LA PRIMA RIVISTA EUROPEA DI SOFTWARE PER PERSONAL COMPUTER

PERSONAL SOFTWARE

ANNO 2 N. 5
MARZO-APRILE 1983 L. 3.500

UNA PUBBLICAZIONE DEL GRUPPO EDITORIALE JACKSON

Spedizione in abb. postale Gruppo III/70

- COMPORRE MUSICA CON L'ATARI
- I SEGRETI DEI PERSONAL: ZX80/ZX81–PET/CBM–SPECTRUM–VIC 20
- MINICALC PER ZX81
- IL NUOVO BASIC STANDARD
- TECNICHE DI RICERCA
- PRETTY PRINTER PER L'APPLE II

ORA C'E'!

ZX Spectrum

- 16 o 48 kbytes RAM.
- grafica ad alta risoluzione (256x192 punti).
- 8 colori da utilizzare con la più assoluta libertà per testo, sfondo, bordo, in campo diretto o inverso, con due gradi di luminosità, a luce fissa o lampeggiante.
- Tastiera multifunzione con maiuscole, minuscole, simboli grafici, caratteri definibili dall'utente.
- BASIC Sinclair esteso con funzioni a un tasto per programmare in fretta e senza errori.
- Funzioni specifiche per la grafica e per la gestione di dati d'archivio.
- Ampia disponibilità di programmi preregistrati su compact-cassette: giochi, passatempi, educazionali, matematici, gestionali.
- Totale compatibilità con la stampante ZX.
- Disponibilità immediata del volume ALLA SCOPERTA DELLO ZX SPECTRUM in italiano.
- Prezzo eccezionale: 360.000 lire nella versione a 16 kbytes.

sinclair
è distribuito da

REBIT COMPUTER
A DIVISION OF G.B.C.

REBIT COMPUTER
Via Induno, 18
20092 CINISELLO BALSAMO
Casella Postale 10488 MI

ANNO 2 N. 5 MARZO-APRILE 1983

PERSONAL SOFTWARE

ARTICOLI

RUBRICHE

GUIDA

In questa guida sono riportati i personal computer e i microprocessori di cui si parla negli articoli e nelle rubriche.

Indirizzate tutta la corrispondenza editoriale a

Personal Software, Via Rosellini 12, 20124 Milano

I manoscritti non rischiesti non saranno restituiti. Le opinioni espresse degli autori non sono necessariamente quelle di *Personal Software*.

È disponibile a richiesta una "Guida per gli autori" contenente le informazioni necessarie alla stesura di un articolo o di un programma, oltre a tutte le indicazioni di carattere amministrativo. Richiedetela all'indirizzo indicato.

UNA PUBBLICAZIONE DEL GRUPPO EDITORIALE JACKSON

PERSONAL SOFTWARE

ANNO 2 N. 5 MARZO-APRILE 1983

DIRETTORE RESPONSABILE: Giampietro Zanga

DIRETTORE: Mauro Boscarol

REDAZIONE: Completo Software - Padova

HANNO COLLABORATO A QUESTO NUMERO:
R. Amasino, E.M. Albani, S. Brown, W.L. Colsher, F. Ferrari Bravo, E. Ferreguti, J. Grout, E. Mitchell, M. Redolfi, F. Santini, M. Spero, S. Ventura.
Copertina e illustrazioni: MORGANA artefatti comunicativi - Padova
Fotocomposizione: Composizioni Grafiche - Padova

CONTABILITÀ: Franco Mancini, Roberto Ostelli, Mariella Luciano, Franca Anelli, Sandra Cicuta, Gabriella Napoli

DIFFUSIONE E ABBONAMENTI: Luigi De Cao, Adela Bel Lozano, Ombretta Giannetto

AUTORIZZAZIONE ALLA PUBBLICAZIONE: Tribunale di Milano n. 69 del 20/2/1982

PUBBLICITÀ: Concessionario per l'Italia e l'Estero Reina s.r.l. Via Washington, 50 - 20146 Milano Tel. (02) 4988066/7/8/9/060 (5 linee r.a.) Telex 316213 REINA I

STAMPA: REWEBA - Brescia

Concessionario esclusivo per la DIFFUSIONE in Italia e all'Estero:
SODIP - Via Zuretti, 25 - 20125 Milano

Spedizione in abbonamento Postale Gruppo III/70
Prezzo della rivista L. 3.500. Numero arretrato L. 6.000. Abbonamento annuo (10 numeri) L. 30.000; per l'Estero L. 48.000
I versamenti vanno indirizzati a: Gruppo Editoriale Jackson - Via Rosellini, 12 - 20124 Milano - mediante emissione di assegno bancario, cartolina vaglia o utilizzando il c/c Postale numero 11666203.
Per i cambi di indirizzo, indicare, oltre naturalmente al nuovo, anche l'indirizzo precedente, ed allegare alla comunicazione l'importo di L. 500, anche in francobolli.

GRUPPO EDITORIALE JACKSON Srl

DIREZIONE, REDAZIONE, AMMINISTRAZIONE:
Via Rosellini, 12 - 20124 Milano - Telefoni: 68.03.68 - 68.00.54

SEDE LEGALE: Via Vincenzo Monti, 15 - 20123 Milano

DIREZIONE EDITORIALE: Giampietro Zanga e Paolo Reina
COORDINAMENTO EDITORIALE: Daniele Comboni

Il software di oggi e di domani

Mauro Boscarol

Verso quale direzione si sta sviluppando il software? A guardarsi attorno, la risposta è una sola: verso l'intelligenza artificiale.

Il software della prossima generazione non sarà solo in grado di trattare delle informazioni nel modo convenzionale che tutti conosciamo, ma sarà anche capace di elaborare delle conoscenze.

Il governo giapponese ne è così convinto che ha dato il via all'operazione per la costruzione di un computer della "quinta generazione", che disporrà di questo tipo di capacità. Oltre ad un alto livello tecnologico (per esempio circuiti ad effetto Josephson invece dei semiconduttori classici) che garantirà una maggiore velocità e potenzialità, la macchina, che sarà pronta nel 1987, disporrà di tutto il software necessario a conferirle quella che si chiama "intelligenza artificiale".

Di che cosa si tratta? La programmazione convenzionale, quella a cui tutti siamo abituati, è la programmazione di algoritmi che danno in dettaglio il procedimento passo-passo per la soluzione di un problema. Se si tratta di calcolare una funzione matematica, o elaborare delle buste paga, tutto bene.

Ma molte attività umane, come per esempio l'invenzione di dispositivi scientifici, il ragionamento, l'apprendimento, richiedono una certa "intelligenza".

Consideriamo un solo campo: l'esperienza. Prerogativa di un esperto umano, di un consulente, è prendere decisioni di fronte a dati incompleti e incerti, facendo ragionamenti del tipo: "se...allora forse... ma allora...potrebbe essere...". Questo tipo di ragionamenti, non possono essere effettuati da un computer provvisto di software tradizionale. Ci vuole una programmazione particolare: entriamo appunto nel campo dell'intelligenza artificiale.

I programmi che riescono a condurre ragionamenti del tipo appena illustrato, e a trarre delle conclusioni, "consigliando" l'utente, vengono detti *sistemi esperti*.

Un sistema esperto opera come un vero e proprio consulente. Fa domande circa l'argomento di cui è "esperto", cercando di determinare tutti i fattori che costituiscono le condizioni necessarie per poter trarre delle conclusioni con una certa credibilità. In parole povere, un sistema esperto "ragiona" quasi come un essere umano, per arrivare ad una conclusione, una diagnosi, un consiglio.

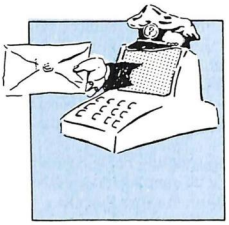
Ce ne sono già di operanti, alcuni molto famosi. Uno dei primi è MYCIN, un esperto medico, sviluppato a Stanford nel 1973. È un programma interattivo che assiste il medico nel raggiungere una conclusione diagnostica, ed opera sulla base di regole di lavoro del tipo "se...allora". Ce ne sono altri, specializzati nelle diverse discipline (geologia, arredamento, chimica,...).

Ma i sistemi esperti sono solo uno degli aspetti dell'intelligenza artificiale. Ve ne sono altri, ugualmente interessanti: l'apprendimento, il saper giocare e battere un avversario, il risolvere problemi e rompicapo, il saper dimostrare teoremi matematici, il saper interpretare i linguaggi "naturali" (inglese, italiano) piuttosto che i linguaggi di programmazione, il saper autoprogrammarsi, ed altri ancora.

Può interessare l'intelligenza artificiale nel campo dei personal computer?

Non sono dimensioni e potenzialità troppo ridotte per poter arrivare a simili risultati? A mio parere il fascino dell'intelligenza artificiale sta proprio qui: non è questione di enormi memorie, di velocità supersoniche o di multiprocessori, ma semplicemente (si fa per dire) di idee: anche una piccola macchina, programmata con le idee giuste, sa fare cose strabilianti. E dunque, l'intelligenza artificiale è campo che si apre anche alle piccole macchine, ai dilettanti, agli hobbysti: le sorprese e le soddisfazioni, è garantito, ci saranno per tutti. ■

POSTA



A tutti i possessori di DAI

Voglio subito complimentarmi per l'ottima rivista: finalmente un qualcosa che si interessi solo di software, lasciando da parte appunto l'hardware.

Ma veniamo al dunque. Posseggo un DAI 48 K ed il sempre valido ZX81. Mi rivolgo a voi per alcune delucidazioni.

È possibile, modificando con opportuni POKE le matrici relative ai caratteri per il video, ottenere altre figure premendo i tasti? Ciò mi permetterebbe un grosso miglioramento delle capacità grafiche. La mia domanda si riferisce ad entrambi i computer.

Ma a proposito del DAI è mai possibile che un computer di tali prestazioni non venga assolutamente considerato dal pubblico e dalle riviste specializzate? E con questo mi rivolgo a tutti i possessori del DAI: facciamoci sentire.

Dove posso trovare pubblicazioni sul DAI, oltre al manuale, si intende? Ringraziandovi anticipatamente e sperando di veder presto pubblicata questa mia, vi porgo i miei saluti, rinnovando i complimenti alla vostra rivista, che con una rubrica permanente sul DAI sarebbe perfetta!

Riccardo Taccia
Livorno

Giriamo le domande a coloro che hanno a disposizione un DAI. Le risposte che riceveremo verranno pubblicate in questa rubrica.

Qualcosa sul Texas, prego...

Spett.le Redazione,
sono un vecchio lettore di *Bit* e mi sono abbonato con piacere a *Personal Software*, che trovo interessante. Nel momento in cui scrivo è uscito solo il n. 3 della rivista e aspetto con ansia i successivi per sapere se ci sarà un po' di spazio dedicato al computer che mi sono comprato, il TI99/4A della Texas Instruments. Per tutto il 1982, anno in cui ho comprato *Bit* regolarmente tutti i mesi, questa macchina è stata ignorata (non so se è uscita la prova prima perché ho solo dei fascicoli sparsi) e questo mi sembra strano perché la Jackson collabora con la Texas anche per l'*Enciclopedia di elettronica & informatica*. Mi farebbe piacere che in futuro *Personal Software* estendesse i suoi programmi completi anche al 99/4A e anche la rubrica "I segreti dei personal", che trovo molto interessante. Ho letto sul n. 3 che vi interessano conversioni dei vostri programmi per altre macchine e vorrei sapere se ciò vale anche per il TI99/4A, perché se li pubblicate, almeno quelli che ritenete interessanti, io ve li mando.

Elio Vanneschi
Arezzo

Siamo estremamente interessati a tutto ciò che riguarda il Texas 99/4A. Il nostro problema, in questo momento, è che la macchina ha una diffusione ancora limitata (se pur in continua espansione); da ciò deriva una nostra difficoltà a reperire collaboratori preparati. L'abbiamo detto e lo ripetiamo: sollecitiamo i possessori di 99/4A a spedirci: (1) articoli sulla loro macchina, sul software o su qualunque altro argomento attinente o realizzato con la 99/4A; (2) conversioni di programmi già pubblicati in questa rivista; (3) piccole note e brevi routine per i nostri "Segreti dei personal". Dopodiché, se il materiale ci arriva, sarà nostra gioia il pubblicarlo.

Evviva il Commodore 64

Spett.le Redazione,
vi scrivo per complimentarmi per l'ottima rivista da voi realizzata e per fare alcune richieste.

Penso che in molti saremmo contenti se ampliaste la rubrica "Raccolta di routine Basic", in quanto gli esempi in essa contenuti sono facilmente implementabili in programmi scritti per qualsiasi tipo di personal computer, non legati perciò ai vari "dialetti" Basic o alle caratteristiche grafiche di particolari macchine.

La seconda cosa che mi preme è quella di vedere pubblicati programmi e articoli per il Commodore 64, mio da pochi giorni, e che vi assicuro, ha delle possibilità (specie nel campo dei videogiochi) meravigliose!

Infine un suggerimento: quando scrivete articoli per il PET/CBM o programmi, aggiungete le variazioni (minime d'altronde) per farli girare sul Commodore 64.

In questa rubrica rispondiamo alle lettere di carattere generale.
Scrivete a

Personal Software
Via Rosellini 12
20124 Milano

Sicuro che mi possiate accontentare, ringrazio in anticipo.

Giovanni Carella
Napoli

Per il Commodore 64, valgono le stesse considerazioni fatte nella risposta alla lettera precedente per il Texas 99/4A, e valide in generale per ogni personal di nuova produzione. È necessario un certo periodo di tempo prima che siano reperibili degli "esperti" della macchina, proprio perché l'esperienza si fa col tempo. Nel frattempo, rivolgiamo ai possessori di Commodore 64 lo stesso invito fatto ai possessori di Texas 99/4A: aspettiamo da voi articoli, conversioni, e segreti, e pubblicheremo tutto il materiale "pubblicabile" che ci arriverà.

Conversioni ZX81-Spectrum

Spett.le Redazione,
tempo fa vi ho scritto per chiedervi delle informazioni sullo ZX81, che avevo da tempo deciso di acquistare, ma poi ho sentito parlare del nuovo gioiello della Sinclair e ho deciso quindi di aspettare un po' ed acquistare quest'ultimo, anziché lo ZX81. Ormai però mi sono procurato già un "sacco" di programmi per lo ZX81 e, fortunatamente, ho letto su una rivista che questi andranno bene anche per lo Spectrum, ma bisognerà fare alcune modifiche, soprattutto per quanto riguarda i numeri degli indirizzi dei byte di memoria. Ora vorrei sapere se c'è (e qual è) un metodo pratico e veloce per adattare i programmi dello ZX81 allo Spectrum.

Desidererei inoltre che pubblicaste un programma del gioco degli scacchi per lo Spectrum (16 K RAM), dato che non sono ancora riuscito a trovarlo in nessuna rivista (nel caso che un programma di questo tipo fosse troppo lungo potreste anche pubblicarlo a puntate).

Vorrei inoltre sapere quali sono le case editrici che hanno pubblicato fino ad ora dei libri riguardanti lo ZX Spectrum (vorrei sapere l'indirizzo dello ZX User Club, che mi sembra che abbia a disposizione una vasta biblioteca di software per lo Spectrum).

E per finire vorrei sapere se un programma di 16 K RAM per lo ZX81 può entrare nello Spectrum sempre da 16 K RAM.

Gabriele Formaggio
Rovigo

P.S. Ma ogni quanti mesi esce la vostra splendida ed eccezionale rivista?

La nostra rivista è mensile... a partire dal numero scorso. Perlomeno questo è l'impegno che prendiamo con i lettori e che cerchiamo di rispettare. I problemi di una rivista come la nostra sono molti: articoli da leggere, programmi da provare, listati da rifare; certe volte capita di perdere due o tre giorni con un programma e poi decidere di buttare via tutto, perché non ne esce niente.

Come vede, da questo numero, pubblichiamo "I segreti" dello Spectrum, e abbiamo in preparazione un articolo sulla conversione ZX81-Spectrum. Alle sue domande risponderemo, assieme a molte altre, in quell'articolo.

Come programmare uno sfondo scorrevole?

Spett.
Personal Software,
sono un ragazzo di 14 anni e posseggo da questo Natale uno ZX81 Sinclair ed ormai vi programmo un po' di tutto; dalle declinazioni di latino (frequento il liceo scientifico) ai più stravaganti giochi; ma, durante la programmazione dei giochi mi trovo sempre davanti lo stesso problema: nei giochi di movimento come si può rendere l'idea di un "fondale scorrevole" che giri in senso orizzontale?

Distinti saluti e complimenti per la splendida rivista (alla quale ho già provveduto ad abbonarmi!).

Marcello Morchio
Genova

Noi, purtroppo, non abbiamo sottomano un'idea pronta. Cosa ne pensano i nostri lettori?

Il teorema di Jacopini-Böhm

Vi sarei molto grato se pubblicaste la dimostrazione del teorema di Jacopini-Böhm riguardante la potenza espressiva di un linguaggio formato solamente dalle strutture di sequenza e da IF...THEN...ELSE e DO...UNTIL...REPEAT (selezione binaria e iterazione).

Stefano De Santis
Bresso (MI)

Ma che domanda difficile! Il teorema di Jacopini-Böhm (due informatici teorici dell'Università di Roma) è uno dei risultati principali della "teoria" della programmazione strutturata. La sua dimostrazione richiede un livello di approfondimento teorico cui la nostra rivista, per sua natura, non può arrivare. Chi è interessato può vedere l'articolo originale di Jacopini e Böhm "Flow diagrams, Turing machines and language with only two formation rules" in Communications of the ACM *vol. 9, n. 3, maggio 1966.* ■

ELEDRA
PERSONAL COMPUTER
NEWS
FEBBRAIO 1983
10
PUBBLICAZIONE GRATUITA

UTER
GIUGNO 1982
1

Personal
izzazione
solo per
ri come
e i pro-
esto ri-
ostare
quio.
mer-
an-
m-
e

ACE1000

FRANKLIN COMPUTER CORPORATION

Personal Computer

ACE1200

- SOFTWARE COMPATIBILE APPLE II
- TASTIERA COMPLETA MAIUSCOLO/minuscolo
- ESPANDIBILE A 128K RAM

ELEDRA 3S S.p.A. - Viale Elvezia, 18 - 20154 Milano

IN VENDITA PRESSO I RIVENDITORI AUTORIZZATI PERSONAL COMPUTER ELEDRA 3S

RICHIESTA DI ABBONAMENTO GRATUITO

Spedire il coupon in busta chiusa a:
ELEDRA 3S S.p.A. - Viale Elvezia, 18 - 20154 Milano

BIT 13

☐ Desidero ricevere regolarmente Eledra Personal Computer News
☐ Ricevo già EPCN ☐ Desidero avere informazioni su ACE FRANKLIN
☐ Indicatemi il vostro rivenditore più vicino

Cognome e nome _____

Tit. _____ Attività _____

Ditta _____

Indirizzo _____

CAP _____ Città _____ Tel. _____

Tecniche di ricerca

Un confronto tra le prestazioni di tre tecniche di ricerca tabellare

di *Edward Mitchell*

Uno dei compiti più frequenti per cui viene usato un computer è la ricerca di dati specifici in tabelle o liste. Le routine di ricerca vengono applicate a problemi anche molto diversi tra loro, come elenchi telefonici elettronici, sistemi per inventario o per il carico/scarico di merci nelle aziende o nei compilatori Basic. Spesso viene utilizzata una semplice routine di ricerca sequenziale, facendo leggere al programma tutta la lista o la tabella con un ciclo, finché non viene trovato il record cercato. Ma con l'aumentare delle dimensioni della tabella da meno di 50 record a parecchie migliaia, il tempo di ricerca aumenta vertiginosamente. In casi come questi, restano solo due alternative: acquistare un computer più veloce o trovare una tecnica di ricerca più rapida. Per evidenti ragioni si preferisce ricorrere alla seconda soluzione.

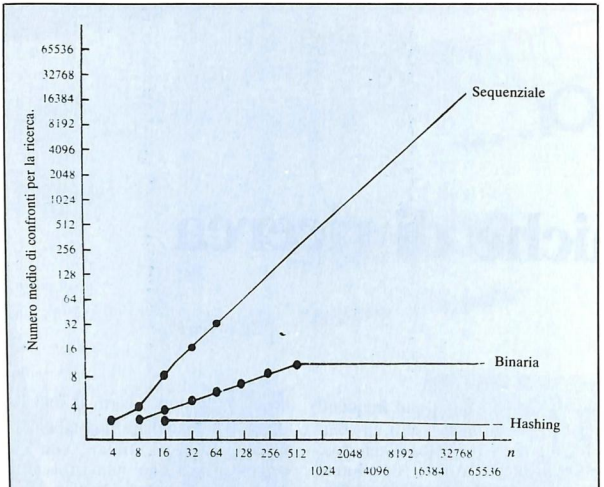
La prima soluzione di un problema che ci viene in mente è di rado la migliore. Esistono parecchi *algoritmi* per esaminare una tabella che non sono così ovvii. Un algoritmo è un insieme di regole ben definite da seguire per risolvere un determinato problema. Da un certo punto di vista, un algoritmo è la mappa della strada che ci porta da un punto ad un altro. Spesso studiando attentamente la configurazione geografica, possiamo trovare un percorso migliore. La ricerca sequenziale non è certo il solo sistema per esaminare una tabella o un file su disco. Infatti, con un computer esistono numerosi altri modi per eseguire una ricerca. Due di questi, la ricerca binaria e il metodo *hashing* (letteralmente "dello spezzatino"), verranno descritti in questo articolo.

Perfino la classica ricerca sequenziale può essere migliorata ordinando i record in modo che quelli che vengono richiamati più spesso appaiano all'inizio della lista.

La fig. 1 confronta le prestazioni relative di questi tre metodi di ricerca. La velocità del metodo hashing è indipendente dalla lunghezza della tabella. Il suo limite, invece, appare con il progressivo riempimento della lista.

La ricerca sequenziale

Un metodo per esaminare una lista di nomi è leggerla dall'inizio fino a incontrare l'elemento cercato, oppure fino alla fine. Questa tecnica, detta *ricerca sequenziale*, è semplice sia da capire che da mettere in pratica sotto forma di programma. Ma per una lista lunga, la ricerca sequenziale può costituire una perdita di tempo. Per esempio, una lista di 100 nomi richiede, in media, 50 confronti per ogni ricerca con esito positivo. Nel caso peggiore, quando il nome cercato non



n	Numero di confronti sequenziali	Numero di confronti binari	Hashing
4	2	2	3
8	4	3	3
16	8	4	3
32	16	5	3
64	32	6	3
128	64	7	3
256	128	8	3
512	256	9	3
1024	512	10	3
2048	1024	11	3
4096	2048	12	3
8192	4096	13	3
16384	8192	14	3
32768	16384	15	3
65536	32768	16	3

Figura 1. *Confronto tra tre tecniche di ricerca. Il grafico mostra che il metodo hashing di solito è il più veloce, necessitando di un numero di confronti minore sia del metodo sequenziale che di quello binario.*

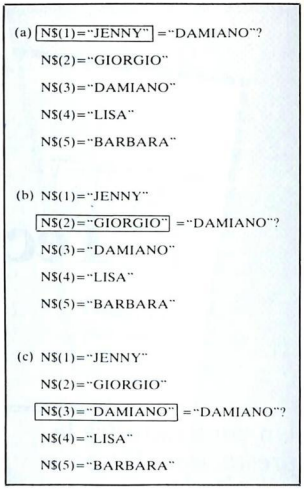


Figura 2. *Il metodo sequenziale all'opera. Questo metodo inizia la ricerca di DAMIANO dall'inizio della lista e prosegue la lettura finché non trova il nome o non raggiunge la fine della lista. In (a) l'algoritmo controlla il nome in N$(1). Poiché N$(1) non contiene DAMIANO, la ricerca prosegue in N$(2) come mostrato in (b) e finalmente trova DAMIANO in (c).*

esiste, dovremo esaminare tutti i 100 termini della lista prima di aver esaurito tutte le possibilità.

Consideriamo la breve lista composta di cinque nomi in fig. 2. Per cercare DAMIANO, dobbiamo cominciare controllando il nome memorizzato in N$(1). Siccome DAMIANO non corrisponde al contenuto di N$(1), proseguiamo con N$(2). Dopo il secondo confronto, l'algoritmo esamina N$(3) e trova il nome DAMIANO.

