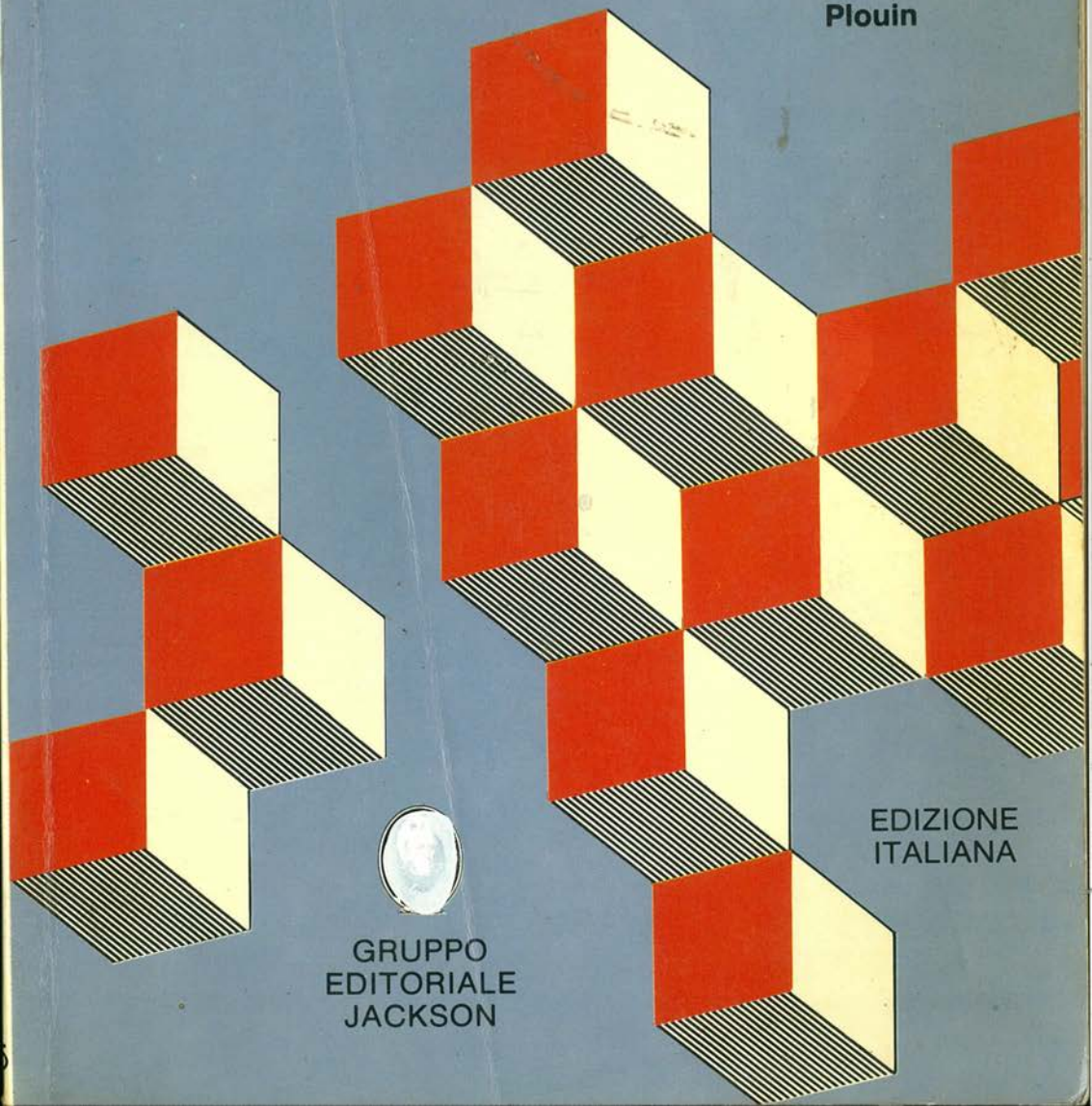
PROGRAMMARE IN BASIC

Michel Plouin



EDIZIONE ITALIANA

GRUPPO EDITORIALE JACKSON

PROGRAMMARE IN BASIC

di
Michel Plouin

GRUPPO
EDITORIALE
JACKSON
Via Rosellini, 12
20124 Milano
Tel. 680368 - 680054 - 68951/2/3/4/5

Il Gruppo Editoriale Jackson ringrazia per il prezioso lavoro svolto nella stesura dell'edizione italiana la signora Francesca Di Fiore e l'ing. Roberto Pancaldi.

Illustrazioni di Anglaïs Champetier

PREFAZIONE

Quest'opera si rivolge al principiante desideroso di apprendere il linguaggio più popolare nel mondo del personal e home computer.

Come tutte le "lingue viventi", il BASIC è applicato in realtà a questa o a quella macchina sotto forma di "dialetti" più o meno particolari. Questo libro si sforza di descrivere in modo metodico i BASIC delle tre macchine più diffuse: TRS 80; Apple; P.E.T. ed i loro derivati. Questo approccio comparato permette di approfondire la conoscenza del linguaggio e dà un'idea del grado di standardizzazione del BASIC. Ciò faciliterà anche la conversione di programmi scritti per altre macchine.

La maggior parte dei BASIC ha conosciuto delle estensioni successive, particolarmente per ciò che riguarda la gestione di archivi (files). L'uso dei dischi (e più spesso di mini dischi a grande capacità) è di gran lunga il più valido per una seria gestione di file; queste estensioni sono poco standard, soprattutto sulle macchine ad uso professionale e la loro descrizione non è prevista in questo libro.

Michel Plouin, *35 anni, ingegnere, è consigliere alla direzione d'informatica d'una impresa francese.*

Ha praticato la programmazione nell'insegnamento o in veste di consulente, ma anche per proprio gusto personale.

Questo passatempo che si avvicina ai giochi così detti nobili, come il bridge o gli scacchi, può avere sbocco in numerose realizzazioni concrete, e particolarmente in campo pedagogico.

DELLA STESSA COLLANA

Programmare in Assembler - Alain Pinaud
Programmare in BASIC - Michel Plouin
Il BASIC e i sui file - Jacques Boisgontier
Programmare in Pascal - Daniel-Jean David et Jean-Luc Deschamps
Come programmare - Jean Clau-de Barbance
La pratica dell'Apple II - Vol. I - Nicole Bréaud Pouliquen
Matematica e Statica - Programmi in Basic
Giochi, Trucchi e calcoli - (29 programmi in Basic)
La realizzazione dei Programmi - Michel Benel Foul

SOMMARIO

CAPITOLO 1

INTRODUZIONE

Il miglior modo per apprendere un linguaggio come il BASIC è di incominciare la lettura di questo libro con una macchina in funzione. È il grande vantaggio dei personal computer e conviene approfittarne. Il lettore che ha già famigliarità con la tastiera di un computer potrà passare al capitolo seguente.

Il tasto che non deve essere dimenticato

Il primo riflesso che deve essere acquisito riguarda il tasto di "ritorno linea" ("Enter" sul T.R.S., "Return" sull'Apple e sul P.E.T.). Si deve tener presente che la macchina non fa altro che obbedire (stupidamente?) a degli ordini e che è necessario anche avvertirla quando si è finito di esprimerne uno. Infatti ciascun comando o linea di programma impostati sulla tastiera vengono presi in considerazione solo dopo la pressione del tasto "Return" o "Enter". Per facilitare le cose, certi ordini possono essere dati battendo un solo tasto speciale, per esempio l'azzeramento dello schermo (tasto CLEAR sul T.R.S.. CLS sul P.E.T.). Certe macchine permettono anche di definire una serie di funzioni particolari mediante tasti speciali: ciascuno corrispondente ad un carattere speciale che la macchina riconosce come un ordine completo.

Altri tasti speciali

Un altro metodo permette di avere un gran numero di caratteri speciali: questi si ottengono con l'aiuto di un tasto "Control" (tasto CTRL dell'Apple) che rende speciali i tasti che vengono battuti mentre il suddetto tasto è abbassato. In questo caso si manipolano due tasti ma c'è una sola battuta ed un solo carattere emesso dalla tastiera verso la macchina. Il tasto "Control" (Controllo) può essere combinato con il tasto "Shift": si manipolano in questo caso tre tasti per generazione di un carattere. (esempio: cancellazione dello schermo sull'Apple con il CTRL - @ vale a dire CTRL-SHIFT-P). Questo metodo permette di ottenere 32 caratteri speciali (corrispondenti alle due prime linee della tabella ASCII in Appendice.) con un solo tasto supplementare in tastiera.

Un altro tasto particolare è il tasto "Escape" (ECS sull'Apple). Questo rappresenta un carattere speciale (tra quelli speciali...) indica che il carattere normale che segue rappresenta un ordine speciale. Ci sono allora due battute

successive e due caratteri emessi verso la macchina. Queste sequenze Escape, utilizzate dall'Apple, sono molto ricorrenti per comandare stampanti o terminali intelligenti.

Esercitatevi sulla tastiera

Se battete un tasto a caso, poi in tasto Enter/Return, otterrete in generale un messaggio di errore, poi il passaggio a capo con il carattere di "pronto". Questo carattere indica che la macchina è pronta e attende un ordine. Esso permette anche di riconoscere quale linguaggio è attivo quando la macchina ne possiede diversi.

Nel momento in cui si impostano dei caratteri, la macchina provvede mano a mano alla visualizzazione e permette il ritorno indietro per eventuali correzioni.

Quando si batte il tasto Enter/Return, la macchina analizza l'ordine ricevuto e provvede alla sua esecuzione. L'esecuzione è immediata se l'ordine non comincia con caratteri numerici; se l'ordine comincia con una cifra, viene inserito in memoria con quel numero per la successiva esecuzione: così si scrive una linea di programma.

Il sistema di esecuzione immediata è chiamato **comando**; l'ordine, preceduto da un numero, è chiamato **istruzione**. La maggior parte degli ordini possono essere eseguiti sia come comandi che come istruzioni; si vedrà che ciò è molto pratico per mettere a punto i programmi. provate vari comandi: per esempio scrivete:

PRINT "CUCU" (Enter/Retur)
PRINT 256 + 1024 (Enter/Return)

Provate un programma costituito da due linee:

1 PRINT "CUCU (Enter/Return)
2 GOTO 1 (Enter/Return).

Non accadrà nulla! Esaminate invece il vostro programma scrivendo:

LIST (Enter/Return)

Attivate l'esecuzione scrivendo:

RUN (Enter/Return)

Provate a fermare... (sull'Apple Control-C).

Non abbiate paura di fare esperimenti. Non c'è rischio di sciupare nulla nel battere (moderatamente) la tastiera. La cosa peggiore che vi può succedere è di dover spegnere e poi riaccendere la macchina se non riuscite a riprenderne il controllo.

Il microprocessore

Un linguaggio per l'informatica serve a fare funzionare una macchina dando degli ordini a un dispositivo di calcolo (un microprocessore nelle macchine che ci interessano) associato a diverse risorse come le memorie e le periferiche.

Il lavoro del microprocessore consiste nell'andare a cercare un ordine immagazzinato nelle memorie, riconoscerlo ed eseguirlo. Esso può realizzare diverse operazioni elementari alla cadenza di 100.000 cicli al secondo, ma è necessario fornirgli ogni spiegazione.

La programmazione

Un linguaggio evoluto, come il BASIC, permette di esprimere questi ordini elementari in modo sintetico, liberando il programmatore da compiti fastidiosi, grazie ad un vocabolario e a regole d'impiego molto facili da tenere a mente.

La nozione di **variabile**, affrontata nel Capitolo 2, libera il programmatore dalla minuziosa gestione della memoria. Essa permette di dare un nome simbolico alle diverse informazioni che si possono in tal modo manipolare senza preoccuparsi di dove esse sono immagazzinate nella memoria.

Queste informazioni, delle quali si può disporre con la semplice menzione del loro nome, saranno manipolate per mezzo delle istruzioni di calcolo e delle **funzioni** descritte nel **Capitolo 3,** funzioni numeriche e funzioni che lavorano su caratteri alfanumerici.

Lo **svolgimento** delle azioni comandate da tutte queste istruzioni non viene fatto a caso: una famiglia di **istruzioni logiche**, descritte nel **Capitolo 4**, consente di dare una struttura logica ai programmi. Queste istruzioni permettono di fare dei test, di andare mediante passi successivi da una parte all'altra di un programma, di ripetere più volte una serie di operazioni.

Inoltre è necessario **fare comunicare la macchina** con l'esterno essenzialmente per mezzo di una tastiera di uno schermo e, eventualmente, di altri dispositivi di "I/O" (entrata-uscita) o periferici: questo è l'oggetto del **Capitolo 5**, che descrive le istruzioni di dialogo con la macchina ed il suo programma.

Il BASIC dispone di potenti **funzioni speciali** che numerosi linguaggi evoluti utilizzati nell'informatica professionale non hanno: simulazione con numeri casuali, comunicazione diretta con la memoria. La chiamata di un sottoprogramma in linguaggio macchina esiste in genere anche negli altri linguaggi evoluti. Ciò è descritto nel **Capitolo 6**.

I piccoli sistemi normalmente permettono anche effetti grafici disponibili di solito su qualche sistema professionale specializzato e costoso. Questi effetti speciali sono esaminati nel **Capitolo 7**; in questo campo, l'evoluzione tecnica è rapidissima e le novità all'ordine del giorno.

Si constaterà che la standardizzazione dei BASIC pressochè completa per le istruzioni elementari, scompare quando si desiderano sfruttare tutte le possibilità particolari di un **sistema** (effetti grafici colore, effetti acustici). Ci siamo impegnati a sperimentare queste varianti e a confrontare le macchine più diffuse; non abbiamo descritto certe funzioni troppo specifiche, come la memorizzazione dei files su cassette, le possibilità di editing, le funzioni dei sistemi dotati di dischi (DOS) che vanno oltre lo scopo di quest'opera.

Il **Capitolo 8**, "Preparazione dei programmi", descrive i problemi che si pongono al momento della concezione di un programma, della sua messa a punto e della sua ottimizzazione.

All'appendice 1 troverete una descrizione del codice ASCII che è la base di ogni scambio d'informazione tra macchine e periferiche; esso standardizza la codificazione in binario delle informazioni alfanumeriche. In linea di massima, i caratteri speciali, che servono a ordinare differenti azioni collegate alla macchina, dovrebbero inserirsi in questo codice: abbiamo descritto i caratteri speciali e le funzioni dei tasti di tre delle macchine più diffuse (Apple II, P.E.T. e TRS-80).

Per comprendere bene la rappresentazione dell'informazione per mezzo di un codice, o più generalmente ciò che è scritto nella memoria di un computer, è necessario avere qualche nozione di calcolo binario o esadecimale, che sono brevemente affrontate nell'Appendice 2.

Inoltre, abbiamo riunito nell'appendice 3 degli esempi di programmi. Si tratta di programmi completi che illustrano alcune possibilità di applicazione e che sono raccompandati allo scopo di permettere la comprensione e l'adattamento da parte del lettore.

Troverete infine un repertorio di istruzioni e parole chiave in BASIC, che si intende completo per certo numero di Personal Computer attualmente disponibili. Sono indicate le eventuali particolarità e con l'aiuto di esempi sono presentate le funzioni dell'istruzione e il suo formato.

CAPITOLO 2

LE VARIABILI

Il principale vantaggio dei linguaggi evoluti è quello di dispensare il programmatore dalla manipolazione dei dati direttamente in memoria.

Con il BASIC basta dar loro un nome senza conoscere il luogo esatto dove l'informazione verrà posta. Ecco perchè anche questi linguaggi vengono chiamati simbolici.

Questa nozione di variabile non ha niente a che vedere con le variabili e le incognite utilizzate nelle funzioni e nelle equazioni algebriche.

Essa designa semplicemente un'informazione chiamata con un nome, in opposizione a una costante che è un'informazione (numeri e caratteri) che appare semplicemente nell'istruzione in corso di esecuzione.

Bisogna rispettare certe regole (restrittive) nella scelta dei nomi.

Per contro, si possono scegliere parecchi metodi di immagazzinamento delle informazioni nella memoria e certe istruzioni speciali permettono di conoscere la locazione esatta assegnata dal BASIC che tiene aggiornate in permanenza le tabelle delle variabili incontrate dall'inizio dell'esecuzione di un programma.

```
10 REM ESEMPI DI VARIABILI
20 I1 = J + K
30 N = N + 1
40 A% = INT(A) : A$ = "+"
50 REM A,A% ED A$ SONO VARIABILI DISTINTE
60 REM 1 (2,30) E "+" (2,40) SONO DELLE COSTANTI
```

```
10 REM ESEMPI DI PAROLE RISERVATE
20 T = TOTALE + T1:REM ERRORE FACILE DA RILEVARE
30 N1 = NOTA:REM ERRORE PIU' SUBDOLO
40 REM DA' N1 = OPPOSTO LOGICO DI A (NOTA E NOT A CONFUSI)
```

```
10 REM ESEMPIO DI CONFUSIONE TRA 2 NOMI
20 NUM = NUMERO + 1      :REM ERRORE NON SEGNALATO
30 REM EQUIVALE A NU = NU + 1
```

```
10 REM SI DISPONE DI PARECCHIE MIGLIAIA DI NOMI
20 REM A1 - Z9 PIU' AA - ZZ FANNO 936 COMMBINAZIONI
30 REM SI ARRIVA CON 4 FORMATI A 3744  ( % $ # )
40 REM E ANCORA PIU' CON INSIEMI  ( T <> T(1)...)
```

```
10 REM TIPO DEFINITO ESPLICITAMENTE CON SUFFISSO
20 I% = 3  :REM FORMATO INTERO - 2 BYTES
30 A$ = "Z":REM FORMATO STRINGA (1 BYTE PER CARATTERE)
40 REM SUFFISSI PARTICOLARI AL TRS :
50 REM D# (DOPPIA PRECISIONE - 16 CIFRE), R! (REALE SEMPLICE PRECISIONE -
   6 CIFRE)
```

```
10 REM OTTIMIZZAZIONE : VELOCITA' E OCCUPAZIONE DI MEMORIA SONO
20 REM 2 CRITERI ANTAGONISTI ( SI PUO' SEMPRE MIGLIORARE L'UNO
 A DETRIMENTO DELL'ALTRO)
30 REM OCCUPAZIONE DI MEMORIA : UTILIZZARE NOMI CORTI,POCO NUMEROSI
40 REM VELOCITA' : LA TAVOLA DEI NOMI E' COSTITUITA NELL'ORDINE
50 REM D'APPARIZIONE NEL PROGRAMMA. SI FARA' IN MODO
60 REM CHE LE VARIABILI CHIAMATE FREQUENTEMENTE APPAIANO
70 REM IN PRIMO. LE COSTANTI RALLENTANO L'ESECUZIONE
80 REM IMMAGAZZINARLE ALL'INIZIO IN UNA "VARIABILE"  (ES. N1 = 1 IN PRINCIPIO)
```

Nomi delle variabili

Nei personal computer più correnti, il BASIC si avvale al massimo dei nomi di due caratteri (uno solo per TRS livello I). Il BASIC ignora generalmente i caratteri che seguono i nomi più lunghi; è pericoloso utilizzare quest'agevolazione a causa del rischio di confusione se due nomi incominciano con due identici caratteri.

D'altra parte, i nomi non devono contenere delle combinazioni di caratteri che formino una parola del linguaggio (istruzioni, comandi) che si chiamano per questo motivo **parole riservate**. Così TOTAL sarà respinto perchè i due primi caratteri formano la parola riservata TO. Con i nomi di uno o due caratteri, il rischio è minimo; TO, IF, ON, OR e AT in BASIC sono generalmente, parole riservate.

È grazie a questa regola delle parole riservate che il BASIC può ignorare gli spazi nelle istruzioni. L'omissione degli spazi non aiuta la chiarezza dei programmi, ma economizza l'ingombro se la disponibilità della memoria è limitata.

Nei BASIC che autorizzano nomi di variabili con più di due caratteri è più facile avere dei nomi espliciti (per esempio Somma al posto di S) ed i programmi risultano quindi più chiari; ma sono anche molto più voluminosi, ivi comprese le tabelle gestite dal BASIC.

Il suffisso che definisce un tipo di variabile (vedere formati) fa parte integrante del nome e aumenta il numero di variabili disponibili simultaneamente; per esempio A, A% e A$ sono tre variabili distinte. Sarà dunque raro il caso di limitazione del numero dei nomi possibili. Ciò nonostante, l'utilizzazione dei nomi mnemonici porta a diminuire gradualmente il numero delle possibilità, (W, Y, H sono delle lettere poco utilizzate come abbreviazioni).

```
10 REM TIPI DI VARIABILI
20 REM IN ASSENZA DI INDICAZIONI IL TIPO E' REALE (IMPLICITO)
30 REM % SUFFISSO DI VARIABILE INTERA
40 REM $ "  "  "STRINGA
50 REM
60 REM ALTRI SUFFISSI NON STANDARD
70 REM ! REALE SEMPLICE PRECISIONE  (TRS)
80 REM # REALE DOPPIA PRECISIONE  (TRS)
90 REM ES.
100 I% = 3                    :REM OCCUPA 2 BYTES IN MEMORIA
110 I% = 55000 :REM ERRORE : LIMITE - 32768 A 32767
120 A$ = "BUONGIORNO"
130 PI = 3.1415926535:REM CHE VOGLIO INSEGNARE UN NUMERO
  UTILE AI PIU' BRAVI
```

```
10  REM DEFINIZIONE ESPLICITA (TRS)
20 DEFINT A,I-N              :REM A1,AZ,I,LP,N9 SONO INTERI
30 DEFSTR AB,Z               :REM AB$Z = STRINGHE. AA,AC RESTANO INTERI
40 DEFDBL D                  :REM DOPPIA PRECISIONE ( 16 CIFRE , 8 BYTES)
50 DEFSNG S                  :REM SEMPLICE"( 6 CIFRE , 4 BYTES )
60 REM IL NOME DELLE VARIABILI OCCUPA TRE BYTES OLTRE I DATI
```

```
10 REM ACCESSO AL DATO IN MEMORIA (TRS)
20 P = 3.14
30 A = VARPTR (P)            :REM PUNTATORE DELLA VARIABILE
40 PRINT PEEK (A),PEEK (A+1),PEEK (A+2),PEEK (A+3)
!   !   !
  CIFRE MENO SIGNIFICATIVE !  ESPONENTE + 128
 CIFRE PIU' SIGNIFICATIVE
50 Z$ = ""
60 A = VARPTR (Z$)           :REM ACCESSO ALLA LISTA DELLE CATENE
70 PRINT PEEK (A)            :REM LUNGHEZZA (IN QUESTO CASO 6 )
80 A1 = PEEK (A+1) + 256*PEEK (A+2)      :REM INDIRIZZO DELL' INIZIO
90 PRINT CHR$ (PEEK (A1))                :REM LETTERA "C"
```

Formati e tipi di variabili

I differenti formati di variabili (o di costanti) permettono di codificare, sia per il programmatore che per il processore che lavora a livello linguaggio-macchina, il modo in cui le differenti informazioni immagazzinate o manipolate sono rappresentate in memoria.

Prima di tutto esiste un formato per i numeri interi algebrici (cioè con un segno); è il formato unico per BASIC limitati (esempio Apple con l'Integer Basic). Nei BASIC usuali, una variabile il cui nome contiene il suffisso % è trattata in formato intero. Le operazioni sui numeri interi hanno per risultato un numero intero, specialmente la divisione che da un quoziente tronco, non arrotondato.

L'interprete BASIC, per trattare un numero che ha riconosciuto di formato intero, riserva automaticamente due byte, il che limita i numeri interi ai valori da −32767 a + 32767 (cf. appendice: calcolo esadecimale). Una costante senza punto decimale (nel BASIC il punto anglosassone prende il posto della virgola) è ritenuta intera; la stessa costante sarà trattata come reale se si aggiungerà un

punto decimale: per esempio 3. al posto di 3 semplicemente.

Con alcuni BASIC ci si può liberare dal suffisso % che è poco piacevole da trattare, definendo come interi tutta una categoria di nomi di variabili.

DEFINT I - N ricorderà certe abitudini a coloro che sono abituati al FORTRAN, linguaggio nel quale sono implicitamente definite come intere le variabili i cui nomi cominciano per I, J, K, L, M o N. Questa definizione entra in gioco dopo l'esecuzione dell'istruzione (define Integer) e non per tutto il programma. Essa può essere messa in causa per ulteriori definizioni possono coesistere le variabili dotate di suffisso, e A% sarà distinto da A a meno che quest'ultimo sia stato dichiarato intero.

```
10 REM PROGRAMMA NON OTTIMALE ( MA CHIARO )
20 FOR I = 1 TO K + 25 STEP U
30 N = N + 1                :REM CONTATORE
40 PRINT A,B
50 NEXT
```

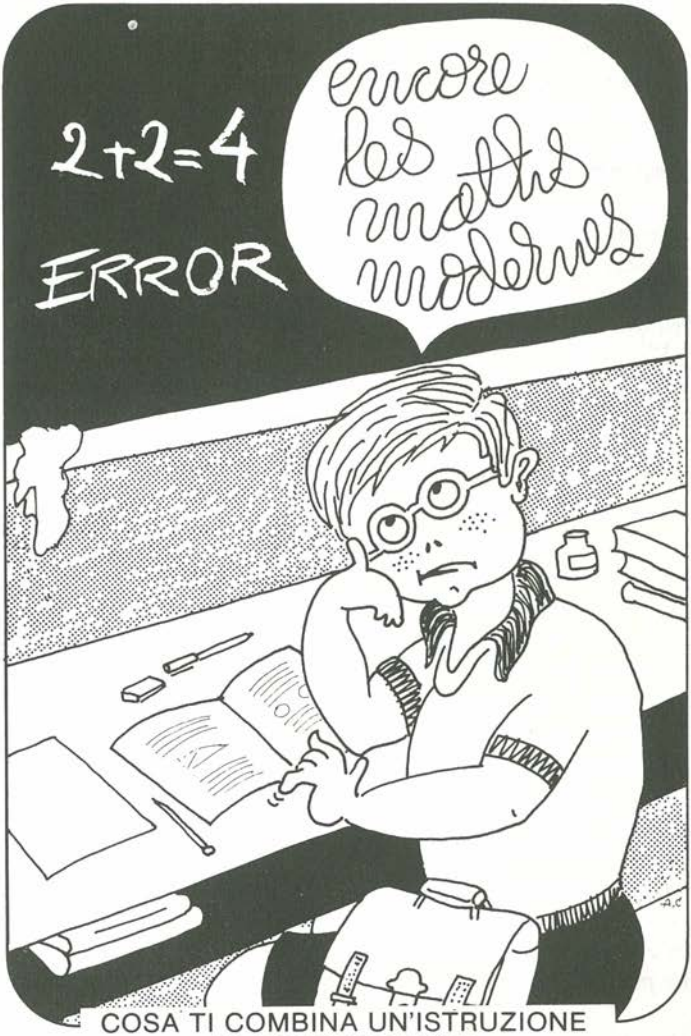
```
10 REM VERSIONE OTTIMIZZATA ( MENO LEGGIBILE )
20 U%=1: FOR I=U% TO K + 25 STEP U:N = N+ U%:PRINT A,B: NEXT
30 REM E' BENE GENERALMENTE DARE UN NOME DI VARIABILE
40 REM A TUTTE LE " COSTANTI " COSI' RAGGRUPPATE IN UNA
50 REM PARTE DEL PROGRAMMA. CIO' FACILITA LA MESSA A PUNTO
60 REM E SOPRATTUTTO LE MODIFICHE O CAMBIAMENTO DI MACCHINA
70 REM ES.: 64 CHE DIVENTA 40 CARATTERI PER LINEA ECC...
```

```
10 REM ES.: DI COSTANTE CARATTERE
20 REM IN UN PROGRAMMA LUNGO E' BENE FARE UNA TAVOLA
30 REM DEI MESSAGGI ( MIGLIORE COERENZA, MODIFICA SEMPLICE)
40 Z1$ = "RICOMINCIA DALL' INIZIO"
50 Z2$ = "INTRODUCETE L' ARTICOLO SEGUENTE"
```

La convenzione DEFINT I - N ereditata dal FORTRAN è molto pratica per i numeri interi, particolarmente gli indici; è un pò uno standard di fatto ma niente impedisce, secondo i bisogni, di allargare o restringere un gruppo di variabili dichiarate intere.

Per un piccolo programma semplice si può sovente trascurare il tipo di variabile intera e lavorare con dei reali (tipo di standard implicito, cioè in assenza d'indicazione contraria); si tratta di formati per indicare al BASIC come immagazzinare e trattare i numeri, e niente impedisce di trattare un numero intero come un numero reale.

Per contro, per un programma voluminoso o la cui velocità di esecuzione è critica, sarà conveniente utilizzare il formato intero che porta a un guadagno di spazio e di velocità.



COSA TI COMBINA UN'ISTRUZIONE
DI ASSEGNAZIONE

Quando un'informazione non è numerica, generalmente alfanumerica, la si designerà con il termine stringa di caratteri o variabile di tipo carattere. Le stringhe di caratteri sono designate con dei nomi, secondo le stesse regole delle variabili numeriche, seguiti però dal carattere $. A$ è distinto da A e da A%.

Come per le variabili numeriche, certi BASIC permettono di eliminare il suffisso dollaro con una definizione esplicita (DEF STR).

Si chiama costante quando l'informazione è presente nel corpo dell'istruzione invece di essere designata con il suo nome. Se si tratta di una costante di tipo carattere, essa deve essere messa tra virgolette, il che permette al BASIC di distinguerla da un nome di variabile.

Calcoli ed espressioni numeriche

Un'espressione numerica è un insieme di calcoli eseguiti o da eseguire a partire da costanti o da variabili.

Il segno uguale, nell'istruzione di assegnazione non ha niente a che vedere con la constatazione di una eventuale eguaglianza (segno = utilizzato nell'istruzione IF).

Esso dà l'ordine all'interprete BASIC di valutare l'espressione a destra del segno = e di copiare il risultato nella variabile il cui nome è posto a sinistra.

LET A = B + C (assegnare ad A il valore B + C)

Il LET è sovente facoltativo ed è molto pratico ometterlo perchè è l'istruzione di più frequente utilizzo, ma in questo caso non si tratta più di un BASIC standard; la comprensione del segno = è generalmente il primo ostacolo che si incontra quando si scopre la programmazione.

