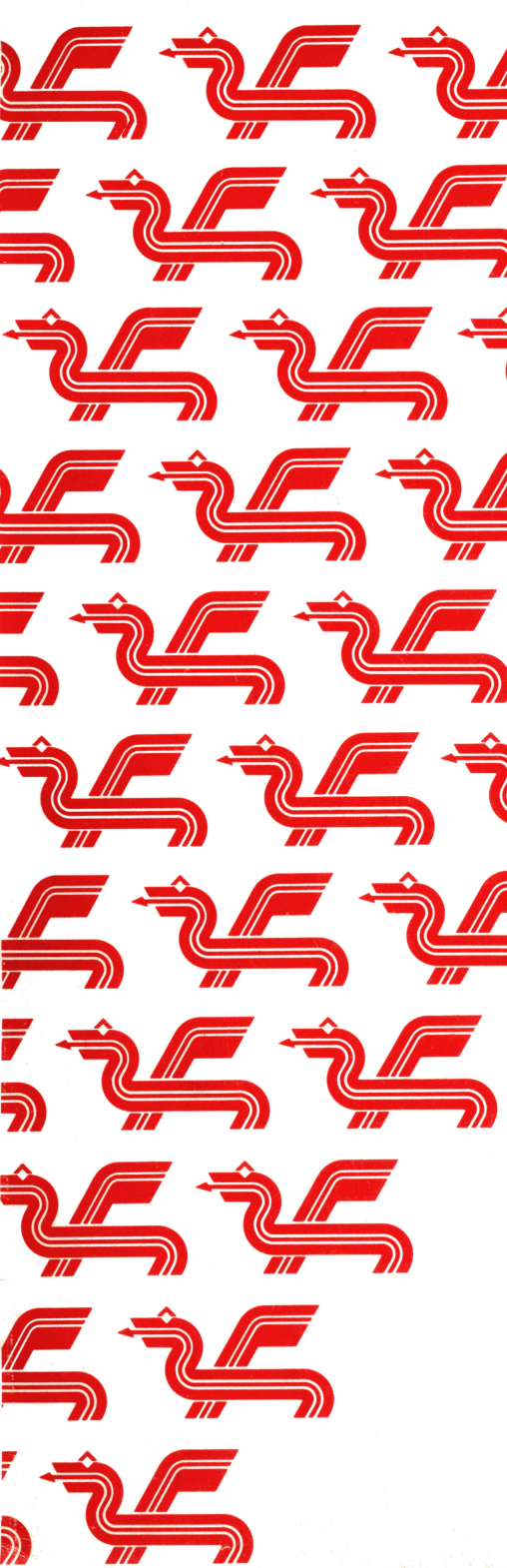


Edizione Italiana

An introduction to BASIC programming using the DRAGON

Data Ltd

By Richard Wadman

ISTRUZIONI OPERATIVE

Nella scatola oltre a questo manuale troverete:

1. Il vostro computer Dragon
2. Un alimentatore
3. Un cavo di collegamento TV.

Oltre a queste cose vi servira' un normale televisore. Il vostro Dragon funzionera' sia con un TV a colori che con uno in bianco e nero. Per sfruttare al meglio le capacita' grafiche del Dragon pero', sarebbe bene collegarlo ad un TV a colori.

Potrete inoltre aumentare le capacita' del computer aggiungendovi le seguenti opzioni:

1. Un registratore a cassetta per memorizzare programmi e dati
2. Una stampante
3. Joysticks per i giochi
4. Disk drives per la memorizzazione di massa di programmi e dati.

Nessuna di queste opzioni e' indispensabile, ma un registratore a cassetta vi risparmiera' di ribattere ogni volta i vostri programmi e dati.

LEGENDA

1. PRESA TV
   Collegamento ad una normale presa da antenna TV.

2. PULSANTE RESET
   Serve a far ritornare il computer allo stato originale. Interrompe immediatamente l'esecuzione del programma o operazioni di input/output. Un eventuale programma attualmente in memoria vi rimane dopo l'uso del reset.

3. PRESA JOYSTICK DI SINISTRA

4. PRESA JOYSTICK DI DESTRA
   Per entrambe vengono usate prese DIN a 5 pins che permettono di collegare i joysticks, disponibili come accessori opzionali.

5. PRESA DI INPUT/OUTPUT PER CASSETTA
   Presa DIN a 5 pins per collegamento del registratore a cassetta. Cavo di collegamento disponibile come accessorio.

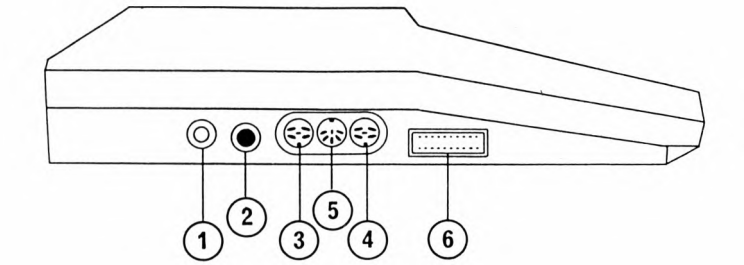
6. PORTA STAMPANTE PARALLELA
   Collegamento per una stampante tipo centronics per mezzo di un cavo standard centronics.

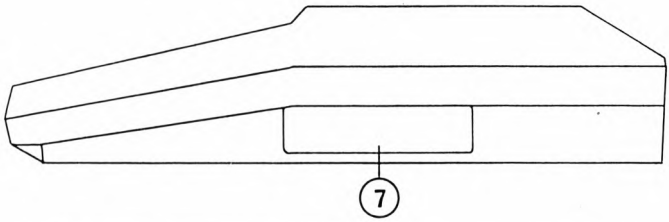
7. FESSURA CARTRIDGES DI PROGRAMMA
   Usata per i cartridges dei giochi. Il cartridge DEVE essere inserito a macchina spenta.

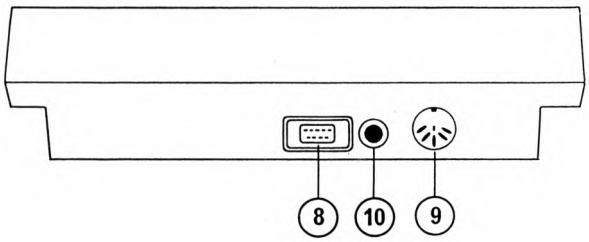
8. PRESA ALIMENTATORE
   Per il collegamento dell'alimentatore di dotazione.

9. PRESA MONITOR
   Per collegamento del monitor a colori.

10. INTERRUTTORE GENERALE
    Controlla l'alimentazione del computer.


1
2
3
5
4
6


7


8
10
9

COLLEGAMENTO DEL DRAGON 32

1. Collegate il cavo di collegamento TV alla presa antenna del televisore ed alla presa TV (1) sul Dragon.

2. Collegate l'alimentatore alla relativa presa (8) sul Dragon. L'altro filo dall'alimentatore dovrebbe essere inserito in una presa da muro a 3 pins.

3. Accendete la TV e il computer.

4. Utilizzando un canale libero sulla TV, regolate il sintonizzatore finche' lo schermo non mostri un quadrato verde dal bordo nero (nel caso di un televisore bianco e nero, risultera' un quadrato grigio chiaro dal bordo nero).

   Nel quadrato sara' riportato il seguente messaggio:

   (C)1982 DRAGON DATA LTD
   16K BASIC INTERPRETER 1.0
   (C)1982 BY MICROSOFT

   OK

   Ora il vostro computer e' pronto per l'uso.

UTILIZZO DI CARTRIDGES DI PROGRAMMA

Collegate il Dragon al televisore come sopra. PRIMA diaccendere il computer, inserite il cartridge, con l'etichetta sul lato superiore, nell'apposita fessura (7).

Prima di inserire o togliere un cartridge assicuratevi che il computer sia **spento**. In caso contrario potrebbero danneggiarsi sia il cartridge che il computer.

UTILIZZO DEI JOYSTICKS PER GIOCHI

Sono disponibili come accessori opzionali vari tipi di joysticks. I joysticks adatti al vostro computer sono del tipo a potenziometro. Per collegarli, basta inserire gli spinotti nelle relative prese (3 e 4).

UTILIZZO DI UN REGISTRATORE A CASSETTA

Per memorizzare programmi e dati dal Dragon, puo' essere

utilizzate qualunque registratore a cassetta di qualita' discreta. Il registratore deve avere prese per il controllo a distanza (remote), l'auricolare (ear) e il microfono (mic). Sono disponibili, come accessori, cavi di collegamento per la maggior parte dei registratori. Il collegamento al computer e' la presa (S). I collegamenti al registratore dipendono dal modello di quest'ultimo. Vedete il capitolo 4 del Manuale per istruzioni in merito all impiego del registratore.

COME AVER CURA DEL VOSTRO DRAGON

1. Tenete ogni tipo di liquido lontano dal computer. Il Dragon non apprezza il te' o il caffe' quanto voi.

2. Assicuratevi che tutti fili liberi siano disposti in modo da non costituire un pericolo. Inciamparvi potrebbe costarvi caro.

3. Assicuratevi che tutte le spine siano ben inserite nelle prese **prima** di accendere la macchina.

4. Spegnete tutto e staccate la macchina dall'alimentazione dopo l'uso.

5. Per pulire la superficie esterna e la tastiera, prima staccate la corrente. Poi, con un panno leggermente inumidito,pulite la superficie e la tastiera. NON usate agenti a base di spiriti.

INDICE

CAPITOLO

PREMESSA

Questo manuale serve a chi desidera imparare a programmare il proprio computer DRAGON con il linguaggio BASIC.

Il BASIC e' un liguaggio di programmazione di grande potenza, ma nello stesso tempo facile da apprendere. Comprende meno di 100 frasi, cioe' meno dell'1% del vocabolario della persona normale.

Pur sembrando difficile al primo approccio, come qualsiasi argomento nuovo, la programmazione in realta' non fa che risolvere un problema mediante certi passi logici. Il segreto sta nel prendervela con calma, assicurandovi di aver compreso il passo precedente prima di affrontare quello successivo. Il fatto che commetterete degli errori non deve preoccuparvi; fa parte della procedura di apprendimento. Non recherete alcun danno al computer facendo un errore di programmazione; basta trovare l'errore, correggerlo e proseguire. Sperimentate pure - l'approccio "Che cosa succedera' se faccio cosi'?" puo' essere un modo molto veloce di scoprire esattamente che cosa puo' fare la vostra macchina. Provate a introdurre ed eseguire ogni esempio per scoprire non soltanto che cosa fa ma anche perche' lo fa.

Con tempo e pazienza scoprirete che il computer non e' soltanto una macchina per i cartrdges dei giochi. La programmazione puo' essere una attivita' affascinante e divertente, a parte il piacere che puo' derivare dai risultati.

► Per assistere i principianti a comprendere e sviluppare programmi in BASIC, una freccetta appare nel margine accanto a voci di particolare importanza. Queste sono regole da ricordare.

►► Se avete gia' usato il BASIC,attenzione al simbolo delle due freccette; questo indica caratteristiche speciali del BASIC Dragon.

CAPITOLO I

PER COMINCIARE

LA TASTIERA

Avete predisposto il computer secondo le istruzioni e ora siete pronti per cominciare. Allora accendete la macchina. Sullo schermo dovrebbe apparire un quadrato VERDE con una scritta.(In caso negativo, spegnete la macchina e controllate i collegamenti). La scritta dipendera' del tipo di macchina, se i disk drives sono collegati o meno, ecc. - quindi non ce ne occupiamo. L'ultima riga, pero', e' sempre uguale:

OK

OK e' il segnale mediante il quale il computer dice che e' pronto a ricevere istruzioni. Dovete aspettare il segnale prima di battere qualunque cosa. Al di sotto dell'OK vi e' un quadratino lampeggiante, detto cursore, che indica il punto della riga in cui vi trovate.

Ora guardate la tastiera: assomiglia a una macchina da scrivere con qualche carattere in piu'. E si comporta pure come una macchina da scrivere - se premete alcuni tasti vedrete apparire il carattere corrispondente sullo schermo. Osservate il modo in cui il cursore si sposta dopo ogni battuta per vedere a che punto siete arrivati. Se premete [SHIFT] e Ø [zero] insieme e continuate a battere, vedrete che le lettere sullo schermo sono diventate verdi su fondo nero. In questo manuale utilizzeremo Ø per rappresentare lo zero, per distinguerlo dalla lettera O. Il computer e' molto pignolo per quanto riguarda questa distinzione. I caratteri verdi su fondo nero sono lettere minuscole (a,b,c,d) ma saranno visualizzate come tali soltanto sulla stampante. TUTTI i vostri comandi o istruzioni al computer devono essere dati in lettere maiuscole - quindi premete [SHIFT] e Ø (zero) insieme ancora una volta e battete ancora un po'. Dovreste avere di nuovo delle lettere nere su fondo verde.

Ora provate il tasto [<-]. Questo tasto, che d'ora in avanti chiameremo tasto di BACKSPACE, serve per spostare il cursore indietro lungo la riga,facendo sparire le lettere immediatamente a sinistra del cursore stesso. E' utile per correggere degli errori - basta tornare indietro fino alla lettera sbagliata e ribatterla.

Per pulire lo schermo completamente premete [CLEAR]; ogni cosa scompare dallo schermo e il cursore si sposta nell'angolo in alto a sinistra. [CLEAR] non fa che pulire lo schermo, non interessa i dati memorizzati nel computer.

Esercitatevi per un po' di tempo con la tastiera per familiarizzare con la posizione dei tasti e con il metodo per correggere gli errori. Poi pulite lo schermo e assicuratevi che siete nel modo maiuscolo.

USIAMO IL DRAGON COME UNA CALCOLATRICE

Il vostro computer puo' funzionare in due modi - **immediato** (esegue il comando subito), **differito** (memorizza una serie di istruzioni da eseguire successivamente in forma di programma). Nel modo immediato il computer funziona come una calcolatrice.

Il computer utilizza un linguaggio chiamato BASIC. Nel BASIC ci sono delle parole particolari per comandare il computer. Il comando PRINT, per esempio, dice al computer di scrivere cio' che segue sullo schermo.

Provatelo: battete PRINT 12 + 7 e poi premete il tasto [ENTER]; sullo schermo dovrebbe apparire

19

OK

Provatene un altro. Battete PRINT 12 + 8/2 poi premete [ENTER]

16

OK

Se avete commesso un errore di digitazione avrete ricevuto la risposta

?SN ERROR

Questo significa "errore di sintassi", cioe'il computer non riconosce qualcosa, di solito perche' e'stata scritta in modo errato. Il computer dara' messaggi di errore quando non capisce il comando e talvolta quando capisce il comando ma lo considera una richiesta illogica o impossibile.

Battete PRINT 3/0 e premete [ENTER]

Il messaggio sara'

?/0 ERROR

che significa che avete tentato di dividere per zero, il che e' impossibile.

I messaggi di errore sono molto sintetici per risparmiare spazio,ma un lista completa con le probabili cause e' riportata nell'Appendice C.

Per tornare agli errori di sintassi, bisogna notare che il computer esige l'ortografia esatta dei suoi comandi in BASIC. Se notate un errore prima di premere [ENTER] potete usare il tasto di backspace [<-] per tornare indietro e correggerlo.Altrimenti, per il momento non potete che ribattere la riga correttamente.

REGOLE ARITMETICHE

Finora i nostri esempi hanno richiesto due sole operazioni aritmetiche, cioe' la somma (+) e la divisione (/). La parola operazione significa qualcosa che chiediamo al computer di fare. Il computer puo' eseguire 6 operazioni aritmetiche e esistono delle regole fisse di esecuzione. Nell'esempio sopra

```
PRINT 12 + 8/2
```

non e' chiaro se la risposta giusta e' 16 o 10.

12 piu' 8 diviso 2 fa 10, o 8 diviso 2 fa 4, piu' 12 fa 16.

La risposta del computer e' 16 a causa dell'ordine di esecuzione delle operazioni aritmetiche. Le operazioni e le priorita' sono le seguenti:

1.Il segno meno

Si ha quando il segno del meno viene utilizzato per indicare un numero negativo

```
PRINT -3 + 2
```

Il computer prima applica il valore meno al numero. cosi' -3 + 2 fa -1. Se il computer facesse la somma prima, - 3 + 2 farebbe - 5, ma non e' cosi'.

2.Esponenziazione

Esponenziazione significa elevazione a potenza. 5 elevato alla potenza 4,(5^4),e' pari a 5 x 5 x 5 x 5.

Dopo aver applicato tutti i segni meno, il computer esegue tutte le esponenziazioni.

```
PRINT 4 + 3 ↑ 2
```

e' calcolato elevando al quadrato tre (3 x 3 = 9) poi sommando 4 per ottenere un totale di 13. Se ci sono piu' esponenziazioni vengono calcolate da sinistra a destra. Provate un esempio

PRINT2 ↑ 3 ↑ 2 ↑ 3 (il tasto [↑] alla sinistra della tastiera viene utilizzato per la funzione di esponenziazione).

viene calcolato moltiplicando 2 per se stesso 3 volte (2 x 2 x 2 = 8) poi moltiplicando il risultato per se stesso (8 x 8 = 64) e poi moltiplicando quest'ultimo risultato per se stesso tre volte (64 x 64 x 64 = 262144).

3. Moltiplicazione

Il segno utilizzato dal computer per la moltiplicazione e * in modo da non creare confusione con la lettera X. Questo segno e' ottenuto premendo i tasti [SHIFT] e [:].

p.e.
PRINT 5 * 2 + 3

viene valutato come 13 (5 x 2 = 10 piu' 3 = 13).

4. Divisione

Il segno usato dal computer per la divisione e' / quindi 3 : 2 viene scritto 3/2.

PRINT 5/2 + 3

viene calcolato come 5.5 (5 : 2 = 2.5 piu' 3 = 5.5).

La moltiplicazione e la divisione hanno la stessa precedenza, cioe' la stessa priorita'. Quando operatori aritmetici hanno la stessa priorita' vengono valutati da sinistra a destra.

PRINT 5 + 2 * 3 + 4/2

e' calcolato come 13 (2 x 3 = 6, 4 : 2 = 2, 5 + 6 + 2 = 13).

5. Addizione

Il segno per l'addizione e' +

6. Sottrazione

Il segno per la sottrazione e' -

L'addizione e la sottrazione hanno uguale priorita' quindi sono calcolate anch'esse da sinistra a destra dopo l'esecuzione di tutte le altre operazioni di priorita' piu' alta.

Per riassumere l'ordine di priorita' seguito dal computer nell'esecuzione delle operazioni matematiche, e' il seguente:

PRIMO - (segno meno per indicare numeri negativi)
SECONDO ↑ (esponenziazione, da sinistra a destra)
TERZO * / (moltiplicazione e divisione, da sinistra a destra)
QUARTO + - (addizione e sottrazione, da sinistra a destra)

Seguono delle espressioni aritmetiche da valutare. In ogni caso, fatelo prima mentalmente (o con carta e matita) e poi provatelo sul computer. Se la vostra risposta e' diversa da quella del computer cercate di capire perche'. A meno che non abbiate gia' un notevole esperienza con il metodo seguito dal computer nella valutazione delle espressioni, dovreste eseguire questi esempi uno per uno. La maggior parte dei cosidetti "errori di computer" sono causati dal fatto che il programmatore segue regole diverse da quelle seguite dal computer. Le risposte non sono riportate, ma se gli esempi vengono inseriti esattamente come sono scritti, il computer dara' la risposta corretta.

```
PRINT 3 + 2
PRINT 4 + 6 - 2 + 1
PRINT 8 * 4
PRINT 4 ↑ 2 + 1
PRINT 5/4 - 1
PRINT 5 - 4/2
PRINT 6 * - 2 + 6/3 + 8
PRINT 4 + - 2
PRINT 2 * 2 + 3 * 4
PRINT 8/2/2/4
PRINT 20/2 * 5
PRINT 8 * 2/2 + 5 * 3 * 2 ↑ 2
```

Non e' necessario battere PRINT ogni volta, il carattere punto interrogativo (?) puo' sostituirlo.

? 3 + 2 e' uguale a PRINT 3 + 2.

Ormai dovreste essere abituati a premere il tasto [ENTER] dopo ogni riga.
Il messaggio OK indica che il computer e' pronto.
Il tasto [ENTER] indica al computer che voi siete pronti.

Adesso che avete imparato le priorita', vediamo come si puo' modificarle. Per fare cio' si utilizzano le parentesi. Supponiamo che volete dividere 14 per 4 piu' 3, se scrivete - 14/4 + 3 la risposta sara' 6.5 perche' 14 viene diviso per 4 e poi si somma 3 al risultato. Ma non e' questo che volevate. Per ottenere la risposta 2 bisogna scrivere 14/(4 + 3).

Le parentesi modificano la priorita': cio' che sta' tra parentesi viene eseguito per primo. Se ve ne sono diverse si comincia l'operazione da quelle piu' interne.

12/(3 + (1 + 2) ↑ 2) viene valutato come segue

a) 12/(3 + 3 ↑ 2)	(1 + 2) viene calcolato per primo
b) 12/(3 + 9)	poi 3 ↑ 2
c) 12/12	poi (3 + 9)
d) 1	per ultima viene eseguita la divisione

Ecco alcune espressioni da valutare. Ripetiamo, se non avete una grande esperienza con il funzionamento del computer, i pochi minuti dedicati a risolverle vi aiuteranno ad utilizzare il vostro computer in modo piu' efficiente.

```
?44/(2 + 2)
?(44/2) + 2
?4 + (-5 * 2)
?100/(200/(2*(9-5)))
?42/((9/3 + 1.75 + (5/4))
```

LA STAMPA DI PAROLE

Finora abbiamo usato soltanto dei numeri nei nostri comandi PRINT. Il profano spesso concepisce il computer come un "tritanumeri", ma questa non e' la sola funzione del computer. Un computer puo' essere utilizzato anche per manipolare dei caratteri. Con il termine caratteri si intendono le lettere A - Z, i numeri 0 - 9, i segni di punteggiatura e altri caratteri speciali che incontreremo piu' avanti. Il BASIC ci permette di manipolare gruppi di caratteri che chiameremo stringhe.

Una stringa puo' essere qualunque combinazione di caratteri - anche uno spazio puo' costituire un carattere importante di una stringa. Perche' il computer sappia che si tratta di una stringa e non di un numero, la stringa viene racchiusa fra virgolette (""). Alcuni esempi di stringhe sono -

"IL TITOLO", "Z+*?!", "MR J.P.SMITH".

"AXY 479W", "01-479-6172"

Queste ultime due stringhe potrebbero essere la targa di una macchina e un numero telefonico e per quanto entrambi contengano dei **caratteri numerici**, non utilizzeremo una tale serie di cifre per fare della vera aritmetica. Per quanto riguarda il BASIC, una serie di cifre fra virgolette non costituisce un numero; la stringa "12345" e ben diversa dal numero 12345. Infatti, le stringhe e i numeri vengono memorizzati in posizioni completamente separate nella memoria del computer.

provate PRINT "2 + 5 ="; 2 + 5

La prima parte del comando PRINT appare sul video precisamente come e' stata scritta nella stringa.(Le virgolette servono a racchiudere la stringa, non ne fanno parte). La seconda parte dell'istruzione viene valutata come un'espressione numerica, cosi sul video dovrebbe essere visualizzato

2 + 5 = 7

Come abbiamo gia' visto supra,lo spazio puo' essere un carattere importante di una stringa. Nelle istruzioni del BASIC gli spazi non contano; servono solo a facilitare la lettura. Nelle stringhe, invece, gli spazi sono importanti perche' quando una stringa viene visualizzata sullo schermo essa e' una riproduzione esatta del gruppo di caratteri fra le virgolette. Quando incontrerete nel corso del manuale degli esempi in cui intervengono delle stringhe, dovrete quindi rispettare gli spazi inclusi all'interno delle virgolette.

A questo punto dovreste essere in grado di utilizzare il computer per risolvere dei problemi semplici. Ribadiamo ancora una volta che dovreste provare almeno alcuni degli esercizi se non siete molto pratici con il computer.

1) Jim e' alto 168 cm. Qual'e' la sua altezza in pollici?
(1 pollice = 2.54cm)

2) Una ricetta richiede 0.75 kg di farina. Quante libbre di farina occorrono?
(1 kg = 2.2 lbs).

3) La vostra macchina consuma 88 litri in un viaggio. All'inizio del viaggio il contachilometri registrava 10746 e alla fine del viaggio registra 11193. Quanti chilometri per litro avete fatto?

4) Depositate $15 in un conto corrente che paga interessi al tasso del 11% all'anno. Quanto avrete dopo 5 anni?
($M = C(1 + T/100)^N$ C = capitale iniziale, T = tasso di interesse annuo, N = numero di anni, M = capitale risultante dopo N anni).

5) Quanti giri fa una ruota da bicicletta da 26 pollici in un miglio?
(1 miglio = 5280 piedi, circonferenza = π x diametro, π = 3.14159).

RISPOSTE

1) 66.14 pollici

2) 1.65 lbs = 1 libbra 10 once

3) 9.625

4) $25.27

5) 775.69

CAPITOLO II

SIGNIFICATO DEI NOMI

COSTANTI

Tutti gli esempi che abbiamo usato finora fanno uso solo di **costanti**. Una **costante** e' precisamente cio' che dice di **essere** - qualcosa che non cambia. 3.145 e' una costante; se la cambiamo in 3.146 e' un'altra costante. Le costanti sono utili nella programmazione ma meno di quanto lo siano le **variabili**.

VARIABILI

Una **variabile** e' qualcosa che puo' cambiare di valore. Nell'equazione

X + 5 = Y

X e Y sono le variabili in quanto entrambe possono assumere molti valori che rendono l'equazione sempre valida. Nel computer, le variabili rappresentano posti nella memoria del computer - qualcosa di simile ad una serie di caselle o scatole. Per identificarle bisogna dar loro un nome (come X o Y). Le variabili del vostro computer sono di due tipi - **numeriche** o **stringa**, e di due dimensioni - **semplici** o **matrici**. Sappiamo gia' la differenza fra costanti numeriche e stringhe - le variabili numeriche contengono costanti numeriche (numeri) e le variabili stringa contengono costanti stringa (caratteri). Per il momento considereremo le variabili semplici; tratteremo le matrici piu' avanti.

DENOMINAZIONE DI VARIABILI

Per dare un nome ad una variabile **numerica** potete usare qualunque combinazione di una lettera con una lettera o un numero.

N, AA, X, TI, Y, Z9, L5, BZ, PQ, K9

sono tutti validi esempi di nomi di variabili numeriche.

►In realta' il computer accetta un nome di variabile di qualsiasi lunghezza ma riconosce soltanto i primi due caratteri del nome; cosi' accettera' nomi come

BIRRA BICCHIERE BIBITA

ma saranno considerati come la stessa variabile BI. Lo stesso vale per le variabili stringa

NOMI DI VARIABILI NUMERICHE

Una variabile numerica puo' contenere soltanto numeri.

Il nome di una variabile numerica puo' essere formato da qualsiasi combinazione di lettere e numeri ma **deve** cominciare con una lettera.

► Poiche' il computer riconosce soltanto i primi due caratteri di un nome, nomi come

STELLA, STATO, STRETTO

saranno considerati uguali (ST).

I nomi di variabile piu' lunghi di due lettere sono utili per ricordarvi il loro contenuto,

NUMERO, CONTATORE, SOMMA

saranno accettati ma occuperanno in memoria piu' spazio di NU, CO, SO.

FREDDO$ FRANA$ FRIVOLO$

► sono tutti considerati FR$.

Per dare un nome a una variabile **stringa** si possono utilizzare le stesse combinazioni usate per le variabili numeriche ma il nome deve essere seguito dal segno dollaro ($).

A$, F7$, MN$, Z0$, FP$

sono tutti validi esempi di variabili stringa.

ASSEGNAZIONE DI VALORI ALLE VARIABILI

Come utilizziamo queste variabili ? Battete il seguente esempio:

```
A = 5                  ricordatevi di premere il tasto
B = 2                  [ENTER] dopo ogni riga
C = A + B
D = D + 3
PRINT A,B,C,D          (assicuratevi di battere le virgole)
```

Sul video dovrebbero apparire

```
5       2

7       3
```

La prima riga chiede al computer di memorizzare il valore 5 in una variabile denominata A; la seconda riga memorizza 2 nella variabile B. (E' il computer a decidere il punto esatto in cui memorizzare queste variabili, voi dovete fornire soltanto il nome). La terza riga dice di trovare i valori memorizzati nelle variabili A e B, sommarli e mettere il risultato in una variabile denominata C. Fatto questo, le variabili A e B contengono sempre i valori originali (5 e 2 nel nostro caso) e C contiene la somma (7). Per chi conosce l'algebra la quarta riga potra' sembrare un po' strana. Ma questo e' perche' nel linguaggio BASIC il segno di eguaglianza (=) non ha lo stesso significato che ha nella matematica. Il segno = nel linguaggio BASIC significa assegnare a, o prendere l'espressione a destra del segno di eguaglianza, (eventualmente valutarla), e metterla nella variabile a **sinistra** del segno.

► Questo tipo di riga si chiama istruzione di assegnazione, e la parte **sinistra** deve **sempre** essere una variabile. Una espressione tipo 2 = B + C puo' avere senso in algebra ma **non** in BASIC.

► Ritornando all'espressione D = D + 3, questo significa prendere il contenuto attuale della variabile D (in questo caso 0 perche' non le abbiamo ancora assegnato alcun valore), sommarlo con 3 e rimetterlo nella variabile D (in altre parole incrementare la variabile D di 3).

Tutto cio' puo' sembrare un po' complicato ma e' un tipo di istruzione molto utile (e anche molto comune) nella programmazione. Essa evidenzia anche un'altra caratteristica delle variabili - esse possono contenere un solo valore alla volta. Se assegnate ad una variabile un valore (cioe' essa appare a **sinistra** in una espressione di assegnazione), questo valore sostituisce quello precedente ed il valore vecchio e' perso. Potete pero' copiare il valore contenuto in una variabile (o inserendolo in un'altra variabile o utilizzandolo in una espressione come (C = A + B) quante volte volete senza modificarlo.

Ora se battete

```
A = B                ricordatevi di premere [ENTER]
B = 17               dopo ciascuna linea
D = D + 2
PRINT A,B,C,D
```

la risposta sara'

```
2        17
7         5
```

La variabile A ora contiene una copia di B dell'esempio precedente, la variabile B contiene un nuovo valore (17) ed i vecchi valori di A e B sono persi. La variabile C e' rimasta invariata. La variabile D e' diventata 5 perche' conteneva 3 nel precedente esempio e le abbiamo aggiunto 2.

Le variabili stringa si comportano nella stessa maniera (tranne che il nome deve finire con il segno $). Provate questo esempio

```
A$ = "QUESTA E' UNA STRINGA"
B$ = " MOLTO"
C$ = " LUNGA"
D$ = A$ + B$ + B$ + B$ + B$ + B$ + B$
D$ = D$ + C$
PRINT D$
```

Sul video apparira'

```
QUESTA E' UNA STRINGA MOLTO MOLT
O MOLTO MOLTO MOLTO MOLTO LUNGA
```

► Nelle righe 1,2 e 3 abbiamo assegnato valori alle variabili stringa A$, B$ e C$. Nella quarta riga abbiamo aggiunto sei copie di B$ ad A$. Con le variabili stringa, il segno piu' (+) non ha lo stesso significato che ha con le variabili numeriche. Significa che vogliamo aggiungere in coda alla prima stringa. (Per coloro ai quali non piacciono le parole lunghe si chiama concatenazione).

Nella riga 5 aggiungiamo C$ in fondo alla nuova D$. Questo e' un metodo per costruire frasi con il computer.

STRINGHE E NUMERI VANNO TRATTATI SEPARATAMENTE

Ricordatevi di tenere separate le variabili numeriche da quelle stringa; si possono memorizzare soltanto numeri in variabili numeriche e soltanto stringhe in variabili stringa; espressioni del tipo

```
D = "STRINGA"
A$= 6
B = A$ * 2
```

danno il messaggio di errore ?TM ERROR (errore per tipo non congruente)

►Il segno piu' (+) e' l'unico operatore aritmetico che si puo' utilizzare con stringhe e variabili a stringa; tutti gli altri (-,*,/,↑) daranno messaggi di errore.

I COMANDI IN SINTESI

Per il resto del manuale metteremo certe pagine in cornice. Queste pagine daranno dettagli su ogni comando presentato. Di solito ci sara' in fondo alla pagina un piccolo programma di prova per dimostrare l'uso del comando.Studiate questi programmi con attenzione, eseguiteli passo per passo, e cercate di scoprire cosa succedera' prima di farli eseguire dal computer. Questi programmi hanno lo scopo di dimostrare come funziona una particolare espressione, ma spesso comprendono suggerimenti utili che potrete includere piu' avanti nei vostri programmi.

NOMI DI VARIABILI STRINGA

Una variabile stringa puo' contenere soltanto stringhe.

► Un nome di variabile stringa puo' essere formato da qualunque combinazione di lettere e di numeri, ma **deve** cominciare con una lettera e terminare con il segno $.

Come per i nomi delle variabili numeriche, il computer riconoscera' soltanto i primi due caratteri, cosi'

ANDATA$, AN1$, AN2$

saranno considerate uguali (AN$).

CAPITOLO III

FINALMENTE, UN PROGRAMMA

Finora il vostro computer ha fatto poco piu' che riprodurre la riga che avete appena introdotto come input. Ora cominciamo a costruire un programma.

Un programma e' un insieme di istruzioni che ordina al computer di fare qualcosa. Un programma BASIC comprende un certo numero di righe. Ogni riga si compone di due parti: un numero di riga e una o piu' istruzioni. Se' una riga comprende piu' di una istruzione, esse devono essere separate da due punti (:). Un'istruzione e' un comando come quelli che abbiamo gia' adoperato.

```
PRINT A$ : A = 47
```

Ecco un programma BASIC:

```
10 CLS0
20 PRINT"COME TI CHIAMI?"
30 INPUT NOME$
40 I = RND(255) : J = RND(9)-1
50 CLSJ
60 PRINT@200 + J, NOME$;
70 SOUND I,2
80 GOTO 40
```

Come vedete, questo programma contiene dei nuovi comandi BASIC. Non preoccupatevene per il momento, verranno spiegati piu' avanti. Notate la forma di un programma BASIC - una serie di righe, ognuna composta da un numero di riga e almeno una istruzione (la riga 40 ne ha due).

COME INSERIRE UN PROGRAMMA IN MEMORIA

Per inserire un programma nella memoria del computer, bisogna prima liberare la memoria da ogni altra cosa che puo' esservi. Per fare cio' battete NEW e poi premete [ENTER]. Fatto questo inserite ogni riga operando esattamente come sopra, cioe' premendo [ENTER] al termine di ciascuna riga. Vedrete che non succede niente quando premete [ENTER]. Una riga che inizia con un numero non viene eseguita subito ma soltanto memorizzata. Quando si lancia l'esecuzione di un programma, esso inizia con la riga contrassegnata dal numero piu' basso, la esegue e poi passa al numero successivo. Poiche' la sequenza di esecuzione di un programma dipende dal numero di riga, potete inserire le righe in qualunque ordine, il computer le riordinera' nella sequenza giusta.

