-

neralmente eseguita una volta, dopo l'apertura del file mediante OPEN. Tutti i record sono quindi letti e scritti secondo questo formato. Tuttavia è possibile eseguire delle istruzioni FIELD differenti per leggere record di formato differente o anche utilizzare più FIELD per leggere o scrivere nel medesimo record secondo formati differenti. Le variabili di campo così definite non possono essere manipolate come variabili qualsiasi. Il loro contenuto è sempre utilizzabile per essere visualizzato, trasformato o assegnato a un'altra variabile. Invece, si può depositarvi un valore solo tramite le istruzioni LSET, RSET e MID$()=. Queste variabili stringa possono anche ricevere dei numeri dopo che questi sono stati convertiti dalle funzioni MKI$ (numeri interi), MKS$ (numeri reali a precisione singola) o MKD$ (numeri a precisione doppia). La dimensione di un numero convertito da queste funzioni è di:
– 2 byte se è intero
– 4 byte se è a precisione singola
– 8 byte se è a precisione doppia

Esempio:

FIELD#2,15 AS NOME$,4 AS Q$ definisce, per il canale 2, due campi: il primo di quindici caratteri chiamato NOME$, il secondo di quattro caratteri chiamato Q$. A Q$ è possibile attribuire un numero a precisione singola.

FILES numero di canali, *lunghezza*. **dimensione area protetta**
istruzione

Riserva il numero di canali utilizzabili simultaneamente in un programma.

Questo numero non può essere superiore a 15.
Ogni canale riservato occupa una zona di 281 byte (o 153 a densità singola) per le informazioni concernenti il file e la zona di transito dei record.
All'inizializzazione, il numero di canali è fissato a 2. Se l'istruzione FILES reca un'indicazione di lunghezza (non obbligatoria), questa lunghezza fissa la somma delle lunghezze dei record dei file ad accesso diretto aperti simultaneamente. Questa lunghezza è fissata a 256 byte all'inizializzazione (due file con record di 128 byte).
Per esempio, per poter aprire tre file ad accesso diretto le cui lunghezze di record sono rispettivamente 30, 40 e 50 caratteri, occorre riservare una lunghezza di almeno 120 caratteri. L'istruzione FILES azzera tutte le variabili esistenti. Deve essere dunque eseguita in modalità diretta o all'inizio del programma.

L'argomento dimensione area protetta riserva in memoria una zona tampone di n tracce, con n che può variare da 0 a 25, per ottimizzare gli accessi fisici al QDD o al dischetto. I valori di default sono n=3 per un QDD e 0 per un dischetto.

Esempio:

FILE 3,120

FIX(numero)
funzione

Ritorna la parte intera del numero.

La parte intera viene ottenuta per troncamento. Il risultato è un numero reale. Il risultato di FIX equivale a quello di INT per i numeri positivi, ma è diverso in caso di numeri negativi.

Esempio:

```
?FIX(3 4),INT(3.4)
3 3
?FIX(-3.6),INT(-3.6)
-3 -4
```

FKEY$(numero) *
funzione

Ritorna il codice del tasto funzione di un numero indicato.
Il numero è una cifra compresa tra 1 e 10.

Esempio:

```
?FKEY$(3)
```

FOR...NEXT
FOR variabile = **valore iniziale TO valore finale** *STEP passo*
...
NEXT *variabile, variabile*

Istruzione d'iterazione che permette di ripetere una sequenza d'istruzione per un numero prestabilito di volte.

La variabile che segue FOR è chiamata variabile di controllo e può essere numerica di tipo intero o a precisione singola. Ad ogni istruzione FOR deve corrispondere una istruzione NEXT, ma ad un'istruzione NEXT possono corrispondere più istruzioni FOR. In questo caso, le variabili di controllo di ciascuna istruzione FOR sono indicate dopo NEXT.

Dapprincipio, la variabile di controllo prende il valore iniziale indicato. Poi l'interprete verifica se la variabile di controllo è superiore al valore finale.
- Se è superiore, le istruzioni del ciclo non vengono eseguite e l'esecuzione prosegue dopo NEXT.
- Se è inferiore, le istruzioni comprese tra FOR e NEXT vengono eseguite.

L'incontro di NEXT provoca l'incremento della variabile di controllo del valore passo (STEP), e il ritorno al test precedente.

Il ciclo d'iterazione viene eseguito fino a quando la risposta al test è positiva. Nel caso di un passo negativo, l'interprete verifica se la variabile di controllo è inferiore al valore finale e la variabile di controllo viene decrementata del valore di STEP. Se l'opzione STEP non è precisata, l'incremento per default è 1.
In Basic 128, è possibile uscire dal ciclo tramite l'istruzione EXIT. L'esecuzione continua quindi dopo NEXT, anche se la variabile di controllo non ha raggiunto il valore finale.

Cicli nidificati

I cicli FOR possono essere nidificati, vale a dire un ciclo FOR può contenere un altro ciclo FOR utilizzando una variabile di controllo differente.
Se più cicli hanno la stessa istruzione finale, è possibile utilizzare un NEXT multiplo per tutti i cicli:

NEXT variabile1, variabile2....
Se la variabile di controllo viene omessa, l'istruzione NEXT corrisponde al ciclo FOR più vicino non ancora concluso.

Esempi:

```
10 K=10
20 FOR I=1 TO K STEP 2
30 K=K+10
40 PRINT I;K
50 NEXT I
```

Il valore finale della variabile di controllo rimane 10 benchè K sia stato modificato nel corso dell'esecuzione.

```
10 I=0:X=0
20 FOR I=3 TO 2
30 X=X+1
40 NEXT I
50 PRINT I;X
```

Il ciclo non è stato eseguito e la variabile di controllo mantiene il valore iniziale (3).

```
10 DIM A(20,10)
20 FOR I=1 TO 20
30 FOR J=1 TO 10
40 A(I,J)=1
50 NEXT J,I
```

Si sono così impostati due cicli nidificati per l'inizializzazione di una matrice.

```
100 FOR I=1 TO 20
110 IF A(I)=0 THEN EXIT
120 NEXT I
130 IF I=21 THEN PRINT"NESSUN ELEMENTO NULLO"
```

Si ricerca un elemento nullo in una matrice e si esce mediante EXIT.

FRE(dato)
funzione

Permette di conoscere lo spazio di memoria libero.

Sono disponibili quattro informazioni:
1. FRE(0) dà lo spazio totale disponibile in memoria.
2. FRE(1) dà lo spazio disponibile nella zona di lavoro dell'interprete (tra &H6100 e &H9FFF).
3. FRE(2) dà lo spazio disponibile per il programma e i dati (tra &HA000 e &HDFFF nei banchi di memoria).
4. FRE(A$) dà lo spazio disponibile per le stringhe. Questo spazio è fissato a 300 caratteri all'inizializzazione. Notare che FRE(0)=FRE(1)+FRE(2).
Basic 1 non fornisce FRE(1) né FRE(2).

Esempio:

```
10?FRE(0);FRE(1);FRE(2);FRE(A$)
20 X$="0123456789"
30?FRE(0);FRE(1);FRE(2);FRE(A$)
```

FWD numero *
istruzione

Sposta la tartaruga per lo spazio indicato.

Il numero deve essere compreso tra –255 e 255. Se il numero è negativo, la tartaruga erretra. Se la tartaruga è visibile, viene cancellata dalla sua vecchia posizione e appare nella nuova.
Se il tracciamento è attivo, lo spostamento viene visualizzato sullo schermo.
Tutte le modifiche di dimensione, direzione o rotazione sono prese in considerazione prima dello spostamento. La tartaruga spostata è l'ultima indicata da TURTLE. Se non è stata eseguita alcuna istruzione TURTLE, viene visualizzato il messaggio di errore "Illegal Function Call" (richiamo funzione non valida).

Esempio:

```
FWD 30        sposta la tartaruga di trenta punti nella direzione indicata.
```

GET (colonna 1, riga1) – *(colonna 2, riga 2)*,**elemento matrica ***
istruzione

Memorizza nella matrice numerica la porzione d'immagine grafica contenuta nel rettangolo definito dai due vertici (colonna 1, riga 1) e (colonna 2, riga 2).

La porzione d'immagine è obbligatoriamente composta da un numero intero di caratteri. Le coordinate dei due vertici sono quindi date in caratteri: colonne da 0 a 39

(o 79) e righe da 0 a 24. Se il secondo vertice non è stato precisato, viene preso per default il vertice in basso a destra (39,24) o (79,24) a 80 colonne.
La matrice numerica deve essere stata precedentemente dichiarata e deve essere di tipo intero. In una sola matrice possono essere memorizzate più immagini. Queste sono sistematicamente compattate e ordinate partendo dal basso della matrice, cioè dall'indice più alto verso l'indice più basso.
L'elemento matrice designato nell'istruzione GET è l'elemento d'indice più alto per questa immagine. L'immagine è memorizzata partendo dall'elemento precedente. Dopo l'esecuzione, l'elemento designato contiene il valore dell'indice del primo elemento libero della matrice. Questo elemento potrà servire per una nuova istruzione GET o LOADP con la stessa matrice.
L'immagine è compattata nella matrice, il volume che essa occupa dipende dal suo contenuto.
Se la matrice è troppo piccola, viene visualizzato "Out of Memory" (memoria esaurita).
La porzione d'immagine così memorizzata può essere trasferita in un altro punto dello schermo mediante PUT e può essere memorizzata in file mediante l'istruzione SAVEP. La modalità di compattazione è compatibile con COLORPAINT e PRAXITEL.

Esempio:

DIM T%(1000) dichiara la matrice di ricezione
GET(0,0)–(9,9),T%(1000) memorizza la porzione d'immagine partendo dalla fine della matrice
?T%(1000) contiene il primo elemento libero
817 e cioè 817
GET(0,10)–(9,19),T%(817) memorizza la seconda porzione partendo dal primo elemento libero
?T%(817) primo elemento libero nuovo
760 cioè 760 ... e così di seguito

GET (colonna 1, riga 1) – *(colonna 2, riga 2)*, **matrice ◆ istruzione**

Memorizza in una matrice numerica la porzione d'immagine grafica contenuta nel rettangolo definito dai due vertici (colonna 1, riga 1) e (colonna 2, riga 2). La matrice numerica (intera o a precisione singola) deve essere stata precedentemente dichiarata e la sua dimensione deve essere sufficiente per ricevere l'immagine, tenendo conto del fatto che un byte può ricevere un solo punto dell'immagine e che un elemento della matrice comprende 2 o 4 byte a seconda che la matrice sia intera o a precisione singola. La porzione d'immagine così memorizzata può essere trasferita in un altro punto dello schermo mediante PUT.

DIM T (625)
GET (150,80)–(190,140),T

GET#n. canale, *n. record*
Istruzione

Trasferisce un record da un file ad accesso diretto (di cui viene specificato il numero di canale) alla zona tampone in memoria centrale.

Il recupero del contenuto del record può quindi essere fatto tramite INPUT#, o direttamente nelle variabili di campo se precedentemente è stato eseguito un FIELD. Se il numero di record non è stato precisato, GET legge il record successivo del file o il primo se non è ancora stato letto o scritto alcun record.

Esempio:

GET#2,10 legge il decimo record del file n. 2

GOSUB numero riga
Istruzione

Permette di saltare ad un sottoprogramma.

L'esecuzione di un sottoprogramma richiamato tramite GOSUB inizia alla prima istruzione della riga indicata dal numero. L'esecuzione del sottoprogramma termina mediante l'istruzione RETURN, che provoca il ritorno all'istruzione immediatamente seguente al GOSUB di chiamata.
Un sottoprogramma può avere più punti d'entrata e può contenere più istruzioni RETURN.
Un sottoprogramma può richiamare un altro sottoprogramma e così di seguito fino a saturazione della memoria. Un sottoprogramma può richiamare se stesso.
Un tentativo di eseguire l'istruzione GOSUB verso una riga inesistente provoca la visualizzazione del messaggio di errore "Undefined Line" (riga non definita).
Un sottoprogramma può essere posto ovunque in un programma, ma è preferibile separarlo dal programma principale mediante un'istruzione END.

Esempio:

```
10 GOSUB 100
20 PRINT"primo ritorno"
30 GOSUB 100
40 PRINT"secondo ritorno"
50 END
100 PRINT"Sottoprogramma"
110 RETURN
RUN
```

```
10 INPUT N
20 GOSUB 100
30 PRINT"Fattoriale:";F
40 END
100 F=1
110 IF N=1 THEN RETURN
```

```
120 F=F*N:N=N-1:GOSUB 110
130 RETURN
```

calcolo periodico di fattoriale N

GOTO numero riga
istruzione

Permette il salto incondizionato ad una riga di programma.

Se la riga designata contiene un'istruzione eseguibile, questa sarà eseguita, quindi saranno eseguite le istruzioni seguenti.
Se la riga designata non contiene istruzioni eseguibili (REM o DATA), l'esecuzione continua dalla riga successiva. Se la riga designata non esiste, viene visualizzato il messaggio "Undefined Line".
In modalità diretta, questa istruzione permette di proseguire l'esecuzione di un programma interrotto a partire dal numero indicato. A differenza di "RUN numero riga", questa non inizializza le variabili e non chiude i file aperti. La scrittura di un ciclo è più facile con DO...LOOP.

Esempio:

```
10 PRINT"LOOP...INFINITO"
20 GOTO 10
```

ciclo infinito con GOTO

GRS(numero)
funzione

Ritorna il carattere utente del numero indicato.

I caratteri utente sono numerati da 0 a 127 e devono essere stati precedentemente dichiarati mediante CLEAR (terzo parametro) e definiti da DEFGR$.
GR$(i) equivale a CHR$(128+i).

Esempio:

```
?GR$(23)          visualizza il carattere utente n.23
```

HEAD*
funzione

Indica la direzione della tartaruga attiva.

Il risultato è un numero compreso tra 0 e 255.

La tartaruga attiva è l'ultima indicata da TURTLE.

Esempio:

```
?HEAD
62
```

HEAD *TO* **numero** *
istruzione

Imposta o modifica la direzione del movimento della tartaruga attiva.

Se è presente l'opzione TO, la direzione è impostata in modo assoluto, altrimenti viene modificata in modo relativo.
Il numero che indica la direzione è compreso tra –255 e 255. Se il numero è maggiore, si tiene conto solo del resto della divisione per 256 (modulo).
Un numero positivo significa spostamento verso destra, un numero negativo significa spostamento verso sinistra. La tartaruga attiva è l'ultima designata da TURTLE.

Esempio:

ROT 64	modifica la direzione di 90 gradi verso destra
ROT 320	idem
ROT TO –32	imposta la direzione a 45 gradi verso sinistra

HEX$ (numero)
funzione

Visualizza una stringa di caratteri che rappresenta il numero in notazione esadecimale (in base 16).

Se l'argomento non è intero, viene troncato. Il suo valore deve essere compreso tra –65536 e 65535.

Esempio:

```
?HEX$(65535)
FFFF
```

IF condizione { **GOTO n. riga** / **THEN istruzioni** } *ELSE* { *n. riga* / *istruzioni* }
Istruzione

Permette salti condizionati.

Questa istruzione si esegue nel modo seguente:
dapprima viene valutata la condizione.

– Se il risultato è VERO, le istruzioni che seguono THEN vengono eseguite e, se è presente ELSE, le istruzioni che lo seguono sulla riga vengono ignorate.
– Se il risultato è FALSO, le istruzioni che seguono ELSE vengono eseguite oppure, se ELSE è assente, l'esecuzione passa alla riga successiva del programma.
In ogni caso, se l'elenco delle istruzioni è sostituito da un numero di riga, avviene il salto alla riga indicata.

Esempio:

```
IF B*B–4*A*C < 0 GOTO 500          se negativo, salto alla riga 500
```

Istruzioni IF nidificate

E' possibile far figurare all'interno di una istruzione IF una o più istruzioni IF nel limite di una riga logica. L'interpretazione di IF nidificato è la seguente:
– ogni ELSE è associato all'istruzione THEN più vicina che non è associata ad un altro ELSE.

Esempio:

```
IF A<B THEN IF B<C THEN PRINT"A< C"ELSE PRINT"B>=C"
IF A<B THEN IF B<C THEN PRINT"A<C"ELSE PRINT"B>=C"ELSE PRINT
"A>=B"
```

Attenzione:
I test di uguaglianza dei numeri reali non sono sempre esatti poichè dipendono dalla precisione. Occorre dunque preferire a questi test i test che verificano una differenza minima.

Esempio:

```
IF ABS(A–B)<1E–5 THEN PRINT"A e B uguali"
```

INKEY$
funzione

Ritorna l'ultimo carattere battuto sulla tastiera o una stringa vuota se non è stato battuto alcun carattere.

Questa funzione non ha eco sullo schermo.
Tutti i caratteri, compresi i caratteri di controllo, tranne CTRL–C e STOP, sono presi in considerazione da INKEY$. In particolare, la pressione di ENTER produce il carattere Ritorno Carrello.