In generale, una ricerca sequenziale con esito positivo in una lista di *n* elementi disposti a caso richiede in media *n*/2 confronti. Il tempo necessario per la ricerca è direttamente proporzionale alla dimensione *n*. Se *n* raddoppia, anche il tempo medio di ricerca raddoppia. Una lista di 1000 nomi richiede un tempo dieci volte maggiore di una di 100.

Aggiungere un nuovo nome alla lista è semplice. Se assumiamo N uguale al numero dei nomi della lista di prima, allora poniamo N=N+1 e mettiamo il nome nuovo in N$(N). L'algoritmo 1 definisce la ricerca di tipo sequenziale in un linguaggio tipo Basic. Il listato 1 propone un programma tipo per la ricerca sequenziale, adatto per ogni personal computer IBM. Tutti i programmi descritti in questo articolo dovrebbero girare su tutti i computer che usano il Microsoft Basic.

Come si può osservare nel listato 1, un nome viene cancellato dalla lista, trovando la posizione del no-

```
10    MAX=100
20    DIM N$(MAX)
100   PRINT
      "SCEGLI: A(GGIUNGI  T(ROVA  S(TAMPA  C(ANCELLA  L(ASCIA
110   GET C$: PRINT C$
120   IF C$="A" THEN GOSUB 200: GOTO 100
121   IF C$="T" THEN GOSUB 300: GOTO 100
122   IF C$="S" THEN GOSUB 400: GOTO 100
123   IF C$="C" THEN GOSUB 500: GOTO 100
124   IF C$="L" THEN GOSUB 600: GOTO 100
130   GOTO 100
200   REM-----------------------------------------------------
210   REM SCRIVE UN NOME
220   INPUT"BATTI IL NOME DA AGGIUNGERE ";S$
225   IF S$="" THEN RETURN
230   GOSUB 2000
240   IF F=0 THEN GOSUB 1000
250   IF F=1 THEN PRINT S$;" E' GIA' NELLA LISTA.": RETURN
260   IF F=2 THEN PRINT"LA TABELLA E' GIA' COMPLETA CON ";MAX;
      " NOMI": RETURN
270   RETURN
300   REM-----------------------------------------------------
310   REM CERCA UN NOME
320   INPUT"BATTI IL NOME DA CERCARE ";S$
330   GOSUB 2000
340   IF F=0 THEN PRINT"NON TROVATO": RETURN
345   PRINT"TROVATO IN LOCAZIONE ";G: RETURN
350   RETURN
400   REM-----------------------------------------------------
410   REM STAMPA LA LISTA
420   GOSUB 3000
430   RETURN
500   REM-----------------------------------------------------
510   REM CANCELLA UN NOME
520   INPUT"BATTI IL NOME DA CANCELLARE ";S$
530   GOSUB 2000
540   IF F=0 THEN PRINT"NON TROVATO": RETURN
545   GOSUB 4000: PRINT"CANCELLATO"
550   RETURN
600   REM LASCIA
610   STOP
1000  REM-----------------------------------------------------
1010  IF N=MAX THEN F=2: RETURN
1015  F=0
1020  N=N+1
1030  N$(N)=S$
1040  RETURN
2000  REM-----------------------------------------------------
2010  REM RICERCA
2020  G=1
2030  IF G>N THEN F=0: RETURN
2040  IF S$=N$(G) THEN F=1: RETURN
2050  G=G+1
2060  GOTO 2030
3000  REM-----------------------------------------------------
3010  REM STAMPA
3020  FOR I=1 TO N
3030  PRINT N$(I)
3040  NEXT I
3050  PRINT
3060  RETURN
4000  REM-----------------------------------------------------
4010  REM CANCELLAZIONE
4020  N=N-1
4025  IF G=N+1 THEN RETURN
4030  FOR I=G TO N
4040  N$(I)=N$(I+1)
4050  NEXT I
4060  RETURN
```

Listato 1. *La tecnica di ricerca sequenziale.*

me nella lista e spostando tutti i nomi seguenti indietro di un posto, e ponendo N=N−1.

L'algoritmo 1 può essere migliorato per ridurre il numero di passi eseguiti nel ciclo di ricerca. Aggiungiamo il passo 0:

Poni N$(N+1)=S$

Togliamo il passo 2.

Tutte le ricerche vengono ora fermate dopo aver raggiunto N$(N+1) e il tempo di esecuzione si riduce perché per ogni passo del ciclo vengono eseguite solo due istruzioni anziché tre. Come parte del passo 3, occorre verificare se I>N, nel qual caso l'algoritmo deve dedurre che S$ non è stato trovato.

Miglioriamo la ricerca sequenziale

Se certi record sono più richiesti di altri, è desiderabile che appaiano più vicini all'inizio della lista, così da trovarli più rapidamente. Se 5 nomi in una lista di 100 sono cercati nel 50% delle occasioni, ha un senso metterli all'inizio della lista. Allora ci possiamo aspettare che il 50% delle ricerche vengano terminate entro il quinto confronto. Se assumiamo che gli altri 95 siano distribuiti casualmente, una ricerca con esito positivo richiederà in media 47 confronti, sempre meglio dei 50 richiesti da una disposizione casuale.

Visto che possiamo anche non sapere in anticipo con che frequenza verranno cercati i nomi, questa tecnica viene applicata meglio per tabelle che possono essere aggiornate per riflettere la distribuzione di richieste stimata. Ma usando una struttura dei dati chiamata *lista*, è facile creare una tabella che ordina automaticamente i record secondo la frequenza di accesso man mano che i nomi vengono inseriti o richiamati.

La ricerca binaria

Usando una ricerca binaria, il tempo necessario può diminuire, con l'ulteriore vantaggio che, se aumentiamo le dimensioni della tabella, il tempo aumenta ad un ritmo minore. La *ricerca binaria* prende il nome dal modo in cui la tabella viene ripetutamente divisa in due, finché viene trovato il record esatto o finché ciascun pezzo rimasto non è più ulteriormente divisibile. L'algoritmo per la ricerca

Figura 3. *La ricerca binaria di ENRICO comincia dalla metà della lista. Poiché ENRICO è minore di ERICA, l'algoritmo scarta tutti i nomi maggiori di ERICA e si concentra sulla metà superiore della lista. In (b) la ricerca è andata troppo indietro, così elimina tutti i nomi minori di CARLO. In (c) è ancora troppo basso, così l'indice avanza di un posto e trova ENRICO in (d).*

binaria richiede che i nomi siano ordinati secondo qualche criterio. Nel nostro esempio, l'ordinamento è alfabetico.

Prima di entrare nei dettagli della ricerca binaria, proviamo a eseguire una rapida ricerca per cercare di intuire come funziona. Invece di cominciare dall'inizio, partiamo dal centro della lista. Per localizzare ENRICO nella lista di fig. 3, confrontiamo ENRICO con il nome che appare al centro della lista, ERICA. Poiché ERICA è, alfabeticamente, *maggiore* di ENRICO, possiamo scartare tutta la metà della lista che segue ERICA, questa compresa.

La ricerca binaria fa il suo secondo tentativo controllando il nome che appare a metà tra l'inizio e la metà della lista. Confrontando CARLO con ENRICO, vediamo che la ricerca è andata troppo indietro. Possiamo eliminare allora tutti i nomi minori o uguali a CARLO.

Esaminiamo quindi il nome a metà dell'intervallo tra CARLO e ERICA, provando con la locazione numero 6, DARIO. Vedendo che è rimasta una sola locazione da esaminare la ricerca binaria trova ENRICO al posto 7. Per la lista di fig. 3, la ricerca binaria individuerà il nome o determinerà la sua inesistenza in quattro o meno tentativi. I tentativi per le ricerche con esito positivo in tabelle di 16 elementi sono in media 3: molto meno degli otto confronti previsti in media per la ricerca sequenziale.

La ricerca binaria trova una soluzione con così pochi tentativi perché divide iteratamente a metà la lista più piccola.

Dopo il primo tentativo, ci sono solo otto locazioni rimaste da controllare. Dopo la seconda comparazione, ne rimangono quattro. Ogni volta divide per due i nomi rimasti nella lista.

Finalmente, quando la lista non può più essere divisa ulteriormente, o si è trovato il nome cercato, o questo non esiste nella tabella. Ovviamente un metodo che dimezza le possibilità al primo colpo sarà più veloce del metodo sequenziale.

Prestazioni della ricerca binaria

Per una lista breve, come quella in fig. 3 con appena 15 nomi, il risparmio in tempo di ricerca è trascurabile rispetto al metodo sequenziale. D'altronde, il tempo effettivo impiegato dal computer per esaminare una lista breve è così piccolo, che fa veramente poca differenza quanto tempo impiega un particolare metodo di ricerca.

La richiesta di attenzione del computer per le comparazioni aggiuntive e per eseguire i calcoli connessi alla ricerca binaria può effettivamente rendere quest'ultima più lunga di quella sequenziale per liste piccole. Un algoritmo teoricamente veloce può non esserlo granché se tradotto per uno dei tipici centri di calcolo ed eseguito in *time-sharing*.

Ma con l'aumentare delle dimensioni della lista, i vantaggi della ricerca binaria diventano evidenti. Per una lista di 65535 nomi, la ricerca binaria garantisce il termine della ricerca in 16 o meno tentativi, ossia 2000 volte circa in meno del metodo sequenziale. Il numero massimo di comparazioni richieste per una lista qualsiasi di lunghezza n è uguale alla parte intera del logaritmo in base due di n, più uno.

Per chi non ha dimestichezza con i logaritmi, il numero di confronti è uguale al numero di volte che n può essere diviso ripetutamente per due, continuando ad avere un resto intero (per esempio, $\log_2 8=3$, perché $8/2=4$; $4/2=2$; $2/2=1$, cioè tre divisioni).

Aggiunta di nomi in una tabella ordinata

Aggiungere nomi ad una tabella ordinata è un processo lungo e poco efficente, come risulta anche dall'algoritmo 3. Per aggiungere un termine, dobbiamo prima controllare che non sia già presente nella tabella. Se esiste, l'algoritmo genera un messaggio di errore, altrimenti la ricerca binaria si ferma nel punto in cui deve essere inserito il nuovo nome. Tutti i nomi seguenti vengono allora spostati di un posto in avanti e quello nuovo viene inserito.

Per esempio, per inserire DAVIDE nella lista di fig. 4, chiamiamo

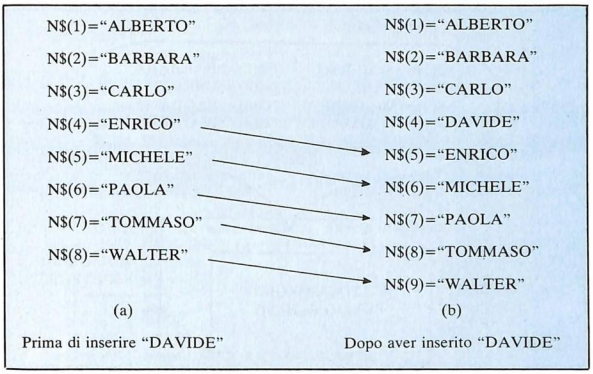


Figura 4. *Aggiunta di DAVIDE in una tabella ordinata. Dopo aver trovato la corretta locazione per il nuovo nome, tutti i nomi seguenti sono spostati in giù di un posto nella lista. Poi a N$(4) viene assegnato il valore di DAVIDE.*

prima la routine di ricerca binaria (algoritmo 2) per vedere se c'è già. Poiché il nome non c'è, creiamo un "buco" al posto in cui deve stare. L'algoritmo 2 termina la ricerca con G che contiene il valore della locazione precedente quella in cui deve essere inserito il nuovo termine.

La fig. 4 illustra l'inserzione, dove i nomi che seguono DARIO vengono spostati di un posto avanti nella lista, e DAVIDE viene inserito alla locazione 7. L'algoritmo 3 migliora il precedente, usando dei "puntatori" alle stringhe e spostando i puntatori anziché le stringhe.

La ricerca binaria può essere migliorata ordinando la tabella secondo la frequenza di riferimento. Usando un "albero binario" il nome più frequentemente usato nella lista appare al centro, così verrà esaminato per primo in ogni ricerca.

Il listato 2 è un gruppo di routine di esempio per implementare una ricerca binaria in Basic.

Le righe da 1 a 999 sono state omesse, essendo le stesse del listato 1. La subroutine 1000 aggiunge il nome S$ alla lista; la 2000 esamina la tabella cercando il nome S$, e torna con la sua locazione in G, mentre la subroutine 4000 cancella il nome alla locazione G. Bisogna eseguire un GOSUB 2000, prima di saltare a 4000, così G assumerà il valore corretto della locazione.

```
1000  REM RICERCA BINARIA
1010  REM-AGGIUNGE IL NOME
1012  IF N=MAX THEN F=2: RETURN
1013  F=0
1015  N=N+1
1020  FOR I=N TO G+1 STEP-1
1030  N$(I)=N$(I-1)
1040  NEXT I
1060  N$(G+1)=S$
1070  RETURN
2000  REM-----------------------------------------------
2010  REM-CERCA UN NOME
2020  L=1: R=N
2030  G=INT((L+R)/2)
2040  IF R<L THEN F=0: RETURN
2050  IF S$=N$(G) THEN F=1: RETURN
2060  IF S$<N$(G) THEN R=G-1: GOTO 2030
2070  L=G+1: GOTO 2030
3000  REM-----------------------------------------------
3010  REM-STAMPA LA LISTA
3020  FOR I=1 TO N
3030  PRINT N$(I)
3040  NEXT I
3050  PRINT
3070  RETURN
4000  REM-----------------------------------------------
4010  REM-CANCELLA UN NOME
4020  N=N-1
4030  FOR I=G TO N
4040  N$(I)=N$(I+1)
4050  NEXT I
4060  RETURN
```

Listato 2. *Subroutine di ricerca binaria.*

Il metodo hashing

Il miglior modo di ricerca in assoluto sarebbe una sfera di cristallo, con cui poter conoscere profeticamente la locazione del nome che cerchiamo. Immaginate una scatola magica, come in figura 5, che fornisce, specificando un nome, la sua esatta locazione in una lista. Un modo di ricerca di questo tipo si chiama "hashing", e converte il nome cercato in un numero che è l'indice di un vettore di nomi (la lista).

Per vedere come opera il metodo hashing, inventiamo una funzione HASH(STRINGA), che associa un numero intero ad una stringa nel modo seguente:

numero locazione = HASH(S$)

dove S$ è il nome cercato. Per convertire un nome in un numero assegniamo un valore numerico ad ogni carattere del nome e poi sommiamo questi valori. Associamo il numero 65 alla lettera A, il 66 alla B e così via, fino alla Z che vale

90. Questa corrispondenza lettera-numero è stata scelta perché i numeri sono i codici ASCII con cui il computer rappresenta internamente i caratteri. La maggior parte dei Basic ha una funzione del tipo C=ASC(S$) che fornisce il codice ASCII del primo carattere di S$. Per esempio, se S$="D", allora ASC(S$) darà 68.

Per produrre l'"hash", la funzione HASH somma i codici corrispondenti a ciascun carattere appartenente alla stringa. Per esempio, per ottenere l'hash di DINO si sommano i valori dei codici per ogni carattere in questo modo:

D	I	N	O	somma
68	73	78	79 =	298

DINO, perciò, ha un hash di valore 298.

Se i nomi sono memorizzati in un vettore definito con DIM N$(100), allora sorge un altro piccolo problema. Come facciamo ad usare il numero 298 come indice per N$ se questo deve essere minore o uguale a 100? Per comprimere il valore dell'hash nell'intervallo da 0 a 100, dividiamo 298 per 100 e il resto ottenuto, ossia 98, è l'effettivo valore dell'indice.

Per esempio, se definiamo R come resto di X diviso Y, nella maggior parte dei Basic possiamo scrivere:

```
        R=MOD(X,Y)
oppure  R=X MOD Y
oppure  R=(X/Y-INT(X/Y))*Y
```

Per inserire DINO nella tabella hash, N$(98) viene posto uguale a DINO. Più tardi, al momento della ricerca di DINO, con la stessa procedura si ottiene l'indice hash 98, trovando immediatamente il nome in N$(98).

Manipolazione delle collisioni

Un problema sorge quando alla funzione HASH diamo un altro nome che dà risultato 98. Per esempio, il nome "ADA" ha hash 198:

A	D	A	somma
65	68	65 =	198

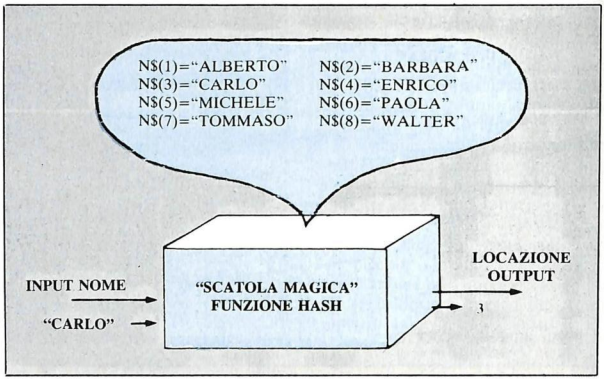


Figura 5. *La funzione hash opera come una scatola magica, che converte il nome CARLO in un numero che è la locazione dove va posto CARLO nella lista. Effettivamente la magia non c'entra. La funzione hash usa semplicemente l'aritmetica per convertire la stringa di caratteri in un numero. Naturalmente, più di un nome può avere lo stesso numero hash, provocando una collisione.*

Scalando come prima nell'intervallo da 0 a 100, otteniamo ancora il valore 98. Controllando N$(98) troviamo che nella tabella c'è già il nome DINO. L'algoritmo crea una collisione, ogniqualvolta due o più nomi lo stesso hash. Ci sono parecchi modi per manipolare le collisioni, ma qui ne mostriamo solo uno.

Quando ADA trova hash in un indice già usato, l'algoritmo decrementa l'indice e tenta l'inserimento alla locazione 97. Se N$(97) è libera, allora l'algoritmo la pone uguale a ADA; se invece è già occupata, l'indice viene di nuovo decrementato, ripetendo il processo finché non trova una locazione disponibile. Dopo aver incontrato lo zero, l'indice ritorna alla locazione 100 e continua a decrementare fino a trovare un buco libero nella lista.

Nel peggiore dei casi, l'algoritmo hashing può degenerare in una ricerca sequenziale, cercando dalla prossima locazione fino all'ultimo posto libero. Questa è la ragione per cui la funzione di hash deve cercare di eliminare sia le collisioni, che ogni tendenza dei nomi a concentrarsi in una certa area della tabella. Per massimizzarne l'efficenza, la tabella non dovrebbe mai essere riempita per più dell'80% della massima capacità. Con una tabella piena all'80%, il metodo hashing richiede circa tre confronti per ogni ricerca con esito positivo, senza riguardo alla dimensione della lista. Nel caso di ricerca con esito negativo, occorrono circa 13 confronti, perfino con tabelle molto grandi.

Ricerca su una tabella hash

Cercando ADA, si ripete il processo visto per l'inserimento. ADA, tramite la funzione HASH, fornisce l'hash 98. Se l'algoritmo non trova corrispondenza, l'indice viene decrementato a 97 e viene trovato il nome.

La ricerca termina quando l'algoritmo trova il nome cercato oppure quando l'indice incontra una casella della lista non utilizzata. Se il nome non esiste e la tabella è completa, l'algoritmo si perde in un ciclo senza fine. Perciò, in una lista lunga N si possono inserire al massimo $N-1$ nomi; deve esserci, come minimo, un posto vuoto nella tabella.

ALGORITMO 1
Algoritmo della ricerca sequenziale

Sia N$( ) un vettore di stringhe contenente la lista di nomi da cercare. Sia N la lunghezza della lista. Esempio: se ci sono cinque nomi, essi appariranno da N$(1) a N$(5) e N è uguale a 5. Sia S$ il nome da cercare. I serve per tenere conto delle locazioni esaminate della lista.

Passo	Azione
1	Sia I=1
2	IF I>N THEN ricerca terminata, S$ non trovato
3	IF S$=N$(I) THEN ricerca terminata, S$ trovato alla locazione I
4	Sia I=I+1 e GOTO passo 2

ALGORITMO 2
Algoritmo della ricerca binaria su tabella ordinata

Sia N$( ) una lista di nomi ordinata alfabeticamente. Sia N il numero dei nomi nella lista. Sia L il confine inferiore dell'intervallo di nomi da considerare, e R il confine superiore. Sia G il tentativo a metà strada tra L e R. S$ è il nome cercato nella lista N$( ).

Passo	Azione
1	Sia L=1 e R=N, confini iniziali dell'intervallo
2	G=INT((L+R)/2), prova a metà tra R e L
3	IF S$=N$(G) THEN ricerca terminata, G locazione di S$
4	IF S$<N$(G) THEN R=G−1 ELSE L=G+1, posiziona i nuovi confini
5	GOTO passo 2

ALGORITMO 3
Inserimento di un nuovo nome in una lista ordinata

Passo	Azione
1	Esegui l'algoritmo 2
2	Se viene trovato S$ con un errore, il nome esiste già nella lista
3	N=N+1
4	FOR I=N TO G+1 STEP −1: N$(I)=N$(I−1): NEXT I, sposta i nomi avanti di un posto nella lista
5	N$(G+1)=S$, inserisci il nome nella lista

ALGORITMO 4
Ricerca in un tabella hash

Sia S$ il nome da cercare nella lista di nomi N$( ). La funzione HASH( ) è descritta nel testo. S è la lunghezza della tabella.

Passo	Azione
1	H=HASH(S$)
2	IF N$(H)=" " THEN fine, S$ non trovato
3	ELSE IF N$(H)=S$ THEN fine, S$ alla locazione H
4	H=H−1, decrementa e prova di nuovo
5	IF H=0 THEN H=S
6	GOTO passo 2

ALGORITMO 5
Aggiunta di nomi ad una lista hash

S è uguale alla dimensione massima della tabella −1. Per esempio se MAX=100, allora S=MAX−1, ossia 99.

Passo	Azione
1	IF N=S THEN errore, lista piena
2	N=N+1, incrementa il numero dei nomi
3	H=HASH(S$), calcola l'hash
4	IF N$(H)=" " OR N$(H)="*" THEN N$(H)=S$, fine, nome inserito
5	H=H−1, collisione, perciò decrementa e riprova
6	IF H=0 THEN H=S
7	GOTO passo 4

ALGORITMO 6
Eliminazione di nomi da una lista hash

Passo	Azione
1	Chiama l'algoritmo 4
2	IF S$ non c'è THEN errore ELSE segna la locazione N$(H)="*" come "cancellata"

Rimozione di inserimenti hash

Cancellare un nome da una lista non è così semplice come potrebbe sembrare di primo acchito. Per esempio, se togliamo il nome DINO dal posto numero 98, quando cerchiamo ADA, troveremo l'hash 98, ma, risultando questo vuoto, l'algoritmo concluderà che ADA non esiste nella lista. Per risolvere questo problema, dobbiamo marcare la locazione 3 come "cancellata" anziché libera, così la ricerca potrà continuare alla locazione 2.

Il listato 3 mostra inserzione, rimozione e ricerca di record di dati in una tabella hash. Come nei precedenti listati, S$ contiene il nome da aggiungere o cercare nella lista. Per rimuovere un nome, bisogna chiamare la subroutine 2000 per verificare che il nome cercato esista realmente, poi con un GOSUB 4000 lo si cancella effettivamente.

Non occorre sommare tutti i valori di tutti i caratteri del nome. Se sapete che ci sono solo pochi nomi, potete ottenere l'hash anche solo con i primi pochi caratteri di ciascun nome.