Attenzione, esiste il rischio di conflitto tra i vari formati. Il caso più semplice consiste nel voler fare dei calcoli pertendo dalla lettere: si commetterà un errore. Al contrario, l'assegnazione di un risultato numerico a una variabile di tipo carattere potrà essere realizzata per conversione: per esempio, il numero 1979 diventa una catena di 4 caratteri: 1, 9, 7, 9.

Le conversioni più delicate implicano l'unione di interi e di reali: l'interprete converte, ad ogni operazione mista, l'intero in reale (o il reale semplice in reale doppia precisione). Ma non può fare meglio; in un'espressione un pò complicata, un numero può essere troncato prima di essere convertito alla precisione superiore, da cui degli errori minimi (perdita di precisione che si aggrava per ripetizione) e degli errori grossolani (percentuali troncate a 0 prima di essere moltiplicate per 100 e addizionate: si trova 0) che sono più facilmente scopribili.

Nel modo comando l'espressione Print permette di sorvegliare facilmente ciò che succede senza conoscere a memoria le regole di conversione.

I segni utilizzati sono gli stessi usati in matematica con qualche limitazione dovuta alle macchine: + e − sono identici; per la moltiplicazione si usa il segno

* invece del segno X e non può essere omesso come in algebra.

Contrariamente alla confusione che fanno gli uomini, la macchina non rischia di confondere il segno della moltiplicazione con la x minuscola.

Ragioni di carattere tipografico impediscono di segnare gli esponenti con piccoli caratteri scritti più in alto: si utilizza quindi il carattere^ oppure ↑ (freccia rivolta verso l'alto). Per la stessa ragione le linee di frazione sono oblique / e prendono il posto del segno diviso ed i denominatori figurano sulla linea orizzontale dei numeratori.

\+ − * / ↑

Bisogna porre attenzione all'ordine di priorità nella valutazione delle espressioni: nei casi semplici esso corrisponde al "buon senso"; per esempio A + 3 * B vale come A + (3 * B) e non (A + 3) * B.

Quando l'espressione si complica, è prudente utilizzare le parentesi che sono calcolate con priorità e permettono al programmatore di gestire l'ordine dei calcoli.

Usualmente il problema si pone con una frazione (divisione) seguita da una moltiplicazione: non c'è più alto (numeratore) e basso (denominatore) ed è prudente scrivere (A/B) * C piuttosto che conoscere a memoria le regole (esse

```
10 REM ISTRUZIONE DI ASSEGNAZIONE
20 LET A = B + C                    : REM LET E' FACOLTATIVO
30 N = N + 1                        : REM ES.: FREQUENTE ( COME CONTATORE )
------------
10 REM ES.: ISTRUZIONE TROPPO LUNGA ( RISCHIO DI ERRORE )
20 PRINT (-B + SQR(B*B - 4 * A * C)) / ( 2 * A )
30 REM STESSO CALCOLO SCOMPOSTO
40 D = B * B - 4 * A * C
45 REM POSSIBILE AGGIUNTA DEL CONFRONTO  ' D > 0 ', TRALASCIATO QUI
50 D = SQR ( D ): X1 = (-B + D ) / ( 2 * A )
60 PRINT " RADICE = ";X1
70 REM MESSA A PUNTO PIU' FACILE MA PROGRAMMA PIU' LENTO E
80 REM CON MAGGIORE OCCUPAZIONE DI MEMORIA
------------
10 REM ES. DI ERRORE PER CONVERSIONE ABUSIVA
20 REM AUMENTO DEI SALARI DEL 12%
30 INPUT "QUALE E' IL VOSTRO SALARIO ";SA
40 T% = 12/100 : AU = SA * T%
50 PRINT " AVETE AVUTO UN AUMENTO DI ";AU;" LIRE"
RUN
```

possono variare da un BASIC all'altro) e scrivere A/B * C che vale come (A/B) * C e non A/ (B * C).

Quest'ordine di priorità riguarda anche le conversioni.

Per esempio nel calcolatore una percentuale con A e B interi, (A/B) * 100

potrà valere 0 se A< B, ed è consigliabile eseguire prima le moltiplicazioni.

Se si lavora su numeri molto grandi o molto piccoli, bisogna diffidare delle espressioni complicate: troppe moltiplicazioni intermedie possono dare un numero troppo grande (che supera la capacità), troppe divisioni un numero nullo.....

Ad eccezione di ragioni particolari (disponibilità di memoria) non è bene scrivere istruzioni molto lunghe che risultano più difficili da mettere a punto e che, se danno un messaggio d'errore, è più difficile da rilevare.

Il processore, lavora al massimo su due variabili alla volta e deve eseguire stadi intermedi suddividendo l'espressione multipla.

```
10 REM ELENCO DEI TIPI
20 DATA MELE,PERE,SCOUBIDOUS,CAROTE,PATATE
30 REM ELENCO DEI PREZZI UNITARI
40 DATA 10,12,999,8.50,3.40
50 DIM I$(5),P(5)
60 FOR I = 1 TO 5: READ I$(I):NEXT
70 FOR I = 1 TO 5: READ P(I):NEXT
80 REM SI DISPONE ORA DEI TIPI E DEI RELATIVI PREZZI
200 INPUT "NUM. ARTICOLO , QUANTITA' ";NA,Q
210 PRINT Q;"KG DI ";I$(NA);" COSTANO ";Q * P(NA)
```

```
10 REM SOTTOPROGRAMMA PER CONVERTIRE DA 0 A 15
20 REM IN ESADECIMALI (DA 0 A F)
30 DATA 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
40 DIM ES$(16)
50 FOR I = 0 TO 15:READ ES$(I):NEXT
60 REM ORA ES$(K) DA K IN ESADECIMALE
........
200 IF K < 0 OR K > 15 THEN STOP
210 PRINT ES$(K)
220 RETURN
```

```
10 REM ADATTAMENTO AL BASIC APPLE INTEGER ( SENZA APPLESOFT )
20 DIM I$(30)                    : REM TIPI UNITI E DELLA STESSA LUNGHEZZA
30 I$ = "MELE  PERE  SCOUBICAROTEPATATE"
......
210 PRINT I$(6 * NA - 6,6 * NA - 1) : REM ES.: I$(6,11) DA' "PERE  "
```

```
20 DIM ES$(15)
30 ES$="0123456789ABCDEF"
......
210 PRINT ES$(K,K): REM ES$(11,11) DA' "B"
```

Tabelle e indici

Quando un gruppo di dati o di variabili hanno qualche fattore in comune, in particolare quando, un trattamento identico dev'essere effettuato su ciascuno di essi oppure su uno solo non conosciuto a priori, è utile designarli con un nome di famiglia quindi si potranno elaborare "automaticamente" grazie, per esempio, ai loop. La famiglia è una tabella, e i numeri sono gli indici.

Per analogia con le coordinate cartesiane classiche, si dice che la tabella ha una, due o più dimensioni se viene utilizzata con uno, due o più indici.

Il BASIC non è in grado di trattare una tabella se non è informato della sua misura, ciò si ottiene con l'istruzione DIM (dimension). Per esempio DIM T (9) definisce un insieme di dieci variabili T (0), T (1), ..., T (9).

Contrariamente a ciò che succede in certi linguaggi più potenti, T può designare un'altra variabile indipendente non influenzata dall'insieme di variabili create dall'istruzione DIM T (9).

Per un vantaggio proprio degli interpreti, la misura non dev'essere necessariamente stabilita in anticipo, poichè è al momento dell'esecuzione dell'istruzione DIM che si realizza una prenotazione della zona memoria che corrisponde alla tabella così definita. Ciò è utile per programmi voluminosi o per memoria di poca capacità, e permette di adattare la misura al meglio, frazionando eventualmente l'elaborazione.

Si può anche liberare la memoria occupata dalle stringhe di caratteri con l'istruzione CLEAR (o CLR) totale o parziale, per creare nuove matrici in stadi successivi.

L'indice può essere rappresentato da un'espressione numerica, e lui stesso, definito come tabella. Questa tecnica può sembrare inutilmente complicata: essa permette per esempio di semplificare l'aggiornamento di tabelle inerenti intestazioni o di aggiornare il puntatore nella gestione dei dati.

```
10 REM CODIFICA E DECODIFICA
20 REM CHIAVE DI DECIFRAZIONE
30 DATA Y,R,F,B,E,L,W,P,C,G,M,K,S,H,A,O,X,T,V,I,Z,N,D,U,J,Q
40 DIM C$(25),D$(25)
50 REM D$ CONTERRA' LA CHIAVE DI DECIFRAZIONE(INVERSIONE )
60 FOR I = 0 TO 25 :READ C$(I)
70       D$(ASC(C$(I)) - ASC("A"))=CHR$(ASC("A")+I)
80 NEXT
```

```
100 REM CODIFICA:UN TESTO BATTUTO NON CIFRATO SI VISUALIZZA IN CODICE
110 A$ = INKEY$: IF A$ = ""THEN 110      : REM INKEY$ PER IL TRS 80
120 I = ASC(A$) - ASC("A"): PRINT C$(I);
130 GOTO 100
.........
200 REM DECODIFICA:UN TESTO BATTUTO IN CODICE SI VISUALIZZA NON CIFRATO
210 GET A$: IF A$ = "" THEN 210
220 I = ASC(A$) - ASC("A") :PRINT D$(I);
230 GOTO 200
```

```
10 REM VERSIONE PER APPLE INTEGER(SENZA APPLESOFT )
20 DIM C$(25),D$(25)              : REM VETTORI DI 26 CARATTERI AL MASSIMO
30 C$ = "YRFBELWPCGMKSHAOXTVIZNDUJQ"
40 FOR I = 1 TO 25:K = ASC(C$(I))-ASC("A") : REM ASC PRENDE IL PRIMO CARATTERE
50       D$(K,K) = CHR$(ASC("A")+I)
60 NEXT
..........
100 REM CODIFICA
110 K = PEEK(-16384):IF K > 127 THEN 110 : REM PER APPLE
120 POKE -16368,0 : I = K - ASC("A"): PRINT C$(I)
130 GOTO 100
```

```
10 REM IL PASSAGGIO DA VETTORI A LUNGHEZZA MOBILE
20 REM A VETTORI GESTITI CON DIM E' FACILE PER IL CASO DI
30 REM LUNGHEZZE COSTANTI(ES.: TITOLI DI LUNGHEZZA FISSA )
```

In generale, le tabelle possono contenere delle stringhe di caratteri, o degli interi, o dei reali, ma non l'unione di questi tre tipi. Le tabelle di stringhe non sono ammesse in certi BASIC (Apple Integer). L'istruzione DIM ha allora un'altro senso: DIM Z$ (20) crea una stringa di 20 caratteri al massimo; DIM Z$ (11) definisce la stringa che va dall'undicesimo al ventesimo carattere e Z$ (11,15) dall'undicesimo al quindicesimo.

Gli altri BASIC gestiscono automaticamente la lunghezza delle stringhe fino a 255 caratteri e fin tanto che vi è spazio disponibile.

Il metodo del BASIC Apple Integer, ispirato dall'APL dove una stringa di 20 caratteri è in effetti un tabulato di 20 caratteri separati, è poco comodo per delle tabelle di nomi o di intestazioni, ma rende più agile le manipolazioni nel corpo di una stringa (estrazioni di parti di un nome, di un'indirizzo).

Associati a delle istruzioni DATA (dati incorporati nei programmi), le tabelle permettono di tradurre facilmente un codice in un altro o di gestire la corrispondenza tra due tabelle. Certamente essi sono necessari per fare del calcolo matriciale (programmazione lineare, risoluzioni di sistemi di equazioni lineari).

```
10 REM SOTTOPROGRAMMI CHE SIMULANO DELLE FUNZIONI
20 REM ES.: VALORE ASSOLUTO ABS
30 IF X<0 THEN X=-X
40 RETURN
50 REM ES.: PARTE INTERA ALGEBRICA INT(X)
60 REM DA' L'INTERO PRECEDENTE X (ANCHE PER X<0)
70 I%=X                    :REM LA CONVERSIONE TRONCA I DECIMALI
80 IF I%>X THEN I%=I%-1
90 RETURN
--------------
10 REM FUNZIONI DEFINITE DA PROGRAMMA
20 REM 26 NOMI POSSIBILI: FNA...FNZ
30 REM ES.: FUNZIONE IPOTENUSA
40 DEF FNI(A,B) = SQR(A*A + B*B)          :REM (P.E.T.)
..........
100 H5=FNI(3,4)
110 PRINT"IPOTENUSA ";H5
```

CAPITOLO 3

LE FUNZIONI

In un linguaggio come il BASIC, le funzioni rappresentano una vera panoplia di mezzi tra i quali alcuni sono indispensabili. Essi si dividono in due campi principali: le funzioni aritmetiche che permettono di calcolare rapidamente e senza scrivere programmi supplementari delle grandezze matematiche. Talvolta vengono chiamate funzioni "implementate". Il calcolo è sempre effettuato con il metodo dello sviluppo in serie e si potrebbe sempre scrivere un programma che dia lo stesso risultato (con una esecuzione più lenta poichè le funzioni implementate sono scritte in linguaggio macchina).

L'altro grande campo riguarda le manipolazioni di stringhe di caratteri descritte nella seconda parte di questo capitolo; queste funzioni sono molto utili per l'impaginazione e il dialogo.

Oltre a questi due gruppi di funzioni, esistono speciali istruzioni che formano l'argomento di particolari capitoli (Capitolo 6 - funzioni speciali e Capitolo 7 - effetti grafici). Infine, certi BASIC permettono di "fabbricare" delle funzioni.

Questi utilizzatori di funzioni non sono in realtà che dei piccoli sotto-programmi, generalmente limitati ad una linea, ma la cui chiamata è a volte più pratica e corrispondente alla sintassi di una funzione. Si troverà un esempio qui a lato.

Funzioni numeriche

Una funzione è una parola chiave associata ad uno o più argomenti tra parentesi: quando il BASIC incontra questa parola chiave, fa scattare l'esecuzione di un sotto-programma che restituisce un valore in funzione degli argomenti.

Da un punto di vista formale, una funzione è differente da un comando il quale può formare da solo un'istruzione completa: la funzione occupa il posto di una variabile in una espressione, generalmente segue un'istruzione di assegnazione (segno uguale), oppure segue un istruzione PRINT o in un'espressione logica.

A volte non c'è argomento (funzione che rimanda l'ora TI o lo spazio memoria libero MEM), a volte l'argomento è "fasullo", chiama un sotto-programma assembler USR (0).

Alcuni BASIC permettono di includere delle funzioni programmate per l'utente. È una variante (formale, ma a volte più pratica) di sotto-programma.

Accanto alle funzioni che lavorano sulle stringhe di caratteri (vedere oltre) vi è tutto un gruppo di funzioni numeriche che permettono di riprendere la maggior parte delle usuali funzioni matematiche.

```
10 REM FUNZIONE ABS
20 REM ES.: TRACCIATO DI UNA CURVA ALGEBRICA
30 REM ESECUZIONE LENTA MA PROGRAMMA SEMPLICE
40 REM VALEVOLE PER OGNI EQUAZIONE F(X,Y)=0
50 REM L'INIZIO S DEFINISCE UNA APPROSIMAZIONE +\- STRETTA E VISUALIZZATA
60 S = 30
70 FOR X=0 TO 127:FOR Y=0 TO 47          :REM SCANSIONE DELLO SCHERMO (TRS80)
80 F = X*X+2*Y*Y-45*X-60*Y-X*Y+650       :REM EQUAZIONE DI UNA CONICA
90 IF ABS(F)<S THEN SET(X,Y)             :REM (TRS) VISUALIZZA X,Y
100 NEXT
--------------------
10 REM FUNZIONE SGN
20 REM ES.:VARIABILE INDICE E TRIPLICE SCAMBIO
30 DIM Z$(10)
40 Z$(0)="DEBITRICE":Z$(1))="NULLA":Z$(2)="CREDITRICE"
..........
100 ON SGN(T)+1 GOTO 150,200,250
150 REM TOTALE NEGATIVO, CALCOLO INTERESSI DEBITORI
..........
300 REM EDITING DEI RISULTATI
310 PRINT"POSIZIONE ";Z$(1+SGN(T)),ABS(T)
--------------------
10 REM FUNZIONE INT
20 PRINT INT(3.1416),INT(-2.17)
RUN
3         -3
30 REM ARROTONDAMENTO ALL'INTERO PIU' VICINO
40 AR = INT(A+0.5)                  :REM ES.:INT(2.7+0.5) DA' 3
--------------------
10 REM FUNZIONE CINT (TRS80)
20 REM = INT NEL FORMATO INTERO
--------------------
10 REM ES.: ARROTONDAMENTO IN FRANCHI E CENTESIMI S=17.6666 F.
20 F = INT(S):C=INT(100(S-F))
30 PRINT F;",";C;"FF"
RUN
17,66FF
```

Valore assoluto e segno

La funzione valore assoluto ABS è d'uso frequente in matematica (in gestione contabile ci si interessa sempre al segno dei numeri.....).

La funzione SGN ritorna -1,0 oppure 1, secondo se l'argomento è negativo, nullo o positivo. Essa permette (aggiungendo 1 oppure 2) d'avere un indice 0,1, 2 oppure 1, 2, 3 che può, in certi casi, facilitare la scrittura: salto diretto a tre punti del programma grazie ad un GOTO calcolato (scambio triplo), scelta fra tre intestazioni della tabella.

Valore intero, valore doppia precisione, conversione di precizione

La funzione INT (per Integer, intero) ritorna la parte intera di una variabile.

Per ottenere un arrotondamento più vicino all'intero (certi linguaggi hanno la funzione corrispondente) basta aggiungere 0.5 e prendere il valore intero.

```
10 REM FUNZIONE FIX(TRS80)
20 REM TRONCA LA PARTE DECIMALE (INT(X)+1 SE X<0 )
30 I% = -2.3:PRINT INT(-2.3),FIX(-2.3),I%
RUN
-3        -2        -2
```

```
10 REM FUNZIONE CDBL(TRS80)
20 REM ES.: CALCOLO DEL NUMERO DI NEPERO
30 E# = 1:X# = 1
40 FOR I = 1 TO N : X# = X# / CDBL(N) : E# = E# + X# : NEXT
50 PRINT"E = ";CSNG(E#)
60 REM CSNG ELIMENA GLI ULTIMI DECIMALI NON SIGNIFICATIVI
```

```
10 REM TEOREMA DI PITAGORA
20 IF SIN(X)↑2 + COS(X)↑2 <>1 THEN PRINT
      "IL QUADRATO DELL' IPOTENUSA MI INDUCE IN ERRORE"
```

```
10 REM TROVARE PI CON ARCO-TANGENTE
20 PRINT 4*ATN(1)
```

```
10 REM TRACCIARE UN CERCHIO - COORDINATE POLARI
20 R = 20: X0 = 50:Y0 = 25        :REM RAGGIO E COORDINATE DEL CENTRO
30 FOR U=0 TO 6.28 STEP 0.1       :REM 6.28 = 2PI RADIANTI = 1 GIRO
40 X = X0 + 2.3*R*COS(U)          :REM IL COEFF. 2.3 DISOVALIZZA
50 Y = Y0 + R*SIN(U)     :REM IL CERCHIO (SU TRS)
60 SET(X,Y)                       :REM PER TRS; PLOT PER L' APPLE
70 NEXT
```

```
10 REM FUNZIONI EXP E LOG
14 PRINT EXP(1),LOG(0.0001)       :REM E↑1=2.718; LOG(0)= -INFINITO
18 PRINT 2.7↑1.4,EXP(1.4*LOG(2.7))
20 REM QUALE INCREMENTO RISULTA DAL 10% IN 10 ANNI
25 PRINT EXP(10*LOG(1.1))
```

```
10 REM FUNZIONE SQR O SQRT (RADICE QUADRATA)
20 PRINT SQR(2),2↑0.5
```

INT restituisce un numero intero in formato reale (dunque non condizionato dal limite ± 32567).

La funzione CINT ritorna la parte intera, ma in formato intero, limitato a ± 32567. I calcoli sono più rapidi quando si lavora con dei numeri di formato intero.

Un'altra funzione, poco utile, tronca la parte decimale: funzione FIX (TRS 80).

La funzione CDBL ritorna lo stesso valore dell'argomento, ma in formato doppia precisione. Per una costante, per esempio, è più semplice scrivere 1# al posto di CDBL (1).

Questo permette in una divisione tra variabili dove non si è voluta definire la doppia precisione, di operare una conversione che dia un risultato in doppia precisione. In un totale si limita così il cumolo degli errori di arrotondamento, senza dichiarare tutte le variabili in doppia precisione (economia di spazi).

Nello stesso spirito, la funzione CSNG ritorna un valore nel formato reale semplice; al momento dell'edizione, essa permette di eliminare i decimali superflui; nel caso di un gran numero di calcoli intermedi, si può così presentare un numero con tutti i suoi decimali significativi, dato che l'aumento di precisione è necessario per evitare l'accumolo degli errori d'arrotondamento.

Queste funzioni di conversione di formato sono un pò superflue poichè è sufficiente effettuare ed attribuire l'argomento a una variabile intermedia dichiarata al formato voluto, ciò realizza automaticamente la conversione.

Esse portano un guadagno di velocità d'esecuzione, che sarà sempre apprezzabile.

Funzioni trigonometriche

Sono i classici SIN, COS, TAN, per SINUS, COSINUS e tangente. Gli argomenti devono essere in radianti (un giro completo = 2 π radianti oppure 6.28 per 360° cioè 1 radiante = 360/6.28° = 57.29578° e 1 grado = 6.28/360 radianti = 0.0174533).

Sono diponibili anche le funzioni inverse di tangente: ATN (arc tg) cioè che se Y vale ATN (X), X vale TAN (Y) oppure ancora SIN (Y)/COS (Y). Come tg 45° = 1 e 45° = π/4 si ottiene facilmente π = 4 ATN (1). Si verificherà facilmente che TAN (π/2) è infinito.

```
10 REM IMMAGINE BINARIA DI UN NUMERO DECIMALE
20 REM CON DIVISIONI SUCCESSIVE PER 2
30 INPUT A
40 Q% = A/2
50 R% = A - 2 * Q%
60 REM 2 MOVIMENTI DEL CURSORE A SINISTRA PER SCRIVERE DA DESTRA
70 REM A SINISTRA AL FINE DI PRESENTARE LE UNITA' A DESTRA
80 PRINT CHR$(24);CHR$(24)          :REM TRS
80 PRINT "<- <-"                    :REM PET
90 PRINT R%;                        :REM VISUALIZZA UN BIT (0/1)
100 IF Q%>0 THEN A = Q%:GOTO 40
```

```
10 REM SCOMPOSIZIONE DI UN INDIRIZZO SU 2 BYTES
20 REM CON DIVISIONI SUCCESSIVE PER 256
30 A1% = A/256                      :REM A1=BYTE PIU' SIGNIFICATIVO
40 A2% = A-256*A1%                  :REM A2= " MENO "
```

```
10 REM CONVERSIONE DI T SECONDI IN H/M/S
20 M% = T/60
30 S% = T-60*M%
40 H% = M%/60
50 M% = M%-60*H%
60 PRINT H%;"/";M%;"/";S%
70 REM IDEM SE H = GRADI SESSAGESIMALI
```

Esponenziali - logaritmi - radice quadrata

Queste funzioni giocano un grande ruolo in matematica e in fisica. Se Y vale EXP (X), X vale LOG (Y). L'argomento di un logaritmo deve essere positivo. EXP (1) da il numero di Neper. L'operatore di elevazione ↑ (oppure ^) è molto potente: X↑Y è calcolato nel modo seguente: EXP (Y*LOG (X)).
Per calcolare un quadrato od un cubo sarà preferibile, se ha importanza la velocità, scrivere X*X o X*X*X invece di X↑2 e X↑3.

Un logaritmo decimale (non ce ne dovrebbe essere bisogno con le possibilità di cui sopra!) si ottiene con LD = (LOG (X)/ LOG (10)).

La radice quadrata si ottiene con la funzione SQR (scritta a volte SQRT), abbreviazione di "square root". Constaterete che i professori di matematica non hanno torto di esigere che l'argometno sia positivo.

Funzione modulo e divisione in numeri interi

La nozione di divisione aritmetica è sovente utilizzata in programmazione: a partire da due interi *a* e *b* trovare un quoziente e un resto intero tale che a = b * q + r con r <b.

Si dice che *a* è congruo a *r* modulo *b*. Certi BASIC (Integer Apple) hanno una funzione che dà *r* in funzione di *a* e *b*: R = A MOD B.

Si ottiene lo stesso risultato con una divisione di numeri messi prima di tutto in formato intero:

$$R = A\% - (A\%/B\%) * B\%$$

o ancora $Q\% = A/B : R = A - B * Q\%$

Le divisioni successive di A per B (cioè A diviso per B dà un primo resto e un quoziente; questo quoziente diviso per B dà un secondo resto e un quoziente, ecc....fino all'ultimo quoziente che è nullo) forniscono una scomposizione del numero A scritto in base B (con il primo resto = cifra delle unità, secondo resto cifra dei B, poi il quadrato dei B, ecc....).

Se B vale 2, si ottiene l'immagine binaria (i resti successivi che valgono 0 oppure 1) di A. Se B vale 16, si ottiene l'immagine esadecimale del numero A, i resti valgono da 0 a 15. Occorre ancora un piccolo sotto-programma che trasformi, 10, 11, 12, 13, 14, 15, in A, B, C, D, E, F per la tradizionale notazione esadecimale.

Infine, se B vale 256, si ottiene una scomposizione per byte molto utile con PEEK e POKE quando bisogna passare da un indirizzo (generalmente 0 a 65535) alla sua immagine decimale su più byte (generalmente 2).

Se B vale 100, si può anche formare una somma con i centesimi (in assenza del formato USING). Se B vale 60, si realizza la conversione sessagesimale del tipo ore/minuti/secondi (o gradi degli angoli).

```
10 REM CONCATENAZIONE
20 IF CODICE=1 THEN Z$="SIG." ELSE Z$="SIG.RA"
30 Z$ = Z$+" DUPONT"
40 PRINT Z$
```

```
10 REM TEST E CONFRONTO
20 INPUT "VOLETE FARE UN GIOCO ";Z$
30 IF Z$<>"SI" THEN STOP
40 DATA 4,RENATA,CARLA,MICHELE,GIANNI
50 READ N:DIM Z$(N):FOR I=1 TO N:READ Z$(I):NEXT
60 REM GIOCO CON PERMUTAZIONI - POCO EFFICACE MA SEMPLICE
70 FOR I=1 TO N-1
80 IF Z$(I+1) < Z$(I) THEN 150
90 NEXT
100 REM FINE DEL GIOCO
........
150 REM SI PERMUTANO DUE VICINI E SI RICOMINCIA
160 REM PER PERMUTARE 2 VALORI OCCORRE UN VALORE INTERMEDIARIO
170 A$ = Z$(I):Z$(I) = Z$(I+1): Z$(I+1) = A$:GOTO 70
```

```
10 REM FUNZIONE LEN PER SOTTOLINEARE UN NOME
20 PRINT N$:FOR I=1 TO LEN(N$):PRINT "=";:NEXT
```

```
10 REM EQUIVALENTI DI LEFT$,RIGHT$ IN INTEGER BASIC APPLE
20 Z$(5) = Z$              :REM CANCELLA I 5 CARATTERI DI SINISTRA
30 Z$ = Z$(3,11)           :REM ESTRAE I CARATTERI DA 3 A 11
40 Z$(5) = ""              :REM ESTRAE E CANCELLA I CARATTERI DA 1 A 5
```

```
10 REM FUNZIONE MID$ ES.:SCRIVERE UN NOME A ROVESCIO
20 N$ = "LAVAL"            :REM PALINDROME COME"ESOPE RESTE ICI ET SE REPOSE"
30 FOR I=LEN(N$) TO 1 STEP -1
40 PRINT MID$(N4,I,1);:NEXT
50 REM CERCARE DEI PALINDROMI ARITMETICI ES.:"21*21=441"
                                            "31*31=961"
```

```
10 REM SINTASSI : LEFT$ (VAR.STRINGA,LUNGHEZZA)
                  RIGHT$( "",")
                  MID$  ( "",POSIZIONE,LUNGHEZZA)
```

Manipolazione delle stringhe di caratteri

Le funzioni numeriche viste qui sopra permettono di effettuare dei calcoli abbastanza elaborati, ma non permettono di effettuare l'impaginazione e la presentazione o più generalmente tutti i dati non numerici.

Tutto un gruppo di funzioni permette di trattare questi dati chiamati "stringhe di caratteri". Si dovrà diffidare della possibile ambiguità nei riguardi di una cifra che può essere considerata sia per il suo valore numerico che come carattere. La parola "carattere" deve essere considerata nel senso di un codice strettamente associato ad un carattere di stampa.

Si dovranno manipolare dei caratteri quando si vorrà impaginare, "allestire" un tabulato, fare un gioco, utilizzare i caratteri speciali che permettono di controllare la visualizzazione

Prima di esaminare le funzioni di manipolazione delle stringhe, citiamo le operazioni che si possono effettuare sulle stesse. Non si possono fare dei

```
10 REM FUNZIONE ASC (CODICE ASCII)
20 REM ES.: ESAMINARE I CODICI EMESSI DA OGNI TASTO
30 Z$=INKEY$:IF Z$="" THEN 30            :REM TRS80 (GET PER PET E APPLE)
40 PRINT Z$;" ";ASC(Z$)
RUN
* 42
--------------------
10 REM FUNZIONE CHR$
20 REM ES.: VISUALIZZARE L'EFFETTO SU PRINT DEI CARATTERI SPECIALI
30 INPUT K:PRINT CHR$(K);:GOTO 30
40 REM VISUALIZZARE L' ALFABETO RAPPRESENTABILE
50 INPUT K1,K2                           :REM PROVARE PER EVITARE
60                                       :REM ALCUNI CARATTERI SPECIALI
70 FOR K=K1 TO K2:PRINT CHR$(K);K:NEXT
80 GOTO 50
--------------------
10 REM FUNZIONE STRING$ (N,CARATTERE)TRS80
20 PRINT STRING$(35,"*")                 :REM IMPAGINAZIONE DI TABULATO
30 FOR I=1 TO N:PRINT STRING$(X(I),"="):NEXT
40                                       :REM FARA' UN MISTOGRAMMA
50 B$ = STRING$(3,196)                   :REM RIUNISCE 3 CARATTERI SPECIALI
60 REM ES.: MESSA IN OPERA DI UNA TABULAZIONE VERTICALE (TRS80)
70 DIM VT$(10)
80 FOR J=1 TO 10:VT$ = STRING$(I,26):NEXT        :REM 26=SALTO LINEA
90 REM PRINT VT$(N) SALTERA' N LINEE (N DA 1 A 10)
--------------------
10 REM FUNZIONE STR$(ESPRESSIONE)
20 REM ES.: PREEDIZIONE CHE SIMULA "PRINT USING" SU 8 COLONNE
30 R$ = STR$(R):IF LEN(R$) > 8 THEN R$=" ****** "
40 R$ = STRING$(8-LEN(R$),"=")+R$
50 PRINT R$                 :REM "==125.40" SONO 8 CARATTERI
```

calcoli, ben inteso; l'operatore + realizza la concatenazione, vale a dire la "saldatura" di più stringhe fra di loro. Gli operatori di comparazione $>$ $<$ e $=$ funzionano anche per le stringhe, con le regole corrispondenti all'ordine lessicografico usuale, con JEAN inferiore a JEANNOT (l'ordine esatto corrisponde al codice ASCII in appendice, vale a dire 0 < 1.....<9 <bianco-.....<A.....<Z ecc.....).