Esempio:

```
10 FOR I=1 TO 1000:NEXT I
20 A$=INKEY$
30 PRINT"CARATTERE:";A$
```

INMOUSE variabile1, variabile2 *
INPUTMOUSE variabile1, variabile2
Istruzioni

Leggono le coordinate del mouse.

INMOUSE effettua una lettura immediata, INPUTMOUSE attende che venga premuto uno dei due pulsanti.

Esempio:

```
INMOUSE X,Y
```

INPEN variabile colonna, variabile riga
INPUTPEN variabile colonna, variabile riga
Istruzioni

Permettono di leggere le coordinate del punto dello schermo puntato dalla penna ottica.

INPEN effettua una lettura immediata, INPUTPEN attende che il contatto della penna ottica sia interrotto per effettuare la lettura.
La prima variabile riceve la coordinata orizzontale (tra 0 e 319), la seconda variabile riceve la coordinata verticale (tra 0 e 199).
In modalità 80 colonne, il valore della colonna è un numero pari compreso tra 0 e 638.
Quando non è possibile effettuare la misurazione, alle due variabili viene dato il valore –1. Questa situazione si presenta in caso di:
- cattiva regolazione della penna ottica,
- luminosità insufficiente dal punto mirato (nero o rosso per esempio),
- puntamento al di fuori della zona utile dello schermo.

Esempio:

```
100 DO
110 INPUTPEN C,R
120 IF C<>1 THEN EXIT
130 LOOP
140 ?C,R
```

INPUT *stringa di caratteri;* **elenco variabili**
INPUT *stringa di caratteri;* **elenco variabili**
Istruzione

Istruzione di introduzione dati da tastiera.

Il messaggio (se esistente) contenuto nella stringa di caratteri compare sullo schermo seguito da un punto interrogativo e da uno spazio se la delimitazione è

un punto e virgola. Il punto interrogativo non è raffigurato se ta delimitazione è una virgola. I caratteri successivi che compaiono sullo schermo sono obbligatoriamente di misura normale.
L'utente deve quindi introdurre i dati previsti dall'elenco delle variabili. Questi dati devono essere dello stesso tipo delte variabili dell'elenco, altrimenti compare sullo schermo il messaggio "Redo from start" (reintrodurre dati). I diversi dati sono separati da virgole. L'introduzione di una stringa deve essere racchiusa tra virgolette se contiene una virgola o degli spazi significativi all'inizio o alla fine. Le variabili possono essere etementi di matrice ma non matrici.
INPUT non può essere utilizzato in modalità diretta.

Esempio:

```
10 INPUT"GIORNO,MESE,ANNO";G%,M%,A%
```

INPUT# n. canale, elenco variabili
istruzione

Legge il file del quale si precisa il n. di canale e attribuisce i valori letti alle variabili menzionate nell'elenco.

Il tipo dei dati (numerici o stringa) deve corrispondere al tipo delle variabili. Questa istruzione segue le stesse regole che INPUT richiede per i dati introdotti da tastiera. Se le variabili sono più numerose dei dati, viene visualizzato il messaggio "Input Past End". Per evitare l'errore, utilizzare la funzione EOF.

Esempio:

```
INPUT#2,A$,N                    legge i dati in A$ e N in successione
```

INPUT$ (numero caratteri, *#n. canale)*
funzione

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

Questa funzione permette di leggere un numero determinato di caratteri sul canale considerato e permette di conoscere in modo esatto il contenuto di un file.
Se il n. di canale non è precisato, la lettura viene fatta partendo dalla tastiera senza eco sullo schermo. CTRL–C e STOP vengono trasmessi e non interrompono l'esecuzione. In fase di introduzione da tastiera questa istruzione permette di evitare di premere il tasto ENTER.

Esempio:

```
INPUT$(3,#1)        lettura di 3 caratteri sul fite 1
```

INPUTTURTLE variabile colonna, variabile riga *
Istruzione

Permette di conoscere le coordinate della tartaruga attiva.

La prima variabile riceve il n. di colonna, la seconda il n. di riga. I valori ricevuti vanno da –32768 a 32767. La tartaruga attiva è l'ultima designata da TURTLE.

Esempio:

```
10 TURTLE 1
20 DO
30 FWD 10:HEAD 16
40 INPUTTURTLE C,R:LOCATE 0,0:PRINT C,R
50 LOOP
```

A ciascuno spostamento, la riga 40 legge le coordinate della tartaruga e te fa comparire sullo schermo.

INPUTWAIT n. riga; durate, *stringa di caratteri, elenco variabili**
INPUTWAIT n. riga; durata, *stringa di caratteri;elenco variabili*
Istruzioni

Permettono di teggere dei dati partendo da tastiera, specificando il tempo assegnato per la risposta.

Il primo parametro indica la riga da eseguire se il tempo è passato.
Il secondo parametro indica in secondi il tempo massimo a disposizione per battere la risposta.

Il resto dell'istruzione è identico a INPUT. In particolare, la stringa di caratteri deve eseere seguita da un punto e virgola, se si desidera che sullo schermo compaia un punto interrogativo. tt formato ridotto è il seguente:
INPUTWAIT n. riga; durata, "",
che permette di ottenere una temporizzazione seguita da un salto ad un'altra istruzione.

Esempio:

```
10 PRINT"INIZIO"
20 DO
30 INPUTWAIT 100;10,"Valore di A";A
40 PRINT A
50 LOOP
60 END
100 PRINT"Troppo lento"
110 GOTO 10          attende la risposta per 10 secondi

INPUTWAIT 900;5,"",          attende 5 secondi e salta alta riga 900
```

INSTR *(posizione,* **stringa 1, stringa 2)**
funzione

Ritorna la posizione della stringa 2 nella stringa 1.

Se il primo parametro è presente, la ricerca parte dalla posizione indicata, altrimenti viene fatta partire dal primo carattere della stringa 1.
Il risultato della funzione INSTR è 0:
– se la stringa 2 non figura nella stringa 1,
– se la stringa 1 è vuota,
– se la posizione di partenza è superiore alla lunghezza della stringa 1.
Se la stringa 2 è vuota, il risultato della funzione è la posizione di partenza (default=1).
Se la stringa 2 è presente nella stringa 1, la funzione INSTR ritorna la sua posizione.

Esempio:

```
?INSTR("E' BELLO","E")
2
?INSTR(6,"E' BELLO","E")
0
?INSTR(1,"E' BELLO","")
1
```

INT (numero)
funzione

Ritorna il numero intero più alto inferiore o uguale al numero specificato.

Il risuttato è un numero reale. L'argomento può essere intero, reale o a doppia precisione.

Esempio:

```
?INT(76.75)
76
?INT(-76.1)
-77
```

Per arrotondare al numero intero più vicino, bisogna utilizzare: INT(X+.5)

INTERVAL ON *
INTERVAL OFF
istruzioni

Permettono di avviare o arrestare il temporizzatore definito dall'istruzione ON INTERVAL.

KILL descrittore di file
comando

Cancella il file in questione, vale a dire cancella i suoi riferimenti nell'indice e libera lo spazio che esso occupa sul dischetto.

Nella descrizione del file, il nome della periferica non è obbligatorio, ma è obbligatoria l'estensione. La periferica di default è il drive 0 o quella impostata da DEVICE. L'istruzione KILL funziona solo per un file su dischetto. Se il file non esiste, appare il messaggio "File Not Found" (file non trovato).

Esempio:

KILL "1:BUONGIORNO.BAS" sopprime il file BUONGIORNO.BAS nel dischetto 1

LEFT$ (stringa, lunghezza)
funzione

Visualizza la parte sinistra della stringa per la lunghezza specificata.

Se la lunghezza è nulla, il risultato è una stringa vuota. Se la lunghezza richiesta è superiore alla lunghezza della stringa, il risultato è l'intera stringa.

Esempio:

```
?LEFT$("MELODRAMMA",4)
MELO
```

LEN (stringa)
funzione

Ritorna la lunghezza della stringa.

Vengono contati tutti i caratteri della stringa. Non si deve dimenticare che le lettere minuscole accentate sono codificate su tre caratteri.

Esempio:

```
?LEN("CASA")
4
?LEN("è")
3
```

LET variabile=espressione *
istruzione

Permette di attribuire un valore ad una variabile.

La parola chiave LET è facoltativa.
La variabile e l'espressione devono essere dello stesso tipo, altrimenti viene visualizzato il messaggio "Type Mismatch" (errore tipo).

Esempio:

```
10 D=12
20 SOMMA=D+20
30 A$="BELLA"
40 B$=A$+"VISTA"
```

LiNE *(c1,r1)*–**(c2,r2) carattere,** *colore, sfondo, inversione*
LiNE *(c1,r1)*–**(c2,r2),** *colore*
istruzione

Traccia una linea tra i due punti specificati.

Questa istruzione funziona in modalità carattere quando l'argomento carattere è presente.
La sua sintassi è identica a quella di BOX.

Esempio:

LINE(1,9)–(15,15)"+",3,0 traccia una linea di + tra i due punti di colore 3 e di sfondo 0
LINE(100,60)–(163,96)2 traccia una linea di colore 2 tra i due punti grafici
LINE–(200,200) la linea parte dall'ultimo punto tracciato

Vedere anche STEP.

LiNE iNPUT *stringa di caratteri*;**variabile stringa**

LiNE INPUT *stringa di caratteri*,**variabile stringa**
istruzione

Permette di introdurre in una variabile stringa una sequenza contenente al massimo 255 caratteri qualsiasi.

La fine della risposta è data dal tasto ENTER. Alla variabile stringa vengono assegnati tutti i caratteri battuti ad eccezione del carattere ritorno carrello.
La presenza di una stringa di caratteri provoca la visualizzazione del messaggio in essa contenuto.

Non viene visualizzato il punto interrogativo sia che la stringa sia seguita da una virgola sia che sia seguita da un punto e virgola.
E' molto pratico usare questa istruzione per leggere una riga di programma di cui non si conosce la struttura.

Esempio:

```
10 LINE INPUT "CHE COSA?";AS
20 PRINT A$
```

LINE INPUT *#n. canale,* **variabile stringa**
istruzione

Legge il record successivo del file in una variabile stringa prendendo tutti i caratteri.

Esempio:

```
LINE INPUT #2,X$
```

LIST *descrittore del file, n. riga 1 – n. riga 2*
comando

Produce il listato del programma contenuto in memoria, su video o sul file scelto.

Le righe di programma da listare vengono indicate nel modo seguente:

LIST riga	lista la riga
LIST.	lista la riga corrente
LIST riga1–riga2	lista tutte le righe dalla riga1 alla riga2
LIST riga1–	lista tutte le righe dalla riga1 alla fine
LIST –riga2	lista tutte le righe dall'inizio alla riga2

Se il file è un file su disco, LIST equivale a SAVE con l'opzione ,A.
Se non viene precisato nessun file, il listato viene visualizzato sul video.

Esempio:

LIST "LPRT:(80)",1000–19990 invia il listato sulla stampante parallela dalla riga 1000 alla riga 1990
LIST "t:PG3.BAS" invia il listato di tutto il programma al file PG3.BAS
LIST invia il listato di tutto il programma ai video.

LOAD descrittore file,*R*
comando

Carica in memoria il programma di cui è stato precisato il nome e il supporto su cui si trova.

Il comando LOAD chiude tutti i file aperti, cancella il programma dalla memoria, quindi effettua il caricamento vero e proprio del programma.
Vengono annullate anche tutte le variabili.
Con l'opzione ,R viene eseguito immediatamente il programma caricato e i file che erano già aperti prima del caricamento restano aperti. La periferica di default è il drive 0, ovvero quello che era stato impostato da DEVICE.

Esempio:

LOAD "2:OTELLO",R carica il programma OTELLO dal dischetto 2, lascia i file aperti e lancia l'esecuzione
LOAD "TICTAC" carica il programma TICTAC

LOADM descrittore file,*traslazione,R*
istruzione

Carica una zona di memoria in binario a partire da un file creato da SAVEM o dall'assemblatore.

Il caricamento viene effettuato nel banco corrente. Se viene indicata una traslazione, vengono spostati da questa traslazione i dati o il programma. La traslazione sposta anche l'indirizzo di esecuzione.
Se si tratta di un programma, l'opzione R ne provoca l'esecuzione immediata.

Esempio:

LOADM "OROLOGIO",&H200,R carica OROLOGIO.BIN, lo sposta di 512 byte e lancia la sua esecuzione all'indirizzo dato in SAVEM-512.
LOADM "FORMATO" carica FORMATO.BIN

LOADP descrittore file,elemento di matrice *
istruzione

Carica nella matrice l'immagine contenuta nel file specificato.

Si ottiene un risultato simile a quello ottenuto con l'istruzione GET.
La matrice numerica deve essere dichiarata precedentemente e deve essere di tipo intero.
In una stessa matrice possono essere memorizzate diverse immagini che vengono automaticamente compattate e disposte a partire dalla base della matrice, e cioè in modo descrescente partendo dall'indice maggiore.
Se non è espressa l'estensione del file, viene assunta per default l'estensione .MAP.

Esempio:

Per recuperare un'immagine Colorpaint:

DIM IM%(2000)

LOADP"0:MONALISA",IM%(2000)
PUT (0,0),IM%(2000) carica il file immagine MONALISA.MAP dal drive 0 nella matrice IM% partendo dall'elemento 999. Se l'immagine occupa 400 byte, dopo il caricamento IM%(1000) ne contiene 799.

LOC(n. canale)
funzione

Indica in che punto avviene la scrittura (o la lettura) in un file aperto tramite il numero di canale specificato.

Se il file è ad accesso sequenziale, LOC indica il numero di settori scritti o letti a partire dall'apertura del file. Se 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#.

Esempio:

LOC(2) posizione della lettura o della scrittura in corso sul file 2

LOCATE *colonna,riga,cursore*
Istruzione

Fissa la posizione del cursore sullo schermo in modalità carattere.
Il primo parametro indica la colonna e deve essere compreso tra 0 e 39 (o 79). Il secondo indica la riga e deve essere compreso tra 0 e 24.
Il terzo parametro rende invisibile il cursore (valore 0) oppure lo rende visibile (valore 1). Questi tre parametri sono facoltativi nel BASIC 128. Il posizionamento del cursore definito con LOCATE può avvenire anche fuori dalla finestra di visualizzazione definita da CONSOLE.

Esempio:

LOCATE 10,15:PRINT "BUONASERA"

LOF (n. canale)
funzione

Dà la posizione dell'ultimo record nel file aperto tramite il canale specificato.

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

Esempio:

LOF(1) posizione dell'ultimo record del file 1

LOG(numero)
funzione

Dà il logaritmo neperiano del numero in base e.

Nel BASIC 128, il risultato viene dato con un numero a doppia precisione se l'argomento è a doppia precisione.

Esempio:

```
?LOG(5)
1.60944
```

LOOP*
istruzione

Segna la fine di un ciclo e trasferisce l'esecuzione al DO corrispondente.

(Vedere anche l'istruzione DO).

LSET variabile stringa=stringa di caratteri
istruzione

Assegna a una variabile stringa una stringa di caratteri giustificandola a sinistra.

La stringa di caratteri viene impostata partendo da sinistra. Se la lunghezza della stringa è inferiore rispetto alla dimensione della variabile, la parte rimanente viene completata con degli spazi. Se la lunghezza è maggiore, viene eliminata la fine della stringa. E' possibile assegnare dei numeri a una variabile stringa dopo averli convertiti con:

MKI$()	per un numero intero
MKS$()	per un numero a precisione singola
MKD$()	per un numero a doppia precisione *

Questa istruzione è usata molto spesso per depositare dei dati in una variabile di campo definita con FIELD.
Nel BASIC 1 la variabile deve essere una variabile di campo.

Esempio:

LSET ID$=NOME$+COGN$ assegna a ID$ la concatenazione di NOME$ e COGN$
LSET Q$=MKS$(QU) assegna il valore numerico di QU dopo la conversione.

MAX(elenco di numeri) *
funzione

Ritorna il valore massimo contenuto in un elenco di numeri.

I numeri possono essere a singola o doppia precisione.

Esempio:

```
?MAX(10,12,5.6#)
12
```

MERGE descrittore file,*R*
comando

Carica in memoria il programma indicato e lo aggiunge al programma residente.

Le righe di programma caricate sono classificate con quelle del programma residente e si sovrappongono se sono identiche. Il programma da caricare deve essere salvato con l'opzione A.
L'esecuzione del nuovo programma è lanciata alla fine del caricamento con l'opzione R.

Esempio:

MERGE "1:TEST",R carica il programma TEST dal dischetto 1, lo fonde al programma in memoria e lancia l'esecuzione.
MERGE "MENU" fonde il programma MENU al programma residente.

MID$(stringa,posizione,*lunghezza)*
funzione

Estrae una sottostringa della lunghezza specificata dalla stringa indicata.

posizione è un numero compreso tra 1 e 255, lunghezza è un numero compreso tra 0 e 255.
Se lunghezza è di valore troppo alto, o se non è presente questo parametro, la sottostringa arriva fino alla fine della stringa. Se posizione è maggiore della lunghezza della stringa, il risultato è una stringa vuota.