Questa è una tecnica molto utile per la tabella delle parole chiave nei compilatori o interpreti Basic o per riconoscere i comandi nei programmi. Come esempio considerate un piccolo interprete Basic che usi le parole chiave IF, THEN, GOTO, GOSUB, FOR, NEXT, RETURN, PRINT e INPUT. Se calcoliamo l'hash di tutte queste parole, usando solo la loro prima lettera, l'unica collisione è tra GOTO e GOSUB.

```
1000  REM RICERCA HASC
1010  REM AGGIUNGE UN NOME
1030  IF N>=MAX-1 THEN F=2: RETURN
1040  N=N+1: F=0
1050  GOSUB 6000
1060  IF N$(H)="" OR N$(H)="*" THEN N$(H)=S$: RETURN
1070  H=H-1
1080  IF H=0 THEN H=MAX
1090  GOTO 1060
2000  REM-----------------------------------------------
2010  REM CERCA UN NOME
2020  GOSUB 6000
2030  IF N$(H)="" THEN F=0: RETURN
2040  IF N$(H)=S$ THEN F=1: G=H: RETURN
2050  H=H-1
2060  IF H=0 THEN H=MAX
2070  GOTO 2030
3000  REM-----------------------------------------------
3010  REM STAMPA LA LISTA
3020  FOR I=1 TO MAX
3030  IF LEN(N$(I))>0 THEN IF N$(I)<>"*" THEN PRINT N$(I)
3040  NEXT I
3050  PRINT
3070  RETURN
4000  REM-----------------------------------------------
4010  REM CANCELLA UN NOME
4020  N$(H)="*"
4025  N=N-1
4030  RETURN
5000  REM-----------------------------------------------
5010  REM LASCIA
5020  GOTO 600
6000  REM-----------------------------------------------
6010  REM CALCOLA H=HASC(S$)
6020  H=0
6030  FOR I=1 TO LEN(S$)
6040  H=H+ASC(MID$(S$,I,1))
6050  NEXT I
6060  H=H-INT(H/MAX)*MAX+1
6070  RETURN
```

Listato 3. *Il metodo hashing.*

Sommario

Gli algoritmi descrivono l'insieme di regole precise che i computer devono seguire per risolvere un problema.

L'algoritmo di ricerca sequenziale è solo uno dei molti modi esistenti per esaminare una lista o un file di record. La ricerca binaria e l'algoritmo hashing mostrano come si possano apportare grossi miglioramenti alla soluzione di un problema semplice, come la ricerca in una lista di nomi. E naturalmente ce ne sono molti altri e anche parecchie variazioni di ognuno.

I programmi perdono una grossa parte del loro tempo a manipolare strutture di dati; perciò, i nostri programmi dovrebbero trovare il miglior metodo disponibile per organizzare le informazioni nella memoria del computer. ■

4

Il controllo del numero di partita IVA

Mauro Boscarol

Rispetto al controllo del codice fiscale, illustrato nel precedente numero di questa rubrica, il controllo della partita IVA si presenta più semplice.

Dal punto di vista legislativo, il controllo della partita IVA è definito nelle "Avvertenze generali" dell'allegato 3 al D.M. 31 dicembre 1981 pubblicato sulla G.U. n. 3 del 5 gennaio 1982 con le rettifiche della G.U. n. 37 dell'8 febbraio 1982. (Tale controllo deve essere effettuato da tutti i soggetti che presentano gli allegati IVA su supporto magnetico.)

Il numero di partita IVA consiste di 11 caratteri numerici, di cui i primi sette individuano il numero del soggetto, quelli dall'ottavo al decimo l'ufficio IVA e l'undicesimo è il carattere di controllo.

Il seguente algoritmo di controllo è stabilito dalla legge citata.

1 I primi sette caratteri non possono essere tutti nulli.
2 I caratteri dall'ottavo al decimo devono essere compresi tra 001 e 095.
3 Moltiplicare per due le cifre di posto pari.
4 Sommare le cifre di posto dispari con le singole cifre ottenute dalla moltiplicazione per due di quelle di posto pari.
5 Il carattere di controllo deve essere uguale al complemento a 10 della somma ottenuta.

Per esempio, il numero di partita IVA della SIP è

00580600013

Per controllare la regolarità di questo codice, moltiplichiamo per due le cifre di posto pari, mentre lasciamo invariate quelle di posto dispari, ottenendo

1	2	3	4	5	6	7	8	9	10	11
0	0	5	8	0	6	0	0	0	1	
0	0	5	1+6	0	1+2	0	0	0	2	

e sommiamo le singole cifre ottenute (escluso ovviamente il carattere di controllo). Otteniamo 17, il cui complemento a 10 è appunto 3 (l'undicesimo carattere).

La routine di controllo

Per semplificare la realizzazione della routine di controllo, possiamo condurre una analisi preliminare del problema.

Per le cifre di posto dispari non ci sono difficoltà. Quelle di posto pari possono essere naturalmente da 0 a 9 (colonna *a* di tavola 1). Moltiplicandole per due si ottengono i numeri di colonna *b*, e sommando le singole cifre si ottengono i numeri di colonna *c*.

Il problema è dunque costruire una funzione che, dato il valore della colonna *a*, dia come risultato il valore corrispondente della colonna *c*.

a	*b*	*c*
0	0	0
1	2	2
2	4	4
3	6	6
4	8	8
5	10	1
6	12	3
7	14	5
8	16	7
9	18	9

Tavola 1.

La funzione è questa:

$$c=2*a-9*\mathrm{INT}(a/5)$$

A questo punto la realizzazione del programma è molto semplice.

Le linee da 10 a 50 sono un esempio di utilizzazione. La routine inizia nella riga 10000, dove viene posto a zero il valore della somma. Nella riga successiva viene posto a 1 il valore del flag F (1=errato, 0=corretto).

Nella riga 10020 si controlla che le prime sette cifre non siano tutte nulle, e nella successiva che il numero costituito dalle cifre di posto 8, 9 e 10 sia compreso tra 1 e 95. Esauriti questi controlli preliminari si inizia la somma delle cifre, con un FOR a passo 2. In ogni passo si somma direttamente la prima cifra (posto dispari) nell'istruzione 10080, e quindi si somma il valore della funzione detta sopra, calcolata con la seconda cifra.

Nella riga 10140 si calcola il complemento a 10 della somma ottenuta. Tuttavia se la somma ottenuta è multipla di 10, questa riga fornisce il valore 10 e non 0. La riga 10150 corregge questo errore.

Infine la riga 10160 controlla il risultato ottenuto con il carattere di controllo della partita IVA fornita.

```
 10 INPUT P$
 20 GOSUB 10000
 30 IF F=0 THEN PRINT "CORRETTO"
 40 IF F=1 THEN PRINT "ERRATO"
 50 STOP
 9000 REM
 9010 REM CONTROLLO DELLA PARTITA IVA
 9020 REM
 9030 REM PARAMETRO IN INPUT:
 9040 REM P$ (PARTITA IVA)
 9050 REM
 9060 REM PARAMETRO IN OUTPUT:
 9070 REM Z
 9080 REM (Z=0 CORRETTA, Z=1 ERRATA)
 9090 REM
10000 S=0
10010 F=1
10020 IF VAL(LEFT$(P$,7))=0 THEN RETURN
10030 G=VAL(MID$(P$,8,3))
10040 IF G<1 OR G>95 THEN RETURN
10050 FOR I=1 TO 10 STEP 2
10060 D$=MID$(P$,I,1)
10070 IF D$<"0" OR D$>"9" THEN RETURN
10080 S=S+VAL(D$)
10090 D$=MID$(P$,I+1,1)
10100 IF D$<"0" OR D$>"9" THEN RETURN
10110 D=VAL(D$)
10120 S=S+2*D-9*INT(D/5)
10130 NEXT I
10140 T=10-(S-INT(S/10)*10)
10150 IF T=10 THEN T=0
10160 IF T<>VAL(RIGHT$(P$,1)) THEN RETURN
10170 F=0
10180 RETURN
```

Le proposte per il nuovo Basic Standard

Dopo più di dieci anni dal Minimal Basic, sono in corso di approvazione le specifiche del nuovo Basic

di *Mauro Boscarol*

Nel 1963-64 John G. Kemeny e Thomas E. Kurtz del Dartmouth College (Hanover, New Hampshire, USA) svilupparono un linguaggio di programmazione che chiamarono Basic. Già di per sé, Basic significa "basilare", "semplice". Ma, come è uso degli americani, essi vollero far derivare questo nome dalle iniziali di una denominazione più complessa: *Beginners All-purpose Simbolic Instruction Code*, che precisa il senso che vollero dare al linguaggio; letteralmente: codice per istruzioni simboliche per scopi generali e per principianti.

Il linguaggio originale era pensato appunto come strumento per insegnare agli studenti i fondamenti della programmazione. Poiché non era stato progettato per trattare problemi "reali", la versione originale del Basic era alquanto limitata nelle caratteristiche e nelle istruzioni. Inoltre era naturale che venisse usata con piccoli computer o con terminali *time-sharing*, gli unici sistemi affrontabili da un utente alle prime armi.

Nonostante le sue limitazioni, la popolarità del Basic crebbe ogni giorno di più. Essendo un linguaggio interpretato (cioè un linguaggio che traduce ed esegue le istruzioni una per una) era molto adatto all'insegnamento *trial and error* (cioè per approssimazioni successive) della programmazione. Il tempo di risposta era immediato e lo studente apprendeva immediatamente la procedura corretta.

Queste qualità del linguaggio convinsero varie industrie produttrici di computer ad implementarlo sulle loro macchine aggiungendo nuove istruzioni e possibilità per "estendere" il linguaggio. Ogni produttore aggiungeva il suo particolare insieme di "caratteristiche speciali".

Ora, l'idea principale nell'usare un linguaggio evoluto come il Basic sta nel poter scrivere programmi in una forma che possa essere eseguita su più computer diversi. Fino a

I Basic

Dartmouth Basic
Il Basic originale sviluppato nel 1963-64 da Kemeny e Kurtz al Dartmouth College.

Microsoft Basic
Una estensione del Dartmouth Basic particolarmente adatta per i personal computer. È stata sviluppata negli anni '70.

ANSI Minimal Basic
Lo standard pubblicato dall'ANSI nel 1978 con la sigla X3.60-1978. Si può acquistarne una copia scrivendo all'American National Standards Institute, Inc., 1430 Broadway, New York, N.Y. 10018.

che si usano le istruzioni "fondamentali" del Basic, è possibile trasferire un programma da un computer ad un altro con relativa facilità. Ma nel momento in cui si iniziano ad usare alcune delle istruzioni "estese" di una particolare versione di Basic (quelle che danno al linguaggio delle capacità veramente interessanti) non si può più contare sulla trasferibilità su diversi sistemi.

Per porre rimedio a questa situazione l'*American National Standards Institute* (ANSI), analogo al nostro UNI, ha deciso, all'inizio degli anni '70, di costituire una commissione (la commissione X3J2) incaricata di sviluppare uno standard per il Basic. La commissione ha tenuto la sua prima riunione nel gennaio del 1974.

Ora siamo in pieni anni '80 e lo standard definitivo non è stato ancora pubblicato.

Di questa situazione hanno tuttavia approfittato alla fine degli anni '70 due intraprendenti giovanotti americani che hanno creato il "Microsoft Basic", un linguaggio che rispetto al Dartmouth Basic presenta numerosi miglioramenti e diverse istruzioni aggiuntive. La velocità dell'interprete, la sua compattezza e la possibilità di collegamenti con il linguaggio macchina ne hanno fatto il linguaggio ideale per il personal computer, la cui industria era allora appena nata ma già in piena espansione. Il Microsoft Basic è stato implementato su NCR, ADDS, DTC, Commodore PET/CBM/VIC, Apple, Ohio Scientific, TRS-80, Sorcerer, Atari, Texas, NEX e la lista può allungarsi a piacere.

Il Microsoft Basic è diventato, per un certo periodo di tempo, lo standard *de facto* dei personal computer. Ovviamente ogni produttore aveva (ed ha) la sua particolare versione, modellata sulle caratteristiche della propria macchina; tutte le versioni tuttavia si rifacevano a principi generali che contraddistinguevano appunto il Microsoft Basic.

Ma torniamo al comitato dell'ANSI, che avevamo lasciato alla riunione del gennaio 1974. Perché da allora ad oggi non si è riusciti a pubblicare uno standard? Il processo di standardizzazione è complesso e articolato, ma è necessaria questa lentezza? La risposta data da E. Kurtz, uno degli inventori del Basic, che oggi presiede appunto quel comitato, è duplice.

Innanzitutto, egli dice, dal '74 ad oggi sono stati prodotti due standard: il Minimal Basic, apparso nel '78, e la "bozza" apparsa l'anno scorso. Inoltre, egli continua, la standardizzazione del Basic è complicata dal fatto che si tratta di un linguaggio in rapida evoluzione. In altre parole, il Basic è un bersaglio mobile.

Il Minimal Basic è in effetti stato pubblicato nel 1978 (con sigla X3.60-1978). Ma, a causa del rapido sviluppo della tecnologia in quegli anni, che ha reso obsolete le sue modeste capacità, non è riuscito ad affermarsi. Tanto più che in quegli anni imperava il Microsoft Basic, che, come abbiamo visto, era esattamente ciò che ci voleva per i micro di allora.

Ed è corretta anche la seconda affermazione di Kurtz. Partito con una dozzina di istruzioni, e la possibilità di manipolare solo numeri, oggi il Basic ha molte più istruzioni, per la manipolazione di numeri, stringhe, vettori, matrici, file, grafica e musica. Le variabili di una sola lettera ora hanno nomi più lunghi. Esistono i Basic strutturati con IF...THEN...ELSE, insomma una miriade di Basic.

Tutte queste diverse versioni sono state raccolte nella "bozza" di

Il cammino del nuovo standard

Nell'estate del 1972 la bozza di standard è stata approvata. Successivamente è stata trasmessa al comitato X3 e quindi verrà trasmessa al comitato SPARC (*Standard Planning and Requirements Committee*). L'approvazione del comitato è prevista per l'inizio di quest'anno, e a quel punto inizierà un periodo di 120 giorni di "commenti del pubblico".

Alla fine di questo periodo i vari comitati ANSI studieranno le reazioni del pubblico e decideranno se fare o no revisioni allo standard proposto. Per luglio è prevista una riunione del comitato X3J2 per l'approvazione finale.

ACOS	arco coseno
ANGLE	angolo, data la base e l'altezza
ASIN	arco seno
CEIL	opposto di INT, parte intera superiore
COSH	coseno iperbolico
COT	cotangente
CSC	cosecante
DATE	data
DEG	da radianti a gradi
EPS	il minimo numero positivo (epsilon)
FP	parte decimale, X−INT(X) se X>0
INF	il massimo numero positivo (infinito)
IP	parte intera, INT(X) se X>0
LOG10	logaritmo in base 10
LOG2	logaritmo in base 2
MAX	massimo
MIN	minimo
MOD	modulo (resto di divisione intera per numeri positivi)
PI	pi greco
RAD	da gradi a radianti
REM	resto (come MOD per i numeri positivi)
ROUND	arrotondamento
SEC	secante
SINH	seno iperbolico
TANH	tangente iperbolica
TIME	ora, minuti, secondi
TRUNCATE	troncamento, riduce le cifre significative

Tavola 1. *Le nuove funzioni numeriche del Basic Standard (angoli espressi in gradi o radianti).*

standard, il che significa che il compito tecnico del comitato è terminato. Ora la bozza deve seguire la trafila burocratica per essere approvata (vedi riquadro) e se tutto va bene fra qualche anno avremo il nuovo Basic.

Quali sono le caratteristiche del nuovo Basic Standard

Riassumiamo qui di seguito le principali novità.

Commenti

Oltre al REM si potrà usare il punto esclamativo (!).

Nomi delle variabili

Fino a 31 caratteri, maiuscoli o minuscoli.

Operazioni numeriche

La grossa novità è che l'aritmetica sarà decimale in virgola mobile (e non binaria o esadecimale). Quindi 2.29+4.71 darà 7.00 non 6.99999. E ancora .1+.1+.1+.1+.1+.1+.1+.1+.1+.1 darà esattamente 1.

Funzioni numeriche

Oltre a quelle già presenti nel Minimal (ABS, ATN, COS, EXP, INT, LOG, RND, SGN, SIN, SQR, TAN) verranno implementate quelle indicate in tavola 1.

Operazioni di stringa

La concatenazione (&) e l'estrazione di sottostringa: RIGA$(4:7) fornisce dal quarto al settimo carattere della stringa RIGA$.

L'indicazione di sottostringa può apparire anche a sinistra del segno di uguale. Spariscono così SEG$, MID$, LEFT$, RIGHT$, ecc.

Funzioni di stringa

La nota CHR$ e la sua opposta, ORD. La nota STR$ e la sua opposta VAL. LCASE$ e UCASE$ convertono da maiuscolo a minuscolo e viceversa. DATE$ e TIME$ forniscono la data e l'ora sotto forma di stringa. POS cerca una stringa in un'altra stringa.

Istruzione LET

Sorpresa: è obbligatoria. La ragione principale è di ridurre il numero di parole riservate, mantenendo semplice la scansione dell'istruzione. In questo modo solo le funzioni senza argomenti sono riservate (come RND) e le parole NOT, PRINT, REM e ELSE. Non si può cioè scrivere LET RND=3. Però si può scrivere LET INPUT=3.

Vettori e matrici

Solo ad una o due dimensioni. Non esistono più i dimensionamenti per *default*. Vettori e matrici devono essere dimensionati prima dell'uso.

Una serie di istruzioni MAT comprendono input ed output di matrici, moltiplicazione scalare, somma, sottrazione e moltiplicazione di matrici, e le funzioni INV (inversione), TRN (trasposizione), DOT e DET (determinante).

Salti e decisioni

Ci saranno sempre GOTO e IF...THEN. Inoltre, l'IF multistruzione:

```
IF espressione THEN
...
ELSE
...
END IF
```

che naturalmente può essere scritta

Quasi Basic

Nel numero di aprile 1979 di *Byte* un lettore ha scherzosamente proposto alcune nuove istruzioni e funzioni per il Basic Standard. Eccone alcune:

Assegnazione

```
10 LET A # 4
20 LET A ≃ 19
```

Assegna ad A qualunque valore che non sia 4.
Assegna ad A un valore *quasi* uguale a 19.

IF-MAYBE (se-forse)

```
30 IF G=12 MAYBE 40
```

Se G è 12, *forse* vai a 40 (provate a fare delle nidificazioni di questa IF).

ABOUT (circa)

```
40 FOR N=1 TO ABOUT 100
   :
50 NEXT N
```

Questa istruzione si usa quando non si è completamente sicuri sul numero di volte che il ciclo deve essere eseguito.

FUZZ (confusione)

```
60 FUZZ=39
```

Questa funzione dice al Basic come trattare gli errori. Se FUZZ=0 il programma viene eseguito correttamente indipendentemente dagli errori presenti. Se FUZZ=99 il sistema va in *crash* appena viene rilevato un minimo errore logico o anche sintattico.

Un "bug" nel Basic

W. D. Maurer, del dipartimento di ingegneria elettrica e informatica dell'università George Washington ha trovato un errore comune a molte versioni di Basic per microcomputer. Il programma che illustra il "bug" è il seguente.

```
10    DIM T(100)
20    PRINT"QUANTI NUMERI?"
30    INPUT N
40    PRINT"BATTI ";N;" NUMERI"
50    FOR C=1 TO N
60    INPUT T(C)
70    NEXT C
80    FOR C=1 TO N
90    IF T(C)=0 THEN 130
100   NEXT C
110   PRINT"ZERO NON E' PRESENTE"
120   GOTO 140
130   PRINT"ZERO E' PRESENTE"
140   FOR R=1 TO N-1
150   FOR C=R+1 TO N
160   IF T(R)=T(C) THEN 210
170   NEXT C
180   NEXT R
190   PRINT"NESSUNA DUPLICAZIONE"
200   GOTO 220
210   PRINT"T(";R;")=T(";C;")"
220   END
```

Il programma accetta alcuni numeri da tastiera, verifica la presenza dello zero e di duplicati. Ecco un esempio di esecuzione corretta

```
QUANTI NUMERI?
?5
BATTI 5 NUMERI
?3
?7
?9
?23
?9
ZERO NON E' PRESENTE
T(3)=T(5)
```

e un esempio di esecuzione errata

```
QUANTI NUMERI?
?5
BATTI 5 NUMERI
?4
?0
?7
?12
?6
ZERO E' PRESENTE
?NEXT WITHOUT FOR ERROR IN 180
```

Non c'è una semplice spiegazione per questo "bug". È chiaro che l'errore avviene nella linea 180. Il messaggio d'errore dice che c'è un NEXT senza FOR, il che è chiaramente falso. L'errore è stato riscontrato su

APPLE (Applesoft)
Atari 800 e 400
Commodore PET
Ohio Scientific Challenger 1P
TRS-80 Modello I (level II Basic)

Come si comporta il vostro sistema? E come spiegate l'errore? Se volete saperne di più, leggete a p. 188 del numero di gennaio 1981 di *Byte*.

anche su una riga.

Cicli

Oltre il FOR...NEXT, ci sarà il DO...LOOP, che può essere unito ad un WHILE o a un UNTIL

```
DO UNTIL I>N OR A<R
...
LOOP
DO WHILE A=3
...
LOOP
```

Da un ciclo di DO si potrà uscire con un EXIT LOOP.

Scelta multipla

L'istruzione SELECT permetterà di scegliere una tra tante alternative

```
SELECT DADO
CASE 7,11
PRINT"VINCI"
CASE 2,3,12
PRINT"PERDI"
CASE ELSE
PRINT"TIRA ANCORA"
END SELECT
```

Funzioni, sottoprogrammi e chaining

Nel Minimal Basic era possibile definire istruzioni su una sola riga e subroutine. Con il nuovo standard sono possibili le definizioni di funzioni su più righe, i sottoprogrammi e il *chaining* di programmi. Le definizioni di funzioni su più righe iniziano con DEF e finiscono con END DEF.

I sottoprogrammi sono esterni al programma principale ed esterni tra loro. Le variabili interne sono locali, i parametri possono essere numeri, stringhe, vettori o matrici o numeri di canale (per i file).

L'istruzione CHAIN permette ad un programma di mandare in esecuzione un altro programma, che può essere in un linguaggio diverso. I due programmi si passano dalle informazioni mediante una lista di argomenti.

Input e output

L'istruzione LINE INPUT permette di introdurre una intera linea con virgole, apici e blank. L'istruzione PRINT USING permette di dare un formato all'output.

File

Saranno previsti quattro tipi di file: sequenziali, a lunghezza variabile di record (*stream*), relativi (accesso casuale) e con chiave, e tre tipi di record: *display*, interni e *native*. I record *display* sono quelli prodotti da istruzioni PRINT: stringhe di caratteri con CR e LF al termine. I record interni conten-

gono valori in qualche formato interno.

Trattamento degli errori

La costruzione per intercettare le "eccezioni", come giustamente vengono chiamate dal Basic Standard, è questa

```
WHEN EXCEPTION IN
...
USE
...
END WHEN
```

Sono previste altre istruzioni per costruzioni più elaborate.

Grafica

L'istruzione PLOT può tracciare punti o linee con varie modalità:

```
PLOT X,Y
PLOT (X1,Y1);(X2,Y2)
PLOT
```

Se il fascio è *off* le prime due istruzioni tracciano rispettivamente un punto in (X,Y) e una linea da (X1,Y1) a (X2,Y2). Se il fascio è *on* tracciano anche una linea dal punto precedente a (X,Y) o (X1,Y1) rispettivamente. La terza istruzione spegne il fascio se è *on*.

I punti da tracciare sono dati in coordinate utente, specificati da un'istruzione WINDOW.

Il programmatore può specificare il formato fisico dello schermo, predisporre il colore e lo stile delle linee, e verificare lo stato di queste quantità.

Complicati disegni si possono realizzare mediante disegni più semplici con istruzioni PIC, che sono come sottoprogrammi. Possono poi essere trasformati, ruotati e spostati in vari modi.

Conclusioni

I primi commenti al nuovo standard sono già arrivati. Alcuni positivi ed altri negativi.

I positivi insistono soprattutto sull'orientamento strutturato che è stato dato al nuovo Basic, e sui sottoprogrammi esterni.

Le caratteristiche grafiche dello standard proposto hanno ricevuto un cauto benvenuto dagli insegnanti di computer.

I commenti negativi sono più vari. C'è chi obietta la mancanza di una dichiarazione di tipo variabile. E gli si risponde che è un tributo da pagare alla semplicità di implementazione. Chi, ironicamente, protesta invece sulla complessità del nuovo linguaggio, che rende difficile e lontana la sua implementazione sui vari computer.

La filosofia del nuovo standard, in conclusione, non sembra essere quella di produrre uniformità tra le varie versioni di Basic, bensì di convogliare gli implementatori verso un obiettivo comune, e quindi aumentare le possibilità che un dato programma Basic possa essere usato su diversi sistemi.