Il segno = di assegnazione funziona come per le variabili numeriche. In alcuni BASIC la lunghezza non è gestita automaticamente e la stringa di sinistra, che riceve, dovrà essere abbastanza grande per aggiungere ciò che le viene inviato (Apple).

La funzione LEN (da LENGTH, lunghezza) ritorna un numero (da 0 a 255) uguale alla lunghezza della stringa data in argomento. Per una stringa nulla, che figura nella tavola dei nomi delle variabili, ma che non contiene niente, la lunghezza è 0. Tale funzione è utile per l'edizione: per esempio per sottolineare

una parola la cui lunghezza non era ancora conosciuta o per elaborare le singole lettere di una parola non conosciuta a priori.

Le funzioni LEFT$, RIGHT$, MID$ ritornano una sottostringa estratta a sinistra, a destra o nel corpo della stringa in argomento. Nel BASIC, con una lunghezza di stringa non automatica, queste funzioni sono sostituite con degli indici: Z$ (i, j) determina la sottostringa della stringa Z$ dal carattere i al j.

Le funzioni ASC e CHR$ permettono di passare da un carattere al codice ASCII corrispondente e viceversa.

ASC permetterà di manipolare dei caratteri secondo il loro ordine alfabetico, dunque con dei loop e delle istruzioni numeriche, poichè i codici dalla A alla Z sono consecutivi. Se le manipolazioni includono delle cifre, si dovrà diffidare del tratto dei 7 caratteri (: àα) che separano 9 da A. Se l'argomento di ASC contiene più caratteri, solo il primo viene preso in considerazione.

```
10 REM FUNZIONE VAL(VAR.STRINGA)
20 REM ES.: UNIONE DI VARIABILI DIVERSE IN UNA FRASE
30 REM DI DIALOGO O IN UNO SCHEDARIO
40 INPUT Z$
50 Z = VAL(Z$):IF Z=0 THEN 100  :REM VAL IGNORA UNA STRINGA NON NUMERICA
60 REM LA RISPOSTA INIZIAVA CON UN NUMERO, ORA DISPONIBILE IN Z
..........TRATTAMENTO DI Z
70 REM TRATTAMENTO DELLA SEQUENZA ( PRELEVA IL NUMERO DI Z$)
80 Z$ = RIGHT$(Z$,LEN(Z$)-LEN(STR$(Z)))
90 REM TRATTAMENTO DI UNA PARTE ALFABETICA
```

CHR$ permette di creare tutti i caratteri (codici da 0 a 255) e particolarmente quelli che non possono essere scritti in una costante battendo un tasto della tastiera: caratteri grafici, caratteri speciali, movimenti di cursore differenti dai movimenti usuali comandati dall'istruzione PRINT. Vedere le appendici Codice ASCII e caratteri speciali.

La funzione STRING$ (da non confonde con la STR$) è una sorta di CHR$ con la particolarità che permette di creare una stringa i cui caratteri sono tutti identici. Il carattere da duplicare è indicato in chiaro con il suo codice ASCII, ciò facilita l'edizione delle tabelle (tratti orizzontali con caratteri grafici).

Le funzioni STR$ e VAL permettono di passare da un numero (o da una espressione numerica) alla stringa di caratteri formata dalla cifre di questo numero (o del numero risultante dall'espressione) e viceversa. La reciprocità non può essere applicata all'espressione (STR$ (2 + 2) che, da il carattere "4" ma VAL ("4") non può che dare il numero 4).

STR$ permette di preparare un'edizione, permettendo al programma di conoscere la lunghezza del numero da visualizzare per sottolinearlo o inquadrarlo in un tabulato. Si possono anche concatenare dei caratteri con dei numeri (esempio CR per Credito, FF per una somma).

Si potrebbe raggiungere lo stesso risultato con PRINT ma la logica del programma è qualche volta semplificata se si dissocia la visualizzazione della messa in bella forma dei dati; una parte comune del programma può provvedere a tutte le edizioni con delle linee o elementi di strutture, preparati in vari modi, nei diversi punti del programma.

VAL permette di unire delle variabili di caratteri e delle variabili numeriche e di leggere il tutto in un formato carattere, altrimenti non si potrebbero leggere delle variabili di tipo misto, salvo che non siano in un ordine conosciuto a priori (per esempio un nome o una quantità).

Ciò si applica anche ad un'inpostazione di dati (INPUT) dove la stessa istruzione potrà indifferentemente leggere dei numeri o dei caratteri. Se dei numeri e delle lettere sono muniti nella stringa-argomento, VAL convertirà il primo numero incontrato a partire da sinistra.

```
10 REM SALTO DIRETTO GO TO (ANDARE IN..)
20 GOTO 100              :REM GLI SPAZI SONO FACOLTATIVI
100 REM LA DESTINAZIONE DEL GOTO PUO' ESSERE UNA REM
110 REM CIO' PERMETTE DI MODIFICARE LE ISTRUZIONI SEGUENTI
120 REM SENZA RISALIRE AL GOTO, MA RALLENTA L' ESECUZIONE.
130 ON K GOTO 200,300,400        :REM IDEM GOSUB PAG. 48
                                 IGNORATO SE K<1 O K>3
```

```
10 REM ISTRUZIONE IF + CONTATORE
20 N = N+1
.....
100 IF N<50 THEN GOTO 20         :REM LA SEQUENZA SARA' ESEGUITA 50 VOLTE
```

```
10 REM ESPRESSIONE LOGICA
20 IF ( A>B AND N<20 ) OR J=K+L THEN 100
30 REM IF(...) THEN 100 EIF (...) GOTO 100
40 REM SI EQUIVALGONO,MA E' SEMPRE MEGLIO USARE THEN
50 REM CHE E' NECESSARIO SE L' ISTRUZIONE SEGUENTE
60 REM NON E' UN GOTO OPPURE SE VE NE SONO DIVERSE
```

```
10 REM OPERATORI BOOLEANI
20 REM AND - OR - NOTSI APPLICANO A DEGLI INTERI ES.:
30 PRINT 7 AND 13                :REM IN BINARIO 0111 E 1101
RUN
5
40 REM 5 = 0101 IN BINARIO
```

```
10 REM 0 AND N DARA' SEMPRE0 (ZERO BINARIO = 00000000)
20 REM -1 OR N DARA' SEMPRE -1 (-1 BINARIO = 11111111)
30 REM QUESTO PERCHE' VERO=-1 E FALSO=0
40 REM ES.: IF ESP.A OR ESP.B(ESP.A=-1 QUINDI CONDIZIONE VERIFICATA)
       IF ESP.A AND ESP.B (ESP.A= 0 QUINDI CONDIZIONE NON VERIFICATA)
```

CAPITOLO 4

LOGICA DI SVOLGIMENTO DI UN PROGRAMMA

La possibilità di saltare da una parte di un programma ad un'altra è una caratteristica essenzialmente di un computer: si può considerare che tale possibilità lo differenzia dai vecchi meccanografici che richiedevano l'"antico" uso di scrivere i programma su schede perforate.

Esiste il salto semplice "incondizionato", per esempio alla fine programma, per ricominciare automaticamente, o per aggirare un gruppo di istruzioni non comprese nella sequenza logica. È l'istruzione GOTO e la sua variante multipla ON....GOTO....

Esistono i salti "condizionati" la cui esecuzione dipende dal precedente svolgimento IF (se-espressione logica vera) THEN (allora-eseguire un'azione).

L'espressione logica è alla base della comparazione di espressioni aritmetiche.

Le espressioni possono essere complicate ed inoltre si possono combinare delle espressioni logiche con AND e OR.

L'istruzione da eseguire (se la combinazione di espressione si rivela vera) è il più sovente un GOTO che rinvia verso le istruzioni che il programmatore ha previsto di eseguire per quel caso. Di conseguenza dietro all'IF vengono le istruzioni da eseguire nel caso contrario.

L'azione da eseguire può essere qualsiasi istruzione. Certi BASIC autorizzano un gruppo di istruzioni dietro il THEN. È molto pratico ma conviene stare in guardia: tutto il seguito della linea fa allora parte del gruppo THEN di azioni da eseguirsi se la condizione è vera.

Alcuni BASIC permettono di aggiungere un gruppo ELSE da eseguire solamente se l'espressione logica è falsa. Si può passare facilmente a terminare il gruppo THEN con un GOTO.

Il loop infinito

Un classico errore, che era molto noioso sui grandi computer, che si pagava con lo spreco del tempo macchina si chiama il loop infinito: dopo aver fatto RUN, il programma si avvia e non succede più niente!

In questo caso un errore di logica ha probabilmente fatto in modo che una

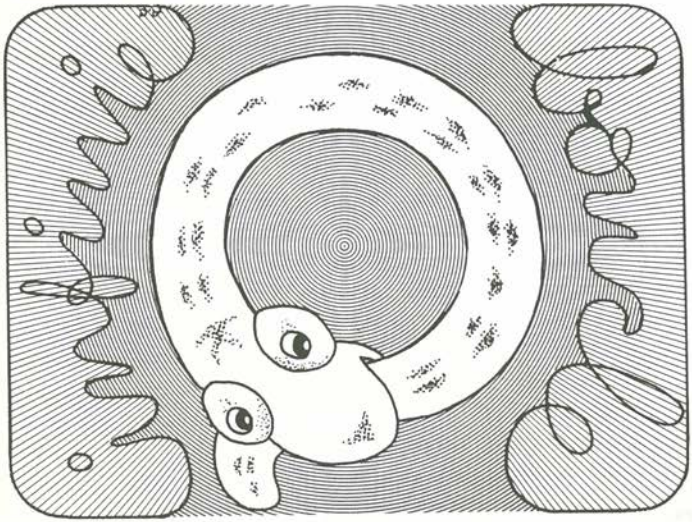
combinazione sfortunata di un GOTO generi un loop che si inanelli su se stesso. Con un piccolo sistema è sufficiente fare BREAK o STOP (control-C sull'Apple).

In certi casi si ha necessità di un loop infinito per inibire il defilamento automatico che verrebbe a danneggiare un bel grafico o un tabulato, cancellando una o due linee dal lato dove si trova il cursore per inserire READY e il "prompt". È sufficiente terminare il programma con un GOTO su se stesso al posto di END o STOP.

```
10 REM ISTRUZIONE ELSE (ALTRIMENTI..) DOPO IF...THEN (SE...ALLORA)
20 IF N=1 THEN A=A+1 ELSE B=B+1
100 REM L' AGGIUNTA DI UN GOTO ALLA FINE DEL GRUPPO THEN (QUI DOPO A=A+1)
110 REM SOSTITUISCE L' ISTRUZIONE ELSE
120 IF N=1 THEN A=A+1:GOTO 140
130 B = B+1
140 ......
```

```
10 REM LOOP INFINITO ( ARRESTO CON BRAK/STOP/CONTROL-C)
20 GOTO 20
30 REM ARRESTO CON BATTUTA DI UN TASTO
40 Z$ = INKEY$: IF Z$="" THEN 40          :REM TRS80 (GET PER PET)
50 STOP                    :REM OPPURE SEGUITO
```

```
10 REM SCAMBIO (STATO NORMALE A=0 )
20 IF A=1 THEN A=0:.......        :REM SEQUENZA FUORI NORMA
.......
50 IF (CONDIZIONE) THEN A=1       :REM AL PROSSIMO LOOP SARA' ESEGUITA
                                    LA SEQUENZA FUORI NORMA
..........
100 GOTO 20
```



Lo scambio

È un modo di programmare che permette di apportare una variante in alcuni casi all'esecuzione di un loop che ritorna un numero di volte non conosciuto in precedenza, vale a dire GOTO in loop dal quale si esce per mezzo di un test IF.

Ciò evita di risolvere un loop quasi identico (in generale si tratta di un inizio o termine di una sequenza).

```
10 REM LOOP FOR
20 REM ES.: CALCOLO DELLA SOMMA DEI PRIMI N INTERI
30 INPUT N:S=0
40 FOR I=1 TO N
50 S = S+I
60 NEXT
70 PRINT S,N*(N+1)/2
80 GOTO 30
```

```
10 REM NIDIFICAZIONE DI LOOP
20 FOR I=1 TO 100
.........
50FOR J=A TO B STEP C    :REM LA SCRITTURA SFALSATA
.........                :REM FACILITA LA COMPRENSIONE
90NEXT J
.........
120 NEXT I
130 REM SE NON VI E' NULLA TRA 90 E 120 UN SOLO NEXT E SUFFICIENTE
140 REM NEXT J,I OPPURE NEXT:NEXT
```

```
10 REM CALCOLO DI UNA TABELLA DI ATTUALIZZAZIONE
20 INPUT"DURATA (ANNI) ";N
30 INPUT"TASSO MINIMO E MASSIMO (ES.: 5,12) ";T1,T2
40 FOR I=1 TO N:PRINT
50 FOR T=1+T1/100 TO 1+T2/100 STEP 0.01
60 PRINT T^I;" %";
70 NEXT
80 REM STEP 0.01 DEFINISCE UN INCREMENTO DELL' 1%
90 REM CIO' FORNISCE UNA COLONNA CON PERCENTUALE TRA T1 E T2
```

Lo scambio consiste in una variabile ausiliare che prenderà i valori 0 oppure 1, per esempio: lo scambio sarà "tallonabile" (come nei treni elettrici) se la sequenza corrispondente al caso eccezionale rimette lo scambio al valore ordinario.

I loop programmati

L'istruzione FOR/NEXT permette, in maniera facile, di creare un contatore e di eseguire un gruppo di istruzioni un numero di volte voluto. Si può ben fare la stessa cosa se con un GOTO e con dei test IF su un contatore, ma ciò implicherebbe un numero maggiore di istruzioni e obbligherebbe a riflettere

sui valori limite che dipendono dalle rispettive posizioni del test e dell'incremento del contatore.

```
FOR contatore = inizio TO fine STEP incremento
,
,
,
,
NEXT
```

È necessario fornire un nome di variabile al contatore (si prende sovente I, J ... ma non è necessariamente un numero intero).

```
10 REM USCITA DAL LOOP E RINVIO ALLE ISTRUZIONI SEGUENTI
20 REM ES.: RICERCA DDI UNA PAROLA IN UNA TABELLA DATI
30 FOR I=0 TO 100
40 READ T$
50 IF T$=M$ THEN 100              :REM ESCE SE TROVATO
60 IF T$<M$ THEN 80               :REM PASSA AL SEGUENTE
70 ........
80 NEXT
90 PRINT"ASSENTE NELLA TABELLA":STOP
100 ....... (SEGUITO)
```

```
10 REM TEMPORIZZAZIONE
20 REM ES.: LAMPEGGIO DI UN CARATTERE
30 PRINT @ N,"*":GOSUB 200
40 PRINT @ N," ":GOSUB 200:GOTO 30
200 FOR I=1 TO 500:NEXT:RETURN          :REM RITARDO
300 REM SI PUO' SOSTITUIRE 'PRINT @' CON UN MOVIMENTO
310 REM A SINISTRA DEL CURSORE 'PRINT"<-*" E "<- " (PET)
```

```
10 REM TEMPORIZZAZIONE
20 REM ES.:PRESA DI UN CARATTERE IN TEMPO LIMITE
30 REM (GIOCHI;SICUREZZA ALLA CONVALIDA.. PER ESEMPIO)
40 Z$ = INKEY$:IF Z$<>"" THEN 70
50 T = T+1:IF T>500 THEN 150
60 GOTO 40
70 REM ANALISI DEL CARATTERE,DECREMENTO EVENTUALE DEL TEMPO T
........
80 T = 0                  :REM AZZERAMENTO DEL CONTATORE TEMPO
........
150 REM FINE DEL RITARDO
160 PRINT"RICOMINCIARE DAL PRINCIPIO"   :REM OPPURE ALTRA AZIONE
........
```

```
10 REM TEST DEI RIFLESSI: PREMERE UN TASTO AL COMPARIRE DI '**'
20 CLS:FOR I=0 TO RND(1000)+1000:IF INKEY$<>"" THEN PRINT"IMBROGLIONE"
30 NEXT:PRINT"**"                       :REM RITARDO IMPREVEDIBILEDA 1000 A 2000
40 N = N+1:Z$=INKEY$:IF Z$="" THEN 40
50 PRINT N:IF N<18 THEN PRINT"SEI RAPIDO"
60 GOTO 20
```

Si possono annidare dei loop: vale a dire creare un sottogruppo con un secondo contatore in seno al gruppo che lo comprende: ad ogni esecuzione del gruppo principale il sottogruppo sarà ripetuto sotto il controllo del secondo. È necessario precisare quindi i limiti, menzionando a lato del NEXT il nome del contatore del loop interessato.

Se il limite coincide, si scriverà NEXT I2, I1 rispettando l'ordine in modo che il sottogruppo resti bene inglobato nel gruppo che lo contiene.

Il contatore messo a punto con l'istruzione FOR è accessibile al programma: con questo si possono fare dei calcoli e l'esecuzione dei loop non sono necessariamente identiche. D'altra parte, però, si sciupa il funzionamento se si modifica il contenuto del contatore: è un errore frequente in un vasto programma prendere inavvertitamente per variabile di lavoro intermedio un nome (I, J, K) già utilizzato come contatore nel corso del programma.

Si può uscire da un loop "non importa in quale punto" (per mezzo di un test IF per esempio) prima che il contatore abbia preso tutti i valori previsti: al

```
10 REM ISTRUZIONE GOSUB
20 REM ES.: DUMP DI MEMORIA.IL SOTTOPROGRAMMA VISUALIZZA UN CARATTERE ESADEC.
30 DATA 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
40 DIM H$(15):FOR I=0 TO 15:READ H$(I):NEXT
50 INPUT"INDIRIZZO DI PARTENZA ";A
60 REM VISUALIZZAZIONE DI 16 RIGHE DI 16 BYTES
70 FOR L=0 TO 15
80 PRINT:PRINT A+16*L;" ";:REM IL PRIMO PRINT VA A CAPO
90 FOR J=0 TO 15
100 J = PEEK(A+16*L+I)          :REM LEGGE UN BYTE
110 K%=J/16:GOSUB 1000          :REM VISUALIZZA 1/2 BYTE
120 K%=J-16*K%:GOSUB 1000       :REM " " "
130 PRINT" ";                   :REM SPAZIO TRA BYTE E BYTE
140 NEXT
150 Z$=INKEY$:IF Z$="" THEN 150
160 IF Z$=" " THEN A=A+256:GOTO 60      :REM PAGINA SEGUENTE
170 GOTO 50
1000 PRINT H$(K%);:RETURN
```

```
10 REM GOSUB CALCOLATO
20 ON K GOSUB 1000,1100,1200
30 REM SALTO A 1000,1100,1200 SE K=1,2 O 3
40 REM COME PER GOTO,L' ISTRUZIONE E' IGNORATA SE K<1 O K>3
........
1000......
1040 RETURN
1100......
1160 RETURN
1200......
1230 RETURN
```

contrario, è necessario entrare "dalla porta", vale a dire tramite l'istruzione FOR e non nel mezzo di un loop, ciò che provocherebbe normalmente un errore al primo NEXT che va ad incrementare un contatore non messo a punto o quello di un altro loop.

Un GOTO rinviato al NEXT termina il loop in corso e fa passare al loop seguente (o alla sequenza se il contatore ha così raggiunto il limite fissato).

Dal momento che un programma contiene dei loop, interviene la dimensione tempo: i loop possono lavorare un gran numero di volte (ricordiamo che i processori sono disciplinati) ed è facile realizzare un programma dando l'impressione che lavori lentamente. Quando si vuole un funzionamento automati-

co senza l'intervento dell'utilizzatore ad ogni inserimento d'informazione, è necessario realizzare un temporizzatore che s'arresterà uno o due secondi per lasciare il tempo di leggere un messaggio. Vi è pure la necessità di misurare il tempo trascorso per fare della sintesi musicale e per fare sistemi a telecomando.

Un loop "vuoto" che non fa altro che rincrementare un contatore, permette di attendere un lasso, di tempo lungo quanto si vuole. Volendo è possibile regolarlo e si può arrivare ad ottenere soste molto precise. La regolazione è più delicata se il programma deve fare qualcosa nello stesso tempo: allora si hanno forzatamente dei test di diramazione e dunque dei tratti disuguali in lunghezza delle istruzioni svolte ad ogni giro. Se ciò intralcia, è necessario "riempire" i rami corti con delle istruzioni inutili, ciò rallenterà l'insieme. Si può quindi avere bisogno di un vero orologio in un tempo reale, vale a dire, al quale si può accedere ad ogni momento, ma che misura il tempo senza essere influenzato dall'attività del processore.

Sottoprogrammi

Un sottoprogramma evita la duplicazione delle sequenze d'istruzioni identiche che è necessario eseguire più volte nel corso di un programma.

Un GOTO permetterebbe di accedere a quella parte ripetitiva, ma per ritornare poi nello stesso punto sarebbe richiesto un sistema di scambio un pò complesso.

L'insieme GOSUB/RETURN è equivalente ad un GOTO speciale che memorizza il punto di partenza per poi ritornare all'istruzione che segue al GOSUB.

Il sottoprogramma termina per mezzo del RETURN che rinvia all'istruzione che seguiva la chiamata del sottoprogramma.

Come per il GOTO, è disponibile una variante di scambio:

ON K GOSUB invio a 1, invio a 2, ecc.

Di solito si scrivono i sottoprogrammi in testa o in coda, ma non al centro di un programma principale, per facilitare la comprensione dello stesso. Se l'esecuzione veloce è importante, la disposizione dei sottoprogrammi non è indifferente. Certi BASIC esplorano i numeri di linea in senso crescente per i GOTO e decrescente per i GOSUB. In questo caso è interessante disporre i sottoprogrammi richiamati spesso in coda al programma.

Un sottoprogramma può chiamarne un altro, ma il BASIC deve immagazzinare i dati di ritorno intermedi e si ha dunque un limite a questa sovrapposizione, di solito sono possibili fino a 16 richiami in cascata.

Per questa stessa ragione, un RETURN senza GOSUB provocherà un errore.

Contrariamente ad altri linguaggi evoluti, il BASIC non permette di scrivere dei sottoprogrammi indipendentemente dal programma che si richiama, poichè uno stesso nome di variabile definisce la stessa variabile ed il sottoprogramma fa parte integrante dell'insieme del programma.

```
10 REM PRINT
20 REM LE ISTRUZIONI SPECIFICHE PER LE PERIFERICHE NON
30 REM SONO DESCRITTE QUI (LPRINT,PRINT$1,...)
40 PRINT A,B,"CUCULO",3.14*LOG(2);"=";"?",Z$(3)
```

```
10 REM SCORRIMENTO AUTOMATICO A FONDO PAGINA
20 X = X+.5:S = 7+6*SIN(X)
30 FOR I=1 TO 64 STEP S:PRINT STRING$(I,">"):NEXT(TRS80)
40 FOR I=64 TO 1 STEP -S:PRINT STRING$(I,">"):NEXT (TRS80)
50 GOTO 20
```

```
10 REM CANCELLAZIONE SCHERMO DA PROGRAMMA
20 CLS            :REM TRS80
20 PRINT"CLR"     :REM PET IL "CLR" E' VISUALIZZATO COME CARAT.GRAFICO
20 CALL-936       :REM INTEGER BASIC APPLE
20 HOME           :REM APPLESOFT
30 REM IN ALCUNE MACCHINE UN POKE CANCELLA LO SCHERMO
```

```
10 REM POSIZIONAMENTO ASSOLUTO
20 REM TRS80 : MOVIMENTI DEL CURSORE CON CHR$(..) E CARATTERI SPECIALI
30 PRINT @64,"INIZIO SECONDA LINEA"
20 REM PET :MOVIMENTI DEL CURSORE CON CHR$(..) E CON TASTI SPECIFICI
30 PRINT"HOME/<-/->....."
20 REM APPLE : CON CHR$(..) E VTAB/HTAB
30 VTAB 2         :REM SECONDA LINEA
```

CAPITOLO 5

DIALOGO CON LA MACCHINA

Visualizzazione e stampa

L'istruzione generale di visualizzazione è PRINT seguita da una lista di argomenti. L'utilizzo di virgole tra gli argomenti provoca una spaziatura automatica, sotto forma di tabulazione implicita, che offre un'edizione corretta (salvo con l'unione di numeri grandi e piccoli) ma che è difficilmente controllabile.

L'utilizzo del punto e virgola offre uno spazio nullo e permette di fare ciò che si vuole, ma si dovrà gestire lo spazio inserendo dei vuoti. Un punto e virgola in fondo alla lista annulla il passaggio automatico alla linea seguente. Il PRINT seguente continuerà la visualizzazione o la stampa di seguito all'ultimo carattere. Il programmatore è così padrone del formato, ma dovrà sorvegliarne tutti i dettagli.

```
10 REM PRINT TAB (TABULAZIONE)
20 REM VISUALIZZAZIONE DI UNA SINUSOIDE
30 X = X+.3
40 PRINT TAB(32+32*SIN(X))"*"    :REM LINEA DI 64 CARATTERI
50 GOTO 30                       :REM 20+20*SIN CON 40 COLONNE
60 REM IMPAGINAZIONE DI UN TABULATO
70 PRINT TAB(5) "*";TAB(8) A;TAB(20) B;TAB(28) "*"
```

```
10 REM ES.:IMPAGINAZIONE COMPLETA DI UN TABULATO
20 CLS           :REM TRS80 =CALL-936 APPLE=HOME APPLESOFT=PRINT"CLR" PET=
30 PRINT TAB(30)"TITOLO"
40 PRINT TAB(30)"======="        :REM OPPURE STRING$(5,"=")
50 PRINT
60 PRINT STRING$(57,"*")         :REM TRATTEGGIO SUPERIORE
70 FOR I=0 TO 6:PRINT TAB(1+8*I)"* ";Z$(I);:NEXT
80 PRINT TAB(57)"*"              :REM TERMINA LE 7 COLONNE DI 8 SPAZI
90 PRINT STRING$(57,"*")
100 FOR I=0 TO 6:PRINT TAB(1+8*I)"*";TAB(4+8*I)T(I);
110 NEXT:PRINT TAB(57)"*"
*********
```

```
10 REM ABBREVIAZIONE DI PRINT CON '?' (TRS80,PET,APPLESOFT)
20 ?:?A;B
LIST 20
20 PRINT:PRINT A;B
```

```
10 REM PRINT USING
20 PRINT USING"###.##";A
                12.33
30 REM NUMEROSE VARIANTI E POSSIBILITA' SECONDO LA MACCHINA
```

Non si può gestire il formato di visualizzazione accontentandosi dello scorrimento automatico: sullo schermo, in basso, l'istruzione PRINT fa scattare lo spostamento di una riga verso l'alto di tutto ciò che è presente questo fà perdere il contenuto della prima linea in alto.

Quando si vuole gestire il formato di visualizzazione, per fare un tabulato o dei giochi, è necessario prevedere tutto; da principio cancellare lo schermo, poi utilizzare diversi comandi secondo il BASIC disponibile: PRINT α (TRS) permette di avviare la visualizzazione in un punto dato dai valori compresi tra 0 e 1023. Si può spostare il cursore dopo averlo ricondotto in alto a sinistra (PET) o fare direttamente una tabulazione verticale e orizzontale (Apple). Si può anche riservare una parte dello schermo per dei dati (INPUT) senza distruggere un tabulato in corso di esecuzione.

```
10 REM ESEMPI DI ISTRUZIONI SPECIFICHE ALLE MACCHINE
20 REM TRS 80
30 PRINT# 1,A,B          :REM USCITA SU CASSETTA
40 PRINT# 2,...          :REM IDEM CON INTERFACCIA
50 INPUT# 1,A,B          :REM INGRESSO DA CASSETTA
60 LLIST                 :REM LISTING SU STAMPANTE
70 LPRINT                :REM STAMPA
----------
80 REM PET
90 PRINT#3,A,B           :REM 3 DEFINITO DA OPEN
100 INPUT#3,...          :REM CON CASSETTA 1,2,STAMPANTE
----------
110 REM APPLE
120 REM USCITA DATI SU CASSETTA TRAMITE SOTTOPROGRAMMA
130 REM IN LINGUAGGIO MACCHINA.
```

La maggior parte dei BASIC hanno la stessa forma di tabulazione orizzontale per mezzo di un comando ausiliare associato alla PRINT che può figurare più volte nell'elenco degli argomenti: TAB (N) provoca lo sposamento del cursore nella colonna N della stessa linea (o della linea seguente se questa posizione è già stata superata).

Per visualizzare una linea vuota (bianca), (in realtà nera sullo schermo in assenza di inversione video), è sufficiente programmare un'istruzione PRINT senza argomento. Se l'ultimo PRINT termina con un punto e virgola, questo nuovo PRINT esegue solamente un ritorno a capo, e un secondo PRINT (secondo ritorno a capo) fornirà una linea vuota.

L'istruzione PRINT è frequente, e parecchi BASIC utilizzano il punto interrogativo come abbreviazione nella scrittura di questa istruzione. Questa abbreviazione è memorizzata (o seguita in forma diretta) e messa in lista come fosse scritta per esteso.

Certi BASIC estesi dispongono di un'istruzione di formattazione (PRINT USING) seguita da una catena di caratteri, che può essere una costante ma anche una variabile referenziata con il proprio nome. Il vantaggio consiste nel

prevedere i diversi campi di applicazione come il numero di decimali da visualizzare, la stampa o meno di zero non significativi, il riempimento a sinistra con asterischi, l'inserimento di spazi (si eviterà l'inserimento di virgole per separare i gruppi di tre cifre, come usano fare gli anglosassoni).