Esempio:

```
?MID$("PATATRAC",3,2)
TA
?MID$("PATATRAC",5)
TRAC
```

MID$(variabile stringa,inizio,*lunghezza*)=stringa di caratteri
istruzione

Permette di sostituire una parte di una variabile stringa con un'altra stringa.

La lunghezza della variabile stringa resta comunque invariata. Il secondo parametro indica la posizione del primo carattere da sostituire nella variabile.
Il terzo parametro, facoltativo, indica la lunghezza della parte da sostituire.
– Se la stringa di sostituzione è più lunga della parte da sostituire, viene usato solo l'inizio di questa stringa.

Esempio:

```
10 A$="MICRO-COMPUTER"
20 MID$(A$,6,8)="PROCESSORE"
30 PRINT A$
RUN
MICROPROCESSOR
```

– Se non viene precisata la lunghezza, la sostituzione viene fatta fino alla fine della variabile, se è sufficientemente lunga la stringa di modifica.

Esempio:

```
10 A$="SAN LUIGI"
20 MID$(A$,5)="GIOVANNI"
30 PRINT AS
RUN
SAN GIOVA
```

Questa istruzione può essere usata per modificare il valore di una variabile di campo definita in FIELD.

MIN(elenco numeri) *
funzione

Ritorna il valore minimo contenuto in un elenco di numeri.

I numeri possono essere a singola o a doppia precisione.

Esempio:

```
?MIN(12,3,4,5,6)
3,4
```

MKD$(numero) *
funzione

Permette di convertire un numero a doppia precisione in una stringa di caratteri per

poterlo assegnare successivamente con l'istruzione LSET o RSET a una variabile stringa.

La stringa di caratteri così ottenuta è lunga 8 byte. Questa stringa di caratteri è codificata e non può assere letta direttamente.

Esempio:

```
MKDS(TOTALE #)
MKDS(1.2345678901234)
```

MKI$(numero)
funzione

Permette di convertire un numero intero in una stringa di caratteri per poterlo assegnare successivamente con l'istruzione LSET o RSET a una variabile stringa.

La stringa di caratteri così ottenuta è lunga due byte.

Esempio:

```
MKI$(Q%)
MKI$(-12536)
```

MKS$(numero)
funzione

Permette di convertire un numero a precisione singola in una stringa di caratteri per poterlo assegnare successivamente con l'istruzione LSET o RSET a una variabile stringa.

La stringa di caratteri cos ottenuta è lunga quattro byte.

Esempio:

```
MKS$(X)
MKS$(6.023E+23)
```

MOTOR ON/MOTOR OFF
istruzioni

Queste istruzioni permettono di avviare o arrestare il motore del registratore a cassette.

Esempio:

```
MOTOR ON
```

MTRiG(numero) *
funzione

Ritoma lo stato del pulsanle del mouse.

Il numero designa il pulsante: 0 (sinistra)
1 (destra)

Il risultato è –1 (VERO) se il pulsante è premuto o 0 (FALSO) in caso contrario.

Esempio:

IF MTRIG(0) THEN PRINT "FUOCO"

NAME descrittore file 1 AS descrittore file 2
comando

Permette di cambiare il nome di un file su dischetto.

Il descrittore del file 1 contiene il nome del file da ridefinire.
Il descrittore del file 2 contiene il nuovo nome dei file.
il nuovo nome non deve corrispondere a quello di un file già esistente.
Nel BASIC 128 è possibile indicare un commento insieme al nome del file.

Esempio:

NAME "1:TIC.BAS" AS "1:TAC.BAS"
NAME "TESTO.DAT" AS "(note)SUPERATO.DAT"

NEXT elenco variabili
istruzione

Incrementa o decrementa le variabili di controllo di uno o più cicli FOR...NEXT.

(Per il funzionamento di questa istruzione vedere FOR).

NEW
comando

Cancella il programma presente in memoria e tutte le variabili, comprese quelleri-servate in COMMON.

Questo comando non chiude i file aperti e non modifica le opzioni stabilite prece-dentemente con CLEAR.

Esempio:

NEW

OCT$(numero) *
funzione

Trasforma in ottale (base 8) il numero contenuto in una stringa di caratteri.

Se l'argomento non è intero, viene troncato.

Esempio:

```
?OCT$(30)
36
```

ON ERROR GOTO n. riga
istruzione

Permette il salto condizionato alla riga di programma indicata se nel corso dello svolgimento dell'esecuzione del programma è stato individuato un errore.

Viene effettuata a partire dalla riga indicata la sequenza di trattamento degli errori e vengono sospese le interruzioni di ON INTERVAL.

Abitualmente questa sequenza si conclude con RESUME. L'istruzione ON ERROR GOTO di solito si trova all'inizio del programma, ovvero all'inizio delle sequenze dove potrebbe verificarsi un errore. ON ERROR GOTO 0 annulla l'effetto di ON ERROR GOTO n. e riporta allo stato normale "messaggio d'errore" e "interruzione dell'esecuzione".
Anche l'istruzione CLEAR annulla l'effetto di ON ERROR GOTO n. .

Esempio:

```
10 ON ERROR GOTO 100
20 INPUT X
30 PRINT 1/X
40 ON ERROR GOTO 0
50 END
100 IF ERR=11 THEN PRINT "Divisione impossibile"
110 RESUME 20
```

ON espressione GOSUB elenco n. riga
ON espressione GOTO elenco numero riga
istruzioni

Permettono il salto condizionato a una riga o a un sottoprogramma la cui riga è determinata dal valore dell'aspressione.

L'esprassione viene valutata e il suo vatore, dopo essere stato arrotondato all'intero più vicino, definisce il livello nel listato del sottoprogramma da richiamare. Se per esempio quasto valore è 4, l'esecuzione passa alla riga o al sottoprogramma indicato dal quarto numero.

Se il valore dell'espressione è nullo oppure maggiore del numero delle righe del listato, l'esecuzione passa all'istruzione seguente.
E' possibile omettere i numeri di riga che corrispondono ai valori non utilizzati dell'espressione.

Esempio:

```
10 INPUT "Numero";N
20 ON N GOSUB 100,,200
30 END
100 PRINT "SOTTOPROGRAMMA 1"
110 RETURN
200 PRINT "SOTTOPROGRAMMA 2"
210 RETURN
```

a seconda che si risponda 1 o 2, viene eseguito il sottoprogramma corrispondente.

Esempio:

```
10 INPUT "Numero";N
20 ON SGN(N)+2 GOTO 100,200
30 PRINT "Positivo"
40 END
100 PRINT "Negativo"
110 END
200 PRINT "Zero"
210 END
```

ON INTERVAL=numero GOSUB n. riga *
ON INTERVAL=numero GOTO n. riga
istruzioni

Permettono di inserire un'interruzione logica sul clock interno. Il numero indica in decimi di secondo il valore del clock che attiva l'interruzione.
L'esecuzione di INTERVAL ON lancia il conteggio sul clock interno. Il clock conta in decimi di secondo. Quando viene raggiunto il numero indicato, viene sospesa l'esecuzione del programma e viene eseguita la sequenza delle istruzioni che cominciano con il numero indicato.
Se viene usata l'istruzione GOSUB, la sequenza deve terminare con RETURN che riprende l'esecuzione del programma là dove era stata sospesa.
L'esecuzione di INTERVAL OFF interrompe il conteggio e di conseguenza sospende le interruzioni. L'esecuzione della sequenza di istruzioni può essere interrotta da un'altra interruzione.
Se la sequenza di istruzioni è abbastanza lunga, è necessario eseguire INTERVAL OFF all'inizio per evitare che il programma sia interrotto durante l'esecuzione.

Esempio:

```
10 ON INTERVAL=1800 GOTO 100
20 INTERVAL ON
30 DO:LOOP
100 PRINT "E' COTTO"
110 END
```

Il messaggio viene visualizzato esattamente tre minuti dopo l'esecuzione della riga 20.

ON KEY=carattere GOSUB n. riga *
ON KEY=carattere GOTO n. riga
istruzioni

Permettono di definire interruzioni logiche battendo un carattere particolare.

Il carattere è indicato da una costante di un solo carattere ("A") o dal primo carattere di una costante ("ABCDE") o dal suo corrispondente in codice ASCII (65). Dopo aver eseguito questa istruzione, premendo il tasto corrispondente al carattere si sospende l'esecuzione e si toglie il controllo alla riga di comando indicata. Il ritorno al punto dell'interruzione viene dato da RETURN, se viene usata l'istruzione GOSUB. Con questa istruzione è possibile specificare dieci tasti diversi. I tasti specificati non possono essere letti da INKEY$. Se il numero di riga è uguale a 0, l'interruzione richiesta si disattiva con il tasto relativo.

Esempio:

```
10 ON KEY="?" GOSUB 100
20 DO:PRINT "Qualsiasi cosa":LOOP
30 END
100 PRINT "Ecco alcune spiegazioni:...."
110 RETURN
```

ONPEN GOSUB elenco n. riga
ONPEN GOTO elenco n. riga
istruzioni

Permettono il salto condizionato (GOTO) o un richiamo del sottoprogramma (GOSUB) a seconda dell'area letta dalla penna ottica.

Le aree dello schermo devono essere definite precedentemente con l'istruzione PEN.
L'esecuzione di ONPEN si svolge nel modo seguente:
- verifica del contatto della penna ottica
- quando il contatto è interrotto, viene letta l'area puntata
- salto alla riga il cui numero ha la stessa priorità nel listato di quella dell'area letta con l'istruzione PEN.

Il livello nel listato viene contato a partire da 0. Se la penna ottica legge un punto situato fuori dalle aree definite, l'esecuzione del programma continua in sequenza.

Esempio:

```
10 CLS
20 PEN 0:(10,15)-(30,35)
30 PEN 1:(40-15)-(60,35)
40 PEN 2:(70,15)-(90,35)
50 ONPEN GOSUB 100,200,300
70 END
100 PRINT "AREA 1"
110 RETURN
200 PRINT "AREA 2"
210 RETURN
300 PRINT "AREA 3"
310 RETURN
```

OPEN "I",#n. canale, descrittore file
OPEN "O",#n. canale, descrittore file
istruzioni

Aprono un file sequenziale in lettura o in scrittura tramite il numero di canale indicato.

Tutte le operazioni su questo file dovranno essere compiute tramite questo numero di canale. Il numero di canale deve essere compreso tra 1 e 16. La periferica di default è il drive 0 oppure quello fisssto da DEVICE.

OPEN "I" apre il file in lettura. Uno stesso file può essere aperto in lettura contemporaneamente su diversi canali.

Esempio:

OPEN "I",#2, "1:RISUL.DAT" apre in lettura il file RISUL.DAT sul drive 1 con il numero di canale 2.

OPEN "O" apre il file in scrittura. Uno stesso file può essere aperto solo una volta in scrittura. Su disco, l'apertura del file comporta l'inizializzazione del file; se il file esisteva già con questo nome, viene cancellato.

Esempio:

OPEN "O",#1,"2:TELEF.DAT" apre in scrittura il file TELEF.DAT sul drive 2 con il numero di canale 1.
OPEN "O",#3,"LPRT:" apre in scrittura il file numero 3 sulla stampante parallela.

OPEN "D",# n. canale,descrittore file,lunghezza
OPEN "R",# n. canale,descrittore file,lunghezza
Istruzioni

Aprono il file ad accesso diretto di cui è indicato il nome, in lettura e in scrittura, tramite il numero di canale indicato.

Il numero di canale deve essere compreso tra 1 e 16. La lunghezza, facoltativa, stabilisce la lunghezza del record. Questa lunghezza deve essere inferiore o uguale alla lunghezza indicata in FILES. Se non viene precisata la lunghezza in OPEN, il record occupa un settore, cioè 128 byte nei dischetti a densità singola o 255 byte nei dischetti a doppia densità.
La periferica di default è il drive 0 oppure quello impostato da DEVICE.
Si può aprire il file ad accesso diretto solo sui dischetti. I due formati OPEN "D" e OPEN "R",... sono equivalenti.

Esempio:

OPEN "D2,#2,"1:AGENDA.VEC",32 apre il file ad accesso diretto AGENDA.VEC tramite il numero di canale 2, con una lunghezza del record di 32 caratteri.

PAINT(colonna,riga),*colore* *
Istruzione

Riempie con il modello corrente un'area di colore omogenea partendo dal punto situato in (colonna,riga) con il colore precisato.

Se non è stato precisato il colore, viene assunto per default il colore corrente.

Il riempimento viene effettuato con il modello definito dall'istruzione PATTERN.
L'area viene riempita per default in modo uniforme. In tutti i casi, questo comporta la variazione di un solo colore, primo piano o sfondo a seconda del segno.

Esempio:

PAINT(100,125),3 l'area viene riempita a partire dal punto 100,125 con il colore giallo.

PAINT(colonna,riga),*colore* ◆
Istruzione

Riempie una forma chiusa partendo dal punto situato in (colonna,riga) con il colore specificato. Il colore deve essere lo stesso del contorno che delimita la superficie da colorare.

L'ultimo parametro, che dovrà avere un valore 0 o 1 permette di dipingere con effetto ombreggiato se il valore dato è 1. In questo caso viene illuminato solo un punto su due e in questo modo si ottiene l'effetto ombreggiato. Se viene omesso

questo parametro, viene assunto come valore di default il valore corrente. Il suo valore iniziale è 0.

Esempio:

```
PAINT(100,125),3
PAINT(30,30)
```

Vedere anche STEP.

PALETTE(numero) *
funzione

Ritorna il valore reale del colore del codice indicato.

Il risultato è un numero compreso tra –4096 e 4095. Questo numero corrisponde al colore reale associato al numero indicato. Se il numero è negativo, il colore corrispondente sarà trasparente.

Esempio:

```
?PALETTE(3)
748
```

PALETTE numero,colore reale,*trasparenza* *
Istruzione

Assegna un colore reale al numero di colore indicato.

Esistono 4096 colori reali (da 0 a 4095). Possono essere usati contemporaneamente solo 16 di questi colori.
Questa istruzione permette di assegnare un colore reale (numero compreso tra 0 e 4095) a un colore utilizzabile (numero compreso tra 0 e 15).
Il secondo parametro può essere indicato in codice positivo (colore del primo piano) o negativo (colore dello sfondo). Il terzo parametro, facoltativo, indica se il colore così definito deve essere trasparente (valore 1) o non trasparente (valore 0). Se non viene espresso questo parametro, il colore mantiene lo stato precedente.

Esempio:

```
PALETTE 3,748,1    assegna il colore 748 al numero 3 e lo rende trasparente
PALETTE 14,2970    assegna il colore 2970 al numero 14.
```

PATTERN carattere *
Istruzione

Stabilisce il modello di riempimento.

Il carattere che verrà usato come modello di riempimento può essere dato sotto forma di stringa di caratteri (si considera il primo carattere della stringa) oppure del suo codice ASCII corrispondente. Se il codice ASCII è maggiore di 127, si tratta di un carattere grafico definito dall'utente.
Il modello viene usato successivamente in tutte le istruzioni di riempimento: BOXF, CIRCLEF, PAINT.

Esempio:

PATTERN"/"	il carattere / servirà da modello
PATTERN 129	il secondo carattere grafico servirà da modello
PATTERN GRS(1)	come sopra

PEEK(Indirizzo)
funzione

Ritorna il contenuto del byte che si trova all'indirizzo indicato.

L'indirizzo è un numero compreso tra –65536 e 65535. Il risultato è un numero intero compreso tra 0 e 255. In Basic 128, se l'indirizzo è compreso tra &H6000 e &H9FFF, il byte è quello del banco corrente, cioè quello che è stato stabilito da BANK o, per default, il banco maggiore.

Esempio:

BANK 1:PRINT PEEK(&H6100) dà il contenuto del byte che si trova in &H6100 nel banco 1.

PEN *elenco di aree*
con area=
numero;*(colonna1,riga1)–(colonna2,riga2)*
istruzione

Definisce aree rettangolari sullo schermo e attribuisce loro un numero per l'utilizzo di ONPEN.

Il numero delle aree è compreso tra 0 e 7.
Le aree possono essere definite in modalità carattere o in modalità grafica.

- modalità carattere:

Ogni area corrisponde ad un solo carattere ed è definita da un solo punto le cui coordinate vengono date in caratteri: colonne da 0 a 39 e righe da 0 a 24.

Esempio:

PEN3;(30,15),4;(32,15) definisce le aree 3 e 4 in modalità carattere

- modalità grafica

Ogni area viene definita dai due vertici opposti del rettangolo come nell'istruzione

BOX. Le coordinate sono espresse in modalità grafica: colonne da 0 a 319 e righe da 0 a 199.

Esempio:

PEN0;(10,10)–(40,40),1;(50,10)–(80,40) definisce le due aree numerate 0 e 1.