Provate a introdurre il programma. Se commettete un errore prima di premere [ENTER] usate il tasto con la freccia [←]per tornare indietro come prima. Se invece vi accorgete dell'errore soltanto dopo aver premuto [ENTER], ripetete la riga. Il computer memorizzera' soltanto l'ultima versione della riga stessa.

Dopo aver battuto tutto il programma, per vedere come sono state memorizzate dal computer le righe che avete appena introdotto, battete LIST e premete [ENTER]. Notate che LIST non e' preceduto da nessun numero. In assenza di un numero, il computer esegue l'istruzione immediatamente. Controllate che il programma sia giusto (in caso negativo ribattete le righe sbagliate).

Ora finalmente siamo pronti per far girare il programma. Battete RUN e premete [ENTER]. Il video verra' pulito ed apparira' in alto la richiesta di dare il vostro nome. Battete il vostro nome e premete [ENTER].

Il computer si scatenera' -

Il video lampeggera' in diversi colori ed il vostro nome apparira' in continuo movimento al centro. Tutta questa attivita' sara' accompagnata da strani suoni (se vi ricordate di alzare il volume della TV).

► Questo continuera' all'infinito se non lo interrompete. Un modo di fermarlo e' togliere la corrente - ma questa non e' una soluzione molto soddisfacente - perdereste il programma. Il modo migliore di fermare il programma e' quello di premere il tasto BREAK.

PASSO PER PASSO

Ora che avete visto che cosa fa il programma, lo spiegheremo, riga per riga.

►► 10 CLS

Questa e' la riga con il numero piu' basso e quindi la prima ad essere eseguita. Questa istruzione indica al computer di cancellare tutto cio' che appare sullo schermo e mettere il fondo nel colore normale (verde)

20 PRINT"COME TI CHIAMI?"

Questa e' l'istruzione PRINT che abbiamo gia' visto. Essa fa scrivere il messaggio in alto sul video.

30 INPUT NOME$

L'istruzione INPUT ordina al computer di aspettare finche' voi battete qualcosa che il computer poi memorizzera' nella variabile che segue l'istruzione.

LIST

L istruzione LIST visualizza il programma attualmente in memoria. Non e' preceduta da un numero di linea.

Se il programma e' troppo lungo perche' si possa vedere tutto sullo schermo, potete fermare il listing premendo insieme i tasti [SHIFT] e [@], (bisogna essere veloci). Si puo' far ripartire la lista premendo qualsiasi altro tasto.

Per listare una parte soltanto del programma potete utilizzare

LIST n1-n2

ove n1 e n2 sono due numeri di riga (n2 deve essere maggiore di n1)

►LIST 40-100

visualizzera' tutte le righe fra la riga 40 e la riga 100

►LIST -80

visualizzera' tutte le righe dall'inizio del programma fino alla riga 80

►LIST 120-

visualizzera' tutte le righe dalla riga 120 fino alla fine del programma.

RUN

Il comando RUN e' utilizzato per lanciare l'esecuzione di un programma.

Non e' preceduto da nessun numero.

Se volete far girare il programma partendo da un punto diverso dall'inizio, battete

RUN numero di riga

dove **numero di riga** e' il numero della riga dalla quale volete partire.

RUN 250

NEW

Il comando NEW pulisce la memoria ed azzera tutte le variabili.

Notate che non ha un numero di riga.

E' buona pratica battere NEW prima di inserire un programma per assicurarsi che non sia rimasto qualcosa del vecchio programma che potrebbe interferire con quello attuale.

ISTRUZIONE DI ASSEGNAZIONE

L'**istruzione di assegnazione** e' usata per assegnare un valore ad una **variabile**.

La forma dell'istruzione di assegnazione e' la seguente

►LET **Variabile = espressione**

La parte LET dell'istruzione fa parte del BASIC standard, ma non e' necessaria sul vostro computer, quindi non apparira' in nessun listato di programma di questo manuale.

La parte **variabile** dell'istruzione puo' essere qualunque nome di variabile.

La parte **espressione** puo' essere una costante, un'altra variabile o una combinazione di entrambe collegate da operatori (+,-,*,ecc.). Poiche' le stringhe e le variabili numeriche non si possono mescolare, sia le **variabili** che le **espressioni** devono essere dello stesso tipo.

►Il segno di equaglianza (=) non ha lo stesso significato che ha nell'algebra; e' meglio interpretarlo come "assegnare a". Cio' vuol dire che la stessa variabile puo' apparire in entrambe le parti dell'assegnazione; come per esempio:

```
40 X = X + 1
```

che ordina al computer di aggiungere 1 al valore attuale di X e rimetterlo in X.

```
10 S=0 : N=0 : CLS5
20 PRINT@71,"SCRIVI UN NUMERO";
30 INPUT X : CLS5
40 S=S+X : N=N+1
50 PRINT@195,"HAI SCRITTO ";N;" NUMERI.";
60 PRINT@256,"IL VALORE MEDIO E' ";S/N;
70 GOTO20
```

(Ricordate che il computer in questo caso chiamera' la variabile NO$, e ignorera' tutte le altre lettere. Comunque e' spesso utile dare un nome piu' lungo ad una variabile in modo da renderla piu' comprensibile. Un nome piu' lungo serve anche a ricordare l'uso attuale della variabile).

```
40 I = RND(255) : J = RND(9)-1
```

La riga 40 dimostra come la stessa riga puo' contenere piu' di una istruzione. (Notate i due punti (:) che separano una istruzione dall'altra. Questa riga presenta anche una nuova ►istruzione, RND. Questa istruzione genera un numero a caso. E' come pescare un numero da un cappello. Il numero fra parentesi dopo RND dice al computer la gamma dei numeri da cui scegliere. Nella prima istruzione I = RND(255), RND(255) significa scegliere un numero intero a caso fra 1 e 255 e poi memorizzare questo numero nella variabile I. Nella seconda istruzione, la gamma per il numero a caso e' fra 1 e 9, ma dopo la scelta viene sottratto 1 prima di memorizzare il numero in J. Quindi J sara' un numero compreso fra 0 e 8.

```
►►50 CLS(J)
```

Questa riga cancella tutto lo schermo. Questa volta, pero', il colore dello sfondo dipendera' dal valore di J. Sono disponibili 9 colori numerati da 0 a 8. Questa e' la riga che fa lampeggiare lo schermo in diversi colori.

```
►►60 PRINT@200+J,NOME$;
```

Questa e' una versione piu' sofisticata del vecchio comando PRINT. Ordina al computer di scrivere il valore di NOME$ (in questo caso il vostro nome), cominciando da un punto specifico dello schermo. In questo caso il punto e' 200+J cioe' un punto compreso fra 200 e 208. La posizione 200 e' sulla riga 7 alla colonna 8. (Consultare la tabella PRINT@ per vedere come lo schermo e' suddiviso). Poiche' il valore di J cambia, sembrera' che il vostro nome saltelli sulla riga.

```
►►70 SOUND I,2
```

Questa e' la riga che produce gli strani suoni. Il comando SOUND ordina al computer di utilizzare il generatore di tono per creare dei suoni; il tipo di suono e la sua durata sono determinati dai due numeri che seguono il comando.

```
80 GOTO 40
```

Il comando GOTO significa semplicemente andare alla riga specificata dopo il comando (in questo programma 40).

PRINT

Il comando PRINT viene utilizzato per visualizzare l'output.

Puo' essere utilizzato per l'output di costanti, del valore di una variabile, di stringhe e anche per valutare espressioni.

Se una istruzione PRINT comprende piu' di una voce, queste ► dovrebbero essere separate o da una virgola (,) o da un punto e virgola (;).

►►La virgola fa scrivere l'output in due colonne, ognuna di 15 caratteri. (Se la prima voce e' piu' lunga di 15 caratteri occupera' tutta la riga. La seconda voce apparira' sulla riga successiva).

►Il punto e virgola fa comprimere l'output. Le stringhe verranno scritte una accanto all'altra, e le voci numeriche avranno uno spazio su entrambi i lati. Il punto e virgola ferma il cursore nella sua ultima posizione, pronto a riprendere la stampa quando il programma arrivera' alla istruzione PRINT successiva.

Una istruzione PRINT senza alcuna voce stampa una riga in bianco.

```
10 CLS
20 PRINT
30 PRINT"         IL COMANDO PRINT"
40 PRINT"          DIMOSTRAZIONE"
50 PRINT
60 PRINT"COLONNA UNO","COLONNA DUE"
70 PRINT 14.2,13.7
80 PRINT 1,2,7,11
90 PRINT
100 A$="OUTPUT":B=3
110 PRINT"SECONDA RIGA"
130 PRINT A$;"COMPRESSO SULLA RIGA";B
140 PRINT
150 PRINT"QUESTO APPARIRA'";
160 PRINT" COME UNA RIGA"
170 PRINT"FINE",
180 DIMOSTRAZIONE
```

INPUT

Quando un programma arriva ad una istruzione di INPUT, si ferma ad aspettare che qualcosa venga introdotto da tastiera. Il comando INPUT deve essere seguito da uno o piu' nomi di **variabile** separati da virgole.

```
25 INPUT A,B,F$,H7
```

Questa istruzione richiede l'input di 4 voci. Cio' si puo' fare inserendo un elemento alla volta, premendo il tasto [ENTER] dopo ogni voce, oppure in un'unica soluzione separando ciascun elemento dal successivo con una virgola, per esempio

```
146.2,78.1,STRINGA,3 [ENTER]
```

Prima di una istruzione di INPUT e' buona regola scrivere un messaggio che precisi cosa viene richiesto. Per far cio' potete utilizzare una istruzione PRINT, oppure includere il messaggio nell'istruzione di INPUT come segue:

```
35 INPUT "INSERISCI DUE NUMERI";N1,N2
```

Notate il punto e virgola che separa la stringa dalla lista di INPUT.

Dovete assicurarvi che il tipo di INPUT sia giusto (stringhe per variabili stringa e numeri per variabili numeriche), altrimenti il programma si blocchera' con ?REDO e dovrete ribattere l'input.

Non e' necessario racchiudere le stringhe fra virgolette quando vengono assegnate a variabili stringa; sia STRING che "STRING" sono accettabili.

Nel seguente programma di dimostrazione notate:

a) la riga 170 utilizza il comando INPUT per fermare il programma fino al momento giusto. A$ non conterra' niente.
b) l'uso di variabili stringa (C$,P$ e T$) per evitare di battere lo stesso messaggio piu' di una volta.

```
10 CLS : T$="DIMOSTRAZIONE DEL COMANDO INPUT"
20 P$="PREMI [ENTER] PER CONTINUARE"
30 C$="INTRODUCI 4 NUMERI,"
40 PRINT : PRINT T$ : PRINT
50 PRINT C$;"PREMI IL"
60 PRINT"TASTO [ENTER] DOPO CIASCUNO"
70  INPUT A,B,C,D
```

```
80  PRINT : PRINT"HAI INTRODOTTO QUESTI VALORI:-"
90  PRINT A;B;C;D : PRINT
120 PRINT"ORA ";C$;" SEPARATI";
130 PRINT"DA VIRGOLE E PREMI [ENTER]."
140 INPUT A,B,C,D
150 PRINT : PRINT"QUESTA VOLTA I NUMERI ERANO:-"
160 PRINT A;B;C;D
170 PRINT : PRINT P$ : INPUT A$ : CLS
180 PRINT : PRINT T$ : PRINT
190 INPUT"INTRODUCI UNA STRINGA E UN      NUMERO";B$,N
200 PRINT : PRINT"LA STRINGA CHE HAI INTRODOTTO   ERA '";B$;"'"
220 PRINT : PRINT"E IL NUMERO ERA INVECE";N
230 PRINT : PRINT : PRINT : PRINT "      FINE DIMOSTRAZIONE."
```

Cosi' il programma ritorna alla riga 40 dove sceglie un altro numero a caso per I ed anche per J. Poiche' il colore del fondo e' determinato da J, questo cambia quando il programma raggiunge la riga 50. E poiche' I pure e' cambiato, il suono cambia alla riga 70. Quando il programma raggiunge la riga 80, ritorna alla riga 40 dove sceglie....(e cosi' via, finche' non premete il tasto BREAK).

COME APPORTARE MODIFICHE

Abbiamo visto il nostro primo programma. Non abbiamo fatto molto ma e' un inizio. Siamo partiti ed abbiamo visto alcuni nuovi comandi. Una spiegazione dettagliata si trova nelle pagine "incorniciate" sparse nel testo. (Ogni cornice contiene anche un piccolo programma per illustrare il comando; dovreste provare ad eseguire questi programmi).

Se non siete stati molto impressionati dal primo programma, forse vorrete apportarvi delle modifiche. Modificare un programma gia' battuto e' semplice. Poiche' la sequenza di un programma BASIC viene determinata dal numero di riga, si puo' cambiare la riga battendone una nuova con lo stesso numero. La nuova riga sostituisce quella vecchia. Provate ad inserire una nuova riga 70.

```
70 SOUND I,K
```

Una nuova riga puo' essere inserita in un programma assegnandole un numero che la collochera' nel posto giusto.

Provate a battere

```
45 K = RND(20)
```

Poiche' questa riga ha numero 45, verra' inserita fra 40 e 50. (Quando scriverete i vostri programmi potrete numerare le righe con qualunque numero compreso fra 0 e 63999. E' buona norma utilizzare i numeri 10, 20, 30 ecc. in modo da avere la possibilita' di inserire nuove righe se necessario).

Provate a far girare il programma modificato (battete RUN), la nuova riga 45 sceglie un altro numero a caso, questa volta fra 1 e 20. La riga modificata 70 utilizza questo numero a caso (contenuto nella variabile K), per cambiare la durata del suono.

COSTRUZIONE DI PROGRAMMI

E' facile starsene seduti alla tastiera e battere le righe di un programma. Ma i problemi tendono a sorgere dopo. La digitazione del testo dovrebbe essere l'**ultima** fase della creazione di un nuovo programma.

RND

Il comando RND genera dei numeri a caso.

RND e' una **funzione**. Nel BASIC, una funzione e' qualcosa che prende uno o piu' numeri ed esegue delle operazioni che hanno come risultato un solo valore. I numeri usati dalla **funzione si chiamano argomenti** e sono sempre messi fra parentesi dopo il nome della **funzione**.Si dice che la funzione **restituisce** al programma un valore che e' il risultato delle operazioni in essa contenute.

La funzione RND **restituisce** un numero a caso, che dipende dal valore dell'argomento della funzione.

► Se il valore dell'**argomento** e' 0 (RND(0)), la funzione restituisce un valore tra 0 e 1.

► Se il valore dell'argomento e' maggiore di 0 (RND(6)), la funzione restituisce un valore che e' un numero intero compreso fra 1 ed il valore dell'argomento. (RND(6) restituira' 1,2,3,4,5 o 6 , quale non potete saperlo perche' e' casuale !).

```
10 CLS
20 PRINT@6,"INSERISCI UN NUMERO";:INPUT N
30 CLS : PRINT@192,"3 NUMERI A CASO PER N=";N
40 PRINT@270,RND(N)
50 PRINT@302,RND(N)
60 PRINT@334,RND(N)
70 GOTO 20
```

CLS

Il comando CLS viene usato per pulire il video e stabilire il colore di fondo. Il colore di fondo normalmente e' verde. Se usate CLS da solo il colore sara' verde.

Per cambiare il colore di fondo aggiungete un numero fra 0 e 8 dopo il comando CLS, (CLS2).

I colori disponibili sono:

0- NERO	1- VERDE	2- GIALLO
3- BLU	4- ROSSO	5- BEIGE
6- AZZURRO	7- MAGENTA	8- ARANCIONE

La sfumatura esatta di questi colori dipendera' dal vostro televisore.

Noterete, pero', che qualunque sia il colore del fondo, il computer stampera' tutto il testo in nero o in verde.

```
10 CLS
20 PRINT@0,"    DIMOSTRAZIONE DEL COLORE                DI FONDO";
30 PRINT@192,"INSERISCI UN NUMERO TRA 0 E 8";
40 INPUT C
50 CLSC
60 PRINT@288,"QUESTO E' IL COLORE DI FONDO:";C
70 GOTO 20
```

Cominciate con carta e matita e scrivete cio' che intendete fare. Dividete il problema in parti separate. Molti problemi da risolvere con il computer si dividono facilmente in almeno tre parti:

1) Preparazione, titoli, istruzioni ed input dei dati,
2) Calcoli,
3) Visualizzazione dei risultati.

Ora affrontate ogni parte separatamente, suddividendole ulteriormente finche' non vi trovate con una azione semplice tipo "aggiungere 1 al contatore". Ordinate queste azioni in una sequenza logica, scrivetele per esteso, poi cominciate con la parte successiva. Alla fine avrete una serie di passi (alcuni molto elementari) all'interno di ogni parte. Chiunque legga quanto risulta dovrebbe essere in grado di risolvere il problema, qualunque esso sia, utilizzando soltanto l'aritmetica semplice. (Nel linguaggio del computer questo piano di azione si chiama **algoritmo**). Ora basta trasformare il vostro piano in un linguaggio comprensibile al computer. Idealmente ogni passo del vostro piano dovrebbe tradursi in una riga di programma BASIC.

Dopo la traduzione dovreste provare separatamente ogni parte per assicurarvi che fa cio' che dovrebbe fare, prima di assemblare il programma completo. Quando il programma gira bene non festeggiate l'avvenimento bruciando tutti i vostri fogli di carta! Conservate il piano definitivo poiche' alcune parti potrebbero essere utili in altri programmi. Inoltre alcuni errori si manifestano soltanto dopo un certo tempo, quando avrete dimenticato cio' che avevate fatto. E' anche utile introdurre commenti nel programma stesso per ricordarvi cio' che sta succedendo. A questo scopo il BASIC ha l'istruzione REM. Questa istruzione non produce alcun effetto. Il BASIC ignora tutto cio' che segue sulla stessa riga l'istruzione REM; quindi se appaiono sulla stessa riga piu' di una istruzione, abbiate cura di mettere l'istruzione REM per ultima. Esiste una versione abbreviata dell'istruzione REM che utilizza la l'apostrofo (') , che si ottiene battendo [SHIFT] e [7].

```
10  REM PROGRAMMA PER CALCOLARE LA MEDIA
140 A = B*C : GOTO 15 : REM TORNA ALL'INIZIO
48  K = K+1 : 'INCREMENTA IL CONTATORE
```

Anche se le istruzioni REM occupano del posto in memoria, non e' saggio farne a meno completamente. Puo' essere esasperante cercare di capire i propri programmi un anno dopo averli scritti se non sono stati inseriti dei commenti.

Tutta questa parte puo' sembrare molto noiosa, ma e' un fatto noto nel mondo del computer che si passa piu' tempo a fare il **debugging** (localizzare gli errori in un programma) che non a scrivere il programma.

Print @ Grid

PRINT@

Il comando PRINT@ si usa per collocare l'output in un punto preciso sul video.

A questo scopo il video e' suddiviso in un reticolo 16 x 32 cosi' da ottenere 512 posizioni. Consultate lo schema sulla pagina accanto per vedere come il reticolo e' numerato.

La forma del comando PRINT@ e :

►► PRINT@ **espressione, lista di stampa**

L' **espressione** puo' essere un numero, una variabile o una espressione aritmetica, purche' il suo valore sia compreso fra 0 e 511.

La **lista di stampa** e' la stessa usata nel comando PRINT, puo' contenere numeri, variabili, stringhe o espressioni, separati da virgola o punto e virgola.

Se avete considerato che il vostro video ha 16 righe, allora l'istruzione,

```
PRINT@32 * (RIGA-1),A
```

stampera' il valore di A all'inizio di una riga dello schermo. La riga dipendera' dal valore della variabile RIGA (un numero compreso fra 1 e 16).

L esempio che segue e' un modo volutamente complicato per ottenere un risultato piuttosto semplice. Cercate di capire cosa fa, fate girare il programma, poi scrivetene una versione piu' semplice per ottenere lo stesso risultato.

Notate l'uso del punto e virgola alla fine delle istruzioni per evitare che venga cancellato il resto della riga. (Se non ci credete, provate ad introdurre la riga 110 senza mettere il punto e virgola in fondo).

```
10 CLS : P$="PRINT@" : N=1
20 RIGA=12 : A$="SULLO SCHERMO"
30 PRINT@32*(RIGA-1)+19,"PER SCRIVERE";
40 PRINT@448+8,"OVUNQUE";
50 PRINT@259,"QUESTO";:PRINT@266,"DIMOSTRA";
60 PRINT@32*(RIGA-1)+6,"ESSERE USATO";:PRINT@12,"PAGINA";
70 PRINT@448+16,A$
80 PRINT@32*(RIGA-1)+1,"PUO'"
90 PRINT@275,"COME ";P$;
```

```
100 PRINT@134,"DIMOSTRAZIONE ";P$;
110 PRINT@451,"DATI";
120 PRINT@19,N
130 GOTO 130
```

L'ultima riga manda il computer in un "loop" senza fine, che non cambia niente. Cio' impedisce l'apparizione sul video del messaggio OK quando il programma e' finito. Premete il tasto BREAK per fermare il programma.

SOUND

Il comando SOUND genera un tono di un particolare grado di intensita' e durata. Richiede due argomenti:

►► 50 SOUND P,D

P e un numero compreso fra 1 e 255. Il tono piu' basso e' 1, quello piu' alto e' 255. Il "Do" medio del pianoforte e' P=89.

D puo' essere qualunque numero fra 1 e 255. D = 16 da una nota della durata di circa 1 secondo.

```
10 CLS
20 PRINT@6,"DIMOSTRAZIONE SOUND."
30 PRINT@64,"INTRODUCI UN NUMERO TRA 1 E 255,";
40 PRINT@96,"PER L'INTENSITA' DELLA NOTA";: INPUT P
50 PRINT@192,"INTRODUCI UN NUMERO PER LA     DURATA DELLA NOTA";
60 INPUT D : CLS(RND(8)-1)
70 SOUND P, D
80 GOTO 10
```

GOTO

Il comando GOTO ha la forma:

► GOTO **numero di riga**

Il **numero di riga** deve essere un numero (non una variabile), e deve esistere nel programma. Se il **numero di riga** non esiste il programma si blocchera' con un ?UL ERROR (riga non definita).

L'istruzione GOTO viene eseguita immediatamente e quindi non serve mettere un'altra istruzione sulla stessa riga dopo una istruzione GOTO. Il programma non ci arrivera' mai!

```
20 CLS
30 GOTO 60
40 PRINT"RIGA 40"
50 GOTO 80
60 PRINT"RIGA 60"
70 GOTO 40
80 PRINT"FINE DEL PROGRAMMA"
```

Se lavorate secondo i nostri suggerimenti c'e' una probabilita' minore di inserire errori nel programma, e se per caso ne vengono inseriti, essi sono piu' facilmente localizzabili.

UN ESEMPIO DI PROGRAMMAZIONE

PROBLEMA- Utilizzate il computer per simulare il lancio di due dadi.

PARTE 1 - Visualizzazione di titoli e istruzioni

a) Pulire il video
b) Scrivere il titolo
c) Scrivere le istruzioni
d) Scrivere i sottotitoli, primo dado, secondo dado.

PARTE 2 - Trovare i valori dei due dadi

(Quando si lancia un dado e' casuale quale dei 6 lati appare)
a) Il primo dado e' un numero a caso fra 1 e 6
b) Il secondo dado e' un numero a caso fra 1 e 6

PARTE 3 - Scrivere la risposta

a) Scrivere il valore del primo e del secondo dado.

PARTE 4 - Fermare il programma e ripetere se e' necessario.

a) Fermare il programma
b) Richiedere un altro lancio
c) Ripetere parte 1,2 e 3 secondo la necessita'

Traducendo tutto cio' in linguaggio BASIC (con commenti aggiuntivi) otteniamo:

```
10 'PROGRAMMA SIMULAZIONE LANCIO DEI DADI
20 '
30 'PRIMA SEZIONE
40 CLS: REM PULIZIA DELLO SCHERMO
50 PRINT"       SIMULAZIONE DEI DADI":'REM TITOLO
60 PRINT
70 PRINT"USATE IL TASTO [BREAK] PER       TERMINARE
   IL PROGRAMMA":'ISTRUZIONI
80 PRINT
90 PRINT"PRIMO DADO","SECONDO DADO":'INTESTAZIONI
100 PRINT
110 REM FINE DELLA PRIMA SEZIONE
```

REM

►Il comando REM si usa per inserire commenti in un programma. Il computer ignora tutto cio' che segue un REM (o la sua forma abbreviata ') sulla stessa riga.

```
10 REM QUESTA E' UNA RIGA DI COMMENTO
35 D = B*B-4*A*C : 'CALCOLO DEL DETERMINANTE
```

```
120 REM
130 REM SECONDA SEZIONE
140 D1=RND(6) : 'LANCIO PRIMO DADO
150 D2=RND(6) : 'LANCIO SECONDO DADO
160 'FINE SECONDA SEZIONE
170 '
180 'TERZA SEZIONE
190 PRINT D1, D2 : 'EVIDENZIA RISULTATI
200 'FINE TERZA SEZIONE
210 '
220 'QUARTA SEZIONE
230 INPUT"PREMI [ENTER] PER LANCIARE     I DADI";A$
240 GOTO 40 : 'TORNA ALL'INIZIO
250 'FINE DELLA QUARTA SEZIONE
```

Questo programma in realta' non ha bisogno di tanti commenti. Scivetene la vostra versione.

Non esiste una versione "corretta" del programma, una versione corretta e' una che funziona. Vi possono essere modi piu' raffinati o piu' efficienti per risolvere lo stesso problema. In linea generale, un programma piu' breve e' anche piu' veloce ed occupa meno memoria.

Terminiamo questo capitolo con una versione piu' "raffinata" dello stesso programma. Notate come la suddivisione del problema in piu' parti talvolta permette di assemblare il programma secondo una sequenza diversa, ma con un risultato simile.

```
10 'PROGRAMMA SIMULAZIONE LANCIO DEI DADI
20 '
30 'PRIMA SEZIONE
40 CLS0 : PRINT@8,"SIMULAZIONE DADI";
50 PRINT@166," PRIMO";:PRINT@178,"SECONDO";
60 PRINT@199,"DADO";:PRINT@211,"DADO";
70 PRINT@452,"USA [BREAK] PER TERMINARE";
80 'QUARTA SEZIONE
90 PRINT@356,"PREMI [ENTER] PER LANCIARE";
100 INPUT A$
110 'SECONDA SEZIONE
120 D1=RND(6) : D2=RND(6)
130 'TERZA SEZIONE
140 PRINT@264,D1;:PRINT@276,D2;
150 GOTO 90
```

CAPITOLO IV

IL BUON ORDINE

PREPARAZIONE DEL REGISTRATORE

Alcuni programmi cominciano ad essere piuttosto lunghi ed e' noioso dover ribattere un programma ogni volta che volete farlo girare. Memorizzare i vostri programmi su nastro, e richiamarli in memoria, e' pero' abbastanza facile. Vi serve un registratore a cassetta e un cavetto di collegamento.

Potrete usare qualunque tipo di registratore a cassetta purche' abbia i seguenti requisiti:

a) possa registrare da una fonte esterna (mediante la presa jack normalmente contrassegnata da AUX o LINE IN)

b) abbia un uscita per altoparlante o auricolare (una presa jack contrassegnata da EAR,o MONIT, o L/S, o SPKR.

c) possa essere avviato o fermato mediante comando remoto (da inserire nella piccola presa jack di solito chiamata REM e posta accanto alla presa microfono)

d) possa operare collegato alla rete di alimentazione generale. Cio' non e' essenziale ma pile scariche possono compromettere il buon esito della memorizzazione e del recupero dei programmi.

Per collegare il registratore al computer inserite lo spinotto DIN nella presa TAPE sul lato sinistro del computer.

I tre spinotti all'altra estremita' del cavetto si collegano al registratore nel modo seguente:-

1) Il jack piu' piccolo si inserisce nella presa segnata REM

2) Il jack grande con il filo grigio si collega alla presa AUX

3) l'altro jack grande con il filo nero si inserisce nella presa EAR (presa dell'auricolare).

Accendete il registratore, inserite un nastro e riavvolgetelo fino all'inizio. Ora regolate il volume al 6 (o a meta' tra ON e FULL). Adesso siete pronti a memorizzare i vostri programmi.

MEMORIZZAZIONE DI PROGRAMMI SU NASTRO

Battete un programma ed eseguitelo per assicurarvi che funzioni correttamente. Poi procedete come segue:-

1) Premete i pulsanti PLAY e RECORD contemporaneamente finche' non si bloccano.

2) Battete il comando

►► CSAVE "PROGRAM 1" e premete [ENTER].

Il nome "PROGRAM 1" puo' essere sostituito dal nome che volete (deve iniziare con una lettera e non avere piu' di 8 caratteri). Quando premete [ENTER] il motorino del registratore si attivera' e il programma verra' registrato. Dopo un po' il messaggio OK riapparira' sullo schermo e il registratore si fermera'.

Il programma rimane nella memoria del computer, sul nastro avete registrato una copia. Avete salvato un programma chiamato PROGRAM 1 (o quello che sia) sul nastro.

CARICAMENTO DI PROGRAMMI IN MEMORIA

Per recuperare un programma salvato in precedenza battete NEW per liberare la memoria di ogni programma esistente. La procedura e' la seguente:-

1) Riavvolgete il nastro all'inizio
2) Premete il pulsante PLAY finche' non si blocca
3) Battete il comando

►► CLOAD "PROGRAM 1" e premete [ENTER].

Ora il motorino del registratore si avviera' e una lettera S apparira' nell'angolo in alto a sinistra del video. Cio' significa che il computer sta cercando (searching) il programma. Quando l'avra' trovato la S sara' sostituita da

►► F PROGRAM 1

Quando appare il messaggio OK, e la cassetta si ferma, il programma e' stato caricato nella memoria del computer. Per controllare che sia veramente presente, battete

LIST

Se il programma non c'e',o se appare lo scritto I/O ERROR, forse bisogna regolare il registratore diversamente. Controllate i collegamenti, poi ripetete la sequenza di salvataggio e caricamento con diverse regolazioni del volume finche' non ci riuscite.

►► CSAVE

CLOAD

SKIPF

Il comando CSAVE salva un programma su cassetta. Il nome del programma non deve avere piu' di otto caratteri

CSAVE "PROGRAM"

Per salvare su cassetta dati usate il parametro supplementare A cosi' l'informazione verra' memorizzata nel formato ASCII. Potra' quindi essere letta da un comando INPUT #-1.

CSAVE"DATA",A

Il comando CLOAD carica dalla cassetta in memoria un file programma specificato.

CLOAD "PROGRAM"

Il comando SKIPF salta fino al file successivo a quello del programma specificato, o alla fine del programma specificato.

SKIPF"PROGRAM"

COME SALVARE PIU' PROGRAMMI

Per salvare piu' programmi su nastro, dovete evitare di registrare su programmi gia' registrati. Il nastro deve quindi essere posizionato per registrare dopo l'ultimo programma. I passi sono:-

1) Riavvolgete il nastro fino all'inizio
2) Premete PLAY finche' non si blocca
3) Battete il comando

```
►► SKIPF"PROGRAM 1"
```

Il motorino si avviera', il computer cerchera' il programma (S), lo trovera' (F), leggera' oltre il programma, poi fermera' il motorino e dara' il messaggio OK.

4) Premete il pulsante STOP
5) Premete insieme i pulsanti PLAY e RECORD, assegnate un nome al nuovo programma poi salvatelo (CSAVE).

Dopo aver salvato questo programma siete posizionati oltre la fine del programma stesso e quindi, se volete, potete battere e salvare altri programmi.

SUGGERIMENTI PER UNA BUONA REGISTRAZIONE

1) Collegate il registratore alla rete di alimentazione generale in modo da assicurare velocita' costanti.
2) Utilizzate cassette nuove di prima qualita'. Per quanto possano sembrare piu' convenienti le cassette piu' lunghe (C120), e' meglio adoperare quelle piu' corte (C30 o C12).
3) Iniziate la ricerca sempre dall'inizio della cassetta, non fidatevi del contatore.
4) Non lasciate i pulsanti PLAY e RECORD azionati, premete STOP una volta finita l'operazione di salvataggio o di caricamento.
5) Riavvolgete le cassette prima di metterle via.
6) Mettete un'etichetta sulle cassette subito dopo il salvataggio e, nel caso di programmi importanti, togliete la linguetta di protezione.

Anche se state attenti, possono sempre accadere degli incidenti. Alcuni programmi possono rappresentare un investimento notevole di tempo e di fatica - quindi fatene un'ulteriore copia su un'altra cassetta.

L'EDITOR

Man mano che crescono di lunghezza le righe di un programma, cresce anche la possibilita' di errori di battitura. Finora

l'unico rimedio e' stato quello di ribattere l'intera riga. Con l'EDITOR cio' non e' piu' necessario. L'EDITOR vi permette di spostarvi avanti e indietro su una riga, cambiando, eliminando o inserendo caratteri. Per chiamare l EDITOR battete,

EDIT numero di riga [ENTER]

dove numero di riga e il numero della riga del programma sulla quale volete lavorare. La riga completa verra visualizzata sullo schermo. Poi apparira' il numero della riga con accanto il cursore lampeggiante.

COME SPOSTARSI LUNGO LA RIGA

Ora il cursore si trova all'inizio della riga. Per avanzare lungo la riga premete la barra spaziatrice. Il cursore si spostera', visualizzando i caratteri passati. Per spostarci indietro possiamo utilizzare il tasto di backspace [<-]. Potrete accelerare il movimento battendo un numero seguito dal tasto, cioe' [5] [SPAZIO] spostera' il cursore 5 spazi in avanti e [3] [<-] spostera' 3 spazi all'indietro. Mediante questi due tasti potete posizionare il cursore su qualsiasi carattere della riga. Non vedrete il carattere in quanto il cursore ci lampeggiera' sopra. Altri due comandi vi permetteranno di saltare direttamente al punto giusto della riga. Per aggiungere in fondo a una riga, battete [X]. Il cursore saltera' alla fine della riga e dovrete soltanto battere i caratteri che volete aggiungere. Utilizzando la funzione ricerca potete spostarvi direttamente sul carattere desiderato. Battete [S] seguito dal carattere che vi interessa. (Notate che [S][A] portera' il cursore sulla prima A della riga. Se ve ne e' piu' di una e volete la terza, bisogna battere [3][S][A] che significa "cerca le terza A della riga".