Bibliografia

T. E. KURTZ: "On the Way to Standard Basic", *Byte* giugno '82, p. 182.

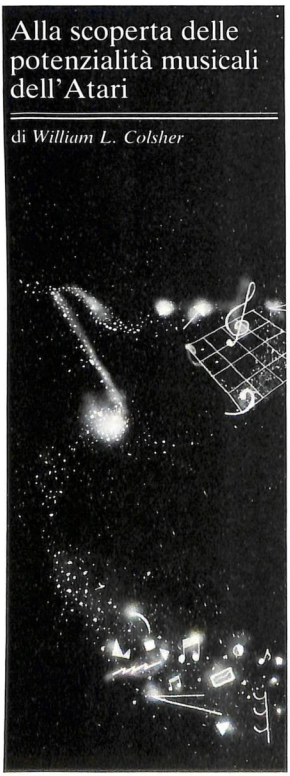
R. ANDERSON: "The Proposed ANSI Basic Standard", *Byte* febbraio '83, p. 194.

■

Comporre musica con l'Atari

Alla scoperta delle potenzialità musicali dell'Atari

di *William L. Colsher*



Quasi tutti i microcomputer possiedono un generatore di suoni. L'Apple II e l'Apple III hanno altoparlanti incorporati, come pure l'Hewlett Packard HP-85. Musica di qualche tipo si può ottenere anche tramite il collegamento con l'unità a cassetta di un TRS-80. Ma finora nessuno ha scoperto le potenzialità degli Atari 400 e 800. Ogni Atari ha incorporato un sintetizzatore a quattro voci. Ciascuna voce è in grado di riprodurre una singola nota a vari volumi e con diversa qualità timbrica, indipendentemente dalle altre tre. Le voci sono controllate con l'istruzione SOUND, definita nella tavola 1.

Giacché si possono riprodurre quattro diverse note alla volta, potete comporre degli accordi musicali. Questo programma mostra come si può fare.

Un accordo è formato da almeno tre note (tutti quelli che useremo noi saranno di tre note). Sarebbe noioso e poco pratico codificare espressamente le tre note di ciascuno dei quattro accordi per ognuna delle 12 note della scala. Per fortuna c'è una relazione fissa tra la nota fondamentale di un accordo e le altre. Perciò per ogni accordo è sufficiente conoscere il tipo e la sua nota fondamentale.

La tavola 3 mostra le relazioni numeriche tra la nota fondamentale e le altre nei quattro accordi che abbiamo scelto per il nostro programma. Una volta note queste relazioni, scrivere il programma è veramente facile.

La tastiera dell'Atari ha per l'appunto 12 caratteri disposti da sinistra a destra, senza contare i tasti di controllo (A, S, D, F, G, H, J, K, L, ;, + e * sulla seconda riga). Potete leggere la tastiera dell'Atari così com'è, ma sfortunatamente il valore che si ottiene non è il valore ASCII del carattere scelto. La ta-

voce	Voce è un intero da 0 a 3 che stabilisce la voce del sintetizzatore che dev'essere scelta da questa istruzione.
nota	Nota è un intero da 0 a 255 che dice al sintetizzatore quale nota riprodurre. I numeri più grandi danno le note più basse. La tavola 2 elenca alcuni di questi numeri e le note musicali a cui corrispondono.
tono	Tono è un intero pari da 0 a 14. Il valore 10 produce una nota normale.
volume	Volume è un intero da 1 a 15. Se si sta usando più di una voce, il volume totale non deve superare 32.

Tavola 1. *Parametri dell'istruzione SOUND per l'Atari.*

Tavola 3. *I fattori di moltiplicazione della nota fondamentale che producono la seconda e la terza nota dei vari accordi.*

Accordo	Moltiplicatore per la seconda nota	Moltiplicatore per la terza nota
Maggiore	.79166	.66666
Settima	.79166	.5625
Minore	.84027	.66666
Minore settima	.84027	.5625

vola 4 contiene i valori relativi ad ogni tasto usato nel programma, oltre alla nota o accordo ad esso assegnato. Le righe dalla 200 alla 260 stabiliscono nel programma queste relazioni.

Le righe dalla 1000 alla 1120 sono quelle che impiegano la maggior parte del tempo richiesto dal programma. Se non è stato premuto alcun tasto, l'istruzione PEEK(764) dà il valore 255. Dopo averne premuto uno, in quella locazione deve essere riposto il valore 255. Ciò viene effettuato nella riga 1110 e nella penultima riga di ogni routine di composizione dell'accordo.

Se è stato premuto un tasto, il programma controlla subito se si trattava di un tasto per la scelta dell'accordo (da 1 a 4). Se è così, salta alla routine corrispondente e riproduce l'accordo in questione usando l'ultima nota scelta. Ciò in relazione al modo in cui sono stati scritti i nomi degli accordi. Per riprodurre un accordo in re minore, dovreste prima premere il tasto H per scegliere una nota re e poi il tasto 3 per l'accordo minore.

Se il tasto premuto non era riservato alla scelta dell'accordo, il pro-

Nota	Numero di SOUND
La	144
Si bemolle	136
Si	128
Do	121
Do diesis	114
Re	108
Mi bemolle	102
Mi	96
Fa	91
Fa diesis	85
Sol	81
La bemolle	76

Tavola 2. *Il numero dell'istruzione SOUND corrispondente alle note dell'ottava che parte dal La basso fino al Do di mezzo.*

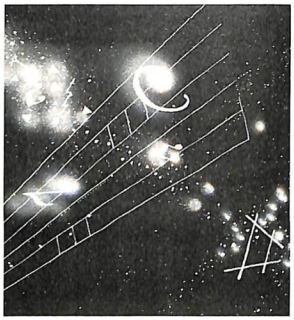
```
10 DIM REALTONE(12,2)
100 LASTBYTE=0
200 REM ***PREPARA LA STRINGA PER LA NOTA VERA
210 FOR I=1 TO 12
220 READ A,B
230 REALTONE(I,1)=A:REALTONE(I,2)=B
240 NEXT I
250 DATA 63,144,62,136,58,128,56,121,61,114,57,108
260 DATA 1,102,5,96,0,91,2,85,6,81,7,76
1000 BYTE=PEEK(764):REM *** LEGGE LA TASTIERA
1010 IF BYTE=255 THEN GOTO 1000:REM *** NESSUN TASTO
1020 IF BYTE=31 THEN GOTO 2000:REM *** ACCORDO MAGGIORE
1030 IF BYTE=30 THEN GOTO 2100:REM *** ACCORDO DI SETTIMA
1040 IF BYTE=26 THEN GOTO 2200:REM *** ACCORDO MINORE
1050 IF BYTE=24 THEN GOTO 2300:REM *** ACCORDO 7 MINORE
1060 REM *** CONTROLLA SE E' STATA CAMBIATA NOTA
1070 IF BYTE=LASTBYTE THEN GOTO 1000
1075 LASTBYTE=BYTE
1080 FOR I=1 TO 12
1090 IF BYTE=REALTONE(I,1) THEN TONE=REALTONE(I,2)
1100 NEXT I
1110 POKE 764,255:REM *** RIATTIVA LA TASTIERA
1120 GOTO 1000
2000 REM *** SUONA UN ACCORDO MAGGIORE
2010 SOUND 0,TONE,10,8
2020 SOUND 1,INT(TONE*0.79166+0.5),10,8
2030 SOUND 2,INT(TONE*0.66666+0.5),10,8
2040 POKE 764,255
2050 GOTO 1000
2100 REM *** ACCORDO DI SETTIMA
2110 SOUND 0,TONE,10,8
2120 SOUND 1,INT(TONE*0.79166+0.5),10,8
2130 SOUND 2,INT(TONE*0.5625+0.5),10,8
2140 POKE 764,255
2150 GOTO 1000
2200 REM *** ACCORDO MINORE
2210 SOUND 0,TONE,10,8
2220 SOUND 1,INT(TONE*0.84027+0.5),10,8
2230 SOUND 2,INT(TONE*0.66666+0.5),10,8
2240 POKE 764,255
2250 GOTO 1000
2300 REM *** ACCORDO DI SETTIMA MINORE
2310 SOUND 0,TONE,10,8
2320 SOUND 1,INT(TONE*0.84027+0.5),10,8
2330 SOUND 2,INT(TONE*0.5625+0.5),10,8
2340 POKE 764,255
2350 GOTO 1000
```

Tasto premuto	PEEK(764)	Nota o accordo riprodotto
A	63	La
S	62	Si bemolle
D	58	Si
F	56	Do
G	61	Do diesis
H	57	Re
J	1	Mi bemolle
K	5	Mi
L	0	Fa
;	2	Fa diesis
+	6	Sol
*	7	La bemolle
1	31	Maggiore
2	30	Settima
3	26	Minore
4	24	Minore settima

Tavola 4. *Corrispondenza tra i tasti dell'Atari e le note o i tipi di accordo prodotti nell'istruzione PEEK(764) della riga 1000 del listato.*

gramma esamina la sua tabella delle note e dei valori dei tasti (vedi la tavola 4) e, se si trova il valore relativo al tasto, pone il valore corrispondente nella variabile TONE. In questo modo non si modifica l'accordo da comporre.

Questo programma è solo un'introduzione alle potenzialità musicali dell'Atari. Se fate riferimento di nuovo alla tavola 1, noterete che c'è un parametro di controllo per il volume nella funzione SOUND. Potrebbe per esempio essere gestito da tastiera con le frecce verso l'alto e verso il basso. Noterete anche che ho usato solo tre delle quattro voci. Si potrebbe aggiungere un secondo "manuale" che riproduca le singole note usando la quarta voce. Il parametro del tono in SOUND può dar luogo ad alcuni suoni misteriosi e piacevoli, usando la quarta voce si può ottenere una sezione ritmica. Inoltre, potete sempre aggiungere altri accordi. ■



Il sistema Pretty Printer per l'Apple

Un sistema che permette la stampa ordinata di un listato

di *Sandro Ventura*

Molti lettori ci spediscono listati di programmi.

Generalmente, dopo averne deciso la pubblicazione, il compito più gravoso della redazione consiste nel ribattere il listato con una giusta distribuzione degli spazi, in modo che possa essere leggibile e ben impaginato. Questo programma fa automaticamente tutte queste cose per i listati dell'Apple II.

Chiediamo dunque ai nostri lettori di utilizzare il sistema Pretty Printer per ogni listato che inviano alla redazione, stampandolo su 64 colonne.

E invitiamo i possessori di PET, VIC, TRS-80, ecc. a convertire questo programma per le loro macchine (naturalmente i problemi sono diversi: i caratteri grafici, ...). [M.B.]

Molti linguaggi di alto livello (Pascal, C o Lisp) fanno uso di programmi di stampa appositamente concepiti per reimpaginare automaticamente il testo di un programma, in modo da renderlo più leggibile e quindi di più immediata consultazione.

Il programma qui proposto, scritto in Basic, è inteso appunto per produrre un listato "ordinato" dei programmi dell'Apple II. Impiegando il potente sistema operativo su disco, permette di manipolare programmi sorgente in Basic, in modo di conferire loro una veste adatta all'impaginazione. Il programma è in grado di elaborare programmi scritti in entrambi i tipi di Apple Basic (Integer o Applesoft).

Descrizione

Il programma consiste di due file, che devono risiedere nello stesso dischetto.

Il primo file chiamato APPRINT, ha lo scopo di trasferire il vostro programma dalla memoria dell'Apple, in cui l'avete caricato, in un text file su disco.

Il secondo file di nome PRETTY legge tale text file e ne elabora i dati per produrne una stampa impaginata.

Una caratteristica molto importante di tale stampa impaginata consiste nella possibilità di predeterminare il numero massimo di caratteri per riga, poiché il programma si occuperà automaticamente di spezzare le frasi qualora superino il margine stabilito.

Tutti i listati di questo articolo

```
10    D$=CHR$(4)
20    PRINT D$;"OPEN APPRINT"
30    PRINT D$;"WRITEAPPRINT"
40    PRINT"30000 D$=CHR$(4):REM CTRL-D"
50    PRINT"30005 PRINT D$;"+CHR$(34)+"OPEN LISTING"+CHR$(34)+
      ":PRINT D$;"+CHR$(34)+"DELETE LISTING"+CHR$(34)
60    PRINT"30010 PRINT D$;"+CHR$(34)+"OPEN LISTING"+CHR$(34)+
      ":PRINT D$;"+CHR$(34)+"WRITE LISTING"+CHR$(34)
70    PRINT"30020 POKE 33,30:LIST 1,30000"
80    PRINT"30030 PRINT D$;"+CHR$(34)+"CLOSE LISTING"+CHR$(34)
90    PRINT"30040 END"
100   PRINT"RUN 30000"
110   PRINT"RUN PRETTY"
120   PRINT D$;"CLOSE APPRINT"
```

Listato 1. *Programma per creare il text file Apprint.*

```
30000 D$ =  CHR$ (4): REM  CTRL-D
30005  PRINT D$;"OPEN LISTING": PRINT D$;"DELETE LISTING"
30010  PRINT D$;"OPEN LISTING": PRINT D$;"WRITE LISTING"
30020  POKE 33,30: LIST 1,30000
30030  PRINT D$;"CLOSE LISTING"
30040  END
RUN 30000
RUN PRETTY
```

Listato 2. *Contenuto del text file Apprint.*

sono stati ottenuti con questo sistema.

Come creare il sistema

Anzitutto dovete procurarvi un disco nuovo e inizializzarlo. Poi battete il programma riportato nel listato 1 ed eseguitelo. Questo programma crea un text file di nome APPRINT il cui contenuto è riportato nel listato 2. Una volta che vi siete accertati che il text file APPRINT è stato creato correttamente, potete cancellare il programma che vi è servito per crearlo.

APPRINT infatti è un file che viene creato *una tantum* (in senso letterale, non governativo).

Dopo aver creato APPRINT dovete accingervi a battere il listato 3 e salvarlo su disco con il nome di PRETTY.

A questo punto il sistema è creato e potete farlo funzionare.

Come funziona

Per far funzionare il sistema, dovete eseguire le seguenti operazioni:

1 Caricare in memoria il programma Basic che volete elaborare;
2 Inserire il dischetto contenente APPRINT e PRETTY e battere EXEC APPRINT.

A questo punto, le linee di programma di APPRINT vengono automaticamente messe in coda al programma da listare, e l'ultima riga, il RUN 30000 manda in esecuzione le righe dalla 30000 in poi che trasferiscono il programma in un file speciale di nome LISTING. Dopo il trasferimento viene automaticamente fatto partire il programma PRETTY.

Il principale inconveniente di questo programma è la relativa lentezza di esecuzione. La prima parte, di lettura e di prima elaborazione del programma, può impiegare da mezzo minuto a quasi dieci minuti di tempo, a seconda della lunghezza del programma da impaginare. In questo stadio ogni frase viene memorizzata in un elemento di una matrice (A(1), A(2),..., A(*n*)).

L'utente a questo punto deve scegliere il numero di colonne su cui impaginare, ed ha inizio la stampa. Premendo un tasto qualsiasi la stampa si interromperà fino al prossimo tasto premuto.

Limiti del sistema

Anzitutto, per la stampa si presume che lo slot della stampante sia il numero 1. Per cambiare numero, basterà cambiare opportuna-

Listato 3. *Pretty Printer.*

```
10    TEXT: DIM A$(500)
20    N=1: D$=CHR$(4): PRINT D$;"OPEN LISTING": PRINT D$;
      " READLISTING"
30    A$(0)="": C=0
40    HOME: VTAB 10: PRINT
      "STO LEGGENDO ED ELABORANDO IL PROGRAMMA."
50    GET A$
60    IF A$=CHR$(13) AND LEN(A$(N))=0 THEN 50
70    IF A$=CHR$(13) THEN N=N+1: A$(N)="": C=0: GOTO 50
80    A$(N)=A$(N)+A$
90    IF A$(N)<>"30000" THEN 50
100   PRINT D$;"NOMON": VTAB PEEK(37): CALL-868
110   N=N-1
120   PRINT D$;"CLOSE LISTING"
130   GOSUB 790
140   HOME: VTAB 16: INPUT"QUANTE COLONNE?";LUN: GOSUB 690
150   HOME: VTAB 16: PRINT"UN'ALTRA STAMPA?";: GET A$: PRINT A$:
      IF A$="S" THEN 140
160   STOP
170   REM
180   REM COMPATTAZIONE E RILEVAMENTO DELLA PUNTEGGIATURA
190   REM
200   FOR F=7 TO LUN+2: IF F>LUN THEN 260
210   IF F>=LEN(Z$) THEN RETURN
220   IF MID$(Z$,F,1)=CHR$(34) THEN VIR=VIR+1: P?=PV: PV=F
230   AP=1: IF VIR/2=INT(VIR/2) THEN AP=0
240   IF F=7 AND MID$(Z$,7,1)<>" " THEN 440
250   IF MID$(Z$,F,1)=CHR$(34) AND VIR/2<>INT(VIR/2) AND
      MID$(Z$,F-1,1)=" " THEN Z$=LEFT$(Z$,F-2)+MID$(Z$,F):
      F=F-1: GOTO 440
260   IF AP=1 THEN 440
270   AP=0: AC=0
280   IF MID$(Z$,F,1)=" " AND SP THEN Z$=LEFT$(Z$,F-1)+MID$(Z$,
      F+1): GOTO 280
290   SP=0
300   IF MID$(Z$,F,1)=" " THEN SP=1: GOTO 440
310   O$=MID$(Z$,F,1):
      IF O$<>"+" AND O$<>"-" AND O$<>"*" AND O$<>"/" AND O$<>"("
      AND O$<>"=" AND O$<>"<" AND O$<>">" AND O$<>"
      " AND O$<>"," AND O$ <  > "^" THEN 380
320   IF MID$(Z$,F-1,1)=" " THEN Z$=LEFT$(Z$,F-2)+MID$(Z$,F):
      F=F-1: GOTO 320
330   DF$=MID$(Z$,F,1)
340   IF F>LUN THEN 360
350   IF DF$="+" OR DF$="=" OR DF$="," OR DF$=";" THEN F=F
360   IF MID$(Z$,F+1,1)=" " THEN Z$=LEFT$(Z$,F)+MID$(Z$,F+2):
      GOTO 360
370   GOTO 440
380   IF MID$(Z$,F,1)=")" AND MID$(Z$,F-1,1)=" " THEN Z$=
      LEFT$(Z$,F-2)+MID$(Z$,F): F=F-1: GOTO 380
```

(segue)

mente l'istruzione 700. Per avere l'output sullo schermo del video, basterà invece cancellare la 700.

Il numero massimo di colonne è predisposto nella riga 700 a 80. Se avete una stampate a più colonne, variate opportunamente la riga 700.

Altre limitazioni riguardano il programma da formattare: i suoi numeri di linea devono essere minori di 30000, e non deve contenere più di 500 linee. Quest'ultimo parametro è presente nell'istruzione DIM A$(500) nella riga 10 di PRETTY, e quindi può essere variato agendo su questa DIM, ma compatibilmente con la memoria centrale a disposizione.

Riguardo invece alla prima questione, se un programma ha numeri di linea superiori a 30000 si può cambiare questo valore nel file APPRINT (creandone quindi un altro, con un valore opportuno).

Regole di formattazione

Le regole di formattazione usate dal programma sono principalmente le seguenti:

1. Il listato viene eseguito su un numero di colonne che l'utente specifica prima della stampa.
2. Per contenere la lista entro questo numero il programma spezza le istruzioni troppo lunghe secondo queste regole:
   - le singole parole chiave non vengono mai divise;
   - in caso di ":", di "+" o di ";" il programma provvede a spezzare le frasi in questi punti;
   - il programma cerca inoltre, per quanto possibile, di evitare di spezzare le stringhe incluse fra virgolette.
3. Gli spazi vengono distribuiti in questo modo:
   - uno spazio sempre dopo ":"
   - nessuno spazio (diversamente dal listato Apple che ne distribuisce generosamente ovunque) attorno ai dieci caratteri della linea 310 e prima delle virgolette.

Segue listato 3.

```
390   IF MID$(Z$,F,1)=":" AND MID$(Z$,F+1,1)<>" " THEN Z$=
      LEFT$(Z$,F)+" "+MID$(Z$,F+1)
400   IF MID$(Z$,F,1)=":" AND MID$(Z$,F-1,1)=" " THEN Z$=
      LEFT$(Z$,F-2)+MID$(Z$,F): F=F-1: GOTO 400
410   IF F>LUN THEN 430
420   IF MID$(Z$,F,1)=":" THEN DP=F
430   IF MID$(Z$,F-1,2)="FN" AND MID$(Z$,F+1,1)=" " THEN Z$=
      LEFT$(Z$,F)+MID$(Z$,F+2): GOTO 430
440   NEXT F
450   RETURN
460   REM
470   REM SPEZZAMENTO EVENTUALE DELLA FRASE
480   REM
490   AP=0: AC=0: VIR=0
500   DP=0: PV=0: PI=0: P7=0: SP=1: GOSUB 200
510   IF LEN(Z$)<LUN THEN PRINT Z$: RETURN
520   IF AP=1 THEN 910
530   IF MID$(Z$,LUN,1)=CHR$(34) THEN 1050
540   AC=0: X3=0
550   IF DP THEN PRINT LEFT$(Z$,DP): Z$="       "+MID$(Z$,DP+2):
      GOTO 500
560   FOR J=LUN+1 TO 7 STEP-1
570   IF J=LUN+1 THEN 600
580   IF MID$(Z$,J,1)=CHR$(34) THEN X3=ABS(X3-1)
590   IF X3=1 THEN 610
600   IF MID$(Z$,J,1)=" " THEN 620
610   NEXT J
620   IF J>PI AND J>7 THEN 650
630   IF PI THEN PRINT LEFT$(Z$,PI): Z$="       "+MID$(Z$,PI+1):
      GOTO 500
640   J=LUN
650   PRINT LEFT$(Z$,J): Z$="       "+MID$(Z$,J+1): GOTO 500
660   RETURN
670   REM
680   REM SEZIONE STAMPA
690   REM
700   PR# 1: PRINT CHR$(9);"80N"
710   HOME
720   FOR I=1 TO N: Z$=A$(I): GOSUB 490
730   IF PEEK(-16384)>127 THEN POKE-16368,0
740   NEXT I: PRINT
750   PR# 0: RETURN
760   REM
770   REM GIUSTIFICA A SINISTRA LE ISTRUZIONI
780   REM
790   IF LEFT$(A$(1),1)=" " THEN GOTO 870
800   FOR I=1 TO N
810   J=1
820   A=ASC(MID$(A$(I),J,1)): A=A-128*INT(A/128):
      IF(A>47 AND A<58) OR A=32 THEN J=J+1: GOTO 820
830   IF J>6 THEN 850
840   FOR K=J TO 6:
      A$(I)=LEFT$(A$(I),J-1)+"  "+RIGHT$(A$(I),LEN(A$(I))-J+1):
      NEXT
850   IF MID$(A$(I),7,1)=" " THEN A$(I)=LEFT$(A$(I),6)+
      MID$(A$(I),8): GOTO 850
860   NEXT I
870   X=FRE(0): RETURN
880   REM
890   REM SPEZZA LE STRINGHE FRA VIRGOLETTE
900   REM
910   IF AC=1 THEN 1050
920   FOR W=PV TO LEN(Z$): IF MID$(Z$,W,1)=CHR$(34) THEN 940
930   NEXT W
940   IF W-PV>LUN THEN GOTO 1050
950   FOR DR=PV TO 7 STEP-1
960   D$=MID$(Z$,DR,1)
970   IF MID$(Z$,DR,5)="PRINT" THEN DR=DR+4: DU=DR+1: GOTO 1040
980   IF MID$(Z$,DR,5)="INPUT" THEN DR=DR+4: DU=DR+1: GOTO 1040
990   IF D$=":" THEN DU=DR+2: GOTO 1040
1000  IF D$=";" OR D$="," OR D$="+" THEN DU=DR+1: GOTO 1040
1010  NEXT DR
1020  FOR DR=PV TO 7 STEP-1: IF MID$(Z$,DR,1)=" " THEN DU=DR+1:
      GOTO 1040
1030  NEXT DR: DR=PV-1: DU=DR+1
1040  PRINT LEFT$(Z$,DR): Z$="       "+MID$(Z$,DU): AC=1:
      VIR=VIR-1: GOTO 500
1050  REM TAGLIO
1060  PRINT LEFT$(Z$,LUN): Z$="       "+MID$(Z$,LUN+1)
1070  AC=1: GOTO 500
```

Estensioni e miglioramenti

Molti miglioramenti sono possibili. La questione della velocità si può risolvere compilando il programma PRETTY. Attualmente, per un programma di 250-300 linee il tempo di stampa è abbastanza elevato (20-30 minuti). Con la compilazione dovrebbe scendere a 5-7 minuti.