Se viene definito un numero di area senza che questo sia seguito dalle coordinate, viene eliminata l'area e non può più essere selezionata con ONPEN.
Un'istruzione PEN senza argomento comporta l'eliminazione di tutte le aree definite.
Se le due aree hanno una parte in comune, quando la penna ottica legge questa parte, ONPEN permette il salto corrispondente all'area caratterizzata dal numero inferiore.

PLAY elenco delle stringhe di caratteri
Istruzione

Produce la musica definita nelle stringhe di caratteri.

Le stringhe di caratteri possono produrre solo delle note, delle pause o degli attributi.
– note: DO RE MI FA SO LA SI seguite da # o b se si vuole un diesis o un bemolle,
– pausa: P
– attributi: e cioè attacco, durata, ottava e tempo.

Significato	Attributo	Campo	Valore di default
smorzamento	A	0–255	0 nota continua
durata	L	1–96	24 semiminima
ottava	O	1–5	4
tempo	T	1–255	5

Smorzamento:
A0 corrisponde ad un suono continuo, A255 a un suono molto smorzato.
Durata:
L96 corrisponde a una semibreve, L48 a una minima,..., L3 a una semibiscroma.
Ottava:
O1 è l'ottava più bassa, O5 l'ottava più acuta.
Tempo:
T1 è il tempo più veloce, T255 è il tempo più lento.
Un attributo A,L,O,T agisce su tutte le note che seguono fino a quando non interviene un altro attributo a modificare il suo valore.
Gli spazi sono ammessi in un qualsiasi punto della stringa; il punto e virgola può servire per separare le note.

Esempio:

```
10 AS="O5 DOREMIDOREPREMIFAFAMIPDOREMIDOREPREMIFASODO"
20 PLAY A$,AS
```

L'interruzione di una sequenza PLAY con il BASIC 128 può essere effettuata tramite la pressione simultanea dei tasti CTRL e C per alcuni secondi.

POINT (colonna,riga)
funzione

Ritorna il numero del colore del punto indicato.

Le coordinate sono indicate in modalità grafica: colonna da 0 a 319 e riga da 0 a 199.
Il numero è positivo se il punto appartiene al primo piano, negativo se appartiene allo sfondo.

POKE indirizzo,numero
istruzione

Deposita il numero nel byte che si trova all'indirizzo indicato.

Il numero deve essere compreso tra 0 e 255.
L'indirizzo deve essere compreso tra –65536 e 65535.
Nel Basic 128, se l'indirizzo è compreso tra &H6000 e &H9FFF, il byte è quello del banco corrente e cioè quello impostato da BANK oppure è per default il banco maggiore.
Non è possibile usare questa istruzione sulla ROM.

POS(n. canale)
funzione

Ritoma la posizione del puntatore sulla riga.

La posizione più a sinistra viene indicata con 0. Il numero di canale indica la periferica interessata. Se questo numero è uguale a 0 o viene omesso, la funzione ritoma la posizione del cursore sullo schermo (numero compreso tra 0 e 39 o 79).
La funzione CSRLIN permette di conoscere il numero di riga in cui si trova il cursore.

PRINT *elenco dati*
istruzione

Visualizza su video i dati secondo l'ordine dell'elenco.

La parola chiave PRINT può essere sostituita con un punto interrogativo. La visualizzazione inizia dal punto in cui si trova il cursore. Un punto e virgola permette di visualizzare il dato successivo subito dopo quello precedente.
Una virgola permette di visualizzare il dato seguente all'inizio della tabulazione seguente. Le tabulazioni fisse prevedono una spaziatura di tredici caratteri a partire dal carattere 0. Se l'elenco dei dati non termina con una virgola o con un punto e virgola, la visualizzazione termina con il passaggio alla riga successiva.

Se i dati occupano più di 40 (o 80) colonne, la visualizzazione continua alla riga successiva.
Se un dato deve essere visualizzato alla fine della riga e non resta abbastanza spazio, la visualizzazione avviene automaticamente sulla riga successiva. In questo caso, le lettere minuscole accentate vengono considerate come se fossero tre caratteri.
I numeri sono sempre seguiti da uno spazio. I numeri positivi sono preceduti da uno spazio, i numeri negativi dal segno meno. Il numero viene visualizzato con l'esponente se il valore del numero è al di fuori dell'intervallo compreso tra 1E–6 e 1E+6 per i numeri a precisione singola, o 1D–16 e 1D+16 per i numeri a doppia precisione.
Le stringhe di caratteri vengono visualizzate come sono. Con i caratteri a doppia altezza, il ritorno a capo provoca un'interlinea doppia. Tuttavia la prima visualizzazione avviene sulla riga del cursore e sulla riga superiore, quindi spesso è necessario farla precedere da un'istruzione PRINT vuota.

Esempio:

```
10 X=1
20 PRINT X,X+1,X+2
30 PRINT X;X+1;X+2
40 PRINT "X=";X
```

PRINT#n. canale,*elenco dati*
Istruzione

Scrive i dati (variabili o costanti) in un file di cui viene precisato il numero di canale. I dati vengono registrati nello stesso modo in cui vengono visualizzati su video con PRINT (spazi, separatori,...). Il file può essere sequenziale o ad accesso diretto. In quest'ultimo caso PRINT# scrive nel buffer di registrazione. Il trasferimento della registrazione nel file avviene con PUT #.

Esempio:

```
PRINT #2,A$,N
PRINT #1,"ARTURO
```

PRINT USING stringa di caratteri;elenco dati
istruzione

Permette visualizzazioni secondo un formato descritto nella stringa di caratteri.

I dati dell'elenco sono separati con delle virgole o dei punti e virgola.
La stringa di caratteri contiene caratteri che precisano il formato della visualizzazione. Il significato di questi caratteri è il seguente:

formato di stringhe di caratteri *

! significa che deve essere visualizzato solo il primo carattere della stringa.
%% significa che la lunghezza esatta dell'area di visualizzazione è la stessa di quella compresa tra i due segni di percentuale (%).

La stringa è giustificata a sinistra al momento della visualizzazione, dopo che sono stati eliminati i caratteri in eccesso o dopo aver completato lo spazio restante con degli spazi.
& significa che la stringa deve essere visualizzata tale e quale. Questo permette di evitare salti di riga in caso di una stringa troppo lunga.

Esempio:

```
10 A$="CHE BELLA GIORNATA"
20 PRINT USING "!";A$
30 PRINT USING "%%";A$
40 PRINT USING "&";AS
RUN
C
CH
CHE BELLA GIORNATA
```

formato del numeri

rappresenta la posizione di ogni numero. Il numero viene giustificato a destra al momento della visualizzazione. Se l'area è troppo piccola, il superamento della stessa viene indicato con la visualizzazione del carattere % all'inizio del numero. Se l'area è troppo grande, il resto è completato con degli spazi.
. rappresenta il punto decimale. Può essere inserito in un qualsiasi punto per separare la parte intera dalla parte decimale di un numero. Se è necessario, i numeri vengono arrotondati prima della visualizzazione.

Esempio:

```
10 AS=###.##
20 PRINT USING A$;12
30 PRINT USING A$;-12.345
40 PRINT USING A$;.1234
50 PRINT USING A$;1234.5678"
RUN
12
-12.35
0.12
%1234.57
```

+ all'inizio o alla fine del formato significa che il segno + deve sempre precedere o seguire un numero positivo. – alla fine del formato significa che il segno – deve sempre seguire un numero negativo.

Esempio:

```
PRINT USING "+###.#";12.34
+12.3
PRINT USING "##.#-";-12.34
12.3-
```

** all'inizio del formato significa che gli spazi iniziali devono essere sostituiti con degli asterischi.

Esempio:

```
PRINT USING "**##.#";1.23
```

```
***1.2
```

$$ all'inizio del formato significa che il segno dollaro deve essere visualizzato immediatamente a sinistra del numero.

Esempio:

```
PRINT USING "$$##.#";1.23
```

```
$1.2
```

**$ all'inizio del formato combina i due effetti precedenti.

Esempio:

```
PRINT USING "**$##.#";1.23
```

```
***$1.2
```

, a sinistra del punto decimale significa che le migliaia devono essere separate da una virgola nella parte intera.

Esempio:

```
PRINT USING "########,.##";1234567
1,234,567.00
```

^^^^ alla fine del formato significa che deve essere usata la notazione esponenziale.

Esempio:

```
PRINT USING "##.## ^^^^";123.45
1.23E+02
```

formato generale *

Il formato generale deve contenere tanti formati elementari quanti sono i dati da visualizzare. I formati elementari possono essere separati con delle costanti che saranno visualizzate tra i dati. Se il formato generale non contiene abbastanza formati elementari, viene ripreso all'inizio per la visualizzazione degli altri dati che seguono.

Esempio:

```
PRINT USING "HO VENDUTO###PROGRAMMI A###.##DOLLARI"
;53,100.50
HO VENDUTO 53 PROGRAMMI A 100.50 DOLLARI
PRINT USING "##.#";1.23,2.34;3.45
1.2 2.3 3.5
```

PRINT#n. canale, USING stringa di caratteri;elenco dati
istruzione

Funziona come l'istruzione PRINT #, ma i dati vengono scritti seguendo il formato precisato da USING.
I dati vengono registrati nello stesso modo in cui vengono visualizzati su video con PRINT USING.

Esempio:

PRINT #1,USING "####.##";120.5

PSET (colonna,riga) *carattere,colore,sfondo,inversione*
PSET (colonna,riga), *colore*
istruzione

Attiva un punto che si trova in una data colonna e in una data riga.
Questa istruzione funziona in modalità carattere quando l'argomento carattere è presente e in modalità grafica quando è assente. In modalità carattere, la colonna è un numero compreso tra 0 e 39 (o 79) e la riga è un numero compreso tra 0 e 24. In modalità grafica, la colonna e la riga sono dei numeri compresi tra –32768 e 32767 nel Basic 128. Nel Basic 1, la colonna è un numero compreso tra 0 e 319 e la riga un numero compreso tra 0 e 199.
La parte rimanente della sintassi è identica a quella di BOX.

Esempio:

PSET (10,20)"X",3,0 attiva il carattere X alla colonna 10, riga 20, con colore giallo su sfondo nero.
PSET (55,115) attiva il punto grafico che si trova alla colonna 55, riga 115 con il colore corrente.

Vedere anche STEP.

PTRIG
funzione

Ritorna il valore VERO (–1) se è interrotto il contatto della penna ottica, in caso contrario ritorna il valore FALSO (0).

PUT (colonna,riga),elemento matrice*
latruzione

Riporta su schermo la parte dell'immagine conservata nella matrice a partire dall'elemento indicato.

La parte dell'immagine è composta obbligatoriamente da un numero intero di caratteri. Le coordinate del vertice sono quindi date in caratteri: la colonna da 0 a 39 (o 79) e la riga da 0 a 24. Il vertice indica l'angolo superiore sinistro dell'immagine. L'immagine deve essere stata memorizzata precedentemente nella matrice numerica intera con un'istruzione GET, oppure caricata nella matrice con un'istruzione LOADP.
Se la parte dell'immagine è troppo grande non viene visualizzata.

Esempio:

PUT (10,12),IM%(300) riporta su schermo la parte di immagine conservata nella matrice a partire dall'elemento 299, al punto che si trova alla colonna 10, riga 12.

PUT (colonna 1, riga 1)–(colonna 2, riga 2), ◆ matrice
Istruzione

Riporta su schermo la parte di immagine contenuta nella matrice, nel rettangolo di cui vengono dati i vertici opposti: colonna 1, riga 1 e colonna 2, riga 2. L'immagine memorizzata precedentemente nella matrice numerica da un'istruzione GET deve corrispondere alle stesse dimensioni (numero di punti in colonne e righe).

PUT#n. canale,n. record
Istruzione

Trasferisce un record di un file ad accesso diretto (di cui viene dato il numero di canale) dall'area tampone al dischetto.
Il contenuto del record deve essere stato depositato precedentemente con l'istruzione WRITE# o PRINT#, oppure direttamente nelle variabili di campo con LSET, RSET o MID$()=. Se non è stato precisato il numero di record, PUT# scrive nel record successivo del file oppure nel primo se non è ancora stato letto e scritto alcun record.

Esempio:

PUT #1,12 deposita il dodicesimo record del file n. 1

READ elenco variabili
Istruzione

Legge i dati delle istruzioni DATA e li assegna alle variabili dell'elenco.
Le variabili devono essere dello stesso tipo dei dati da leggere. In caso contrario viene visualizzato il messaggio "Syntax Error" (errore di sintassi).
Una sola istruzione READ può leggere diverse righe DATA e viceversa una riga DATA può essere letta da diverse istruzioni READ.
La prima istruzione READ comincia a leggere i dati dalla prima istruzione DATA del programma. Le altre letture vengono fatte in sequenza.

Se non vi sono dati sufficienti, appare il messaggio "Out of Data" (Dati esauriti).
E' possibile iniziare o ricominciare la lettura a partire da una determinata riga di programma servendosi dell'istruzione RESTORE.

Esempio:

```
10 READ COGN$,NOME$,ETA'
20 PRINT COGN$,NOME$,ETA'
100 DATA ROSSI,LUCIANO,77
```

REM *testo*
istruzione

Permette di inserire il testo di osservazioni, titoli, commenti in un programma.
La parola chiave REM può essere sostituita con un apostrofo. Tutto quello che segue REM fino alla fine della riga viene ignorato dall'interprete ma compare nel listato del programma.
E' possibile saltare a una riga di programma che contiene una REM; l'esecuzione inizia a partire dalla prima riga di programma che segue la REM.
E' possibile aggiungere alla fine della riga dei commenti separandoli dalle istruzioni con un apostrofo.

Esempio:

```
10 REM Titolo del programma 1 gennaio 1986
20 PRINT "Buongiorno" 'messaggio di benvenuto
30 A=2+2 'calcolo
40 PRINT "Buonasera" :REM messaggio di cortesia
```

RENUM *nuovo n. 1a riga* *vecchio n. 1a riga* *vecchio n., passo ultima riga*
comando

Permette di rinumerare un programa oppure una parte di programma.
Tutte le indicazioni sono facoltative e sono previsti dei valori di default.
Il primo numero indica il nuovo numero (il valore di default è 10).
Il secondo numero indica a partire da quale numero (vecchio) bisogna iniziare a rinumerare il programma (il valore di default è la prima riga del programma).
Il terzo numero indica il vecchio numero dell'ultima riga (default=fine del programma).
Il quarto numero indica lo scarto tra ogni nuova riga (default=10).
RENUM cambia di conseguenza in tutto il programma tutti i numeri delle righe rinumerate nei salti condizionati e nei richiami dei sottoprogrammi. Se nella riga R1 (vecchio numero) si richiama una riga R2 inesistente, appare il seguente messaggio: Undefined line R2 in R1 dove R1 è il numero della riga rinumerata.

Le virgole sono indispensabili, ad eccezione dell'ultima. La rinumerazione di una parte di programma è possibile solo se le nuove righe restano all'interno dello spazio tasciato dalle righe invariate: il nuovo numero 1 della prima riga rinumerata deve rimanere maggiore del numero della riga invariata che lo precede e il nuovo numero dell'ultima riga rinumerata deve restare inferiore al numero della riga invariata che segue.

Esempio:

RENUM 1500, 1000, 1340, 5 rinumera la parte di programma compresa tra 1000 e 1340 con un passo 5 cominciando dalla riga 1500.
RENUM 100. 10 rinumera il programma a partire da 10 con un passo 10 cominciando da 100.
RENUM rinumera tutto il programma con un passo 10 cominciando da 10.

RESET *
istruzione

Inizializza il microcomputer.

Dopo questa istruzione si ritorna al menu iniziale.

Esempio:

RESET

RESTORE *n. riga*
istruzione

Permette di rileggere delle costanti in un'istruzione DATA della riga specificata.

Dopo un'istruzione RESTORE, il primo READ prende i valori dal primo DATA che incontra a partire da questa riga. Se non viene indicato alcun numero di riga, il primo READ prende i valori a partire dal primo DATA nel programma.

Esempio:

```
10 READ A,B
20 PRINT A,B
30 RESTORE
40 READ A
50 PRINT A
100 DATA 12,23,34
```

RESUME *n. riga*
RESUME *NEXT*
Istruzione

Permette di ritornare al programma principale dopo una sequenza di trattamento di errori richiamata con ON ERROR GOTO.

L'esecuzione riprende dal numero di riga indicato, se è presente, oppure dall'istruzione che ha provocato l'errore, se non viene precisato.
Se è presente l'opzione NEXT, l'esecuzione riprende dall'istruzione che segue quella che ha provocato l'errore. Se si incontra questa istruzione senza aver richiamato ON ERROR GOTO, viene segnalato un errore "Resume without Error".

Esempio:

```
10 ON ERROR GOTO 100
20 INPUT X
30 PRINT SQR(X)
40 ON ERROR GOTO 0
50 END
100 IF ERR=5 THEN "Numero negativo"
110 RESUME 20
```

RETURN
Istruzione

Permette di ritornare al programma principale dopo un'istruzione GOSUB.