MODIFICHE

Una volta posizionato il cursore, mediante la barra spaziatrice, il tasto di backspace o il tasto S, potrete:-

a) **Cancellare** un carattere battendo [D]. In questo modo cancellerete il carattere sotto il cursore. Per cancellare piu' caratteri, [5][D] cancellera i cinque caratteri successivi a partire della posizione del cursore.

b) **Cambiare** un carattere battendo [C], seguito dal carattere nuovo. [C][F] cambiera' il carattere sotto il cursore in F. Come per la cancellazione, [3][C] cambiera' i tre caratteri successivi in quelli battuti di seguito.

c) **Inserire** caratteri battendo [I] seguito dai caratteri che desiderate inserire. Una volta battuto [I], l'editor entra nel modo inserimento. In questo modo tutto cio' che battete

►► EDIT

Il comando EDIT si usa per modificare il contenuto della riga specificata

EDIT numero di riga

Una volta nel modo EDIT,si possono utilizzare i seguenti comandi.

L	Visualizza la riga allo stato attuale
C carattere	Cambia il carattere corrente
n C caratteri	Cambia i successivi **n** caratteri con i nuovi caratteri
I	Inserisce caratteri
D	Cancella il carattere corrente
n D	Cancella gli **n** caratteri successivi
H	Cancella il resto della riga dalla posizione attuale e entra nel modo inserimento
X	Estende la riga, sposta il cursore in fondo e entra nel modo inserimento
S carattere	Cerca un carattere e si ferma alla sua prima apparizione
n S carattere	Cerca un carattere e si ferma alla sua ennesima apparizione
K	Cancella il resto della riga dalla posizione attuale
n K carattere	Cancella la riga fino all'**ennesima** apparizione del del carattere
n [BARRA SPAZIATRICE]	Fa avanzare il cursore di **n** spazi. In assenza di n si assume 1.
N [<-]	Sposta il cursore indietro di **n** spazi. In assenza di n si assume 1.
[SHIFT][ ]	Lascia il modo inserimento e ritorna al modo editor
[ENTER]	Lascia l'editor, memorizza la riga e ritorna alla tastiera.

viene inserito nella riga. Per lasciare il modo inserimento e ritornare al modo editor normale, dovete battere [SHIFT] e [↑] insieme.

Visualizzate lo stato attuale della riga battendo [L].La riga verra' visualizzata con le eventuali modifiche finora apportate. Dopo aver visualizzato la riga il cursore ritorna all'inizio.

Lasciate l'editor premendo il tasto [ENTER]. in questo modo la riga modificata verra' ricollocata nel programma in memoria. Poi apparira' il messaggio OK. Potete lasciare l'editor in qualunque momento - basta premere [ENTER].

Ecco un esempio di editing. A prima vista puo' sembrare complicato ma vi sorprendera' quanto rapidamente ne diventerete padroni dopo pochissima pratica. I tasti che dovete premere sono indicati fra parentesi quadrate.

```
[E] [D] [I] [T] [1] [Ø] [ENTER]

1Ø PRINT "SI SONO MOLTI ORRORI NELLA RIGA

1Ø ■                          ( ■ indica la posizione del cursore)

[7] [SPAZIO]                  spostati sette spazi piu' avanti
[I] [N] [O] [N] [SPAZIO]      inserisci NON e uno spazio
[SHIFT] [↑]                   lascia il modo inserimento
[C] [V]                       cambia la S in V
[S] [M]                       cerca la prima M
[5] [D]                       cancella MOLTI
[I] [P] [I] [U] [']           inserisci PIU'
[SHIFT] [↑]                   lascia il modo inserimento
[SPAZIO]                      spostati uno spazio piu' avanti
[C] [E]                       cambia O in E
[X]                           spostati a fine riga e mettiti in modo
                              inserimento
["]                           aggiungi le virgolette per chiudere la
                              stringa
[SHIFT] [↑]                   lascia il modo inserimento
[L]                           lista la riga

1Ø PRINT "NON VI SONO PIU' ERRORI NELLA RIGA"
1Ø ■

[ENTER]                       memorizza la riga e lascia l'EDITOR
```

Vi sono altri due comandi EDITOR che dovrebbero essere usati con molta attenzione.

a) Il carattere **"kill"**. Battendo [K] si cancella tutto quello che si trova sulla riga a partire dalla posizione del cursore. [K] seguito da un carattere cancella tutto dal cursore alla prima apparizione di carattere. [3] [K] [A] cancella tutto cio' che e' compreso tra il cursore e la terza A successiva.

b) Il carattere **"hack"**. Battendo [H] si cancella tutto cio' che si trova tra la posizione del cursore e la fine della riga e si entra in modo inserimentoIl comando e' utile per ribattere la parte terminale di una riga.

Se la vostra familiarita' con la tastiera e' tale da non commettere alcun errore di digitazione, potete ignorare questa sezione. Per la maggior parte dei programmatori, pero', l'EDITOR rendera' la vita molto piu' facile. Inserite uno degli esempi visti in precedenza, salvatelo su nastro se volete, e trasformatelo in un nuovo programma facendo uso dell'EDITOR.

ALTRI COMANDI DEL SISTEMA

Comandi come LIST e RUN sono comandi del sistema. Essi non fanno parte del programma ma sono comandi diretti al computer perche' faccia qualcosa subito. Ecco alcuni altri comandi del sistema che facilitano il lavoro di programmazione.

DEL per cancellare righe di programma

►► Per eliminare una riga da un programma finora abbiamo battuto il numero della riga seguito da [ENTER]. Questo sistema va bene per 1 o 2 righe, ma per 30 o 40 come facciamo?

DEL **numero di riga - numero di riga**

cancella una serie di righe a partire dal primo numero di riga fino al secondo numero di riga compreso.

DEL 100-250

cancella dal programma in memoria tutte le righe tra la 100 e la 150 compresa.

Il comando DEL puo' essere usato anche sotto altre forme

DEL 20	cancella solo la riga 20
DEL 30-	cancella tutte le righe dalla 30 alla fine
DEL-200	cancella tutte le righe dall'inizio del programma fino alla riga 200 compresa
DEL-	cancella l'intero programma.

DEL

Il comando DEL viene utilizzato per cancellare righe dal programma attualmente in memoria.

► DEL **numero di riga 1 - numero di riga 2**

Il comando cancella tutte le righe comprese tra **numero di riga 1** e **numero di riga 2** compresi gli estremi. Entrambi i numeri di riga sono opzionali e il comando puo' essere adoperato nelle seguenti forme:

DEL -	Cancella l'intero programma
DEL -100	Cancella dall'inizio fino alla riga 100
DEL 300-	Cancella dalla riga 300 fino alla fine
DEL 40	Cancella solo la riga 40
DEL 100-200	Cancella tutte le righe comprese tra 100 e 200

RENUM

Il comando RENUM permette di cambiare tutti o una parte dei numeri di riga. Il comando RENUM cambia anche i rimandi delle istruzioni di salto (GOTO ecc.) per assicurare che il flusso del programma non cambi.

► RENUM **nuova riga, riga iniziale, incremento**

Il comando rinumera tutte le righe a partire dalla **riga iniziale** con il numero contenuto in **nuova riga** piu' l'**incremento.** Tutti i parametri sono opzionali e l'istruzione puo' assumere le seguenti forme:

RENUM	Rinumera l'intero programma. Le righe verranno numerate 10,20,30,...
RENUM 100	Rinumera l'intero programma. Le righe verranno numerate 100,110,120,...
RENUM 100,50,5	Rinumera a partire dalla vecchia riga 50. Le righe verranno numerate 100,105,110,...
RENUM,,20	Rinumera l'intero programma. Le righe verranno numerate 10,30,50,...

Notate che se un parametro viene omesso e quello successivo e' utilizzato, in luogo del parametro omesso occorre inserire una virgola. RENUM non puo' essere utilizzato per alterare la sequenza delle righe.

RENUM per la rinumerazione delle righe di programma

Il comando RENUM ricordera' tutti o alcuni dei numeri di riga del vostro programma. Cambiera' anche il numero di riga contenuto nelle istruzioni GOTO, GOSUB, IF THEN, ON GOTO, ON GOSUB per assicurare che tali istruzioni continuino a saltare allo stesso punto. Incontremo queste istruzioni piu' avanti.

►RENUM **nuova riga, riga iniziale, incremento**

Nuova riga e' il nuovo numero da assegnare alla prima riga da rinumerare. **Riga iniziale** e' la riga a partire dalla quale desiderate rinumerare, e **incremento** e' l incremento da utilizzare tra le righe rinumerate. Ciascuno di questi parametri puo' essere omesso (o anche tutti). Se omettete nuova riga il computer assumera' 10. L'omissione di **riga iniziale** comporta la rinumerazione dell'intero programma e l'omissione di **incremento** comporta la rinumerazione delle righe con incremento 10.

```
RENUM          rinumera tutto il programma in 10,20,30,...
RENUM 100,50,5 rinumera dalla riga 50 in poi in 100,105,110,...
               Le righe che precedono la 50 restano invariate
RENUM 110,,2   rinumera tutto il programma in 110,112,114,...
RENUM,,5       rinumera tutto il programma in 10,15,20,...
```

Notate che se omettete un parametro, ma volete utilizzare uno dei successivi, dovete inserire una virgola al posto del parametro omesso.

► TR per vedere in dettaglio il flusso del programma

A volte quando avete difficolta' con un programma e' utile poter conoscere esattamente il percorso fatto dal programma. Il **"trace"** vi consente di farlo. Battendo TRON **prima** di lanciare l'esecuzione del programma, voi abilitate il "trace". A questo punto sullo schermo appariranno in successione i numeri di riga per i quali il programma passa. Questo vi permette di vedere se il percorso seguito dal programma e' corretto. Per disattivare il "trace", battete TROFF.

► STOP per fermare il programma e CONT per riprendere

Un programma può essere interrotto durante l'esecuzione inserendo una riga contenente il comando STOP.

```
185 STOP
```

Questa linea causa l interruzione del programma quando raggiunge la riga 185. Sullo schermo apparira' un messaggio che dice che il programma si e' fermato e a quale riga si e' fermato. A questo punto potete controllare il contenuto di ogni variabile usando PRINT o ? Per far ripartire il programma battete CONT (sta per continua) e il programma proseguira' con la riga successiva allo STOP.

Adesso siete in grado di rinumerare programmi, cancellare righe indesiderate, cambiare il contenuto di qualunque riga, e salvare il risultato su cassetta. Poiche' adesso possiamo mantenere i nostri programmi, cominceremo a costruire dei programmi che fanno qualcosa di piu' interessante che far lampeggiare lo schermo e produrre strani rumori.

TRACE

Si puo' seguire il flusso del programma per mezzo del "trace". In questo modo viene stampato il numero di ogni riga per cui il programma passa. Il "trace" deve essere attuato prima di lanciare l'esecuzione del programma.

►TRON attiva il "trace"
►TROFF disattiva il "trace"

Entrambi sono comandi diretti e non richiedono un numero di riga.

END
►STOP
CONT

Il comando END fa terminare l'esecuzione del programma e restituisce il controllo alla tastiera.

Il comando STOP ferma l'esecuzione del programma alla riga che contiene il comando stesso. Il messaggio BREAK AT N appare sullo schermo ad indicare che il programma e' stato fermato al numero di riga N. Per far ripartire il programma usate CONT (continua), senza numero di riga. Il programma proseguira' a partire dalla riga successiva al comando STOP. Un programma non puo continuare dopo una istruzione END, ma deve essere rilanciato.

CAPITOLO V

SALTI QUA' E LA'

Alla fine del capitolo 3, abbiamo parlato di come costruire un programma considerandolo come una serie di parti separate. Ora vedremo come dirigere il percorso seguito dal programma in modo da unire queste parti. Cio' e' possibile con le **istruzioni di salto**. Abbiamo gia' incontrato l'**istruzione di salto** GOTO. Si dice che essa e' un'istruzione di salto **incondizionato** in quanto non appena il programma raggiunge l'istruzione GOTO salta subito al numero di riga specificato e riparte da lì. Il programma non ha nessuna scelta in merito, GOTO significa "va immediatamente" e non "forse" o "qualche volta". Finora abbiamo usato l'istruzione GOTO principalmente per tornare all'inizio del programma. Per quanto sia estremamente utile far ripetere piu' volte una parte del programma, e' improbabile che lo faremo all'infinito. (E' anche vero che non e' molto pratico dover dipendere dal tasto BREAK per fermare il programma).
Fortunatamente il linguaggio BASIC ci fornisce una quantita' di istruzioni che ci permettono di controllare l'andamento del programma.

OPERAZIONI DI SCELTA

La prima di queste istruzioni e' una estensione del vecchio amico GOTO. E' l'istruzione ON...GOTO, che ha la forma seguente:

► **ON espressione numerica GOTO lista di numeri di riga**

L'**espressione numerica** viene valutata e, se e' necessario, troncata per ottenere un numero intero (cioe' si tralasciano i numeri dopo il punto decimale). Il controllo si trasferisce poi ad uno dei numeri di riga della lista. Se l'espressione e' valutata come 1 il programma va al primo numero di riga indicato sulla lista, se e' valutata come 2 va al secondo e cosi' via. Se invece il valore dell'espressione e' minore di 1 o maggiore del numero delle righe nella lista, il computer ignorera' l'istruzione completamente e passera' alla riga successiva. Un valore negativo per l'espressione, pero', provochera' l'interruzione del programma con un messaggio di errore. Di solito e' buona norma controllare, prima di arrivare all'istruzione ON...GOTO, che il valore sia nella gamma desiderata. Ecco alcuni esempi dell'istruzione.

```
140 ON P GOTO 200,300,400
210 ON X-4 GOTO 20,40,700,10,690
185 ON B*C/D-E GOTO 115,285,900,40
```

L'istruzione ON...GOTO e' un modo utile per scegliere tra un numero di opzioni e puo' essere considerata una istruzione di salto condizionato. L'adopereremo in questo senso nell'esempio della sezione successiva.

► ON...GOTO

L'istruzione ON...GOTO esegue un salto multidirezionale a numeri di riga specificati

ON espressione GOTO lista di numeri di riga

L espressione viene calcolata (e troncata se non e' un numero intero). Poi il programma salta al numero nella lista la cui posizione e' pari al valore dell'espressione. Se il valore dell'espressione e' 4, l'istruzione ON...GOTO scegliera' il quarto numero della lista. Il valore dell'espressione non deve essere negativo, ne risulterebbe un errore. Se il valore dell'espressione e' zero o maggiore del numero di righe elencate, l'istruzione verra' ignorata ed il programma passera alla riga successiva.

```
10 CLS : PRINT"SOLUZIONE DI UN'EQUAZIONE DI    SECONDO GRADO"
20 INPUT"A, B, C";A, B, C: IF A=0 THEN 20
30 R=-B/(2*A):D=B*B-4*A*C:S=SGN(D)
40 P=SQR(D*S)/(2*A)
50 ON S+2 GOTO 80,60,70
60 PRINT"RADICI INTERE": PRINT R, R: END
70 PRINT"RADICI REALI":PRINT R+P, R-P: END
80 PRINT"RADICI COMPLESSE";PRINT R, R: PRINT P, -P: END
```

DECISIONI

Una forma di salto condizionato molto piu' utile e' data dall'istruzione IF...THEN. Probabilmente questa e' l'istruzione piu' potente di tutto il linguaggio BASIC, e come tale puo' andare da forme molto semplici a forme molto complesse. Nella sua forma piu' semplice:

► IF condizione THEN numero di riga

significa "IF (SE) la condizione e' vera THEN (ALLORA) va al numero di riga.., altrimenti prosegui alla riga successiva.

```
120 IF D > 9 THEN 250
180 IF A$ = "YES" THEN 600
```

Nella riga 120 il programma trasferira' il controllo alla riga 250 se, **e soltanto se**, il valore di D e' maggiore di 9. Se D e' minore o uguale a 9 il programma riprendera' alla riga seguente. Nella riga 180 il salto alla riga 600 accadra' soltanto se la variabile stringa A$ contiene i caratteri YES. In questo caso deve essere identico: YESS, YEH, YEA, YEP, Y, OK, (e persino 'yes' in minuscolo) faranno si' che il programma ignori l'istruzione di salto.

Nella sua forma completa l'istruzione IF...THEN appare come segue:

► IF condizione THEN azione1 ELSE azione2

che significa "SE la condizione e' vera ALLORA esegui l'azione 1 ALTRIMENTI (ELSE), cioe' se non e' vera, esegui l'azione 2".

```
210 IF P = 3 THEN PRINT"VERO" ELSE PRINT"FALSO"
```

In questo caso apparira' VERO soltanto se P = 3. Se P ha un qualsiasi valore diverso da 3 apparira' FALSO. In entrambi i casi il programma passera' alla riga successiva. Sia l'**azione 1** che l'**azione 2** possono contenere piu' di una istruzione.

```
210 IF P = 3 THEN PRINT"VERO" : R = R + 1 : GOTO 560
    ELSE PRINT"FALSO" : L = L + 1
```

Se P e' uguale a 3 il computer scrivera' VERO, aggiungera' 1 alla variabile R e riprendera' il programma alla riga 560. Se P e' uguale a qualsiasi altro valore, scrivera' FALSO, aggiungera' 1 ad L e passera' alla riga successiva.

Per l'**azione 1** e l'**azione 2** il computer accettera' qualsiasi legittima istruzione BASIC, comprese eventuali altre istruzioni IF...THEN. L'unica limitazione deriva dal fatto che l'istruzione IF...THEN deve stare su una sola riga. (Una riga puo' contenere un massimo di 256 caratteri compreso il numero di riga).

Fino a questo punto abbiamo considerato soltanto cio che succede dopo che la condizione e' stata valutata. La parte decisionale del vostro calcolo sta nel valutare la condizione. Una condizione puo' essere TRUE (VERA) o FALSE (FALSA), nient'altro. (Il computer da a TRUE il valore 1 e a FALSE il valore 0). Una condizione semplice ha la forma:

espressione1 relazione espressione2

dove le **espressioni 1 e 2** sono normali espressioni BASIC, come quelle che si trovano nell'istruzione di assegnazione. Entrambe le espressioni devono essere dello stesso tipo (o numeriche o stringhe). Una **relazione** puo' essere una qualsiasi delle seguenti.

SIGNIFICATO	SIMBOLO	ESEMPIO
Uguale a	=	60 IF X=Y+2 THEN 100
Minore di	<	110 IF A*B+2<C/2 THEN 20
Maggiore di	>	185 IF A$>B$ THEN PRINT A$
Minore di o uguale a	<=	220 IF 4*W9<=B/Z9 THEN A=A-1
Maggiore di o uguale a	>=	415 IF B7>=0 THEN X=0
Non uguale a	<>	80 IF A$<>"YES" THEN 999

Notate che una relazione puo' essere applicata a stringhe nonche' ad espressioni numeriche. Quando si confrontano stringhe viene controllato un carattere alla volta, quindi le istruzioni IF...THEN possono essere utilizzate per controllare l'ordine alfabetico.

Per esempio la condizione "A"<"B" e' vera perche' la lettera A precede la lettera B nell'alfabeto. La condizione "AAA"<="AA" e' falsa perche' AAA comparirebbe dopo AA in un vocabolario.

Le condizioni possono essere estese combinando 2 o piu' condizioni per mezzo degli operatori AND (E), OR (O).

```
270 IF A = 4 AND B = 7 THEN 500
```

L'operatore AND significa che entrambe le condizioni devono essere vere contemporaneamente perche' tutta l'espressione condizionale sia vera. Nell'esempio, se A e' uguale a 4 ma B ha un valore diverso da 7, l'intera espressione sara' falsa ed il programma passera' alla riga successiva.

L'operatore OR significa che se una delle condizioni e' vera, tutta l'espressione e' vera.

```
320 IF D < 3 OR F + G > 25 THEN 100
```

Il programma saltera' alla riga 100 se D e' minore di 3 indipendentemente dal valore di F e G. Viceversa, saltera' alla riga 100 se F + G e' maggiore di 25, indipendentemente dal valore di D.

IF...THEN...ELSE

La forma completa del comando e':

► IF **condizione** THEN **azione1** ELSE **azione2**

L'istruzione verifica la **condizione** che sara' vera o falsa. Se e' vera verra' eseguita l'azione 1, se e' falsa l'azione 2.

Una condizione comprende un'espressione, una relazione e un'espressione. Le espressioni possono essere espressioni BASIC qualsiasi, dello stesso tipo (cioe' entrambe numeriche o entrambe stringhe). La relazione e' uno dei seguenti operatori:

= uguale a	<> non uguale a
> maggiore di	< minore di
>= maggiore di o uguale a	<= minore di o uguale a

Le condizioni possono anche essere combinate mediante gli operatori logici AND, OR, NOT.

Condizione AND **condizione** TRUE (VERA) **soltanto se entrambe le condizioni sono vere.**
Condizione OR **condizione** TRUE (VERA) **se una delle condizioni e' vera.**
Condizione NOT **condizione** TRUE (VERA) **se la condizione e'** FALSE (FALSA).

Le **azioni 1 e 2** possono essere qualsiasi istruzione BASIC, compresa un'altra istruzione IF.

La parte ELSE del comando e' opzionale e puo' venire omessa, nel qual caso, se la condizione e' FALSA, il programma prosegue alla riga successiva.

```
10 CLS: PRINT@10,"INDOVINELLO":N=RND(100):T=0
20 INPUT"INDOVINA IL MIO NUMERO";G :T=T+1
30 IF G=N THEN PRINT"HAI INDOVINATO IN";T;"TENTATIVI":END
40 IF G>N THEN PRINT"NO - E' PIU' PICCOLO" ELSE PRINT
   "NO - E' PIU' GRANDE"
50 GOTO 20
```

► INKEY$

Il comando INKEY$ e' una **funzione.** Controlla la tastiera per vedere se e' stato premuto un tasto ed in caso affermativo **restituisce** il carattere stringa del tasto.

Puo' essere utilizzato per inserire un singolo carattere in una variabile stringa e non richiede che venga premuto il tasto [ENTER] dopo l'introduzione.

```
10 CLS:PRINT@9,"PREMI UN TASTO";
20 PRINT@38,"E TI DIRO' QUALE ERA.";
30 B$=" NON STAI PREMENDO ALCUN TASTO "
40 C$="IL TASTO CHE HAI PREMUTO ERA:-"
50 A$=INKEY$
60 IF A$="" THEN PRINT@192,B$; ELSE PRINT@192,C$;A$;
70 FOR D=1 TO 600 : NEXT D : GOTO 50
```

Il seguente programma e' una versione semplificata di un comune programma educativo. I commenti del programma indicano cosa fa ciascuna parte. Notate l'uso di ON...GOTO per scegliere l'opzione e di istruzioni IF...THEN per controllare l'aritmetica. Alla riga 800 introduciamo un'altra parola nuova, INKEY$. Questo comando controlla la tastiera per vedere se un tasto e' stato premuto. In caso affermativo il carattere premuto sara' memorizzato in A$. Non e' necessario premere il tasto [ENTER] dopo l'introduzione del dato. Vedere la cornice INKEY$.

Il programma sembra molto piu' lungo di quanto non sia realmente perche' piu' della meta' e' costituito da commenti. Per risparmiare spazio in futuro non saremo cosi' generosi con i commenti.

```
10 REM PROGRAMMA DI PRATICA ARITMETICA
20 REM
30 REM AZZERA I CONTATORI E TROVA IL LIVELLO DI DIFFICOLTA'
40 REM
50 R=0 : W=0 : CLS
60 T$="PRATICA ARITMETICA"
70 PRINT@7,T$
80 PRINT@64," INTRODUCI UN NUMERO TRA 1 E 10"
90 PRINT@128," PER IL LIVELLO DI DIFFICOLTA'":INPUT L:
   L1=10*L-1
100 REM
110 REM EVIDENZIA LE OPZIONI
120 REM
130 CLS:PRINT@7,T$
140 PRINT@71,"1. ADDIZIONE."
150 PRINT@103,"2. SOTTRAZIONE."
160 PRINT@135,"3. MOLTIPLICAZIONE."
170 PRINT@167,"4. DIVISIONE."
180 REM
190 REM STABILISCE LE POSIZIONI DI STAMPA
200 REM
210 P=224 : Q=352
220 REM
230 REM SELEZIONE DELLE OPZIONI
240 REM
250 PRINT@P,"  QUALE VUOI PROVARE ";:INPUT A
260 REM
270 REM SCEGLIE DUE NUMERI PER IL PROBLEMA
280 REM
```

```
290 N1=RND(L1) : N2=RND(L1)
300 REM
310 REM SALTO ALL'OPZIONE SCELTA
320 REM
330 ON A GOTO 390,450,530,600
340 REM
350 REM OPZIONE SBAGLIATA-RIPROVA
360 REM
370 CLS:SOUND 160,3:GOTO 130
380 REM
390 REM SEZIONE ADDIZIONE
400 REM
410 PRINT@P,"           ADDIZIONE"
420 PRINT@0,"  QUANTO FA";N1;"PIU' ";N2;:INPUT N4
430 N3=N1+N2
440 IF N4=N3 THEN 690 ELSE 730
450 REM
460 REM SEZIONE SOTTRAZIONE
470 REM
480 PRINT@P,"          SOTTRAZIONE"
490 PRINT@0,"  QUANTO FA";N1;"MENO ";N2;:INPUT N4
500 N3=N1-N2
510 IF N4=N3 THEN 690 ELSE 730
520 REM
530 REM SEZIONE MOLTIPLICAZIONE
540 REM
550 PRINT@P,"        MOLTIPLICAZIONE"
560 PRINT@0,"  QUANTO FA";N1;"PER  ";N2;:INPUT N4
570 N3=N1*N2
580 IF N4=N3 THEN 690 ELSE 730
590 REM
600 REM SEZIONE DIVISIONE
610 REM
620 PRINT@P,"           DIVISIONE"
630 PRINT@0,"  QUANTO FA";N1;"DIVISO";N2;:INPUT N4
640 N3=N1/N2
650 IF N3=N4 THEN 690 ELSE 730
660 REM
670 REM RISPOSTA CORRETTA
680 REM
```

```
690 R=R+1:PRINT@P,"           ESATTO.":GOTO 770
700 REM
710 REM RISPOSTA SBAGLIATA
720 REM
730 W=W+1:PRINT@P,"           ERRATO.":
    PRINT@P+32,"      LA RISPOSTA E';N3:GOTO 770
740 REM
750 REM CONTROLLO PER RIPETIZIONE E PULIZIA RIGHE
760 REM
770 FOR D=1 TO 600 : NEXT D
780 PRINT@P," ":PRINT@P+32," ":PRINT@Q," "
790 PRINT@P,"VUOI TENTARE ANCORA ? (S/N)"
800 A$=INKEY$:IF A$="" THEN 800
810 IF A$="S" THEN 130
820 REM
830 REM STAMPA RISULTATI E FINE
840 REM
850 CLS:PRINT@128,"HAI OTTENUTO";R;"RISPOSTE ESATTE":
    PRINT@160,"E";W;"RISPOSTE ERRATE"
860 END
```

USO DEL DO LOOP

Spesso un programma ha bisogno di ripetere piu' volte una sequenza di righe. Il tipo di istruzione di salto che segue permette di farlo. Introducete questo piccolo programma e fatelo girare

```
10 CLS
20 FOR I = 1 TO 50
30 PRINT  198,"CONTATORE I="
40 PRINT  214,I
50 NEXT I
60 PRINT  396,"FINE DEL LOOP"
```

E' troppo veloce per vedere che cosa succede, quindi aggiungete la riga

```
45 FOR J = 1 TO 100 : NEXT J
```

Come vedete le righe che si ripetono sono 30, 40 e 45. Una sequenza ripetuta di questo genere si chiama "loop" (cerchio o anello) perche' il programma esegue un giro, in questo caso ripartendo dalla riga 20.

Le istruzioni che controllano il "loop" sono alla riga 20 (inizio del giro) ed alla riga 50 (fine del giro). La riga 20 significa "FOR (PER) tutti i valori fra 1 e 50 in numeri interi, esegui le seguenti istruzioni fino a raggiungere l'istruzione NEXT". La variabile I fa da contatore ed assume i valori 1,2,3...50. Abbiamo aggiunto la riga 45 perche' il programma contava troppo velocemente. Il loop contenuto in questa riga non comprende alcuna istruzione da eseguire, quindi il contatore (ora la variabile J) conta solo da 1 a 100. La pausa e' sufficiente a rallentare l'altro giro in modo che possiamo leggere i numeri sul video. Se adesso cambiamo la riga 20 in

```
20 FOR I = 1 TO 50 STEP 2
```

il contatore contera' a partire da 1 incrementando di 2 (1,3,5,7...49). L'aggiunta dello STEP ci permette di decidere l'incremento del contatore. Senza la parola STEP il contatore assume un incremento unitario. Modificate l'esempio introducendo le righe seguenti:

```
15 INPUT"INIZIO,FINE,STEP";A,B,C
20 FOR I = A TO B STEP C
70 PRINT  428,"E I=";I
```

Adesso fate girare il programma con vari valori per INIZIO, FINE e STEP. Provate alcuni dei seguenti valori

INIZIO	FINE	STEP
50	1	1
-50	50	5
1	10	0.5
2.6	3.9	0.1
1	-2	1

Avrete notato che si puo' contare sia in avanti che indietro. L'unica regola e' che il valore iniziale deve essere minore del valore finale se lo "step" e' positivo, e maggiore se e' negativo. Dall'ultimo esempio (1,-2,1) vedrete che il giro verra' eseguito una volta lo stesso, anche se infrangete la regola (non si puo' contare da 1 a -2 con salti di +1.

I loops possono "annidarsi" uno dentro l'altro (il loop alla riga 45 e' "annidato" dentro l'altro che comincia alla riga 20). Dovete stare attenti a chiudere i loops nell'ordine giusto.

```
20 FOR I = 1 TO 10
30 FOR J = -2 TO 4 STEP 0.6
40 FOR K = 1 TO 0 STEP -0.1
.
.
.
.
.
.
100 NEXT K
110 NEXT J
120 NEXT I
```

► Ogni istruzione FOR deve avere una istruzione NEXT corrispondente. La variabile che segue NEXT indica a quale FOR appartiene. Dovrete chiudere i loops nell'ordine inverso a quello di apertura. Se le ultime tre righe sopra fossero

```
100 NEXT J
110 NEXT K
120 NEXT I
```

il programma si fermerebbe ed apparirebbe un messaggio di errore perche' si sovrappongono i loops J e K.

Se tutti i loops terminano nello stesso punto le istruzioni NEXT possono essere unite

```
100 NEXT K,J,I
```

ma l'ordine delle variabili deve sempre essere come detto in precedenza.

Le variabili utilizzate per definire un loop (nell'esempio I,A,B e C) possono essere usate anche al di fuori del loop. Ma non potrete modificare l'inizio, la fine ne' il tasso di incremento (STEP). Potreste cambiare il contatore introducendolo alla sinistra di un'istruzione di assegnazione, ma non e' una buona usanza.

Altre istruzioni di salto, tipo GOTO o IF...THEN, possono essere adoperate all'interno di un loop per lasciarlo prima del termine del loop stesso. Non e' pero' possibile saltare in mezzo ad un loop da un punto del programma esterno al loop stesso. **Dovete assolutamente** iniziare un loop con una istruzione FOR.

Il seguente programma dimostra l'uso di loops annidati. Simula un orologio digitale. Per renderlo veramente preciso forse dovrete regolare il loop ritardante alla riga 220. Notate l'impiego di INKEY$ per fermare ed attivare l'orologio alle righe 90,150 e 170.

```
10 CLS0 : PRINT  10,"OROLOGIO DIGITALE"
20 'SCRIVE LE POSIZIONI
```

► FOR...NEXT...STEP

Il comando FOR...NEXT e' costituito da due istruzioni:

FOR **variabile numerica** = espressione TO espressione
STEP espressione

e,

NEXT **variabile numerica**

Le istruzioni FOR...NEXT operano insieme per controllare quante volte viene eseguita una parte di un programma. La tecnica si chiama **looping**.

Le **espressioni** vengono valutate ed il loop conta dal valore della prima espressione TO (A) il valore della seconda espressione secondo un tasso di incremento dato dalla terza espressione. Il valore corrente del contatore e' contenuto nella **variabile numerica** .Il loop verra' eseguito almeno una volta anche se l'intervallo e l'incremento non sono congruenti. Se la parte STEP dell'istruzione non c'e', viene assunta uguale a +1.

I loops possono annidarsi uno dentro l'altro, ma devono rispettare l'ordine giusto.

Potete saltare fuori da un FOR...NEXT usando GOTO, IF...THEN o istruzioni simili. Non potete pero' saltare dentro un loop dall'esterno.

```
10 CLS : CLEAR : DIM L(1000)
20 INPUT"INTRODUCI UN NUMERO";N:IF N>1000 OR N<2 THEN 20
30 CLS : PRINT"NUMERI PRIMI COMPRESI TRA 2 E":N
40 FOR I=2 TO N : IF L(I)<0 THEN 80
50 PRINT I;
60 IF I>SQR(N) THEN 80
70 FOR J=I TO N STEP I : L(J)=-1 : NEXT J
80 NEXT I
```

```
30 P = 261 : Q =196
40 'BLOCCO ISTRUZIONI
50 PRINT@386,"PREMI SPAZIO PER FERMARE E PARTIRE";
60 PRINT@424," E 'R' PER RIPARTIRE";
70 'VISUALIZZA ED ATTENDE PER PARTIRE
80 PRINT@Q," ORE";:PRINT@Q+9,"MINUTI";:PRINT@Q+21,"SECONDI";
90 A$=INKEY$:IF A$<>" " THEN 90
100 'INIZIO LOOP PER ORE, MINUTI E SECONDI
110 FOR H=0 TO 23:FOR M=0 TO 59:FOR S=0 TO 59
120 SOUND 220,1
130 'LOOP PER DECIMI DI SECONDO
140 FOR T=0 TO 9
150 A$=INKEY$:IF A$=" " THEN 200
160 'CONTROLLO PRESSIONE DI UN TASTO
170 A$=INKEY$:IF A$="R" THEN 110
180 IF A$<>" " THEN 170
190 'STAMPA IL TEMPO
200 PRINT@P,H;:PRINT@P+10,M;:PRINT@P+19,S;".";T;
210 'LOOP DI REGOLAZIONE TIMER
220 FOR D=1 TO 13: NEXT D
230 NEXT T
240 'FINE LOOP DECIMI
250 NEXT S,M,H
260 'FINE LOOP SECONDI, MINUTI E ORE
270 PRINT@448,"FERMATO"
280 END
```

SUBROUTINES

Ormai dovreste aver capito che i programmi hanno una struttura complessa che si compone di blocchi piu' piccoli assemblati insieme. Questi blocchi possono servire piu' volte in diversi punti del programma. La struttura del programma spesso puo' essere semplificata se queste situazioni vengono trattate come **subroutines**. Una subroutine e' una parte secondaria di un programma, o un programma all'interno di un programma. La caratteristica principale di una **subroutine** e' che, quando viene **richiamata**, esegue una sequenza di righe e poi ritorna al punto di partenza. Per richiamare una subroutine si adopera l'istruzione

► GOSUB **numero di riga**

dove **numero di riga** e' il numero della riga del programma dove inizia la subroutine. Poiche' da una subroutine bisogna tornare al programma chiamante, ogni subroutine termina con l'istruzione

RETURN

Il GOSUB si comporta in modo analogo al comando GOTO. La differenza sta' nel fatto che con GOTO il programma salta ad una certa riga e procede da li', non ritorna a meno che non incontri un altro comando GOTO. Introducete il seguente esempio e fatelo girare con RUN

```
10 CLS : PRINT"NEL PROGRAMMA PRINCIPALE"
20 GOSUB 50
30 PRINT"DI NUOVO NEL PROGRAMMA PRICIPALE"
40 END
50 PRINT"NELLA PRIMA SUBROUTINE"
60 GOSUB 90
70 PRINT"DI NUOVO NELLA PRIMA SUBROUTINE"
80 RETURN
90 PRINT"NELLA SECONDA SUBROUTINE"
100 RETURN
```

La sequenza dei numeri di righe del programma e la seguente:

10,20,50,60,90,100,70,80,30,40

Notate come la prima subroutine (righe 50-80) richiama la seconda (righe 90,100). L'istruzione END (FINE) che abbiamo inserito significa esattamente quello che vuol dire: si adopera per indicare dove il programma termina realmente. Provate a togliere la riga 40 e fate girare di nuovo il programma. Dovreste ottenere ?RG ERROR IN 80 perche' il programma ha incontrato un'istruzione RETURN senza essere stato mandato ad una subroutine. Il programma e' "caduto" dentro alla prima subroutine. Dovete sempre proteggere le subroutines assicurandovi che vi si entri soltanto con una istruzione GOSUB e che se ne esca soltanto con RETURN. Potete adoperare GOTO, IF...THEN e simili istruzioni di salto all'interno della subroutine, ma esse non devono provocare un salto ad una riga al di fuori della subroutine.