Altri miglioramenti riguardano la forma: sarebbe interessante includere nel programma un RENUMBER opzionale che rinumerasse tutte le linee a partire, per esempio, da 10 e con incremento 10 (o con valori variabili, dati in input). Infine potrebbe essere utile far precedere i numeri di linea ai quali c'è un riferimento (GOTO, GOSUB) da una freccia, per indicare che a quelle istruzioni si può arrivare anche dall'esterno.

Conclusioni

Il programma è utile ed interessante di per sé. Ma la tecnica di base utilizzata per manipolare un programma Basic come text file può essere di spunto per creare impaginatori più elaborati, programmi di conversione (per esempio da un Basic ad un altro) od altri più impegnativi programmi di editing. Lasciamo ai lettori l'onere di ulteriori esperimenti.

Spiegazione delle principali routine

10-130 Legge da disco il programma da impaginare e lo trasferisce nella matrice A(I)

200-450 Elimina spazi superflui, ne aggiunge se mancanti, controlla le stringhe alfanumeriche tra virgolette

490-660 Controlla lo spezzettamento delle istruzioni per mantenersi entro i limiti del numero di colonne

680-750 Preleva dalla matrice A(I) ogni istruzione per elaborarla

770-870 Incolonna a sinistra i numeri di istruzione (solo per Applesoft poiché l'Integer li ha già incolonnati)

890-1040 Evita che vengano spezzate stringhe più corte del numero di colonne

1060-1070 Spezza le stringhe più lunghe del numero di colonne ■

Questo programma è disponibile su un floppy disc per l'Apple II. Vedete in fondo alla rivista il "Servizio Programmi".

(continua dal numero precedente)

CHR$ funzione

La funzione CHR$ restituisce il carattere rappresentato dal codice ASCII dato come argomento. Per esempio CHR$(67) è il carattere B. La funzione CHR$ permette di stampare caratteri non normalmente accessibili dalla tastiera. (Si veda il riquadro sul codice ASCII).

Se il tuo computer non l'ha

Può essere simulata creando una siringa con tutti i caratteri ASCII (escluse le virgolette che servono da delimitatori):

A$="□!□$%'()*+,-.?0123456789...UVWXYZ"

Il primo dei caratteri ASCII stampabili è lo spazio (qui indicato con □), codice 32. Se si vuole stampare una A si deve sottrarre 31 dal codice di A, che è 65. Quindi 65−31=34, e MID$(A$,34,1), o A$(34,34) secondo l'interprete, significherà il carattere A.

CHR$(65)=MID$(A$,34,1)

e in generale, per N maggiore di 31,

CHR$(N)=MID$(A$,N−31,1).

Talvolta CHR$ viene sostituita con CHR, CHAR$, CHAR.

CINT funzione

Nel TRS-80 livello II CINT converte un numero nel suo valore intero (C=converte, INT=intero). Il numero assegnato a CINT non può essere maggiore di +32767 o minore di −32767. Per esempio, CINT(−4.65) stampa −5.

^ (accento circonflesso) operatore

È il simbolo dell'elevamento a potenza. Per esempio 3^2 significa 3^2 cioè 9. In alcuni calcolatori è sostituito dalla freccia verso l'alto (vedi). L'accento circonflesso è usato nell'Apple e può essere sostituito da CHR$(94) e CHR$(222).

CLEAR comando, istruzione

Azzera tutte le variabili e le stringhe. Talvolta può essere seguito da un numero, che specifica il numero di byte da riservare per la memorizzazione delle stringhe.

Per esempio 10 CLEAR 150 riserva 150 byte in memoria centrale per la memorizzazione delle stringhe.

In alcuni interpreti è sostituito da CLR (vedi).

CLOAD comando

Il comando CLOAD viene usato da alcuni interpreti per caricare un programma da un nastro su cassetta (C=cassetta, LOAD=carica). Per esempio CLOAD"A" cerca il programma di nome "A" su nastro e lo carica in memoria. Se non è specificato un nome, viene caricato il primo programma incontrato.

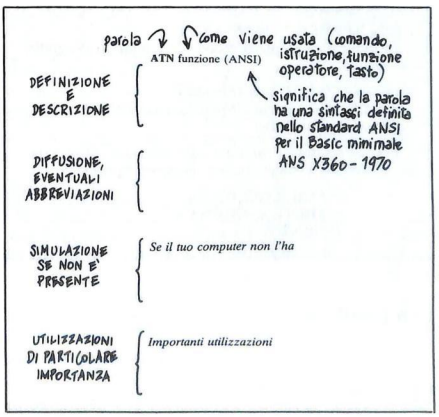
Talvolta CLOAD è sostituito da LOAD (vedi).

CLOSE comando, istruzione

Viene utilizzato per chiudere un file aperto da un comando o istruzione OPEN. Per esempio CLOSE 1 chiude il file il cui numero è 1.

CLR comando, istruzione

Il comando CLR azzera tutte le variabili. In Applesoft e in Basic TRS-80 livello II viene usato CLEAR (vedi). In Integer Basic e nel PET Basic, viene usato CLR.



Come usare questo dizionario

Il codice ASCII

Il codice ASCII (*American Standard Code for Information Interchanges*) è la codifica maggiormente usata nei personal computer per rappresentare lettere, numeri, caratteri grafici, funzioni speciali e caratteri speciali. Altri codici usati sono il BCD (*binary coded decimal*: decimale codificato binario) e l'EBCDIC (*extended binary coded decimal interchange code*). BCD è un codice a sei bit, mentre ASCII e EBCDIC sono codici a otto bit.

I codici ASCII, essendo formati da 8 bit, sono 256 e vanno da 00000000 a 11111111 (in esadecimale da $00 a $FF, in decimale da 0 a 255). In alcuni sistemi il bit più significativo è di controllo, e quindi i codici sono 128.

Una volta, il codice ASCII era uniforme per tutti i computer. Ora ogni sistema ha un differente codice ASCII, ma le caratteristiche essenziali si possono riassumere come segue.

Codici da 0 a 31 (da $00 a $1F)
Sono codici di controllo, che ogni macchina interpreta diversamente (vedi tavola 1). Di solito sono disponibili da tastiera battendo CTRL e il tasto di una lettera, oppure sono presenti con un tasto apposito.

Codici da 32 a 90 (da $20 a $59)
Questi sono gli stessi su quasi tutte le macchine, e rappresentano le cifre, le lettere maiuscole e i caratteri speciali più comuni (vedi tavola 2).

Codici da 91 a 95 (da $60 a $64)
Rappresentano alcuni caratteri speciali, diversi da sistema a sistema (vedi tavola 3).

Codici da 96 a 127 (da $65 a $7F)
In molti sistemi questi codici sono duplicati dei codici da 32 a 63.

Codici da 128 a 255 (da $8F a $FF)
Sono funzioni speciali, ripetizione di altri codici, o caratteri speciali o grafici.

Per vedere tutti i caratteri stampabili del vostro sistema e il relativo codice, battete questo programma

```
10 FOR X=32 TO 255
20 PRINT X; CHR$(X),
30 NEXT X
```

CLR/HOME tasto

Vedi CLS.

CLS comando, istruzione

Azzera lo schermo e porta il cursore in alto a sinistra. In Applesoft lo stesso effetto si può ottenere con HOME (vedi) o con ESC seguito da SHIFT @. Con GR e HGR si azzera lo schermo ma si passa al modo grafico; in Integer Basic si può usare CALL–936.

Nel PET Basic il tasto SHIFT CLR/HOME azzera lo schermo: si può usare anche tra le virgolette di una istruzione PRINT, o con PRINT CHR$(147). Il solo tasto CLR/HOME, senza SHIFT, riporta il cursore in alto a sinistra, senza azzerare lo schermo. Anche questo comando si può usare tra le virgolette di una istruzione PRINT o con PRINT CHR$(19).

Esa	Dec	Sigla	Significato
$00	0	NUL	*Null* (riempitivo)
$01	1	SOH	*Start of header* (inizio testata)
$02	2	STX	*Start of text* (inizio testo)
$03	3	ETX	*End of text* (fine testo)
$04	4	EOT	*End of transmission* (fine trasmissione)
$05	5	ENQ	*Enquire* (richiesta)
$06	6	ACK	*Acknowledge* (conferma)
$07	7	BEL	*Bell* (campanello)
$08	8	BS	*Backspace* (spazio indietro)
$09	9	HT	*Horizontal* (tabulazione orizzontale)
$0A	10	LF	*Line feed* (avanzamento linea)
$0B	11	VT	*Vertical tab* (tabulazione verticale)
$0C	12	FF	*Form feed* (avanzamento foglio)
$0D	13	CR	*Carriage return* (ritorno carrello)
$0E	14	SO	*Shift out*
$0F	15	SI	*Shift in*
$10	16	DLE	*Data link escape* (commutazione trasmissione)
$11	17	DC1	*Device control 1*
$12	18	DC2	*Device control 2*
$13	19	DC3	*Device control 3*
$14	20	DC4	*Device control 4*
$15	21	NAK	*Negative acknowledge* (conferma negativa)
$16	22	SYN	*Synchrounous idle* (sincronizzazione)
$17	23	ETB	*End of transmission block* (fine del blocco di trasmissione)
$18	24	CAN	*Cancel* (non valido)
$19	25	EM	*End of medium* (fine registrazione)
$1A	26	SUB	*Substitute* (sostituzione)
$1B	27	ESC	*Escape* (commutazione)
$1C	28	FS	*File separator* (separazione file)
$1D	29	GS	*Group separator* (separazione gruppi)
$1E	30	RS	*Record separator* (separazione record)
$1F	31	US	*Unit separator* (separazione unità)

Tavola 1. *I codici ASCII da 0 a 31*

Esa	Dec	Carattere	Esa	Dec	Carattere
$20	32	spazio	$3E	62	>
$21	33	!	$3F	63	?
$22	34	"	$40	64	@
$23	35	#	$41	65	A
$24	36	$	$42	66	B
$25	37	%	$43	67	C
$26	38	&	$44	68	D
$27	39	'	$45	69	E
$28	40	(	$46	70	F
$29	41	)	$47	71	G
$2A	42	*	$48	72	H
$2B	43	+	$49	73	I
$2C	44	,	$4A	74	J
$2D	45	-	$4B	75	K
$2E	46	.	$4C	76	L
$2F	47	/	$4D	77	M
$30	48	0	$4E	78	N
$31	49	1	$4F	79	O
$32	50	2	$50	80	P
$33	51	3	$51	81	Q
$34	52	4	$52	82	R
$35	53	5	$53	83	S
$36	54	6	$54	84	T
$37	55	7	$55	85	U
$38	56	8	$56	86	V
$39	57	9	$57	87	W
$3A	58	:	$58	88	X
$3B	59	;	$59	89	Y
$3C	60	<	$5A	90	Z
$3D	61	=			

Tavola 2. *I codici ASCII da 32 a 90*

CMD comando

Nel PET Basic, il comando CMD è simile a PRINT# eccetto che in questo modo il computer resta connesso con il dispositivo esterno. PRINT e LIST vengono quindi effettuate sul dispositivo esterno. Questa è la procedura per listare un programma su stampante

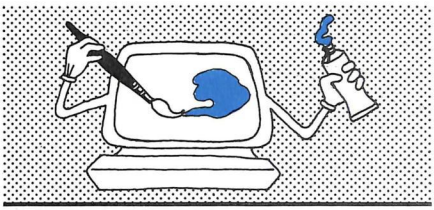
Esa	Dec	Apple II	Commodore	TRS-80	Atom
$61	91	[	[	↑	@ inverso
$62	92	\	\	↓	A inverso
$63	93	]	]	←	B inverso
$64	94		↑	→	C inverso
$65	95	_	←	_	D inverso

Tavola 3. *I codici ASCII da 91 a 95 su alcune macchine.*

OPEN 1,4: CMD 1: LIST

dove 1 è il numero logico del dispositivo, 4 il numero fisico (solitamente la stampante). Completata la lista, il dispositivo va chiuso con

PRINT#1: CLOSE 1



COLOR istruzione e comando

Nell'Apple specifica il colore da visualizzare sullo schermo. Per esempio COLOR=3 indica che i prossimi punti da visualizzare saranno viola.

Parametro	Bianco e nero	Bassa risoluzione	Alta risoluzione
0	nero	nero	nero 1
1	¼ bianco	magenta	verde
2	¼ bianco	blu scuro	blu
3	½ bianco	viola	bianco 1
4	¼ bianco	verde scuro	nero 2
5	grigio	grigio 1	rosso
6	½ bianco	blu medio	violetto
7	¾ bianco	blu chiaro	bianco 2
8	¼ bianco	marrone	–
9	½ bianco	arancio	–
10	grigio	grigio 2	–
11	¾ bianco	rosa	–
12	½ bianco	verde	–
13	¾ bianco	giallo	–
14	¾ bianco	acquamarina	–
15	bianco	bianco	–

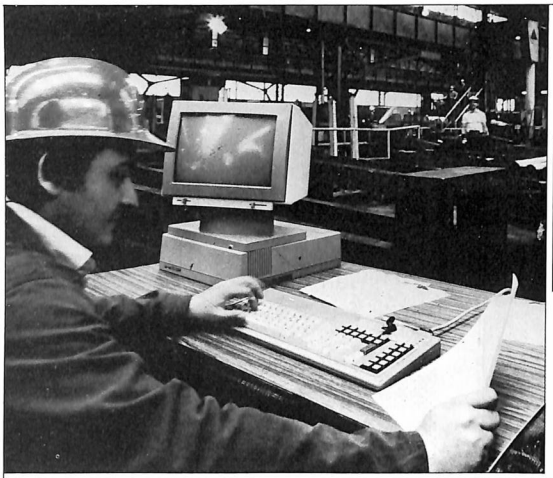
CONT comando

È una abbreviazione di "continua" e fa ripartire un programma dopo che è stato fermato da STOP, END o con il tasto BREAK o STOP. CONT non permette la ripartenza dopo un errore, un editing, un NEW.

Sono anche usate le abbreviazioni CON (Integer Basic) e C. (TRS-80 livello I).

COS (ANSI) funzione

La funzione COS fornisce il coseno dell'argomento tra parentesi. Per esempio, Y=COS(X) assegna alla variabile Y il valore del coseno di X, angolo espresso in radianti. ■



NOVITA'!

CORSO DI TECNICA DIGITALE

IL PROGRESSO DELL'ELETTRONICA PER IL TUO PROGRESSO PROFESSIONALE

Il minuscolo computer che regola una lavabiancheria, il videoterminale che permette di sorvegliare e di dirigere il montaggio robottizzato di un'automobile. Ecco solo due esempi dei progressi dell'elettronica. Progressi continui che richiedono la presenza di esperti in tecniche digitali nell'industria, nei servizi, nelle telecomunicazioni. Sarà proprio questo nuovo corso per corrispondenza Scuola Radio Elettra la base di partenza per inserirti in uno di questi settori o per migliorare il tuo attuale livello professionale. O, ancora, per entrare nell'affascinante mondo degli hobbisti della microelettronica. Con il metodo Scuola Radio Elettra, basato sulle esercitazioni pratiche, ti accorgerai di come studiare possa essere appassionante. Con le lezioni e i materiali che ti saranno forniti dalla Scuola e che resteranno di tua proprietà, realizzerai tutte le esperienze previste dal programma di studio e inoltre costruirai il DIGILAB, il tuo laboratorio digitale da tavolo per tanti diversi circuiti applicativi (termometro digitale, contasecondi elettronico, chiave elettronica...). Al termine del Corso un Attestato testimonierà la tua preparazione. Spedisci il tagliando. Riceverai, gratis e senza impegno, una completa documentazione a colori.

Il DIGILAB, il laboratorio digitale che rimarrà di tua proprietà.

Scuola Radio Elettra
Via Stellone 5/26C •10126 Torino
Da trent'anni insegna il lavoro.

PER CORTESIA, SCRIVERE IN STAMPATELLO

SCUOLA RADIO ELETTRA Via Stellone 5/26C 10126 TORINO

Contrassegnate con una crocetta la casella relativa al corso o ai corsi che vi interessano.

- ☐ Elettronica radio TV (novità)
- ☐ Radio stereo
- ☐ Televisione bianco e nero
- ☐ Televisione a colori
- ☐ Elettrotecnica
- ☐ Elettronica industriale
- ☐ Amplificazione stereo
- ☐ Alta fedeltà (novità)
- ☐ Fotografia
- ☐ Elettrauto
- ☐ Programmazione su elaboratori elettronici
- ☐ Disegnatore meccanico progettista
- ☐ Esperto commerciale
- ☐ Impiegata d'azienda
- ☐ Tecnico d'officina
- ☐ Motorista autoriparatore
- ☐ Assistente e disegnatore edile
- ☐ Lingue
- ☐ Sperimentatore elettronico
- ☐ Dattilografia (novità)
- ☐ Disegno e pittura (novità)
- ☐ Cosmesi (novità)
- ☐ Tecnica digitale (novità)

Nome
Cognome
Professione Età
Via
N.
Località
Cod. Post. Prov.
Motivo della richiesta: per hobby ☐ per professione o avvenire ☐

Tagliando da compilare, ritagliare e spedire in busta chiusa (o incollato su cartolina postale)

CANARD

Duplicazione abusiva, pericolo reale o timore infondato?

Conviene difendersi dalla pirateria software?

di *J. Grout*

Questo articolo, pur risentendo dell'ambiente in cui opera l'autore, quello del mercato americano, ci è sembrato interessante da proporre ai nostri lettori per la chiara analisi del problema della pirateria software.

Traduzione di *Flavio Santini*

Il timore della dilagante pirateria di software può essere in parte collegato ad una coscienza sporca.

Pensate al programmatore che lavora sodo per realizzare un programma che gli farà guadagnare un sacco di soldi. Dopo aver speso molto tempo per eliminare tutti gli errori, il programmatore inserisce una copia del prodotto con la sua documentazione in una busta e poi, mentre sta per spedirla, è assalito da un dubbio. "Non sarà che il produttore di software si impossessa del mio programma e mi lascia con un pugno di mosche?" E nella maggior parte dei casi è sufficiente esaminare una qualsiasi biblioteca di software per capire che la duplicazione abusiva di programmi si sta diffondendo.

Alla prima conferenza sulla protezione del software del Regno Unito, Julian Allason, un ex commerciante di software, nella sua nota di apertura ha affermato: "Se dovessi ricominciare la mia carriera nel mondo della programmazione, farei il pirata!". Molti sorridevano sornioni.

Tutti sanno che spesso è sorprendentemente facile copiare, e quindi trafugare, un programma per computer. Nella stessa conferenza, uno che si autoproclamava pirata ha chiesto a tutti quelli che non avevano mai fatto una copia abusiva di un programma di alzare la mano. In tutta la sala si sono alzate meno di cinque mani.

Se nel programma non è previsto un qualche tipo di protezione, la duplicazione è la cosa più facile di questo mondo. Se invece il programma è protetto, i maghi del software possono sbizzarrirsi nel tentare di copiarlo.

Attualmente si sta solo affrontando il problema di quali siano i canali di sviluppo della pirateria del software. Alcuni sostengono che la maggioranza dei furti di software avviene su scala amichevole: un dilettante copia un certo programma e lo passa ad un collega interessato. Una panoramica sugli utenti di PET del Regno Unito mostra che per ogni prodotto comperato, ce ne sono due e mezzo di duplicati. I dirigenti di industria forniscono spesso rapporti di uno a dieci tra i prodotti acquistati e quelli rubati.

Wayne Green, editore di *Microcomputing* e di altre riviste e fondatore della Instant Software, offre una taglia di 10 000 dollari per ogni informazione su ladri di software. Potrebbe essere abbastanza per convincerci a denunciare come pirata il nostro migliore amico. Soprattutto è un dato che sottolinea la serietà con cui viene trattato il problema.

Ma quanto grave è il problema della pirateria? È il caso che il programmatore, nel presentare il suo

primo gioiello, dubiti che la casa di software si duplichi il prodotto e lo diffonda a sua insaputa? La politica più redditizia nel mondo della programmazione è proprio la disonestà, come diceva scherzando Julian Allason?

Come ha detto in un suo articolo Christopher Kern, la rilevanza economica del furto di software tra gli hobbysti è piuttosto ridotta. Il vero problema sorge quando i pirati duplicano programmi commerciali e li diffondono su larga scala. Ma le grandi case, come l'Atari, con rendite in gran parte dovute ai giochi per computer e quindi più soggette alle attenzioni di trafugatori hobbysti, possono non essere d'accordo. Chi sono i pirati di cui parliamo?

Tre categorie di pirati

"Credo che i pirati di software si dividano in tre categorie" ha detto Ken Klein, dirigente della Stoneware Microcomputer Product, che ha prodotto il noto programma *DB Master* per la gestione di basi di dati.

"C'è chi duplica i vostri prodotti e li vende sottoprezzo. Credo che sia il caso più odioso perché disonesto in ogni senso del termine."

Klein ritiene che questo genere di pirata non sia un commerciante di software, ma piuttosto una persona estranea che, possedendo una buona preparazione tecnica, è in grado di accedere ai programmi.

"Sarebbero i primi nella lista delle persone da mettere in galera" ha detto Klein. "Un altro gruppo di pirati è probabilmente da ricercarsi nei club di programmatori. Non voglio parlar male di tutti i club. Alcuni svolgono funzioni utili e non ho proprio niente contro di essi. Ma esistono altri club che hanno come scopo principale l'acquisto di un unico programma perché tutti i membri possano utilizzarlo.

Per poter programmare a tempo pieno, bisogna poter guadagnare bene. Se ogni prodotto che un programmatore realizza viene trafugato, ci sono poche possibilità che questi venga dovutamente ricompensato. E se non può permettersi di vivere scrivendo programmi, deve dedicarsi a qualcos'altro. A lungo termine ciò significa produrre meno programmi. E sarà impossibile sviluppare soluzioni per i problemi reali della nostra società".

Klein spiegava che la produzione di un unico programma, *DB Master*, ha richiesto cinque anni di lavoro. Sosteneva che sarebbe estremamente duro rinunciare ai profitti ad esso relativi. "Il pirata di software non pensa alle paghe di 30 dipendenti ed ai costi di produzione", aggiungeva Klein.

Secondo lui la terza classe di pirati è costituita da coloro che duplicano i programmi per distribuirli agli amici.

"Di questi, tutto sommato, non è il caso di lamentarsi. Hanno delle ragioni comprensibili: vogliono solo farsi un piacere. Capita molto spesso. Se potessi introdurmi nei loro affari e dire: 'Guardate che non è onesto; non potete farlo!' probabilmente smetterebbero. Il loro scopo non è quello di guardagnare. Lo fanno per amicizia."

Quando ho chiesto a Klein se ha avuto a che fare con molti pirati di software, mi ha risposto che ne ha visti alcuni, ma non molti. Molti commercianti avvertono che le informazioni sulla diffusione del fenomeno della pirateria nell'industria dei microcomputer sono molto vaghe.

Quanti sono i pirati?

"Credo di non aver mai parlato con un pirata di software", afferma Doug Carlston della Broderbund Software, produttrice di *Apple Panic*. "Certamente la pirateria esiste, ma io sono l'ultima persona con cui un pirata si confiderebbe."

"Il concetto è confuso per molta gente. La dispersione delle informazioni, la loro distribuzione, sono considerate una buona cosa. Scoprirete che molti dei nostri autori, se fossero onesti, ammetterebbero di possedere, nelle proprie biblioteche, materiale di cui si sono appropriati senza pagare. Credo che sia così dappertutto." Carlston dubita che un pirata di software possa fare molti soldi.

"Come può vendere il suo prodotto? Sarebbe costretto a portarlo in giro chiuso nel baule della macchina mantenendo il segreto? Non credo che i commercianti facciano così. Conosco dei commercianti che distribuiscono copie di giochi come stimolo all'acquisto dei loro computer. Ma sono casi rari.