(Vedere anche l'istruzione GOSUB).

RIGHT$(stringa di caratteri,lunghezza)
funzione

Visualizza la parte destra della stringa per la lunghezza precisata.
Se la lunghezza è nulla, il risultato è una stringa vuota. Se la lunghezza richiesta è maggiore della lunghezza della stringa, il risultato è ta stringa intera.

Esempio:

```
?RIGHT$(MELODRAMMA",6)
DRAMMA
```

RND*(numero)*
funzione

Ritorna un numero reale, pseudoaleatorio, compreso tra 0 e 1 esclusi.

t numeri ottenuti con questa funzione sono diversi e suddivisi uniformemente nell'intervallo aperto 0,1 se l'argomento della funzione è un numero positivo o se non è riportato l'argomento. La sequenza dei numeri generati viene inizializzata nuovamente ad ogni esecuzione del programma.
Un argomento negativo inizializza nuovamente ta sequenza a partire da un vatore che dipende dall'argomento.

Esempio:

```
10 DO
20 FOR I=1 TO 5
30 PRINT RND
40 NEXT I
50 PRINT RND(-2)
60 LOOP
RUN
```

ROT *
funzione

Ritorna l'orientamento della tartaruga attiva.

Il risultato è un numero compreso tra 0 e 255. Un risultato positivo indica l'orientamento in senso orario. La tartaruga attiva è l'ultima fissata da TURTLE.

Esempio:

```
?ROT
167
```

ROT *TO* **numero ***
Istruzione

Fissa o modifica l'orientamento della tartaruga attiva.

Se è presente l'opzione TO, l'orientamento è fissato in modo assoluto. In caso contrario è modificato relativamente al suo valore precedente.
Il valore assoluto del numero è compreso tra 0 e 255. Se questo valore è maggiore, viene preso in considerazione solo il resto della sua divisione per 256 (modulo).
Un numero positivo indica un orientamento verso destra, un numero negativo verso sinistra.
La tartaruga attiva è l'ultima fissata da TURTLE.

Esempio:

ROT TO –32	fissa l'orientamento a –45 gradi verso sinistra
ROT 64	modifica l'orientamento di 90 gradi verso destra
ROT 320	produce la stessa modifica.

RSET variabile stringa=stringa di caratteri
istruzione

Deposita una stringa di caratteri in una variabile stringa giustificandola a destra.

La stringa di caratteri è depositata cominciando da destra. Se la sua lunghezza è inferiore alla lunghezza della variabile, la parte rimanente è completata a sinistra con degli spazi; se la sua lunghezza è maggiore, viene eliminata la fine della stringa. I numeri possono essere depositati in una variabile stringa dopo essere stati convertiti con:
MKI$() per un numero intero
MKS$() per un numero a precisione singola
MKD$() per un numero a doppia precisione.
Questa istruzione viene usata molto spesso per depositare dati in una variabile di campo definita con FIELD. Nel Basic 1, la variabile deve essere una variabile di campo.

Esempio:

RSET ID$=NOME$+COGN$ deposita nella variabile ID$ il contenuto delle due stringhe NOME$ e COGN$ giustificando a destra.
RSET Q$=MKI$(1234) deposita in Q$ il numero 1234 dopo la conversione.

RUN *n. riga*
RUN descrittore file,*R*
comando

Lancia l'esecuzione del programma residente in memoria dopo il suo eventuale caricamento.

Se è indicato un numero di riga, l'esecuzione del programma residente comincia dalla riga specificata. Se è indicato un file, viene caricato in memoria il programma contenuto nel file che viene poi eseguito a partire dall'inizio. Con l'opzione R, i file già aperti prima del lancio del programma restano aperti.

Esempio:

RUN 100 lancia l'esecuzione dalla riga 100
RUN "1:OTELLO",R carica il programma OTELLO e lo esegue a partire dall'inizio pur mantenendo tutti i file aperti.

SAVE descrittore file
SAVE descrittore file,A
SAVE descrittore file,P
Istruzione

Salva il programma presente nella memoria centrale con il nome e sul supporto indicati.

Sul disco, se esisteva già un file con quel nome, il file viene sostituito dal nuovo programma.
L'estensione di default è BAS. Con l'opzione A viene salvato il programma in formato letterale, identico al listato su video, che potrà essere allora utilizzato da MERGE.
Con l'opzione P (protetto) il programma viene salvato in formato codificato. Al momento del caricamento successivo, il programma non potrà essere né listato né modificato. Si consiglia di salvarlo anche nel formato abituale.

Esempio:

SAVE "CASS:OTELLO",P salva su cassetta il programma residente con il nome OTELLO.BAS, nel formato protetto.
SAVE "TICTAC",A salva il programma nel formato letterale (ASCCI)
SAVE "PROVA" salva normalmente il programma

SAVEM descrittore file,indirizzo 1,indirizzo 2,indirizzo 3
Istruzione o comando

Salva una parte di memoria nel file binario.

L'indirizzo 1 indica l'inizio dell'area da salvare, l'indirizzo 2 indica la fine dell'area da salvare, l'indirizzo 3 indica l'indirizzo del lancio dell'esecuzione al momento del caricamento con LOADM servendosi dell'opzione R oppure con EXEC senza parametri.
Se questi indirizzi sono compresi tra &H6000 e &H9FFF, l'area di memoria si trova nel banco di memoria corrente. SAVEM può servire a conservare un'area di memoria precisa oppure un programma in binario.

Esempio:

SAVEM "OROLOGIO",&H8800.&H8900,&H8800 salva l'area binaria compresa tra &H8800 e &H8900.

SAVEP descrittore di file, elemento matrice *
Istruzione

Salva in un file una parte di video compattata in una matrice.

La matrice deve essere una matrice di numeri interi e può contenere diverse immagini che però possono essere salvate una sola per volta.
L'immagine deve essere compattata a partire dall'elemento indicato precedentemente con un'istruzione GET.
L'estensione di default è MAP.
L'immagine può essere recuperata successivamente con l'istruzione LOADP e visualizzata con PUT.

Esempio:

SAVEP "EINSTEIN", IM%(300) salva l'immagine compattata a partire dall'elemento 299 nella matrice IM% con il nome EINSTEIN.MAP.

SCREEN(colonna,riga)
funzione

Ritorna il codice ASCII del carattere visualizzato nella colonna e nella riga indicate.

Le coordinate vengono date in caratteri: colonna da 0 a 39 e riga da 0 a 24.
Questa funzione riconosce solo i caratteri della tabella ASCII e le lettere minuscole accentate. Se il carattere non viene riconosciuto, il risultato è 0.
Vengono attribuiti alle lettere minuscole accentate i seguenti codici:

129 à
133 è
134 é
137 ì
141 ò
145 ù

Solo BASIC 128:

149 °
150 §

Esempio:

```
10 LOCATE 2,4:PRINT "A"
20 PRINT SCREEN(2,4)
RUN
A
65
```

SCREEN primo piano,*sfondo,contorno,inversione,trasparenza*
Istruzione

Imposta i colori della visualizzazione per tutto lo schermo. Tutti i parametri sono facoltativi ma almeno uno deve essere presente.
Nella modalità a 80 colonne, è possibile modificare i colori nel Basic 128 solo con l'istruzione PALETTE.
I primi tre parametri indicano rispettivamente:
– il colore dei caratteri e delle linee tracciate
– il colore dello sfondo
– il colore del contorno dello schermo.
Se è presente il quarto parametro, significa che devono essere invertiti i colori di primo piano e di sfondo.

Se è presente il quinto parametro, significa che l'immagine TV deve essere trasparente. Questa opzione non funziona se non con l'interfaccia apposita. La mancanza dell'interfaccia, provoca un errore fatale.

Esempio:

SCREEN t,0,5 caratteri rossi su sfondo nero e contorno magenta
SCREEN,,,1 inverte i colori precedenti

SCREENPRINT
istruzione

Invia il contenuto del video alla stampante parallela.

Questa istruzione funziona soltanto con le stampanti PR90–582 e PR90–600.

Esempio:

SCREENPRINT

SEARCH stringa di caratteri, *n. riga 1 – n. riga 2*
comando

Visualizza sullo schermo tutte le righe di programma in cui è presente la stringa di caratteri specificata, dalla riga 1 alla riga 2 compresa.

La stringa di caratteri da ricercare può essere data in una costante o in una variabile.
E' possibile effettuare la ricerca su tutto il programma o su una sua parte oppure sulla riga corrente usando una sintassi indentica a quella di LIST.

Esempio:

SEARCH "NOME$", 1000–2000 ricerca la variabile NOME$ nelle righe di programma comprese tra 1000 e 2000
SEARCH "NBJ",. ricerca NBJ nella riga corrente
SEARCH "520" ricerca 520 in tutto il programma

SGN(numero)
funzione

Riforna il segno del numero

Il risultato è

— 1 se il numero è positivo
— 0 se il numero è nullo
— – t se il numero è negativo.

Esempio:

```
PRINT SGN(–2)
–1
```

SHOW *
funzione

Indica se la tartaruga attiva è visibile (risultato –1) o invisibile (risultato 0).

Esempio:

```
?SHOW
–t
```

SHOW *visibile,libera* *
Istruzione

Stabilisce lo stato della tartaruga attiva.

Il primo parametro stabilisce se è visibile o meno.
Se viene impostato 0, la tartaruga è invisibile: se viene impostato t, la tartaruga è visibile.
Il secondo parametro stabilisce se la tartaruga è libera o meno. Se viene impostato il valore 1, la tartaruga viene visualizzata ad ogni modifica.
Se viene impostato il valore 0, la tartaruga è visuafizzata solo ad ogni spostamento.

Esempio:

```
SHOW 1,1    la tartaruga attiva è invisibile e libera
```

SIN(numero)
funzione

Ritorna il seno del numero calcolato in radianti.

Nel Basic 128, il risultato è dato in doppia precisione se l'argomento è a doppia precisione.

Esempio:

```
PRINT SIN(t.5)
 997495
```

SKIPF *"nome file"*
istruzione

Ferma il nastro della cassetta dopo il file indicato oppure dopo il primo file incontrato se non è stato specificato alcun nome di file.

Esempio:

SKIPF "TICTAC" ferma il nastro dopo il file TICTAC

SPACE$(numero)
funzione

Ritorna una stringa di caratteri composta da tanti spazi quanti sono quelli specificati dal numero.

Esempio:

SPACE$ (30) ritorna una stringa di 30 spazi

SPC(numero)
funzione

Questa funzione può essere usata solo in un'istruzione PRINT.

Esempio:

```
PRINT "A" SPC(10) "B"
A          B
```

SQR(numero)
funzione

Ritorna la radice quadrata di un numero.

Nel Basic 128, il risultato è dato in doppia precisione se l'argomento è a doppia precisione.
Se l'argomento è negativo, appare il messaggio "Illegal Function Call" (richiamo funzione non valida).

Esempio:

```
PRINT SQR(10)
3.16228
```

STEP (spostamento verticale,spostamento orizzontale) *

STEP può essere usato nelle seguenti istruzioni: BOX, BOXF, CIRCLE, CIRCLEF, LINE, PAINT e PSET.
STEP permette di definire le coordinate di un punto relativamente alla posizione dell'ultimo punto definito (il punto di default è l'origine (0,0)). Gli spostamenti possono essere positivi o negativi.

Esempio:

BOX(100,100)–STEP(-50,50):BOX(200,150)–STEP(50,50)

equivale a

BOX(100,100)–STEP(–50,50):BOX STEP(50,50)–STEP(50,50)

STICK(n. joystick)
funzione

Ritoma la posizione del joystick specificato.

I due joystick sono numerati 0 e 1.
Il risultato è:

0 neutro
1. 90°
2. 45°
3. 0°
4. –45°
5. –90°
6. –135°
7. 180°
8. 135°

Esempio:

ON STICK(0) GOSUB... permette il salto al sottoprogramma che corrisponde alla posizione del joystick.

STOP
Istruzione

Sospende l'esecuzione del programma.

Questa istruzione può essere inserita in un punto qualsiasi del programma.
Quando la si incontra, appare il messaggio "Break in...". E' possibile riprendere l'esecuzione del programma con CONT o con GOTO.
Tutti i file restano aperti.

Esempio:

```
10 INPUT A,B
20 C=2A^2+B^2
30 STOP
40 PRINT C
```

STR$(numero)
funzione

Trasforma un numero nella sua rappresentazione stringa.

Il risultato ha lo stesso formato della visualizzazione del numero tramite PRINT.
La funzione inversa di STR$ è VAL.

Esempio:

```
PRINT STR$(68.5):STR$(-1234)
68.5 -1234
```

STRIG(n. joystick)
funzione

Ritorna lo stato del pulsante del joystick specificato.

I due joystick sono numerati 0 e 1.
Il risultato è **-1** se il pulsante è premuto, 0 se è rilasciato.

Esempio:

```
IF STRIG(1) THEN PRINT "TASTO"
```

STRING$(numero,carattere)
funzione

Ritorna una stringa di caratteri identica al carattere indicato e la cui lunghezza è uguale al numero indicato.

E' possibile specificare il carattere con il suo codice ASCII.

Esempio:

```
STRING$(10,"^")     dà una stringa di 10 asterischi
STRING$(20,42)      dà una stringa di 20 asterischi
```

SWAP variabile 1,variabile 2
istruzione

Scambia il contenuto di due variabili dello stesso tipo.

Questa istruzione è molto utile nei programmi di ordinamento (SORT).

Esempio:

SWAP A(I),A(J) scambia i valori degli elementi I e J della matrice A.

TAB(numero)
funzione

Posiziona il cursore all'interno di un'istruzione PRINT o PRINT#.

Il numero indicato, compatibilmente alla largezza della periferica utilizzata (schermo o stampante), indica la posizione del cursore in colonne.
Se questa posizione è a sinistra della posizione corrente, viene effettuato un avanzamento riga.
Un numero negativo o nullo non provoca alcun effetto.
In doppia larghezza, il posizionamento del cursore tiene conto della doppia larghezza.
Vengono rispettate le regole di posizionamento stabilite dai separatori (virgola e punto e virgola).

Esempio:

```
PRINT "A";TAB(10);"B"
A         B
```

TAN(numero)
funzione

Ritoma la tangente di un numero espresso in radianti.

Nel Basic 128, il risultato è dato in doppia precisione se l'argomento è a doppia precisione.

Esempio:

```
?TAN(1.5)
14.1014
```

TRACE *
funzione

Indica se la tartaruga attiva lascia una traccia dei suoi spostamenti (risultato –1) oppure se non lascia una traccia (risultato 0).

Esempio:

?TRACE
–1

TRACE numero *
Istruzione

Attiva o disattiva la traccia lasciata dalla tartaruga attiva.

Se l'argomento è 0 (FALSO), la tartaruga non lascia traccia, se è 1 (VERO) la tartaruga lascia traccia dei suoi spostamenti.

Esempio:

TRACE 1 ta tartaruga lascerà una traccia di tutti i suoi spostamenti

TROFF
Istruzione

Annulla la modalità di traccia inserita con l'istruzione TRON e chiude il file di traccia, se esistente.

TRON descrittore file,*n. riga 1 – n. riga 2*
Istruzione

Attiva la modalità di traccia det programma.

La modalità di traccia determina la visualizzazione su video o la registrazione nel file indicato dei numeri di riga eseguiti. Nel Basic 1 non bisogna indicare il descrittore di file in quanto la visualizzazione avviene sempre su video.
I numeri di riga appaiono tra parentesi quadre.
Se viene indicato il descrittore del fite, il tracciamento viene scritto sul fite. In caso contrario il tracciamento verrà visualizzato sullo schermo.
Vengono indicate soltanto le righe di programma eseguite il cui numero è compreso tra i due numeri di riga specificati. I numeri di riga vengono indicati seguendo la sintassi di LIST.

Esempio:

TRON "LPRT:",200-400 traccia su stampante la parte di programma compresa tra le righe 200 e 400.
TRON traccia tutto il programma su video.

TUNE ◆
istruzione

Permette la regolazione della penna ottica. Quanto si punta la penna ottica sul video, appare una barra verticale che bisogna far coincidere con la penna servendosi dei tasti ← e → . La regolazione viene confermata premendo il tasto ENTER.

TURTLE *n.,colonna,riga,stringa di caratteri* *
istruzione

Definisce e posiziona la tartaruga attiva.

Il primo parametro indica il numero della tartaruga attiva. E' possibile definire fino a dieci tartarughe diverse, numerate da 0 a 9.
I due parametri seguenti stabiliscono la posizione della tartaruga sul video. La colonna e la riga sono comprese tra –32768 e 32767. Questi due parametri non sono obbligatori. Quando non sono presenti, non viene modificata la posizione della tartaruga oppure, se la tartaruga è stata attivata per la prima volta, viene posizionata al centro dello schermo.
Il quarto parametro, anch'esso facoltativo, è una stringa di caratteri che permette di definire la forma della tartaruga. In questa stringa sono presenti delle coppie di valori che indicano ogni rotazione durante uno spostamento.