Come nel caso del comando ON...GOTO, e' disponibile, e di forma analoga, un salto multidirezionale.

► ON **espressione** GOSUB **lista di numeri di riga**

La maggior parte dei programmatori esperti tiene un archivio di subroutines in modo da poter costruire molti nuovi programmi "pescando" da tale archivio. Normalmente e' buona norma numerare

le subroutines con numeri elevati, 10000-, in modo che possano inserirsi in un programma senza necessitare di una rinumerazione.

Molti esempi di subroutines saranno presentati nei capitoli seguenti, quindi per ora non forniremo nessun esempio specifico.

GOSUB

RETURN

ON...GOSUB

Il comando GOSUB trasferisce il controllo del programma all'inizio di una subroutine di programma. Il RETURN ritrasferisce il controllo alla riga successiva all'istruzione GOSUB.

GOSUB e' seguito da un numero di riga, che corrisponde alla prima riga della subroutine.

GOSUB 1600

Una subroutine deve contenere almeno una istruzione RETURN.

Il comando ON...GOSUB permette un salto multidirezionale a subroutines in modo analogo al comando ON...GOTO.

ON espressione GOSUB lista di numeri di riga

Se l'**espressione** e' negativa il programma si fermera' con un messaggio di errore. Se l'**espressione** e' zero o maggiore del numero di voci nella **lista di numeri di riga**, l'istruzione verra' ignorata ed il programma passera' alla riga successiva.

```
10 CLS:INPUT"INTRODUCI DUE NUMERI QUALUNQUE";A,B
20 INPUT"ORA INTRODUCI UN NUMERO TRA 1 E 4";C
30 ON C GOSUB 100,200,300,400
50 PRINT C;"NON E' TRA 1 E 4":GOTO 20
100 PRINT"ADDIZIONE.";A;"PIU ";B;"FA";A+B
110 RETURN
200 PRINT"SOTTRAZIONE.";A;"MENO";B;"FA";A-B
210 RETURN
300 PRINT"MOLTIPLICAZIONE.";A;"PER";B;"FA";A*B
310 RETURN
400 PRINT"DIVISIONE.";A;"DIVISO";B;"FA";A/B
410 RETURN
```

CAPITOLO VI

NUOVE DIMENSIONI

Nel capitolo II parlando dei tipi di variabile, abbiamo detto che ne esistono di due dimensioni, **variabili semplici** e matrici. Finora abbiamo utilizzato soltanto **variabili semplici**.

Se doveste decidere di impiegare il vostro computer per tenere un indice di tutti i vostri libri, o dischi, presto rimarreste senza variabili dove immagazinarli. Sarebbe anche un programma molto difficile da scrivere, per tenere traccia di tutti quei titoli diversi! Il vostro computer salva la situazione con le **variabili array** (vettori o matrici).

ELENCHI E TABELLE

Gli arrays sono particolarmente utili per trattare elenchi di voci, quindi potremo organizzare un elenco di libri nella maniera seguente:

1. Titolo 1
2. Titolo 2
3. Titolo 3
.
.
.

Ora per fare riferimento ai nostri libri potremo richiedere il numero 8 dell'elenco. Nel nostro programma diamo un nome a questa sequenza di variabili (i titoli) e facciamo riferimento ad un solo valore nell'elenco dando il numero di indice. Quindi per prima cosa dobbiamo dare un nome all'elenco.

Il nome di un **array** deve rispettare le stesse norme seguite per le **variabili semplici**. Il computer distingue tra le due in quanto gli **arrays** sono **sempre** seguiti da parentesi che contengono il numero di indice.

A(5) si riferisce alla quinta voce di un array numerico (chiamato A).
D7$(28) si riferisce alla ventottesima voce di un array stringa chiamato D7$.

Per definire un array, dovete fornire al computer il suo nome e la sua dimensione. Cio' si effettua tramite l'istruzione DIM.

►DIM **nome di array** (numero), **nome di array** (numero,numero)

DIM sta per dimensionare. L'istruzione non soltanto identifica l'array ma ne stabilisce la dimensione **massima**. Il **numero** puo' essere qualunque numero positivo o una variabile semplice, purche' a quest'ultima sia gia' stato assegnato un valore.

Stabilite la dimensione di cui effettivamente avete bisogno, perche' arrays molto grandi occupano molto spazio in memoria.

```
10 DIM A(22), NA$(40)
```

Questa riga dice al computer di definire un array chiamato A di lunghezza 22 ed un array stringa chiamato NA$ di lunghezza 40 (in realta' le lunghezze sono 23 e 41 perche' il numero indice parte da 0).

Per riferirsi ad un elemento di un array in un'espressione, basta includere il numero di indice, o "subscript", fra parentesi dopo il nome.

```
25 A(4) = 7.0
32 A(M) = B*C/A
```

La riga 25 fissa l'elemento 4 dell'array A a 7.0. La riga 32 valuta l'espressione e mette il risultato nell'elemento M della **matrice** A (notate che la A a destra della riga 32 e' una **variabile semplice** chiamata A, e non ha niente a che fare con l'**array** A a sinistra).

Gli arrays possono avere anche due dimensioni; la riga

```
10 DIM T(10,5), TB$(12,4)
```

creera' un'array numerico, T, che sara una tabella di 10 righe e 5 colonne. L'array stringa, TB$, avra' 12 righe e 4 colonne. Per esempio, un insegnante potrebbe voler tabulare i risultati degli esami per 25 studenti in 6 materie. Un array ESAME(25,6) basterebbe per lo scopo. Per riferirvi ad un elemento nella tabella avrete bisogno di due indici.

```
25 PRINT ESAME(10,3)
```

visualizzerebbe il voto del decimo studente nella terza materia. Naturalmente, gli indici devono riferirsi a qualcosa che esiste realmente. Scrivere ESAME(30,7) provocherebbe un messaggio di errore perche' non avete definito un array di tali dimensioni.

E' ora di vedere un esempio di utilizzo di arrays. Il seguente programma e' breve ma richiedera uno sforzo da parte vostra per capire come funziona. Esso viene usato per mescolare un mazzo di carte. Ogni carta ha un numero, fra 1 e 52, nell array Y. Le voci vengono selezionate da X a caso, ed inserite nell'array Y. Al termine del programma, Y contiene i numeri da 1 a 52, ma in ordine casuale (cioe' mescolati). Non potete adoperare il generatore di numeri casuali, perche' potete avere un solo numero per carta. La funzione RND potrebbe selezionare lo stesso numero piu volte. La riga 50 scrive l'array mescolato.
Notate come fa uso di una espressione come indice dell'array.

►DIM

Il comando DIM viene adoperato per dimensionare gli arrays. Gli arrays possono essere ad una o due dimensioni, numerici o stringa. I nomi degli arrays rispettano le stesse regole dei nomi delle variabili semplici.

DIM **nome di array** (n), **nome di array** (n,n)

Se la dimensione massima dell'array non supera 10 non e' necessaria l'istruzione DIM.

Per riferirsi ad un elemento di un array in un'espressione, il nome deve essere seguito da un **indice** racchiuso tra parentesi.

A(14,K) N9(B) L$(14,4)

```
10 DIM X(52), Y(52) : CLS
20 FOR I=1 TO 52 : X(I)=I : NEXT I
30 FOR I=52 TO 1 STEP-1
40 J=RND(I) : Y(I)=X(J) : X(J)=X(I) : NEXT I
50 FOR I=1 TO 13 : FOR J=1 TO 4 : PRINT Y(4*(I-1)+J); :
   NEXT J,I : END
```

Gli elementi di un array stringa possono essere spostati nella stessa maniera. Una delle applicazioni piu' comuni degli arrays stringa e' la classificazioni di un elenco secondo l'ordine alfabetico. Nell'esempio che segue un elenco di parole viene ordinato dalla subroutine che inizia alla riga 200. Sono stati scritti molti libri sull'ordinamento per mezzo di computer ed il metodo qui adoperato viene detto "ordinamento per scambio". Non e' necessariamente il migliore, ma e' quello piu' semplice. Se non vi e' familiare questo metodo, provate ad eseguirlo a mano usando soltanto l'elenco D, B, A, E, C.

```
10 CLS : DIM W$(50)
20 INPUT"QUANTE   PAROLE";N
30 CLS : PRINT"ORIGINALI"
40 FOR I=1 TO N : PRINT I;".  ";
50 INPUT W$(I) : NEXT I
60 GOSUB 200
70 PRINT  18,"ORDINATE"
80 FOR I=1 TO N
90 PRINT  18+32*I,I;".  ";W$(I)
100 NEXT I : END
200 M=N
210 F=0 : FOR I=1 TO M-1
220 IF W$(I)<=W$(I+1) THEN 240
230 T$=W$(I) : W$(I)=W$(I+1) : W$(I+1)=T$ : F=1
240 NEXT I : IF F=1 THEN M=M-1 : GOTO 210
250 RETURN
```

LE FUNZIONI

Vi ricordate RND e come l'abbiamo sempre chiamata **funzione** ? Ebbene, non e' la sola. Una **funzione**, nel linguaggio del computer, e' un particolare sottoprogramma che restituisce un unico valore a fronte di una serie di argomenti che gli vengono forniti. Nel linguaggio BASIC una **funzione** ha la forma:

► Nome di funzione (**argomenti**)

e puo' essere utilizzata in un'espressione come qualsiasi altro operatore aritmetico (↑,*,/,+,-). Le funzioni, pero', hanno la priorita' su tutti gli altri operatori, fatta eccezione delle parentesi.

Gli **argomenti** di una funzione sono i valori da assegnare ad una funzione, che poi **restituisce** il risultato. Gli argomenti possono essere costanti, variabili o espressioni.

RND(0), RND(X) o RND(A*2+F)

sono tutti argomenti accettabili. Notate che l'argomento e' **sempre** racchiuso fra parentesi.

Il vostro computer vi fornisce diverse funzioni, come RND, che fanno parte del linguaggio BASIC. Le funzioni disponibili "di dotazione" possono essere considerate come appartenenti a 5 classi diverse. Esamineremo una classe alla volta, dando per ciascuna un elenco delle funzioni, una breve spiegazione ed una riga di esempio. D'ora in poi esse compariranno nei programmi, e quindi non presenteremo esempi di programmi per ciascuna, sono troppe!

►Funzioni di Classe I

Queste sono funzioni numeriche, per lo piu' per impieghi matematici. Esse hanno un argomento **numerico** e restituiscono un valore **numerico.** Le funzioni della Classe I possono essere utilizzate soltanto nelle espressioni **numeriche.** Coloro che non conoscono bene le funzioni trigonometriche possono consultare l'Appendice D.

Nome di funzione	Operazione	Esempio
ABS(X)	Valore assoluto di X	100 A=ABS(D*2-C)
ATN(X)	Arcotangente di X, in **radianti** L'inverso di TAN(X)	110 PRINT"ANGOLO="; ATN(R3)
COS(X)	Coseno di X, dove X e' un angolo misurato in **radianti**	510 F7=COS(X+4)
EXP(X)	Eleva la base e (logaritmi naturali) alla potenza X, (e^). Inverso di LOG(X)	215 Q=EXP(-A*A)
►►FIX(X)	Restituisce la parte intera di X, (cioe' tronca tutte le cifre dopo la virgola decimale)	172 N=FIX(Z*0.5)
INT(X)	Tronca se X e' positivo, come per FIX. Se X e' negativo arrotonda per difetto (cioe' INT(-12.0001 e' uguale a 13)	280 P=INT(100*X)

Nome di funzione	Operazione	Esempio
JOYSTK(X)	Restituisce la posizione attuale orizzontale o verticale del joystick di sinistra o destra come segue X=0, joystick destro orizz. X=1, joystick destro vertic. X=2, joystick sinistro orizz. X=3, joystick sinistro vertic.	1040 A=JOYSTK(0) : B=JOYSTK(1)
LOG(X)	Logaritmo naturale di X. Il valore dell'argomento deve essere maggiore di zero. L'inverso di EXP(X)	617 L1=5.2*LOG(W4)
PEEK(X)	Restituisce il contenuto della locazione di memoria con indirizzo X	55 P=PEEK(65208)
POINT (X,Y)	Verifica in grafica a bassa risoluzione se una casella e' occupata o meno da un punto. X deve essere fra 0 e 63 (orizz.) e Y fra 0 e 31 (vertic.). La funzione restituisce il valore 0 se la casella non contiene niente, -1 se contiene caratteri non grafici, e, se la casella contiene un punto, un valore compreso fra 1 e 8 (che indica il colore del punto).	50 IF POINT(5,A)=5 THEN 210
POS(X)	Restituisce la posizione attuale del cursore. Gli unici argomenti sono 0 per la visualizzazione sul video ed 1 per la stampante.	168 IF POS(0)>30 THEN PRINT A$
PPOINT (X,Y)	Verifica, in grafica ad alta risoluzione, se la casella e' occupata o meno da un punto. Restituisce 0 se la casella non contiene niente, altrimenti restituisce il numero del colore del punto.	115 C=PPOINT(A1,A2)

Nome di funzione	Operazione	Esempio
	(X fra 0 e 255, Y fra 0 e 191).	
RND(X)	Restituisce un numero intero a caso fra 1 ed X. Per X=0 restituisce un numero a caso fra 0 e 1.	220 PRINT RND(510); "*";
SGN(X)	Restituisce il segno dello argomento. X negativo restituisce -1 X zero restituisce 0 X positivo restituisce +1.	412 Y=RND(ABS(N)) *SGN(N)
SIN(X)	Seno di X, dove X e' un angolo in **radianti.**	205 S=SIN(K*PI/180)
SQR(X)	Radice quadrata di un argomento ($\sqrt{X}$). X non dovrebbe essere negativo. Se X e' negativo la funzione restituisce $-\sqrt{ABS(X)}$.	330 C=SQR(A*A+B*B)
TAN(X)	Tangente di X, dove X e' un angolo in radianti. L'inverso di ATN(X).	840 R5=B/TAN(EQ-5)

► **Funzioni di Classe II**

Le funzioni di Classe II hanno un argomento **numerico** ma restituiscono un valore **stringa.** Possono essere adoperate esclusivamente nelle espressioni **stringa.**

Nome di funzione	Operazione	Esempio
CHR$(X)	Restituisce il carattere corrispondente al codice dato da X. X deve essere compreso fra 0 e 255. Vedere l'Appendice A per l'elenco dei codici	20 M$=CHR$(143)+CHR$ (128)

Nome di funzione	Operazione	Esempio
HEX$(X)	Calcola il valore esadecimale corrispondente al numero decimale X.	42 PRINT HEX$(30)
STR$(X)	Trasforma una espressione nell'equivalente stringa.	175 A$=STR$(12.49)

►Funzioni di Classe III

Le funzioni di classe III sono funzioni **stringa.** Gli argomenti (di solito almeno due) sono una stringa ed un numero. Restituiscono tutte un valore **stringa** e quindi devono far parte di una espressione stringa.

Nome di funzione	Operazione	Esempio
LEFT$ (X$,N)	Restituisce i **primi** N caratteri della stringa X$.	114 A$=LEFT$(B$,7)
MID$ (X$,M,N)	Restituisce gli N caratteri della stringa X$, a partire dalla posizione M. Se N e' omesso, viene restituita tutta la stringa a destra di M. M deve essere maggiore di 0.	760 K$=MID$(W$,I,4)
RIGHT$ (X$,N)	Restituisce gli **ultimi** N caratteri della stringa X$.	340 T$=RIGHT$(Q$,B+7)
STRING$ (N,C)	Restituisce una stringa di lunghezza N costituita dal carattere definito da C. L'argomento C puo' essere un numero (il codice ASCII del carattere) o il carattere stesso, racchiuso fra virgolette.	400 A$=STRING$(5,67) 410 PRINT STRING$ (32,"*")

►Funzioni di Classe IV

Queste sono funzioni miste simili a quelle della Classe II. Hanno un argomento **stringa** e restituiscono un valore **numerico**, e quindi appaiono soltanto nelle espressioni numeriche.

Nome di funzione	Operazione	Esempio
ASC(X$)	Restituisce il numero di codice ASCII del **primo** carattere della stringa argomento.	715 P=ASC(F$)-64
►►INSTR (P,S$,T$)	Cerca nella stringa S$ la stringa T$ partendo dalla posizione P della stringa S$. Restituisce **0** se non la trova, altrimenti la posizione della stringa T$.	212 F=INSTR(N,X$,"AB")
LEN(X$)	Restituisce la lunghezza della stringa X$. Tutti i caratteri, compresi gli spazi vengono contati. Se la stringa e' vuota, restituisce **0**.	845 N=LEN(N$)
VAL(X$)	Trasforma la rappresentazione di caratteri in cifre. Se la stringa inizia con una lettera, restituisce **0**.	92 Z=VAL(AB$)

►Funzioni di Classe V

Le funzioni di classe V sono funzioni del sistema. Esse non hanno argomenti.

Nome di funzione	Operazione	Esempio
►INKEY$	Controlla la tastiera e restituisce il tasto premuto (se esiste). Restituisce una stringa e quindi deve essere usata in un'espressione stringa.	146 P3$=INKEY$

Nome di funzione	Operazione	Esempio
►►MEM	Fornisce la quantita' di memoria libera disponibile.	PRINT MEM
►►TIMER	Restituisce il contenuto del timer, un valore compreso fra 0 e 65535. Per azzerare usare TIMER=0.	62 T1=TIMER-T 63 TIMER=0

FUNZIONI DELL'UTENTE

A parte le funzioni fornite dal sitema, potete creare voi fino a 26 ulteriori funzioni numeriche. La forma dell'istruzione e':

►DEF FN **lettera (variabile dummy) = formula**

Lettera e' una qualunque lettera fra A e Z. La **variabile dummy** e' una lettera, che verra' sostituita dall'argomento della funzione quando la funzione sara' richiamata. **Formula** e' una espressione BASIC scritta in relazione alla **variabile dummy** e/o altre variabili. Altre funzioni, sia del sistema che dell'utente, possono essere presenti nell'espressione, ma una funzione, **non puo'** richiamare se stessa.

L'equazione $y=((x-3)^2+(x-4)^4)/x^3$,si traduce direttamente in una funzione definita come segue:

```
25 DEF FNY(X)=((X-3)↑2+(X-4)↑4)/X↑3
```

Qui la X e' una **variabile dummy**, non un nome di variabile. Piu' avanti nel programma quando la funzione verra' richiamata, sara' sostituita dall'argomento. Per utilizzare la funzione, basta includerla in una espressione come si fa per quelle del sistema.

```
150 Y(I)=FNY(X)+FNY(W)
```

Le funzioni possono anche essere utilizzate per fornire "routines di servizio" (operazioni usate comunemente). Forse avete fatto caso che l'output dei risultati numerici sul video avviene in maniera un po' disordinata. Il computer cerca di aiutare scrivendo il numero con la massima precisione. Qualche volta questa precisione non e' necessaria e altre volte e' fastidiosa, per esempio quando si devono scrivere totali di cassa.

La seguente funzione puo' essere utilizzata per scrivere fino al numero di cifre decimali desiderato, (D).

```
10 DEF FND(X) = INT(X*10↑D + 0.5)/10↑D
```

Notate che la X e' una variabile dummy, la D **non** e' una variabile dummy, il suo valore deve essere fornito dall'esterno della funzione, possibilmente da una istruzione di INPUT. Per usare questa funzione

```
205 PRINT FND(A)  ecc.
```

Poiche' tutte le funzioni trigonometriche richiedono l'argomento in radianti, una funzione per convertire i gradi in radianti potrebbe essere utile.

```
10 DEF FNR(X) = X/57.295779
```

potrebbe andare bene. Un risultato piu' preciso potra' essere ottenuto utilizzando invece

```
10 DEF FNR(X) = X*ATN(1.0)/45
```

La costante puo' essere creata da

```
20 DEF FNP(X) = 4.0*ATN(1.0)
```

Notate che in questo caso la variabile dummy non ha nessun effetto, e' presente soltanto perche' e' richiesto un argomento. Poiche' non potete richiamare una funzione non ancora definita, e' buona norma inserire le vostre funzioni all'inizio del programma.

ALTERNATIVE ALL'ISTRUZIONE INPUT

Per introdurre valori nelle variabili abbiamo usato un unico sistema - una istruzione di INPUT. Questo sitema e' molto comodo ma, come forse avrete notato, l'istruzione INPUT non accetta certi caratteri. Se cominciate una stringa con degli spazi, questi vanno persi, e se introducete una virgola, tutto cio' che segue va perso. Esiste pero un'alternativa, l'istruzione LINE INPUT.

LINE INPUT "**richiesta**"; variabile stringa

La LINE INPUT si comporta in modo analogo all'INPUT, solo che accettera' tutto, compresi gli spazi e le virgole. La **richiesta** e' identica a quella usata per INPUT e la **variabile stringa** puo' essere qualsiasi variabile stringa. Ogni istruzione di LINE INPUT puo' contenere una sola variabile.

```
25 LINE INPUT"INTRODUCI UNA RIGA DI TESTO";L$
```

Spesso in un programma occorre stabilire un numero di costanti prima che il programma cominci a funzionare veramente.

DEF FN

Il comando DEF FN si usa per definire una funzione numerica dell'utente

► DEF FN**nome(variabile dummy) = formula**

Il **nome** puo' essere qualunque lettera dalla A alla Z.

La **variabile dummy** puo' essere qualunque lettera; viene sostituita dall'argomento quando si adopera la funzione. Si puo utilizzare una sola **variabile dummy**.

La **formula** descrive le operazioni in funzione della variabile dummy e/o altre variabili.

Le funzioni definite dall'utente devono stare in un'unica riga del programma. Una funzione definita puo' utilizzare altre funzioni (o definite o del sistema) nella formula, ma non puo richiamare se stessa.

Una funzione deve essere definita prima di essere utilizzata e quindi dovrebbe comparire all'inizio del programma.

Altre funzioni matematiche possono essere definite come funzioni dell'utente come segue:

```
10 DEF FNS(X) = 1/COS(X) : 'SECANTE
20 DEF FNI(X) = -ATN(X/SQR(-X*X+1))+1.5708
30 DEF FNH(X) = -EXP(X)/(EXP(X)+EXP(-X))*2+1
40 DEF FNM(A) = INT((A/B-INT(A/B))*B+0.5)*SGN(A/B)
50 B = 8 : PRINT FNM(13)
```

LINE INPUT

Il comando LINE INPUT introduce un'intera riga in una variabile stringa.

►LINE INPUT **"richiesta"; variabile stringa**

La **richiesta** e' qualunque messaggio di suggerimento fra virgolette. E' opzionale e, se inclusa, deve essere separata dalla **variabile stringa** mediante un punto e virgola (;). La **variabile** stringa puo' essere qualunque variabile stringa. Una sola variabile puo' comparire nell'istruzione LINE INPUT. La lunghezza della riga memorizzata con il comando LINE INPUT e' di 255 caratteri.

```
10 CLEAR 500 : CLS
20 LINE INPUT"INSERISCI IL TUO NOME COMPLETO";N$
30 LINE INPUT"E L'INDIRIZZO";A$
```

Naturalmente, potreste introdurre le costanti ogni volta che si fa girare il programma con istruzioni di INPUT. C'e' pero' un metodo piu' comodo, che utilizza le istruzioni READ e DATA. Esse sono sempre usate in coppia ed hanno la forma,

► READ **lista di variabili**
► DATA **lista di valori**

L'istruzione READ si comporta come la INPUT, eccetto che invece di fermare il programma ed attendere l'introduzione di un valore, essa cerca il valore in una istruzione DATA che fa parte del programma. Se c'e' piu' di una istruzione DATA, l'istruzione READ comincia dalla istruzione col numero piu' basso e prosegue seguendo l'ordine delle righe.

```
10 DATA 1,2,3,4,5
20 FOR I=1 TO 3
30 READ A : PRINT A : NEXT I
40 READ D,G : PRINT D,G
```

Il suddetto esempio legge la prima voce (1) dell'elenco DATA, la scrive, legge la seconda (2), ecc. Una volta letta una voce, l'indicatore si sposta a quella successiva. La riga 40 legge le ultime due voci. Ora se aggiungete la riga

```
50 READ X : PRINT X
```

e fate girare il programma di nuovo, otterrete ?OD ERROR IN 50. Cio' significa che la READ non ha piu' elementi nell'elenco DATA e quindi e' "senza dati" (OD-out of data). Aggiungete un'altra riga,

```
45 RESTORE
```

Adesso, quando fate girare il programma vedrete che funziona e che X ha il valore 1. L'istruzione RESTORE rimette l'indicatore all'inizio della prima istruzione DATA. Anche le stringhe possono essere usate nelle istruzioni DATA. Dovete assicurarvi che ad ogni sequenza di variabili in una istruzione READ corrispondano valori dello stesso tipo nell'elenco DATA.

Se avete scritto un programma usando molte stringhe forse avete gia' ottenuto un ?OS ERROR. Questo indica che siete andati oltre lo spazio riservato per la memorizzazione di stringhe. Per rendere disponibile spazio in memoria per le stringhe usate l'istruzione CLEAR.

```
10 CLEAR 1000
```

Questa istruzione riserva 1000 bytes di memoria per le stringhe. Poiche' CLEAR ha anche la funzione di azzerare **tutte** le variabili, usatela soltanto all'inizio del programma.

READ
DATA
RESTORE

Il comando READ legge l'elemento successivo nella riga DATA e lo assegna alla variabile specificata in lista.

► READ **lista di nomi di variabili**

La riga DATA memorizza dati all'interno del programma e puo' essere una riga numerata inserita in qualunque punto del programma.

► DATA **lista di variabili**

Si possono usare sia variabili numeriche che stringhe nelle istruzioni READ e DATA, purche' la sequenza sia giusta. Ad una variabile stringa si deve assegnare una stringa, ecc.

Il comando RESTORE rimette l'indicatore dei dati al primo valore della riga DATA col numero piu' basso.

► RESTORE

```
10 CLS : PRINT : PRINT : PRINT
20 READ A,B : IF A=-9999 THEN RESTORE : GOTO 20
30 PRINT A;" + 5 E' UGUALE A "; : INPUT C
40 IF C=B THEN PRINT"ESATTO"ELSE PRINT"SBAGLIATO"
50 FOR D=1 TO 600 : NEXT D : GOTO 10
60 DATA 8,13,12,17,5,10,27,32,14,19,3,8
70 DATA 7,12,6,11,1,6,-9999,-9999
```

► CLEAR

Il comando CLEAR cancella tutte le variabili e riserva dello spazio per la memorizzazione di stringhe.

```
CLEAR 500
```

riserva 500 bytes di memoria per variabili stringa.

Il comando CLEAR puo' essere utilizzato anche per fissare l'indirizzo BASIC piu' elevato in memoria per conservare spazio per routines in linguaggio macchina.

```
CLEAR 200,14000
```

riserva 200 bytes per la memorizzazione di stringhe e fissa l'indirizzo BASIC piu' elevato a 14000. Le routines in linguaggio macchina possono ora essere immagazzinate da 14001 in poi.

Se non si adopera CLEAR, vengono riservati automaticamente 200 bytes per la memorizzazione di stringhe.

UNA PAUSA PER RIFLETTERE

Questo capitolo, insieme ai capitoli 1,2,3 e 5, illustra le regole fondamentali del linguaggio BASIC. Anche se seguiranno delle altre istruzioni, queste sono, in un certo senso, decorative, un po' come la glace sulla torta.

Tutto il materiale finora trattato verra' adoperato spesso nei capitoli seguenti e costituira' parte essenziale di qualunque programma. Siete forse ansiosi di cominciare a fare della grafica con il vostro computer, ma la grafica diventera' tanto piu' facile se vi soffermate un attimo a riflettere per assicurarvi di aver capito bene quanto visto fino a questo punto. Ricontrollate gli esempi e cercate di adattarli a vostre esigenze personali.

Concludiamo questa parte con due ulteriori esempi. Il primo e' una estensione dell'esempio per mescolare le carte presentato nel capitolo precedente. Il programma "distribuisce" una mano di carte. Notate i seguenti punti:

a) il mescolamento delle carte appare ora come una subroutine alla riga 90,
b) l'uso di READ e di DATA ,all'inizio, per definire il contenuto delle matrici,
c) le righe 130 e 140 per identificare il seme e la carta di quel seme.

```
10 DIM X(52),PACK(52),CARD$(13),SUIT$(3)
20 FOR I=0 TO 3: READ SUIT$(I): NEXT I
30 DATA PICCHE,QUADRI,FIORI,CUORI
40 FOR I=1 TO 13: READ CARD$(I): NEXT I
50 DATA ASSO,DUE,TRE,QUATTRO,CINQUE,SEI,SETTE
60 DATA OTTO,NOVE,DIECI,FANTE,REGINA,RE
70 CLS: INPUT"QUANTE CARTE VUOI DISTRIBUIRE ";N
80 GOSUB 190
90 ST=1
100 EN=ST+N-1: IF EN>52 THEN GOTO 80
110 CLS: PRINT@10,"LA TUA MANO": PRINT: PRINT
120 FOR I=ST TO EN
130 S=INT((PACK(I)-1)/13)
140 C=PACK(I)-S*13
150 PRINT TAB(8);CARD$(C);" DI ";SUIT$(S)
160 NEXT I: ST=ST+N
170 PRINT@448,"UN'ALTRA MANO. SI O NO?";:INPUT A$
180 IF A$="SI" THEN 100 ELSE END
190 FOR I9=1 TO 52: X(I9)=I9: NEXT I9
200 FOR I9=52 TO 1 STEP-1
210 J9=RND(I9): PACK(I9)=X(J9): X(J9)=X(I9)
220 NEXT I9: RETURN
```

Il secondo esempio utilizza quasi tutte le funzioni stringa disponibili. Il programma verifica il testo introdotto e riferisce il numero di volte che appare ogni lettera. Questo tipo di programma trova un uso frequente nella decifrazione di messaggi in codice. Con poca fatica lo si puo' adattare per cercare parole o una sequenza di caratteri.

```
10 CLEAR 1000 : CLS : READ A$
20 DATA ABCDEFGHIJKLMNOPQRSTUWXYZ
30 PRINT"INSERISCI UNA RIGA DI TESTO": PRINT
40 LINE INPUT L$
50 FOR I=1 TO LEN(A$): CLS
60 T$=MID$(A$,I,1): C=0: P=1: P$=L$
70 F=INSTR(P,L$,T$)
80 IF F>0 THEN C=C+1 ELSE 140
90 P$=LEFT$(P$,F-1)+STRING$(LEN(T$),CHR$(128))
100 IF F>LEN(L$) THEN 120
110 P$=P$+RIGHT$(L$,LEN(L$)-F)
120 P=F+LEN(T$)
130 IF P<=LEN(L$)-LEN(T$)+1 THEN 70
140 PRINT P$
150 PRINT@354,T$;" TROVATO";C;"VOLTE"
160 PRINT@416,"PREMI SPAZIO PER CONTINUARE,      N PER FINIRE"
170 Z$=INKEY$: IF Z$="" THEN 170
180 IF Z$="N" THEN 200
190 NEXT I
200 CLS : END
```

CAPITOLO VII

LA GRAFICA

Quando il vostro computer visualizza qualcosa sul video, non sta facendo altro che accendere o spegnere dei punti nel tubo catodico, per costruire l'immagine. Se il punto e' acceso appare un puntino colorato, se e' spento appare nero. Tutte le lettere che abbiamo scritto finora sono composte da simili puntini luminosi. La dimensione del punto determina la **risoluzione** in cui state lavorando. Se un punto e' di grande dimensione, vuol dire che state lavorando in **bassa risoluzione** e se un punto e' di piccole dimensioni ci troviamo in **alta risoluzione** (questo perche' piu' piccolo e' un punto, piu' elevato e' il numero di punti disponibili sul video).

►► Il vostro computer ha la possibilita' di operare in cinque diverse risoluzioni, da 512 a 49152 punti. Cio' vi consente una grande flessibilita' per quanto concerne la quantita' di dettagli che potete inserire nelle vostre immagini. Cominceremo creando immagini nella risoluzione piu' bassa, passando gradualmente a quelle piu' alte. I metodi usati per visualizzare disegni e movimenti sullo schermo sono molto simili, indipendentemente dalla risoluzione in cui state lavorando.

LA STAMPA DI IMMAGINI

Ricordate dal capitolo III, quando abbiamo presentato l'istruzione PRINT@, che il video e' suddiviso in un reticolo 16x32. Cio' ci consente di scrivere un carattere in qualunque punto sul video introducendo l'appropriata posizione. Con la funzione CHR$ (vedere il capitolo VI per le funzioni), possiamo generare dei caratteri grafici speciali. Il seguente programma visualizza tutti i caratteri disponibili con CHR$.

```
10 FOR I=1 TO 255 : CLS0
20 PRINT@100,"CHR$(";I;")";
30 PRINT@120,CHR$(I);
40 FOR D=1 TO 600 : NEXT D,I : CLS
```

►► I numeri da 1 a 31 vengono usati per caratteri di controllo, quindi non succede nulla. Da 32 a 127 compaiono i caratteri della tastiera. I codici fra 128 e 255 invece sono i caratteri speciali della grafica. (Una lista completa si trova nell'Appendice A). Questi caratteri grafici sono simboli che rappresentano blocchi colorati che si possono assemblare in forme semplici. L'esempio piu' semplice e' un rettangolo di colore. Per esempio, CHR$(143) produce un rettangolo verde, il colore 1. Aggiungete 16 e CHR$(159) fa un rettangolo giallo, colore 2, e cosi' via. Vi sono 16 simboli, da CHR$(128) a CHR$(143), tutti costituiti da combinazioni di verde e nero. Per ottenere lo stesso simbolo, ma in un colore diverso, basta

aggiungere 16 al codice, per il numero di volte appropriato.