I commercianti si sforzano di vendere dei prodotti. Una volta si puntava sull'hardware, ora i più lungimiranti considerano come sempre più importante il prodotto software. Fino al punto che ciò favorisce il diffondersi della pirateria, cosa per loro non certo conveniente.

È possibile addirittura che siano i commercianti stessi a duplicare abusivamente dei programmi. Ho avuto sentore di grosse case dove la pirateria è giunta a livello endemico, dove è uno scherzo per un dipendente passare un programma ad altri 1500 colleghi. Conosco gente che compera un programma per le paghe e se ne serve in 195 posti diversi. Non comperano 195 copie. Ed il fenomeno è in ascesa."

Alla domanda se una casa editrice di software possa, di nascosto, copiare un programma e diffonderlo senza il consenso del programmatore, Carlston ha risposto con un deciso no. Ha precisato che ci sono rapporti molto stretti tra programmatori di computer e gruppi di utenti. Secondo Carlston, se un produttore di software si impossessasse del prodotto offertogli dall'autore, la voce si diffonderebbe immediatamente.

"Non credo che i venditori guadagnino su prodotti trafugati. Se qualcuno ne mettesse sul mercato uno, dicesse di averlo realizzato e ne proclamasse l'originalità, non riuscirebbe a diffonderlo. I commercianti al minuto ed i distributori sarebbero molto a disagio; ci arriverebbero dozzine di telefonate per chiederci che cosa sta succedendo."

A proposito dell'esitazione dei venditori a trattare copie abusive, Carlston ha detto: "Potenzialmente ci sono delle responsabilità. Quando l'Atari si mise a perseguire coloro che avevano duplicato il gioco del *Pac Man* non si limitò a far causa alle case di software, ma si rivolse a tutti gli anelli della catena: un dettagliante di Washington, delle case di vendita per posta, dei distributori. Volevano prenderli tutti! In linea di massima, non credo che la pirateria sia un problema. La sua incidenza non è rilevante. Se un programmatore non è tranquillo, stipuleremo con lui un tacito contratto. Cerchiamo di mantenere la massima sicurezza. Di notte chiudiamo i programmi in posti sicuri." Carlston ha anche diffidato i programmatori a spedire copie non protette agli amici. Con grande sorpresa dopo pochi giorni scoprirebbero che il loro programma è diffuso in tutto il paese.

Carlston ha raccomandato di non fidarsi delle parole. Egli è uno dei molti produttori di software che hanno espresso la preoccupazione che la pirateria danneggi la reputazione degli editori. Al di là delle difficoltà di distribuzione dei prodotti abusivi, il timore della pirateria potrebbe avere l'effetto di scoraggiare la presentazione di un programma ad una casa di software.

Il software è protetto dalla legge?

Sembra che nel mondo dei microcomputer sia diffusa l'opinione che il software è protetto dalla legge, ma non dal sistema legale.

I casi giudiziari relativi alla duplicazione abusiva di programmi per computer si sono rivelati quasi sempre sfavorevoli al pirata; tuttavia, la quantità di denaro necessaria per perseguire legalmente una trasgressione dei diritti di produzione rende il ricorso alla giustizia difficilmente proficuo. Troppo spesso il rimborso dei danni subiti è di gran lunga inferiore alle spese sostenute.

Irwin Taranto della Taranto and Associates Inc., produttore di software specializzato in prodotti commerciali e servizi di supporto, ha commentato il problema legale.

"Anche se le leggi fossero perfette, ci vorrebbero grosse spese per garantirsi la loro protezione. Potrebbero essere necessari 50000 dollari per risparmiarne 5. Il sistema legale non è in grado di risolvere il problema. Potrei trovarvi una decina di casi. Ne cito uno. Ho sostenuto una spesa di 20000 dollari per un processo che me ne ha rimborsati 1000. Che guadagno! Se fossi andato avanti mi sarebbe costato 50000 dollari; ho pensato bene di fermarmi a 20000. Sì, cerchiamo di difenderci con la legge; ci serviamo dei nostri avvocati per scrivere diffide. Ma in pratica l'impresa è ardua."

Taranto ha soggiunto, comunque, di non aver mai introdotto nei suoi programmi una qualche forma di protezione. Questa decisione è conseguenza soprattutto della sua filosofia commerciale.

"Io mi occupo di commercio. Vendo sistemi commerciali, non giochi. Io vendo il fatto che un cliente può chiamarmi in qualsiasi momento per avere il supporto di cui ha bisogno per continuare a lavorare. Indipendentemente da quale sia il suo problema, può venire qui o chiamarmi al telefono.

Non mi piace il fatto che qualcuno si duplichi un programma e lo dia ad un amico. Ma la cosa non mi preoccupa, perché non conviene né a chi lo offre, né a chi lo riceve. Spiego perché. Se mi accorgo che un mio cliente ha diffuso il mio

programma, non gli rivolgo più la parola. Così facendo egli rinuncia al mio sostegno, e perde il 90 per cento del valore del mio programma. In secondo luogo, neppure l'acquirente può chiamarmi per avere la mia consulenza. Se gli sono necessarie cinque, sei, dieci ore per risolvere da solo un problema, alla fine tutto risulterà più dispendioso che se avesse comperato regolarmente il programma.

Di solito i miei clienti non diffondono i programmi. Li mostreranno, ne consiglieranno l'acquisto, ma quando qualcuno è interessato, viene da me. Si può trafugare il mio software, non la mia consulenza.

Sono convinto che se adattate il vostro programma alle necessità del cliente, sarà suo interesse vedervi continuare proficuamente la vostra attività."

Come opporsi alla pirateria

L'arma di Taranto contro la duplicazione abusiva di software è la persuasione, a partire dal fatto che il produttore originale del programma può effettuare la manutenzione più efficientemente. Ma le case produttrici di software hanno preso in considerazione numerose altre contromisure.

Prescindendo dai risultati a cui può portare, il ricorso alla giustizia provoca dei disagi psicologici che sono un buon deterrente per i pirati. Nel caso di B. B. Roberts, per sua stessa ammissione pirata di software nella zona di Las Vegas, l'Atari ha addirittura assoldato un detective. Nel giro di tre giorni B.B. Roberts è stato scoperto ed indotto a scrivere un articolo dove spiegava tutti i fastidi che aveva avuto.

Altre possibilità sono l'uso di marchi di fabbrica più o meno visibili, ROM per sola esecuzione, e scatole nere elettroniche per escludere il pirata ficcanaso.

Per ora molti produttori sembrano credere più nell'ingegno tecnico dei pirati che nei sistemi di protezione che potrebbero creare.

Jim Ayers, consulente di programmazione e programmatore della Mill Valley, California, ha detto la sua a proposito dei metodi di protezione del software.

"Questi accorgimenti sono ridicoli", ha detto Ayers, ed ha citato il caso di un giovane valido programmatore, membro dell'Apple Core (un club di utenti di Apple di San Francisco). Ayers ha detto che il giovane aveva scritto un programma di copia a nibble

per inserirlo nella libreria dell'Apple Core. Per certa gente i programmi di copia a nibble sono soprattutto un mezzo per copiare abusivamente dei programmi protetti.

Ayers ha continuato spiegando che l'offerta del giovane programmatore riuscì quasi a creare una spaccatura nel club. Alcuni membri affermarono che, accettando il programma nella libreria, si sarebbe incentivata la pirateria. Quando terminò la discussione se accettare o respingere il programma, tutti i dirigenti del club minacciarono di dimettersi, e la sezione internazionale si stava preparando a disdire lo statuto.

"Poi una casa di software del luogo che cercava un programmatore si rivolse a me. Io feci il nome del ragazzo. Dissi loro che dovevano dargli la possibilità di fare qualcosa di impegnativo in modo che non gli rimase tempo per scrivere il pericoloso programma di copia." Ma non accettarono il consiglio.

L'idea di Ayers è che molta gente è in grado di superare le tecniche di protezione dei programmi e che quindi non vale la pena di svilupparle, spendendo altro denaro.

"Temo che la pirateria si stia sviluppando. Non ci sono metodi di protezione sicuri. Ma spero che il tessuto morale del paese non sia così logoro da non poter contare ancora sull'onestà."

Molti produttori di software sono d'accordo. Tra quelli che abbiamo intervistato, la maggior parte ha mostrato di fidarsi dell'onestà di base dei propri clienti. Nessuno ha riconosciuto di aver fatto affari grazie alla pirateria. I produttori di programmi per applicazioni commerciali si sono detti fiduciosi che gli uomini d'affari in media non siano interessati a programmi duplicati abusivamente. E che comperino il software dai venditori legittimi piuttosto che arrischiare i loro guardagni per il risparmio offerto da una copia abusiva.

Ayers ha detto anche di aver saputo di una casa di computer pirata di software. Tutti i suoi clienti avevano lo stesso numero di serie sul sistema operativo e ricevevano manuali di documentazione fotocopiati. Ayers ha riferito che ad un certo punto sembrava che il pirata avesse venduto lo stesso computer due volte. Il suo errore si rifletté su di lui. "Rimase tagliato fuori", ha affermato Ayers.

Come può difendersi un programmatore?

Un avvocato californiano esperto di leggi sulla tutela del software (che non vuole che il suo nome venga riportato) ha sostenuto che gli autori di programmi possono in qualche modo proteggersi contro i produttori senza scrupoli.

"Il software è protetto come tutto il resto. Il programmatore è tutelato quanto l'autore di un libro. Solo la legge può impedire che un programma venga duplicato, così

come è l'unico mezzo per impedire che si distribuiscano le fotocopie di un libro. È esattamente lo stesso problema.

Le difficoltà dei programmatori derivano da due considerazioni. Da un lato, le autorità non applicano la legge e il programmatore deve arrangiarsi da solo.

Dall'altro, i pirati ci sono, li defraudano delle loro creazioni e li lasciano spesso senza possibilità di reagire alla perdita di guadagno subita.

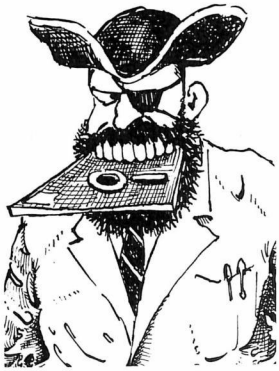
Prima dell'atto sui diritti d'autore del 1978, gli editori di piccole riviste specialistiche venivano in effetti danneggiati dall'uso indiscriminato delle fotocopie. La stessa cosa succede adesso con il software. Non si può evitare la duplicazione di software se la codifica non viene effettuata in modo che nessuno possa accedervi, nel qual caso si dovrebbe discutere se il prodotto è tutelabile, perché nessun altro può controllare se è stato violato.

La legge di tutela è rimasta così per anni. Non c'era modo di impedire ai pianisti dei vari locali di suonare la musica di qualcun altro, e quindi di violare i diritti d'autore. Poi una organizzazione chiamata ASCAP (*American Society of Composers, Authors and Publishers*) si occupò del loro caso, con dipendenti pagati che controllavano che ogni locale avesse pagato la apposita tassa annuale."

L'avvocato ha auspicato che anche nel commercio del software si crei una situazione analoga. Si potrebbero controllare i più noti distributori, commercianti e club per salvaguardare i diritti d'autore.

"Questo è il momento propizio per farlo, perché il settore è ancora all'inizio del suo sviluppo. Se fosse giunto ad uno stadio più avanzato, tutti riterrebbero perfettamente legale duplicare il software. Credo che ora i membri dei club siano attenti al fenomeno, sentono che alcune cose non sono proprio oneste. Alcuni controllori potrebbero far loro visita, scoprendo del software copiato, e quindi avvertirli con una lettera in cui si dice: "Ehi, siete stati scoperti!". L'industria privata potrebbe così difendersi.

Per assicurarsi la protezione da un punto di vista legale, tutto ciò che il programmatore ha da fare è registrare il programma all'ufficio per la tutela dei diritti d'autore. La formalità è semplice e veloce, e la tassa relativa è di 10 dollari. Il richiedente deve allegare due copie del listato di programma (alcuni dicono che basta un listato e le ultime dieci pagine del codice oggetto) provando così *prima facie* che il



programma è suo. Se è in grado di dimostrare che un altro programma ha lo stesso listato, il tribunale prenderà provvedimenti.

A Tujunga, California, si è formato un nuovo gruppo per la lotta contro la pirateria del software. La ASP (*Association for Software Protection*) è una associazione senza scopo di lucro il cui fine principale consiste nell'istruire gli utenti finali sugli aspetti legali del commercio di software e di assicurare il rispetto di alcune norme di comportamento all'interno dell'industria.

Robin Robinson, presidente dell'ASP, ha stimato che solo per il cinque per cento i furti di software sono volutamente disonesti. A suo parere la maggior parte delle violazioni di proprietà è dovuta all'ignoranza sull'origine del software. I membri dell'ASP sperano di riuscire a fugare la falsa immagine diffusa sugli accordi per la proprietà del software.

Robinson suggerisce che sia l'ASP a introdurre presso i suoi membri la pratica del rispetto del software e che li aiuti a creare un mercato chiuso a coloro che non sono in regola.

Come consiglio ai programmatori inesperti, Robinson ha proposto loro di chiedere ai produttori un contratto esplicito, prima di consegnare qualsiasi programma. Poi, se il contratto verrà violato, il programmatore consulterà un avvocato per far valere i propri diritti.

Dovrebbero esserci anche alcune disposizioni per i produttori responsabili della distribuzione del programma. Robinson ha asserito che, anche se il ricorso ad un avvocato può sembrare una precauzione piuttosto costosa, può garantire la sopravvivenza ai programmatori.

Ma sotto sotto siete anche voi pirati?

La stima di Robinson che solo il cinque per cento dei furti di programmi siano fatti su commissione impone di chiedersi se la pirateria è effettivamente un problema così serio, come qualcuno sostiene. È probabile che le stime crescano e diminuiscano proporzionalmente a quanto chi le fa teme la pirateria. Gli autori ed i produttori di programmi affermano senza esitazione che essa esiste, ma pochi ritengono che possa riguardare un giro d'affari considerevole.

Tutto sommato i traffici che avvengono tra gli hobbysti o a livello di amicizia non riguardano più di tanto i produttori di software.

Quelli con cui ho parlato sono sinceramente propensi a rispettare il rapporto con gli aspiranti programmatori, forse perché capiscono i loro dubbi pensando a quelle difficoltà che erano le loro fino a poco tempo fa. Un consiglio che vale sempre è che il programmatore dovrebbe rivolgersi solo alle case di software che hanno una buona reputazione. Anche se i produttori in generale dicono che i programmatori non hanno grossi motivi per preoccuparsi. ■

Contraerea: le capacità grafiche dello ZX Spectrum

di *Marcello Spero*

Una delle caratteristiche che rendono lo ZX Spectrum estremamente attraente fra le macchine della sua categoria è senz'altro la possibilità di una gestione del video fra le più complete.

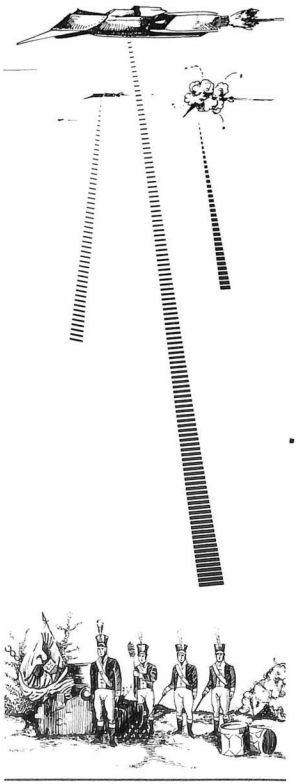
Quello che vedremo è un uso "poco serio", ma non per questo meno interessante, di alcune di queste potenzialità.

Il programma visualizza un aereo che vola al di sopra di una batteria contraerea, e può sganciare bombe per cercare di distruggerne i cannoni; le bombe vengono sganciate premendo il tasto M.

Il fuoco della contraerea si riduce tanto più quanti più cannoni vengono distrutti, ma attenti: il tiro è molto accurato, e dovete quindi distruggere almeno un cannone molto rapidamente, se volete avere qualche possibilità di sopravvivenza; oltretutto non potete sganciare una nuova bomba finché la precedente non ha toccato terra.

Il programma principale arriva fino alla linea 299, e le linee REM chiariscono la funzione delle subroutine.

La prima sezione, fino alla linea 100, oltre che richiamare le istruzioni e preparare lo schermo, provvede alla definizione dei caratteri grafici speciali. Questo avviene con la linea 5, che richiama l'apposita subroutine; i dati sono alle linee da 1100 a 1120, in forma decimale.



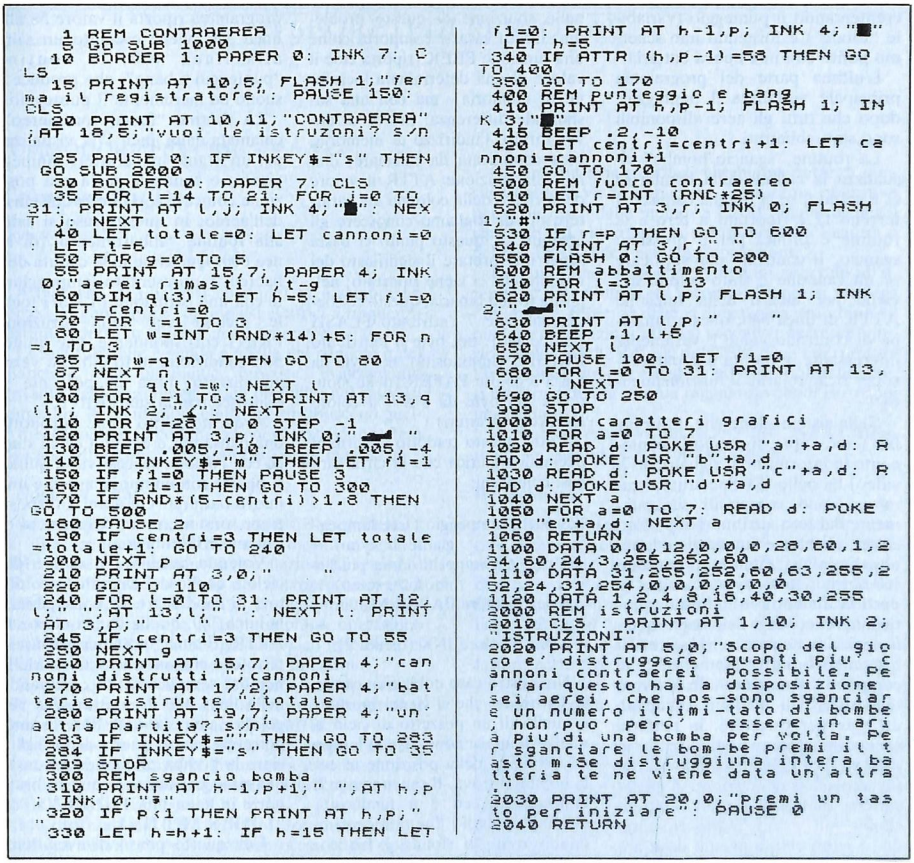
Ricordo che i dati relativi alla grafica definibile sono otto numeri per ogni carattere, la cui forma binaria rappresenta con 1 i punti del colore INK, e con 0 quelli del colore PAPER di ognuna delle otto righe che compongono il carattere; ad esempio, una riga tutta di colore PAPER, cioè del colore dello sfondo, sarà rappresentata dal numero binario 00000000 cioè zero, mentre una riga tutta di colore INK, cioè del colore dei caratteri, sarà rappresentata dal numero binario 11111111 cioè 256.

Questi dati sono quindi letti da una istruzione READ, e immessi con una istruzione POKE nel rispettivo carattere.

La sagoma dell'aereo è ottenuta con i caratteri grafici "A", "B" e "C"; la bomba è il carattere "D", mentre il cannone è la "E" grafica.

Il programma, essendo stato listato dopo essere stato fatto girare, riporta al posto delle lettere grafiche il carattere già definito, come apparirà anche a voi dopo la prima esecuzione, a patto che abbiate collocato le lettere giuste al posto giusto.

Le linee 30 e 35 predispongono lo schermo al colore bianco per il cielo, verde per il terreno e nero per la cornice (BORDER); l'istruzione CLS, dopo aver cambiato il colore dello sfondo, è, come probabilmente sapete, molto importante: infatti, se lo schermo non



```
  1 REM CONTRAEREA
  5 GO SUB 1000
 10 BORDER 1: PAPER 0: INK 7: C
LS
 15 PRINT AT 10,5; FLASH 1;"fer
ma il registratore": PAUSE 150:
CLS
 20 PRINT AT 10,11;"CONTRAEREA"
;AT 18,5;"vuoi le istruzoni? s/n
"
 25 PAUSE 0: IF INKEY$="s" THEN
 GO SUB 2000
 30 BORDER 0: PAPER 7: CLS
 35 FOR l=14 TO 21: FOR j=0 TO
31: PRINT AT l,j; INK 4;"■": NEX
T j: NEXT l
 40 LET totale=0: LET cannoni=0
: LET t=6
 50 FOR g=0 TO 5
 55 PRINT AT 15,7; PAPER 4; INK
 0;"aerei rimasti ";t-g
 60 DIM q(3): LET h=5: LET f1=0
: LET centri=0
 70 FOR l=1 TO 3
 80 LET w=INT (RND*14)+5: FOR n
=1 TO 3
 85 IF w=q(n) THEN GO TO 80
 87 NEXT n
 90 LET q(l)=w: NEXT l
100 FOR l=1 TO 3: PRINT AT 13,q
(l); INK 2;"▄": NEXT l
110 FOR p=28 TO 0 STEP -1
120 PRINT AT 3,p; INK 0;"▄▄ "
130 BEEP .005,-10: BEEP .005,-4
140 IF INKEY$="m" THEN LET f1=1
150 IF f1=0 THEN PAUSE 3
160 IF f1=1 THEN GO TO 300
170 IF RND*(5-centri)>1.8 THEN
GO TO 500
180 PAUSE 2
190 IF centri=3 THEN LET totale
=totale+1: GO TO 240
200 NEXT p
210 PRINT AT 3,0;"     "
220 GO TO 110
240 FOR l=0 TO 31: PRINT AT 12,
l;" ";AT 13,l;" ": NEXT l: PRINT
AT 3,p;"  "
245 IF centri=3 THEN GO TO 55
250 NEXT g
260 PRINT AT 15,7; PAPER 4;"can
noni distrutti ";cannoni
270 PRINT AT 17,2; PAPER 4;"bat
terie distrutte ";totale
280 PRINT AT 19,7; PAPER 4;"un'
altra partita? s/n"
283 IF INKEY$="" THEN GO TO 283
284 IF INKEY$="s" THEN GO TO 35
299 STOP
300 REM sgancio bomba
310 PRINT AT h-1,p+1;" ";AT h,p
; INK 0;"▄"
320 IF p<1 THEN PRINT AT h,p;"
 "
330 LET h=h+1: IF h=15 THEN LET
 f1=0: PRINT AT h-1,p; INK 4;"■"
: LET h=5
340 IF ATTR (h,p-1)=58 THEN GO
TO 400
350 GO TO 170
400 REM punteggio e bang
410 PRINT AT h,p-1; FLASH 1; IN
K 3;"■"
415 BEEP .2,-10
420 LET centri=centri+1: LET ca
nnoni=cannoni+1
450 GO TO 170
500 REM fuoco contraereo
510 LET r=INT (RND*25)
520 PRINT AT 3,r; INK 0; FLASH
1;"#"
530 IF r=p THEN GO TO 600
540 PRINT AT 3,r;" "
550 FLASH 0: GO TO 200
600 REM abbattimento
610 FOR l=3 TO 13
620 PRINT AT l,p; FLASH 1; INK
2;"▄▄"
630 PRINT AT l,p;"     "
640 BEEP .1,l+5
650 NEXT l
670 PAUSE 100: LET f1=0
680 FOR l=0 TO 31: PRINT AT 13,
l;" ": NEXT l
690 GO TO 250
999 STOP
1000 REM caratteri grafici
1010 FOR a=0 TO 7
1020 READ d: POKE USR "a"+a,d: R
EAD d: POKE USR "b"+a,d
1030 READ d: POKE USR "c"+a,d: R
EAD d: POKE USR "d"+a,d
1040 NEXT a
1050 FOR a=0 TO 7: READ d: POKE
USR "e"+a,d: NEXT a
1060 RETURN
1100 DATA 0,0,12,0,0,0,28,60,1,2
24,60,24,3,255,252,60
1110 DATA 31,255,252,60,63,255,2
52,24,31,254,0,0,0,0,0,0
1120 DATA 1,2,4,8,16,40,30,255
2000 REM istruzioni
2010 CLS : PRINT AT 1,10; INK 2;
 "ISTRUZIONI"
2020 PRINT AT 5,0;"Scopo del gio
co e' distruggere quanti piu' c
annoni contraerei possibile. Pe
r far questo hai a disposizione
sei aerei, che pos-sono sganciar
e un numero illimi-tato di bombe
; non puo' pero' essere in ari
a piu' di una bomba per volta. Pe
r sganciare le bom-be premi il t
asto m.Se distruggiuna intera ba
tteria te ne viene data un'altra
."
2030 PRINT AT 20,0;"premi un tas
to per iniziare": PAUSE 0
2040 RETURN
```

viene cancellato, permane il vecchio colore di sfondo, e il nuovo colore compare solo quando viene visualizzato qualcosa, creando strani e colorati (quanto indesiderati) effetti.