Esistono quattro tipi di coppie di valori possibili: RaDn, RaUn, LaDn e LaUn, dove a indica l'angolo di rotazione e n il numero che indica l'entità dello spostamento.

Gli angoli e gli spostamenti devono essere espressi con dei numeri compresi nella gamma da 0 a 255. Un angolo di 256 è uguale e a 360 gradi.
R significa rotazione a destra e L rotazione a sinistra.
D significa spostamento lasciando traccia e U spostamento senza traccia.
Non vengono contati gli spazi.

Esempio:

TURTLE 1,80,120,"L0D40L64D40" definisce la tartaruga n. 1 a forma di angolo retto e la posiziona in colonna 80, riga 120
A$="L0D40L128U20L64D20":TURTLE 2,,,A$ definisce la tartaruga n. 2 a forma di T e la posiziona al centro dello schermo.
TURTLE 0 attiva la tartaruga n. 0

UNLOAD *n. drive*
istruzione

Chiude tutti i file aperti sul dischetto posto nel drive di cui è stato precisato il numero e ricopia tutte le informazioni utili sul dischetto.

UNLOAD considera come drive di default il drive n. 0 oppure quello definito con DEVICE.
Questa istruzione consente di togliere senza rischi il dischetto dal drive dopo l'arresto di un programma di scrittura file causato da CTRL–C o da un errore.

Esempio:

UNLOAD 1 chiude tutti i file sul dischetto 1

USRn.(dati)
funzione

Richiama la funzione utente in linguaggio macchina in base al numero indicato (numero di default: 0).

La funzione deve essere stata definita precedentemente con DEFUSRn.
Per usare questa funzione, vedere l'Appendice 4.

Esempio:

A=USR2(F) richiama la funzione utente n. 2 con la variabile F come argomento.

VAL (stringa)
funzione

Ritoma il valore numerico di una stringa di caratteri.

Se il primo carattere non è un numero decimale, oppure +, –, un punto oppure uno spazio, il risultato della funzione è nullo.
La funzione inversa di VAL è STR$.

Esempio:

```
PRINT VAL("15 scatole di dischetti")
15
```

VARPTR(variabile)
funzione

Ritoma l'indirizzo del primo byte della variabile indicata.

Il risultato è un numero intero compreso tra –32768 e 32767.
Occorre aggiungere 65536 per avere l'indirizzo reale quando questo è negativo.
Nel Basic 128, dopo l'esecuzione di questa funzione, il banco corrente diventa quello in cui si trova la variabile ricercata. Si può ottenere questo numero di banco con l'istruzione BANK. Questa funzione viene usata molto spesso per i moduli in linguaggio macchina.
Per informazioni sull'organizzazione delle variabili in memoria vedere l'Appendice 3.

Esempio:

PRINT VARPTR(A) dà l'indirizzo della variabile A

VERIFY ON
VERIFY OFF
Istruzioni

Con l'opzione ON, verranno controllate tutte le scritture su dischetto. L'opzione OFF disattiva questo controllo.

Esempio:

VERIFY ON

WAIT Indirizzo,maschera 1,maschera 2 *
istruzione

Sospende l'esecuzione di un programma fino a quando non appare una configurazione di bit all'indirizzo dato.

L'indirizzo è un numero compreso tra –65536 e 65535. Le due maschere sono dei numeri compresi tra 0 e 255. L'istruzione funziona nel modo seguente:
– viene calcolata l'espressione booleana seguente: (contenuto dell'indirizzo XOR maschera 2) AND maschera 1
– l'esecuzione del programma non prosegue fino a quando questa espressione non è nulla.
La maschera 2 è facoltativa. In questo caso non viene applicato l'operatore XOR.
Questa istruzione viene usata molto spesso per aspettare che accada un certo evento su un circuito periferico. E' possibile uscire dalla fase di attesa con CTRL–C.
Per usare questa istruzione è necessario avere una buona conoscenza della struttura interna del microcomputer.

WINDOW (colonna 1, riga 1)–(colonna 2, riga 2) *
istruzione

Definisce la finestra di visualizzazione delle istruzioni grafiche.

La finestra è un rettangolo definito da due vertici opposti, il primo in alto a sinistra e il secondo in basso a destra. Le coordinate relative alle colonne sono dei numeri compresi tra 0 e 319 (o 639) e le coordinate relative alle righe sono comprese tra 0 e 199.
Dopo l'esecuzione di questa istruzione, le istruzioni di tracciamento grafico BOX, BOXF, LINE, PSET, CIRCLE, CIRCLEF e PAINT saranno visibili solo all'interno di questa finestra.

Esempio:

```
WINDOW (100,100)–(200,180)
```

WRITE# n. canale,*elenco dati*
istruzione

Scrive l'elenco dei dati nel file di cui è stato specificato il numero di canale.

Le stringhe di caratteri sono scritte tra virgolette e i vari dati sono separati con delle virgole. Dopo l'ultimo dato, WRITE# scrive due caratteri: avanzamento riga e ritorno carrello.

Nel caso di file ad accesso diretto, WRITE# scrive nel buffer di registrazione. Il trasferimento della registrazione nel file avviene con PUT#.

Esempio:

```
WRITE #2,NOME$,TASSE
```

ZOOM *
funzione

Ritorna la dimensione della tartaruga attiva.

Il risultato è un numero compreso tra 0 e 255.
La dimensione normale di una tartaruga al momento della definizione con TURTLE è 16.

Esempio:

```
?ZOOM
32
```

ZOOM *TO* **numero** *
istruzione

Fissa o modifica la dimensione della tartaruga attiva.

Se TO è presente, la dimensione è fissata in modo assoluto in base al valore indicato, in caso contrario la dimensione della tartaruga viene incrementata di questo valore.

La dimensione di una tartaruga può variare tra 0 e 255, ma la lughezza di un segmento della tartaruga non deve superare il valore 255.
Seguendo il valore del secondo parametro di SHOW, la dimensione viene modificata immediatamente (tartaruga libera) oppure solo dopo aver eseguito FWD o TURTLE.

Esempio:

TURTLE 3:ZOOM TO 32 fissa la dimensione della tartaruga 3 a 32
ZOOM t0 aumenta di 10 la dimensione della tartaruga attiva.

Appendici

Appendice 1

Organizzazione di un dischetto

Questa appendice è destinata a coloro che vogliono saperne un po' di più sul modo in cui il BASIC 128 gestisce un dischetto. Le informazioni che vengono date in questa appendice sono un po' più tecniche ma non è necessario conoscerle per un uso appropriato dei file.

Organizzazione fisica

Un dischetto è costituito da tracce concentriche contenenti ognuna 16 settori. Ogni settore contiene 128 byte utili se un dischetto è a singola densità o 255 byte utili se il dischetto è a doppia densità. Le tracce sono numerate a partire da 0, mentre i settori a partire da 1.

La divisione delle tracce in settori avviene al momento dell'inizializzazione del dischetto da parte del BASIC. Si dice allora che la settorizzazione è logica. Il dischetto, o meglio il supporto di mylar, è dotato di un unico foro d'indice sulla circonferenza. Questo foro unico serve a reperire la posizione del primo settore di ogni traccia.

	QDD	SD 5"1/4	DSDD 5"1/4	DD 3"1/2
N. totale tracce	25	40	2 x 40	80
Numerazione tracce	0-24	0-39	0-39	0-79
N. totale settori	400	640	2 x 640	1280
Dimensione settore (in byte)	128	128	255	255
Capacità in Kbyte	50	80	2 x 160	320

Organizzazione fisica dei dischetti PC 128

QDD: Quick Disk Drive
SD: Singola densità
DD: Doppia densità
DS: Doppia faccia

Indice dei file

Per motivi di comodità di accesso, l'indice dei file si trova sulla traccia 20 del dischetto. In questo modo diminuisce la distanza tra i dati di un file e le informazioni generali che lo riguardano e che sono contenute nell'indice.
L'indice inizia nel settore 3 e occupa 14 settori. Ogni file che fa parte dell'indice ha un'area riservata fissa di 32 byte. Quest'area contiene le informazioni seguenti:

Numero di byte	Contenuto
8	Nome del file, allineato a sinistra, completato con spazi a destra se necessario. Se il file viene cancellato con KILL, il primo byte contiene 0. Se l'area riservata all'indice non è ancora stata utilizzata, il primo byte contiene FF (esadecimale).
3	Estensione del nome del file (BAS,DAT,BIN,...), allineato a sinistra, completalo con spazi a destra se necessario.
1	Tipo del file: 0: programma BASIC 1: file dati BASIC 2: programma in linguaggio macchina 3: file testo
1	Tipo di dati: 0: i byte contengono dei numeri binari FF: i byte contengono dei caratteri in codice ASCII.
1	Numero del primo blocco attribuito al file (i blocchi vengono numerati a partire da 0).
2	Numero di byte occupati nell'ultimo settore del file.
8	Commento associato al file (in caso di SAVE, COPY, NAME).
8	Inutilizzati.

Tutte le informazioni scritte nell'indice dei file vengono gestite interamente dal BASIC. In particolare, il numero di byte occupati nell'ultimo settore può essere scritto solo al momento della chiusura. Un buon motivo per non dimenticare CLOSE.

Tabella di allocazione della memoria del dischetto

Abbiamo visto con l'istruzione DIR che la memoria riservata ad ogni file non si conta in settori ma in blocchi di otto settori, cioè 1 Kbyte per i dischetti a singola densità e 2 Kbyte per i dischetti a doppia densità.
Il settore numero 2 della traccia 20 contiene la tabella di allocazione della memoria, cioè le informazioni sui blocchi del dischetto.
In questa tabella, ogni blocco è rappresentato da un byte. Attenzione: il primo byte del settore non viene utilizzato. Il primo blocco viene rappresentato dal secondo byte, il secondo blocco dal terzo byte, ecc.
Il valore del byte dà le indicazioni sull'occupazione del blocco corrispondente (1):

Valore (esadscimale)	Significato
tra 0 e 4B	Il blocco fa parte di un file, il byte contiene il numero del prossimo blocco dello stesso file.
tra C1 e C8	Il blocco è l'ultimo di un file, il byte contiene il numero di settori del blocco occupati dal file a cui viene aggiunta la costante C0.
FE	Il blocco è riservato e non può essere usato per un file.
FF	Il blocco è libero.

Questa tabella di allocazione viene aggiornata man mano che si scrive sul file, ma non viene aggiornata ogni volta che si scrive in un file. Questo è il motivo per cui è fondamentale utilizzare UNLOAD quando si interrompe bruscamente un'operazione di scrittura senza che il file sia stato chiuso. UNLOAD ricopia questa tabella sul dischetto ed evita dei problemi se viene sostituita da un'altra nel drive.

Come vedere il contenuto del dischetto DSKI$ e DSKO$

Se si vuole visualizzare il contenuto del dischetto settore per settore, si può usare una funzione che è simile a PEEK (), che viene usata per vedere il contenuto della memoria centrale.
DSKIS(0.10.3)

visualizza il contenuto (una stringa da 128 o 255 caratteri) del settore 3 della traccia 10 del drive 0. Con questa funzione è possibile esaminare come sono memorizzati i dati in un file ad accesso diretto.
Questa funzione viene completata dalla funzione DSKO$ che permette di scrivere nel settore scelto. Quindi per scrivere la stringa di caratteri SCT$, con una lunghezza massima di 128 o 255 byte, nel settore 3 della traccia 10 del drive 0 bisogna battere:
DSKOS0,10,3,SCT$

Utilizzando la funzione DSKI$() oppure l'istruzione DSKO$ non bisogna dimenticare che le tracce vengono contate da 0 a 39 o da 0 a 79 e i settori da 1 a 16. Si consiglia inoltre di non usare la funzione DSKO$ su un dischetto che contiene dei programmi o dei dati preziosi. Un comando sbagliato potrebbe cancellarti. Esercitarsi prima su delle copie per evitare rischi.

Esscuzione automatica di un programma

Se si usa spesso lo stesso programma, è possibile lanciare immediatamente la sua esecuzione dopo l'avviamento del sistema. Per farlo bisogna salvare il programma in questione con il nome "AUTO.BAT".
Al momento dell'avviamento, quando si sceglie l'opzione 1 del menu, l'interprete BASIC va a cercare sul dischetto il programma AUTO.BAT, lo carica e lo esegue.

Protezione dischetto

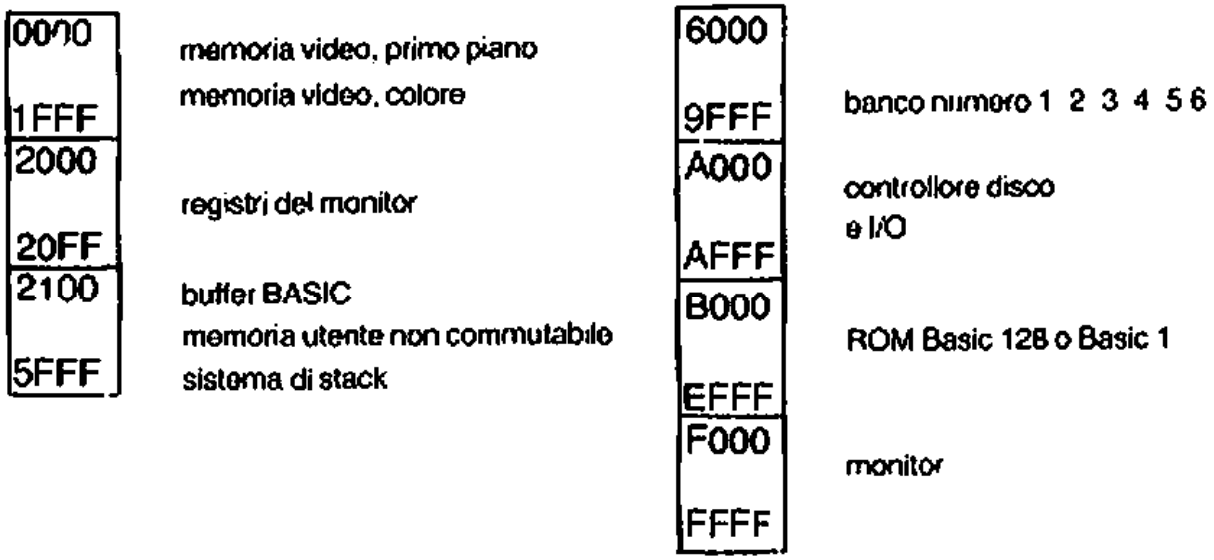
Il terzo parametro di FILES permette di riservare un'area tampone di memoria al fine di ottimizzare l'accesso fisico al dischetto o al QDD in lettura e in scrittura.
Il principio è quello di precaricare in memoria al momento della lettura una traccia intera invece di un solo settore in modo che risulti immediato l'accesso al settore seguente. Al momento della scrittura, la protezione disco memorizza i settori da scrivere e li trasferisce in pacchetti sul dischetto. Comunque per essere sicuri che sono stati trasferiti tutti i file è necessario eseguire un'istruzione CLOSE.
L'istruzione DSKO$ non passa per la protezione.
Il parametro che deve essere introdotto è il numero delle tracce riservate nella memoria RAM. Una traccia rappresenta 2K sui dischetti a singola densità e 4K sui dischetti a doppia densità. La dimensione massima della protezione è 25 tracce. I valori di default sono 0 tracce per le unità a disco e 3 tracce per i QDD.

APPENDICE 2

Organizzazione della memoria utente

Suddivisione della memoria

Lo spazio di memoria è organizzato nel modo seguente:



La zona compresa tra 2000 e 20FF corrisponde alla pagina zero del monitor e contiene tutti i registri di lavoro.
La zona compresa tra 2100 e 5FFF, non commutabile, viene usata dall'interprete BASIC per le zone tampone, per il trattamento dei dati e tutti i puntatori sulle variabili e per il programma. La zona dei banchi di memoria A000 e DFFF contiene il programma, le variabili e i dati in stringhe di caratteri (96 Kbyte disponibili).

Conoscere lo spazio di memoria disponibile: FRE()

La funzione FRE permette di conoscere i diversi parametri dello spazio di memoria libero:

FRE(2)	dà il volume in byte dello spazio libero nei banchi di memoria a disposizione del programma e dei dati.
FRE(1)	dà il volume dello spazio libero nella memoria fuori banchi a disposizione per i buffer e i puntatori.
FRE(0)	dà il volume dello spazio totale disponibile: FRE(0)=FRE(1)+FRE(2)
FRE(X$)	dà il volume in byte dello spazio libero per le stringhe di caratteri.

Esame e modifica dei byte in memoria: PEEK e POKE

E' possibile conoscere il valore di un byte qualunque della memoria con:

PEEK(indirizzo)

PEEK(&H4000) dà il contenuto del byte situato all'indirizzo 4000 (esadecimale).
La modifica di un byte si ottiene con:

POKE indirizzo,valore

POKE&H4000,128 assegna il valore decimale 128 (80 esadecimale) al byte con indirizzo 4000 (esadecimale).
Per scegliere il banco di memoria da esaminare, occorre usare l'istruzione BANK.