+16 giallo	+32 blu	+48 rosso
+64 beige	+80 azzurro	+96 magenta (porpora)
	+112 arancione	

Il seguente programma dimostra l'effetto dell'aumento di 16 del codice (si puo' usare anche per regolare il colore sul vostro televisore, usate C=143).

```
10 CLS0 :
20 FOR I=1 TO 14 : FOR J=C TO 255 STEP 16
30 FOR K=1 TO 4 : PRINT CHR$(J); : NEXT K,J,I
40 GOTO 40
```

Come vedete dall'esempio precedente, i caratteri CHR$ possono essere scritti direttamente sul video. Ma poiche' essi sono dei caratteri, possono essere inseriti in variabili stringa, il che e' molto piu' comodo in quanto si maneggiano con maggior facilita'.

Ora cominciamo a costruire un'immagine. Disegnamo un castello - primo perche' e' una forma semplice, e secondo perche' fa vedere come, partendo da una base semplice, potete aggiungere sempre maggiori dettagli. Introducete e fate girare ogni parte del programma man mano che lo presentiamo, cosi' potrete vedere le fasi di costruzione dell'immagine.

Per prima cosa dobbiamo costruire un muro che attraversa tutto il video, quindi sara' largo 32 blocchi, e lo faremo alto sei blocchi.

```
10 CLEAR 500 : CLS0
20 FOR I=1 TO 6 : FOR J=1 TO 32
30 MURO$=MURO$+CHR$(207) : NEXT J,I
40 PRINT@256,MURO$;
200 GOTO 200
```

La prima riga riserva lo spazio per le stringhe che useremo. Le righe 20 e 30 "costruiscono" il muro in blocchi di colore beige, CHR$(207), e lo memorizzano in una stringa chiamata MURO$. La riga 40 scrive MURO$ che appare sul video come un blocco colorato . L'ultima riga (200) serve soltanto a fermare l'immagine sul video.

Successivamente aggiungiamo i bastioni, che si ottengono con blocchi alternati in beige e nero. Questa volta ci serve una sola fila.

```
50 FOR I=1 TO 16 : B$=B$+CHR$(128)+CHR$(207) : NEXT I
60 PRINT@224,B$;
```

La riga 50 costruisce i bastioni e la 60 li disegna evidenziandoli al di sopra del muro.

Ora ci serve una torre. La torre si costruisce con lo stesso materiale del muro, quindi prendiamo dei blocchi da MURO$.

```
70 P=11
80 FOR I=1 TO 3 : PRINT@128+32*I+P,LEFT$(MURO$,10); : NEXT I
```

La riga 80 prende 10 blocchi da MURO$ e costruisce 3 file in mezzo al muro. Il valore di P colloca la torre al centro. Se volete mettere la torre altrove, cambiate il valore di P. La torre e' larga 10 blocchi, quindi P puo' essere qualunque numero tra 0 e 22.

Facciamo la stessa cosa per i bastioni della torre.

```
90 PRINT@128+P,LEFT$(B$,10);
```

Adesso il nostro castello base e' completo. Possiamo aggiungere le feritoie per gli arcieri usando un carattere deiverso e la sovrapposizione.

```
100 FOR I=2 TO 32 STEP 4 : PRINT@288+I,CHR$(206); : NEXT I
```

Il castello deve avere anche un ingresso, quindi inseriamo delle caselle nere nella posizione giusta.

```
110 G$=CHR$(128)+CHR$(128)+CHR$(128)
120 FOR I=353 TO 417 STEP 32 : PRINT@I+P+5,G$; : NEXT I
```

L'ingresso e' memorizzato in G$ e stampato alla riga 120 (usando il P dell'istruzione PRINT per mantenere il portone allineato con la torre).

Un castello con un ingresso sempre aperto non e' molto utile, quindi ci vuole un cancello verticale. Usando un altro carattere (142+112) possiamo costruire qualcosa di simile.

```
130 P$=CHR$(254)+CHR$(254)+CHR$(254)
140 FOR I=353 TO 417 STEP 32 : PRINT@I+P+5,P$;
150 FOR K=1 TO 300 : NEXT K,I
```

La riga 130 costruisce un cancello arancione che viene visualizzato dall'alto verso il basso nella riga 140. Il loop K ritarda l'operazione in modo che il cancello si "abbassa" lentamente.

Ora lasciamo il castello. Se volete, potete aggiungere altri particolari, per esempio una bandiera sulla torre, un fossato azzurro, ecc.

IMMAGINI MOBILI

Nell'esempio del castello, le righe 140 e 150 ci hanno mostrato come creare il movimento delle immagini, disegnando le parti in successione. Questo tipo di movimento e' limitato alla apertura e chiusura di porte. Un sistema molto piu' valido e' quello di disegnare l'intera immagine, cancellarla e ridisegnarla in una posizione leggermente diversa. Poiche' dovete ridisegnare l'immagine costantemente, la parte che riguarda il disegno dovrebbe essere una subroutine. Per disegnare qualcosa di simile ad una figura umana, costruiamo un blocco 3x4. La fila superiore sara' la testa, quella in mezzo il corpo e l'ultima le gambe.

```
10 CLEAR 500 : CLS0
20 M1$=CHR$(128)+CHR$(193)+CHR$(194)+CHR$(128)
30 M2$=CHR$(196)+CHR$(207)+CHR$(207)+CHR$(200)
40 M3$=CHR$(128)+CHR$(202)+CHR$(197)+CHR$(128)
```

Ora la figura e' memorizzata in tre variabili stringa M1$, M2$ e M3$. Adesso ci serve la subroutine per scrivere le stringhe nell'ordine giusto.

```
500 P=32*Y+X
510 PRINT@P,M1$; : PRINT  P+32,M2$;
520 PRINT@P+64,M3$; : RETURN
```

Questa subroutine disegna la figura su tre righe, una direttamente sotto l'altra, cominciando da un punto determinato da Y e X (vi ricordate il reticolo X,Y ? X va da zero a 31 in senso orizzontale, Y da 0 a 15 in senso verticale). Nell'esempio che segue dovrete inserire tutte le righe fino alla 160 prima di poter far girare il programma.
Adesso per spostare la figura dobbiamo cambiare la posizione, cioe' X o Y. Cio' si puo' fare dalla tastiera. Utilizzeremo INKEY$ per scorrere la tastiera, ed i caratteri piu' indicati da usare sono i tasti con le frecce. Anche a questi tasti, come per le lettere, corrispondono dei codici.

```
[<-]CHR$(8)
[->]CHR$(9)
[ ↓ ]CHR$(10)
[ ↑ ]CHR$(94)
```

Utilizzeremo X per mantenere la posizione orizzontale della figura, e Y per mantenere quella verticale. Se viene premuto il tasto di "backspace" [<-], vogliamo spostarci a sinistra, e quindi sottrarre 1 dal valore attuale di X. Dobbiamo pero' essere sicuri di non uscire dal video.

```
90 GOSUB 500
100 A$=INKEY$ : IF A$=""THEN 100
120 IF A$=CHR$(8) THEN X=X-1 : IF X<0 THEN X=0
```

La riga 100 attende che venga premuto un tasto. Se questo tasto e' [<-], la riga 120 sottrae 1 dal valore di X, controlla se siamo fuori dallo schermo e, in caso affermativo, ferma il movimento sul lato sinistro. Per spostarci a destra la procedura e' simile.

```
130 IF A$=CHR$(9) THEN X=X+1 : IF X>28 THEN X=28
140 IF A$=CHR$(94) THEN Y=Y-1 : IF Y<0 THEN Y=0
150 IF A$=CHR$(10) THEN Y=Y+1 : IF Y>13 THEN Y=13
160 GOSUB 500 : GOTO 100
```

Nella riga 130 fissiamo il valore massimo ammissibile per X a 28. In ogni caso esso non puo' superare 31 (ricordatevi che la nostra figura e' larga 4 blocchi). Lo stesso vale per Y, dobbiamo lasciare spazio per disegnare le tre righe. La riga 160 rimanda alla subroutine di disegno per poi tornare a verificare l'azionamento di un altro tasto. Se adesso fate girare il programma, dovreste riuscire a far muovere la figura attraverso lo schermo., ma non e' bello perche' non abbiamo eliminato la figura dalla posizione precedente. Per fare cio' ci serve una stringa di abblencamento ed una subroutine per disegnarla.

```
50 BL$=CHR$(128)+CHR$(128)+CHR$(128)+CHR$(128)
600 P=32*Y+X : PRINT@P,BL$;
610 PRINT@P+32,BL$; : PRINT@P+64,BL$;
620 RETURN
```

La nuova subroutine si comporta esattamente come l'altra, solo che questa volta disegna un blocco di quadretti neri. Ora possiamo cancellare la figura immediatamente prima di spostarla.

```
110 GOSUB 600
```

A questo punto dovreste essere in grado di spostare la figura dove volete sullo schermo. Nella sua forma attuale, questo programma e' soltanto un esempio, ma figure mobili di questo genere si possono inserire in giochi e programmi educativi per bambini.

UNA NUOVA RISOLUZIONE

►► Passiamo ora al livello di risoluzione successivo. Questo ha un reticolo 32x64, il che significa 2084 punti sullo schermo. Questo livello e quello precedente di 32x16 rappresentano schermi a **bassa risoluzione** e, volendo, possono essere usati assieme. Per indirizzare i punti su questo nuovo schermo ci sono due comandi.

►► SET(X,Y,C) e RESET(X,Y)

Il comando SET accende il punto X,Y nel colore C. La X (fra 0 e 63) e la Y (fra 0 e 31) rappresentano la posizione orizzontale e verticale come prima. La C e' un numero fra 0 e 8 che rappresenta il numero del colore scelto per il punto.

Il comando RESET spegne il punto X,Y. Usando questi comandi in modo alternato si puo' creare del movimento. Provate il seguente programma.

```
10 CLS0 : X1=0 : Y1=0 : XI=2 : YI=2
20 X2=X1+XI : IF X2>63 OR X2<0 THEN XI=-XI :
   SOUND 180,1 : GOTO 20
30 Y2=Y1+YI : IF Y2>31 OR Y2<0 THEN YI=-YI :
   SOUND 180,1 : GOTO 30
40 SET(X2,Y2,8) : RESET(X1,Y1) : X1=X2 : Y1=Y2 : GOTO 20
```

Potete vedere cosa fa, ma come lo fa? Cominciando dal punto X1, Y1, il programma aumenta X1 di una piccola quantita' XI, e Y1 di YI, per creare un nuovo punto, X2,Y2. Nella riga 40 viene visualizzato il nuovo punto e quello vecchio (X1,Y1) scompare. Il punto X2,Y2 diventa quello vecchio ed il programma ritorna alla riga 20 per creare un altro X2,Y2. In questa maniera la "pallina" si sposta attraverso lo schermo. Quando raggiunge il bordo, cambia il segno di incremento. Questo significa che X, per esempio, comincia a diminuire provocando il cambio di direzione. Questo causa il "rimbalzo". Se cambiate la grandezza dell'incremento (XI e YI nella riga 10), potete spostare la pallina a velocita' diverse. Questo tipo di programma e' la base della maggior parte dei giochi con la palla per il computer, ma di solito questi programmi sono scritti in linguaggio macchina, non in BASIC.

Un altro uso dei punti mobili si trova nei giochi di tipo tiro a segno. Tali giochi richiedono la possibilita' di spostarsi sullo schermo e di "sparare". Potremo utilizzare i tasti con le frecce come prima, ma un sistema molto migliore e' quello di utilizzare i joysticks.

►►I joysticks si collegano al computer, nelle apposite prese sul fianco, e permettono un controllo molto piu' preciso del movimento che non i tasti con le frecce. La posizione del joystick viene letta dalla funzione JOYSTK. JOYSTK(0) restituisce la posizione orizzontale del joystick di destra, e JOYSTK(1) la posizione verticale. JOYSTK(2) e JOYSTK(3) fanno la stessa cosa per il joystick di sinistra. Poiche' il valore restituito dalla funzione in ogni caso varia fra 0 e 63, il valore dovra' essere dimensionato in base alla risoluzione dello schermo con la quale state lavorando.

SET

Il comando SET e' usato per fissare un punto specifico sullo schermo a bassa risoluzione in un colore specifico.

SET(x,y,c)

x e y sono le coordinate del punto dello schermo. x deve essere compreso fra 0 e 63 ed y fra 0 e 31.

c e' il codice del colore desiderato. Deve essere fra 0 e 8.

```
10 CLS0 : SET(5,27,8) : SET(6,27,8)
20 FOR X=0 TO 6 : FOR Y=28 TO 30
30 SET(X,Y,8) : NEXT Y,X
40 FOR X=7 TO 63 : FOR D=1 TO 200 : NEXT D
50 FOR Y=27 TO 30 : IF Y=27 THEN RESET(X-2,Y)
60 SET(X,Y,8) : RESET(X-7,Y) : NEXT Y,X
70 GOTO 70
```

RESET

Il comando RESET viene usato per cancellare un punto stabilito dal comando SET. Si usa nel modo grafico a bassa risoluzione.

RESET(x,y)

x,y sono le coordinate del punto da cancellare. x deve essere compreso fra 0 e 63, e y fra 0 e 31

Il punto viene fissato nel colore del fondo e quindi e' cancellato. Vedete SET per un esempio.

```
10 CLS0 : FOR I=0 TO 3
20 PRINT@74+32*I,"JOYSTK(";I;")  ";JOYSTK(I)
30 NEXT I : FOR D=0 TO 400 : NEXT D : GOTO 10
```

Eseguite questo programma e spostate i joysticks. Vedrete come i valori cambiano con la posizione dei joysticks. Potete adoperare anche il pulsante sul joystick. Aggiungete la riga

```
25 P=PEEK(65280) : PRINT@202,"VALORE DEL PULSANTE  ";P;
```

► La funzione PEEK dice al computer di guardare una specifica locazione della memoria. L'indirizzo di memoria 65280 contiene il risultato della verifica del pulsante. In questo momento sara' o 127 o 255. Se premete il pulsante di **sinistra**, diventera' 125 o 253, se premete quello di **destra**, diventera' 126 o 254. (Premendoli contemporaneamente otterrete 124 o 252).

Adesso cominceremo a lavorare su di un programma di gioco - una battaglia spaziale tra due astronavi. Possiamo usare i joysticks per spostare le astronavi ed il pulsante per sparare.

Per prima cosa pensiamo alle astronavi. Utilizzeremo un metodo simile a quello usato per costruire la figura nell'esempio precedente. Ogni astronave sara' costituita da un blocco 2x3, uno giallo e l'altro blu, memorizzati nelle matrici S$ ed S2$.

```
10 CLEAR 500 : FOR I=0 TO 5 : READ S(I) : NEXT I
20 DATA 128,131,128,134,140,137
30 FOR Y=0 TO 1 : C=(Y+1)*16
40 S$(Y)=CHR$(S(0)+C)+CHR$(S(1)+C)+CHR$(S(2)+C)
50 S2$(Y)=CHR$(S(3)+C)+CHR$(S(4)+C)+CHR$(S(5)+C)
60 NEXT Y
```

Se non vi piace la forma delle astronavi, progettate la vostra e cambiate i dati nella riga 20.

Successivamente dobbiamo leggere la posizione dei joysticks, controllare se siamo ancora sullo schermo e calcolare il punto in cui disegnare le astronavi.

```
70 FOR Y=0 TO 1 : A(Y)=JOYSTK(Y*2)
80 B(Y)=INT(JOYSTK(1+Y*2)/2)
85 IF A(Y)>58 THEN A(Y)=58
90 IF A(Y)<2 THEN A(Y)=2
100 IF B(Y)>27 THEN B(Y)=27
110 L(Y)=INT(B(Y)/2)*32+INT(A(Y)/2) : NEXT Y
```

Le posizioni dei joysticks sono lette, una alla volta, alla riga 70. I limiti sono stabiliti nelle righe 80-100 (ricordatevi le dimensioni delle astronavi).

Il risultato finale viene poi trasformato in un valore per il comando PRINT@. (Questo e' un esempio di uso contemporaneo dei due schermi a bassa risoluzione - i joysticks operano in uno ed il comando PRINT@ nell'altro).

Ora dobbiamo disegnare le astronavi e tornare a vedere se i joysticks sono stati spostati.

```
120 CLS0 : FOR Y=0 TO 1 : PRINT@0,Z(0); : PRINT@26,Z(1);
130 PRINT@L(Y),S$(Y); : PRINT@L(Y)+32,S2$(Y); : NEXT Y
170 A$=INKEY$ : IF A$=""THEN 70
180 CLS : END
```

La riga 120 scrive il punteggio (ma non siamo ancora arrivati a cio'). Per terminare il gioco, premete qualunque tasto, altrimenti il programma va alla riga 70 e rilegge la posizione dei joysticks.

Fate girare il programma fino a questo punto e controllate che le astronavi si spostino su tutto lo schermo.

Il passo successivo e' quello di azionare i cannoni e visualizzare i missili al plasma. Questa e' la parte piu' difficile perche' dobbiamo controllare i pulsanti dei joysticks e determinare chi sta sparando. Inoltre, poiche' le astronavi possono andare in ogni direzione, dobbiamo conoscere la direzione dello sparo. Per semplificare le cose, permetteremo che il missile si possa spostare dalla nave da cui parte verso il bersaglio, solamente lungo una linea orizzontale. Esso viaggera' all'altezza dell'astronave che sta sparando.

```
140 P=PEEK(65280)
150 IF P=125 OR P=253 THEN F=0 : T=1 : GOSUB 200
160 IF P=126 OR P=254 THEN F=1 : T=0 : GOSUB 200
```

Queste righe controllano i pulsanti e decidono quale astronave sta sparando. Il missile verra' visualizzato dalla subroutine 200.

```
200 V1=B(F) : H1=A(F) : H2=A(T) : ST=1
210 IF H1>H2 THEN ST=-1
220 FOR H=H1+ST*5 TO H2+2 STEP ST
240 SET(H,V1,4) : SOUND 200,1 : RESET(H-2*ST,V1)
250 NEXT H : RETURN
```

Notate lo scambio dello step alla riga 210, se le posizioni sinistra e destra sono invertite. Il movimento del missile e' realizzato alla riga 240.

A questo punto non vi resta che controllare se avete "colpito". In caso affermativo, fatevi i complimenti dovuti e registrate il punteggio.

La funzione POINT si usa per verificare un colpo. La sua forma e' POINT(X,Y), dove X,Y e' il punto che volete verificare. La funzione restituisce 0 se il punto e' nero, altrimenti da' il numero di codice del colore.

Poiche' lo schermo e' nero e miriamo nella direzione giusta (speriamo), ci basta sapere se c'e' un punto colorato acceso lungo la traiettoria. Se aggiungiamo alla subroutine la seguente riga,

```
230 IF POINT(H,V1)>0 THEN GOSUB 300 : RETURN
```

in caso di colpo riuscito, il programma andra' alla subroutine 300 (dove, tra l'altro, calcoliamo il punteggio). Al ritorno dalla subroutine 300, non ha piu' senso sparare un altro colpo e quindi il programma lascia la subroutine 200 e ricomincia.

```
300 Z(F)=Z(F)+1
310 FOR K=1 TO 15 : I=RND(5)-2 : J=RND(4)-2
320 SET(H+I*ST,V1+J,8) : SOUND(RND(95)),1
330 NEXT K : RETURN
```

Questa subroutine calcola il punteggio, disegna l'esplosione e crea gli effetti sonori.

Per quanto sia lungo soltanto 28 righe, questo programma e' sufficiente per produrre un gioco abbastanza movimentato. Lasciamo a voi, come esercizio, affinarlo a tal punto da portarlo a livello di gioco commerciale.

Questo capitolo e' stato lungo e complesso, ma contiene quasi tutti gli elementi che vi occorrono per fare della grafica con il vostro computer, indipendentemente dal livello di risoluzione che state adoperando.

CAPITOLO VIII

LA GRAFICA PIU' AVANZATA

Passiamo adesso alla grafica ad alta risoluzione che e' qualcosa di completamente distinto rispetto a quella a bassa risoluzione. I due livelli di grafica a bassa risoluzione possono essere utilizzati assieme e vengono visualizzati su quello che viene definito "text screen". La grafica ad alta risoluzione, invece, non puo' essere mescolata con il testo. Si puo' passare dall'una all'altro, ma non si puo' scrivere un testo sullo schermo ad alta risoluzione, ne' disegnare grafica ad alta risoluzione sul "text screen".

Ogni volta che si traccia qualcosa in alta risoluzione, il computer scrive le istruzioni riguardanti il modo di visualizzare le informazioni in una parte della memoria che si chiama "video RAM". La video RAM viene interpretata per la TV e trasformata in immagini. Alcune "pagine" (normalmente quattro) vengono riservate allo scopo nella video RAM. Man mano che aumenta il dettaglio del disegno, aumenta anche il numero di istruzioni necessarie per visualizzare il risultato. Piu' istruzioni richiedono piu' spazio e quindi dovrete riservare piu' pagine. Cio' si ottiene con il comando PCLEAR, seguito dal numero di pagine che desiderate riservare (massimo 8),

►► PCLEAR 8

Poiche' ogni pagina occupa 1536 locazioni di memoria, riservate soltanto quanto realmente serve. Lo spazio disponibile in memoria e' fisso, quindi piu' spazio riservate per le pagine grafiche, meno spazio vi rimane per i programmi. PCLEAR si comporta in modo analogo a CLEAR e dovrebbe essere usato all'inizio del programma.

LA SCELTA DEL MODO

Lo spazio che dovrete riservare dipende dal livello di risoluzione che desiderate usare. L'alta risoluzione ha lo svantaggio che non e' possibile adoperare tutta la gamma di colori disponibili in bassa risoluzione. I colori disponibili e la risoluzione sono determinati dal modo nel quale lavorate. Il modo viene stabilito con il comando PMODE.

►► PMODE **modo, pagina iniziale**

dove **modo** e' un numero fra 0 e 4 e **pagina iniziale** e' la "pagina" della video RAM sulla quale volete disegnare. Come prima, il video e' suddiviso in reticoli. Questa volta, pero', basta ricordarsi un'unica dimensione (256x192). Anche se la risoluzione cambia con i diversi modi, vi riferite sempre a punti sullo schermo in un reticolo 256x192.

PCLEAR

PCLEAR e' usato per riservare pagine grafiche per i modi ad alta risoluzione.

►► PCLEAR n

n deve essere un numero fra 1 e 8. Se si omette l'istruzione PCLEAR, si assume PCLEAR 4.

Poiche' ogni pagina grafica richiede 1536 bytes di memoria, riservate soltanto lo spazio che vi serve.

Numero PMODE	Dimensioni reticolo	Dimens. punto	Pagine usate	COMBINAZIONE DI COLORI DISPONIBILI	
				SCREEN 1,0	SCREEN 1,1
0	128X96		1	Nero(0), Verde(1)	Nero(0), Beige(5)
1	128x96		2	Verde(1), Giallo(2), Blu(3), Rosso(4)	Beige(5), Azzurro(6), Magenta(7), Arancio(8)
2	192x128		2	Nero(0), Verde(1)	Nero(0), Beige(5)
3	192x128		4	Verde(1), Giallo(2), Blu(3), Rosso(4)	Beige(5), Azzurro(6), Magenta(7), Arancio(8)
4	256x192		4	Nero(0), Verde(1)	Nero(0), Beige(5)

SCREEN

►► Il comando SCREEN viene usato per cambiare il video dal modo grafico al modo testo e viceversa.

SCREEN **tipo, combinazione di colori**

tipo puo' essere 0 per testo e grafica a bassa risoluzione o 1 per la grafica ad alta risoluzione.

La **combinazione di colori** e' 0 o 1. La combinazione per il testo e' 0, nero su verde, o 1, nero su arancione. Per la grafica ad alta risoluzione, la **combinazione di colori** disponibile dipende dal modo in cui si lavora come segue:

PMODE	SCREEN 1,0	SCREEN 1,1
0	Nero, Verde	Nero, Beige
1	Verde, Giallo, Blu, Rosso	Beige, Verde, Magenta, Arancio
2	Nero, Verde	Nero, Beige
3	Verde, Giallo, Blu, Rosso	Beige, Azzurro, Magenta, Arancio
4	Nero, Verde	Nero, Beige

La differenza sta nella grandezza del punto disegnato. Il modo scelto determina anche quali colori si possono utilizzare. Ogni modo ha due combinazioni di colori disponibili. La combinazione di colori viene scelta con il comando SCREEN, che determina anche il tipo di video.

►► SCREEN **tipo, combinazione di colori**

Il **tipo** e' 0 per il video testo e 1 per il video di alta risoluzione. La **combinazione di colori** e' pure 0 o 1. Il default che abbiamo usato fino a questo punto e' SCREEN 0,0. Questo fissa il video testo in nero su sfondo verde. (E' possibile usare SCREEN 0,1, che da' un testo nero su sfondo arancione, ma ogni volta che e' il computer a scrivere tornera' nero su verde). Per visualizzare il video di alta risoluzione bisogna stabilire il **tipo** a 1. La tabella della pagina precedente presenta i modi e le combinazioni di colori. Avrete anche notato che, al variare del modo da 0 a 4, aumenta il numero di pagine necessarie.

Quindi, visualizzare un video grafico PMODE 0 richiede una sola pagina di memoria, mentre PMODE 3 e PMODE 4 ne richiedono 4. Una volta scelta la combinazione di colori, il computer sceglie il colore di questa combinazione che ha il numero piu' basso come colore del fondo. Il colore dal numero piu' alto viene usato per il primo piano. Per esempio, con PMODE 3 e SCREEN 1,0, il computer disegnera' in rosso su sfondo verde. Potete cambiare i colori del fondo e del primo piano con il comando COLOR,

►► COLOR **primo piano, fondo**

dove **primo piano** e **fondo** sono i codici dei colori desiderati per quel determinato modo.

VECCHI AMICI

Ricorderete i comandi CLS, SET, RESET e POINT della grafica a bassa risoluzione. Comandi equivalenti sono disponibili anche per l'alta risoluzione. Essi si chiamano PCLS, PSET, PRESET e PPOINT per indicare il loro nuovo stato. Hanno la stessa funzione di prima, PCLS pulisce il video ad alta risoluzione, e se seguito da un codice di colore stabilisce il colore del fondo. PSET accende un puntino e PRESET lo spegne. PPOINT controlla se il punto e' acceso o spento. Il seguente esempio scorre ogni modo e combinazione di colore, uno per uno.

Vengono accesi dei puntini sul video in un colore a caso; i puntini dovrebbero essere contenuti in un reticolo rettangolare. Gli spazi vuoti nel reticolo sono dovuti al fatto che il colore a caso e' uguale al colore del fondo, oppure non e' disponibile nella combinazione di colori.

```
10 FOR P=0 TO 4 : PMODE P,1
20 FOR S=0 TO 1 : SCREEN 1,S
30 PCLS : FOR I=50 TO 150 STEP 20
40 FOR J=50 TO 150 STEP 20
50 C=RND(8) : PSET(I,J,C) : NEXT J,I
60 FOR D=1 TO 1000 : NEXT D,S,P
```

Osservate attentamente la grandezza del punto, questa e' la risoluzione disponibile in quel modo.

TRACCIAMO UNA RIGA, DA QUALCHE PARTE

Possiamo disegnare dei punti sul video, e poi? La cosa piu' ovvia e' collegare i punti per formare una riga. Fortunatamente esiste un apposito comando, LINE. Cancellate le righe 40 e 50 dal precedente esempio e cambiate la riga 30 in,

```
30 PCLS : LINE(10,180)-(245,10), PSET
```

e fate girare il programma. Viene disegnata una riga che va dal basso a sinistra all'alto a destra. Il significato dell'istruzione e' il seguente: traccia una riga dal punto iniziale (10,180) al punto finale (245,10) nel colore del primo piano (PSET). Se cambiate PSET in PRESET la riga verra' disegnata nel colore del fondo. Se disegnate usando il colore del fondo, non si vede niente. Questo puo' essere usato anche per cancellare una riga disegnata precedentemente. PSET e PRESET sono parti essenziali del comando LINE, e non hanno in questo caso alcuna relazione con gli omonimi comandi usati per accendere o spegnere punti.

Non e' sempre necessario specificare il punto iniziale di LINE. Senza alcun punto iniziale, la riga iniziera' dal punto finale piu' recente. (Se l'istruzione LINE non e' ancora stata usata nel programma, si assume 128,96, il centro del video, come l'ultimo punto finale). Aggiungete un'altra riga al nostro esempio.

```
40 LINE-(130,180), PSET
```

Adesso viene tracciata una riga dall'ultimo punto finale (245,10) fino ad un punto in fondo al video (130,180).

COLOR

Il comando COLOR e' usato per cambiare i settaggi di default (cioe' quelli che assume il computer) dei colori per il fondo e per il primo piano nei modi grafici ad alta risoluzione.

►► COLOR **primo piano, fondo**

Sia **primo piano** che **fondo** sono dei numeri fra 0 e 8, che rappresentano il codice di colore. Entrambi i colori devono essere compresi nella combinazione di colori disponibili per il modo attuale.

PCLS

Il comando PCLS e' usato per pulire il video e settare il colore di fondo prescelto nella grafica ad alta risoluzione.

►► PCLS **c**

dove **c** e' il codice del colore del fondo desiderato. Esso deve essere un colore della combinazione disponibile per il modo di lavoro. Se il colore non e' disponibile o se si omette **c**, si assume il colore di default.

Per i codici di colore, vedete la cornice per CLS.

PSET

Questa e' la versione per l'alta risoluzione del comando SET.

►► PSET(x,y,c)

accende il punto (x,y) nel colore **c**. x deve essere fra 0 e 255 ed y fra 0 e 191.

c e' il codice di colore fra 0 e 8 e deve essere un colore compreso nella combinazione disponibile.

PRESET

Questa e' la versione in alta risoluzione del comando RESET.

►► PRESET(x,y)

spegne il punto (x,y). (Cioe' lo mette nel colore di fondo). x deve essere fra 0 e 255, y fra 0 e 191.

Per disegnare un quadrato o un rettangolo, potreste utilizzare quattro righe, ma c'e' un'estensione del comando LINE che si prende cura di questo. Adoperate l'EDITOR per aggiungere ,B alla riga 30, che ora dovrebbe essere:

```
30 PCLS : LINE(10,180)-(245,10), PSET, B
```

Adesso, invece di una riga diagonale avrete un rettangolo. Per disegnare un rettangolo, basta specificare la posizione dei due angoli diametralmente opposti ed aggiungere B alla istruzione LINE. Ritornate nel modo EDITOR ed aggiungete F in fondo alla riga 30.

```
30 PCLS : LINE(10,180)-(245,10), PSET, BF
```

La F in piu' dice al computer di riempire il rettangolo con il colore del primo piano. Un comando cosi' flessibile deve avere qualche utilita', quindi disegnamo un'immagine.
Cominceremo, come la prima volta, nel campo dell'edilizia, ma questa volta costruiamo una casa. Fate girare il programma dopo l'inserimento di ogni singola sezione per vedere come si sviluppa l'immagine.
Prima stabiliamo la risoluzione e disegnamo il corpo della casa.

```
10 PMODE 3,1 : SCREEN 1,0 : PCLS
20 LINE(60,48)-(200,144), PSET, B
260 GOTO 260
```

Poi aggiungiamo il tetto.

```
40 LINE(60,48)-(130,20), PSET
50 LINE-(200,48), PSET
```

Aggiungiamo anche un garage, con una porta.

```
70 LINE(200,144)-(255,94), PSET, B
90 LINE(210,144)-(245,104), PSET, BF
```

Possiamo adoperare la stessa tecnica per la porta della casa.

```
100 LINE(160,144)-(188,105), PSET, BF
```

Per disegnare le finestre ci serve un rettangolo con due righe come traversine.

```
110 LINE(85,132)-(135,108), PSET, B
120 LINE(110,108)-(110,132), PSET
130 LINE(85,120)-(135,120), PSET
```

Analogamente facciamo le finestre del piano superiore.

```
140 LINE(90,84)-(125,64), PSET, B
150 LINE(90,74)-(125,74), PSET
160 LINE(110,84)-(110,64), PSET
```

LINE

Il comando LINE viene usato per disegnare righe e rettangoli nella grafica ad alta risoluzione.

►►LINE(x1,y1)-(x2,y2), a, b

x1,y1 sono le coordinate del punto iniziale della riga.
x2,y2 sono le coordinate del punto finale della riga.
a e' PSET o PRESET. Se si utilizza PSET, la riga viene disegnata nel colore del primo piano. Se invece si usa PRESET, la riga viene disegnata nel colore del fondo.
b e' un parametro opzionale. Se viene usato puo' essere B o BF. Se e' B, viene disegnato un rettangolo invece di una riga, lo spigolo in alto a sinistra del rettangolo sara' x1,y1, e lo spigolo in basso a destra sara' x2,y2. Se si usa BF viene disegnato un rettangolo e riempito con il colore del primo piano.

```
10 PMODE 4,1 : SCREEN 1,1 : PCLS 5 : COLOR 0,5
20 FOR I=1 TO 10000
30 X=X+L*SIN(R) : Y=Y+L*COS(R)
40 IF X<-128 OR X>128 THEN 90
50 IF Y<-96 OR Y>95 THEN 90
60 LINE -(X+128,Y+96), PSET
70 R1=R1+60 : R=R1/57.29578 : L=L+0.5
80 NEXT I
90 GOTO 90
```

```
170 LINE(155,64)-(175,84), PSET, B
180 LINE(165,84)-(165,64), PSET
190 LINE(155,74)-(175,74), PSET
```

e completiamo la nostra costruzione con un camino,

```
200 LINE(150,40)-(160,15), PSET, BF
```

Questo piccolo programma dimostra quanto velocemente si puo' costruire un'immagine, utilizzando un solo comando.
Naturalmente, si assume che sappiate dove tracciare le righe. Il sistema piu' facile per trovare i punti e' quello di prendere una copia del reticolo per la grafica riportato all'Appendice B, fare lo schizzo e leggere i punti dallo schizzo.

UNA MACCHIA DI COLORE

La nostra casa e' un po' scialba. Ci vuole una mano di vernice per ravvivarla. Cosi' diciamo al DRAGON di tirare fuori i pennelli e mettersi al lavoro. Il comando PAINT vi permette di dipingere qualsiasi forma usando qualunque colore disponibile. Tutto cio' che dovete fare e' dirgli dove iniziare, quale colore deve adoperare e il colore del bordo che segna il limite della zona da dipingere.