Questo formattamento potrebbe essere realizzato con un sottoprogramma un pò complicato e che sarebbe necessario inserire in ciascun programma.

La possibilità di manipolare i formati con il programma, di utilizzarli per più variabili da visualizzare, agevola notevolmente il lavoro per applicazioni che richiedono molte impaginazioni (per esempio tabulati, contabilità). Per mag-

```
10 REM DIALOGO SEMPLICE
20 PRINT"BATTERE 0 PER CONTINUARE":INPUT A
30 IF A=0 THEN GOTO ....(SEGUITO)
40 END
```

```
10 REM DIALOGOONORMALE
20 INPUT"VOLETE CONTINUARE ";Z$
30 Z$=LEFT$(Z$,1):IF Z$="S"THEN GOTO...'SEGUITO
40 IF Z$="N" THEN GOTO...'FINE          ' E' L'ABBREVIAZIONE DI REM (TRS80)
50 PRINT"RISPONDERE SI O NO"
60 GOTO 20              :REM RIFIUTA RISPOSTE DIVERSE DA SI E NO
70 REM ACCETTA S,SI,SANDRO...
```

```
10 REM DIALOGO SOFISTICATO. NUMEROSE VARIANTI POSSIBILI
20 DIM Z$(3):Z$(0)="RISPONDERE SI O NO"
30 Z$(1)="FATE DUNQUE ATTENZIONE":Z$(2)="ZUT":Z$(3)="MI FERMO"
.........
100 INPUT"DOMANDA ";R$
110 IF R$="RISPOSTA 1"THEN GOTO ....    '(CASO N. 1)
.........
130 IF R$="RISPOSTA ULT."THEN GOTO ...  '(ULTIMO CASO PREVISTO)
140 PRINT Z$(IE)
150 IF IE=3 THEN STOP           :REM ULTIMA PAROLA
160 IE = IE+1:GOTO 100          :REM GRADUAZIONE DELLE RISPOSTE
170 REM IL PROGRAMMA CONTA LE RISPOSTE SCORRETTE E
180 REM PUO' FARNE UNA STATISTICA O UNA NOTA
190 REM SI POTREBBE ANCHE INSERIRE UN TEMPORIZZATORE (ES. PAG.46)
```

```
10 REM INPUT PROTEGGE CONTRO UN RETURN NULLO (PET)
20 INPUT" ←←←";R$               :REM 3 SPAZI,1 CARAT.,3 CURSORE A SINISTRA
30 IF R$="" THEN PRINT"RISPONDETE QUESTO O QUELLO":GOTO 20
40 REM SI EVITA COSI' UN ARRESTO INTEMPESTIVO
```

```
10 REM UTILIZZAZIONE VOLONTARIA DI RISPOSTA NULLA (TRS)
20 A = 3.14:B = 2.718           :REM VALORE IMPLICITO NORMALMENTE
30 INPUT"CAMBIAMENTO DI PARAMETRI ";A,B
40 REM ACCELLERA IL DIALOGO SE UNA RISPOSTA E' PIU' FREQUENTE
```

giori dettagli ci si riporterà allo specifico manuale di ogni personal computer.

L'istruzione PRINT con un numero di indicazione di periferica (lo schermo o display è la periferica privilegiata ed implicita nelle macchine usuali) attiva la

scrittura su una stampante, l'accesso ad un disco o al registratore a cassette. In generale il BASIC sa ciò che deve fare grazie ad una istruzione OPEN che precisa le caratteristiche della periferica. Scrivendo le routine necessarie si potrà pilotare qualsiasi periferica in BASIC: questa operazione è limitata per le macchine semplici fornite del solo schermo.

Tecniche di dialogo

L'essenziale di un linguaggio interattivo come il BASIC è di permettere il dialogo con la macchina. Si tratta in realtà di un dialogo con un programma in corso di esecuzione su quella macchina, e che qualcuno ha programmato con un susseguirsi di domande, di risposte articolate attorno ad un organigramma.

Il computer pone domande al suo operatore grazie all'istruzione INPUT da una lista variabili. Al momento dell'esecuzione il BASIC visualizza un punto di domanda e attende che vengano introdotti dalla tastiera i valori corrispondenti che sono poi assegnati alle variabili ed il programma riprende.

Si possono impostare valori separati da una virgola su una sola linea (terminata con RETURN/ENTER) o una per linea (il BASIC non accetta le impostazioni successive). In caso di errore (in particolare incompatibilità di formato), il BASIC richiede una reimpostazione (REDO?). Certi BASIC ignorano un dato di troppo, altri provocano un errore.

Il punto interrogativo può a volte essere reso più esplicito da un testo tra virgolette in capo alla linea: ciò economizza un'istruzione PRINT per porre la domanda.

La formulazione della domanda può enunciare le possibili risposte: si darà in questo modo un'impressione di "intelligenza" maggiore non facendo apparire queste coercizioni. Il programma allora deve essere in grado di analizzare diverse varianti di risposte, e di condurre l'utente verso una risposta, corretta, indicando, per esempio, le risposte possibili.

L'analisi delle risposte sarà più spinta se si desidera realizzare un programma "riservato" che non si arresterà su risposte di un utente che non conosca il programma (certi BASIC permettono anche di inibire il tasto BREAK).

Una falla appare qualche volta se si batte "ENTER/RETURN" senza nessun inserimento di dati: certi BASIC continuano in sequenza, restando invariate le varianti della lista INPUT. Altri BASIC in questo caso si fermano (READY) e bisogna impostare l'istruzione CONT per riprendere. Si può evitare questo inconveniente visualizzando prima una risposta, poi facendo ritornare indietro il cursore in modo che un'eventuale risposta prenda il posto del carattere già visualizzato.

Ben inteso questo tipo di programma interrogherà su delle stringhe di caratteri unicamente per evitare errori di incompatibilità (carattere letto in formato numerico).

Il dialogo con l'aiuto dell'istruzione INPUT presenta un inconveniente: l'esecuzione è sospesa in attesa dell'impostazione da tastiera dei dati richiesti.

Un'istruzione speciale permettere al BASIC d'accedere ad un carattere appena battuto sulla tastiera. È sufficiente che questa istruzione sia eseguita frequentemente perchè il programma sia preavvertito immediatamente dell'entrata di dati mentre funziona.

```
10 REM ES. DI DIALOGO RAPIDO CHE NON INTERFERISCE CON LO SCHERMO
20 REM ES.:PRESA CONTABILE
........
50 Z$ = INKEY$:IF Z$="" THEN 50          :REM ATTENDE IL SEGUITO
60 REM V VALIDO,R RICAPITOLA,C CANCELLA
70 IF Z$="V" GOTO ...                    :REM ARTICOLO SEGUENTE
80 IF Z$="C" THEN ...                    :REM CANCELLA IL DATO
90 IF Z$="R" THEN ...                    :REM ELENCO DEI DATI
100 PRINT"V=VALIDO, E=ERRORE, R=RICAPITOLAZIONE ?":GOTO 50
110 REM EQUIVALENTE DELLA LINEA 50 SU APPLE:
(50) K=PEEK(-16384):IF K<128 THEN 50
(55) POKE -16368,0:Z$=CHR$(K)
```

```
10 REM LETTURA DI UN NUMERO DA TASTIERA SENZA VISUALIZZARLO
20 REM ES.:CODICE SEGRETO O IMPAGINAZIONE DA SALVAGUARDARE
30 N$ = ""
40 Z$ = INKEY$:IF Z$="" THEN 40          :REM TRS80
40 GET Z$:IF Z$="" THEN 40               :REM PET
40 K = PEEK(-16384):IF K<128 THEN 40     :REM APPLE INTEGER
45 POKE -16368,0:Z$=CHR$(K)
40 GET Z$                                :REM APPLESOFT
50 IF Z$=CHR$(13) THEN 90                :REM BATTUTO RETURN
60 N$ = N$+Z$            :REM CONCATENAZIONE CARATTERE PER CARATTERE
70 GOTO 40               :REM CAR. SPECIALI : TRS=13 - PET=13 - APPLE=141
80 REM NUMERO DISPONIBILE
90 N = VAL(N$)
.........
```

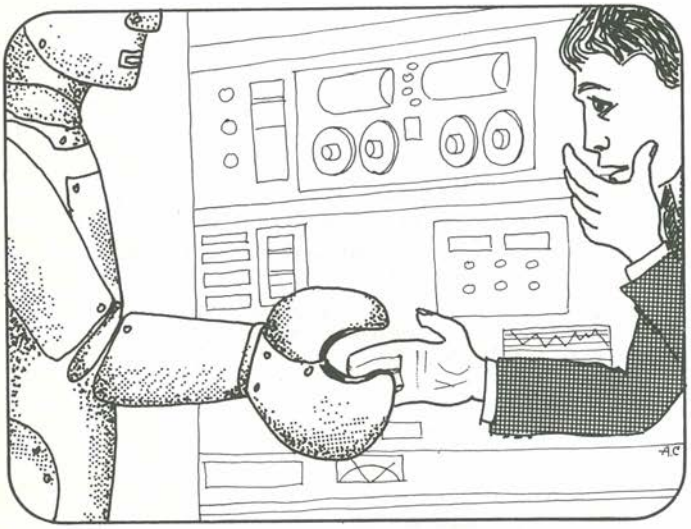
```
10 REM MULTITRATTAMENTO RUDIMENTALE
20 REM ES. DI CONSULTAZIONE DI TABELLE DURANTE L'EDITING DI UNO SCHEDARIO
30 FOR I=1 TO 1000
40 LPRINT "NUM",I;TAB(15)"NOME";TAB(30)N$(I)     :REM STAMPA
50 Z$ = INKEY$:IF Z$<>"" THEN GOSUB 500
60 NEXT
500 IF Z$=CHR$(13) THEN 600      :REM COME ESEMPIO PRECEDENTE
510 T$ = T$+Z$:RETURN            :REM AGGIUNGE CARATTERE DOPO CARATTERE
600 PRINT N$(VAL(T$)):T$="":RETURN
```

Per contro, sta al programma di riunire le parole carattere per carattere, di ricornoscere una parola prevista o un comando ENTER/RETURN. L'esecuzione generale del programma viene allora rallentata.

Si realizza così una interazione attiva con il programma in corso di esecuzione: in un gioco con animazione si possono modificare le traiettorie o provocare differenti incidenti; l'informazione impostata non altera la visualizzazione. Per

un dialogo rapido, si può sopprimere la battitura del tasto ENTER/RETURN se le parole chiave sono conosciute in anticipo. Questa soluzione è pratica se si imposta un carattere alla volta. Si possono pure realizzare effetti speciali (automatismi-musica elettronica-emissione di codice morse).

```
10 REM SCANSIONE TASTIERA E COMANDO DI UN MOVIMENTO
20 REM SI SONO SCELTI DUE TASTI ARBITRARIAMENTE (QUI <- E -> )
30 REM E DETERMINATO EMPIRICAMENTE DUE TEST INDICANTI
40 REM SE L'UNO O L' ALTRO E' PREMUTO
50 I = PEEK(14400):IF I=64 THEN N=N+1:IF N>63 THEN N=63
60 J = PEEK(14400):IF J=32 THEN N=N-1:IF N<0 THEN N=0
70 PRINT TAB(N)"*":GOTO 50
80 REM AGGIUNGENDO OSTACOLI (ALTRI ELEMENTI VISUALIZZATI)
90 REM NELLA STESSA LINEA E TESTS DI N SI OTTIENE UN GIOCO
100 REM DI CORSA AUTOMOBILISTICA
110 REM I VALORI 14400(14336+64),32 E 64 PER I TASTI
120 <- E -> SONO STATI OTTENUTI CON L'AIUTO DEL PROGRAMMA A PAG. 64
```



Questa istruzione speciale non è standardizzata: funzione INKEY$ (TRS) comando GET (P.E.T.) esame della locazione -16384 (Apple) complicato dalla messa a zero della locazione -16368. INKEY$ oppure GET ritorna l'ultimo

tasto battuto dopo l'esecuzione dell'INKEY$ o GET precedente. Vi è dunque la memorizzazione di un solo carattere, che viene perduto se un INKEY$ o GET non lo sfrutta prima della battuta di un nuovo carattere. Se non vi sono state delle battute dopo l'INKEY$ o GET precedente, l'INKEY$ o GET ritorna una stringa vuota anche se il tasto resta abbassato, di modo che si legga una sola volta il carattere battuto. Il GET dell'Applesoft funziona differentemente: il programma si ferma e attende una battuta sulla tastiera il che non permette l'interazione in tempo reale.

Al contrario l'esame degli indirizzi che costituiscono la tastiera (funzione PEEK) darebbe un risultato finchè un tasto resta abbassato: per riconoscere un carattere a questo modo, bisogna inoltre esaminare 6 o 7 indirizzi e decodificare i risultati.

```
10 REM FUNZIONE RND (RANDOM - CASUALE)
20 REM TRS80
30 A = RND(0)              :REM RITORNA UN REALE DA 0 A 1
40 A = RND(N)              :REM RITORNA UN INTERO DA 1 A N
50 RANDOM                  :REM ROTTURA DELLA SEQUENZA (LIMITI INCLUSI)
--------------------
20 REM PET E APPLESOFT
30 I = RND(N)              :REM RITORNA UN REALE TRA 0 E 1
40 REM N IN ALCUNI CASI SERVE DA AGGANCIO A UNA SEQUENZA
50 REM ARGOMENTI DIVERSI DANNO SEQUENZA DIVERSE
60 I = INT(1+(N-1)*RND(36))      :REM DA' UN INTERO DA 1 A N
--------------------
20 REM APPLE INTEGER
30 I = RND(N)              :REM DA' UN INTERO DA 1 A N
--------------------
10 REM UTILIZZAZIONE NEI GIOCHI
20 I = RND(6)              :REM SIMULA UN TIRO AI DADI
30 I = RND(52)             :REM DA' UNA CARTA
40 REM FARE ATTENZIONE AI VALORI LIMITE
50 REM E VERIFICARE CHE ESSI APPAIANO AL MOMENTO DELLA MESSA A PUNTO
--------------------
10 REM ILLUSTRAZIONE GRAFICA DI UNIFORMITA' DELLA CASUALITA'
20 SET(RND(127),RND(47))         :REM TRS80
20 PLOT RND(40),RND(40) :REM APPLE -AGGIUNGERE COLOR=RND(16)
30 GOTO 20
--------------------
10 REM DIMOSTRAZIONE EMPIRICA DI UN PARADOSSO PROBABILISTICO
20 REM "BATTENDO A CASO SU UNA TASTIERA SI E' CERTI
30 REM DI COMPORRE UN QUALSIASI TESTO IN UN TEMPO FINITO"
40 DATA A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,Q,X,Y,Z
50 DIM L$(26):FOR I=1 TO 26:READ L$(I):NEXT
60 INPUT"TESTO ";T$:L1=LEN(T$):C=0
70 I = RND(26):PRINT L$(I);:H$=RIGHT$(H$+L$(I),L1)
80 C = C+1:IF T$<>H$ THEN 70
90 PRINT:PRINT"COINCIDENZA DOPO";C;"BATTUTE A CASO"
100 REM IN PRATICA E' LUNGO, ANCHE PER UN TESTO DI 3 LETTERE!
```

CAPITOLO 6

FUNZIONI SPECIALI

Simulazione con numeri casuali

Tutti i BASIC hanno una funzione che fornisce un numero preso a caso in un intervallo. Si può simulare così una distribuzione statistica, la conduzione di un'impresa con speranze di guadagno e di perdite, e soprattutto programmare numerosi giochi: dadi, roulette, carte, battaglia navale, ecc.

La funzione è in generale chiamata RND (N) ma non è standardizzata.

Alcuni BASIC danno un numero compreso da 0 a 1, e l'argomento tra parentesi è "fasullo" (serve solo per rispettare la sintassi formale della funzione): altri BASIC danno un intero tra 0 e N, oppure 1 e N; infine altri permettono i due modi di funzionamento secondo il valore assegnato a N. Si passa facilmente dal numero tra 0 e 1 al numero tra 0 e N moltiplicando per N, poi prendendo il valore intero (partire da N-1 e aggiungere 1 per arrivare a un numero da 0 a N) e viceversa.

Ci sono vari metodi alla base della funzione RND: l'una consiste nel prendere le cifre di un clock che gira molto rapidamente. Il caso si svolgerà bene se non si richiama la funzione ad intervalli troppo ravvicinati in rapporto alla velocità di calcolo del clock oppure troppo regolari. Un altro metodo consiste nel provocare un superamento della capacità. Si ha allora uno "pseudo-caso" che fornisce sempre la stessa sequenza di numeri "a caso" se si riesegue un programma nelle stesse condizioni a partire dalla messa sotto tensione: ciò può "in certi casi" disturbare e un'istruzione RANDOM o RANDOMIZE permette di rompere quella sequenza all'inizio di un programma.

Esame e scrittura diretta in memoria

Queste possibilità sono molto importanti se ci si interessa a quello che c'è dentro la macchina, per comunicare con dei programmi scritti in linguaggio-macchina e allo stesso modo per comunicare con delle periferiche che occupano una posizione memoria (memory mapped): si può dunque con opportune istruzioni, andare a vedere ciò che succede in memoria.

La funzione PEEK (in inglese: guardare furtivamente) ritorna un numero intero da 0 a 255 che rappresenta il contenuto di un byte di memoria all'indirizzo indicato (da 0 a 65535, qualche volta da -32767 a +32767 con -1 equivalente

```
10 REM FUNZIONE PEEK
20 REM SI METTE IN EVIDENZA IL FUNZIONAMENTO DELLA TASTIERA
30 REM OGNI TASTO COLLEGA ALCUNE LINEE DATI CON ALCUNE
40 REM LINNE INDIRIZZO, SETTANDO A 1 ALCUNI BITS DI CERTI
50 REM BYTES, CHE E' SUFFICIENTE CONTROLLARE
60 DIM A(7):A(0)=14336          :REM TRS80 - INIZIO INDIRIZZI TASTIERA
70 FOR I=1 TO 7:A(I)=A(0)+2↑(I-1):NEXT  :REM A(0)+1,2,4,8,16...
80 FOR I=1 TO 7
90 PRINT TAB(5*I)PEEK(A(I));
100 NEXT:GOTO 80        :REM I VALORI SCORRERANNO IN BASSO ALLO SCHERMO
110 REM SI VEDRA' FACILMENTE L' AZIONE DI CIASCUN TASTO
```

```
10 REM CONSULTAZIONE DI TABELLE DEL BASIC
20 REM ES.:REALIZZAZIONE DELLA FUNZIONE MEM(TRS) O FRE(PET) SU APPLE
30 A1 = PEEK(204)+256*PEEK(205)
40 A2 = PEEK(202))++256*PEEK(203)
50 PRINT"MEMORIA LIBERA = ";A2-A1;" BYTES"
```

```
10 REM FUNZIONE POKE
20 REM ES.:ESPLORAZIONE DEL GENERATORE DI CARATTERI
30 REM ELIMINANDO GLI EFFETTI SUL PRINT DEI CARATTERI SPECIALI
40 A = 15360            :REM INDIRIZZO INIZIALE SCHERMO (TRS) PET=32768
50 FOR I=0 TO 255:POKE A+I,I:NEXT
```

```
10 REM USO DI POKE PER LA MODIFICA DEI PARAMETRI
20 REM LISTA ILLIMITATA (CONFRONTARE I MANUALI RELATIVI)
30 REM TRS
40 POKE 16424,N         :REM N RIGHE PER PAGINA IN STAMPA
50 POKE 16553,255       :REM RIABILITA IL RESTORE DOPO L'USO DELLA
                        CASSETTA O ERRORE SU DATA
60 REM PET
70 POKE 59409,52        :REM SPEGNE LO SCHERMO (60 RIACCENDE)
80 POKE 59468,14        :REM SET CARATTERI MINUSCOLI (12=MAIUSCOLI)
90 REM APPLE
100 POKE 50,63          :REM INVERSIONE VIDEO
110 POKE 50,127         :REM LAMPEGGIO VIDEO
**********
```

a 65535, -32767 a 32768). L'ordine POKE (in inglese: cacciare il proprio naso dappertutto) scrive il carattere corrispondente ad un numero intero entro 0 e 255 all'indirizzo indicato.

Con il PEEK si può esaminare una posizione dello schermo (per esempio per evidenziare in un'animazione, uno scontro e programmare un rimbalzo), si può evidenziare la battuta di un tasto della tastiera (nelle macchine semplici dove la tastiera più economica è costruita su qualche posizione dello spazio di memoria), si possono consultare alcune tabelle gestite in linguaggio macchina, si possono scrivere in BASIC routine di aiuto alla programmazione in linguaggio-macchina (con l'istruzione POKE è possibile realizzare lo svuotamento memoria, disassembler, mini-assembler etc).

Con POKE si può modificare una posizione che non ha importanza in memoria RAM (il solo rischio è di disturbare il programma in corso; vedere le tabelle del BASIC. In questo caso è necessario rinizializzare il sistema con RESET o più drasticamente spegnendo e riaccendendo la macchina). POKE eseguito ad un indirizzo corrispondente ad una locazione di memoria ROM

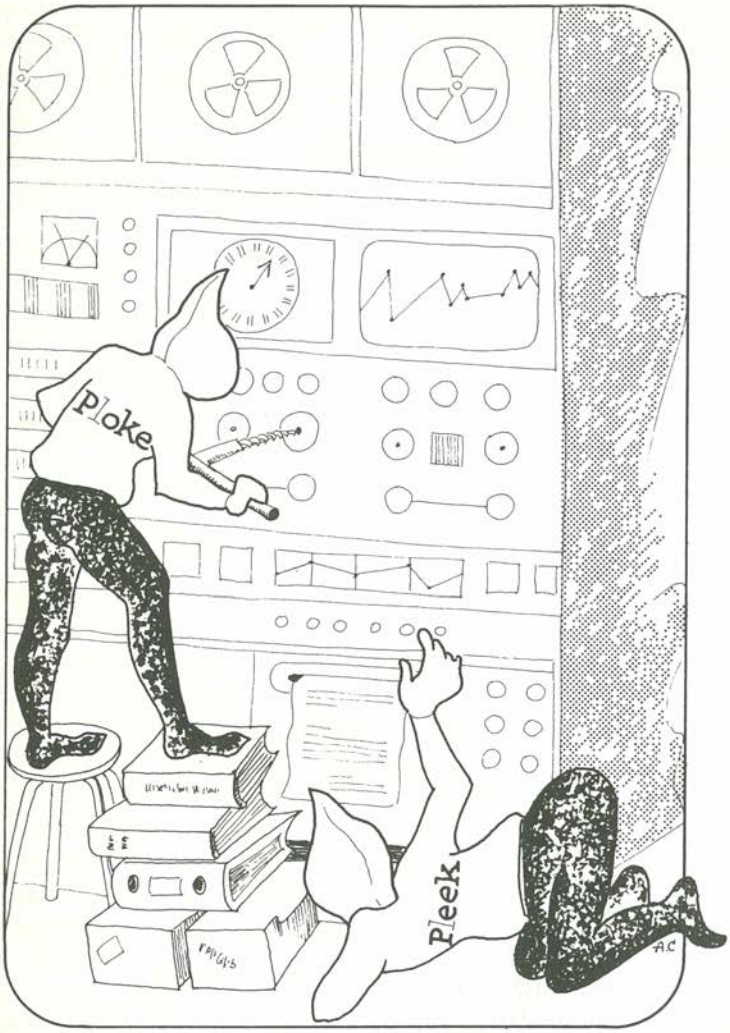
oppure PROM non ha alcun effetto. L'istruzione POKE permette di visualizzare un carattere in qualsiasi posizione dello schermo (nelle macchine comuni dove la memoria di schermo fa parte dello spazio accessibile al microprocessore). Il programmatore deve allora fare attenzione agli errori bizzari di programma dovuti ad una POKE intempestiva sulla zona programma del BASIC. Si realizzano numerosi comandi speciali con una POKE ad un indirizzo particolare (per esempio inversione di video, dimensione di visualizzazione, passaggio ad un set di caratteri grafici, ecc.).

Una periferica può disporre di un indirizzo di memoria (sovente una per

```
10 REM ACCESSO ALLE PORTE I/O: INP E OUT
20 REM COMPLEMENTO DI PEEK E POKE PER LE MACCHINE Z-80 E 8080
30 REM DOVE LE PORTE HANNO INDIRIZZI PROPRI (0-255)
40 REM E NON INDIRIZZI DELLA MEMORIA DEL MICROPROCESSORE
50 REM ES.:TRS (PORTA 255 NELLA VERSIONE BASE)
60 PRINT INP(255)        'RILEVA UN VOLTAGGIO > 0 < 2 V SUL JACK LETTURA
70 OUT 255,0             'PONE 0.5 V SUL JACK DI REGISTRAZIONE
           1                      1 V
           2                      0 V
80 OUT 255,1:OUT 255,2:GOTO 80   'RONZIO PROGRAMMABILE
90 REM IL BASIC E' TROPPO LENTO PER FARE DIRETTAMENTE DELLA MUSICA
100 OUT 255,4            'CHIUDE IL RELAIS (MARCIA CASSETTA)
110 OUT 255,8            'SETTA IL VIDEO A LINEE DI 32 CARATTERI
120 OUT 255,N            'AZIONE MULTIPLA RIFERITA AD OGNI BIT
130 REM ES.: SE N NON CONTIENE IL BIT CORRISPONDENTE A 4
140 REM IL RELAIS E' APERTO.SI POSSONO COMANDARE APPARECCHI BASSO VOLTAGGIO
```

leggere e una per scrivere): vale a dire che l'hardware corrispondente segnala la presenza di questo indirizzo e rilegge allora il carico dei dati (un byte) all'entrata o all'uscita sulla periferica, così realizzabile per mezzo di PEEK o POKE.

Alcuni BASIC hanno delle istruzioni analoghe per indirizzare delle linee di I/O specializzate e non appartenenti allo spazio di memoria del microprocessore: queste sono le istruzioni INP e OUT (vedere i particolari nei manuali). È il caso soprattutto del TRS 80 dove la linea numero 255 serve a comandare l'interfaccia cassetta, la misura dei caratteri visualizzati, il relais di attivazione e arresto del motore del registratore a cassette.


Ploke
Pleek
A.C

```
10 REM ESCLUSIONE AL BASIC DI UNA ZONA DI MEMORIA
20 REM ES.: TRS
SYSTEM
/0
MEMORY SIZE ? 20000              :REM RISERVA GLI INDIRIZZI DA 20000
```

```
10 REM INSERIMENTO DI UN SOTTOPROGRAMMA IN LINGUAGGIO MACCHINA
20 DATA 0,0,0,0,0,201,999
30 REM OGNI CIFRA RAPPRESENTA UN BYTE. QUI IL SOTTOPROGRAMMA
40 REM E' FASULLO (0=OPERAZIONE NULLA,201=RET)
50 REM 999 E' UN INDICE DI FINE PER IL BASIC
60 AD = 20000:A=AD       :REM INDIRIZZO INIZIALE DI INSERIMENTO
70 READ I:IF I>255 THEN          (NON SEMPRE L'INDIRIZZO DI ENTRATA)
80POKE A,I:A=A+1:GOTO 70
90 REM MESSA IN OPERA INDIRIZZO DI CHIAMATA PER USR(N)
100 I% = 20000/256:POKE 16527,I%:REM TRS (POKE 2,... PET)
110 I% = 20000- I%*256:POKE 16526,I%:REM TRS (POKE 1,... PET)
.........
150 A=USR(N)             :REM PROVOCA LA CHIAMATA DEL SOTTOPROGRAMMA
.........
```

```
10 REM CHIAMATA DIRETTA AD UN INDIRIZZO
20 SYS(A)                :REM PET
30 CALL(A)               :REM APPLE
40 REM PROGRAMMAZIONE PIU' SEMPLICE CHE CON USR
```

```
10 REM IDEE DI UTILIZZAZIONE
20 REM CHIAMATA DI NUMEROSE ROUTINES NEL BASIC
30 REM EFFETTI SPECIALI (SFORTUNATAMENTE POCO O NULLA DOCUMENTATI)
40 REM ES.: ACUSTICA,OSCILLOSCOPIO FREQUENZA MEDIA O BASSA
50 REM ES.: SCORRIMENTO DELLO SCHERMO VERSO UN LATO O VERSO L'ALTO
60 REM ES.: PILOTAGGIO DI STAMPANTE NON FORNITO DAL COSTRUTTORE
70 REM NECESSITA' DI CONOSCERE L' HARDWARE E L' ASSEMBLER
```

Sottoprogrammi il linguaggio macchina

La programmazione in linguaggio macchina va oltre lo scopo di questo libro. È un pò più difficile della programmazione in BASIC e dipende strettamente dal microprocessore attorno al quale la macchina è costruita.

Certe funzioni, sono possibili solo in linguaggio macchina. La scrittura risulta più fastidiosa e la messa a punto più lunga: è molto interessante servirsi dei due linguaggi nello scrivere i programmi, in linguaggio macchina solamente lo stretto indispensabile per ottenere la prestazione richiesta, e tutto il resto del programma in BASIC.

Ciò può essere effettuato chiamando un sottoprogramma in linguaggio macchina; come per il GOSUB, il ritorno si farà all'istruzione BASIC seguente. La chiamata si fa per mezzo della funzione USR (N). L'argomento è qualche volta "fasullo" (serve solamente a rispettare la forma sintattica della funzione), qualche volta è un numero intero che sarà comunicato al sottoprogramma richiamato. L'indirizzo del sottoprogramma, prima della chiamata per mezzo della USR (N), deve essere posta per mezzo di una POKE in una certa posizione (16526 e 16527 sul TRS, 1 e 2 sul PET).