Le linee 40 e 60 inizializzano le variabili; le linee da 70 a 90 determinano le posizioni dei cannoni, mentre la linea 85 controlla che queste siano diverse l'una dall'altra; la linea 100, infine, visualizza i cannoni. In questa linea dovete sostituire alla sagoma del cannone il carattere grafico "E", per i motivi detti prima.

Con il ciclo compreso fra le linee 110 e 200 sono controllati il movimento dell'aereo e la sua visualizzazione; anche qui la sagoma va sostituita con i rispettivi caratteri grafici, che in questo caso sono "A", "B" e "C", seguiti da uno spazio che ha la funzione di cancellare l'ultimo carattere man mano che l'aereo avanza. In linea 140 avviene il test per controllare se il tasto M è premuto; in questo caso il flag f1 viene settato (cioè posto uguale ad uno); ciò fa sì che il programma salti alla routine "sgancio bomba"; se questo non avviene, viene introdotta una istruzione PAUSE per mantenere uguale la velocità di esecuzione.

Il fuoco della contraerea è controllato dalla linea 170; i valori possono essere variati per rendere il gioco più facile o più difficile; quelli dati mi sembrano un buon compromesso; la pausa in linea 180 ha la stessa funzione della precedente, cioè uniformare la velocità.

La variabile "centri" in linea 190 controlla se sono stati distrutti tutti

e tre i cannoni, in questo caso incrementando il punteggio (variabile "totale") e fornendo uno schermo pulito con una nuova batteria.

L'ultima parte del programma principale visualizza il punteggio, dopo che tutti gli aerei disponibili sono stati abbattuti.

La routine "sgancio bomba" visualizza la caduta della bomba se f1 è settato; se la bomba colpisce il terreno f1 è riportato a zero e la routine è pronta per il prossimo sgancio. Il controllo per verificare se un cannone è stato colpito avviene per mezzo della funzione ATTR di linea 340; questa funzione di controllo video è veramente interessante, e merita una parentesi per ricapitolarne il funzionamento.

Nella sistemazione che il display file (cioè l'area di memoria contenente le informazioni riguardanti il video) ha nello ZX Spectrum, i caratteri sono conservati separatamente dai loro attributi (cioè i loro colori, eventuale lampeggio e luminosità); infatti, mentre i primi sono suddivisi in otto byte ciascuno (e posti in memoria in un ordine particolare che, lungi dall'essere quello logico in sequenza, rispecchia il movimento del "pennello" che compone l'immagine sullo schermo televisivo), gli attributi occupano un byte per carattere, in perfetta sequenza, e sono posti in coda al display file. Questo comporta che un controllo sugli attributi di un carattere, piuttosto che sul carattere stesso, può essere molto più rapido, a patto che sappiamo dove andare a cercare gli attributi di quel particolare carattere. La funzione ATTR ci viene incontro proprio nella soluzione di questo problema; infatti essa si comporta come una funzione PEEK (riporta cioè il valore di una determinata locazione di memoria), ma con una sostanziale differenza: invece di dover fornire l'indirizzo di memoria, cosa complicata da calcolare, diamo alla funzione ATTR il valore della riga e della colonna del carattere di cui vogliamo conoscere gli attributi. A questo punto ci basta saper interpretare il significato del numero che ci viene riportato; nella sua forma binaria, questo contiene: nel byte 7 l'attributo FLASH (lampeggio), nel byte 6 l'attributo BRIGHT (luminosità), nei byte da 5 a 3 il colore PAPER (dello sfondo) e nei byte da 2 a 0 il colore INK (dei caratteri).

Tutto questo tradotto in termini decimali significa che la cifra sarà così composta:

128 per il lampeggio (1 se lampeggiante, 0 se no) +
64 per la luminosità (1 se più luminoso, 0 se no) +
8 per il colore PAPER (da 0 a 7) +
1 per il colore INK (da 0 a 7).

Nel nostro caso dobbiamo capire se l'attributo che ci viene riportato è quello di un pezzetto di cielo o quello di un cannone, dando la riga e la colonna della posizione in cui la bomba si trova. Il cannone non è lampeggiante ed è a luminosità normale, quindi i due primi casi ci danno zero: lo sfondo è bianco, cioè 7, che per 8 dà 56; il cannone stesso è rosso, cioè 2, che per 1 dà 2; totale 56+2=58. Infatti, se il programma riporta il valore 58 alla linea 340, viene eseguito un salto alla linea 400, la routine "punteggio e bang", che produce il suono ed incrementa il punteggio.

La routine "fuoco contraereo", chiamata dalla linea 170, visualizza per un istante un carattere lampeggiante, e controlla se la sua posizione coincide con quella del muso dell'aereo; in questo caso si salta alla routine "abbattimento" di linea 600. Qui avviene la caduta dell'aereo, rappresentato lampeggiante con una istruzione FLASH locale, inclusa cioè nella istruzione PRINT corrispondente, che quindi non condiziona tutto ciò che verrà visualizzato d'ora in poi, ma si esaurisce con la sua PRINT.

A questo punto viene introdotta una pausa per creare un certo stacco con l'azione successiva, e quindi si riparte con un nuovo aereo e una nuova batteria. Gli aerei a disposizione sono sei, mentre il numero di cannoni è illimitato.

Volendo cambiare il colore del cielo o quello dei cannoni, ricordatevi di modificare di conseguenza la linea 340, quella che contiene la ATTR. È anche possibile variare il tasto che comanda lo sgancio della bomba, semplicemente cambiando nella linea 140 il carattere fra virgolette; una variante interessante potrebbe essere quella di far sganciare la bomba premendo un tasto qualsiasi: per ottenere questo basta porre in linea 140 IF INKEY$ <> "" THEN LET f1=1.

Con questo penso di aver detto tutto: aspetto i vostri consigli e miglioramenti. Buon divertimento! ■

COMMODORE PET/CBM

Tempus fugit...

Ettore M. Albani

All'interno di qualsiasi computer vi è sempre un "orologio generatore" di impulsi elettrici a frequenza costante, detto in inglese *clock generator*.

La sua funzione è simile a quella del pendolo o del bilanciere, che, oscillando, provoca l'avanzamento costante degli ingranaggi in un orologio: nel caso del computer, le oscillazioni elettriche sostituiscono quelle meccaniche e le commutazioni di stato di una rete logica sostituiscono l'avanzamento degli ingranaggi.

Il clock del CBM

Su un personal computer come il CBM, il "tic-tac" ha una frequenza relativamente bassa: appena 1 MHz, ovvero un milione di oscillazioni al secondo! Ciò vuol dire che un ciclo del microprocessore del CBM (il 6502) ha la durata di un milionesimo di secondo; se si pensa che la più complessa operazione gestibile con il 6502 abbisogna di 7 cicli macchina, ci si rende conto dell'enorme velocità di calcolo.

Eppure questa velocità di funzionamento è tra le più basse e non è infrequente trovare nei personal clock di 4 MHz o addirittura di 10 MHz.

Nel futuro la velocità sarà ancora maggiore: basti pensare che con le nuove giunzioni a superconduttori si sfiora la frequenza fantastica di 1 GHz (un miliardo di cicli al secondo)!

Ma ritorniamo al 6502 del CBM; fortunatamente la struttura interna di questo microprocessore del tipo a *pipeline* compensa in gran parte la bassa frequenza di clock. Volendo fare un paragone con lo Z80, quest'ultimo, a parità di velocità di esecuzione, dovrebbe avere un clock di frequenza circa doppia.

All'utente medio del CBM queste notizie possono sembrare pure dissertazioni accademiche; in realtà il tempo necessario ad eseguire un'istruzione è importantissimo nel caso di attività ricorrenti: se qualcuno ha provato a far eseguire un sort di un elenco di nomi con l'interprete Basic del CBM, avrà notato che la velocità di esecuzione dipende essenzialmente dal tipo di algoritmo scelto, dal tipo di istruzioni usate e dal numero di nomi da ordinare.

Il clock del CBM in un secondo oscilla un milione di volte ed il 6502, pensando ad una media di 4 cicli per istruzione, ne può eseguire circa 250000 al secondo.

Occorre tuttavia ricordare che buona parte di queste istruzioni viene impiegata dal sistema operativo per il rinfresco della memoria RAM dinamica, per il rinfresco del video, per la scansione della tastiera e del bus IEEE-488 e per altre operazioni.

Restano dunque "poche" istruzioni riservate all'utente ed eseguibili in un secondo. Se poi si impiega l'interprete Basic anziché il linguaggio macchina, il numero di istruzioni utili (al secondo) decresce ulteriormente, poiché il Basic del CBM deve interpretare l'istruzione prima di eseguirla ed essa, normalmente, si compone di una somma ragguardevole di istruzioni macchina.

A titolo di esempio, un ciclo FOR...NEXT effettua in un secondo circa 1000 iterazioni.

Il programma

Mediante un semplice programma è possibile misurare il tempo necessario ad eseguire qualsiasi istruzione del Basic. Il programma 1 utilizza la variabile numerica riservata TI, che è un contatore interno del CBM; il suo contenuto, infatti, viene incrementato di una unità (*jiffy*) ogni sessantesimo di secondo. Si noti che le linee da 100 fino a 140 servono a calcolare il fattore correttivo K (tempo necessario ad effettuare 1000 interazioni) da sottrarre al tempo totale di esecuzione calcolato dalla linea 210 fino alla 250.

```
100 REM * CALCOLO FATTORE CORRETTIVO
110 T=TI
120 FOR I=1 TO 1000
130 NEXT
140 K=TI-T
200 REM * CALCOLO TEMPO TOTALE
210 T=TI
220 FOR I=1 TO 1000
230 REM * INSERIRE QUI IL COMANDO
240 NEXT
250 T=TI-T
300 REM * RISULTATO IN MSEC
310 PRINT (T-K)/60
```

Listato 1.

Le prove sono state effettuate con un CBM 4032; alcuni dei risultati appaiono in fig. 1, altri li potrete ricavare inserendo l'istruzione nella linea 230.

Naturalmente questi risultati, pur essendo validi da un punto di vista formale, possono essere considerati solo a livello qualitativo perché altri fattori possono influenzare il tempo medio impiegato ad eseguire l'istruzione.

Ad esempio, ogni volta che il Basic utilizza una

COMMODORE PET/CBM

variabile, viene effettuata la scansione dei nomi di tutte le variabili già eventualmente create: questo tempo dipende, ovviamente, dal numero di confronti da effettuare ovvero dalla posizione che il nome della variabile assume nel "vocabolario" del CBM.

Un altro esempio potrebbe essere quello della famigerata subroutine *garbage collection*, che alla lettera significa "raccolta dei rifiuti". L'utente del CBM avrà notato, operando con ampi vettori di variabili, che ad un certo punto la macchina sembra bloccarsi per un tempo più o meno lungo: la causa è la suddetta subroutine, attivata dalla mancanza di spazio in memoria.

Assegn.	Comando	msec
	PRINT	18.7
	PRINT 123.4	34.5
X=123.4	PRINT X	27.8
"	PRINT X;	11.2
	PRINT "123.4"	20.6
X$="123.4"	PRINT X$	20.1
"	PRINT X$;	2.6
"	PRINT X$+X$	22.1
	PRINT TAB(1)	2.0
	PRINT SPC(1)	2.2
	PRINT PEEK(100)	23.6
	POKE 2000,31	7.0
	READ	6.2
	X%=100	3.8
	X=100	3.5
	X$="100"	1.4
	XX$="100"	1.5

Figura 1.

Nel periodo di tempo durante il quale il computer sembra essere inattivo, viene effettuata una "raccolta dei rifiuti" rappresentati dalle vecchie stringhe alfanumeriche associate in precedenza ad una variabile.

Infatti, il nuovo contenuto di una variabile stringa non è memorizzato nelle stesse locazioni del precedente, ma in altre locazioni libere. La subroutine di *garbage collection* individua nella RAM tutti i vecchi contenuti delle variabili in memoria, li cancella e compatta i nuovi contenuti, aggiornando i puntatori delle variabili. Il sistema operativo esegue questa subroutine non solo, come si accennava, quando manca spazio in memoria, ma anche quando interpreta il comando FRE(). Nel Basic 4.0 del CBM 4032/8032, questo problema è stato parzialmente risolto con l'aggiunta di particolari subroutine che velocizzano questa operazione.

Consigli pratici

Ogniqualvolta avrete problemi di velocità di esecuzione, tenete presente questa serie di consigli.

1. Inizializzate tutte le variabili usate dal programma all'inizio dello stesso; cioè, ponete a zero (o ad altro numero) tutte le variabili numeriche, ed a stringa nulla tutte le variabili alfanumeriche. Questa operazione serve a creare il "vocabolario" delle variabili: perciò inizializzate prima le variabili più frequentemente adoperate. Ricordate di dimensionare i vettori sempre dopo l'inizializzazione delle variabili semplici, poiché la creazione di anche una sola variabile semplice dopo aver dimensionato un vettore provoca uno spostamento nell'ambito del "vocabolario" dell'intero vettore verso la parte alta della memoria.

2. Utilizzate il più possibile delle variabili al posto di costanti: se per esempio vi capita di dover stampare sul device 5, inizializzate due variabili (P e D) indicanti rispettivamente il file logico ed il device fisico (P=128, D=5). Un ciclo FOR...NEXT sarà più veloce se vengono utilizzate delle variabili al posto di costanti.

3. A proposito del ciclo FOR...NEXT, cercate di terminare in ogni caso il ciclo, per evitare problemi di riempimento dello *stack* (catasta operativa); ciò vuol dire che, in caso di chiusura anticipata del ciclo, sarà utile porre la variabile di ciclo uguale all'estremo superiore dell'intervallo:

```
100 REM * CICLO
110 FOR I=1 TO 20
120 X=I*(I+2)
130 IF X>100 THEN I=20: GOTO 150
140 PRINT X
150 NEXT
```

Sempre a proposito del ciclo FOR...NEXT, ricordate di usare il meno possibile il nome delle variabili dopo il NEXT: ciò velocizzerà il ciclo.

4. Le variabili numeriche semplici si dividono in intere ed a virgola mobile. La loro occupazione di memoria è la medesima, ma la velocità di esecuzione di comandi che utilizzano le prime è superiore. Un forte risparmio di memoria ed un aumento di velocità si ha, invece, nel caso di vettori numerici. Infatti, un vettore numerico intero occupa 2 byte per elemento, mentre quello a virgola mobile ne occupa 5.

5. Nel caso di stringhe, ricordate che la variabile semplice occupa 7 byte più la lunghezza del suo contenuto, mentre il vettore stringa occupa per ogni elemento 3 byte più la lunghezza del relativo contenuto. ■

SINCLAIR ZX80/ZX81

READ, DATA e RESTORE e l'uso del registratore

Enrico Ferreguti

Il più delle volte durante la conversione nel Basic del Sinclair di programmi di altri calcolatori ci si inchioda in istruzioni sconosciute, o perlomeno che non si ha l'idea di come tradurre.

Nel 90% dei casi l'inghippo è costituito dalle istruzioni READ, DATA e RESTORE, che non hanno corrispondenti nel linguaggio dello ZX.

La funzione di queste istruzioni è la registrazione di blocchi di dati nel programma con accesso sequenziale.

DATA segnala che dopo questa istruzione si trovano i dati separati dalla virgola.

READ invece legge sequenzialmente i dati all'interno delle DATA (numerici o alfanumerici), cioè ad ogni lettura il puntatore si sposta sul prossimo dato sulla riga, o, se il dato era l'ultimo, sulla prossima che ne contiene.

RESTORE infine riposiziona il puntatore dei dati sulla prima DATA del programma. In effetti in alcuni Basic si può posizionare il puntatore su una riga qualsiasi a discrezione del programmatore; purtroppo questa possibilità non viene inclusa nella nostra simulazione.

Con lo ZX queste istruzioni sono simulate nella seguente maniera (si veda il listato).

DATA: i dati sono contenuti in REM all'inizio del programma e non disseminati per tutto il listato.

READ: per chiamare questa routine bisogna usare l'istruzione GOSUB READ o GOSUB 9900; al ritorno dalla chiamata il dato letto sarà contenuto nella stringa R$.

Quindi se troviamo READ C$ con lo ZX dovremo scrivere:

```
GOSUB READ
LET C$=R$
```

Se troviamo invece READ C$,B$ allora scriveremo:

```
GOSUB READ
LET C$=R$
GOSUB READ
LET B$=R$
```

RESTORE: si batte semplicemene GOSUB RESTORE o GOSUB 9950 e il puntatore si riposizionerà sulla prima REM.

```
   1 REM ROMA.MILANO.VENEZIA.TOR
INO.NAPOLI.PALERMO.GENOVA.BOLOGN
A.
   2 REM 1267.352.287352.2615.
 100 GOSUB 9950
 110 FOR K=1 TO 6
 120 GOSUB READ
 130 PRINT R$
 140 NEXT K
 145 LET S=0
 146 PRINT
 150 FOR K=1 TO 4
 160 GOSUB READ
 170 LET V=VAL R$
 180 PRINT V;"+"
 190 LET S=S+V
 200 NEXT K
 210 PRINT "---------"
 220 PRINT "=";S
 230 STOP
9900 REM READ
9905 LET R$=""
9910 LET BYTE=BYTE+1
9915 IF PEEK (BYTE)=27 THEN RETU
RN
9920 IF PEEK (BYTE)=118 THEN LET
 BYTE=BYTE+5
9925 LET R$=R$+CHR$ (PEEK (BYTE)
)
9930 GOTO 9910
9950 REM RESTORE
9955 LET READ=9900
9960 LET BYTE=16513
9965 LET RESTORE=9950
9970 RETURN
```

Importante

- Le REM vanno sistemate *tutte* all'inizio e, come si vede dal listato, i dati possono trovarsi su varie REM di lunghezza variabile.
- Per usare queste routine bisogna prima inizializzarle all'uso. Per fare ciò si sistemi un GOSUB 9950 all'inizio del programma (dopo le REM), che nel nostro caso si trova in riga 100.
- Bisogna sempre far seguire un punto ai dati, anche quando ci si trova a fine riga.

Dopo aver ben esaminato il listato di esempio non dovrebbe risultare difficile tradurre un programma con le DATA e inserire molti dati ed informazioni nei nostri programmi senza bisogno di fare i salti mortali per caricarli.

Ricordiamo che ciò che ci permette di utilizzare questo metodo è proprio l'originalità del sistema ZX81, in quanto l'inizio della zona utente, in quasi tutti i calcolatori, se ne va in giro per la memoria. Nel nostro caso, invece, resta ben fissa a partire dalla locazione 16509 e risulta quindi facile esplorarla e pokarci dentro quello che vogliamo.

SINCLAIR ZX80/ZX81

Consigli sull'uso del registratore

1. Usare solo nastri di buona qualità.
2. Non serve acquistare nastri da 60, 90 o 120 minuti, non li riempireste mai, usate nastri da 46 minuti o meno.
3. Cercate di mantenere il massimo ordine, catalogando accuratamente tutti i programmi contenuti nella cassetta. Non sarebbe una cattiva abitudine far iniziare il nastro con un piccolo archivio in cui registrare il nome dei programmi, l'occupazione di memoria e altri dati che li riguardano.
4. Il tono dovrà sempre essere al minimo nei registratori predisposti per la sua regolazione.
5. Non alimentare i registratori con pile ma possibilmente con la corrente di rete. Man mano che le pile si scaricano, diventano possibili sbalzi di velocità ai motori di trazione della cassetta.
6. Per il trasferimento di dati tra registratore e computer usare solo gli speciali cavi in dotazione, o i cavi schermati a bassa impedenza usati per il collegamento di apparecchi HiFi.
7. Non è sempre bene utilizzare piastre stereo. È preferibile usare piccoli registratori, che tra l'altro migliorano la portabilità del sistema e ne riducono l'ingombro.
8. Se è possibile, usare sistemi di soppressione del fruscio (Dolby, HighCom, ecc.), che tendono ad esaltare le alte frequenze, e cioè proprio quelle destinate al flusso di impulsi costituenti il programma.
9. In registrazione, il volume deve essere sempre tenuto basso, mentre durante il caricamento dovrà essere al massimo. Certe volte capita di non poter caricare un programma fin dall'inizio; ciò è confermato da strane inchiodature e NEW involontari del sistema. Talvolta è sufficiente solo abbassare il volume quel tanto che basta al computer per ricevere lo start e capire il nome del programma. Quando invece il programma caricato è zeppo di imprecisioni e con caratteri senza senso sparsi per tutto il listato, allora la causa è da imputare ad una cattiva registrazione.
10. Se in un programma non è importante il salvataggio dei dati in esso contenuti (p.es. è un gioco o contiene formule) conviene cancellarli con un CLEAR. In questo modo si riduce il tempo di salvataggio e caricamento su cassetta. Tutte queste piccole precauzioni non solo possono eliminare fastidiosi inconvenienti con il registratore, ma permettono anche un facile trasporto di programmi da computer a computer. ■

SINCLAIR ZX SPECTRUM

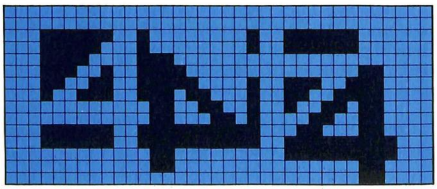
Le più interessanti caratteristiche di gestione del video

Marcello Spero

Iniziamo con questo numero a parlare dello ZX Spectrum. Come tutti sapranno senz'altro la nuova macchina della Sinclair, pur ricalcando sostanzialmente lo schema dello ZX81, a tutto beneficio di quanti su questo hanno imparato, presenta molte sostanziali, e in un certo senso sensazionali, novità.

Da questo mese inizieremo ad esaminare tutte quelle caratteristiche che, pur se accennate nel manuale, non sono sufficientemente sviluppate e spiegate, secondo un criterio abituale della Sinclair, che preferisce lasciare le cose per così dire "a metà" per dare ampio spazio all'inventiva e allo studio. Frutto di questa politica è l'enorme sviluppo di software e hardware aggiuntivo che, soprattutto in Gran Bretagna, segue l'immissione sul mercato di ogni nuova macchina Sinclair.

In questo numero approfondiremo in particolare alcune delle nuove potenzialità di gestione del video, che rappresentano la novità più attraente.



Provate il programma UNO: circa due volte al secondo viene visualizzato un carattere; quindi l'intero schermo viene invertito. Come certamente sapranno quelli che conoscono lo ZX81, su questa macchina per ottenere un effetto analogo era indispensabile ricorrere ad una routine in linguaggio macchina; qui, invece, si è ricorso ad una stringa di 704 spazi, visualizzati in INVERSE e OVER. L'attributo OVER opera un or esclusivo fra il carattere presente in una data posizione del video e il carattere che gli viene sovrapposto; in questo caso, però, non c'è niente con cui fare or-ex, trattandosi di una stringa di spazi, e l'effetto si limita ad una inversione dello schermo in un tempo molto breve. Come si può notare dalla linea 10 non è necessario caricare 704 spazi nella strin-

SINCLAIR ZX SPECTRUM

ga: infatti l'istruzione DIM carica il vettore corrispondente con zeri se questo è numerico, con spazi se è di caratteri. Per chi non avesse capito, 704 è il numero di caratteri contenuti in uno schermo (22 righe × 32 colonne).