PAINT(x,y), a, b

dove x,y sono le coordinate dell'inizio, e a, b sono i codici dei colori della cornice e del bordo. Aggiungete la seguente riga al programma della casa,
30 PAINT(90,90), 2, 4

Questo significa dipingere di giallo (colore 2) fino ad incontrare il bordo rosso (colore 4), iniziando dal punto (90,90). Fate girare il programma per vedere che cosa fa. Ora cancellate la riga, inseritela di nuovo come riga 195 ed eseguite il programma ancora una volta.
195 PAINT(90,90), 2, 4

Notate come ora il colore si ferma ai bordi delle finestre che prima non c'erano. Colorate il garage nella stessa maniera.

```
80 PAINT(210,140), 2, 4
```

Ora il tetto. Se omettete di introdurre un colore od un bordo in una istruzione PAINT, viene scelto per entrambi l'attuale colore del primo piano.
60 PAINT(130,25)

Concludiamo disegnando l'orizzonte ed il cielo

```
210 LINE(0,64)-(60,64), PSET
220 LINE(200,64)-(255,64), PSET
230 PAINT(0,54), 3, 4
```

PAINT

Il comando PAINT e' usato nei modi grafici ad alta risoluzione per riempire una figura di colore.

►► PAINT(x,y), c, b

x,y sono le coordinate del punto da dove si vuole iniziare a colorare.
c e' il codice del colore da usare per dipingere. Esso deve essere fra 0 e 8 e trovarsi nella combinazione di colori disponibili per il modo che si sta usando. Se questo codice e' omesso, si usa il colore attuale del primo piano.

b e' il codice del colore del bordo che delimita la figura. Anch'esso deve essere fra 0 e 8. La pittura passera' sopra un bordo di qualsiasi altro colore. Se esso e' omesso, viene assunto il colore attuale del primo piano.

Vedete CIRCLE per avere un esempio di uso.

Ora la casa e' un po' piu' allegra. Forse vorrete continuare a migliorarla. Aggiungete un sentiero e un recinto. Oppure dedicate un po' di tempo ad esercitarvi con la grafica, disegnando delle figure a vostra scelta e colorandole per vedere che cosa succede.

DISEGNIAMO CERCHI

Abbiamo visto righe, quadrati e rettangoli, ed ora vediamo i cerchi. L'istruzione CIRCLE disegna cerchi, ellissi ed archi.

►►**CIRCLE(x,y), raggio, colore, hwratio, inizio, fine**

Il punto **x,y** e' il centro del cerchio, il **raggio** e' il raggio del cerchio misurato in punti video. Il **colore** e' uno dei colori disponibili nel modo in cui lavorate (se e' omesso viene assunto il colore del primo piano). Gli altri parametri servono per disegnare ellissi ed archi; li tratteremo piu' avanti. Prima vediamo cosa succede con i cerchi.

```
10 FOR P=0 TO 4 : PMODE P,1
20 SCREEN 1,1 : PCLS
30 FOR R=120 TO 10 STEP-10
40 CIRCLE(128,96), R : NEXT R
50 FOR D=1 TO 500 : NEXT D,P
```

Questo programma disegna una serie di cerchi al centro del video. I cerchi sono difficili da disegnare e per ottenere un risultato molto preciso probabilmente dovrete adoperare PMODE 4.

Notate che non c'e' nessun problema se un cerchio va al di la' del bordo del video. Se tentate di disegnare una riga fino ad un punto non esistente sul video, non si disegna per niente, specialmente nei modi ad alta risoluzione. Provate ad inserire

```
42 LINE-(300,40), PSET
```

ed osservate il risultato

Si puo' usare il comando PAINT anche per colorare i cerchi,

```
45 PAINT(128,96)
```

riempie il centro del bersaglio.

Con il parametro **hwratio** potete cambiare il cerchio in una ellisse. **hwratio** significa rapporto altezza/larghezza. Nel comando CIRCLE, la larghezza rimane costante, due volte il raggio. Con **hwratio** si puo' variare l'altezza; se e' maggiore di 1 il "cerchio" sara' piu' alto che largo. Un valore minore di 1 stringe il cerchio nell'altro verso, piu' largo che alto.

►►CIRCLE

Il comando CIRCLE permette di disegnare cerchi, ellissi ed archi. Esso puo' essere usato soltanto nei modi grafici ad alta risoluzione.

CIRCLE(x,y), r, c, hw, inizio, fine

x	e' la coordinata x del centro del cerchio (fra 0 e 255)
y	e' la coordinata y del centro del cerchio (fra 0 e 191)
r	e' il raggio del cerchio, misurato in punti video
c	e' un codice di colore (fra 0 e 8), deve essere uno di quelli della combinazione disponibile. Se esso e' omesso viene usato il colore del primo piano.
hw	e' il rapporto altezza-larghezza (fra 0 e 255), usato per disegnare ellissi. Se e' omesso, si assume 1.
inizio	e' l'inizio dell'arco del cerchio (da 0 a 1). La posizione 0 rappresenta le ore 3. Se omesso, si assume 0.
fine	e' la fine dell'arco (da 0 a 1). Il disegno procede in senso orario partendo dall'**inizio**. La posizione 0.5 rappresenta le ore 9. Se omesso si assume 1.

```
10 PMODE 3,1 : SCREEN 1,0 : PCLS
20 CIRCLE(180,156),28,3 : PAINT(180,156),3,3
30 CIRCLE(110,156),28,3 : PAINT(110,156),3,3
40 CIRCLE(144,80),68,4,1,0,.5
50 LINE(212,80)-(76,80),PSET : LINE-(48,32),PSET
60 PAINT(144,82) : CIRCLE(144,80),70,4,.8,.79,1
70 LINE(160,80)-(160,28),PSET : PAINT(210,75)
80 GOTO 80
```

La larghezza lungo l'asse X (orizzontale) e' quindi sempre la stessa, si modifica soltanto l'altezza lungo l'asse Y (verticale). Quando **hwratio** e' 0 il "cerchio" e' una linea orizzontale, e quando invece **hwratio** e' molto grande, esso diventa quasi una riga verticale (in realta' un rettangolo lungo e stretto). Il valore piu' grande ammesso e' 255. Cambiate le righe 30 e 40 nel nostro esempio come segue:

```
30 FOR H=0.5 TO 3 STEP 0.5
40 CIRCLE(128,96),40,,H : NEXT H
```

Le virgole in piu' che vedete nella riga 40 servono perche' abbiamo tralasciato il parametro del **colore**.

L'ultima estensione del comando CIRCLE offre la possibilita' di disegnare archi (parti di cerchio). Per usare questa opzione dovete specificare l'inizio e la fine dell'arco. Entrambi questi valori devono essere un numero fra 0 e 1. Il punto d'inizio di un cerchio e' equivalente alla posizione delle ore 3 sull'orologio. Il disegno procede in senso orario. Per esempio, un inizio 0.25 e fine 0.75 disegna dalle 6 alle 12, cioe' la meta' sinistra del cerchio. Per disegnare la meta' superiore, cominciate a 0.5 e terminate a 1.0. Il seguente programma utilizza degli archi per fare un disegno.

```
10 PMODE 4,1 : SCREEN 1,1 : COLOR 0,5 : PCLS
20 FOR R=15 TO 60 STEP 5
30 CIRCLE(128,96+R),R,,1,.5,1
40 CIRCLE(128,96-R),R,,1,0,.5
50 CIRCLE(128-R,96),R,,1,.75,.25
60 CIRCLE(128+R,96),R,,1,.25,.75
70 FOR D=1 TO 500 : NEXT D
80 NEXT R
90 GOTO 90
```

PASSIAMO DA UNA PAGINA ALL'ALTRA

Un sistema per animare disegni e' quello di mettere su ogni pagina un disegno leggermente diverso e sfogliare le pagine velocemente. Ricordatevi che il numero di pagine e' stato fissato con il comando PCLEAR, ed il secondo parametro di PMODE determina la pagina sulla quale scrivete. Naturalmente dovete tener conto della risoluzione con la quale lavorate. In PMODE 3 e PMODE 4 ogni video grafico richiede 4 pagine, quindi ha senso passare soltanto dalla pagina 1 alla pagina 5. In PMODE 1 e PMODE 2, che richiedono 2 pagine, passate dalla 1 alla 3, 5, 7, ecc. Il seguente esempio dimostra come si fa. Provate ad inserire tutti i valori di PMODE.

```
10 PCLEAR 8 : PCLS
20 INPUT"MODO";M : ON M GOTO 40,40,50,50
30 S=1 : GOTO 60
40 S=2 : GOTO 60
50 S=4
60 FOR P=1 TO 8 STEP S : PMODE M,P : PCLS
70 LINE(128,0)-(128,(P-1)*15), PSET
80 CIRCLE(128,P*15),20 : PAINT(128,P*15) : NEXT P
90 FOR P=1 TO 8 STEP S : GOSUB 150 : NEXT P
100 IF M>2 THEN D=4 : S1=3 ELSE D=7 : S1=S
110 FOR P=D TO 1 STEP -S1 : GOSUB 150 : NEXT P
120 GOTO 90
150 PMODE M,P : SCREEN 1,1
160 FOR T=1 TO 20 : NEXT T : RETURN
```

Le righe da 60 a 80 disegnano la figura nelle sue variazioni sulle diverse pagine. Tutto cio' accade senza essere visualizzato perche' non e' stato dato nessun comando SCREEN. Il resto del programma visualizza ogni pagina una per una, passando prima in avanti e poi indietro per dare il senso del movimento. Come vedete, piu' pagine utilizzate, piu' scorrevole e' il movimento.

Un altro sistema per costruire immagini mobili e' quello di usare il comando PCOPY,

►► PCOPY **pagina d'origine** TO **pagina di destinazione**

Potete copiare il contenuto di qualsiasi pagina su qualsiasi altra pagina, purche' precedentemente si sia riservata la pagina con PCLEAR. PCOPY puo' essere usato anche per immagazzinare duplicati su una pagina PMODE 3 o PMODE 4. Il seguente programma dimostra come si usa PCOPY a questo scopo. Notate con quale cura bisogna collocare la figura.

```
10 PCLEAR 8 : PMODE 3,4 : PCLS
20 LINE(100,20)-(140,40), PSET, BF
30 CIRCLE(50,25), 20
40 CIRCLE(200,50), 20
50 FOR D=3 TO 1 STEP -1
60 PCOPY 4 TO D : NEXT D
70 FOR P=4 TO 1 STEP -1 : PMODE 3,P
80 SCREEN 1,1 : FOR I=1 TO 1000 : NEXT I,P
90 GOTO 90
```

In PMODE 3 (e 4) la visualizzazione comprende 4 pagine, essendo la prima il quarto piu' alto del video, la seconda quello successivo, e cosi' via.

Quindi se copiate il contenuto della pagina 4 alla pagina 1, duplicherete il quarto superiore del video sulla parte inferiore. Con PMODE 1 e PMODE 2 si puo' ottenere lo stesso effetto, solo che il video sara' suddiviso in due e non in quattro parti.

Coloro che vogliono procedere con la grafica possono passare direttamente al capitolo 10, per gli altri, ora ci sara' un breve intervallo musicale.

PCOPY

►► PCOPY e' un comando per la grafica ad alta risoluzione che si usa per copiare il contenuto di una pagina grafica su di un'altra pagina.

PCOPY **origine** TO **destinazione**

origine e **destinazione** devono essere numeri fra 1 e 8 e devono riferirsi a pagine precedentemente riservate con il comando PCLEAR. Lo spazio necessario per uno schermo grafico varia in funzione del modo e dovrebbe essere considerato quando si usa PCOPY.

```
PCOPY 3 TO 5
```

CAPITOLO IX

SUONI ELETTRONICI

COME AGGIUNGERE UNA COLONNA SONORA

►► La grafica (e spesso altri programmi) puo' essere resa piu' interessante con l'aggiunta del suono. Abbiamo gia' usato il comando SOUND a questo fine, soprattutto nell'esempio del capitolo VII. Un modo piu' semplice di aggiungere il suono e' quello di fornirlo voi stessi. Non intendiamo che accompagniate i vostri programmi cantando - almeno non esattamente. Il computer fa uso di un registratore a cassette per memorizzare i programmi e puo' anche far girare un nastro su richiesta. I comandi MOTOR ON e MOTOR OFF servono proprio a questo, insieme ai comandi AUDIO ON e AUDIO OFF che collegano o interrompono l'uscita della cassetta all'altoparlante TV. Cio' significa che la vostra grafica puo' essere accompagnata da una musica di fondo se inserite queste istruzioni nei vostri programmi.
Analogamente, su un piano piu' serio, un nastro preparato anticipatamente potrebbe dare delle istruzioni e fare delle domande in un programma educativo. Il seguente esempio mostra quanto e' semplice. Se non avete un nastro a disposizione, usate uno dei vostri nastri per programmi. Gli strani suoni che sentirete sono un esempio di come i computers parlano tra loro.

```
10 CLS : PRINT@135,"PREMI LA BARRA SPAZIATRICE"
20 PRINT@195,"PER FERMARE O AVVIARE IL REGISTRATORE"
30 A$=INKEY$ : IF A$<>" " THEN 30
40 IF F=0 THEN MOTOR ON : AUDIO ON : F=1 : ELSE
   MOTOR OFF : AUDIO OFF : F=0
50 GOTO 30
```

Riavvolgete il nastro fino all'inizio e premete il tasto PLAY, poi fate girare il programma. Premendo la barra spaziatrice potrete fermare o avviare il playback del nastro.

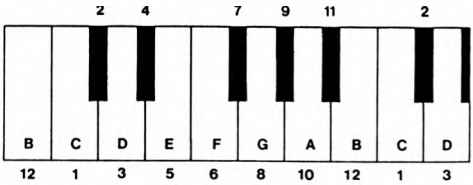
Con questo metodo si potrebbe scrivere un programma educativo del tipo domanda e risposta adatto a diverse materie, basterebbe cambiare il nastro delle domande. Analogamente, potreste fornire ai vostri cartoni animati una musica idonea o degli effetti sonori.

SUONIAMO !

Come alternativa, potete far suonare la musica dal computer. Il comando PLAY trasforma in suoni il contenuto di una stringa.

►► PLAY **stringa**

dove **stringa** puo' essere una costante stringa o una variabile stringa. Comunque essa non puo' essere una stringa qualsiasi, ma deve essere una stringa musicale composta da **nota, ottava, lunghezza nota, tempo** e **pausa.** La **nota** e' ovviamente la nota musicale che volete suonare. Il modo piu' semplice di fare cio' e' quello di introdurre la lettera che rappresenta una delle note musicali standard A (la), B (si), C (do), D (re), E (mi), F (fa), G (sol). Per indicare un diesis usate # o + (F# o F+ per fa diesis) e per un bemolle - (B- per si bemolle). Il computer non riconosce B# o C- in quanto non esistono nella scala dodecafonica. Un altro sistema per introdurre una nota e' quello di usare il numero che rappresenta la sua posizione nella scala dodecafonica.



Le note ed i loro numeri corrispondenti sono indicati nella tastiera riportata qui sopra.

Il comando PLAY puo' essere utilizzato come un comando diretto, il che e' utile per controllare una stringa musicale prima di inserirla nel programma. Come tutti i bravi musicisti cominceremo esercitandoci con le scale.

```
PLAY"CDEFGABCCBAGFEDC"     La scala in do
PLAY"GABCDEF#GGF#EDCBAG"   La scala in sol
```

Ebbene, la scala in do e' quasi giusta, ma quella in sol e' un bel pasticcio. Questo perche' le scale passano in un'altra ottava e noi dobbiamo dirlo al computer. Per scegliere l'ottava usate O seguito da un numero fra 1 e 5. O2 (che comprende il do medio) e' fissato automaticamente quando si accende il computer. Finche' non viene modificata, viene usata l'ottava corrente, quindi e' meglio specificare all'inizio quale ottava volete utilizzare. Proviamo le scale ancora una volta.

```
PLAY"O3CDEFGABO4CCO3BAGFEDC"
PLAY"O3GABO4CDEF#GG#EDCO3BAG"
```

Per far suonare la scala in do utilizzando i numeri invece delle lettere, scrivete

```
PLAY"O3;1;3;5;6;8;10;12;O4;1;1;O3;12;10;8;6;5;3;1"
```

Notate il separatore (;) usato nella stringa. Potete usare il punto e virgola ovunque, ma con i numeri di solito serve ad evitare confusione.

►► AUDIO

Il comando AUDIO controlla il collegamento del registratore a cassette con l'altoparlante del televisore. AUDIO ON attiva il sonoro sull'altoparlante e AUDIO OFF lo disattiva.

►► MOTOR

Il comando MOTOR controlla l'azionamento del motore del registratore. MOTOR ON attiva il motore e MOTOR OFF lo ferma.

Il tasto play del registratore deve essere premuto perche' il comando possa essere efficace.

Siccome la stringa musicale e' sempre una stringa, la si puo' manipolare con tutte le normali operazioni per stringhe. L'esempio seguente suona la scala do per tutta la gamma di possibilita' del comando PLAY.

```
10 A$="CDEFGAB" : FOR I=1 TO 5
20 B$="O"+STR$(I)+A$
30 PRINT B$ : PLAY B$ : NEXT I
```

Con lo stesso sistema, usando numeri invece di lettere, possiamo suonare l'intera scala musicale.

```
10 FOR I=1 TO 5 : A$="O"+STR$(I)+";"
20 FOR J=1 TO 12 : PLAY A$+STR$(J) : NEXT J,I
```

Nella maggior parte dei motivi musicali, le note sono raramente della stessa lunghezza e quindi dobbiamo stabilire la durata di ogni nota. Cio' si fa con il parametro **lunghezza nota** della stringa musicale (L). La lettera L e' seguita da un numero fra 1 e 255. Normalmente, pero', il numero rappresenta le lunghezze comunemente usate in musica. Piu' grande e' il numero piu' breve e' la nota. L1 e' una nota intera, L2 una mezza nota, L4 un quarto di nota, ecc. E' possibile avere 1/255esimo di nota, ma pochi compositori lo adoperano. Chi di voi conosce la musica conoscera' le note "punteggiate". Il "punto" vi dice di aumentare la lunghezza della nota della meta' del suo valore normale. Per ottenere lo stesso risultato con il comando PLAY, dovete mettere un punto (o quanti punti desiderate) dopo il numero del parametro L.

```
L4.=1/4+1/8 = una nota da 3/8
```

Adesso ne sappiamo abbastanza per suonare un motivo semplice. Introducete il seguente programma con attenzione,

```
5 CLEAR 500
10 A$="O2L4GG;L2GDL4BB;L2BGL4GB;
       O3L2DDL4CO2B;L1AL4AB;
       O3L2CCO2L4BA;L2BGL4GB;
       L2ADL4F#A;L1G;"
20 B$=A$+A$ : PLAY B$
```

I separatori (;) vengono qui usati per indicare la divisione fra le battute, anche se non sono veramente necessari. Dovreste riconoscere il motivo "Clementine", ma viene suonato troppo lentamente. Il parametro **tempo** provvede a questo. Esso e' rappresentato dalla lettera T seguita da un numero fra 1 e 255. Piu' alto e' il numero, piu' velocemente viene suonato il motivo. Provate a modificare la riga 20 in

```
20 B$=A$+A$ : PLAY"T6"+B$
```

►► PLAY

Il comando PLAY si usa per generare una sequenza musicale. L'argomento e' una espressione stringa, o una costante stringa, o una variabile stringa. La sua forma e'

PLAY **musica**

dove **musica** e' costituita dai seguenti elementi:

nota una lettera fra "A" e "G" o un numero fra 1 e 12

ottava "O" seguita da un numero fra 1 e 5. Valore di default O2. I valori di default sono stabiliti dal computer quando viene acceso.

lunghezza nota "L" seguita da un numero fra 1 e 255. Default L4

tempo "T" seguita da un numero fra 1 e 155. Default T1.

volume "V" seguita da un numero fra 1 e 31. Default V15.

lunghezza pausa "P" seguita da un numero fra 1 e 255.

esecuzione di sottostringhe "X" seguita da una variabile stringa e un punto e virgola.

Una nota diesis o bemolle si indica con "+" o "#" per un diesis e "-" per un bemolle.

Il parametro **lunghezza nota** puo' essere modificato con l'aggiunta di un punto (.) dopo il numero, (L2.) per rappresentare una nota punteggiata.

L'ottava, il volume, il tempo e la lunghezza nota possono essere modificati usando uno dei seguenti suffissi:

```
+  aggiunge uno al valore attuale
-  sottrae  uno al valore attuale
>  moltiplica il valore attuale per due
<  divide il valore attuale per due
```

```
10 X$="O3L4EF#L4.EL8AAG#ABL4O+C#O-B"
20 A$="XX$;O4C#O-AF#O+DC#O-BL2AXX$;
       O+C#DEL8DO-BL<AG#L<AL4.BL8O+C#L4
       DO-BL4.+C#L8DL<EC#L4.EL8EEEEEL1
       EL4.EL8DC#EDO-BL<AG#L<A"
30 PLAY"T2V20"+A$
```

Fate delle prove cambiando il 6 e trovate un valore che secondo voi rappresenta la velocita' giusta del motivo.

Di solito la musica richiede anche la possibilita' di inserire pause fra brani, nonche' di variare il volume di certi passaggi. Il parametro **pausa** e' la lettera P seguita da un numero. Si comporta in maniera analoga al parametro **lunghezza nota** (L), solo che non potete mettere dei punti dopo il numero. Per inserire una pausa equivalente ad una nota L4., si dovrebbe utilizzare P4P8. Il parametro **volume** ci permette di variare il volume inserendo la lettera V seguita da un numero fra **0** e 31; piu' alto e' il numero, piu' potente e' il volume. L'esempio seguente utilizza il parametro volume per produrre un crescendo.

```
10 A$="V1002L4GG;L1GP4V14L4GGG;
       L1GP4V18L4GGG;L2BL4BBBV22L2BL4BBB;
       V2603L2DL4DDDL2DL4DDD;
       V30L1GL2.F#L4C#;L2EDC02A;
       L1GL2AL4.DL8A;L2B"
20 PLAY"T5"+A$
```

Spesso un motivo musicale contiene un brano che si ripete piu' volte nel motivo. Piuttosto di introdurre questo brano piu' volte, e' meglio metterlo in una variabile stringa a parte. Il comando di **esecuzione di sottostringhe** (X) permette a questa sottostringa di apparire come una parte della normale sequenza dei comandi PLAY. La X deve essere seguita dal nome della variabile stringa **e un punto e virgola**, per esempio:

```
10 X$="O3L2GBO4C;DL4CO3BAG;"
20 Y$="L2ADD;L1.A"
30 Z$="L2ADD;L1.G"
40 A$="XX$;XY$;XX$;XZ$;L2BGG;O4CO3L4
       BAGF#;XY$;XX$;XZ$;"
50 PLAY"T8"+A$
```

Avremo potuto usare una sottostringa nel nostro esempio "Clementine", basta cambiare la riga 20 in

```
20 PLAY"T6XA$;XA$;"
```

►► Il punto e virgola **deve** seguire il segno del dollaro ($), le sottostringhe, e i numeri usati per le note (invece di lettere). Questi sono gli unici posti in cui il punto e virgola e' essenziale.

►► C'e' ancora una opzione che si puo' utilizzare con i parametri volume (V), ottava (O), tempo (T) e lunghezza nota (L). Invece di un numero dopo la lettera, potete usare uno dei seguenti suffissi:

+ aggiunge uno al valore attuale
- sottrae uno al valore attuale
> moltiplica il valore attuale per due
< divide il valore attuale per due

Potremmo riscrivere il nostro ultimo esempio di scala per includere questa ultima opzione,

```
10 PLAY"O1C" : FOR I=1 TO 4 : PLAY"DEFGABO+C" : NEXT I
```

E dove si trova la musica ? Naturalmente potreste comporre la vostra musica, ma per noi comuni mortali, i fogli di musica scritti per strumenti come il flauto e la tromba rappresentano delle fonti molto valide.

Anche se non avete la minima predisposizione per la musica suonata dal computer, non dovete trascurare del tutto il comando PLAY. Gli amanti del gioco possono adoperarlo per degli effetti molto utili. Provate uno degli esempi di questo capitolo con il parametro tempo fissato a T255.

L'ultimo esempio fa uso di tutta la tecnologia moderna del comando PLAY su di una canzone scritta 400 anni fa. Ne sarebbe impressionato l'autore ?

```
10 A$="O3L2E;L1GL2AL2.BL4O+C#L2O-B;
       L1AL2F#L2.DL4EL2F#;L1GL2EL2.EL4DL2E;
       L1F#V10L2DV8L1O-BV6L2O+E;L1GL2AL2.B
       L4O+C#L2O-B;L1AL2F#L2.DL4EL2F#;
       L2.GL4F#L2EV8L2.D#V10L4C#V15L2D#;
       L1.EL1EP1;"
20 B$="O4L1.DL2.DL4C#O-L2B;L1AL2F#L2.D
       L4EL2F#;L1GL2EL2.EL4DL2E;L1F#L2D
       O-L1BO+L2B;O+L1DL2DL2.DL4C#O-L2B;
       L1AL2F#L2.DL4EL2F#;L2.GV10L4F#L2
       EV6L2.D#L4C#V4L2D#;V15L1.EL2EP1;"
30 PLAY"T10XA$;XB$;XA$;XB$;"
```

CAPITOLO X

ANCORA GRAFICA

Nel capitolo VIII abbiamo dimostrato come si possono usare i comandi LINE e CIRCLE per produrre delle forme regolari come rettangoli, cerchi, ellissi ed archi. Anche se questi comandi sono molto utili, disegnare delle forme molto particolari o irregolari con questi comandi puo' richiedere una notevole bravura. Il sistema per affrontare forme di questo genere e' quello di disegnarle.

Quando si disegna una figura su un foglio di carta si comincia in un punto particolare e si sposta la matita un po' piu' in su, poi a destra, ecc. Il comando DRAW vi permette di ripetere questa procedura sul video. La sua forma e',

►► DRAW **stringa**

dove **stringa** e' una costante stringa o una variabile stringa, che contiene una serie di sottocomandi di DRAW. La procedura e' molto simile a quella del comando PLAY del capitolo precedente.

Di solito la prima operazione che effettua chi si accinge a produrre un disegno e' quella di posizionarsi nel punto iniziale.

M **x,y** significa spostarsi nel punto di coordinate **x,y**; per esempio M 128,96 ci fa posizionare al centro del video. Quando ci si sposta in un punto e' di solito conveniente fare una mossa in bianco, cioe' senza disegnare o sollevando la matita dal foglio. Se non fate cosi' potreste trovarvi con delle righe indesiderate sul disegno. Una mossa in bianco si ottiene usando la lettera B; qualunque istruzione sul disegno che segue la B sara' una riga in bianco. BM 128,96 significa spostarsi al centro del video disegnare.

Ora che avete deciso il punto iniziale, potete spostarvi in su (U), in giu' (D), a destra (R), o a sinistra (L) per quanti punti desiderate. La sequenza U20R20D20L20 fara' disegnare una riga verso l'alto per 20 punti, poi verso destra 20, verso il basso 20 e verso sinistra 20, cioe' un quadrato. Cominciamo a costruire un esempio,

```
10 PMODE 3,1 : PCLS : SCREEN 1,1
20 DRAW"C8;BM120,96;U26;R13;D26;L13"
80 GOTO 80
```

Il punto e virgola in una stringa si usa come separatore. Non e' strettamente necessario, l'abbiamo usato soltanto per rendere piu' facile la lettura. L'esempio disegna un rettangolo nella zona centrale del video.

►►Oltre alle righe verticali ed orizzontali, potete disegnare anche delle righe diagonali. Queste si servono dei sottocomandi E, F, G, H. Per esempio, E12 disegna una riga lunga 12 punti con un angolo di 45 gradi dalla verticale. Tutti gli angoli si misurano a partire dalla verticale come segue:

E 45 gradi	F 135 gradi
G 225 gradi	H 315 gradi

Cosi' si possono disegnare delle righe diagonali in tutte le 4 direzioni. Aggiungete la riga,

```
40 DRAW"L6;U6;E6;BR13;F6;D6;L6;BU26;H6;G6"
```

al nostro programma attuale ed il rettangolo diventa un razzo! Il computer si ricorda l'ultima posizione e riprende il disegno da quel punto. Elaborate la stringa della riga 40 fase per fase per capire come si fa. L'ultima posizione disegnata e' l'angolo in basso a sinistra del rettangolo e BR13 significa spostarsi 13 punti verso destra senza disegnare.

Non essendo stato specificato alcun colore, il razzo viene disegnato nel colore del primo piano (default), ma lo possiamo cambiare usando la lettera C seguita da un numero fra 0 e 8 che rappresenta il codice di uno dei colori disponibili. Usando l'EDITOR, cambiate la riga 40 come segue,

```
40 DRAW"C7;L6;U6;E6;BR13;F6;D6;L6;BU26;H6;G6"
```

ora abbiamo un razzo bicolore! Il disegno puo' essere colorato come le altre forme, ma il comando C cambia il colore di default (del primo piano) e quindi bisogna stare attenti a non coprire tutto.

►►Il disegno e' un po' piccolo e quindi lo ridimensioneremo con il parametro S. La S sta per scala e permette il ridimensionamento del disegno, o di parti del disegno, in unita' di 1/4. S1 riduce il disegno in una scala pari ad 1/4, S2 a 2/4 (a meta'), S8 a 8/4 (il doppio), e cosi' via. La scala di default e' 4/4 (cioe' 1, la grandezza originale). La S puo' essere seguita da qualunque numero fra 1 e 62. Aggiungete la riga

```
15 DRAW"S12"
```

adesso il razzo e' tre volte piu' grande.

►►Un altra opzione disponibile e' il parametro angolo A. Questo ci permette di far ruotare tutto o una parte del disegno, perche' tutte le righe dopo la A verranno disegnate con uno spostamento dato da An dove n e' un numero fra 0 e 3 come segue,

0 0 gradi	1 90 gradi
2 180 gradi	3 270 gradi

Modificate il programma attuale con le righe seguenti

```
10 FOR I=0 TO 3 : DRAW"A"+STR$(I) : PCLS
50 FOR D=1 TO 100 : NEXT D,I
```

Adesso il razzo gira e cambia anche colore! Cio' perche' il computer si ricorda non soltanto l'ultima posizione, ma anche l'ultima definizione di C e di A. Si puo' risolvere il problema inserendo C8 all'inizio della stringa nella riga 30.

La riga 20 dimostra come le stringhe unite da DRAW - come pure quelle del comando PLAY - possono essere adoperate con le funzioni stringa. Analogamente al comando PLAY, si possono anche eseguire delle sottostringhe con il comando X seguito da una variabile stringa, XA$, e da un punto e virgola (;).

```
10 PMODE 3,1 : SCREEN 1,1 : PCLS
20 S$="L8E4F4"
30 D$="A0;XS$;A1;XS$;A2;XS$;A3;XS$;"
40 DRAW"S24"+D$
50 GOTO 50
```

Viene memorizzato un triangolo come una sottostringa in S$ che poi viene usato nella riga 30 per costruire un disegno. Notate che questo e' l'unico posto dove il punto e virgola e' indispensabile, dopo il segno del dollaro ($).

►►L'ultimo parametro e' N, che dice al computer di non aggiornare la posizione del disegno. Esso serve per disegnare una riga come specificato senza usare il punto terminale della riga come nuova posizione. NU10L5 disegna una riga verticale di 10 punti, **ritorna all'inizio della riga**, e disegna 5 punti verso destra (una forma ad L).

```
10 PMODE 3,1 : SCREEN 1,1 : PCLS
20 DRAW"BM128,96;NU25ND25NR25NL25;NE17NF17NG17NH17"
30 GOTO 30
```

Questo esempio disegna righe dal centro verso l'esterno, ritornando sempre al centro per cominciare la riga successiva.

Spesso vorrete aggiungere un altro disegno vicino a quello appena ultimato. Sapete dove sara' il nuovo disegno in relazione a quello vecchio, ma non volete calcolare le coordinate. Allora potete usare lo **spostamento relativo**, come 5 punti a destra e 10 verso l'alto. Il comando **spostamento** (M) permette di fare cio' senza problemi, basta specificare la distanza in termini di piu' o meno il punto attuale, cioe' M+5,-10. Non dimenticate di usare la B per evitare delle righe indesiderate.

```
25 DRAW"BM-25,-25;U10R25D10L25"
```

Aggiungete la riga precedente all'ultimo esempio e verra' disegnato un rettangolo sopra l'ultimo disegno. L'ultima posizione era in 128,96 a causa del parametro N. Adesso ci siamo spostati 25 punti a sinistra e 25 punti in su (ricordatevi che y=0 e' in alto sul video), ed il disegno del rettangolo inizia al punto (103,71).

I risultati di un comando DRAW possono essere uniti con forme
►► derivanti dai comandi LINE e CIRCLE, ma ricordatevi che un successivo ridimensionamento (S), o cambiamento di colore (C) o angolo (A) interessera' soltanto la parte della figura eseguita con il comando DRAW.

Si possono utilizzare i comandi PSET e PRESET insieme con PAINT per aggiungere ulteriori dettagli e colori. State attenti, soprattutto con PAINT, perche' i cambiamenti della parte di figura eseguita con DRAW possono avere la conseguenza di far finire il colore in posti del tutto sbagliati.

"GET" E "PUT"

Avete completato il vostro capolavoro usando LINE, CIRCLE, DRAW ecc. Ora supponiamo di volerlo spostare sul video. Ovviamente si potrebbe cancellarlo tutto e ridisegnarlo altrove. Ma se il disegno e' complicato ci vorra' un po' di tempo. I prossimi due comandi risolvono questo problema. Basta usare GET per ottenere una copia del disegno e PUT per ricollocarlo. Il comando GET vi permette di copiare un'area rettangolare del video in un array, che si puo' rimettere sul video piu' tardi, con PUT.

►► **GET(x1,y1)-(x2,y2), nome di array, G**

x1,y1 e **x2,y2** sono le coordinate dei due angoli, in alto a sinistra ed in basso a destra, del rettangolo che racchiude il disegno che volete memorizzare. Il **nome di array** e' il nome di un **array gia' dimensionato** dove si memorizza la figura (se non vi ricordate piu' degli arrays, date un'occhiata all'inizio del capitolo VI). Le dimensioni dell'array devono essere uguali a quelle del rettangolo di video. La prima dimensione dell'array e' la **larghezza** del rettangolo **(x2-x1)**, la seconda e' la **lunghezza (y2-y1)**. L'ultimo parametro (G) e' opzionale e determina il livello di dettaglio dell'area di video memorizzata. Il parametro G e' indispensabile nei PMODES 0,1 o 3 altrimenti gli spostamenti in senso orizzontale con PUT possono risultare imprecisi.