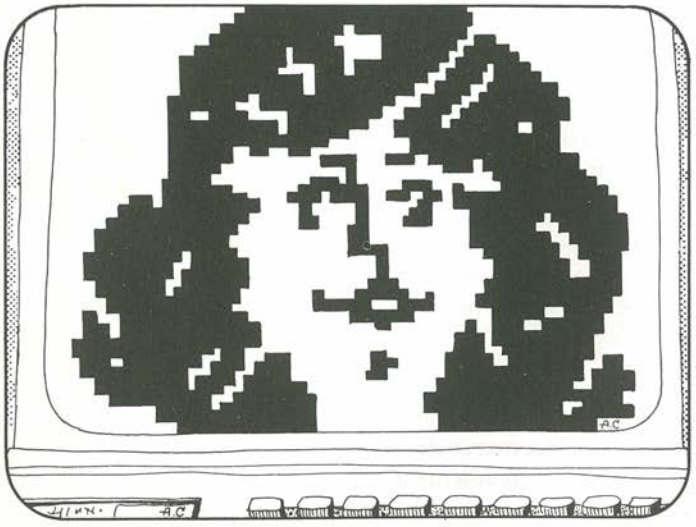
Qualche volta la chiamata è più semplice con un'istruzione SYS (indirizzo) (PET) che evita i POKE preliminari (sono necessarie 2 istruzioni POKE poiché un indirizzo completo sta in due byte), oppure con l'istruzione CALL (Apple).

Il comando SYSTEM seguito dall'indirizzo (TRS) è mal concepito e non può essere utilizzato in un programma BASIC.

Sottoprogrammi semplici faranno scorrere lo schermo in una direzione insolita, o permetteranno di guidare altre periferiche, o di realizzare effetti speciali non accessibili al BASIC del computer utilizzato.

Il BASIC è lui stesso un programma in linguaggio macchina, privilegiato nei confronti degli altri (eventuali) con i quali è necessario definire una suddivisione della memoria, riassunta in un piano di occupazione (questa suddivisione è detta "memory map").

Alla partenza, il BASIC è tenuto a disporre di tutto lo spazio che gli occorre e bisogna tener conto quindi della "memory map" che varia considerevolmente da una macchina all'altra, dato che la zona libera all'utente può essere all'inizio, nel mezzo o alla fine rispetto alle zone riservate al BASIC. Qualche volta la locazione d'inizio della memoria utente può essere modificata all'accensione del sistema (è il ruolo della domanda misteriosa: memory size? del TRS 80 livello 2, alla quale si risponde con un indirizzo al di là del quale la memoria sarà riservata all'utente). Ben inteso, che il linguaggio macchina dipende dal microprocessore e questi sottoprogrammi non saranno necessariamente compatibili da una macchina all'altra.



CAPITOLO 7

EFFETTI GRAFICI E ALTRI EFFETTI

Vi sono più modi per ottenere effetti grafici. Il più semplice consiste nell'utilizzare dei caratteri usuali, specialmente gli asterischi, le lineette, i segni meno o uguale, la lettera I, la barra verticale se questo carattere esiste. Si possono così presentare in modo piacevole le colonne di un tabulato. L'asterisco dà dei tratti meno belli, ma i tratti ottenuti con i segni — = e I sono più difficili da programmare, specialmente per gli angoli e le intersezioni di tratti verticali e orizzontali. I più pazienti aggiungeranno il segno + e distingueranno le linee delle lineette inferiori e superiori (per i bordi dei tabulati), dalle linee del segno.

Se il set dei caratteri disponibile è particolarmente ricco, si possono ottenere effetti semi-grafici: caratteri che permettono di trattare gli angoli dei tabulati, di fare delle sagome con la giusta posizione di semicerchi e di quarti di semicerchi e di ottenere i simboli delle carte da gioco. Questi caratteri specializzati danno degli ottimi effetti grafici (caso del PET), ma non si prestano altrettando bene ad un uso generale.

L'altra opzione semi-grafica consiste nell'imitare la visualizzazione grafica sostituendo ogni punto con un quadrato più o meno grande, che dà un effetto "scacchiera". Se per esempio ci sono 6 caselle per carattere, ciò rappresenta in realtà 64 caratteri completi (2^6) che danno l'immagine di 64 combinazioni possibili (caso del TRS) e la visualizzazione di un punto consiste in realtà nel visualizzare il carattere corrispondente alla nuova combinazione di 6 punti.

Il sistema di utilizzazione di questi caratteri semi-grafici è legata all'hardware. Sovente la memoria video immagazzina caratteri in 7 bits al posto di 8 (per economia). Si può allora disporre di 128 combinazioni che non hanno maiuscole (possibilità di 64 caratteri alfanumerici e 64 caratteri semi-grafici).

Se vi sono più di 128 combinazioni possibili (logicamente 256), occorre passare da un gruppo di 128 all'altro per mezzo di uno scambio dell'ottavo bit.

È ciò che avviene nel PET dove non sono disponibili simultaneamente caratteri maiuscoli e minuscoli (il passaggio si fa per mezzo POKE 59468,14 per le minuscole e 59468, 12 per maiuscole e grafici), mentre sull'Apple o sull'ITT il passaggio è effettuato per mezzo dell'istruzione GR (attivazione della visualizzazione grafica, le 5 linee in basso non cambiano e restano alfanumeriche) e l'istruzione TEXT (riativa la visualizzazione alfanumerica).

I caratteri corrispondenti a una combinazione di caselle (6 nere o bianche sul

TRS, 2 caselle di 16 colori sull'Apple) non sono necessariamente realizzate nel modo consueto con un generatore di caratteri. Sull'Apple un comando speciale fissa i colori dei prossimi punti visualizzati (per spegnere un punto si visualizza un punto di colore nero). Per combinare il colore con un set esteso di caratteri occorre un'architettura più costosa, per esempio 10 bits per ogni carattere visualizzato e non più un byte per 128 simboli (7 bits) con 8 colori (3 bits).

```
10 REM PRESENTAZIONE DI TABULATO CON I CARATTERI USUALI
20 FOR J=0 TO 36:?"=";:NEXT:PRINT
30 REM EQUIVALE A PRINT STRING$(37,"=")
40 FOR J=0 TO 5:PRINT"I←←←←←";:NEXT:PRINT"I":REM ←=SPAZIO
50 FOR J=0 TO 5:PRINT TAB(6*J)"I ";J;:NEXT:PRINT"←←I"
60 FOR J=0 TO 36:PRINT"=";:NEXT:PRINT
```

```
10 REM VISUALIZZA UN OMINO CON IL PET
20 A$="←←↑2 : REM 2 MOVIMENTI DEL CURSORE
   A SINISTRA E 1 IN BASSO
30 B$="◠"+A$+"◡"+A$+"┻"+A$+"▌"+A$+"◠"
40 PRINT B$;"..." : REM VISUALIZZA L'OMINO
```

```
10 REM VISUALIZZARE UN AEROPLANINO CON IL TRS80
20 A$ = CHR$(140)+CHR$(157)      :REM
30 PRINT A$
40 REM QUESTO METODO DA' UNA VISUALIZZAZIONE PIU' RAPIDA CHE CON SET
50 REM E RESET.PRIMA DI TUTTO BISOGNA STABILIRE UNA TAVOLA DI CORRISPONDENZA
```

```
10 REM VISUALIZZARE LA TAVOLA DEI COLORI PRESENTI SU APPLE
20 GR            :REM SET DI CARATTERI GRAFICI=QUADRATI COLORATI
30 FOR I=0 TO 15:COLOR=I:PLOT 0,I:NEXT
40 REM PER ACCELERARE LA VISUALIZZAZIONE IN BASIC SI POTRA', COME SOPRA,
50 REM CERCARE LE CORRISPONDENZE (1 CAR. SPECIALE= 2 CASELLE COLORATE
60 REM E UTILIZZARE PRINT AL POSTO DI PLOT+COLOR. GR E' SEMPRE NECESSARIO
70 TEXT          :REM SET DI CARATTERI NORMALI
```

```
10 REM SET/RESET E PLOT/COLOR. LE COORDINATE CARTESIANE
20 REM CONVENGONO SOPRATTUTTO SE LA FORMA EQUIVALE A UN' EQUAZIONE
30 REM ALTRIMENTI BISOGNA VISUALIZZARE PUNTO PER PUNTO (NOIOSO)
40 FOR I=I1 TO I2
50 SET(I,Y):SET(X,I):SET(I,I)    :REM TRATTI ORIZZONTALI,VERTICALI,DIAGONALI
60 SET(I,10-I):NEXT              :REM 2' DIAGONALE
70 REM IDEM CON PLOT (APPLE)
```

Per comandare una vera visualizzazione grafica, per esempio con una definizione di 512 x 512 punti, ci si imbatte in due difficoltà. Una simile immagine rappresenta 64 Kbyte d'informazione (punti bianchi o neri senza gradazione) e questa massa d'informazioni sarà sempre lenta da trattare con un BASIC interpretato. Del resto il costo dell'hardware necessario è generalmente assai elevato.

Visualizzazione in coordinate cartesiane

È il sistema di istruzioni SET (TRS 80) e PLOT (Apple). Gli argomenti dati sono un'ascissa (0 a sinistra, 127 oppure 39 a destra) e un'ordinata (0 in alto, 47 o 39 in basso). Si ha dunque un quadro positivo con l'asse delle ordinate (Y) inverso in rapporto alle convenzioni matematiche usuali.

Questo sistema permette di prorgammare la visualizzazione di curve e di disegni diversi. Permette di tracciare dei tratti orizzontali o verticali.

```
10 REM ES. DI CURVE: SINUSOIDI
20 N = 1
30 FOR X=0 TO 127          :REM 0 -> 39  PER APPLE
40 Y = 23+23*SIN(X/N)      :REM 11+11*SIN ""
50 SET(X,Y):NEXT           :REM PLOT X,Y  ""
60 N = N+1                 :REM SINUSOIDI SEMPRE PIU' PIATTE
70 CLS:GOTO 30             :REM GR PER APPLE
80 REM ALTRI ESEMPI PAGG. 28 E 30
```

```
10 REM MOVIMENTO SEMPLICE E RIMBALZOO
20 REM SI DEFINISCE UNA VELOCITA' PER MEZZO DEI COMPONENTI VERT. E ORIZ.
30 REM IL RIMBALZO SI OTTIENE INVERTENDO IL SEGNO DI UNA COMPONENTE
40 DEFINT V,X,Y            :REM DICHIARATI INTERI PER ACCELLERARE
50 VX=1:VY=1:X=8:Y=5       :REM VELOCITA' INIZIALE E PARTENZA
60 RESET(X,Y)              :REM APPLE COLOR=0:PLOT X,Y:COLOR=...
70 X=X+VX:Y=Y+VY:SET(X,Y):REM MOVIMENTO DI 1 VALORE X E Y
80 IF X<2 OR X>125 THEN VX=-VX  :REM RIMBALZO
90 IF Y<2 OR X>45 THEN VY=-VY   :REM*APPLE <2 OR >37
100 REM IL MARGINE PERMETTE VELOCITA' DI +/- 2
..... AZIONI DI GIOCO, TEST DI INCONTRI, MODIFICHE DI VELOCITA'
160 GOTO 60
170 REM IL SET (70) E' APPENA PRIMA DI RESET (60),COSI' SI MINIMIZZA
180 REM LO SPEGNIMENTO.PER AVERE SET PRIMA DI RESET OCCORRONO COOR. INTERMEDIE
```

```
10 REM PRINT E UTILIZZAZIONE DI UN CARATTERE COME PROIETTILE
20 Z$=CHR$(183)            :REM SCELTA DI UN CARAT. GRAFICO
30 PRINT CHR$(23)          :REM PASSAGGIO AI CARATTERI DOPPIA LARGHEZZA
40 PRINT@N,Z$              :REM OGGETTO IN POSIZIONE N
50 N = N+1:PRINT@N,Z$;     :REM UN PASSO A DESTRA ECC.....
```

```
10 REM ROTAZIONE DI UNA RUOTA (PET) SULLO STESSO PUNTO
20 A$="/ ←": B$="\↑ ←": C$="/← ←": D$="\← ↑"
30 PRINT A$;:GOSUB 50:PRINT B$;:GOSUB 50:PRINT C$;
40 GOSUB 50:PRINT D$;:GOSUB 50:GOTO 30   :REM LOOP
50 FOR I=1 TO 30:NEXT:RETURN     :REM RITARDO REGOLABILE
```

Animazione

L'animazione consiste nel combinare effetti grafici in un tempo prefissato. Potendo eseguire i programmi assai velocemente, si può ottenere l'illusione dei movimenti, assai rapidi se il programma è semplice come per un gioco con la palla.

Se è necessario programmare dei movimenti complessi con più oggetti, il

BASIC sarà generalmente un pò lento; i video giochi specializzati vengono programmati in linguaggio macchina oppure a partire da circuiti integrati specializzati che realizzano simultaneamente la visualizzazione sullo schermo e la gestione dei punti in movimento.

Un modo seplice, ma grossolano, per ottenere un movimento è quello di utilizzare lo scorrimento automatico messo in moto dall'istruzione PRINT in basso sullo schermo. Un perfezionamento consiste nell'interagire con questo movimento apparente senza fermarlo, (vedere anche INKEY$, GET e PEEK).

Una complicazione supplementare consiste nel sottoporre a test la nuova posizione del punto mobile prima della visualizzazione, ciò che segnala uno scontro o una collisione. Un'altro miglioramento consiste nello spostare più punti indipendenti o ancora nello spostare un punto mobile avente una forma, cioè più punti da spostare simultaneamente.

I punti saranno ottenuti per mezzo di SET o PLOT (punti più fini, ma esecuzione un pò più lenta) o per mezzo di PRINT o POKE (punti corrispondenti a un carattere, 6 o 2 volte più grandi, ma esecuzione più rapida). Se il punto mobile non lascia traccia, bisogna cancellare la vecchia posizione dopo aver visualizzato la nuova, per diminuire l'impressione dello sfarfallio.

I movimenti con PRINT sono ottenuti sia con la variante indirizzo assoluto (PRINTα del TRS) sia visualizzando una stringa formata dal carattere che serve da elemento mobile concatenato (in caso di bisogno in sandwich tra diversi caratteri speciali di movimento del cursore. Questo metodo dà dei movimenti semplici da scrivere ma nei quali è più complicato seguire la posizione; per esempio per segnalare una collisione, rimbalzare sui bordi senza provocare l'azione standard dell'istruzione PRINT (vedere lo scorrimento intempestivo in fondo al paragrafo). La disposizione dei caratteri speciali influirà sull'impressione di sfarfallio.

Generalmente il movimento è ottenuto con la visualizzazione dell'elemento mobile nella sua nuova posizione. La logica si complica con un urto su un ostacolo fisso o un altro elemento mobile secondo gli effetti desiderati: l'ostacolo può restare a posto e non occorre cancellarne la precedente posizione; un altro elemento mobile può cambiare di direzione e nella messa a punto si incontreranno sovente delle difficoltà nella definizione dei test successivi. La più corrente difficoltà riguarda prima di tutto l'uscita dello schermo per mezzo di PLOT o SET oltre le coordinate visualizzabili. Poi in certi casi di rimbalzi, la precedente posizione sfugge alla cancellazione e l'elemento mobile lascia tracce sgradevoli che provocheranno esse stesse altri urti e rimbalzi. Altre eventuali difficoltà: la collisione sfuggirà al test a causa di un cattivo intreccio delle visualizzazioni e delle cancellazioni o, ancora più noioso, perchè le traiettorie si incrementano con delle velocità superiori a 1 e le posizioni istantanee non coincidono. Occorre allora fare il test a tutto il sistema per scoprire l'incontro.

Altri effetti speciali

La prima generazione di personal computer non disponeva affatto di effetti acustici. L'Apple possiede un altoparlante, in realtà un bip acustico che permette qualche semplice effetto: segnalazioni di errore d'impostazione (es. carattere alfabetico in una zona riservata a numeri); utilizzazione per giochi. In linguaggio macchina si può comandare la frequenza del suono ottenuto e suonare una musica elementare (toni semplici, sibilii, glissando lento o rapido).

```
10 REM MOVIMENTO DI UN OGGETTO CHE OCCUPA PIU' CARATTERI
20 REM SECONDO IL SENSO DEL MOVIMENTO. BASTA CANCELLARE UNA
30 REM PARTE DELLA POSIZONE PRECEDENTE
40 REM ES.: SPOSTAMENTO ORIZZONTALE DELL' AEREO PAG .72 (TRS)
50 A$ =" "+CHR$(140)+CHR$(157)  :REM LO SPAZIO INIZIALE SI SPOSTA
51 REM                          E CANCELLA IL DISEGNO
60 FOR I=0 TO 500
70 PRINT@I,A$;:NEXT
80 REM VARIANTE SENZA PRINT@ (INDIRIZZO ASSOLUTO SULLO SCHERMO)
90 A$=" "+CHR$(140)+CHR$(157)+CHR$(24)+CHR$(24):REM 2 CURS. A SIN.
100 PRINT A$:GOTO 100
```

```
10 REM SPOSTAMENTO DELL' OMINO PAG. 72 (PET)
20 A$ ="←←↓"    :REM 3 CURS. A SINISTRA 1 IN BASSO
30 B$ =" ^"+A$+" v"+A$+" ⊥"+A$+" ||"+A$+" ^"(DISEGNO DELL' OMINO)
40 B$ =B$+"←←↑↑↑"       :REM CURSORE A POSTO PER IL SEGUENTE
50 PRINT B$;:GOTO 50    :REM O DISEGNO VERSO DESTRA
                           ¥
60 REM SE SI VUOLE UNO SPOSTAMENTO IN TUTTI I SENSI BISOGNA AGGIUNGERE
70 REM DEI BLANK E ← PER CANCELLARE TUTTO
80 REM QUI SI CANCELLA SOLO LA META' SINISTRA
```

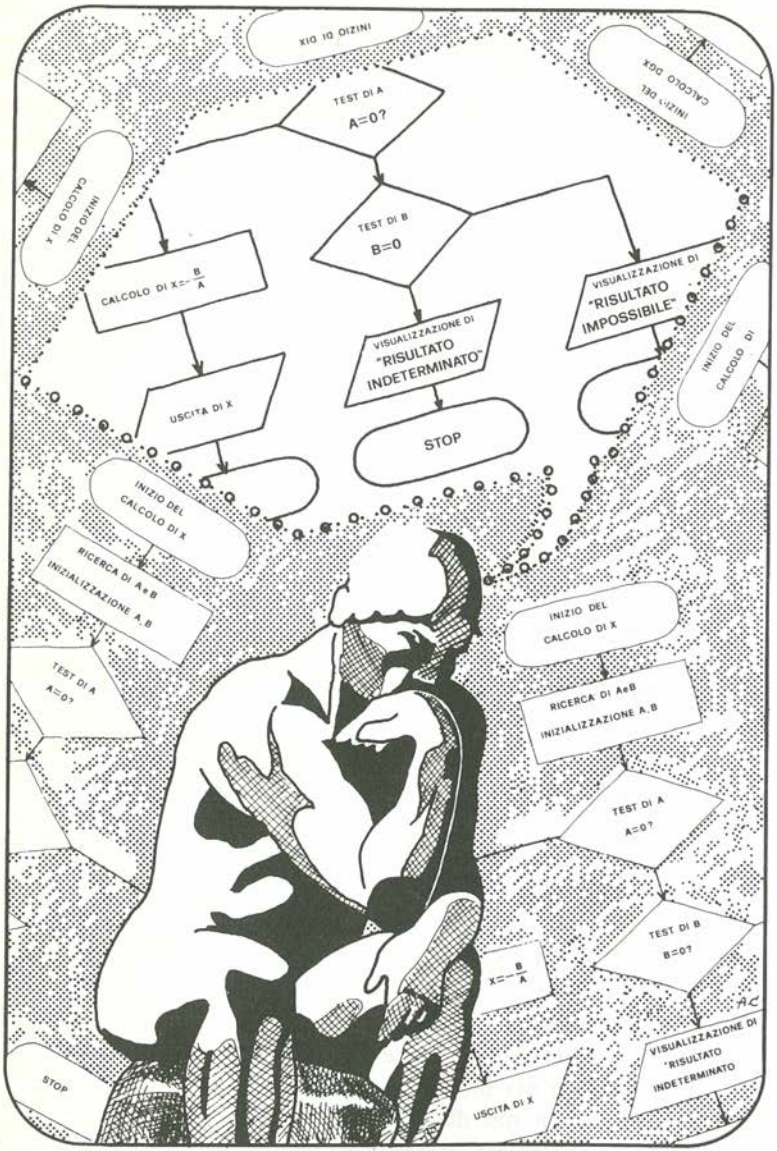
```
10 REM SPOSTAMENTO GENERALIZZATO DI UN OGGETTO INSCRITTO IN UN
11 REM                  RETTANGOLO X1←X2,Y1←Y2
20 AD = 15360:  REM INDIRIZZO INIZIALE SCHERMO
30 A1 =AD+64*Y1+X1:A2=AD+64*Y2+X2        :REM ANGOLI
40 H =Y2-Y1:L=X2-X1
50 AJ =A1:FOR J=0 TO H:A=AJ
60      FOR I=0 TO L:POKE A-64,PEEK(A):A=A+1:NEXT I:REM A-64←63,65
70 AJ =AJ+64:NEXT J

80 AJ =A2:FOR J=0 TO H:A=AJ
90      FOR I=0 TO L:POKE A+64,PEEK(A):A=A+1:NEXT I:REM A+64←63,65
100 AJ =AJ-64:NEXT J

110 AI =A1:FOR I=0 TO L:A=AI
120     FOR J=0 TO H:POKE A-1,PEEK(A):A=A+64:NEXT J:REM A-1←-65,+63
130 AI =AI+1:NEXT I

140 AI =A2:FOR I=0 TO L:A=AI
150     FOR J=0 TO H:POKE A+1,PEEK(A):A=A-64:NEXT J:REM A-1←+65,-63
160 AI =AI-1:NEXT I
```

Certi personal computer più recenti permettono l'accesso a uno o più generatori di suoni direttamente dal BASIC. Alcuni permettono di generare delle forme di caratteri per mezzo di programmi (non ancora disponibili a livello stampanti). Altri hanno delle possibilità di interfaccia a diverse periferiche: Bus IEEE 488 (istruzioni CMD del PET), joystick, penna ottica, comando di relais elettrici, questo dialogo si realizzerà con le istruzioni PEEK-POKE (o INP-OUT).


TEST DI A
A=0?
TEST DI B
B=0
CALCOLO DI X=- B/A
USCITA DI X
VISUALIZZAZIONE DI "RISULTATO INDETERMINATO"
VISUALIZZAZIONE DI "RISULTATO IMPOSSIBILE"
STOP
INIZIO DEL CALCOLO DI X
RICERCA DI A e B
INIZIALIZZAZIONE A,B
TEST DI A
A=0?
TEST DI B
B=0?
X=- B/A
USCITA DI X
VISUALIZZAZIONE DI "RISULTATO INDETERMINATO"
STOP

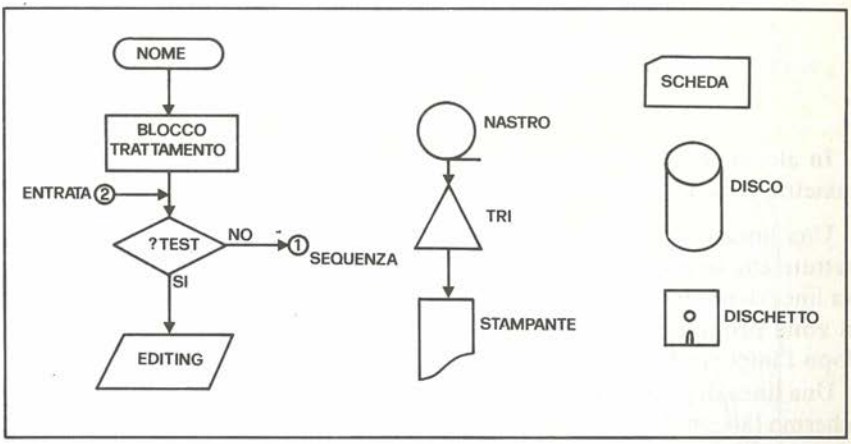
CAPITOLO 8

PREPARAZIONE DEI PROGRAMMI

Prima di incominciare un programma, ad eccezione di casi molto semplici, è quasi indispensabile fare uno schema che rappresenti il concatenamento delle differenti fasi di esecuzione.

Esistono dei segni convenzionali rappresentati qui sotto. Non è indispensabile utilizzarli, ma è necessario disimpegnare dei blocchi la cui funzione sia chiaramente definita: variabili utilizzate in entrata, quel che effettua il blocco, entrate-uscite.

Questa riflessione si chiama analisi e lo schema che ne risulta è l'organigramma (chiamato in inglese flow-chart). Ciascun blocco sarà descritto da vari commenti che saranno inclusi vantaggiosamente nel programma.



Scrittura di un programma

Alla accensione, le macchine più comuni sono pronte a ricevere la battitura di un programma o di comandi; l'interprete BASIC visualizza un segno di riconoscimento (prompt) ed esamina la tastiera. Si dice che ci si trova in ambiente BASIC.

```
READY              PRONTO
>                  (INDICAZIONE DEL SISTEMA BASIC)
10 REM  **************************
20 REM  *  E  S  E  M  P  I  O  *
30 REM  **************************
40 REM
50 REM LE NOTE SERVONO AL COMMENTO E
60 REM ALL' IMPAGINAZIONE. UGUALMENTE AI SISTEMI D'IMPIEGO
70 REM           QUESTA E' UN'ISTRUZIONE MOLTO MOLTO MOLTO MOLTO M
OLTO LUNGA (A CAPO AUTOMATICO). ANNULLAMENTO DELL'IMPAGINAZIONE
-----------------
0 REM NUMERO DI LINEA MINIMO
65529 REM NUMERO DI LINEA MASSIMO (TRS APPLE). A VOLTE 32767
-----------------
PRINT LOG(2)     :REM MODO COMANDO ESEGUITO IMMEDIATAMENTE
0.695
10 REM OGNI ISTRUZIONE PUO' ESSERE ESEGUITA IN MODO COMANDO
20 REM E VICEVERSA
30 REM IL MODO COMANDO PERMETTE DI FARE CALCOLI O TEST
40 REM IMMEDIATAMENTE SU UN' ISTRUZIONE INCERTA
-----------------
10 GOTO 225      'EDIZIONE FINALE  '=REM (TRS80)
-----------------
10 REM PUO' SERVIRE AD INIBIRE UN'ISTRUZIONE SENZA DISTRUGGERLA
20 REM E SENZA DOVERLA RISCRIVERE NUOVAMENTE
50 REM ON ERROR GOTO 200          'E' MEGLIO EFFETTUARE LA MESSA A PUNTO
60 REM                            'PRIMA DI ATTIVARE L'INTERCETTAZIONE DI ERRORE
-----------------
10 REM PUO' SIMULARE UN'ETICHETTA INDIRIZZO
20 GOTO 120
120 REM FINE REGOLARE
130 END
```

In alcuni casi è necessario caricare dapprima l'interprete a partire da una cassetta, o uscire da un sistema monitor o altro (vedere manuali specifici).

Una linea di programma, o un comando, è realizzata da una sequenza di battute che termina con un ritorno carrello (ENTER/RETURN). Differenze tra linee di programma e comando: l'una porta un numero ed è immagazzinata in zona programma: l'altro non ha numero ed è eseguito immediatamente dopo l'impostazione ENTER/RETURN e non viene memorizzato.

Una linea di programma può contenere più caratteri rispetto ad una linea di schermo (80 per il PET, 255 per il TRS 80 per esempio). Una linea di programma può contenere più istruzioni separate da: (due punti), salvo che in certi BASIC limitati. Le linee lunghe rendono i programmi meno chiari (linee visualizzate su parecchie linee di schermo), più difficili da mettere a punto, ma per contro, più compatti (perchè molte istruzioni hanno in comune il numero di linea e il ritorno carrello).

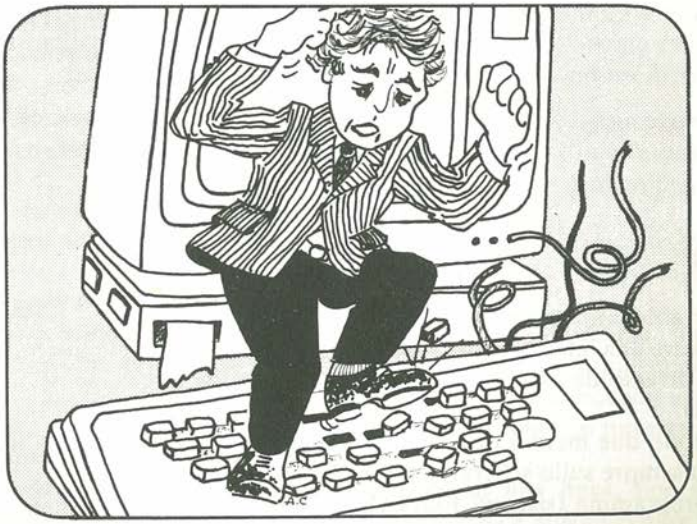
L'istruzione REM permette di effettuare titoli, note che facilitano la comprensione del programma, linee bianche di presentazione. Si ha interesse a servirsene per gli altri... e soprattutto per se stessi! Unica contro indicazione: se

si ha una limitazione della capacità di memoria. Tutto ciò che segue REM nella linea viene ignorato al momento dell'esecuzione, ma immagazzinato in zona programma.

Se un programma è un pò lungo, constaterete che si fanno meno errori preparando una minuta, la più appropriata possibile e scritta solamente su una facciata del foglio.

```
10 REM MESSAGGIO DI ERRORE
20 REM TUTTI I BASIC DANNO MESSAGGI DI ERRORE CON UN NUMERO,SIGLE
30 REM O PIU' O MENO CHIARI. ES.:BS=BAD SUBSCRIPT (ERRORE D'INDICE)
```

```
10 REM TRATTAMENTO D'ERRORE DA PROGRAMMA
20 REM SI PUO' SEMPRE EVITARE L'INTERRUZIONE SU ERRORE CON
30 REM TESTS CHE COMPLICANO E RALLENTANO L'ESECUZIONE.
40 REM "ON ERROR" PERMETTE DI ELIMINARE QUESTI TEST A PRIORI,INTER-
50 REM CETTANDO UN ERRORE E TRATTARLO A POSTERIORI.
..........
80 ON ERROR GOTO 200      :REM ROUTINE DI TRATTAMENTO IN 200
..........
120 ON ERROR GOTO 0       :REM DISATTIVA LA ROUTINE (FUNZIONAMENTO NORMALE)
..........
200 REM ROUTINE D' ERRORE
210 IF ERL=90 THEN..      :REM ERL=VARIABILE CHE DA' LA LINEA DOVE SI E' PRODOTTO
220 IF ERL=110 THEN..     :REM L'ERRORE;QUI SI SOSPETTANO LE LINEE 90 E 110
230 IF ERR/2+1=1 THEN..   :REM ERR=VARIABILE CHE DA' IL CODICE DI ERRORE
240 IF ERR/2+1=4 THEN..   :REM QUI SI TRATTANO GLI ERRORI 1 E 6
..........
260 RESUME                :REM CONTINUAZIONE DEL PROGRAMMA PRINC. ALLA
261                       :REM STESSA LINEA; RESUME NEXT -> ALLA SEGUENTE
262                       :REM RESUME 150 -> ALLA LINEA 150
```



Questo sforzo iniziale è praticamente indispensabile se il vostro sistema non comprende una stampante per mettere a punto un programma che si è scritto, è necessario scoprire gli errori, che sono sempre presenti.