BANK 2 orienta sul banco 2 tutte le istruzioni che agiscono direttamente sulla memoria.
L'esecuzione di PEEK (&H7000) dà il contenuto del byte con indirizzo 7000 (esadecimale) del banco numero 2.
Al momento dell'inizializzazione del sistema, viene selezionato il banco di ordine maggiore, cioè il banco numero 6.

Per scegliere il banco d'ordine maggiore, senza doversi preoccupare del numero di banchi presenti, battere:

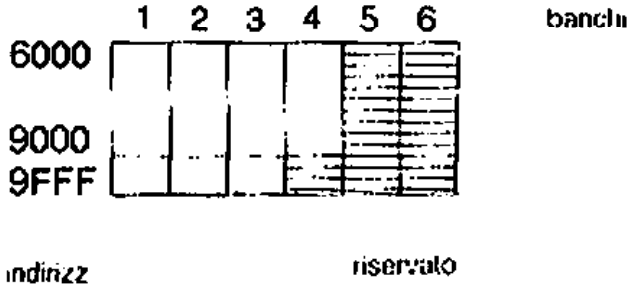
BANK 0

Il risultato è lo stesso ottenuto dopo l'inizializzazione. Il banco di memoria selezionato da BANK viene quindi designato implicitamente per le istruzioni e funzioni seguenti: PEEK, POKE, LOADM, SAVEM, EXEC, DEF USR, CLEAR.

Come riservare areee di memoria: CLEAR

Tutte le modifiche dello spazio di memoria disponibile vengono effettuate con l'istruzione CLEAR.

1. *area stringa di caratteri*
La modifica più frequente riguarda lo spazio disponibile per la memorizzazione delle stringhe di caratteri. Il volume di questo spazio è fissato a 300 caratteri al momento dell'inizializzazione del sistema e può essere modificato con il primo argomento di CLEAR. CLEAR 2000 fissa a 2000 caratteri lo spazio di memoria disponibile.
2. *area riservata nei banchi di memoria*
L'area riservata nella memoria viene usata per memorizzare i dati sotto forma di numeri binari o i moduli in linguaggio macchina. Quest'area riservata fissa permette di utilizzarli partendo da un programma in BASIC.
L'indirizzo dell'area riservata meno 1 viene fissato dal secondo parametro dell'istruzione CLEAR. CLEAR,&H8FFF riserva lo spazio di memoria a partire dall'indirizzo &H9000 fino all'indirizzo &H9FFF nel banco di memoria corrente e riserva interamente i banchi di ordine maggiore.



l'area tratteggiata viene riservata con l'istruzione:

BANK 4:CLEAR,&H8FFF

3. *area riservata fuori dalla zona commutabile*
Viena fissata dal quarto argomento dell'istruzione CLEAR, che influenza lo spazio compreso tra &H2100 e &H5FFF. CLEAR,,,&H4FFF riserva l'area compresa tra &H5000 e &H5FFF.

4. *area riservata ai caratteri utente*
Viene fissata dal terzo parametro dell'istruzione CLEAR. E' sufficiente indicare il numero di caratteri utente massimi usati. Ogni carattere occupa otto byte. CLEAR,,10 fissa a 10 il numero di caratteri utente e riserva lo spazio necessario nella zona dei banchi di memoria.

Una volta che sono state riservate le aree di memoria con l'istruzione CLEAR, non vengono modificate persino in fase di caricamento e di esecuzione di un nuovo programma. E' possibile riservare altre aree di memoria solo usando l'istruzione CLEAR con gli argomenti adatti.

Salvataggio e caricamento di un'aree di memoria: SAVEM, LOADM

Queste due istruzioni vengono usate per conservare in un file un'area precisa della memoria che contiene dei dati o un modulo in binario.
Per salvare l'area di memoria usare il seguente formato: SAVEM descr. del file,ind.,inizio,ind.,fine,ind.dell'esecuzione. Il quarto argomento, e cioè l'indirizzo dell'esecuzione, viene usato solamente per un programma scritto in binario, ma è comunque obbligatorio.
Per caricare un'area di memoria utilizzare il seguente formato: LOADM descr. del file,spostamento,R.
Il secondo argomento, e cioè lo spostamento, indica, quando esiste, di quanti byte bisogna spostare la memoria rispetto al suo indirizzo al momento in cui il file era stato salvato. L'opzione R indica che bisogna eseguire il modulo all'indirizzo di esecuzione salvato in SAVEM, tenendo conto dell'eventuale spostamento.

APPENDICE 3

Rappresentazione delle variabili in memoria

La funzione VARPTR dà l'indirizzo esatto della variabile indicata nell'argomento della funzione e permette quindi di conoscere e di modificare il suo contenuto.
Si posiziona automaticamente sul banco di memoria che contiene la variabile in questione.
Viene usata molto spesso per interfacciare un programma scritto in BASIC con un modulo in linguaggio macchina.

Variabili intere

Queste variabili vengono rappresentate in binario con due byte in complemento a due.
Il primo byte contiene i bit più significativi e il secondo i bit meno significativi.
Il bit più significativo del primo byte rappresenta il segno:

0 numero positivo o zero
1 numero negativo

Gli altri quindici bit permettono quindi di rappresentare dei valori assoluti fino a 2^ 15 cioè 32.768.
I numeri interi sono compresi quindi tra –32768 e 32767. La funzione VARPTR dà l'indirizzo del primo byte.

Esempio:

```
X%=12
?PEEK(VARPTR(X%)).PEEK(VARPTR(X%)+1)
0       12
```

Variabili reali o a precisione singola

Queste variabili vengono rappresentate in virgola mobile (mantissa+esponente) su 4 byte.

I primo byte contiene l'esponente.
L'esponente viene espresso in binario segnato e sottratto di &H60.
Cosi &H81 corrisponde a 2^1
e &H7E corrisponde a 2 –2^
Se l'esponente è uguale a zero, il numero è zero.

Il secondo byte contiene il bit più significativo della mantissa.
Il terzo byte contiene il bit medio della mantissa. Il quarto byte contiene il bit meno significativo della mantissa. Il bit più significativo della mantissa rappresenta il segno del numero. Gli altri bit rappresentano la mantissa in binario, ad eccezione del

primo bit della mantissa che è sempre uguale e 1, e che quindi viene omesso. Il valore della mantissa può quindi essere scritto, in base 2:

0,1 mantissa

Di conseguenza il numero minore positivo vale 2.9874 E–39 e il maggiore vale 1.7014 E+38. La funzione VARPTR dà l'indirizzo del primo byte, quindi l'esponente.

Esempi:

```
NB=3
ADNB=VARPTR(NB)
FOR I=0 TO 3:PRINT PEEK(ADNB+I);:NEXT
130 64 0 0
```

cioè un esponente uguale a $2^{(130-128)}=2^2$ e una mantissa uguale a 0.11000000 00000000 00000000

```
NB=-1
ADNB=VARPTR(NB)
FOR I=0 TO 3:PRINT PEEK(ADNB+I);:NEXT
129 128 0 0
```

cioè un esponente uguale a $2^{(129-128)}=2^1$ e una mantissa uguale a 0.10000000 00000000 00000000 e un segno negativo.

Variabili a doppia precisione

Queste variabili vengono rappresentate in virgola mobile nello stesso modo in cui vengono rappresentate le variabili a precisione singola ma su 8 byte. Il primo byte contiene l'esponente, che viene codificato come per le variabili a precisione semplice, e gli altri 7 byte contengono la mantissa.
I valori minimi e massimi sono gli stessi delle variabili a precisione singola.

Variabili stringa

Queste variabili sono costituite da due parti, un "descrittore" su tre byte e il contenuto della stringa vera e propria.
Il primo byte del "descrittore" contiene il valore che definisce la lunghezza della stringa.
I due byte che seguono contengono l'indirizzo logico della stringa nell'area stringa di caratteri. L'indirizzo dell'area stringa è contenuto negli indirizzi &H2122 e &H2123. Il numero del banco di memoria che contiene l'area stringa è indicato in &H2124.
L'indirizzo fisico della stringa si ottiene aggiungendo all'indirizzo dell'area stringa l'indirizzo logico della stringa. Se l'indirizzo fisico supera &HA000, occorre sottrarre &HA000 e aggiungervi 6000 per ottenere il valore desiderato, questo perchè bisogna tenere in considerazione un accavallamento dell'area stringa tra due banchi di memoria.
La funzione VARPTR dà l'indirizzo del primo byte del descrittore.

Esempio:

```
5' CASO SENZA ACCAVALLAMENTO
10 CHS="ABCDEF"
20 ADCH=VARPTR(CHS)
30 LCH=PEEK(ADCH) ADLOG=256*PEEK(ADCH+1)+PEEK(ADCH+2)
40 PRINT LCH,ADLOG
50 ADZCH=256*PEEK(&H2122)+PEEK(&H2123)
60 BCH=PEEK(&H2124)
70 PRINT HEX$(ADZCH),BCH
80 BANK BCH
90 PRINT CHRS(PEEK(ADZCH+ADLOG))
RUN
6   0
9ED3   6
```

APPENDICE 4

Moduli e funzioni in linguaggio macchina

Questa appendice interessa soprattutto coloro che hanno qualche nozione sulla programmazione del microprocessore 6809 in linguaggio macchina e presuppone una conoscenza di base di questo microprocessore.
Esistono due soluzioni per utilizzare il linguaggio macchina: i moduli eseguibili da EXEC e le funzioni utente USR.

L'istruzione EXEC e i suoi parametri

La prima cosa da fare è riservare lo spazio necessario alle istruzioni in linguaggio macchina usando l'istruzione CLEAR che riserva l'area di memoria utile:

CLEAR,&H9EFF

imposta a &H9EFF l'indirizzo più alto del BASIC nel banco corrente o per default nell'ultimo. Restano quindi liberi 256 byte fino alla fine del banco di memoria in &H9FFF.

A questo punto bisogna memorizzare con delle istruzioni POKE, oppure con LOADM se queste istruzioni sono state memorizzate in un file, le istruzioni che sono pronte per essere eseguite.

Infine si esegue il modulo in linguaggio macchina con:

EXEC &H9F00

se l'indirizzo dell'inizio del modulo è in &H9F00.

Tuttavia, prima di scrivere il modulo in linguaggio macchina, bisogna rispettare alcune regole se poi si vuole ritornare al programma BASIC.
– il modulo deve terminare per RTS,
– i registri S (Stack) e DP (Direct Page) non devono essere modificati. Per poterli utilizzare nel modulo, bisognerà salvarli in fase di input e recuperarli in fase di output.

E' possibile attribuire dei parametri al modulo in linguaggio macchina. Questi parametri devono seguire l'indirizzo ed essere separati con delle virgole.
Il recupero dei parametri nel modulo avviene richiamando l'interprete BASIC.
I punti di entrata utili per poter analizzare i parametri sono i seguenti:

EFE6 RESBAN	routine da richiamare prima di una delle sei routine di lettura dei parametri, in modo da commutare il BASIC verso la RAM del programma. Deve essere richiamata anche dopo PTRGET per continuare l'analisi dei parametri in quanto PTRGET commuta il banco che contiene la variabile
EFE9 LITINT	legge un numero intero con segno, risultato nel registro X
EFEC LITADR	legge un numero intero senza segno (indirizzo), risultato nel registro X
EFEF LITSGN	legge un numero reale a precisione singola, il registro X punta il risultato
EFF2 LITSTR	legge una stringa, in uscita il registro B contiene la lunghezza, il registro X punta il risultato
EFF5 FRMEVL	legge un dato qualsiasi, in uscita si ottiene lo stesso risultato della funzione USR (vedere funzione USR)
EFF8 PTRGET	dà l'indirizzo del parametro, in uscita il registro A contiene il numero di banco, il registro X punta il contenuto, il byte VALTYP($6105) contiene il tipo del risultato (2, 3, 4 o 8)
21C6 IFEND	verifica la fine della riga, in uscita il FLAG Z del registro di stato vale 1 se è stata raggiunta la fine.

Il programma di recupero dei parametri deve essere nella RAM fissa. Se EXEC lancia delle routine di rilevamento interrupt, la routine di rilevamento interrupt deve essere nella RAM fissa.

Le funzioni USRn

Si possono definire dieci funzioni utente in linguaggio macchina con l'istruzione:

DEF USRn=indirizzo
(n è un numero (da 0 a 9),
indirizzo designa l'indirizzo di memoria in cui comincia la funzione).

Questa funzione viene richiamata come una funzione normale: variabile=USRn (espressione).

Il vantaggio principale delle funzioni USR è quello di permettere di passare valori e di ritornare al programma BASIC con il risultato ottenuto.
Vediamo ora come questi valori vengono passati alle funzioni utente.
Quando viene richiamata la funzione, l'accumulatore A contiene il tipo del dato:
2: numero intero
3: stringa di caratteri
4: numero a precisione singola
8: numero a doppia precisione

il registro di indice X punta il valore:
2,X: numero intero
1,X: numero a singola o doppia precisione
0,X: descrittore di stringa

Nel caso in cui il dato è una stringa di caratteri, l'accumulatore B contiene la lunghezza della stringa, il registro U punta il primo carattere della stringa.
Bisogna notare che, salvo per le stringhe di caratteri, l'accumulatore A contiene anche la lunghezza del dato in byte. Ricordiamo che:
– i numeri interi vengono codificati su due byte: il primo byte più significativo, il secondo byte meno significativo,
– i numeri a precisione singola vengono codificati su quattro byte: il primo contiene l'esponente, i successivi la mantissa, con davanti il byte più significativo,
– i numeri a doppia precisione vengono codificati su otto byte: il primo contiene l'esponente, i successivi la mantissa,
– il descrittore di una stringa di caratteri è composto da 3 byte: il primo indica la lunghezza della stringa, gli altri due l'indirizzo del primo carattere.
E' possibile stabilire la lunghezza della stringa con l'accumulatore B o con il descrittore.
Il valore (numero o descrittore stringa) viene sempre memorizzato nello stesso punto (FAC).

All'uscita di un modulo Assembler, occorre rendere compatibili i contenuti dell'accumulatore A e del registro X con il tipo della variabile che riceverà il vatore di output.
In uscita, X deve quindi puntare sullo stesso indirizzo che aveva in fase di entrata.
In tutti i casi, i registri S e DP devono contenere gli stessi valori che avevano in fase di input.

Esempio: aggiungere 256 a un numero intero.
Dopo aver riservato lo spazio (sufficientemente grande) nella riga 20, si leggono le varie istruzioni in linguaggio macchina che vengono introdotte con DATA e si memorizzano (righe da 40 a 70).

Per poter introdurre nell'istruzione DATA un numero variabile di istruzioni, l'ultimo valore, fittizio, è maggiore di 255(&H100) e serve al test di fine lettura.

Il moduto stesso comporta soltanto due istruzioni in linguaggio macchina (non è possibile semplificarle ulteriormente):

```
6C02INC2,X;incrementare in (X)+2
39 RTS;ritorno
```

L'incremento del byte più significativo consiste nell'aggiungere 256. Notare che per i numeri interi il valore non è in (X) ma in (X)+2.

La nuova variabile R% riceve allora il nuovo valore che viene visualizzato con il precedente.

```
10'AGGIUNGERE 256
20 CLEAR,&H9EFF
30 DEFUSR1=&H9F00
40 ADR=&H9F00
50 READ OCT:IF OCT>255 THEN 80
60 POKE ADR,OCT
70 ADR=ADR+1:GOTO 50
80'UTILIZZO
90 N%=10
100 R%=USR(N%)
110 PRINT N%,R%
120 DATA&H6C,&H02,&H39,&H100
```

APPENDICE 5

Elenco delle parole riservate

ABS
AND
ASC
ATN
ATTRB
AUTO

BACKUP
BANK
BEEP
BOX

CDBL
CHAIN
CHR$
CINT
CIRCLE
CLEAR
CLOSE
CLS
COLOR
COMMON
CONSOLE
CONT
COPY
COS
CRUNCH$
CSNG
CSRLIN
CVD
CVI
CVS

DATA
DEF
DEFDBL
DEFINT
DEFSNG
DEFSTR
DELETE
DENSITY
DEVICE
DIM
DIR
DO
DOS
DSKI$
DSKF
DSKO$

ELSE
END
EOF
EQV
ERL
ERR
ERROR
EVAL
EXEC
EXIT
EXP
FIELD
FILES
FIX
FKEY
FN
FOR
FRE

GET
GO
GR$

HEAD
HEX$

IF
IMP
INKEY$
INMOUSE
INPEN
INPUT
INSTR
INT
INTERVAL

KEY
KILL

LEFT$

LEN
LET
LINE
LIST
LOAD
LOC
LOCATE
LOF
LOG
LOOP
LSET

MAX
MERGE
MKD$
MKI$
MKS$
MID$
MIN
MOD
MOTOR
MOUSE
MTRIG

NAME
NEW
NEXT
NOT

OCT$
OFF
ON
OPEN
OR

PAINT
PALETTE
PATTERN
PEEK
PEN
PLAY
POINT
POKE
POS
PRINT
PSET
PTRIG
PUT

READ
REM
RENUM
RESET
RESTORE
RESUME
RETURN
RIGHT$
RND
ROT
RSET
RUN

SAVE
SCREEN
SEARCH
SGN
SHOW
SKIPF
SPACE$
SPC(
SQR
STEP
STICK
STOP
STRIG
STRING$
STR$
SUB
SWAP

TAB(
TAN
THEN
TO
TRACE
TROFF
TRON
TUNE
TURTLE

UNLOAD
UNMASK
USING
USR

VAL
VARPTR

WAIT
WINDOW
WRITE

XOR

ZOOM

APPENDICE 6

Messaggi e codici di errore

1 Next Without For
NEXT senza FOR. E' stata incontrata un'istruzione NEXT prima dell'istruzione FOR corrispondente.

2 Syntax Error
Errore di sintassi. L'istruzione non viene riconosciuta oppure è scritta in modo errato.

3 Return Without GOSUB
RETURN senza GOSUB. E' stata incontrata l'istruzione RETURN prima del GOSUB relativo.

4 Out Of Data
Dati esauriti. Non ci sono più dati nelle righe DATA per l'istruzione READ da eseguire.

5 Illegal Function Call
Richiamo di funzione non valida. E' stata richiamata un'istruzione o una funzione con dei valori non ammessi.