Utilizzeremo il nostro razzo per dimostrare come funziona. Prima dobbiamo determinare le dimensioni dell'array che ci servira'. Il disegno inizia a 120, 96, l'aletta di sinistra e' larga 6 punti, il razzo e l'aletta di destra sommano 13+6 punti, quindi il disegno e' largo 25 punti, da 114,96 a 139,96.

L'altezza e' 26 punti, piu' 5 per il cono del muso, quindi 31 in tutto. Se aggiungiamo qualche punto nei due sensi abbiamo un rettangolo 30x40 con l'angolo in alto a sinistra 112,60 e l'angolo in basso a destra 142,100.

```
10 PMODE 3,1 : SCREEN 1,1 : PCLS : DIM R(29,39)
20 R$="C8BM120,96;U26R13D26L13;C7L6U6E6BR13
      F6D6L6BU26H6G6"
30 DRAW R$
40 GET(112,60)-(142,100), R, G
100 GOTO 100
```

Questo esempio disegna il razzo come prima (adesso tutto in una stringa) e lo memorizza nell'array R. Notate che ci basta una array 29x39 perche' possiamo utilizzare gli elementi dell'array con indice 0.
Ora che abbiamo memorizzato il disegno, vogliamo farlo riapparire sul video. Il comando PUT ha una forma simile a quella di GET

►► PUT(x1,y1)-(x2,y2), **nome di array, azione**

x1,y1 e **x2,y2** sono sempre le coordinate del rettangolo, ma questa volta si riferiscono al rettangolo nel quale volete collocare il disegno, non a quello dov'era. Il **nome di array** individua l'array che contiene il disegno memorizzato. Il parametro **azione** e' opzionale e serve soltanto quando e' stato usato il parametro G con il comando GET. L'**azione** deve essere una delle seguenti parole e determina il modo in cui viene visualizzato il risultato nella sua nuova posizione.

►► PSET Setta (accende) ogni punto settato nell'array di provenienza, cioe' lo visualizza cosi' com'era.

PRESET Resetta (spegne) ogni punto settato nell'array di provenienza. In questo modo il disegno viene cancellato o i colori vengono invertiti rispetto ai colori di fondo e di primo piano originari.

AND Esegue un confronto fra i punti dello schermo originale e quelli dello schermo di destinazione. Se entrambi sono accesi il punto sara' acceso. Se uno o l'altro non e' acceso, il punto sara' spento. Cosi', in caso di sovrapposizione di due figure, saranno visualizzati soltanto i punti che combaciano.

OR Esegue un confronto fra i punti come sopra. Se il punto originale o quello di destinazione e' acceso, verra' acceso il punto sul video. Questo ha l'effetto di sovrapporre i due disegni.

NOT Inverte ogni punto nell'area del video in modo che la figura viene visualizzata con il colore del primo piano come colore di fondo.

GET

Il comando GET puo' essere utilizzato soltanto nei modi grafici ad alta risoluzione. GET copia il contenuto grafico di un'area rettangolare precisata sul video e la memorizza in un array. L'array deve essere stato precedentemente dimensionato di una grandezza appropriata.

►► **GET(x1,y1)-(x2,y2), nome di array, G**

x1,y1 e **x2,y2** sono le coordinate degli angoli, in alto a sinistra ed in basso a destra, del rettangolo sul video.

nome di array e' il nome dell'array precedentemente dimensionato in cui viene immagazzinato il contenuto del rettangolo.

G e' il comando che dice di memorizzare tutti i dettagli GRAFICI. Il comando e' opzionale. Vedete la cornice PUT per un esempio dell'impiego di GET.

Dovete sempre usare PUT nello stesso modo di GET, altrimenti potreste ottenere degli strani risultati. Ora possiamo tornare al nostro esempio e mettere il razzo in un posto diverso. Aggiungete queste righe.

```
50 Y=150 : FOR X=10 TO 210 STEP 40
60 PUT(X,Y)-(X+30,Y+40), R, PSET
80 NEXT X
```

Adesso dovremmo avere una fila di razzi disposti nella parte inferiore del video. Per spostare il razzo lungo la fila, basta aggiungere la riga

```
70 FOR D=1 TO 200 : NEXT D : PCLS
```

Utilizzando i joysticks insieme con i comandi GET e PUT, potete spostare la figura a volonta'.

```
10 PMODE 3,1 : SCREEN 1,1 : PCLS : DIM S(48,48)
20 DRAW"BM24,12;S8;C4;E2H2D4D8R8H8G8R2
       NR6F3R6E3"
30 GET(0,0)-(48,48), S
40 A$=INKEY$ : IF A$="" THEN 40
50 PCLS : A=JOYSTK(0)*3.25 : B=JOYSTK(1)*2.25
60 PUT(A,B)-(A+48,B+48), S : GOTO 50
```

Questo esempio disegna una figura nell'angolo in alto a sinistra dello schermo. Quando premete un tasto qualunque il video si pulisce e la figura puo' essere spostata qua' e la' per mezzo del joystick.

Con questo abbiamo finito la nostra descrizione delle possibilita' grafiche disponibili. Gli esempi presentati sono necessariamente limitati e in nessun modo possono dirsi rappresentativi di quanto si puo' realizzare con poca applicazione e molta pazienza. Il disegno di figure puo' essere semplificato molto se si fa un progetto prima e se si disegna prima su un modulo grafico piuttosto che direttamente sul video.

DRAW

Il comando DRAW disegna una riga o una serie di righe secondo le istruzioni contenute in una stringa. Esso e' utilizzabile esclusivamente nei modi grafici ad alta risoluzione.

►► DRAW **stringa**

La **stringa** puo' essere una costante stringa o una variabile stringa e puo' contenere ciascuno dei seguenti comandi.

M**x,y**	Sposta nel punto di coordinate **x,y**
U**n**	In su **n** punti
D**n**	In giu' **n** punti
L**n**	A sinistra **n** punti
R**n**	A destra **n** punti
E**n**	A 45 gradi per **n** punti
F**n**	A 135 gradi per **n** punti
G**n**	A 225 gradi per **n** punti
H**n**	A 315 gradi per **n** punti
X	Esegue una sottostringa e ritorna
C	Stabilisce il colore della riga
A**k**	Ruota la riga successiva dell'angolo **k**=0 0 gradi **k**=1 90 gradi **k**=2 180 gradi **k**=3 270 gradi
S**k**	Disegna in scala in unita' di 1/4, **k** compreso fra 1 e 62 **k**=1 scala 1/4, **k**=8 scala doppia **k** di default =4
N	Nessun aggiornamento della posizione di disegno
B	In bianco (non disegna, sposta soltanto)

Lo spostamento relativo puo' essere specificato con il parametro M nella forma

M **x offset, y offset**

dove **x offset** e **y offset** sono numeri che specificano la entita' dello spostamento dalla posizione attuale. Entrambi i numeri **devono** essere preceduti o da un segno piu' (+) o da un segno meno (-).

Nella cornice PUT c'è' un esempio dell'impiego del comando DRAW.

PUT

Il comando PUT viene usato per visualizzare il contenuto di un array grafico che e' stato memorizzato con il comando GET.

PUT deve essere adoperato nello stesso modo usato per creare lo array originale, altrimenti i risultati possono essere imprevedibili.

►► **PUT(x1,y1)-(x2,y2), nome di array, azione**

x1,y1 sono le coordinate dell'angolo in alto a sinistra del rettangolo sul video. **x2,y2** e' l'angolo in basso a destra. Il **nome di array** si riferisce all'array precedentemente definita che contiene il dettaglio grafico. Il parametro **azione** e' opzionale, ma e' **obbligatorio** se nel comando GET e' stato usato il parametro G. L'**azione** puo' essere una delle seguenti:

PSET Setta i punti di destinazione come nell'array di origine.

PRESET Resetta ogni punto settato nell'array originale.

AND Confronta i punti dell'array d'origine con quelli dello schermo di destinazione. Se entrambi i punti sono accesi il punto rimane acceso, altrimenti viene spento.

OR Confronta i punti come sopra. Se uno o l'altro punto e' acceso, il punto sul video risultera' acceso.

NOT Inverte lo stato di ogni punto nell'area di destinazione indipendentemente dall'array di origine.

L'area di visualizzazione scelta deve essere delle stesse dimensioni dell'array altrimenti appariranno sul video dei segni incomprensibili.

```
10 PCLEAR 4 : PMODE 3,1 : PCLS : SCREEN 1,1 : DIM W(30,30)
20 DRAW"BM10,12;S8;R1U3R1D2R2U2R1D3R1D2L1
       D2R1D1L2U3L4D3L2U1R1U2L1U2R1U2BR1BD1D2R2
       U2NL2R2D2L2U2"
30 PAINT(11,13),6,5 : GET(0,0)-(30,30),W
40 A$=INKEY$ : IF A$="" THEN 40
50 PCLS : FOR C=0 TO 100 STEP 20
60 FOR A=0 TO 200 STEP 20
70 PUT(A,C)-(30+A,30+C),W
80 PUT(A,C+30)-(30+A,60+C),W
90 PUT(A,C+60)-(30+A,90+C),W
100 PLAY"T255;ABFGBA" : PCLS : NEXT A,C
```

CAPITOLO XI

IL TOCCO FINALE

ULTERIORI IMPIEGHI DI PRINT

Anche se il vostro controllo sul modo di visualizzare i risultati e' piuttosto ampio mediante i comandi PRINT e PRINT@, esiste ancora una possibilita'. Il comando PRINT USING vi permette di specificare esattamente come dovrebbe essere stampata ogni riga. Esso e' particolarmente utile per la redazione di tabelle, moduli e schemi per la contabilita'.

►► PRINT USING **formato, lista di output**

Il **formato** e' una costante stringa o una variabile stringa che contiene le istruzioni relative a come stampare la **lista di output**. La **lista di output** e' la solita lista di costanti e variabili che appare nel normale comando PRINT.

Le istruzioni nel **formato** sono costituite da simboli chiamati "specificatori di campo". Questi sono una serie di caratteri che dicono al computer esattamente quante posizioni di stampa deve usare per scrivere un numero o una stringa.

►► Il simbolo

Questo carattere viene utilizzato per indicare la posizione di ogni cifra di un numero.

```
PRINT USING"###.##";A
```

L'istruzione precedente scrive il contenuto di A in un formato che prevede tre cifre significative prima del punto decimale e due dopo. Se ci sono piu' di due cifre dopo il punto decimale, il numero viene appropriatamente arrotondato. Le eventuali posizioni libere a sinistra del punto decimale vengono visualizzate come spazi. Se il numero e' troppo grande per stare nello spazio disponibile, il computer fa il possibile: scrive egualmente il numero stampandogli davanti un segno % per indicare quello che e' avvenuto.

```
PRINT USING"###.##";13.4695
 13.47
PRINT USING"###.##";1492.878
%1492.88
PRINT USING"###.##";146
146.00
PRINT USING"###";18.76
 19
```

►► Il simbolo *

Generalmente ai ragionieri non piace l'idea di scrivere numeri con spazi davanti, soprattutto per gli assegni. Si può risolvere questo problema con il simbolo asterisco. Se mettete **due** asterischi all'inizio del vostro campo numerico, le posizioni non utilizzate verranno riempite con asterischi.

```
PRINT USING"**####.##";1.492
****1.49
```

►►Il simbolo +

Quando all'inizio di un campo numerico viene inserito il + esso forza la stampa del segno del numero.

```
PRINT USING"+###.##":14.7
 +14.70
PRINT USING"+******.##";-7.4
****-7.40
```

Se il segno piu' (+) viene posto dopo il campo numerico, il segno viene stampato **dopo** il numero.

```
PRINT USING"###.##+";27.86
 27.86+
PRINT USING"###.##+";-1.6
  1.60-
```

Se viene inserito un segno meno come ultimo simbolo, tutti i numeri negativi appaiono seguiti da un segno meno, e i numeri positivi sono seguiti da uno spazio,

```
PRINT USING"#####.##-";-12.418
***12.42-
PRINT USING"###.##-";47.25
 47.25
```

►►Il simbolo ↑↑↑↑

Questo campo permette la stampa dei numeri in forma esponenziale. Le quattro frecce verso l'alto devono seguire il campo del numero.

```
PRINT USING"##.###↑↑↑↑";123456
 1.2346E+05
```

►►Il simbolo !

Questo simbolo viene usato con le stringhe. Esso fa si che venga stampato soltanto il primo carattere della stringa.

```
PRINT USING"!";"CREDIT"
C
```

►►Il simbolo %

Per stampare le stringhe bisogna specificare la larghezza del campo che occuperanno. Cio' si fa con due segni % separati da un numero di spazi. La larghezza del campo sara il numero di spazi piu' due. Se la stringa e' piu' larga del campo disponibile, saranno stampati soltanto i primi **n** caratteri, dove **n** e' la larghezza del campo.

```
PRINT USING"%     %";"DEBIT"
DEBIT
PRINT USING"% %";"BALANCE"
BAL
```

►►Il simbolo $

Il segno dollaro ($) viene usato per rappresentare denaro. Se esso viene posto davanti ad un campo numerico, l'output conterra' un segno dollaro.

```
PRINT USING"$###.##";2.87
$  2.87
```

Se si mettono due segni dollaro, il $ viene stampato subito davanti al numero.

```
PRINT USING"$$###.##";2.87
   $2.87
```

Usato insieme con due asterischi, il segno del dollaro produce il seguente effetto,

```
PRINT USING"**$#.##";14.9
**$14.90
```

Spazi ed altri caratteri presenti nella stringa del formato appariranno anche nell'output,

```
PRINT USING"MEAN  ##.##   TOTAL  ###.##";3.4,40.8

MEAN   3.40   TOTAL   40.80
```

Se la **lista di output** contiene piu' voci del numero di campi nel relativo **formato**, il **formato** riparte dall'inizio

```
PRINT USING"###.##   ";7.84,142.5,.234

  7.84   142.50     0.23
```

Naturalmente, usando il video, la lunghezza della riga scritta con il comando PRINT USING resta limitata a 32. Qualsiasi lunghezza maggiore di 32 provochera' l'automatico proseguimento sulla riga successiva. Se avete una stampante, pero', potrete usare una riga molto piu' lunga (di solito di almeno 80 caratteri). La forma da usare con la stampante e'

►► PRINT#-2,USING **formato, lista di output**

Per quanto riguarda il **formato** e la **lista di output** valgono le considerazioni fatte in precedenza. Il -2 dice al computer di inviare l'output al canale stampante e non al video. Se volete che l'output sia visualizzato e stampato allo stesso tempo dovrete usare due istruzioni PRINT USING.

INPUT E OUTPUT SU CASSETTA

Finora, per tutti i nostri programmi, abbiamo dovuto introdurre tutti i dati occorrenti (o leggerli con READ da una istruzione DATA), e tutti gli output sono andati a finire sul video. Potete pero' usare il vostro registratore per immagazzinare anche dati, oltre che programmi. Questi dati memorizzati possono essere riletti in un'altra occasione. Il registratore viene collegato e predisposto esattamente come per immagazzinare programmi. Successivamente dovete dire al computer che lavorate con "files" di dati. Per ottenere cio' si usa il comando OPEN

►► OPEN **a,#-1, nome di file**

La a deve essere "O" o "I". "O" significa output cioe' i dati di uscita dal computer al nastro. "I" significa input, cioe' i dati di ingresso, dal nastro al computer.

Il #-1 dice al computer che state usando il registratore a cassetta. Il **nome di file** e' il nome che volete dare al file di dati (un nome qualsiasi e' sufficiente, basta che inizi con una lettera e non superi gli 8 caratteri).

Il prossimo passo e' quello di scrivere i dati sul nastro. Cio' si ottiene con il comando PRINT, come segue

PRINT**#-1, lista di output**

L'unica differenza fra questo comando ed il comando PRINT finora usato e' data dalla presenza del #-1. Questo dice al computer di scrivere la **lista di output** sul nastro e non sul video.

Quando avete finito di scrivere i dati, **dovete** chiudere il file con il comando CLOSE

►► CLOSE#-1

►► PRINT USING

Il comando PRINT USING permette un maggior controllo del formato dei risultati in uscita sia sul video che sulla stampante.

PRINT USING **formato, lista di output**

Il **formato** e una costante o variabile stringa contenente gli "specificatori di campo" che indicano come scrivere la **lista di output.** La **lista di output** e' una lista di variabili (o costanti) stringa o numeriche separate da virgole.

Gli "specificatori di campo" sono i seguenti;

CARATTERE	AZIONE	ESEMPIO	RISULTATO
#	Formatta i numeri	"####";147.2	147
.	Punto decimale	"##.##";34.678	34.68
,	Scrive la virgola ogni tre cifre come separatore delle migliaia	"######.#";123456	123,456
**	Riempie gli spazi che precedono il numero con asterischi	"*****.###";1.47	****1.470
$	Pone il segno $ allo inizio del numero	"$#####.##";12.689	$ 12.69
**$	Segno del dollaro fluttuante	"**$####.##";12.689	****$12.69
+	Messo come primo carattere determina la stampa del segno davanti al numero, in ultima posizione ne determina la stampa dopo il numero	"##.##+";-12.689	12.69-
↑↑↑↑	Scrive in formato esponenziale	"##.##↑↑↑↑";12.689	1.27E+01
!	Scrive soltanto il primo carattere della stringa	"!";"CREDIT"	C
%spazi%	Campo stringa. La lunghezza del campo e' uguale al numero di spazi piu' due	"% %";"BALANCE"	BALANCE

Ogni "specificatore di campo" puo' essere separato da un numero qualsiasi di spazi che verranno visualizzati come tali nella riga di output.

```
10 CLS : INPUT"INSERISCI L'ULTIMO BILANCIO";B : C=0 : D=0
20 CLS : T$="    %    %    %     %    %     %"
30 L$="   ####.##  ####.##  ####.##+"
40 PRINT USING T$;"DEBITO","CREDITO","BILANCIO"
50 PRINT USING L$;D,C,B
60 OPEN"I",#-1,"CHEQ"
70 IF EOF(-1) THEN 110
80 INPUT#-1,A : D=0 : C=0
90 IF A<0 THEN D=ABS(A) ELSE C=A
100 B=B+C-D : PRINT USING L$;D,C,B : GOTO 70
110 CLOSE#-1 : END
```

►► OUTPUT SU STAMPANTE

Per coloro che hanno una stampante collegata alla porta parallela I/O esistono variazioni ad alcuni comandi che permettono di indirizzare l'output alla stampante e non al video.

PRINT#-2, lista di output
PRINT#-2, USING formato, lista di output

Il **formato** e la **lista di output** sono gli stessi usati per il video.

POS(-2) fornisce la posizione attuale della testina di stampa.

LLIST lista un programma direttamente su stampante. Si usa come il comando LIST.

L'uso di [SHIFT][0] permette di inviare in output alla stampante lettere minuscole. Si possono usare le minuscole soltanto all'interno di stringhe o istruzioni REM, perche' tutti i comandi al computer devono essere dati in maiuscolo.

Per rileggere i dati, seguite gli stessi passi, solo che questa volta dovete aprire un file per INPUT ed invece di PRINT dovete usare,

INPUT#-1, lista di input

Il comando CLOSE e' uguale in entrambi i casi.

Gli esempi che seguono mostrano come si fa. Prima preparate il registratore ed avvolgete il nastro fino al punto dove volete inserire il file (usate SKIPF). Adesso premete insieme i tasti PLAY e RECORD.

```
10 CLS : PRINT" CREAZIONE RUBRICA TELEFONICA"
20 OPEN"O",#-1,"TELEFONO" : PRINT" INSERIRE XXX,XXX PER FINIRE"
30 PRINT@128,""; : INPUT"NOME ",N$;
40 INPUT"NO. TELEFONICO";T$ : IF
   N$="XXX" OR T$="XXX" THEN 60
50 PRINT#-1,N$,T$ : PRINT@128,"": PRINT@160,"":GOTO 30
60 CLOSE#-1 : END
```

Quando fate girare il programma il nastro parte ed inizia il file sul nastro. Ogni volta che introducete un nome ed un numero essi vengono scritti nel file. (L'istruzione PRINT@128 alla riga 30 serve solamente per pulire la riga sul video). Il programma continuera' in questo modo finche' non introducete XXX,XXX al che' il file si chiude ed il programma termina.

Ora non dobbiamo che rileggerlo. La differenza principale fra output ed input sta nel fatto che con input non dovete tentare di leggere oltre la fine del file. Il comando aggiuntivo, EOF, che serve per input, provvede a questo: controlla se avete raggiunto la fine del file.

Riavvolgete il nastro all'inizio e questa volta premete il solo tasto PLAY.

```
10 CLS : PRINT"LETTURA RUBRICA TELEFONICA"
20 OPEN"I",#-1,"TELEFONO"
30 PRINT"NOME","NUMERO"
40 IF EOF(-1) THEN 60
50 INPUT#-1,A$,B$ : PRINT A$,B$ : GOTO 40
60 CLOSE#-1 : END
```

Quando fate girare il programma adesso il nastro parte e cerca il file "TELEFONO". (Se il file si trova verso la fine del nastro forse dovrete aspettare un po). Poi il programma rilegge il nome e il numero e li visualizza sul video. Notate che non dovete usare lo stesso nome di variabile adoperato per scrivere i dati. **dovete**, pero', usare lo stesso tipo di variabile. Una volta raggiunta la fine del file, esso verra' chiuso ed il programma termina.

Il comando EOF deve comparire prima del comando INPUT#-1, altrimenti otterrete un messaggio di errore IE (per aver cercato di leggere oltre la fine del file).

Non dimenticatevi di chiudere un file se non volete problemi, specialmente quando state introducendo dati in un file.

QUALCHE ULTERIORE SUGGERIMENTO

Ormai siete ben avviati sulla strada per diventare esperti della programmazione con il BASIC, e forse vorrete guardare piu' avanti - al linguaggio macchina. Questa e' la vera lingua del computer, finora gli avete parlato attraverso un interprete che parla il BASIC.

Perche' vi potrebbe interessare ? Perche' le istruzioni in linguaggio macchina agiscono piu' rapidamente, possono occupare meno spazio in memoria, e possono perfino permettervi di fare certe cose che non sono possibili con il BASIC.

L'approccio migliore e' quello di procurarvi un manuale sul linguaggio macchina, con particolare riferimento ai microprocessori della serie 6800. Uno di questi manuali e'

Basic Microprocessors and the 6800 di Ron Bishop, pubblicato dalla **Hayden Book Co. Inc.**

Una volta che avete una base, il vostro computer ha diverse funzioni che vi permettono di usare routines in linguaggio macchina. Ecco in breve alcuni particolari in merito.

►► USRn (**argomento**)

Dove **argomento** puo' essere una stringa o una espressione numerica. Quando si trova una chiamata USR nel programma, il controllo passa all'indirizzo dato nell'istruzione DEF USRn. L'indirizzo specifica il punto di ingresso (entry point) della routine in linguaggio macchina.

DEF USRn = **indirizzo**

n e' fra **0** e **9** e combacia con **n** in USR. L'**indirizzo** deve essere fra **0** e 65535 e contenere l'indirizzo di ingresso per USRn.

►► CLEAR **s,h**. L'istruzione CLEAR dovrebbe essere usata per riservare memoria per le funzioni USR. La **s** si riferisce alla quantita' di spazio per stringhe riservato nella stessa maniera di prima.

La **h** e' il piu' alto indirizzo di memoria che il BASIC puo usare. Lo spazio da **h+1** in su e' riservato per le routines in linguaggio macchina.

POKE. Il comando POKE e' usato per inserire un valore in un indirizzo specifico di memoria.

►POKE **indirizzo, valore**

Per l'**indirizzo** valgono le considerazioni fatte in precedenza. Il **valore** deve essere compreso fra 0 e 255.

VARPTR. Come argomento di una funzione USR puo' essere usato un puntatore a una variabile BASIC. Questo permette ad una funzione USR di accedere al contenuto di un array.

►►VARPTR **(nome di variabile)**

dove **nome di variabile** rappresenta la variabile BASIC cui desiderate accedere. VARPTR e' usato come parte dell'argomento USR come, per esempio,

```
USR0(VARPTR(X))
```

Routines in linguaggio macchina possono essere memorizzate e caricate da cassetta usando CSAVEM e CLOADM.

►►CSAVEM **nome, inizio, fine, entry**
►►CLOADM **nome, offset**

Nome e' il nome del file su cassetta. **Inizio** e' l'indirizzo iniziale della routine in memoria. **Fine** e' l'ultimo indirizzo occupato dalla routine e **entry** e' il punto di ingresso del programma. L'**offset** nel comando CLOADM vi permette di ricaricare la routine in memoria ad un indirizzo fornito da **inizio+offset.**

Una volta caricato il programma, il controllo puo' essere passato alla routine con il comando EXEC,

►►EXEC **indirizzo**

L'**indirizzo** e' l'indirizzo iniziale della routine. Se si omette **indirizzo**, il computer assumera' l'indirizzo di inizio dall'ultimo comando CLOAD.

TASTO	SENZA IL TASTO SHIFT	CON IL TASTO SHIFT
[BREAK]	3	3
[CLEAR]	12	92
[ENTER]	13	13
[BARRA SPAZIATRICE]	32	32
!	33	-
"	34	-
#	35	-
$	36	-
%	37	-
&	38	-
'	39	-
(	40	-
)	41	-
*	42	-
+	43	-
,	44	-
-	45	-
.	46	-
/	47	-
0	48	18
1	49	-
2	50	-
3	51	-
4	52	-
5	53	-
6	54	-
7	55	-
8	56	-
9	57	-
:	58	-
;	59	-
<	60	-
=	61	-
>	62	-

TASTO	SENZA IL TASTO SHIFT	CON IL TASTO SHIFT
?	63	-
@	64	19
A	97	65
B	98	66
C	99	67
D	100	68
E	101	69
F	102	70
G	103	71
H	104	72
I	105	73
J	106	74
K	107	75
L	108	76
M	109	77
N	110	78
O	111	79
P	112	80
Q	113	81
R	114	82
S	115	83
T	116	84
U	117	85
V	118	86
W	119	87
X	120	88
Y	121	89
Z	122	90
↑	94	95
↓	10	91
<-	8	21
->	9	93

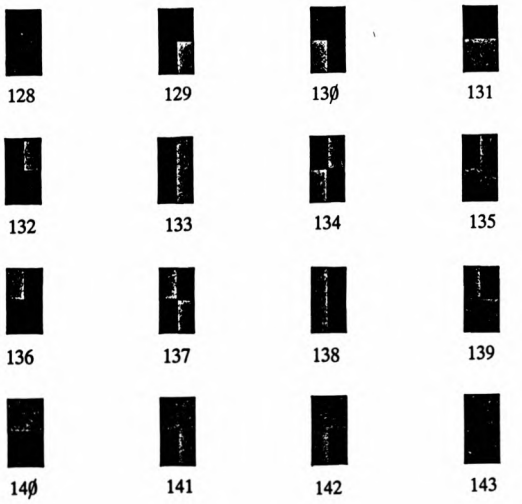
I caratteri senza SHIFT si ottengono adoperando la combinazione [SHIFT][0] per scrivere in minuscolo.

I seguenti caratteri "minuscoli" sono disponibili con la funzione CHR$:

[CHR$(123)
/ CHR$(124)
] CHR$(125)
↑ CHR$(126)
<- CHR$(127)

I caratteri fra 128 e 255 rappresentano i seguenti caratteri grafici

CARATTERI GRAFICI



Per produrre questi caratteri usate CHR$ con il codice appropriato. Per ottenere gli altri colori, aggiungete al codice il numero relativo. Per esempio, PRINT CHR$(142+112) produce il carattere 142 solo che l'area verde e' arancione.

+16 giallo	+32 blu	+48 rosso
+64 beige	+80 azzurro	+96 magenta
	+112 arancione	

APPENDICE B

TAVOLE DI RIFERIMENTO PER LA STAMPA E LA GRAFICA SU VIDEO

I seguenti fogli di lavoro sono utili per progettare il formato per la grafica e per la stampa.

Il primo schema si usa per il comando PRINT@.

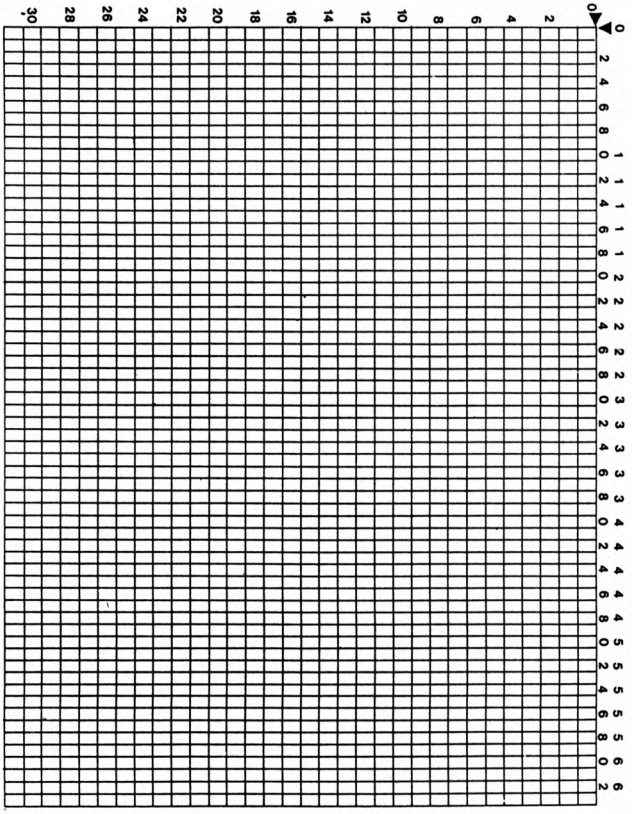
Il secondo serve per la grafica a bassa risoluzione sul "text screen", usando i comandi SET e RESET.

Il terzo serve per il video ad alta risoluzione e per tutti i comandi grafici ad alta risoluzione.

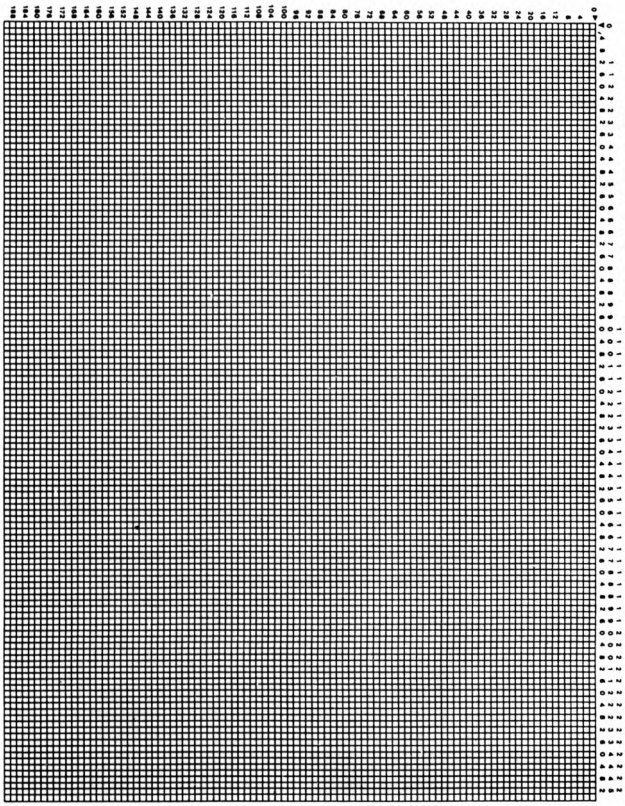
Print @ Grid

0
32
64
96
128
160
192
224
256
288
320
352
384
416
448
480
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

Low Resolution Grid



High Resolution Grid



APPENDICE C

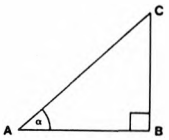
►► CODICI DI ERRORE

CODICE	SPIEGAZIONE
/0	Divisione per 0. Impossibile.
AO	Tentativo di aprire un file che e' gia' aperto. Di solito compare dopo aver premuto RESET per fermare un programma che usa dei files. Spegnete e riaccendete.
BS	Indice errato. Di solito perche' il valore dell'indice e' maggiore della dimensione dell'array dichiarata.
CN	Non si puo' continuare. Compare quando si tenta di usare CONT alla fine di un programma.
DD	Tentativo di ridimensionare un array. Si possono dimensionare gli array solo una volta nello stesso programma.
DS	Istruzione diretta. Di solito appare quando si tenta il CLOAD di un file di dati.
FC	Chiamata di funzione errata. Di solito e' sbagliato il parametro o il tipo di variabile.
FD	Dati di file sbagliati. Provocato dal tentativo di leggere dati stringa in una variabile stringa usando files di dati su cassetta.
FM	Uso del file sbagliato. Deriva dal tentativo di leggere (INPUT) un file che e' stato aperto per output (O), o di scrivere (PRINT) dati su un file aperto per input (I).
ID	Istruzione diretta errata. Tentativo di usare una istruzione utilizzabile solo in un programma, es. INPUT, DEF FN.
IE	Tentativo di introdurre input al di la' della fine di un file. Usate IF EOF(-1) per controllare che cio' non avvenga.
IO	Errore di input/output. Il registratore non e' regolato bene oppure il nastro non va.
LS	Stringa troppo lunga. La lunghezza massima e' di 255 caratteri.
NF	NEXT senza FOR. Di solito si presenta quando si invertono le istruzioni NEXT in un loop annidato.
NO	File non aperto. Operazioni di input e output su un file di dati sono possibili soltanto dopo una OPEN.
OD	Dati esauriti. Una istruzione READ ha letto tutte le istruzioni DATA.
OM	Spazio in memoria esaurito. E' stato usato o e' gia' riservato tutto lo spazio disponibile in memoria.
OS	Spazio per stringhe esaurito. Usate CLEAR per creare dello spazio ulteriore se questo e' disponibile.

CODICE	SPIEGAZIONE
OV	"Overflow". Il numero e' troppo grande perche' il computer lo possa trattare (ABS(X)>1E38).
RG	RETURN senza GOSUB. Probabilmente il programma e' "caduto" in una subroutine (usate END) o si e' saltati in una subroutine.
SN	Errore di sintassi. Di solito provocato da errori di digitazione o dalla punteggiatura sbagliata.
ST	Formula stringa troppo complessa. Dividete la operazione in parti piu' piccole.
TM	Tipo sbagliato. Tentativo di assegnare dati stringa ad una variabile numerica o viceversa.
UL	Riga non definita. Una istruzione di salto e' stata diretta ad una riga che non esiste.