Alcuni errori, come gli errori di battuta, sono facilmente rilevabili e correggibili.

Altri errori riguardano la sintassi, le regole non rispettate, l'ausilio dei messaggi di errore è sovente sufficiente a scoprirli.

Ma rimangono alcuni errori più subdoli detti logici che risultano indirettamente dalle parti di programma eseguite prima.

Risolvere questi errori - qualche volta molto sottili - fa parte del fascino della programmazione. È praticamente impossibile riuscirci se non si lavora sulla carta, almeno che non si tratti di un programma che occupi, in lunghezza, un solo schermo.

Per le stesse ragioni, occorre riportare accuratamente su questa minuta tutte le correzioni effettuate (a meno che naturalmente non possediate una stampante) anche se considerate che il programma è a punto e che il lavoro è terminato.

Lavorate così anche per le successive modifiche che non mancherete di introdurre: per esempio per perfezionare il programma, o per correggere un errore ancora più sottile o corrispondente ad un caso non ancora incontrato.

In mancanza di queste precauzioni si rischia di mettere a dura prova il sistema nervoso... È un peccato maltrattare eventualmente una tastiera che non ha alcuna colpa, o rinunciare a portare a termine la programmazione di una buona idea; ed è molto fastidioso confrontare l'ultima versione immagazzinata in cassetta con una minuta non aggiornata!

Come conseguenza logica, il programmatore previdente deve premunirsi di una buona gomma e di un portamine (mine sottili: da 0,5 a 0,9 mm) in mancanza di una stampante. Se si desidera lavorare confortevolmente, occorre disporre di un buon blocco, un tavolo libero e ben illuminato.

Questi consigli possono apparire semplicistici, ma vengono dimenticati facilmente soprattutto dai "bricoleurs" invaghiti dai fili elettrici e dagli oscilloscopi, nell'eccitazione di affrontare il calcolatore.

Editing

Non abbiamo presentato qui le funziondi di editing che permettono di modificare una linea di programmazione senza ribatterla per intero. Esse sono molto diverse da una macchina all'altra, e si farà riferimento ai manuali specifici.

Esistono due metodi principali: i comandi ed il risultato della modifica si vedono sempre sullo schermo, ma in un caso il BASIC lavora direttamente in zona programma (sistema EDIT del TRS), nell'altro la modifica si fa sulla

linea visualizzata (dunque in zona schermo) e la battuta di RETURN o ENTER invia la nuova linea in zona programma (caso del PET).

Ciascun metodo presenta dei vantaggi e degli inconvenienti. Il secondo necessita delle possibilità di movimenti di cursori e di inserimento e soppressione per mezzo di più sofisticati tasti-tastiera; le modifiche richiedono più manipolazioni, ma queste possibilità di inserimento e soppressione sono utilizzabili nell'esecuzione di un programma per esempio per comporre un disegno sullo schermo.

Nel primo metodo i comandi più potenti (si rimpiazzano i tasti con delle parole-chiave) facilitano le modifiche, ma non sono utilizzabili che nel sistema Edit e i vari movimenti del cursore sono disponibili solo per mezzo di programma (vedi CHR$) e non dalla tastiera in sistema BASIC.

Altre possibilità di editing, non fornite sulle macchine più comuni, permettono di rinumerare e d'inserire delle parti di programma messe a punto. Nello stesso modo una funzione di duplicazione (COPY dalla linea n alla linea p) economizza delle battute su sistemi più costosi: molto spesso, ci sono delle istruzioni un pò lunghe e molto simili ed è pratico copiarle automaticamente e poi inserire qualche modifica. Queste funzioni possono essere realizzate con dei programmi ausiliari da scrivere (o comprare) ma non saranno veramente pratici se non con un sistema a floppy disk.

Numerazione delle Linee

Tutti i programmi in BASIC devono avere le linee numerate. Questo numero gioca parecchi ruoli.

La presenza di un numero indica che le istruzioni della linea che è stata impostata da tastiera, fanno parte di un programma in corso di scrittura o di modifica. Senza questo numero, le istruzioni vengono eseguite immediatamente.

Quest'ultimo modo di lavorare, chiamato comando diretto, permette di utilizzare la macchina come una sofisticata calcolatrice:

```
PRINT SQR (2), (10 000 x 12) / 360.5
```

Essa permette anche di esaminare o modificare le variabili in un programma che si è interrotto per un errore oppure di riprendere l'esecuzione interrotta, ciò che facilita molto spesso la messa a punto.

Il numero di linea serve da indirizzo di riferimento per modificare una linea di programma o annullarla e per introdurre delle linee supplementari. È necessario dunque che questi numeri siano scelti (ordine crescente) e spaziati per combinare possibilità di successive aggiunte. Di solito, si numera inizialmente di 10 in 10.

Se l'interprete BASIC accetta più istruzioni su una linea, non si sarà, teoricamente mai limitati da mancanza di spazio, ma è meglio evitare le linee

troppo lunghe, poichè una falsa manovra (errore di battuta o di correzione) può obbligare a ribattere tutta la linea, con il medesimo rischio di errore.

La numerazione serve infine da etichetta (indirizzo) per i disinserimenti, vale a dire le interruzioni di sequenza durante l'esecuzione del programma (vedi GOTO).

Questa scelta semplificatrice è uno dei pochi punti deboli del BASIC: in altri linguaggi, l'etichetta è indipendente dal numero di linea, cifra o meglio nome non cifrato, che rende i programmi più facili da documentare (si scrive GOTO END al posto di GOTO 910) e più chiari.

Questa semplificazione rende difficile la rienumerazione automatica delle linee, corrente con altri linguaggi, che rende un programma finale più estetico, elimina il rischio di non poter più inserire delle linee e soprattutto permette di fare coabitare in memoria diversi sottoprogrammi scritti separatamente

Ciò non è possibile in BASIC se non sono state previste prima zone di numeri distinti per ciascun sottoprogramma.

Alla impostazione iniziale di un grosso programma, certi computer permettono di economizzare la battuta dei numeri di linee, grazie al comando AUTO. Dal momento che si è battuta una linea (ENTER o RETURN) il BASIC visualizza il numero seguente.

Si arresta l'istruzione AUTO con un tasto (Break sul TRS) o con un comando (MAN sull'apple). Si può riprenderla più volte e facilitare la comprensione e la messa a punto, riservando delle zone di numeri alle differenti parti del programma e dei sottoprogramma.

```
AUTO
10 REM TRS80 COMANDO CHE PERMETTE LA NUMERAZIONE AUTOMATICA
20 REM DELLE LINEE. A OGNI FINE LINEA,IL BASIC
30 REM PRESENTA IL NUMERO SEGUENTE.SI FERMA CON BREAK
AUTO 200,5
200 REM NUMERA AUTOMATICAMENTE
205 REM DI 5 IN 5 A PARTIRE DA 200
210 REM (AUTO 200  FAREBBE DI 10 IN 10)
```

```
AUTO    (APPLE) IDEM ECCETTO CHE SI FERMA CON 'MAN' ANZICHE' CON BREAK
```

```
10 REM UTILIZZAZIONE DI GRUPPI NUMERICI PER DISTINGUERE
20 ...  :REM ALCUNE FASI DEL PROGRAMMA DAI SOTTOPROGRAMMI
30 ...
AUTO 100
100 *           :REM L' ASTERISCO SEGNALA CHE E' GIA' PRESENTE UNA LINEA 100
110 ...         :REM CHE ANDRA' DISTRUTTA SE SI CONTINUA
120 ...:END
AUTO 1000
1000 ...
1090 RETURN
```

Gli anglosassoni hanno un termine figurato per la messa a punto: debugging. Questo stadio è di solito lungo quanto la scrittura del programma.

È raro che un programma funzioni al primo colpo, ma un programma scritto chiaramente, con delle annotazioni, sarà messo a punto più velocemente. Inoltre, è nella messa a punto che i linguaggi interpretati offrono i maggiori vantaggi; al limite si può avviare l'esecuzione (RUN) senza rileggere e correggere man mano gli errori semplici del tipo: errore di battuta o di disattenzione, che provocano l'arresto con un messaggio.

Il lavoro serio inizia in seguito: se il funzionamento non è soddisfacente, è necessario seguire la traccia del cammino logico percorso dalla macchina e verificare che sia conforme a quello che si è scritto nell'organigramma. Per questo si dispone, su certi computer, di una istruzioni TRACE (o TRON) che visualizza il numero di ciascuna linea eseguita.

Per non occupare lo schermo, si può inibire questa azione nelle parti del programma già a punto.

Una volta individuato il tracciato, si possono inserire provvisoriamente degli STOP così all'arresto si può esaminare e provvedere alla modifica di qualsiasi variabile e riprendere l'esecuzione come se non ci fosse stata l'interruzione (CONT - Continue) o in un punto qualsiasi (GOTO). È ancora un vantaggio degli interpreti e del comando diretto.

In aggiunta al TRACE, certe macchine hanno una istruzione DSP (Display) che visualizza automaticamente le variabili specificate, ogni volta che il programma le incontra.

Si ottiene un risultato equivalente mettendo dei PRINT provvisori nei punti chiave.

Si può facilitare l'eliminazione delle istruzioni provvisorie (TRACE, PRINT, STOP) al fine di mettere a punto, dando dei numeri provvisori facili da individuare, per esempio: 11, 21, 131 (o 19, 29, 139) dietro le istruzioni 10, 20, 130.

Una frequente difficoltà riguarda gli algoritmi ed altri loop ripetuti fino a che una condizione sia verificata. L'errore più comune è dimenticare l'inizializzazione (per esempio accumulatore non rimesso a zero e funzionante al primo passaggio poichè RUN rimette le variabili a zero).

Si hanno anche i valori limite: bisogna posizionare il contatore da 0 a 1 alla partenza, fare un test su una disuguaglianza stretta o in senso ampio ($I <= N$ oppure $J<N + 1$...).

Se non è troppo difficile, concepire un loop in circuito permanente è generalmente impossibile scrivere direttamente la parte di programma corrispondente ai valori iniziali: è meglio scrivere su una minuta i valori assunti dalle variabili su i primi giri (e gli ultimi) e di modificare a tentoni il programma.

Più generalmente, un metodo empirico consiste nel fare in sequenza tutti i calcoli a mano fino a che si veda apparire un carattere ripetitivo, ciò permette di liberare un algoritmo e di concepire un organigramma dettagliato.

```
10 REM TRACE/NOTRACE     (OPPURE TRON/TROFF)
20 REM USATO COME COMANDO,SI APPLICA A TUTTO IL PROGRAMMA
30 REM USATO NEL PROGRAMMA, SI APPLICA AD UNA PARTE
40 IF I>100 THEN 350
41 TRACE
50 ON K GOTO 60,80,90
60 K=2:GOTO 50
80 K=1:GOTO 50
90 K=3:.....
RUN
50 60 50 80 50 60 50 80 50 60 50 80 50 60 50 80 ........
--------------------
51 PRINT"K=";K           :REM ISTRUZIONE PROVVISORIA
RUN
50 60 50 80 50 60 .......IL PRINT NON E' STATO MESSO AL GIUSTO POSTO
--------------------
10 REM DSP/NODSP         (APPLE) VISUALIZZAZIONE
20 REM VISUALIZZA LE VARIABILI INDICATE OGNI VOLTA CHE
30 REM SONO MODIFICATE.
40 REM EVITA MOLTI PRINT PROVVISORI MA, COME TRACE,
50 REM TENDE A RIEMPIRE LO SCHERMO ED E' NECESSARIO INSERIRE
60 REM NUMEROSI NODSP, POI ANCORA DSP
71 DSP           :REM INTERESSA TUTTE LE VARIABILI
81 NODSP
91 DSP K,A1
100 K=I:GOTO 120
110 K=0
.........
--------------------
RUN              :REM ESECUZIONE DALL'INIZIO      CON CANCELLAZIONE
RUN 150          :REM      "      DALLA LINEA 150   "      "
GOTO 150         :REM      "      DALLA LINEA 150 SENZA     "
--------------------
CONT             :REM RIPRENDE L' ESECUZIONE FERMATA DA STOP O BREAK
                 :REM NON FUNZIONA DOPO END
```

Esecuzione

L'esecuzione di un programma si avvia con il comando RUN.

In generale questo comando inizializza a zero tutte le variabili del programma.

Se si utilizza l'istruzione GOTO come comando diretto (vale a dire senza numero di linea e dunque eseguito subito) si avvia l'esecuzione al punto indicato, senza rimessa a zero delle variabili.

Se il programma non si arresta da solo, si può fermare il BASIC con il tasto BREAK o STOP che viene esaminato dall'interprete ad ogni esecuzione di istruzione. Se il programma ha richiamato un sottoprogramma in assembler, è possibile arrestare l'assembler solo con il tasto RESET, quando esiste, altrimenti la sola possibilità è quella di spegnere e riaccendere il sistema.

L'istruzione CONT permette di riprendere l'esecuzione di un programma opo un arresto provocato da uno STOP.

Questa istruzione è particolarmente utile durante la messa a punto.

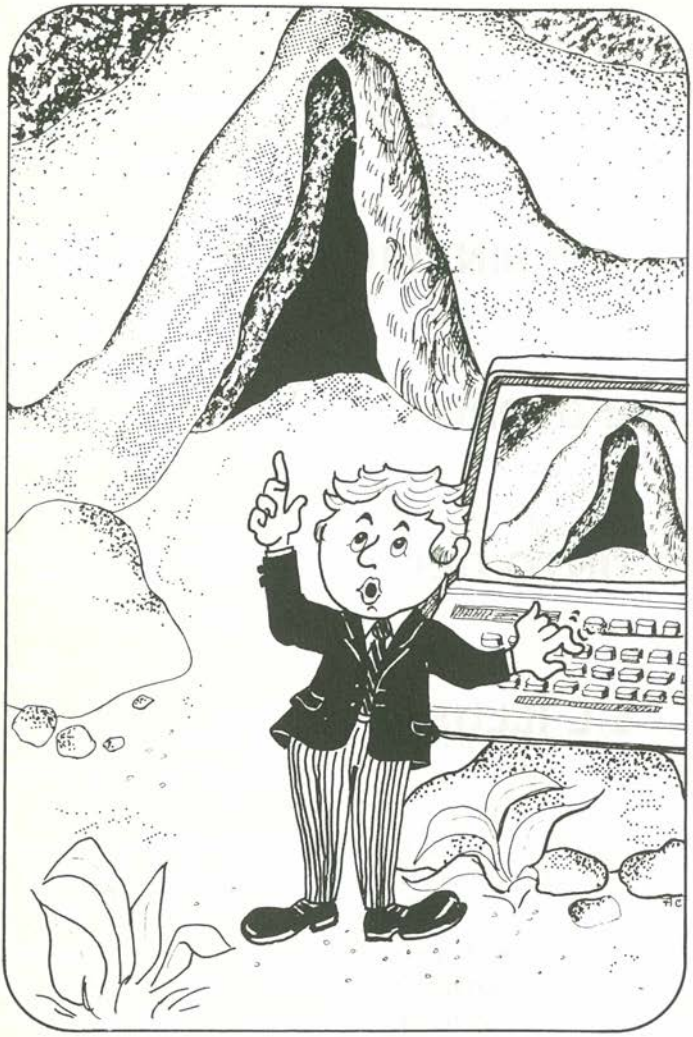
1 CODICE ASCII E CARATTERI SPECIALI

2 CALCOLO BINARIO E ESADECIMALE

3 ESEMPI DI PROGRAMMI

4 RIASSUNTO DEI COMANDI BASIC

5 INDICE E ILLUSTRAZIONI



APPENDICE 1

CODICE ASCII E CARATTERI SPECIALI

È un codice internazionale utilizzato dalla maggior parte dei computer (con un'eccezione l'IBM) per rappresentare delle informazioni alfanumeriche (cifre-lettere-segni) nella memoria e nei registri dei calcolatori. Il format-carattere oppure stringhe di caratteri utilizza il codice ASCII (American Standard Code For Information Interchange) in tutte le piccole macchine.

Il codice normalizzato della ISO (International Standard Organisation) con delle piccole varianti per ogni paese, è definito su 7 bits 128 combinazioni possibili. Le macchine comuni hanno una struttura a byte (8 bits) e resta una ambiguità sull'ottavo bit: a seconda delle macchine, esso è a 1 oppure a 0. I 7 bits sono in generale inquadrati a destra e la conseguenza del passaggio a 1 dei bit di sinistra è di aggiungere 128 al valore decimale corrispondente (codici da 128 a 255 al posto dei codici da 0 a 127). Lo si verifica grazie all'istruzione BASIC PRINT ASC ("A") che darà sia 65, sia 193 (= 65 + 128).

Per comprendere bene la nozione del Codice ASCII e l'azione di certi caratteri speciali, bisogna distinguere tre elementi del mini-computer; la visualizzazione a livello del generatore di caratteri, che in funzione di un byte (o più sovente di 7 bits sulle macchine a basso costo) genera una immagine; la produzione di un byte a partire dalla battuta di un tasto (fatto per mezzo del BASIC sulle macchine a buon mercato, seguito da un'azione); ed infine l'azione attivata con l'istruzione PRINT applicata ad un carattere. Per il codice della lettera A, l'immagine è evidente, la tastiera fabbrica lo stesso codice e PRINT visualizza l'immagine di A. I 3 concetti divergono quando si tratta di caratteri speciali: il tasto di cancellazione genera un codice, questo codice può avere una immagine a partire dal generatore di caratteri; PRINT applicato a questo codice cancella lo schermo, vale a dire, lo riempie del carattere bianco (vuoto) (codice 32).

Stessa difficoltà con i movimenti cursori e certi caratteri grafici (PET). Si esplorerà il generatore di caratteri con POKE (sullo schermo), la tastiera con PRINT ASC (INKEY$) oppure GET C$: PRINT ASC (C$) e l'azione PRINT con PRINT CHR$ (...).

Il codice ASCII riserva certe combinazioni a due funzioni speciali, per esempio 13 per il ritorno carrello, 10 per cursore in basso (line feed) ma queste convenzioni non sono in generale rispettate sulle piccole macchine comuni (sono rispettati solo i codici strettamente alfanumerici).

Questi caratteri sono TC (trasmissione), BEL (campana), BS (cursore a sinistra), HT e VT (tabulazione orizzontale e verticale), FF (form feed), DC (controllo periferiche), ESC (ESCAPE - annuncia un carattere in arrivo utilizzato per una funzione speciale), IS (separatore d'informazione per strutturare un archivio).

I tasti SHIFT e CONTROL non corrispondono ad un codice, permettono solamente di fabbricare 128 o 256 codici a partire da 64 tasti, posizionando a 1 il settimo e l'ottavo bit (in più dei 6 bits per 64 tasti). Si utilizzano abbassandoli simultaneamente con uno dei 64 altri (64 è un numero teorico: si può averne meno, o più con dei tasti speciali di funzione, come break o clear, secondo la costruzione della tastiera).

I caratteri accentati possono essere ottenuti con una stampante: un backspace seguito da virgola (44) farà una cediglia, seguito da apostrofo (39) per accento acuto, da virgolette (34) per dieresi, da freccia in alto (94) per accento circonflesso.

Per effettuare dei trattamenti veritieri di tasti, è necessario uno schermo munito di un generatore di caratteri che disponga di lettere accentate (nessuna sovraimpressione è possibile come sulla stampante); la stampante avrà raramente lo stesso esatto set di caratteri (salvo forse una stampante a matrice) ed il programma di stampa dovrà assicurare la corrispondenza tra i caratteri differenti.

Codice Internazionale ASCII (norme ISO)

— La tavola dà i valori decimali (utilizzabili con CHR$ ASC POKE) e esadecimali leggendo la colonna di sinistra e la linea dall'alto (ES: "OD" per ritorno carrello - CR)

— Ci sono due norme francesi:
senza minuscole: identico all'internazionale salvo | al posto di | in 124 e ~ al posto di - (soprasegnato) in 126,

con minuscole:	35	64	91	92	93	123	124	125	126
	£	à	°	ç	§	é	ù	è	¨
al posto di	#	@	[	\	]	{	/	}	~

— Il codice ASCII è a 7 bits (0÷ 127); certe macchine (APPLE) mettono a 1 l'ottavo bit del byte, ciò che dà i valori da 128 a 255 (esempio A = 128 + 65 = 193).

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL 0	TC1 1	TC2 2	TC3 3	TC4 4	TC5 5	TC6 6	BEL 7	BS 8	HT 9	LF 10	VT 11	FF 12	CR 13	SO 14	SI 15
1	TC7 16	DC1 17	DC2 18	DC3 19	DC4 20	TC8 21	TC9 22	TC10 23	CAN 24	EM 25	SUB 26	ESC 27	IS4 28	IS3 29	IS2 30	IS1 31
2	SP 32	! 33	" 34	# 35	$ 36	% 37	& 38	' 39	(40	) 41	* 42	+ 43	, 44	- 45	. 46	/ 47
3	0 48	1 49	2 50	3 51	4 52	5 53	6 54	7 55	8 56	9 57	: 58	; 59	< 60	= 61	> 62	? 63
4	@ 64	A 65	B 66	C 67	D 68	E 69	F 70	G 71	H 72	I 73	J 74	K 75	L 76	M 77	N 78	O 79
5	P 80	Q 81	R 82	S 83	T 84	U 85	V 86	W 87	X 88	Y 89	Z 90	[91	\ 92	] 93	^ 94	- 95
6	96	a 97	b 98	c 99	d 100	e 101	f 102	g 103	h 104	i 105	j 106	k 107	l 108	m 109	n 110	o 111
7	p 112	q 113	r 114	s 115	t 116	u 117	v 118	w 119	x 120	y 121	z 122	{ 123	| 124	} 125	~ 126	DEL 127

CARATTERI SPECIALI - TASTI DI COMANDO

Azione su PRINT CHR$ e tastiera	TRS codice tasto	PET codice tasto	APPLE codice[1] tasto	
Movimenti cursore				
a sinistra	24	157 ←	esc. K	
a destra	25	29 →	esc. J	
in alto	27	145 ↑	esc. I	
in basso	26	17 ↓	esc. M	
All'inizio della linea	29			
a capo	10,13 ↓			
in alto a sinistra	28	19 Home		
tabulazione	→			
Cancellare				
un carattere a sinistra	8 ←		8←	
inizio della linea	S←		cX	
fine linea	30		esc,E	
il rimanente della pagina	31		esc,F	
tutta la pagina	Clear	147 CLR	esc,sP	
Delimitatore	13 Enter	13 Return	141 Return	
Arresto				
basic	Break	Stop	°C	
sospensione	s@			
rallentam. lista				
Esiting	(Edit) (2)			
soppressione		20 Del		
inserimento		148 Inst		
Effetti speciali				
doppia larghezza car.	23 S→			
ritorno misura Norm.	(3) Clear			
caratter. Grafici	129E 191	da 161 a 223		
cloche			199 cG	
inversione video		18 RVS		
ritorno video norm.		146 OFF		
cursore visibile	14			
cursore invisibile	15			

^{s}X = shift + X cX= control+X esc,X=tasto escape, dopo tasto X

(1) Caratteri speciali impediti da PRINT - l'8° bit è a 1 (A....Z = 193....218) (in Apple Integer BASIC)
(2) Editing per mezzo comandi in sistema EDIT
(3) Possibile senza annullamento per mezzo OUT 255,0

CALCOLO BINARIO E ESADECIMALE

I numeri binari e la loro rappresentazione in ottale (una cifra ottale da 0 a 7 per rappresentare 3 bits) o meglio esadecimale (una cifra da 0 a F per rappresentare 4 bits) giocano un ruolo importante in linguaggio macchina, ma anche in BASIC quando si vogliono utilizzare delle funzioni come PEEK o POKE o comprendere come l'informazione è rappresentata nella memoria e nei registri del calcolatore.

Per contare occorre una base di numerazione (insieme di segni che permettono di scrivere un numero). La base decimale sulla quale apprendiamo a contare dall'infanzia, non è che una base tra altre, che l'uomo ha privilegiato senza dubbio, perchè ha 10 dita: si conta quindi "naturalmente" con l'aiuto di dieci simboli (0,1...9) con i quali formiamo i numeri. In binario si hanno due simboli (per comodità 0 e 1, si potrebbe prendere "ciottolo nero" e "ciottolo bianco"). In esadecimale si hanno 16 simboli (per comodità da 0 a 9 più A,B,C,D,E,F). Con n cifre decimali, si conta da 0 a 10^n -1 (esempio: 999 per 3 cifre). Lo stesso in binario con n bit (1 bit, binary digit, designa una cifra binario), si conta da 0 a 2^n - 1, cioè 0 a 15 con 4 bit, da 0 a 255 con 8, da 0 a 65535 con 16 bits (2 byte).

Queste cifre sono importanti perchè i calcolatori elettronici utilizzano sempre il binario: non si è potuto realizzare fino ad oggi dei sistemi a più stati così rapidi e poco ingombranti se non i circuiti elettronici a due stati (le prestazioni sono ora vicine ai limiti fisici della materia e delle onde elettromagnetiche). Se si avessero degli equivalenti a 10 posizioni (macchine elettroniche) aventi le prestazioni dei circuiti a due stati, nessuno utilizzerebbe più il binario.

Poichè la memoria è organizzata in byte o gruppo di 8 bit (6 permettono solo 64 combinazioni, 7 non permetterebbero di dividere in due, ciò sarebbe fastidioso in parecchi casi), si arriva naturalmente alla capacità di indirizzamento della memoria di 64 Kbyte max (kilo designa per facilità un multiplo di 1024, infatti 64k=65536): gli indirizzi occuperanno 2 byte, un carattere su 1 byte con un'alfabeto che può disporre di 256 caratteri.

Ciò necessità 16 linee di indirizzo e 8 linee di dati a livello hardware, essendo 1 bit rappresentato da una tensione di 0 oppure 5 volts.

Si può utilizzare una base di tipo misto: questo capita sovente in BASIC per

leggere o scrivere un indirizzo con PEEK e POKE: un byte rappresenta una cifra in base 256 e si ha, come con qualsiasi base,

numero = cifra della unità + base × cifra di dozzine + ecc.

cioè: indirizzo = byte meno significativo + 256 × byte più significativo, la "cifra" in base 256 è infatti manipolata dal suo valore decimale ottenuto con PEEK e POKE.