6 Overflow
Il valore numerico ottenuto è troppo grande.

7 Out Of Memory
Memoria esaurita. Non vi è più spazio nella memoria centrale.

8 Undefined Line Number
Numero di riga non definito. Non si possono eseguire istruzioni GOTO o GOSUB senza specificare il numero di riga dove saltare.

9 Subscript Out Of Range
Indice non ammesso. L'indice di un elemento di matrice supera il valore massimo o è negativo.

10 Duplicate Definition
Definizione già esistente. Non è possibile dichiarare due volte la stessa matrice.

11 Division By Zero
Divisione per zero.

12 Illegal Direct
Modalità diretta non valida. L'istruzione non può essere utilizzata in modalità diretta.

13 Type Mismatch
Errore tipo. Non è possibile mettere un numero in una variabile stringa e viceversa.

14 Out Of String Space
Spazio stringa esaurito. Non vi è più spazio sufficiente in memoria centrale nell'area delle stringhe di caratteri.

15 String Too Long
Stringa troppo lunga. Una stringa non può superare i 255 caratteri.

17 Can't Continue
Continuazione impossibile. Il comando CONT non permette di continuare l'esecuzione del programma.

18 Undefined User Function
Funzione utente non definita. Non è stata definita la funzione USR usata.

19 No Resume
Ripresa non possibile. L'istruzione RESUME non è presente nella routine per il trattamento degli errori (ON ERROR GOTO).

20 Resume Without Error
RESUME senza errore. Si incontra l'istruzione RESUME quando in realtà non c'è stato un errore.

21 Undefined Error
Errore non definito (simulato da ERROR).

22 Missing Operand
Operando assente. Manca un operando in un'operazione (addizione, sottrazione, ecc.).

23 For Without Next
FOR senza NEXT. Non è stato incontrato nessun NEXT dopo l'esecuzione di FOR.

24 Can't Exit
Impossibile uscire. L'istruzione EXIT viene usata fuori da un loop.

25 Do Without Loop
DO senza LOOP. Non è stato incontrata nessuna istruzione LOOP dopo l'esecuzione di DO.

26 Loop Without Do
LOOP senza DO. Si incontra l'istruzione LOOP quando non è stata eseguita l'istruzione DO corrispondente. Errori nei file.

50 Bad File Number
Numero file errato. Non è stato trovato questo numero file.

51 Bad File Mode
Modalità file errata. Non è corretta la modalità file usata.

52 File Already Open
File già aperto. Un file già aperto non può essere aperto di nuovo.

53 Device I/O Error
Errore dispositivo di I/O. Problema fisico di accesso alla periferica.

54 Input Past End
Input oltre fine file. Non è possibile leggere al di là della fine del file.

55 Bad File Descriptor
Descrittore file errato. Non è corretto il descrittore del file.

56 Direct Statement In File
Comando diretto nel file. Il file in fase di caricamento contiene un comando di esecuzione diretta.

57 File Not Open
File non aperto. Il file sul quale si vuole leggere o scrivere non è aperto.

58 Bad Data In File
Dati errati nel file. I dati letti in un file non sono del tipo voluto.

59 Device In Use
Periferica già utilizzata.

60 Device Unavailable
Periferica non disponibile.

61 Protected Program
Programma protetto. Il programma è protetto e non può essere listato o salvato.

62 File Not Found
File non trovato. Non esiste il file sul dischetto indicato.

63 Diak Full
Disco pieno. Non vi è più posto disponibile sul dischetto.

64 Too Many Open Disk Files
Troppi file aperti sul disco. Non si possono aprire più file di quanto indicato nell'istruzione FILES.

65 Directory Fult
Indice file completo. Non vi è più spazio nell'indice file.

66 Fife Already Exists
File già esistente. Il nome det nuovo file in NAME corrisponde a quello di un file esistente.

67 Field Overflow
Superamento di campo. La somma delle lunghezze delle variabili di campo in una istruzione FIELD supera la lunghezza del record.

68 String Fielded
Stringa di campo. Una variabile di campo può essere assegnata solo con LSET o RSET.

69 Bad Record Number
Numero di record errato. Il numero di record in PUT o GET non è valido.

70 Bad File Structure
Struttura file errata. Non sono coerenti le indicazioni relative ai blocchi del file nella matrice di allocazione.

71 No Disk
Disco non presente. Il drive è vuoto.

72 Disk Write Protected
Disco protetto in scrittura.

73 Out Of Fielded Buffers
Buffer di file esauriti. Non c'è più spazio sufficiente nella zona riservata ai buffer dei file ad accesso diretto.

74 End of Record
Fine record. La lettura o la scrittura di un file ad accesso diretto supera la fine del record.

75 Verification Failure
Guasto segnatato da verifica. La rilettura dopo la scrittura (VERIFY ON) indica una differenza.

76 Unreadable Diskette
Dischetto illeggibile. Il dischetto non è stato formattato.

78 Bad Picture
Errore immagine. L'immagine non può essere caricata o salvata.

APPENDICE 7

CODICE ASCII

Codice decimale		Codice decimale		Codice decimale		Codice decimale	
000	ZERO	032	Spazio	064	@	096	
001		033	!	065	A	097	a
002	STOP tasto STOP tastiera	034	"	066	B	098	b
003	BREAK (CTRL C)	035	#	067	C	099	c
004		036	$	068	O	100	d
005		037	%	069	E	101	e
006		038	&	070	F	102	f
007	SEGNALE ACUSTICO	039	'	071	G	103	g
008	BS (Back Space) ← tastiera	040	(	072	H	104	h
009	HT (Tabulazione orizz.) → tastiera	041	)	073	I	105	i
010	LF (Avanzamento Riga) ↓ tastiera	042	*	074	J	106	j
011	VT (Tabulazione verti.) ↑ tastiera	043	+	075	K	107	k
012	FF (Salto pagina) CLS HOME tastiera	044	,	076	L	108	l
013	CR (Ritorno a capo) ENTER	045	-	077	M	109	m
014	SO Modalità semigrafica	046	.	078	N	110	n
015	SI modalità alfanumerica	047	/	079	O	111	o
016		048	0	080	P	112	p
017	DC1 Lampeggiamento cursore	049	1	081	Q	113	q
018	DC2 Ripetizione	050	2	082	R	114	r
019		051	3	083	S	115	s
020	DC4 Arresto cursore	052	4	084	T	116	t
021		053	5	085	U	117	u
022	SS2 Tasto caratteri accentati	054	6	086	V	118	v
023		055	7	087	W	119	w
024	CAN Cancella la fine della riga	056	8	088	X	120	x
025		057	9	089	Y	121	y
026		058	:	090	Z	122	z
027	ESC Richiamo di una sequenza	059	;	091	[	123	
028	INS (inserimento tastiera)	060	<	092	\	124	¦
029	DEL (cancellazione tastiera)	061	=	093	]	125	
030	RS (tasto)	062	>	094	^	126	~
031	US Separatore d'articolo	063	?	095	_	127	•

APPENDICE 8

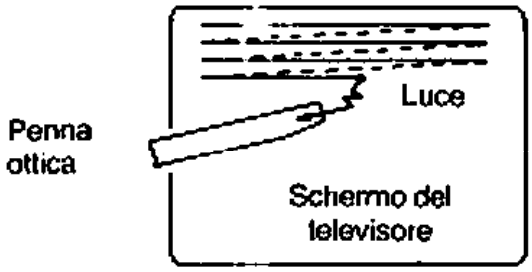
La penna ottica

La penna ottica amplia la possibilità del computer; essa permette di selezionare delle informazioni sullo schermo del televisore e di creare un vero e proprio dialogo interattivo con il computer.

La penna ottica svolge due funzioni indipendenti:
1° - Una localizzazione
2° - Una convalida

Localizzazione

L'immagine che appare sullo schermo del televisore sotto il controllo del computer è costituita dalla scansione di uno spot luminoso su 320 righe, 24 volte al secondo. La localizzazione consiste nell'individuazione della posizione relativa della penna sullo schermo, e cioè nel captare il passaggio dello spot luminoso al momento della scansione. Questa operazione è resa possibile grazie ad un rivelatore sensibile alla luminosità del televisore (elemento fotosensibile).



Scansione dello spot luminoso
Un'immagine = 320 x 200 punti in 1/24 di secondo

Una regolazione scorretta della luminosità del televisore può provocare delle difficoltà nella memorizzazione delle informazioni. D'altra parte, queste ultime non vengono prese in considerazione quando la zona puntata con la penna è di color nero o rosso.

Convalida

La convalida viene assicurata da un interruttore situato nella punta della penna. Il contatto si chiude quando la penna viene premuta contro una superficie piatta, come quella dello schermo televisivo.

Sotto il controllo di un programma, è possibile leggere le coordinate del punto puntato sullo schermo ed avere inoltre la conferma che il contatto sia premuto o meno. La penna ottica può quindi essere utile nelle applicazioni interattive, come ad esempio la progettazione, il disegno, i giochi e l'insegnamento computerizzato.

Collegamento

- Innestare la spina (2) della penna ottica nella presa **(1)** del computer.

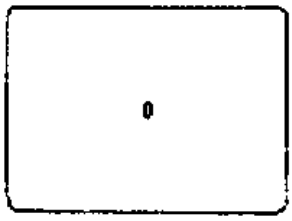
Regolazione della penna ottica

Un monoscopio di regolazione permette al computer di adattare la penna ottica a qualunque apparecchio televisivo. In particolare, il monoscopio garantisce una fase corretta tra l'informazione di rivelazione dello spot luminoso fornito dalla penna ottica ed il segnale di scansione del televisore.

Il metodo per far apparire il monoscopio di regolazione è diverso a seconda che la cartuccia di programma sia inserita o meno nel suo alloggiamento.

1) Senza cartuccia di programma:
Il computer è in funzione **BASIC.**

- Digitare sulla tastiera la parola **TUNE**, indi premere il tasto « **ENTER** ». Il monoscopio appare.

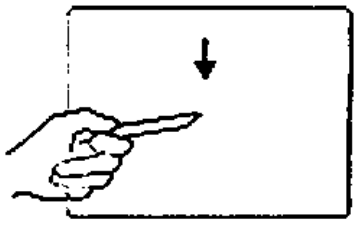


Monoscopio di regolazione della penna ottica

2) Con cartuccia di programma:
- Consultare le istruzioni specifiche della cartuccia di programma inserita, per conoscere le modalità d'impostazione del monoscopio di regolazione.

Regolazione

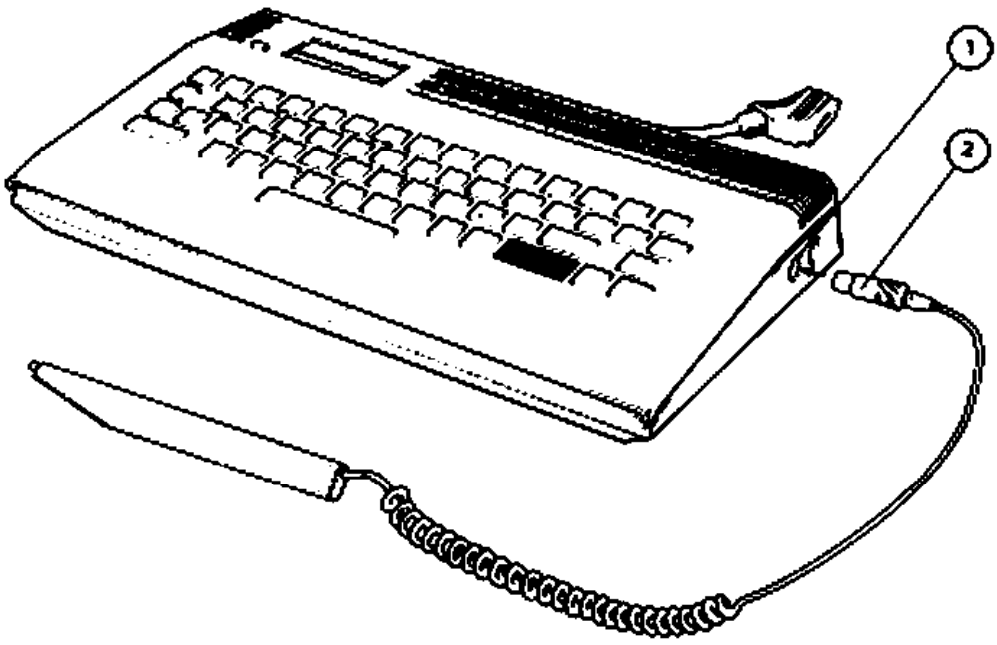
Avvicinare la penna ottica al segmento bianco ed esercitare una leggera pressione:



Un segnale sonoro avverte che la regolazione è stata effettuata. Il monoscopio scompare dallo schermo: la sigla « **OK** » appare ed il cursore lampeggia.

Osservazioni:

La regolazione della penna ottica dovrà essere effettuata di nuovo ogni qualvolta l'alimentazione del computer sia stata interrotta.


1
2

Indice tematico

Questo indice permetterà di ritrovare un'istruzione BASIC relativa ad un argomento specifico. Il numero di pagina corrispondente è reperibile nell'indice analitico.

Comandi e istruzioni generali

RUN
SAVE
SAVEM
SAVEP
SCREENPRINT
SEARCH
STOP
SWAP
TROFF
TRON
WAIT

Funzioni numeriche
ABS
ATN
CDBL
CINT
COS
CSNG
EXP
FIX
INT
LOG
MAX
MIN
RND
SGN
SIN
SQR
TAN

Funzioni relative alle stringhe di carattere
ASC
CHR$
INSTR
LEFTS
LEN
MID$
RIGHT$
SPACE$
STR$
STRING$
VAL

Funzioni varie
BANK
CRUNCHS
EVAL
FKEY$
FRE
HEX$
INKEY$
INPUT$
OCT$
PEEK
USR
VARPTR

FILE
Istruzioni
BACKUP
CLOSE
COPY
DENSYTY
DEVICE
DIR
DIRP
DSKINI
DSKOS
DOS
FIELD

FILES
GET#
INPUT±
KILL
LINEINPUT#
LSET
MOTORON
MOTOROFF
NAME
OPEN
PRINT#
PRINT#USING
PUT#
RSET
SKIPF
UNLOAD
VERIFYON
VERIFYOFF
WRITE#

Funzioni
CVD
CVI
CVS
DSKF
EOF
LOC
LOF
MKD$
MKI$
MKS$
POS

VISUALIZZAZIONE CARATTERI
Istruzioni
ATTRB
CLS
COLOR
CONSOLE
DEFGR$
LOCATE
PALETTE
PRINT
PRINT USING
SCREEN
UNMASK

Funzioni
CSRLIN
GR$
POS
SPC
TAB
SCREEN

VISUALIZZAZIONE GRAFICA
Istruzioni
BOX
BOXF
CIRCLE
CIRCLEF
GET
LINE
PAINT
PATTERN
PSET
PUT
TUNE
WINDOW

Funzioni
POINT

TARTARUGA
Istruzioni
FWD
HEAD
INPUTTURTLE
ROT
SHOW
TRACE
TURTLE
ZOOM

Funzioni
HEAD
ROT
SHOW
TRACE
ZOOM

ALTRI INPUT–OUTPUT
Istruzioni
INMOUSE
INPEN
INPUTMOUSE
INPUTPEN
OPEN...GOTO
OPEN...GOSUB
PEN
PLAY

Funzioni
MTRIG
PTRIG
STICK
STRIG

INDICE

Appendici

Aubin Imprimeur
I. GUGÉ POITIERS