APPENDICE D

FUNZIONI TRIGONOMETRICHE



In un triangolo rettangolo ABC

AB e' il lato **adiacente** all'angolo α
BC e' il lato **opposto** all'angolo α
AC e' l'**ipotenusa.**

Il SENO, COSENO e la TANGENTE dell'angolo α sono definiti come segue:

$$\text{SIN } \alpha = \frac{\text{lato opposto}}{\text{ipotenusa}}$$

$$\text{COS } \alpha = \frac{\text{lato adiacente}}{\text{ipotenusa}}$$

$$\text{TAN } \alpha = \frac{\text{lato opposto}}{\text{lato adiacente}}$$

TUTTE le funzioni trigonometriche del BASIC si assumono misurate in **radianti.** Un **radiante** e' una misura di un angolo in unita' circolari. Un cerchio e' costituito da 360 gradi o 2π **radianti.** (π e' una lettera greca che rappresenta il numero costante 3.1415926). Quindi la conversione e' la seguente:

```
gradi/(180/π) = radianti
radianti*(180/π) = gradi
```

L'inverso di una funzione si ottiene eseguendo l'operazione contraria a quella eseguita per il calcolo della funzione. Per esempio la tangente dell'angolo di 1.5 radianti e

```
TAN(1.5) = 14.101419
```

L'inverso della funzione TAN e' ATN che si adopera per ottenere l'angolo se si conosce gia' la tangente,

```
ATN(14.101419) = 1.5
```

Gli inversi delle funzioni SIN e COS non sono disponibili in BASIC ma possono essere ottenuti usando la funzione ATN nella formula seguente:

```
Inverso di Seno    = (ATN(X/SQR(-X*X+1)))
Inverso di Coseno  = (ATN(X)SQR(-X*X+1))+1.5708
```

INDICE

INDICE

INDICE

INDICE

INDICE

INDICE

INDICE

INDICE

INDICE

DRAGON - ULTERIORI INFORMAZIONI

Questa appendice contiene alcune informazioni addizionali che ci auguriamo vi permettano di ottenere il meglio dal vostro microcomputer Dragon. Particolare importanza riveste il paragrafo che riguarda la gestione del registratore a cassette, in quanto presenta delle differenze rispetto alle informazioni date nel manuale.

Sono state anche inserite informazioni sul collegamento e la regolazione del TV, dettagli sul collegamento della stampante e una copia della mappa di memoria del Dragon.

TELEVISIONE

Una volta collegato il vostro computer ad una televisione e dopo averlo acceso, selezionate un canale libero e regolatelo, come indicato nel manuale. Se l'immagine sul video e' instabile e' necessario regolare la sintonia verticale del TV. Se non esiste una regolazione esterna per la sintonia verticale rivolgetevi al vostro tecnico TV. Per ottenere un'immagine chiara regolate il contrasto, la luminosita' ed il colore secondo le vostre esigenze.

REGISTRATORE A CASSETTA

Per collegare il registratore a cassetta al computer usando il cavo originale Dragon, inserite lo spinotto DIN nella presa contrassegnata TAPE sul lato sinistro del computer. I tre spinotti all'altro capo del cavo vanno collegati al registratore come segue:

i) Lo spinotto jack piu' piccolo con il filo grigio va inserito nella piccola presa jack di solito contrassegnata con REM e situata accanto alla presa microfono.
ii) Lo spinotto jack con il filo grigio va inserito nella presa di solito contrassegnata con AUX o MIC o LINE IN. Se esiste una possibilita' di scelta tra AUX e MIC, usate sempre AUX.
iii) Lo spinotto jack con il filo nero va inserito nella presa contrassegnata con EAR o MONIT, o L/S o SPKR.

Quando usate i comandi di "rewind" o "forward" sul registratore, potrebbe essere necessario staccare lo spinotto jack dalla presa REM perche' i comandi possano operare.

Assicuratevi di inserire questo spinotto dopo l'operazione.

In alternativa, battete MOTOR ON e premete ENTER prima di usare il "rewind" o il "forward". Al termine battete MOTOR OFF e premete ENTER per continuare ad operare con il registratore comandato dal computer.

Se desiderate riutilizzare una cassetta e' raccomandabile che la cassetta venga cancellata prima di registrarci nuovamente sopra.

STAMPANTE

La presa stampante predisposta sul lato sinistro del Dragon e' per una stampante che utilizza un'interfaccia di tipo parallelo centronics (la presa e' contrassegnata dal numero 6 nell'illustrazione contenuta nelle prime pagine del manuale). I collegamenti dei pin sono i seguenti:

PIN 1	Print Strobe	PIN 2	+5 volts
PIN 3	Data bit 0	PIN 4	+5 volts
PIN 5	Data bit 1	PIN 6	0 volts
PIN 7	Data bit 2	PIN 8	0 volts
PIN 9	Data bit 3	PIN 10	0 volts
PIN 11	Data bit 4	PIN 12	0 volts
PIN 13	Data bit 5	PIN 14	0 volts
PIN 15	Data bit 6	PIN 16	0 volts
PIN 17	Data bit 7	PIN 18	0 volts
PIN 19	$\overline{\text{ACK}}$	PIN 20	BUSY

I pin con numeri dispari sono situati nella riga superiore del connettore e quelli con numero pari nella riga inferiore.

CARTRIDGE

E' opportuno assicurarsi che il computer sia spento quando viene inserito o disinserito un cartridge (porta situata a destra del computer).

MAPPA DI MEMORIA DEL DRAGON

Indirizzo decimale	Contenuto	Indirizzo Esadecimale
0-1023	System Use	0-3FF
255	Direct Page RAM	0FF
1023	Extended Page RAM	3FF
1024-1535	Text Screen Memory	400-5FF
	Graphic Screen Memory	
1536-3071	Page 1	600-BFF
3072-4607	Page 2	C00-11FF
4608-6143	Page 3	1200-17FF
6144-7679	Page 4	1800-1DFF
7680-9215	Page 5	1E00-23FF
9216-2559	Page 6	2400-29FF
2560-12287	Page 7	2A00-2FFF
12228-13823	Page 8	3000-35FF
13824-32767	Program and Variable Storage	3600-7FFF
32768-49151	Basic Interpreter	8000-BFFF
49152-65279	Cartridge Memory	C000-FEFF
65280-65375	Input/Output	FF00-FF5F
65376-65503	SAM Control bits	FF60-FFDF
65504-65535	MPU Vectors	FFE0-FFFF

DRAGON 64 SUPPLEMENTO

INDICE

DRAGON 64 Supplemento

Il Dragon 64 e' stato progettato in modo da garantire la compatibilita' verso l'alto del Dragon 32 ed allo stesso tempo offrire una macchina con maggiori possibilita'. Le possibilita' in piu' sono:

1. Ulteriori 32K di RAM.

2. Una interfaccia (seriale) RS232.

3. Tasti con la ripetizione automatica.

Dato che il Dragon 64 e' pressoche' identico al Dragon 32 negli altri aspetti, vengono descritti in questo supplemento soltanto le caratteristiche aggiuntive.

1. Uso della RAM supplementare

Appena acceso il Dragon 64 e' configurato come il Dragon 32, cosi' che il software esistente, sia esso su nastro o su cartridge, puo' essere adoperato come prima. I 32K di RAM supplementare possono essere "inseriti" nella macchina invocando una routine di "bootstrap" che attiva la RAM supplementare e copia l'interprete BASIC negli indirizzi piu' alti dei 64K di RAM. Questo bootstrap viene attivato battendo:

```
EXEC
```

se non sono stati usati altri EXEC, o

```
EXEC 48000
```

se occorre rimpiazzare un precedente indirizzo EXEC.

Ci si puo' rendere conto di essere in modo 64K invece che in modo 32K dal fatto che il cursore lampeggiante e' blu invece che nero.

I 32K di RAM in piu' si sovrappongono agli indirizzi BASIC normali e a quelli della ROM su cartridge, il che significa che il software su cartridge non puo' essere usato nel modo 64K. Il software su cassetta, comunque, sia in BASIC che in linguaggio macchina, puo' utilizzare l'ulteriore RAM disponibile. Poiche' l'interprete BASIC in questo modo e' residente in RAM ed occupa la parte superiore della RAM, rimangono 48K liberi per il sistema e per l'utente e la quantita' di RAM disponibile per il programmatore in BASIC diviene 41241 bytes, sebbene possa naturalmente essere estesa per mezzo di una opportuna istruzione PCLEAR. Per esempio, quando si esegue un programma interamente in linguaggio macchina l'interprete BASIC non e' richiesto e di conseguenza e' disponibile l'intero spazio di 64K RAM.

La mappa di memoria del Dragon 64 nel modo 64K e' riportata in fondo a questo supplemento.

2. Uso dell'interfaccia RS232

Una interfaccia seriale RS232 e' in dotazione al Dragon 64 e puo' essere usata sia nel modo 32K che in quello 64K. Questa interfaccia viene usata per mezzo dei seguenti comandi BASIC:

```
DLOAD"nome del file",<codice di selezione del "baud rate"
                      (velocita' di trasmissione)>
```

e

```
DLOADM"nome del file",<codice di selezione del "baud rate">,
                      <load offset>
```

DLOAD e DLOADM scaricano rispettivamente dal computer programmi in BASIC ed in linguaggio macchina in formato ASCII. I parametri di questi comandi sono opzionali come per i loro equivalenti (CLOAD e CLOADM) usati con il registratore.

Il <codice di selezione del baud rate>, come il nome suggerisce, e' usato per selezionare la velocita' di trasmissione nella comunicazione fra computers. I valori permessi per questo parametro sono elencati qui di seguito.

codice di selezione del baud rate	baud rate risultante
0	110
1	300
2	600
3	1200
4	2400
5	4800
6	9600

Se questo parametro e' omesso viene assunto l'ultimo valore stabilito, e, in mancanza di questo, si assume una velocita' di 1200 bauds.

In BASIC il baud rate dell'interfaccia RS232 puo' essere alterato tramite un'opportuna istruzione POKE, per esempio:

```
POKE&HFF07,(PEEK(&HFF07) AND &HF0) OR B
```

dove la variabile B contiene un valore che specifica uno dei baud rates supportati dalla macchina.

B	Baud Rate
1	50
2	75
3	110
4	135
5	150
6	300
7	600
8	1200
9	1800
10	2400
11	3600
12	4800
13	7200
14	9600

I dati seriali vengono trasmessi e ricevuti come 1 start bit, 8 data bits, 2 step bits e nessun bit di parita'. Percio' le unita' collegate all'interfaccia, come per esempio stampanti seriali, devono adattarsi a questa configurazione.

2.1 Uso di una stampante seriale

L'interfaccia RS232 puo' anche essere usata come interfaccia standard per stampante in luogo dell'usuale interfaccia parallela Centronics. Per selezionare l'opzione seriale si puo' usare la seguente istruzione:

```
POKE &H3FF,1
```

e per selezionare l'opzione parallela (default)

```
POKE &H3FF,0
```

Oltre al byte di selezione della stampante esistono altri due bytes (&H3FD e &H3FE) che specificano in 16 bit un periodo di ritardo di fine riga, dato che certe stampanti lo richiedono. Il periodo di ritardo e' dato in incrementi di 10 millisecondi. Per esempio:

```
POKE &H3FE,50
```

provochera' un ritardo di mezzo secondo.

Una volta selezionata l'opzione della stampante seriale, i comandi BASIC LLIST e PRINT#-2 invieranno caratteri in output attraverso l'interfaccia RS232 piuttosto che per mezzo dell'interfaccia Centronics.

2.2 Uso dell'interfaccia RS232 da BASIC

L'interfaccia RS232 puo' essere utilizzata direttamente da BASIC indirizzando, tramite le istruzioni PEEK e POKE, appropriati registri dell'unita' RS232. Per esempio, il seguente programmino BASIC mandera' un carattere in output dalla RS232:

```
10 REM WAIT UNTIL TX DATA REGISTER IS EMPTY
20 IF (PEEK(&HFF05) AND 16)=0 THEN 20
30 REM NOW SEND THE CHARACTER OUT
40 POKE &HFF04,ASC(CH$)
```

Con un sistema simile si puo' leggere un carattere dall'interfaccia RS232.

```
10 REM WAIT UNTIL RX DATA REGISTER IS FULL
20 IF(PEEK(&HFF05) AND 8)=0 THEN 20
30 REM NOW ACCEPT THE CHARACTER
40 CH$=CHR$(PEEK(&HFF04))
```

Lo schema del connettore RS232 e' riportato in fondo al supplemento.

3. Dispositivo di "auto-repeat" della tastiera

Per evitare l'incompatibilita' con il software Dragon 32 esistente, il dispositivo di auto-repeat viene attivato soltanto nel modo 64K. Comunque, esso puo' essere attivato nel modo 32K come segue:

```
POKE &HFF03,(PEEK(&HFF03) AND &HFE)
POKE &H10D,&HBF
POKE &H10E,&H20
POKE &HFF03,(PEEK(&HFF03) OR 1)
```

Il tempo di riferimento per la velocita' di ripetizione deriva dalla frequenza (50 Hz) di ciclo dell'alimentazione principale e una locazione di memoria RAM (&H11F) contiene il valore di ritardo fra una ripetizione e l'altra. Questa locazione contiene un valore di default pari a 5 che determina un auto-repeat di 10 caratteri per secondo. Questo stesso valore viene anche usato per controllare il ritardo prima che inizi la ripetizione, ma in questo caso il suo valore e' moltiplicato per un fattore 8 che determina un ritardo di default pari a 0.8 secondi prima che l'auto-repeat inizi.

4. Differenze tra il BASIC del Dragon 32 e quello del Dragon 64

Le differenze piu' importanti fra queste 2 macchine sono state gia' descritte nei paragrafi precedenti di questo supplemento. Tuttavia, vi sono alcuni aggiornamenti al BASIC che vengono descritti qui sotto:

4.1 Chiamate USR

Nel Dragon 32 tutte le chiamate USR si riconducono a USR0. Nel Dragon 64 le chiamate USR vengono trattate correttamente sia nel modo 32K che nel modo 64K.

4.2 MEM e VARPTR

Poiche' nel modo 64K sono disponibili piu' di 32K di RAM le funzioni VARPTR e MEM sono state alterate in modo da poter trattare il loro risultato di 16 bits come un numero senza segni. Questo significa che, per valori compresi tra 32768 e 65535, il risultato che esse ritornano non e' un numero negativo.

MAPPA DI MEMORIA DEL DRAGON 64 NEL MODO 64K

Decimal Address	Contents	Hex Address
0-1023	System use	0-3FF
255	Direct Page RAM	0FF
1023	Extended Page RAM	3FF
1024-1535	Text Screen Memory	400-5FF
	Graphic Screen Memory	
1536-3071	Page 1	600-BFF
3072-4607	Page 2	C00-11FF
4608-6143	Page 3	1200-17FF
6144-7679	Page 4	1800-1DFF
7680-9215	Page 5	1E00-23FF
9216-10751	Page 6	2400-29FF
10752-12287	Page 7	2A00-2FFF
12288-13823	Page 8	3000-35FF
13824-49151	Program and Variable Storage	3600-BFFF
49152-65279	BASIC Interpreter	C000-FEFF
65280-65375	Input/Output	FF00-FF5F
65376-65503	SAM Control bits	FF60-FFDF
65504-65535	MPU vectors	FFE0-FFFF

PIN-OUT CONNETTORI DELL'INTERFACCIA RS232 DEL DRAGON 64

DRAGON

GUIDA RAPIDA

OPERATORI MATEMATICI E LOGICI

Simbolo	Operazione	Precedenza
^	Esponenziazione	1
-	Meno per indicare numeri negativi	2
*	Moltiplicazione	3
/	Divisione	3
+	Addizione	4
-	Sottrazione	4
>	Maggiore di	5
<	Minore di	5
=	Uguale a	5
<>	Non uguale a	5
>=	Maggiore di o uguale a	5
<=	Minore di o uguale a	5
NOT	"NON" logico	6
AND	"E" logico	6
OR	"O" logico	6

L'unico operatore matematico utilizzabile con le stringhe e' +, l'operatore della concatenazione.

Gli operatori relazionali possono essere utilizzati per confrontare stringhe e numeri.

TASTI DI COMANDO

[<-] Backspace. Cancella l'ultimo carattere

[SHIFT][<-] Cancella la riga attuale

[BREAK] Interrompe qualunque operazione in corso e restituisce il controllo alla tastiera

[CLEAR] Pulisce il video

[ENTER] Ritorno del carrello, fine della riga attuale di input

[SHIFT][@] Interrompe provvisoriamente l'esecuzione del programma. Premete qualunque tasto per ricominciare

[SHIFT][0] Commutatore maiuscole/minuscole

[BARRA SPAZIATRICE] Inserisce uno spazio

ISTRUZIONI NEL LINGUAGGIO BASIC

CLEAR n,h

Riserva n bytes di spazio di memoria per stringhe e cancella tutte le variabili. h specifica l'indirizzo BASIC piu' alto.
CLEAR 500

CLS c

Pulisce il video e lo setta del colore c specificato
0 - Nero
1 - Verde
2 - Giallo
3 - Blu
4 - Rosso
5 - Beige
6 - Azzurro
7 - Magenta
8 - Arancione

DATA

Memorizza all'interno del programma dati che vengono poi letti con l'istruzione READ.
DATA 4,7,14.2,SOGGETTO

DEF FN

Definisce una funzione numerica dell'utente.
DEF FNA(X)=X*X+3*X

DEFUSR n

Punto di ingresso (entry point) definito per la funzione USR n, n=0-9
DEFUSR2=14000

DIM

Dimensiona uno o piu' arrays.
DIM X(40), A$(7,6), B(10,2)

EXEC indirizzo

Trasferisce il controllo ai programmi in linguaggio macchina all'indirizzo indicato. Se l'indirizzo viene omesso si assume l'ultimo indicato in CLOADM.
EXEC 45043

FOR TO STEP NEXT

Crea un loop di programma che viene eseguito per la gamma di valori indicata. STEP indica l'incremento. Se STEP e' omesso si assume il valore 1.
FOR X=1 TO 10.....NEXT X
FOR A=1.3 TO 7.6 STEP 0.1.....NEXT A
FOR G=50 TO 10 STEP -10.....,NEXT G

GOSUB

Richiama una subroutine che comincia al numero di riga indicato
GOSUB 500

```
GOTO            Provoca un immediato salto del programma al
                numero di riga specificato
                GOTO 45

IF condizione THEN azione 1 ELSE azione 2
                Verifica la condizione; se vera esegue l'azione
                1, se falsa esegue l'azione 2
                IF A<3 THEN 200
                IF B-C>0 THEN X=X+1 ELSE Y=Y-1

INPUT           Provoca una pausa nel programma in attesa di
                input dalla tastiera
                INPUT"INSERISCI UN NOME";N$
                INPUT A, B, C, D

LET             Assegna un valore ad una variabile (opzionale)
                LET X=47.4

LINE INPUT      Permette l'inserimento di una riga dalla
                tastiera, comprese le virgole. Si termina la
                riga con [ENTER].
                LINE INPUT"TITOLO";T$(
                                                        (((((((
ON...GOSUB      Salto multiplo alle subroutines che iniziano
                alle righe specificate
                ON I GOSUB 100,200,300
ON...GOTO       Salto multiplo alle righe specificate
                ON K GOTO 245, 187, 310
POKE locazione, valore   Colloca il valore in una locazione
                specifica di memoria. Il valore deve essere
                fra 0 e 255
                POKE 27852,0
PRINT           Scrive sul video il contenuto della successiva
                lista. La virgola (,) provoca la tabulazione
                fino alla prossima zona di stampa di 16 colonne.
                Il punto e virgola (;) fa si che il cursore
                resti posizionato nel punto in cui e' stato
                scritto l'ultimo carattere
                PRINT"LA RISPOSTA"
                PRINT A, B
                PRINT"TU HAI FATTO";T;"PROVE"
PRINT TAB       Sposta il cursore alla colonna specificata
                PRINT TAB(10);"CREDITI"
PRINT USING     Scrive l'output secondo un formato specificato
                # campo numerico
                $ segno dollaro davanti al numero
                ** riempie con asterischi gli spazi davanti al
                   numero
                !!!! scrive in formato esponenziale
                %spazi% fissa la lunghezza del campo stringa a
                        spazi + 2
                + determina la scrittura del segno
                PRINT USING"##.##";A, B
                PRINT USING"%   %"; A$
```

PRINT@ locazione Scrive in una locazione specifica dello schermo (0-511)
PRINT@8,"PAGINA";N

READ Assegna ad una variabile specificata il valore successivo di una istruzione DATA.
READ A, B, C$

REM Permette l'inserimento di commenti in un programma. Tutto cio' che segue REM in una riga viene ignorato
REM QUESTA E' UNA RIGA DI COMMENTO

RESTORE Riposiziona il puntatore al primo valore della prima istruzione DATA
RESTORE

RETURN Riporta il programma dalla subroutine alla istruzione che segue GOSUB
RETURN

STOP Ferma l'esecuzione del programma alla riga che contiene STOP. Usate CONT per riprendere la esecuzione
STOP

ISTRUZIONI GENERATRICI DI SUONI

PLAY stringa Suona la stringa musicale composta di:
A-G (o 1-12), note
On, ottava n=1 a 5
Vn, volume n=1 a 31
Ln, lunghezza nota n=1 a 255
Tn, tempo n=1 a 255
Pn, pausa n=1 a 255
XA$; esegue la sottostringa contenuta in A$.
Permette anche l'impiego di diesis(#) o bemolle (-).
PLAY"O3L2GBO4CXY$;"

SOUND tono, durata. Suona il tono specificato (1-255) per la durata indicata.
SOUND 180,5

ISTRUZIONI DI CONTROLLO DEL REGISTRATORE

AUDIO — Attiva o disattiva l'output del registratore allo altoparlante TV.

```
AUDIO ON
AUDIO OFF
```

CLOAD — Carica il file di programma da cassetta. Se il nome del programma non e' specificato viene caricato il primo programma incontrato.

```
CLOAD
CLOAD"nome"
```

CLOADM — Carica un programma in linguaggio macchina da cassetta. Si puo' indicare l'offset dall'indirizzo di caricamento.

```
CLOADM
CLOADM"nome"
CLOADM"nome",2500
```

CLOSE — Chiude i files aperti

```
CLOSE
CLOSE#-1
```

CSAVE — Salva un programma su cassetta (il nome del programma non deve avere piu' di 8 caratteri). Se si specifica A, il programma viene salvato nel formato ASCII.

```
CSAVE"nome"
CSAVE"nome",A
```

CSAVEM nome, inizio, fine, ingresso — Salva un programma in linguaggio macchina su cassetta.

```
CSAVEM"nome", 4E, 6F, 5F
```

EOF(-1) — Ritorna TRUE (VERO) se e' stata incontrata la fine del file su cassetta.

```
IF EOF(-1) THEN 480
```

INPUT#-1 Input di dati da cassetta. Il file deve essere aperto.
INPUT#-1, A, B, C

MOTOR Attiva o disattiva il motore del registratore.
MOTOR ON
MOTOR OFF

OPENa,#-1, nome di file
Apre il canale al registratore, a può' essere "I" (input) o "O" (output). Nome di file e' il nome del file di dati.
OPEN"I",#-1,"DATA2"

PRINT#-1 Scrive dati su cassetta. Il file deve essere aperto.
PRINT#-1,X9,LN(J)

SKIPF Salta alla fine del programma su cassetta specificato.
SKIPF"Nome"

ISTRUZIONI DI CONTROLLO DELLA STAMPANTE

LLIST Stampa le righe di programma specificate.
LLIST
LLIST 10-95

OPEN"O",#-2, nome di file
Apre il canale di output alla stampante
OPEN"O",#-2,"OUTPUT"

PRINT#-2 Stampa la lista dei dati che segue l'istruzione.
PRINT#-2,W,A$;X

COMANDI DEL SISTEMA

CONT CONT riprende l'esecuzione del programma dopo una istruzione STOP o l'impiego del tasto BREAK.

DEL Cancella righe di programma

```
DEL 100-350
DEL 10-
DEL -80
```

EDIT Viene usato per modificare il contenuto di una certa riga.

```
EDIT 115 [ENTER]
```

Una volta nel modo EDIT, si possono usare i seguenti sottocomandi:

- nC Cambia n caratteri
- nD Cancella n caratteri
- I Inserisce caratteri
- H Cancella il resto della riga ed inserisce caratteri
- L Visualizza la riga allo stato attuale
- nSc Cerca c dove compare per l'ennesima volta
- X Estende la riga
- n<- Sposta il cursore n spazi verso sinistra
- [SHIFT][!] Esce dal sottocomando
- n[BARRA SPAZIATRICE] Sposta il sursore n spazi verso destra.

LIST Visualizza le righe specificate

```
LIST
LIST 10-95
LIST -200
```

NEW Cancella il programma dalla memoria.

RENUM nuova riga, riga iniziale, incremento
Permette la rinumerazione di tutte o di alcune delle righe di programma.

```
RENUM
RENUM 100, 50, 10
RENUM ,, 20
```

RUN Esegue un programma cominciando dalla riga col numeo piu' basso o dalla riga specificata

```
RUN
RUN 250
```

TROFF Ferma il tracciamento del flusso di programma

```
TROFF
```

TRON Avvia il tracciamento del flusso di programma

```
TRON
```

VARIABILI SEMPLICI

Tipo	Nome	Gamma
Numerica	AB	$10^{\pm 38}$
Stringa	AB$	da 0 a 255 caratteri

Dove A deve essere una lettera e B e' opzionale (cioe' puo' essere una lettera o un numero). Se il nome usato ha piu' di due caratteri, sltanto i primi due saranno considerati.

VARIABILI ARRAY

Tipo	Nome dell'elemento tipico
Numerica	AB(3,4,5)
Stringa	AB$(17,8,2)

I nomi e le gamme sono come quelli delle variabili semplici. I limiti per le dimensioni dell'array sono dati dallo spazio disponibile in memoria.

CARATTERI SPECIALI

'	Abbreviazione per REM
?	Abbreviazione per PRINT
:	Separatore di istruzioni sulla stessa riga
$	Aggiunto al nome della variabile la fa diventare di tipo stringa.

CODICI DI ERRORE

/0	Divisione per zero
AO	File gia' aperto
BS	Indice sbagliato
CN	Non si puo' continuare
DD	Tentativo di ridimensionare un array
DS	Istruzione diretta in un file
FC	Richiamo di funzione illegale
FD	Dati erronei
FM	Modo del file sbagliato
ID	Istruzione diretta sbagliata
IE	Input oltre la fine del file
I/O	Errore di input/output
LS	Stringa troppo lunga
NF	NEXT senza FOR
NO	File non aperto
OD	Dati esauriti in READ
OM	Memoria esaurita
OS	Spazio per stringhe esaurito
OV	Overflow, numero troppo grande
RG	RETURN senza GOSUB
SN	Errore di sintassi
ST	Formula stringa troppo complessa
TM	Tipo non congruente
UL	Riga non definita

ISTRUZIONI GRAFICHE

CIRCLE(x,y),r,c,h,s,e

Disegna un cerchio con centro (x,y) di raggio r e colore c. Rapporto altezza/larghezza h. (Il cerchio puo' iniziare a s e terminare ad e).(s ed e variano da 0 a 1: 0 e' uguale alle ore 3).

```
CIRCLE(128,96),25
CIRCLE(100,50),30,4,1,0,0.5
```

COLOR colore del primo piano, colore del fondo

Stabilisce i colori del primo piano e del fondo fra quelli disponibili.

```
COLOR 0,5
```

DRAW stringa

Traccia delle righe secondo le istruzioni contenute nella stringa come segue:

Mx,y	Sposta in x,y
Un	In su n punti
Dn	In giu' n punti
Ln	A sinistra n punti
Rn	A destra n punti
En	A 45 gradi n punti
Fn	A 135 gradi n punti
Gn	A 225 gradi n punti
Hn	A 315 gradi n punti
Ak	Ruota secondo l'angolo k. 0=0 gradi 1=90 gradi, 2=180 gradi, 3=270 gradi
Sk	Disegna in scala pari a k/4 (k fra 1 e 62)
N	Nessun aggiornamento della posizione di tracciamento
B	Spostamento senza segnare
C	Stabilisce il colore della riga
X	Esegue una sottostringa

GET(x1,y1)-x2,y2), array, G

Legge in array il contenuto grafico dell'area di video definita dal rettangolo con angoli in alto a sinistra (x1,y1) ed in basso a destra (x2,y2). G e' opzionale e specifica tutti i dettagli grafici

```
GET(0,0)-(24,24),X,G
```

LINE(x1,y1)-(x2,y2),a,b

Traccia una riga fra (x1,y1) e (x2,y2). Se (x1,y1) e' omesso si assume la posizione attuale. a deve essere o PSET (sceglie il colore del primo piano) o PRESET (sceglie il colore del fondo). b e' opzionale ma deve essere o B (traccia un quadrato con angoli (x1,y1) e (x2,y2)) o BF (riempie il quadrato con il colore del primo piano).
LINE(10,20)-(35,15),PSET
LINE(0,0)-(128,96),PSET,BF

PAINT(x,y),c,b

Colora il video grafico con il colore c, partendo dal punto (x,y) e fermandosi al bordo del colore b specificato.
PAINT(128,96),1,4

PCLEAR n Riserva n pagine di memoria grafica (n<=8).
PCLEAR 6

PCLS c Setta lo schermo ad alta risoluzione nel colore c. Vedete CLS per i codici di colore.
PCLS 5

PCOPY a TO b

Copia la pagina grafica a nella pagina di destinazione b.
PCOPY 3 TO 4

PMODE modo, pagina

Sceglie il modo di risoluzione (da 1 a 4) e la prima pagina di memoria (fra 1 e 8).
PMODE 4,1

PRESET (x,y)

Resetta il punto (x,y) nel colore del fondo.
PRESET (15,37)

PSET (x,y,c)

Setta il punto (x,y) nel colore c.
PSET (120,95,4)

PUT (x1,y1)-(x2,y2), array, azione

Mette il dettaglio grafico contenuto in array sul rettangolo di video definito dagli angoli in alto a sinistra (x1,y1) ed in basso a destra (x2,y2). L'azione e' opzionale e puo' essere PSET, PRESET, AND, OR o NOT.
PUT (A,C)-(A+24,C+24)

RESET (x,y)

Setta il punto (x,y) del video a bassa risoluzione nel colore del fondo.
RESET (10,12)

SCREEN tipo, gruppo

Sceglie il tipo di video (testo=0, grafica=1) ed il gruppo di colori (0 o 1).
SCREEN 1,1

SET (x,y,c)

Setta il punto (x,y) del video a bassa risoluzione nel colore c.
SET (15,30,8)

POS(d)	Restituisce la posizione di stampa attuale della unita' d. P=POS(-2)
PPOINT(w,z)	Verifica il punto grafico ad alta risoluzione (w,z). Restituisce il codice di colore se e' acceso, 0 se e' spento. P=PPOINT(186,54)
RND(n)	Genera un numero intero a caso fra 1 e n. Se n e' 0, il numero a caso e' fra 0 e 1. X=RND(18)
SGN(a)	Restituisce il segno del numero: 1 positivo, 0 zero, -1 negativo. G=SGN(4*H/3)
SQR(a)	Restituisce la radice quadrata del numero. X=SQR(A+7*C)
TAN(r)	Restituisce la tangente dell'angolo dato in radianti. Z=TAN(2*A3)
TIMER	Restituisce il valore attuale del timer, o permette la regolazione del timer. T=TIMER TIMER=0
USR(n)	Richiama una routine in linguaggio macchina definita dall'utente. F=USR(D)
VARPTR(var)	Restituisce l'indirizzo del puntatore alla variabile specificata. Z=USR(VARPTR(F))

FUNZINI NUMERICHE

Argomenti di funzione

a	valore numerico 10^{38}
n	numero intero 0-65535
r	angolo espresso in radianti
x,y	locazione del video a bassa risoluzione x 0-63 : y 0-31
w,z	locazione del video ad alta risoluzione w 0-255 : z 0-191

ABS(a) Calcola il valore assoluto.

```
AY=ABS(Y)
```

ATN(a) Restituisce l'arcotangente in radianti.

```
X=ATN(A)
```

COS(a) Restituisce il coseno dell'angolo dato in radianti

```
W=COS(2*B)
```

EXP(a) Restituisce l'esponenziale del numero, e^a.

```
Q=EXP(-5.6)
```

FIX(a) Restituisce il valore troncato.

```
Z=FIX(13.43)
```

INT(a) Trasforma il numero in un numero intero.

```
A=INT(5.7)
```

JOYSTK(a) Restituisce la coordinata verticale od orizzontale del joystick

0 = orizzontale, joystick di sinistra
1 = verticale , joystick di sinistra
2 = orizzontale, joystick di destra
3 = verticale , joystick di destra

```
H=JOYSTK(0)
V=JOYSTK(P)
```

LOG(a) Restituisce il logaritmo naturale

```
Z=LOG(7.842)
```

MEM Restituisce l'ammontare dello spazio disponibile in memoria.

```
PRINT MEM
```

PEEK(n) Restituisce il contenuto di una locazione di memoria specifica, n.

```
P=PEEK(65082)
```

POINT(x,y) Verifica il punto grafico a (x,y). Restituisce il codice di colore se e' acceso, 0 se e' spento o -1 se contiene un carattere di testo.

```
T=POINT(15,25)
```

FUNZIONI STRINGA

Gli argomenti di funzione N, M rappresentano numeri interi: X$, Y$ rappresentano stringhe.

ASC(X$) Restituisce il codice ASCII del primo carattere di una stringa.
```
X=ASC$(A$)
```

CHR$(N) Restituisce il carattere corrispondente al codice ASCII specificato (N).
```
A$=CHR$(143)
```

HEX$(N) Restituisce il valore esadecimale dell'argomento come stringa.
```
PRINT HEX$(46)
```

INKEY$ Controlla la tastiera e ritorna il carattere corrispondente al tasto premuto.
```
K$=INKEY$
```

INSTR(N,X$,Y$) Cerca la stringa Y$ all'interno della stringa X$ a partie dalla posizione N di X$. Se la trova ne restituisce la posizione (altrimenti 0).
```
P=INSTR(8,A$,"TED")
```

LEFT$(X$,N) Restituisce i primi N caratteri della stringa X$.
```
B$=LEFT$(A$,9)
```

LEN(X$) Restituisce la lunghezza della stringa X$.
```
k=LEN(K$)
```

MID$(X$,N,M) Restituisce la sottostringa di X$ di lunghezza M cominciando dalla posizione N.
```
B$=MID$(A$,4,1)
```

RIGHT$(X$,N) Retituisce gli ultimi N caratteri della stringa X$
```
B$=RIGHT$(A$,3)
```

STRING$(N,X$) Restituisce la stringa costituita da N copie del primo carattere di X$. Il numero di codice ASCII del carattere desideato puo' sostituire X$.
```
L$=STRING$(32,"+")
PRINT STRING$(5,41)
```

STR$ Trasforma N nella sua rappresentazione stringa.
```
X$=STR$(14.4)
```

VAL(X$) Trasforma i caratteri numerici di X$ in un numero.
```
P=VAL(D$)
```

V1.0 31/08/2020

Release Notes:

1) The original manual did not contain page 108, it went from 107 to 109.
2) Some pages are faded, which is a printing issue in the original.
3) The text on this document was created using OCR, please see the English version for more accurate text and listings.

www.DragonData.co.uk