APPENDICE 3

ESEMPI DI PROGRAMMA

Esempio 1 - Orologio del P.E.T. (P.E.T.)
Esempio 2 - Gestione di un piccolo archivio (T.R.S. 80)
Esempio 3 - Gioco di YAM (P.E.T. e T.R.S.) (Pokerissimo)
Esempio 4 - Visualizzazione in grandi caratteri (Apple II)

```
9 REM  ***************************
10 REM * ES. 1 OROLOGIO DEL PET *
11 REM ***************************
12 REM
13 REM
14 REM
20 GOSUB 2000            :REM COSTANTI
30 GOSUB 3000            :REM MODO D' USO
40 GOSUB 4000            :REM REGOLAZIONE ORA
45 IF MDE$="O" THEN MDE$="N":GOTO 30
50 GOSUB 5000            :REM AVANTI
55 IF ST$="S" GOTO 70
60 GOSUB 6000            :REM CARILLON
65 GOTO 50
70 GOSUB 7000            :REM ARRESTO
75 IF ST$="P" GOTO 50
76 GOTO 99
80 REM TEMPORIZZAZIONE
81 GOSUB 8000:RETURN
90 REM RISPOSTA SI O NO
91 GOSUB 9000:RETURN
99 END

2000 REM         COSTANTI
2010 READ LH$,LM$,LS$
2020 DATA " ORE "," MINUTI "," SECONDI "
2100 REM TABELLA REDATTA DELLE ORE
2110 DIM Z$(23)
2120 FOR K=0 TO 23:READ Z$(K):NEXT
2130 DATA ZERO,UNA,DUE,TRE,QUATTRO,CINQUE
......
2190 DATA VENTITRE
2900 POKE 59468,12
2910 POKE 59409,60
2999 RETURN

3000 REM                 MODO D' USO
3010 PRINT"(CLR) → → → → → → → → → →↓↓↓↓↓↓ (RVS)OROLOGIO DEL PET(OFF)"
3020 PRINT:PRINT:PRINT"VI CHIEDERO' L' ORA"
3030 PRINT:PRINT"(RVS)ATTENZIONE(OFF) VOI MI DARETE UN NUMERO"
3040 PRINT:PRINT"DI SEI CIFRE, NELLA FORMA HHMMSS"
3050 PRINT:PRINT"NE' VIRGOLA, NE' PUNTO FRA I GRUPPI"
3060 PRINT:PRINT"DI DUE CIFRE"
3070 PRINT:PRINT"  PER UNA (RVS)REGOLAZIONE DELL' ORA(OFF) PRECISA"
3080 PRINT:PRINT"INDICATEMI IL PROSSIMO MINUTO E"
3090 PRINT:PRINT"DATEMI IL VIA AL PASSAGGIO"
3100 PRINT:PRINT"  PER (RVS)FERMARMI(OFF) BATTETE (RVS)S(OFF)"
3110 PRINT:PRINT"NON IMPORTA QUANDO DURANTE IL PROGRAMMA"
3120 PRINT:PRINT"=*=*= BATTETE (RVS)T(OFF) PER LEGGERE IL SEGUITO =*=*="
3200 GET C$
3210 IF C$="T" GOTO 3998
3220 GOTO 3200
3998 PRINT"(CLR)"
3999 RETURN

4000 REM                 REGOLAZIONE DELL' ORA
4010 PRINT"...>>>>MI METTERO'' IN MARCIA"
4020 PRINT"NELL' ISTANTE IN CUI BATTERETE RETURN"
4030 PRINT:PRINT"DOPO AVER RISPOSTO O ALLA"
4040 PRINT:PRINT"DOMANDA  VA BENE ?"
4050 PRINT:PRINT:PRINT" SE RISPONDETE N"
4060 PRINT:PRINT"RICOMINCEREMO LA REGOLAZIONE DELL'ORA"
4070 PRINT:PRINT
4080 INPUT"CHE ORA E'?";MH$
4090 IF VAL(MH$)=0 GOTO 4500
4100 IF VAL(MH$)>240000 GOTO 4500
4110 H$ = LEFT$(MH$,2)
4120 M$ = MID$(MH$,3,2)
4130 S$ = RIGHT$(MH$,2)
```

```
4140 E$ = H$+LH$+M$+LM$+S$+LS$
4150 PRINT:PRINT"HO LETTO : ";E$
4160 PRINT"D' ACCORDO";:GOSUB 90
4170 IF R$="N" GOTO 4070
4180 TIME$ = MH$
4190 PRINT"(CLR)"
4200 GOTO 4999
4500 PRINT:PRINT" VI SIETE SBAGLIATI, VOLETE"
4510 PRINT:PRINT"RILEGGERE LE ISTRUZIONI D' USO";:GOSUB 90
4520 IF R$="N" GOTO 4070
4530 MDE$ = "0"
4999 RETURN

5000 REM                  CONTEGGIO
5010 H$ = LEFT$(TIME$,2)
5020 M$ = MID$(TIME$,3,2)
5030 S$ = RIGHT$(TIME$,2)
5040 GET ST$
5050 IF ST$="S" GOTO 5999
5060 PRINT"(CLR)→→→→→→→→→→→→↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓  ";H$:LH$;M$;LM$;S$;LS$
5999 RETURN

6000 REM                  CARILLON
6010 MS$ = RIGHT$(TIME$,4)
6020 IF VAL(MS$)<>0 GOTO 6999
6030 H = VAL(LEFT$(TIME$,2))
6040 PRINT"(CLR)"
6050 FOR I=1 TO 100
6060 PRINT"(RVS)  (OFF) ";Z$(H);LH$;
6070 NEXT I
6080 GOSUB 80             :REM TEMPORIZZAZIONE
6090 IF H<2 GOTO 6998
6100 FOR J=2 TO H
6110 POKE 59409,52        :REM SPEGNE LO SCHERMO
6120 GOSUB 80             :REM TEMPORIZZAZIONE
6130 POKE 59409,60        :REM RIACCENDE LO SCHERMO
6140 GOSUB 80
6150 NEXT J
6998 PRINT"(CLR)"
6999 RETURN

7000 REM                  ARRESTO
7010 PRINT"(CLR)↓↓↓↓↓↓↓↓↓↓↓↓↓"
7020 POKE 59468,14        :REM MINUSCOLE
7030 PRINT"AVETE INTERROTTO IL FUNZIONAMENTO"
7040 PRINT"TUTTAVIA POTETE RIMETTERMI IN"
7050 PRINT"MOTO IMMEDIATAMENTE SENZA DOVER"
7060 PRINT" RICOMINCIARE LA REGOLAZIONE DELL' ORA"
7070 PRINT"SI RICOMINCIA";:GOSUB 90
7080 IF R$="S" THEN ST$="P":GOTO 7900
7090 PRINT"(CLR)↓↓↓↓↓↓↓↓↓↓"
7100 PRINT"ARRIVEDERCI E GRAZIE. A VOSTRA DISPOSIZIONE"
7110 PRINT"PER FARE ... CIO' CHE VORRETE,"
7120 PRINT"QUANDO VORRETE, COME VORRETE"
7130 GOSUB 80:GOSUB 80            :REM TEMPORIZZAZIONE
7900 POKE 59468,12                :REM MAIUSCOLE
7910 PRINT"(CLR)"
7999 RETURN

8000 REM                  TEMPORIZZAZIONE
8100 FOR T=1 TO 1500::NEXT T
8999 RETURN
9000 REM                   RISPOSTA SI O NO
9010 INPUT R$
9020 R$=LEFT$(R$,1)
9030 IF R$="S" OR R$="N" THEN 9999
9040 PRINT "(RVS)BATTERE(OFF) S (RVS) PER SI E (OFF) N (RVS) PER NO"
9050 GOTO 9010
9999 RETURN
```

```
                                                                ES. 2
1 REM ********************************************** SCHEDARIO
2 REM *  ES. 2  GESTIONE DI UN PICCOLO SCHEDARIO  * CHE TIENE
3 REM ********************************************** IN MEMORIA
5 REM
10 CLEAR 2000            :REM RISERVA 2000 BYTES PER LE STRINGHE (PRIMA DI DEF)
20 DEFINT I-N:DEFSTR A,Z         :REM INTERI E STRINGHE (DEF E' AZZERATO CON CLEAR)
30 Z0 = CHR$(28)+CHR$(30)        :REM CURSORE IN ALTO-SINISTRA E RIGA CANCELLATA
35 REM LA LINEA IN ALTO SERVE AI MESSAGGI E AGLI INPUT
40 Z1 = Z0+"999 = FINE/NUMERO/RETURN SOLTANTO=SEGUENTE"
50 Z2 = Z0+"RETURN SOLTANTO=NESSUNA MODIFICA/NUOVO VALORE"
60 Z3 = Z0+"PREPARATE LA CASSETTA-BATTERE UN TASTO PER METTERE IN MOTO"
70 DIM A1(50),A2(50),A3(50)      :REM MAX. 50 ARTICOLI DI 3 SOTTOPARTI
75 REM                           DA SISTEMARE SECONDO I CASI

80 REM           INIZIO
90 CLS:L=1023:D$="0"     :REM PRINT@1023 FARA' SCROLLARE LO SCHERMO
95 REM                   PRINT Z0;... SCRIVERA' IN ALTO0
100 PRINT Z0;"SCHEDARIO ESISTENTE O NUOVO ";:INPUT Z
110 IF Z="N"THEN N=0:PRINT Z0;"NOME DEL NUOVO SCHEDARIO";:INPUT F$:GOTO 600
120 IF Z="E" THEN 150
130 PRINT Z0;"RISPONDERE E O N":FOR I=0 TO 500:NEXT:GOTO 100

140 REM                  CARICAMENTO DELLO SCHEDARIO
150 PRINT Z3;"LETTURA"
160 IF INKEY$="" THEN 160        :REM LOOP DI ATTESA BATTUTA
170 INPUT#-1,N,F$,D$             :REM INIZIO SCHEDARIO
180 PRINT@L,"SCHEDARIO = ";F$;" /DATA = ";D$;" N. ART = ";N+1
190 FOR I=0 TO N:PRINT Z0;I:INPUT#-1,A1(I),A2(I),A3(I):NEXT
195 REM          IL PRINT PERMETTE DI SEGUIRE LO SVOLGIMENTO

200 REM                  PARTE COMUNE
210 REM
230 PRINT Z0;"CONSULTAZIONE, MODIFICA, AGGIUNTE, FINE (A-C-F-M)":INPUT Z
240 IF Z="C" THEN 300
250 IF Z="M" THEN 400
260 IF Z="A" THEN 600
270 IF Z="F" THEN 800
280 PRINT Z0;"BATTERE A,C,F,M":FOR I=0 TO 500:NEXT:GOTO 230

300 REM                  CONSULTAZIONE
310 K = 0                        :REM K=NUMERO ARTICOLO
320 PRINT Z1;:INPUT K:IF K>N THEN 230   :REM N= NUMERO DI ARTICOLI
330 PRINT@L,K,A1(K),A2(K),A3(K);
350 K=K+1:GOTO 320
360 REM K=K+1 PERMETTE DI VEDERE L'ARTICOLO SEGUENTE BATTENDO SOLO RETURN

400 REM                  AGGIORNAMENTO
405 REM
410 REM SI OPERA PER NUMERO DI ARTICOLO (DA 0 A N)
415 REM NEL CASO SI TROVERA' QUESTO NUMERO CON  CONSULTAZIONI SUCCESSIVE
420 PRINT Z1;:INPUT K:IF K>N THEN 230
430 PRINT@L,K,A1(K),A2(K),A3(K);
435 REM VISUALIZZA IL VECCHIO DATO PRIMA DELL' AGGIORNAMENTO
450 A=A1(K):PRINT Z2;A;:INPUT A:A1(K)=A
460 A=A2(K):PRINT Z2;A;:INPUT A:A2(K)=A
470 A=A3(K):PRINT Z2;A;:INPUT A:A3(K)=A
475 REM OGNI PARTE DEL VECCHIO ARTICOLO E' VISUALIZZATO; SI INTRODUCE
476 REM IL NUOVO VALORE O SOLO RETURN SE NON E' CAMBIATO
480 PRINT@L,K,A1(K),A2(K),A3(K);
485 REM E' MOSTRATO IL NUOVO ARTICOLO
490 K=K+1:GOTO 420
495 REM BATTENDO RETURN SOLTANTO SI AGGIORNA L' ARTICOLO SEGUENTE
```

```
600 REM                    AGGIUNTE
605 REM
610 REM SI EFFETTUA CON INSERZIONE RISPETTANDO L'ORDINE DELLO SCHEDARIO
615 REM CHE RESTA ORDINATO IN PERMANENZA
617 REM PERMETTE UGUALMENTE LA CREAZIONE (INGRESSO IN 600 CON N=0)
620 PRINT Z0;"Z=ARRESTO/ NOME DA INSERIRE";:INPUT A
630 REM E' STATO SCELTO QUI UN ORDINAMENTO CON CONFRONTO DI PARTE A1-NOME
640 IF A="Z" THEN 200
645 K=N
650 IF A<=A1(K) THEN K=K-1:GOTO 650
655 REM RICERCA PER NOMI DECRESCENTI IL PRIMO <= A
660 K=K+1                  :REM POSTO DEL NOME DA INSERIRE
670 IF A=A1(K) THEN 760    :REM CASO DI OMONIMIA
680 IF K>N THEN 720        :REM IN TESTA ALLA PILA, SALTA LO SPOSTAMENTO
690 FOR I=N TO K STEP -1          :REM SPOSTA IN GIU' TUTTI I SEGUENTI
700 A1(I+1)=A1(I):A2(I+1)=A2(I):A3(I+1)=A3(I)
710 NEXT
720 N=N+1                  :REM AGGIORNA IL NUMERO DI ARTICOLI
730 A1(K)=A:PRINT Z0;"INTRODUCETE GLI ALTRI DATI";
735 INPUT A2(K),A3(K)
740 PRINT@L,K,A1(K),A2(K),A3(K) :REM VISUALIZZA IL NUOVO ARTICOLO
750 GOTO 620
760 REM          OMONIMIA
770 PRINT Z0;"ESISTE GIA'";:FOR I=0 TO 500:NEXT:GOTO 200
780 REM E' STATO SCELTO DI RIFIUTARE GLI OMONIMI. (LO STESSO NOME
785 REM SEGUITO DA UN BLANK NON E' OMONIMO)


800 REM                    FINE
810 REM
820 PRINT @ L,F$;"CONTIENE ";N+1;" ARTICOLI"
830 PRINT Z0;"0=RITORNO ALL' INDIETRO/INTRODUCETE LA DATA DI AGGIORNAMENTO"
840 INPUT A:IF A<D$ THEN 200
850 REM VENGONO RIFIUTATE LE DATE PRECEDENTI QUELLA DI PARTENZA
860 REM
870 PRINT Z3;                     :REM MESSAGGIO DI PREPARAZIONE
880 IF INKEY$="" THEN 880:        :REM ATTENDE UNA BATTUTA
890 PRINT#-1,N,F$,A               :REM SCRIVE SU CASSETTA
900 FOR I=0 TO N:PRINT#-1,A1(I),A2(I),A3(I):PRINT Z0;I;:NEXT
910 PRINT Z0;"TERMINE"
920 END

    REM PROGRAMMA PER TRS
    SONO POSSIBILI NUMEROSE VARIANTI. SI ADATTERA' FACILMENTE
    LA STRUTTURA DELLO SCHEDARIO. SU APPLE (INTEGER) OCCORRERA' CHE I
    SOTTO-ARTICOLI SIANO DI LUNGHEZZA COSTANTE. PER GESTIRE LA LINEA DI
    SERVIZIO VI E' SEMPRE UN CAR. "HOME". PER SOSTITUIRE "CANCELLA FINE-
    -LINEA CHR$(30)" SI USERA' CALL-868.
    GLI INKEY$ NON SONO INDISPENSABILI (=INPUT).SI SOSTITUIRANNO
    I PRINT@ CON VTAB E HTAB
1000 REM ES. DI VVARIANTI
     AGGIUNTA DI SALTI AL PROGRAMMA PRINCIPALE ES.: IF Z="E" THEN 1000
     ARTICOLAZIONE DIFFERENTE (FINE - INIZIO)       [ 1000=CAPITOLO DI EDITING
     CONSULTAZIONE SU NOME
     320 PRINT Z0;"NOME O NUMERO";:INPUT A      :REM NUMERO 0 INACCESSIBILE
     321 K = VAL(A):IF K>N THEN 320
     322 IF K=0 THEN            :REM A=ALFABETICO
     323 PRINT@ L,K,A1(K),A2(K),A3(K);:GOTO 320
     324FOR K=0 TO N:IF A=A1(K) THEN 323
     325 NEXT
     326 PRINT Z0;"SCONOSCIUTO":FOR J=0 TO 500:NEXT:GOTO 320
     ECC....
```

```
1 REM  **********************
2 REM  *   GIOCO DEL YAM    *
3 REM  **********************

       PROGRAMMA PER PET E TRS80. STA IN 4 KBYTES SE SI ELIMINA
       IL MODO D' USO, LIMITA IL NUMERO DEI GIOCATORI A 2 E IL DIALOGO
       INIZIALE AL MINIMO.

       UN RIASSUNTO DEL GIOCO SI TROVA NELLE ISTRUZIONI D' USO.
       SI GIOCA CON NUMEROSE VARIANTI (TABELLE DIVERSE,ALTRE COMBINAZIONI)
       CHE SI INTRODURRANNO FACILMENTE

4 CLEAR 220:RANDOM       :REM TRS (ALTRIMENTI ERR=OUT OF STRING SPACE)
5 DEFINT A-Z             :REM TRS GUADAGNO SPAZIO E VELOCITA'
6 REM SALVO DEFINIZIONE ULTERIORE OGNI VARIABILE SARA' INTERA
10 DIM N(4),NG(4)        :REM 5 DADI (DA 0 A 4)
15 DIM N$(4),T(4,12),B(12):REM 5 NOMI,12 COMBINAZIONI (0=PUNTEGGIO)
20 DATA 1,2,3,4,5,6,20,30,40,50,1,1      :REM TABELLA
22 REM DA 1 A 6, FULL, POKER, COLORE, ,POKERISSIMO (YAM),SCALA MINIMA E MASSIMA
25 FOR I==11 TO 12:READ B(I):NEXT

29 REM DIALOGO INIZIALE
30 CLS:PRINT"QUANTI GIOCATORI "          :REM TRS
30 PRINT"(CLR)QUANTI GIOCATORI "         :REM PET
40 GOSUB 1000:NG=R                       :REM NG = NUMERO GIOCATORI
50 IF NG<6 THEN 80
60 PRINT"SPIACENTE, SONO PROGRAMMATO PER 5 GIOCATORI AL MASSIMO"
70 GOTO 30
80 N = 0                 :REM N= NUMERO DEL GIOCATORE DI TURNO
90 FOR I=0 TO NJ-1
100 PRINT"NOME DEL GIOCATORE N. ";I+1;:INPUT N$(I):NEXT
110 PRINT"VOLETE LE ISTRUZIONI D'USO ?";
120 GOSUB 1000           :REM SUB 1000 PONE LE DOMANDE
130 IF LEFT$(R$,1)="S"THEN GOSUB 2000:REM MODO D' USO
140 CLS                  :REM TRS
140 PRINT"CLR"           :REM PET
150 DEFSTR H:H=CHR$(28)+CHR$(30) 'TRS HOME E CANCELLAZIONE FINE LINEA
155 H3=CHR$(26):H3=H3+H3+H3      :REM TRS 3 VOLTE CURSORE IN BASSO
156 REM SU PET I MOVIMENTI CURSORE SONO ACCESSIBILI DIRETTAMENTE
160 GOSUB 3000           :REM COSTRUZIONE GRAFICA DEI DADI

200 REM          GIOCO DA 1 A 3 TIRI      (PER 1 GIOCATORE)

210 PRINT"(HOME)"TAB(39)         :REM PET (CANCELLA LINEA IN ALTO)
215 PRINT"(HOME)TOCCA GIOCARE A ";N$(N) :REM PET
210 PRINT H;"TOCCA GIOCARE A ";N$(N)    :REM TRS
220 GOSUB 1000           :REM ENTER (TRS) O NUMERO+RETURN (PET)
240 J=0:H1=R             :REM (PET) H1=NUMERO CASUALE
240 J=0                  :REM TRS         J CONTA I 3 TIRI
250 FOR I=0 TO 4:IF NG(I)=0 THEN K=0:GOSUB 4050
255 REM CANCELLA I DADI DA GIOCARE (NG=0 NON CONTROLLATO)
260 NEXT I
270 GOSUB 5200           :REM INTERVALLO
280 FOR I=0 TO 4         :REM TIRA E VISUALIZZA I DADI DA GIOCARE
290 IF NG(I)=0 THEN K=RND(6):N(I)=K:GOSUB 4000  :REM TRS
290 IF NG(I)=0 THEN K=1+RND(H1)*5:N(I)=K:GOSUB 4000 :REM PET
300 NEXT I
310 GOSUB 5200           :REM RITARDO
320 J=J+1:IF J=3 THEN 600        :REM FINE DEI 3 TIRI

330 REM          RIMESSA IN GIOCO E SCELTA DEI DADI
340 PRINT"(HOME)"TAB(39)                 :REM VEDI Z10
345 PRINT"(HOME)G GIOCA - 1/6 CONTROLLA - S NUOVA SCELTA - T CONTROLLA TUTTO"
340 PRINT H;"G GIOCA - 1/6 CONTROLLA - S NUOVA SCELTA - T CONTROLLA TUTTO"
```

```
350 R$=INKEY$:IF R$="" THEN 350 :REM TRS
350 GET R$:IF R$="" THEN 350     :REM PET
360 IF R$="J" THEN 250           :REM LANCIA I DADI DA GIOCARE
370 IF R$="T" THEN 600           :REM FINE DEI LANCI
380 IF R$="S" THEN GOSUB 410     :REM RIMETTE IN GIOCO TUTTI I DADI
390 R=VAL(R$):IF R>=1 AND R<=6 THEN 480 :REM METTE DA PARTE IL DADO
400 GOTO 340

410 REM RIMESSA IN GIOCO DEI DADI MESSI DA PARTE
420 FOR I=0 TO 4
430 IF NG(I)=0 THEN 460          :REM SCARTA I DADI GIA' IN GIOCO
440 NG(I)=0:K=N(I):GOSUB 4050:PRINT B$;D$(0);    :REM PET
445 REM B$+DADO BIANCO D$(0) ANNULLA IL DADO CONTROLLATO
460 NEXT I
470 RETURN

480 FOR I=0 TO 4         :REM METTE IL DADO INDICATO IN ZONA RISERVATA
490 IF NG(I)<>0 THEN 540         :REMIGNORA I DADI GIA' CONTROLLATI
500 IF R<>N(I) THEN 540
510 NG(I)=1              :REM TROVA IL DADO DA CONTROLLARE
520 K=0:GOSUB 4050       :REM CANCELLA IL DADO IN POSIZIONE DI GIOCO
530 PRINT B$;D$(N(I));   :REM LO VISUALIZZA IN POSIZIONE CONTROLLATA
535 GOTO 340
540 NEXT I
550 GOTO 340             :REM DADO NON TROVATO

600 REM          ANALISI DI UNA SERIE DI TRE COLPI
605 GOSUB 5200:GOSUB 410         :REM RITARDO;TOGLIE I DADI
607 GOSUB 5200
610 PRINT"(HOME)"TAB(39)         :REM VEDI 210
615 PRINT"(HOME)0 PASSA - 1/6 F,P,C,Y,MIN,MAX"; :REM PET
610 PRINT H;"0 PASSA - 1/6 F,P,C,Y,MIN,MAX"     :REM TRS
620 GOSUB 1000

630 IF R$="0" THEN 810           :REM PASSA AL GIOCATORE SUCCESSIVO
640 IF R>=1 AND R<=6 THEN 1100   :REM 1/6
650 IF R$="F" THEN 1140          :REM FULL   (ES.: 11444)
660 IF R$="P" THEN 1220          :REM POKER (ES.: 13333)
670 IF R$="S" THEN 1290          :REM SCALA (ES.: 23456)
680 IF R$="Y" THEN 1380          :REM YAM    (ES.: 55555)
690 IF R$="MIN" THEN 1420        :REM MIN    (ES.: 1+3+4+2+6=16)
700 IF R$="MAX" THEN 1470        :REM MAX    (ES.: 4+5+5+6+6=26)
710 GOTO 610                     :REM RISPOSTE NON PREVISTE

750 REM VISUALIZZAZIONE DEL RISULTATO DOPO CONVALIDA
760 K=8+N:GOSUB 5000     :REM TRS  K=17+N PER PET (SU TRS ANCHE PRINT @)
765 PRINT LEFT$(N$(N),8);
770 FOR I=1 TO 12:PRINT STR$(T(N,I));:NEXT
775 REM L' USO DI STR$ PERMETTE DI SERRARE LE CIFRE
776 REM (NECESSARIO PER IL PET,CON 40 CAR/LINEA)(1 SOLO SPAZIO)
780 S=0
790FOR I=1 TO 12:S=S+B(I)*T(N,I):NEXT
795 T(N,0)=S             :REM ULTIMO PUNTEGGIO
800 PRINT STR$(S);
810 N=N+1:IF N< NG THEN 210      :REM GIOCATORE SEGUENTE
820 N=0                          :REM FINE DI UN GIRO
830 REM LA CLASSIFICA SI STABILISCE SOLO A FINE GIRO
840 S=0:FOR I=0 TO NG-1
845 IF S=T(I,0) THEN I2=-1       :REM PIU' DI UN PRIMO EX-AEQUO
850 IF S<T(I,0) THEN S=T(I,0):I2=I       :REM PRIMO TEMPORANEO
860 NEXT
865 K=22:GOSUB 5000      :REM PET (LA ZONA GIOCO OCCUPA PIU' LINEE)
865 K=14:GOSUB 5000      :REM TRS (BASSO DELLA TABELLA)
870 IF I2=-1 THEN PRINT"EX-AEQUO";TAB(40);:GOTO 210
875 PRINT N$(I2);" E' PRIMO";TAB(40);
890 GOTO 210             :REM GIOCATORE SEGUENTE
```

```
999 REM                  ROUTINE D' INGRESSO
1000 INPUT "   ←←←←";R$:R = VAL(R$):RETURN      :REM PET
1010 REM 3 SPAZI, 1 CARATTERE, 3 CURSORE A SINISTRA
1011 REM EVITA L' ARRESTO IN CASO DI RISPOSTA COL SOLO RETURN
1000 INPUT R$:R = VAL(R$):RETURN        :REM TRS
1010 REM QUI IL SOLO RETURN LASCIA IL VALORE PRECEDENTE

1050 REM CONVALIDA RIFIUTATA (COMBINAZIONE NON FATTA, O GIA' FATTA)
1060 PRINT"(HOME) RICHIESTA NON CONFORME ALLA REGOLA DEL GIOCO";TAB(40) :REM PET
1060 PRINT H;"RICHIESTA NON CONFORME ALLA REGOLA DEL GIOCO";: REM TRS
1070 GOSUB 5200:GOSUB 5200       :REM DOPPIO RITARDO
1080 GOTO 610                    :REM RITORNA ALL' ANALISI DEI RISULTATI

1090 REM CONVALIDA DELLE FIGURE
1099 REM DA 1 A 6                ES. 65662
1100 IF T(N,R)>0 THEN 1050       :REM GIA' PRESA
1110 FOR J = 0 TO 4
1120 IF N(I) = R THEN T(N,R)=T(N,R)+1    :REM SOMMA DEI VALORI DEI DADI
1130 NEXT I:GOTO 750             :REM VISUALIZZAZIONE DEI RISULTATI
1139 REM "FULL"                  ES. 35535
1140 IF T(N,7)>0 THEN 1050       :REM GIA' PRESA
1150 F1 = N(0):F2 = F1
1160 FOR J = 1 TO 4:IF N(J) <> F1 THEN F2=N(I)
1165 NEXT I
1170 FOR I = 1 TO 4:IF N(I) <> F1 AND N(I) <> F2 THEN 1050
1180 NEXT I      :REM A QUESTO PUNTO CI SONO SOLO 2 VALORI (FULL O POKER)
1190  J=0:FOR I=1 TO 4:IF F1=N(J) THEN J=J+1
1195 NEXT I      :REM J DEVE VALERE 1 O 2
    (OPPURE 4 PER UN POKERISSIMO PRESO DA UN FULL)
1200 IF J=0 OR J=3 THEN 1050     :REM RIFIUTO DEI POKER
1210 T(N,7)=1:GOTO 750

1219 REM CONVALIDA DEL POKER             ES. 4 2 4 4 4
1220 IF T(N,8)>0 THEN 1050       :REM GIA' PRESO
1230 J=0:FOR I=1 TO 4:IF N(I)=N(0) THEN J=J+1
1240 NEXT
1250 IF J>2 THEN 1280            :REM 1' DADO UGUALE A 3 O 4 ALTRI
1260 FOR I=2 TO 4:IF N(I) <> N(1) THEN 1050
1270 NEXT
1280 T(N,8)=1:GOTO 750
1285 REM PER UNA NOTAZIONE CHE TENGA CONTO DEL DADO NELLE COMBINAZIONI
1286 REM SI METTEREBBE QUESTO VALORE IN T(N,...) INVECE DI 1
1289 REM CONVALIDA DELLA SCALA           ES. 3 5 2 4 6
1290 IF T(N,9) > 0 THEN 1050
1300 FOR I=1 TO 4
1310 IF N(I-1) > N(I) THEN J=N(I):N(I)=N(I-1):N(I-1)=J:GOTO 1300
1320 NEXT I              :REM I DADI SONO ORA SCELTI
1330 J=0: FOR I=1 TO 4
1340 IF N(I)-N(I-1) <> 1 THEN J=J+1
1350 NEXT
1360 IF J>0 THEN 1050    :REM SE J>1 CONVALIDEREBBE UNA SCALA DI 4
1370 T(N,9)=1:GOTO 750
1379 REM POKERISSIMO             ES. 6 6 6 6 6
1380 IF T(N,10)>0 THEN 1050
1390 FOR I=1 TO 4:IF N(0) <> N(I) THEN 1050
1400 NEXT
1410 T(N,10)=1:GOTO 750          :REM VEDI LINEA 1285
1420 IF T(N,11) > 0 THEN 1050
1430 J=0                 :REM CALCOLO DELLA SOMMA DEI DADI
1440 FOR I=0 TO 4:J=J+N(I):NEXT
1450 IF T(N,12) > 0 AND T(N,12) < J THEN 1050:REM SE C' E' GIA' UN MASSIMO
1460 T(N,11)=J:GOTO 750                  :REM BISOGNA CHE MIN<MAX
```

```
1469 REM CONVALIDA DEL MAX
1470 IF T(N,12) > 0 THEN 1050
1480 J=0
1490 FOR I=0 TO 4:J=J+N(I):NEXT I
1500 IF T(N,11) > 0 AND T(N,11) > J THEN 1050
1510 T(N,12)=J: GOTO 750

2000 REM         MODO D' USO
2010 REM
2020 REM QUESTO E' SOLO UN ESEMPIO. UN BUON MODO D' USO SI OTTIENE
2030 REM EMPIRICAMENTE IN FUNZIONE DEL PUBBLICO E DEVE ESSERE BREVE
2040 CLS                    :REM TRS
2040 PRINT "(CLR)"          :REM PET
2050 PRINT"IL GIOCO DEL YAM SI GIOCA CON 5 DADI"  :REM IMPAGINATO
2060 PRINT:PRINT"SI FANNO DA UNO A TRE TIRI CONTROLLANDO":REM A 40 CAR/LINEA
2070 PRINT:PRINT"SE TENERE DEI DADI DEI TIRI PRECEDENTI."
2080 PRINT PRINT"LO SCOPO E' FORMARE DELLE COMBINAZIONI,"
2090 PRINT"GRUPPI DI 1, DI 2,..., DI 6"
2100 PRINT"FULL (COPPIA+TRIS), POKER (QUADRIS),"
2110 PRINT"SCALA (12345,23456), YAM (5 DADI UGUALI."':REM YAM=POKERISSIMO
2120 PRINT"SI HA DIRITTO ANCHE A COMBINAZIONI CHE DANNO"
2130 PRINT"LA SOMMA DEI DADI, MA UNA, MIN,"
2140 PRINT"DEVE  ESSERE INFERIORE ALL' ALTRA, MAX"
2150 PRINT
2160 PRINT"UNA COMBINAZIONE DEVE ESSERE PRESA SOLO UNA VOLTA"
2170 PRINT:PRINT"BATTERE UN TASTO PER LA PAGINA 2"
2180 R$=INKEY$:IF R$="" THEN 2180        :REM TRS
2180 GET R$:IF R$="" THEN 2180           :REM PET
2190 CLS                                 :REM TRS
2190 PRINT"(CLR)"                        :REM PET
2200 PRINT"PER GIOCARE PREMERE RETURN"   :REM PET, ENTER PER TRS
2210 PRINT
2220 PRINT"ESAMINARE I DADI"
2230 PRINT"SCEGLIERE QUELLI CHE SERVONO PER UNA FIGURA"
2240 PRINT"BATTENDO DA 1 A 6.PER SCEGLIERE DI NUOVO BATTERE R"
2250 PRINT"POI J PER GIOCARE"
2260 PRINT"O T SE SIETE SERVITO"
2270 PRINT
2280 PRINT"INFINE SCEGLIERE UNA COMBINAZIONE BATTENDO"
2290 PRINT"1,2...6,F,C,S,Y,MIN,MAX POI ENTER":REM TRS
2300 PRINT:PRINT:PRINT"BATTERE UN TASTO PER COMINCIARE IL GIOCO"
2310 GET R$:IF R$=""THEN 2310    :REM PET
2310 R$=INKEY$:IF R$="" THEN 2310        :REM TRS
2320 RETURN
2999 REM         CREAZIONE DEI DADI (PET)
3000 B$="←←←←←↓ "        :REM 5 CURSORE A SINISTRA 1 IN BASSO
3005 DIM D$(6)           :REM 6 DADI; D$(0) SERVE A CANCELLARE
3010 H$="▛▀▀▀▀▜"+B$      :REM SOPRA DEL DADO
3020 REM CON B$ OGNI FACCIA DI DADO POSIZIONA IL CURSORE
3030 REM PER LA ZONA SUCCESSIVA
3040 A0$="▌(RVS)     (OFF) ▐ "       :REM FASCIA SENZA PUNTI
3050 A1$="▌(RVS) *   (OFF) ▐ "       :REM " " = SPAZIO
3060 A2$="▌(RVS)*    (OFF) ▐ "
3070 A3$="▌(RVS)*** (OFF) ▐ "
3080 A4$="▌(RVS)  * (OFF) ▐ "
3090 A5$="▌(RVS)* * (OFF) ▐ ":BS$="▙(RVS)▄▄▄(OFF)▟ "
3100 A6$="      "+B$                 :REM B$=BASSO DEL DADO
3110 D$(0)=A6$+A6$+A6$               :REM SERVE A CANCELLARE UN DADO
3120 D$(1)=H$+A0$+A1$+A0$+BS$
3130 D$(2)=H$+A2$+A0$+A4$+BS$
3140 D$(3)=H$+A2$+A1$+A4$+BS$
3150 D$(4)=H$+A5$+A0$+A5$+BS$
3160 D$(5)=H$+A5$+A1$+A5$+BS$
3170 D$(6)=H$+A3$+A0$+A3$+BS$
3199 REM MESSA IN OPERA DELLA TABELLA DEI RISULTATI
3200 K=14:GOSUB 5000             :REM PORTA IL CURSORE IN LINEA 14
```

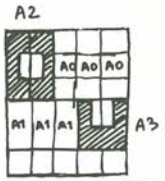
```
3210 FOR J=0 TO 39:PRINT"-";:NEXT I
3220 PRINT TAB(9)"1 2 3 4 5 6 F C S Y MIN MAX TOT":REM 1 SPAZIO
3230 FOR I=0 TO 39:PRINT"-";:NEXT I
3260 RETURN


2999 REM         CREAZIONE DEI DADI (TRS)
3000 DEFSTR A,D:DIM D(6)
3010 B$=STRING$(5,24)+CHR$(26)  :REM 5 CURSORE A SINISTRA 1 IN BASSO
3020 REM CON B$ UNA META' DEL DADO POSIZIONA IL CURSORE PER LA SEGUENTE
3030 A0=CHR$(191)
3040 A1=CHR$(143)
3050 A2=CHR$(183)+CHR$(187)
3060 A3=CHR$(141)+CHR$(142)
3070 A4=CHR$(179)
3080 A5=CHR$(140)
3090 A6=A0+A0
3100 A7=A1+A1+A1
3110 D(1)=A6+A1+A6+B$+A7+A1+A1
3120 D(2)=A2+A0+A6+B$+A7+A3
3130 D(3)=A2+A1+A6+B$+A7+A3
3140 D(4)=A2+A0+A2+B$+A3+A1+A3
3150 D(5)=A2+A1+A2+B$+A3+A1+A3
3160 D(6)=A2+A4+A2+B$+A3+A5+A3
3170 A(8)="      ":D(0)=A8+B$+A8
3200 REM TABELLA DEI RISULTATI
3210 K=6:GOSUB 5000      :REM PROGRAMMA COMPATIBILE PET- SAREBBE PIU'
                         :REM SEMPLICE <<PRINT @>>
3220 PRINT STRING$(63,140)
3230 PRINT TAB(9)"1 2 3 4 5 6 F C S Y MIN MAX TOTAL"
3240 PRINT STRING$(63,140)       :REM GRANDE TRATTO MARCATO
3250 RETURN


3998 REM                  VISUALIZZAZIONE DEL DADO K-ESIMO
3999 REM GOSUB 4050 PRODUCE UNA VISUALIZZAZIONE IMMEDIATA
4000 REM GOSUB 4000 DA' UN EFFETTO CHE SUGGERISCE LA ROTAZIONE DEL DADO
4010 FOR I1=0 TO 2+5*RND(0)      :REM PET DA 2 A 7 ITERAZIONI
4010 FOR I1=0 TO 2+RND(5)        :REM TRS
4020 PRINT H;H3;:GOSUB 5100:PRINT D(RND(6));     :REM TRS
4020 PRINT"(HOME)    ";:GOSUB 5100:PRINT D$(1+R*RND(0));:REM PET
4030 NEXT I1
4050 PRINT"(HOME)    ";:GOSUB 5100:PRINT D$(K)    :REM PET
4050 PRINT H;H3;:GOSUB 5100:PRINT D(K);           :REM TRS
4060 RETURN
4070 REM ATTENZIONE AGLI INDICI IN CASO DI GOSUB FREQUENTI

                GESTIONE DEL CURSORE
5000 PRINT"(HOME)";:FOR I1=0 TO K:PRINT" ";:NEXT I1:RETURN:REM PET
5000 PRINT H;:FOR I1=0 TO K:PRINT CHR$(26);:NEXT I1:RETURN:REM TRS
5100 FOR S=1 TO 8*I:PRINT" ";:NEXT S:RETURN      :REM PET
5100 FOR S=1 TO 9*I:PRINT CHR$(25);:NEXT S:RETURN  :REM TRS
5110 REM 2 ROUTINE SUPERFLUE CON PRINT@ (TRS)

                RITARDO
5200 FOR I1=0 TO 400:NEXT I1:RETURN
5210 REM I RITARDI SONO MOLTO IMPORTANTI
5215 REM SI POTRANNO ABBREVIARE UNA VOLTA ABITUATI AL GIOCO
```



exemple : D(2)

```
1 REM ********************************
2 REM * GENERATORE DI CARATTERI GRANDI *
3 REM ********************************
 QUESTO PROGRAMMA PERMETTE, PARTENDO DA UNA PAROLA O DA UNA FRASE, DI STAMPARE
 UN GROSSO TITOLO PER PRESENTARE UN LISTATO O UN TABULATO, O DI FARE
 SCORRERE SULLO SCHERMO UN TESTO VISIBILE DA LONTANO (SLOGAN, MESSAGGI..)

 FUNZIONA COME IL GENERATORE DI CARATTERI DI UN SCHERMO MONITOR
 DI CUI FACILITERA' LA COMPRENSIONE. SCRITTO PER TRS,PET,APPLE

2 CLEAR 200      :REM TRS (ALLOCAZIONE STANDARD DI 150 E' INSUFFICIENTE)
5 POKE 16553,255          :REM TRS EVITA NOIE CON READ/DATA
10 DIM Z$(94),G$(1259),T$(50),T(50)      :REM APPLE
10 DIM Z$(18),G$(35,6),T(50)             :REM TRS,PET
15 REM SI E' LIMITATO IL TESTO A 50 CARATTERI; 94=5*19-1; 1259=36*7*5-1
17 REM Z$ CONTERRA' DELLE SEZIONI ORIZZONTALI DI CARATTERI
20 DATA " *** "," *   *"," ***** "," **** "," *    "
30 DATA "*  **"," *   *"," *  *"," *    *"," *  *"
40 DATA "* * "," **   *"," **"," * * *"," **  *"
50 DATA " **** "," * * "," *  *"," ** **"
55 REM G$ CONTERRA' PER OGNI CARATTERE GLI INDICI CHE RIMANDANO
56 REM ALLE SEZIONI DI Z$. 18 SEZIONI BASTANO PER A-Z,0-9
60 DATA 1,2,2,3,2,2,2, 4,2,2,4,2,2,4, 1,2,5,5,5,2,1      :REM A B C
70 DATA 4,2,2,2,2,2,4, 3,5,5,4,5,5,3, 3,5,5,4,5,5,5      :REM D E F
80 DATA 1,2,5,5,6,2,1, 2,2,2,3,2,2,2, 1,8,8,8,8,8,1      :REM G H I
90 DATA 9,9,9,9,9,2,1, 10,11,12,12,11,10,2, 5,5,5,5,5,5,3 :REM J K L
100 DATA 2,19,14,2,2,2,2, 2,15,14,6,2,2,2, 1,2,2,2,2,2,1 :REM M N O
110 DATA 4,2,2,4,5,5,5, 1,2,2,2,2,6,16, 4,2,2,4,11,10,2  :REM P Q R
120 DATA 1,5,5,1,9,9,1, 3,8,8,8,8,8,8, 2,2,2,2,2,2,1     :REM S T U
130 DATA 2,2,2,2,2,17,8, 2,2,2,2,14,14,17, 2,2,17,8,17,2,2 :REM V W X
140 DATA 2,2,17,8,8,8,8, 3,9,18,8,7,5,3, 1,2,6,14,15,2,1 :REM Y Z 0
150 DATA 7,12,12,12,12,12,12, 1,2,9,18,8,7,3, 1,9,9,1,9,9,:REM 1 2 3
160 DATA 9,18,8,7,10,3,18, 3,5,4,9,9,2,1, 18,8,1,2,2,2,1 :REM 4 5 6
170 DATA 3,9,18,8,7,7,7, 1,2,2,1,2,2,1, 1,2,2,2,1,8,7    :REM 7 8 9

180 FOR I=0 TO 18:READ Z$(5*I,5*I+4):NEXT I      :REM APPLE
180 FOR I=0 TO 18:READ Z$(I):NEXT I              :REM PET TRS
190 FOR J=0 TO 35          :REM 26 LETTERE + 10 CIFRE
200 FOR J=0 TO 6           :REM 7 SEZIONI PER CARATTERE
210 READ K:K=K-1
220 L=35*I+5*J:G$(L,L+4)=Z$(5*K)          :REM APPLE
220 G$(I,J)=Z$(K)                  :REM TRS PET
230 NEXT J
240 NEXT I
250 INPUT"TESTO ";T$      :REM TESTO DA VISUALIZZARE
260 L=LEN(T$):FOR J=0 TO L-1      :REM CONVERSIONE IN CODICE"A..Z..9"-> 0..35
270 K=ASC(T$(I))          :REM APPLE
270 K=ASC(MID$(T$,I+1,1))         :REM TRS PET
280 T(I)=-1:IF K>64 AND K<91 THEN T(I)=K-65
285 REM APPLE (ASCII + 128) 64=192; 91=219; 65=193
290 IF K>47 AND K<58 THEN T(I)=K-22
300 NEXT I
310 CLS:(PRINT CHR$(23))          :REM TRS (DOPPIA LARGHEZZA IN OPZIONE)
310 PRINT"(CLR)"                  :REM PET
310 CALL -936                     :REM APPLE INTEGER
310 HOME                          :REM APPLESOFT
320 K=0:L1=6                      :REM 6(APPLE PET);5 O 10 TRS;13 SE 80 COLONNE
330 IF L-K<L1 THEN L1=L-K         :REM REGOLA LUNGHEZZA ULTIMA LINEA
340 FOR J=0 TO 6                  :REM SCANSIONE PER 7 LINEE
350 FOR I=0 TO L1-1               :REM SCANSIONE PER CARATTERI ORIZZ.
360 I1=T(I+K):IF I1=-1 THEN PRINT"     ":GOTO 380
370 I1=35*I1+5*J:PRINT G$(I1,I1+4);" "; :REM APPLE
370 PRINT G$(I1,J);" ";           :REM PET TRS
380 NEXT I
390 PRINT:NEXT J
400 PRINT:K=K+L1:IF K>=L THEN 320         :REM RICICLA IL MESSAGGIO
410 GOTO 330     :REM SEQUENZA DEL TESTO
395 FOR I=0 TO 500:NEXT I         :REM RITARDO OPZIONALE SU STAMPANTE
420 REM PER UNA MIGLIORE COMPRENSIONE INSERIRE DEGLI SPAZI
430 REM PER NON TAGLIARE LE PAROLE
```

	ABS	28	Funzione che restituisce il valore assoluto di una espressione numerica V = ABS (X − Y).
	AND	41	Operatore logico "AND" - Si applica a delle espressioni logiche ed a degli interi IF A < B AND B < A THEN PRINT "IMPOSSIBLE".
	ASC	37	Funzione che restituisce il codice ASCII (da 0 a 255) di un carattere K = ASC ("*") : J = ASC (A$).
(T.R.S. liv. 1)	AT		Vedere @ (fine di lista).
	ATN	31	Funzione arco-tangente (Radianti Pπ = 4 * ATN (1).
	AUTO	86	Comanda la numerazione automatica delle linee in un programma in corso di scrittura AUTO - AUTO 100,10
(Apple)	CALL	69	Chiama un sottoprogramma in linguaggio macchina (cf. SYS) CALL-936 attiva la cancellazione dello schermo.
	CATALOG		Sistemi a floppy disk (lista il nome dei files contenuti in un dischetto).
(T.R.S.)	CDBL	31	Restituisce un numero al formato doppia precisione uguale all'argomento.
	CHR$	37	Restituisce un carattere dato dal suo codice (0 a 225), in modo particolare tutti i caratteri non disponibili alla tastiera PRINT CHR$ (23)

(T.R.S.)	CINT	29	Restituisce un numero al formato intero (intero per difetto).
(T.R.S.)	CLEAR	23	Libera le zone di memoria accordate alle stringhe variabili. Es: CLEAR 300 riserva in più 300 byte
(Applesoft)	CLEAR		Inizializza le variabili a zero
(T.R.S.)	CLOAD		Carica programma da cassette (cf. manuale specifico)
	CLOSE		Sistemi a floppy disk. Chiusura logica di un file.
(Apple)	CLR	23	Cf. CLEAR.
(T.R.S.)	CLS	50	Cancella lo schermo.
(P.E.T.)	CMD	77	Comando del bus IEEE 488 (cf. manuale specifico).
(Apple)	COLOR	72	Determina il colore del prossimo carattere o punto visualizzato. COLOR = 0 (nero)....COLOR = 15 (bianco).
	CONT	87	Riprende l'esecuzione arrestata con l'istruzione STOP o con il tasto BREAK.
	COS	31	Funzione coseno (argomento in radianti).
(T.R.S.)	CSNG	31	Restituisce un numero al formato reale in semplice precisione.
	DATA	24	Istruzione che permette di riunire tutti i dati costanti di un programma DATA SI, NO, FORSE, 3.14.16 (virgola qualche volta obbligatoria).

(P.E.T.) (Applesoft)	DEF.FNX	26	Definizione di una funzione nel programma DEF FNA (X) = A + B.
(T.R.S.)	DEF DBL	14	Definizione di gruppi di variabili in doppia precisione, in interi in semplice precisione.
(T.R.S.)	DEF INT	15	In stringa di caratteri.
(T.R.S.)	DEF SNG	14	
(T.R.S.)	DEF STR	14	DEF INT I-K : DEFSTR Z.
	DELETE		Sopprime un gruppo di linee di un programma: DELETE 20 DELETE 100-150.
	DIM	23	Definisce una matrice (e delle stringhe in Apple) DIM T (10), V (5,6) DIM A$ (10).
(Apple)	DSP	87	Istruzione di messa a punto, provoca la visualizzazione delle variabili specificate ogni volta che le stesse vengono modificate DSP T1, T2.
	EDIT		Comando speciale che permette di correggere delle linee di programma senza ricomporle (vedere manuali specifici).
	ELSE	41	Ausiliare di IF, definisce un ramo complementare "ALTRIMENTI" "SE NON" IF A = B POI PRINT "UGUALE" ELSE PRINT "DIFF"
	END	88	Termina l'esecuzione di un programma.
(T.R.S.)	ERL	82	Funzione che restituisce il numero di linea corrispondente all'ultimo errore.

(T.R.S.)	ERR	82	Funzione che restituisce il codice errore (infatti 2 * (codice - 1), codice = ERR/2 + 1.
(T.R.S.)	ERROR		Simula la comparsa di un errore es. ERROR (14) (errore di tipo 14).
	EXP	31	Funzione matematica esponenziale E = EXP (1) A = EXP (B + C).
	FIX	31	Restituisce la parte intera.
(Applesoft)	FLASH		Realizza un inversione continua dello schermo.
	FOR	45	Attiva, per un numero di volte specificate, la ripetizione di un gruppo di istruzioni. FOR I = 1 TO visualizza N SPEP 2
(P.E.T.) (Applesoft)	FRE		Funzione che visualizza la quantità di memoria disponibile PRINT FRE (0) (vedere MEM) # FRE TRS visualizza la memoria disponibile per delle strighe.
(P.E.T) (Applesoft)	GET	59	Acquisisce l'ultimo carattere impostato da tastiera (come INKEY$) GET A$.
	GOSUB	49	Chiama un sottoprogramma (subroutine) alla linea indicata GOSUB 1010.
	GOTO	41	Rompe la sequenza e rinvia verso la prossima istruzione da eseguire GOTO 1000.
(Apple)	GR	73	Passaggio al sistema grafico (40 x 40 punti) nessun operatore.
(Apple)	HCOLOR		Sceglie un colore in alta risoluzione HCOLOR = C (da 0 a 7) alta risoluzione.

(Apple)	HGR		Passa in sistema grafico alta risoluzione.
	HGR2		Con o senza 4 linee di testo.
(Apple)	HLIN		Chiamata di una routine che visualizza dei tratti orizzontali, HLIN 0,39 AT20 (linea di ordinata 20 e ascisse da 0 a 39).
(Applesoft)	HOME		Cancella lo schermo.
(Apple)	HTAB		Fa una tabulazione orizzontale HTAB X.
	IF	41	Test ed esecuzione condizionale di una azione (sovente un salto) IF N = 100 THEN GOTO 36 (-1 = vero, 0 = falso).
(T.R.S.)	INKEY$	59	Prende l'ultimo carattere impostato 10 Z$ = "INKEY$": IF Z = " " THEN GOTO 10.
(T.R.S.)	INP	66	Funzione che restituisce sotto forma decimale (da 0 a 255) il valore presente alla porta d'indirizzo indicato I = INP (A).
	INPUT	57	Interrompe il programma e provoca l'attesa delle variabili specifiche.
	INT	29	Funzione che restituisce il valore intero per difetto di un'espressione numerica. A = INT (X + 0,5).
(Applesoft)	INVERSE		Passaggio in video inverso.
	LEFT$	35	Funzione che estrae la parte sinistra di una stringa di caratteri. A$ = LEFT$ (Z$,L) (In Apple Integer: A$ = Z$ (0,L - 1).

	LEN	35	Funzione che restituisce la lunghezza di una stringa. FOR 1 = 1 TO LEN (Z$): PRINT" =" : NEXT
	LET	19	In generale, forma completa dell' assegnazione LET A = B + C
	LIST		Comanda l'elenco di tutto o parte del programma in memoria. LIST 20 LIST50 - 100 (Apple: LIST 50, 100) LIST 50 -
	LOAD		Carico dei dati (vedere manuali specifici).
	LOG	31	Funzione logaritmo. L'argomento deve essere positivo V = LOG (A) + LOG (B).
(Apple)	MAN	86	Numerazione manuale delle linee di programma (per annullare AUTO).
(T.R.S.)	MEM		Funzione senza argomento che fornisce la quantità di memoria disponibile (vedere FRE) IF MEM 100 THEN PRINT "MEMORY OWERFLOW". Nessun argomento.
	MID$	35	Funzione che estrae da una stringa alla posizione indicata, una sottostringa di lunghezza data: Z1$ = MID$ (Z,$,P,L) Apple Integer: Z1$ =Z$ (P,P + L - 1).
(Apple)	MOD	33	Funzione che dà direttamente il resto di una divisione in numeri interi R=A MOD B (equivalenti: R=A-B* (A/B) con A,B interi.
	NEW		Comando che cancella il programma in memoria (comprese le variabili).

	NEXT	45	Complemento necessario a FOR per delimitare la sequenza da ripetere (loop) NEXT NEXT I NEXT I,J.
Apple	NODSP	87	Annulla l'ultimo DSP attivo.
Applesoft	NORMAL		Fine di inversione video.
	NOT	40	Operatore logico che dà il contrario dell'espressione logica che segue NOT IF NOT (A>B) THEN PRINT "A<=B" (NOT 0 dà - 1) (NOT N dà - (N+1))
	NOTRACE	87	Annulla l'azione dell'istruzione TRACE precedentemente eseguita.
(T.R.S.)	ON ERROR GOTO	82	Gli errori che sopraggiungono saranno trattati dal programma.
Applesoft	ON ERR GOTO		L'oggetto dell'azione standard (messaggi e interruzione)
	ON GOTO	41	Realizza uno scambio multiplo (dirottamento all'indirizzo K) ON K GOTO 100,120,36,60 K deve valere 1,2 ... (altrimenti ignorato).
	ON GOSUB	49	ON K GOSUB 100,36 ... stessa funzione per i sottoprogrammi.
	OPEN		Apertura di file (vedere manuali specifici).
	OR	41	Operatore logico dove IF A>B OR C<D THEN 10 - 1 (OR N dà -1).
(T.R.S.)	OUT	66	Istruzione che presenta sulla porta di indirizzo dato (da 0 a 225) un byte di valore dato (da 0 a 255) OUT P,O.
(Apple)	PDL		Funzione che restituisce la posizione delle paddle di gioco X= PDL (M) (M: da 0 a 3)

(T.R.S.)	RESET	73	Spegne il punto di coordinate indicate (precedentemente acceso per mezzo di SET RESET (X,Y) (X da 0 a 127, Y da 0 a 47).
	RESTORE		Reinizializza i file DATA (il READ seguente leggerà all'inizio del primo DATA).
(T.R.S.)	RESUME	82	Ritorno del sottoprogramma d'errore attivato con ON ERROR GOTO RESUME (alla linea errata) RESUME NEXT (alla linea seguente) RESUME 150 (alla linea 150).
	RETURN	49	Istruzione che consente l'uscita da un sottoprogramma, ritorno al programma dopo il GOSUB di chiamata.
	RIGHT$	35	Funzione che estrae da una stringa di caratteri la parte destra di lunghezza data Z$ = RIGHT$ (Z$,L).
	RND	63	Funzione che restituisce un numero casuale (da 0 a 1 o da 1 a N ecc...) H = RND (N).
	RUN	89	Comando che attiva l'esecuzione di un programma con inizializzazione a zero delle variabili. RUN-RUN 100.
	SAVE		Immagazzina un file su una periferica (memoria di massa).
(Apple)	SCRN		Funzione che restituisce il colore dal punto di coordinate indicate. C=SCRN (X,Y) (X,Y da 0 a 39).
(T.R.S.)	SET	73	Accende il punto dello schermo di coordinate indicate. SET (X,Y) (X da 0 a 127, Y da 0 a 47).
	SGN	29	Funzione che restituisce -1,0 oppure 1 secondo il segno dell'argomento, S= SGN (X).

	PEEK	65	Funzione speciale, ritorna un numero da 0 a 255 che rappresenta il byte all'inizio indicato I=PEEK (A).
(Apple)	PLOT	73	In sistema grafico, visualizza un punto alle coordinate indicate e del colore fissato dall'ultima istruzione COLOR. PLOT X,Y (da 0 a 39).
(T.R.S.)	POINT		Funzione che restituisce -1 (vero) se il punto (x,y) dello schermo è acceso, 0 in caso contrario (falso). IF POINT (X,Y) THEN GOTO .... (X da 0 a 127, Y da 0 a 47).
	POKE	65	Istruzione speciale, scrive all'indirizzo indicato il byte corrispondente a un numero dato (da 0 a 255). POKE A,I (A da 0 a 65535).
Applesoft	POP		In caso di sottoprogramma bloccato, il successivo RETURN risalirà di una linea (dietro al penultimo GOSUB).
	POS		Funzione che restituisce l'indirizzo del cursore sulla linea in corso (da 0 a 63 T.R.S.) (da 0 a 39 P.E.T. e Apple soft) I=POS (0) (argomento "fasullo" ma necessario).
	PRINT	51	Istruzione che visualizza dei dati sullo schermo (attiva anche stampante) PRINT "PIPPO", A,B.
(T.R.S.)	RANDOM	63	Realizza una routine che genera dei numeri a caso.
	READ		Legge i dati raccolti nei Data e li destina alle variabili indicate READ A,B,T, (3).
	REM	38-80	Riserva la linea che contiene REM a delle osservazioni o impaginazioni del programma, non sono eseguite dal programma. REM FINE DEL TRATTAMENTO.

	SGN	29	Funzione che restituisce -1,0 oppure 1 secondo il segno dell'argomento S= SGN (X).
	SIN	31	Funzione trigonometrica. Seno (argomento in radianti, S=SIN(X)).
(P.E.T.)	SPC		SPC (N) provoca la visualizzazione di N spazi.
(Applesoft)	SPEED		Permette di rallentare la velocità di visualizzazione allo schermo. SPEED=V (da 0 a 255).
	SQR	33	Funzione radice quadrata. L'argomento deve essere positivo. C=SQR (A*A+B*B).
	STEP	45	Ausiliare nel loop FOR, precisa il passo del contatore. FOR X=1 TO 3.14 STEP 0.1.
	STOP	89	Provoca l'arresto del programma.
(T.R.S.)	STRING$	37	Funzione che restituisce una stringa ottenuta per ripetizione di un carattere PRINT STRING$ (60, "*")
	STR$	37	Funzione che restituisce una stringa di caratteri a partire da un'espressione numerica. Z$- STR$ (D/T)+ "KM/H".
(P.E.T.)	SYS	69	Chiamata di un sottoprogramma in linguaggio macchina dall'indirizzo indicato SYS(A) (vedere CALL).
(T.R.S.)	SYSTEM	69	Attivazione con chiamata di un programma in linguaggio macchina, rispondere a *? con il nome (caricamento) oppure con l'indirizzo d'entrata.

	TAB	53	Ausiliare di PRINT, inserisce degli spazi fino alla colonna indicata (da 0 a 255 per T.R.S.) PRINT TAB (10) A; TAB (N+1)B.
	TAN	31	Funzione trigonometrica tangente T=TAN (X).
(Apple)	TEXT	73	Ritorna al sistema di visualizzazione alfanumerico (25 linee da 40 caratteri) dopo l'utilizzo del sistema grafico.
	THEN	41	Completamento di IF. Può a volte essere omesso. IF A > B THEN GOTO 10.
(P.E.T.)	TI		Contatore di tempo in 1/60esimo di secondo.
(P.E.T.)	TI$		Contatore di tempo hh, mm, ss
PRINT	TIME$		Regolazione dell'ora con semplice visualizzazione TI$ = "123500".
	TO		Vedere GOTO e FOR.
	TRACE	87	Comando di messa a punto: provoca la visualizzazione dei numeri di linea delle istruzioni operanti.
(T.R.S.)	TRON TROFF	87	Attiva e disattiva il tracciato delle istruzioni operanti.
(T.R.S.)	USING	53	Ausiliare di PRINT. Specifica una stringa che serve da modello di edizione PRINT USING" ###z##";
A-.	USR	69	Chiamata di un sottoprogramma in linguaggio macchina del quale si è messo l'indirizzo nella locazione 0 e 1 (P.E.T. oppure 16526 e 16527 (TRS) istruzione di salto agli indirizzi 10-11-12. A=USR (0).

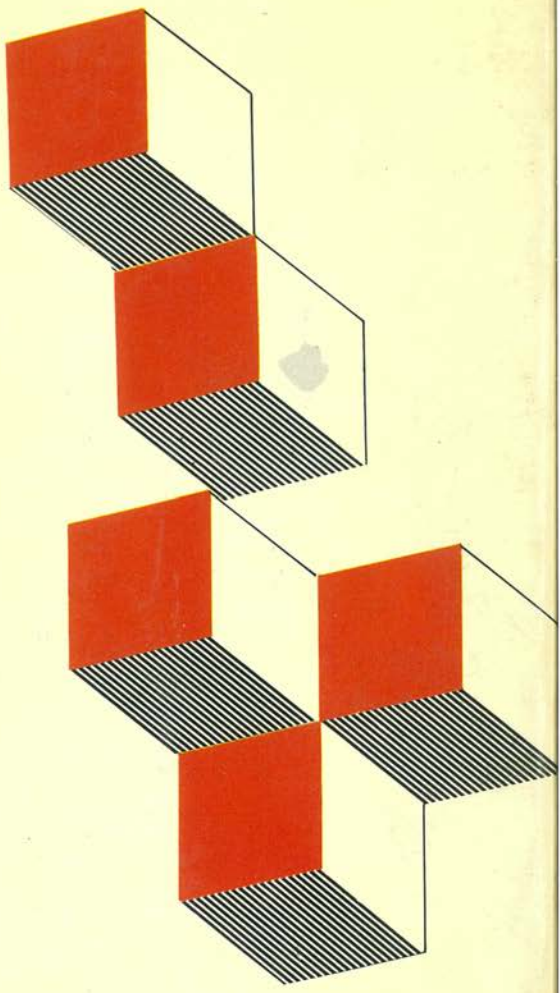
	VAL	38	Funzione che restituisce un valore numerico a partire dai caratteri A=VAL ("31").
(T.R.S.)	VARPTR	14	Restituisce l'indirizzo memoria del dato corrispondente ad una variabile (indirizzo della locazione per una stringa) A=VARPTR (N$).
(P.E.T.)	VERIFY		Comando che verifica la buona esecuzione di un Save (vedere manuale specifico).
(Apple)	VLIN		Chiamata di una routine che visualizza dei tratti verticali VLIN 0,39 AT 20 (linea di ascisse 20 e di ordinata da 0 a 39).
(Apple)	VTAB		Tabulazione verticale.
(P.E.T.) Applesoft	WAIT		Sospende l'esecuzione fino a raggiungimento dell'indirizzo dato (configurazione binaria indicata). WAIT A,B (B da 0 a 255 (PET) WAIT A,B,C (Applesoft) A=addresse, B=AND, C=XOR
(T.R.S.)	@	51	Con PRINT indica una posizione dello schermo. PRINT N "TEX".
	?		Abbreviazione di PRINT.

INDICE ANALITICO

Come tutte "le lingue viventi", il BASIC viene applicato in realtà a questa o a quella macchina sotto forma di dialetti più o meno particolari. Questo libro si sforza di descrivere in modo metodico il BASIC delle tre macchine più diffuse sul mercato mondiale: Apple, PET, TRS 80, e, naturalmente, i loro derivati.

Questo approccio comparato permette di approfondire la conoscenza del linguaggio e dà un'idea del grado di standardizzazione del BASIC. Ciò faciliterà anche la conversione da un determinato personal computer agli altri di programmi esistenti.

Numerosi esempi e programmi fatti "girare" chiariscono i concetti proposti.



L. 8.000

Cod. 513-A ISBN 88-7056-113-5