C'è un altro metodo per lo stesso procedimento: un ciclo che stampi per 704 volte il carattere "spazio inverso"; questo ha il vantaggio di non occupare memoria per la stringa, ma risulta nettamente più lento.

Il programma UNO si riferisce al bianco e nero, ma la sua applicazione ad un programma che utilizza i colori è semplice: basta aggiungere alla linea 30

```
  1 REM programma UNO
 10 DIM i$(704)
 20 PRINT AT RND*20,RND*31 CHR$
(RND*223+32)
 30 PRINT AT 0,0; OVER 1; INVER
SE 1;i$
 40 GO TO 20

  1 REM nuova linea 30
 30 PRINT AT 0,0; OVER 1; INVER
SE 1; PAPER 8; INK 8; FLASH 8; B
RIGHT 8;i$

  1 REM programma DUE
 10 DIM i$(704)
 15 FOR i=1 TO 50
 20 PRINT AT RND*20,RND*31; INK
RND*7; PAPER RND*7; BRIGHT RND;
FLASH RND;CHR$ (RND*223+32)
 25 NEXT i
 30 PRINT AT 0,0; OVER 1; PAPER
8; INK 0; BRIGHT 8; FLASH 8;i$

  1 REM programma TRE
 10 DIM i$(704)
 15 FOR i=1 TO 50
 20 PRINT AT RND*20,RND*31; INK
RND*7; PAPER RND*7; BRIGHT RND;
FLASH RND;CHR$ (RND*223+32)
 25 NEXT i
 30 PRINT AT 0,0; OVER 1; PAPER
6; INK 8; BRIGHT 8; FLASH 8;i$

  1 REM programma QUATTRO
 10 DIM i$(704)
 15 FOR i=1 TO 50
 20 PRINT AT RND*20,RND*31; INK
RND*7; PAPER RND*7; BRIGHT RND;
FLASH RND;CHR$ (RND*223+32)
 25 NEXT i
 30 PRINT AT 0,0; OVER 1; PAPER
8; INK 8; BRIGHT 8; FLASH 0;i$

  1 REM programma CINQUE
  5 INK 6: PAPER 6: CLS
 10 DIM i$(704)
 15 FOR i=10 TO 70 STEP 20
 20 CIRCLE 120,90,i
 25 NEXT i
 30 PRINT AT 0,0; OVER 1; INK 3
;i$
 40 PAPER 7: INK 0
```

(quella che opera l'inversione) tutti gli attributi (PAPER, INK, FLASH, BRIGHT) con parametro 8; in questo modo ogni carattere rimarrà lampeggiante o più luminoso se già lo era, ma il colore dello sfondo sarà scambiato con quello del carattere.

La stessa idea si può applicare per dare a tutto il testo e la grafica presente sullo schermo un particolare colore, omettendo INVERSE 1 (o specificando INVERSE 0) e specificando un colore di INK invece di lasciarlo 8.

Il programma DUE visualizza caratteri con PAPER e INK casuale, e quindi li converte in nero, rispettando gli altri attributi. Se, per effetto della scelta casuale, alcuni caratteri risultassero illeggibili perché dello stesso colore dello sfondo, basta sostituire nella linea 20 INK 9 a INK RND*7.

Lo stesso tipo di sostituzione, fatto nella linea 30, fa sì che i caratteri vengano convertiti in modo da essere sempre leggibili.

Il programma TRE illustra lo stesso procedimento applicato al colore di sfondo.

Il programma QUATTRO permette invece di rendere fissi caratteri lampeggianti: può essere utile se si fanno apparire messaggi lampeggianti, che vanno resi fissi dopo che si è agito di conseguenza.

Notate come in tutti questi esempi la routine di modifica dello schermo è contenuta in una riga; il trucco stato tutto nella stringa di spazi.

Questa tecnica può essere utilizzata per uno scopo completamente diverso: le figure grafiche impiegano un certo tempo ad essere disegnate; volendo dare l'illusione di un plottaggio istantaneo, si può disegnare la figura con sfondo e caratteri dello stesso colore, e quindi cambiando il colore dei caratteri.

Il programma CINQUE fa questo, disegnando quattro cerchi in giallo su sfondo giallo, e quindi facendoli apparire in magenta.

Questi sono solo alcuni esempi: le applicazioni di questa tecnica possono essere moltissime.

Passiamo ad un altro argomento: normalmente, al di fuori delle istruzioni INPUT gli attributi della parte inferiore dello schermo non possono essere controllati, eccetto lo sfondo, che segue il colore del "border".

Come potremmo, ad esempio, avere il cursore e i messaggi del sistema scritti in verde? Anche se di dubbia utilità, questo è possibile modificando la variabile di sistema 23624, che contiene appunto gli attributi per la parte bassa dello schermo e il colore del border. Per inciso le variabili di sistema, che sono quelle memorizzate in RAM dalla locazione 23552 al-

SINCLAIR ZX SPECTRUM

la locazione 23733 e il cui elenco è al capitolo 25 del manuale, sono la chiave per moltissimi trucchi di programmazione, che analizzeremo poco per volta in futuro. La variabile 23624, dicevamo, è così suddivisa

bit	effetto
7	lampeggio parte bassa schermo
6	luminosità parte bassa schermo
5	colore border e parte bassa schermo
4	colore border e parte bassa schermo
3	colore border e parte bassa schermo
2	colore caratteri parte bassa schermo
1	colore caratteri parte bassa schermo
0	colore caratteri parte bassa schermo

Tavola 1. *Significato dei bit della locazione 23624*

Introducendo vari valori in questa variabile, per mezzo di istruzioni POKE, possiamo ottenere per esempio il lampeggio della parte bassa dello schermo, o una sua luminosità più accentuata. Notate che tutto questo non è ottenibile con istruzioni INPUT, in quanto gli attributi introdotti variano solo i messaggi, non i dati immessi da noi. Un'istruzione BORDER cancella le modifiche apportate a questa variabile, riportandola al valore normale. Provate questi due esempi:

POKE 23624, BIN 10111000 (184)
POKE 23624, BIN 01111000 (120)

Per concludere un paio di piccoli trucchetti sempre utili:

- Dovete cancellare una stringa o un numero molto lunghi in cui avete commesso un errore, in un INPUT? Non serve premere il tasto DELETE per un milione di volte: basta premere EDIT (shift 1) per far scomparire tutto quello che avete scritto in un input.
- Il vero equivalente dell'istruzione UNPLOT dello ZX81 non è PLOT OVER 1 ma PLOT INVERSE 1; infatti se usate PLOT OVER per un pixel (puntino) non precedentemente plottato, l'effetto, invece di essere nullo, sarà equivalente a quello di un normale PLOT. Se invece usate PLOT INVERSE, i pixel non plottati rimarranno tali, mentre quelli plottati verranno cancellati. Questo permette per esempio di cancellare i pixel plottati di una stessa riga con un'istruzione DRAW INVERSE, invece di andare a cancellarli uno per uno con tanti PLOT OVER. ■

Ed in linguaggio macchina...

Per coloro che avessero l'esigenza di un effetto assolutamente "istantaneo", ecco una breve routine in linguaggio macchina che può servire a creare gli effetti visti finora, con velocità ovviamente maggiore (senza il fascino di ottenere simili effetti con il Basic, però!).

Si tratta di nove passi di programma che caricano nei byte degli attributi (rimando chi non avesse le idee chiare a proposito al capitolo 24 del manuale) il valore desiderato. Qui di seguito è trascritto l'assembler, mentre nel programmino di caricamento trovate il codice macchina.

```
ld hl, 22528
ld bc, 768
ld (hl), 7
dec bc
ld a, b
or c
ret z
inc hl
jr, −8
```

Per tutti coloro che volessero utilizzarlo, segnalo che nell'ottavo byte del codice (quello che nel programma e nel codice tradotto è uguale a 7) sono contenuti gli attributi che vengono trasferiti allo schermo; così com'è, cioè uguale a 7, indica: FLASH 0, BRIGHT 0, PAPER 0, INK 7 cioè scrittura bianca su fondo nero. Potete ovviamente sostituirlo con il valore di cui avete bisogno; anzi, è facilmente possibile la sostituzione nel corso di un programma in cui la routine viene utilizzata, semplicemente con una POKE *n*+7, "nuovo valore" dove per *n* si intende l'indirizzo di inizio della routine (quello che compare dopo la funzione USR), che utilizzando il programmino di caricamento risulta essere uno più dell'indirizzo della nuova ramtop. Alternando per esempio i valori 7 e 56 (scrittura nera su fondo bianco) in successive chiamate della routine, si possono avere inversioni successive; consiglio in questo caso di intervallare le inversioni con qualche PAUSE, essendo altrimenti troppo veloce la successione delle inversioni (la routine impiega 51444 periodi di clock per essere eseguita, che alla frequenza di clock dello Spectrum, 3,5 MHz, equivale a circa 1,5 centesimi di secondo).

```
  1 REM caricamento
    codice macchina
 10 INPUT "nuova ramtop",r
 20 CLEAR r
 30 LET r=PEEK 23730+256*PEEK 2
3731
 40 FOR i=r+1 TO r+15
 50 READ m
 60 POKE i,m
 70 NEXT i
 80 REM codice macchina
 90 DATA 33,0,88,1,0,3,54,7,11,
120,177,200,35,24,247
```

COMMODORE VIC 20

Disabilitazione del tasto RUN/STOP

Alessandro Guida di Firenze ci svela un piccolo "segreto" del VIC. Lo ringraziamo a nome di tutti i lettori.

La routine che vedremo permette di disabilitare il tasto RUN/STOP anche se il VIC non ha alcuna istruzione diretta che ci dia questo risultato.

Nel computer viene attivata 60 volte al secondo la linea di interrupt. Questo fatto costringe il VIC a interrompere ciò di cui si stava occupando e passare all'esecuzione di una routine, detta appunto routine di interrupt. Questa si occupa di varie cose, come la gestione dell'orologio, il controllo del motore del registratore, la scansione della tastiera, ecc. All'interno di questa routine ne viene chiamata un'altra che analizza il tasto di STOP, e il cui indirizzo di partenza è conservato nelle locazioni $0328-$0329 (*test-STOP Vector*). Il programma presentato cambia il contenuto di questo vettore sostituendo la routine di test originale con una fittizia che non esegue alcun test. Per riabilitare il testo è necessario ristabilire l'indirizzo originale ($F770). La routine va chiamata con RS=0 per disabilitare lo STOP e con RS=1 per riabilitarlo.

```
61000 REM ************************
61010 REM * DISABILITAZIONE R/S *
61020 REM *                      *
61030 REM * RS=0 DISABILITA     *
61040 REM * RS=1 RIABILITA      *
61050 REM ************************
61060 RESTORE
61070 READ XX$
61075 IF XX$<>"**" TEHN 61070
61080 FOR I=0 TO 2: READ XX$
61085 POKE 251+I,VAL(XX$): NEXT
61090 FOR I=0 TO 25: READ XX$
61095 POKE 828+I,VAL(XX$): NEXT
61100 IF RS=0 THEN SYS 828: RETURN
61110 IF RS=1 THEN SYS 841: RETURN
61120 RETURN
61130 DATA **,165,145,96
61140 DATA 120,169,251,141,40,3
61145 DATA 169,0,141,41,3,88,96
61150 DATA 120,169,112,141,40,3
61155 DATA 169,247,141,41,3,88,96
```

Listato 1. *Subroutine Basic di disabilitazione.*

```
F770 LDA $91
F772 CMP #$FE
F774 BNE $F77D
F776 PHP
F777 JSR $FFCC
F77A STA $C6
F77C PLP
F77D RTS
```

Listato 2. *Routine di test-STOP originale.*

```
00FB LDA $91
00FD RTS
```

Listato 3. *Nuova routine.*

```
033C SEI
033D LDA #$FB
033F STA $0328
0342 LDA #$00
0344 STA $0329
0347 CLI
0348 RTS
0349 SEI
034A LDA #$70
034C STA $0328
034F LDA #$F7
0351 STA $0329
0354 CLI
0355 RTS
```

Listato 4. *Programma necessario per modificare il test-STOP Vector.*

■

Minicalc per lo ZX81

Un tabellone elettronico anche per lo ZX81

di *Enrico Ferreguti*

Ebbene sì: la mania dei tabelloni elettronici ha colpito anche lo ZX. Con questo programma potrete provare l'ebbrezza delle tabelle di dati più complicate che vi vengono in mente, lo potrete usare per calcoli lunghi e ripetitivi e potrete sbizzarrirvi con le proiezioni, pensando, tra l'altro, che state usando un programma che ricalca i famosissimi Visicalc, Ozz, ecc. e che non vi è costato niente.

Per coloro che non hanno idea di cosa sia un tabellone elettronico cerchiamo di chiarire la situazione.

Quante volte vi siete trovati davanti a delle tabelle e quante volte avete constatato l'inadeguatezza di penna e calcolatrice? Con il programma Minicalc potremo usare il nostro ZX per eseguire queste totalizzazioni.

Facciamo un esempio: dobbiamo calcolare il bilancio familiare per un arco di 5 mesi. Includiamo le principali voci: luce, gas, telefono, vitto e varie; le organizziamo in una tabella includendo delle altre voci: entrate, uscite, riporto, e tot. entrate (entrate + riporto) (vedi fig. 1).

Adesso è solo necessario riempire le caselle e legarle logicamente fra loro. Se però una volta compilata la tabella volessimo modificare una voce, con carta e penna si sarebbe costretti a ricominciare tutto da capo. Con ZX e Minicalc basteranno pochi secondi per vedere la situazione aggiornata.

Uso del programma

Dopo aver caricato il programma lo si lancia con RUN, dopodiché si incontra la fase di carica-

	MAR	APR	MAG	GIU	LUG
ENTRATE					
TOT. ENTRATE					
LUCE					
GAS					
TEL.					
VITTO					
VARIE					
USCITE					
RIPORTO					

Figura 1.

mento dati. Il programma comincia a mostrarvi sul video la tabella vuota e in fondo la frase CARICAMENTO DATI con l'attesa di input.

Il programma sta aspettando le coordinate della casella da riempire (prima la colonna e poi la riga). L'input della casella può essere nu-

```
  10 DIM W$(9,9,9)
  20 LET X=1
  30 GOSUB 1000
  40 PRINT AT 20,3;"CARICAMENTO
DATI ";
  50 INPUT F
  60 INPUT Z
  65 IF F>9 OR Z>9 THEN GOTO 140
  70 PRINT F;Z
  80 INPUT W$(F,Z)
  90 LET X=F
 100 GOTO 30
 140 DIM W(9,9)
 150 FOR F=1 TO 9
 155 FOR Z=1 TO 9
 160 IF CODE W$(F,Z)>27 AND CODE
 W$(F,Z)<38 THEN LET W(F,Z)=VAL
W$(F,Z)
 170 NEXT Z
 180 NEXT F
 200 CLS
 210 PRINT "ADESSO FERMERO"" IL
PROGRAMMA"
 220 PRINT "PER PERMETTERTI DI M
ODIFICARE"
 230 PRINT "LE RELAZIONI ALLA RI
GA 8000"
 240 PRINT
 250 PRINT "POI PER RICOMINCIARE
 BATTI GOTO 300"
 260 STOP
 300 GOSUB 1000
 304 REM ROUTINE CA
 305 GOSUB 8000
 307 FAST
 310 FOR F=1 TO 9
 320 FOR Z=1 TO 9
 325 IF W$(F,Z)="" THEN GOTO 340
 330 IF CODE W$(F,Z)>27 AND CODE
 W$(F,Z)<38 THEN LET W$(F,Z)=STR
$ W(F,Z)
 340 NEXT Z
 350 NEXT F
 360 SLOW
 370 GOSUB 1000
 380 PRINT AT 20,0;"COMANDI ? (D
X,SX,CA,CH,SE,ST,SA)"
 390 INPUT H$
 400 IF H$="DX" THEN GOTO 2000
 405 IF H$="SA" THEN GOTO 3000
 410 IF H$="SX" THEN GOTO 2100
 415 IF H$="ST" THEN GOTO 4000
 420 IF H$="CA" THEN GOTO 305
 425 IF H$="CH" THEN GOTO 2500
 427 IF H$="SE" THEN GOTO 3500
 430 INPUT F
 440 INPUT Z
 441 IF F>X+2 OR F<X THEN PRINT
AT 20,0;"FUORI SCHERMO : RIPETI
                "
 442 IF F>X+2 OR F<X THEN GOTO 4
30
 445 LET H$="                "
 447 LET K$=W$(F,Z)
 450 FOR H=1 TO 9
 452 LET H$(H)=CHR$ (CODE K$(H)+
128)
 455 NEXT H
 457 PRINT AT Z*2,10*(((F=4 OR F
=7 OR F=1)+2*(F=2 OR F=5 OR F=8)
+3*(F=3 OR F=6 OR F=9))-1)+2;H$
 470 IF CODE W$(F,Z)>27 AND CODE
 W$(F,Z)<38 THEN GOTO 500
 480 INPUT W$(F,Z)
 490 GOTO 520
 500 INPUT W(F,Z)
 510 LET W$(F,Z)=STR$ W(F,Z)
 520 GOTO 370
1000 LET A$="
                "
1010 LET A$(8)=CHR$ (X+156)
1020 LET A$(17)=CHR$ (X+157)
1030 LET A$(27)=CHR$ (X+158)
1035 CLS
1040 LET B$="▒:---------:-------
---:---------:"
1050 LET C$=":
1055 PRINT A$
1060 FOR Z=1 TO 9
1070 PRINT B$
1080 PRINT CHR$ (Z+156);C$
1085 NEXT Z
1090 PRINT B$
1095 FOR F=1 TO 9
1100 PRINT AT F*2,2;W$(X,F);AT F
*2,12;W$(X+1,F);AT F*2,22;W$(X+2
,F)
1110 NEXT F
1200 RETURN
1999 REM ROUTINE DX
2000 LET X=X+1
2010 IF X+2>9 THEN LET X=7
2020 GOTO 370
2099 REM ROUTINE SX
2100 LET X=X-1
2110 IF X<1 THEN LET X=1
2120 GOTO 370
2499 REM ROUTINE CH
2500 PRINT AT 20,0;"QUALE CASELL
A CAMBI ?
                "
2510 INPUT F
2520 INPUT Z
2530 PRINT AT 20,0;"
                "
2540 PRINT AT 20,0;"LA CASELLA "
;F;",";Z;" E"" = ?"
2550 INPUT H$
2560 IF CODE H$>27 AND CODE H$<3
8 THEN GOTO 2600
2570 LET W$(F,Z)=H$
2580 GOTO 370
2600 LET W(F,Z)=VAL H$
2610 GOTO 370
2999 REM ROUTINE SA
3000 PRINT AT 20,0;"CHE NOME DAI
 A QUESTO LAVORO ?  "
3010 INPUT H$
3020 PRINT AT 20,0;"PREPARA IL R
EGISTRATORE
                "
3030 PAUSE 3E2
3040 PRINT AT 20,0;"AVVIALO CON
<RECORD>
                "
3050 PAUSE 150
3060 SAVE H$
3070 GOTO 370
3499 REM ROUTINE SE
3500 PRINT AT 20,0;"QUALE SETTOR
E ?
                "
3510 INPUT N
3520 LET X=N*3-2
3530 GOTO 370
4000 REM ROUTINE ST
4005 PRINT AT 20,0;"QUALE SETTOR
E STAMPO ?
                "
4010 INPUT N
4012 LET Q=N*3-2
4015 PRINT AT 20,0;"QUAL""E"" IL
 NOME DEL LAVORO?"
4017 INPUT H$
4018 LPRINT H$
4020 FOR F=1 TO 9
4030 LPRINT ";";W$(Q,F);":";W$(Q
+1,F);":";W$(Q+2,F)
4040 NEXT F
4050 GOTO 370
7900 REM NELLE RIGHE SOTTOSTANTI
7950 REM SONO MEMORIZZATE LE
7960 REM REL. FRA CASELLE
8000 LET W(2,3)=W(2,6)*W(2,6)
8010 LET W(2,4)=W(2,2)*W(2,3)
8020 LET W(3,3)=W(3,6)*W(2,3)
8030 LET W(3,4)=W(3,2)*W(3,3)
8040 RETURN
8050 RETURN
```

Listato 1. *Il programma Minicalc.*

merico o alfanumerico; nel primo caso la casella sarà operativa, cioè sarà significativa ai fini del calcolo stesso, mentre nel secondo caso sarà dedicata solo al commento. Come in una qualsiasi tabella può essere omessa, a scapito però della chiarezza. Per uscire da questa fase bisognerà fornire un numero maggiore di 9.

Quando il calcolatore chiede le coordinate della casella, il programma si ferma per dare la possibilità di cambiare o riscrivere le relazioni in riga 8000. Vediamo subito di cosa si tratta: quando facciamo una tabella del tipo descritto prima i legami esistenti tra una casella e l'altra sono logici ed evidenti per noi ma non al calcolatore: dobbiamo quindi insegnarglieli. Per esempio della tabella di prima sappiamo che alla voce USCITE dovremo scrivere la somma di LUCE, GAS, TEL, VITTO e VARIE e questo è ciò che vogliamo "insegnare" al computer. Il tabellone, nel calcolatore, è memorizzato come una matrice bidimensionale di 9×9 elementi, quindi, immaginando che marzo sia il mese, batteremo nella riga 8000:

```
LET W(2,8)=W(2,4)+W(2,5)+
   +W(2,6)+W(2,7)
```

che per il computer significa che la casella 2,8 (ricordiamoci che la prima colonna è occupata dal commento) è uguale alla somma delle caselle 2,4, 2,5, 2,6 e 2,7.

Con questo abbiamo già fatto un'importante osservazione: le caselle TOT. ENTRATE, USCITE e RIPORTO sono caselle dipendenti, quindi non si possono modificare (dato che, ogni volta che eseguiamo l'aggiornamento, il valore della nostra modifica viene sostituito dal valore dell'attribuzione in riga 8000) e devono apparire in riga 8000 o seguenti.

Tutte le altre caselle sono indipendenti e si possono modificare (anzi si devono affinché il programma abbia senso) e da loro dipendono le caselle le cui attribuzioni compaiono in riga 8000 (caselle dipendenti).

Fatto questo si riparte con GOTO 300.

Apparirà sullo schermo la griglia del tabellone, il contenuto delle caselle e in fondo il messaggio COMANDI? (DX,SX,CA,CH,SE,ST, SA) il cui significato verrà discusso in seguito.

Per poter modificare le caselle dovremo battere NEW/LINE. Appariranno due richieste di input con cui inseriremo le coordinate della casella da cambiare (prima la riga poi la colonna).

La casella che vogliamo trasformare sarà visualizzata in inverso e si potrà passare alla modifica.

Dopo aver battuto NEW/LINE per far accettare il cambiamento, il video verrà aggiornato; se si desiderano altri cambiamenti si dovrà ripetere la procedura di prima oppure si potrà usare uno dei seguenti comandi:

CA (calcola)

Finora abbiamo solo modificato le caselle; per far adeguare le caselle dipendenti alla modifica, cioè per vedere il cambiamento che ne è seguito, sarà necessario usare questo comando che aggiornerà i dati per poi ritornare alla sezione di input.

SE (settore)

Il tabellone è formato da 9×9 caselle, però è diviso in tre settori da 3×9 caselle (schermate), cioè il settore da 1 a 3, da 4 a 6 e da 7 a 9. Questo comando è importante perché quando si ha bisogno di modificare la casella in un settore diverso da quello in cui stiamo lavorando si dovrà darlo in input.

DX (destra) e SX (sinistra)

Hanno più o meno la stessa funzione di SE però in seguito ad uno di questi non si potrà più fare un input corretto, a meno che non lo si faccia seguire a un comando SE. Servono soprattutto per fare una ricognizione del tabellone di una riga a destra (DX) o a sinistra (SX).

ST (stampa)

Verrà richiesto il nome del lavoro che stiamo eseguendo e il settore da stampare e il suo contenuto sarà trasferito su carta.

SA (salva)

Serve a salvare su cassetta il lavoro finora eseguito: viene richiesto il nome del lavoro e dopo i messaggi per la registrazione verrà eseguito il salvataggio.

CH (*change*, cambia)

Se si vuole cambiare una casella da commento ad operativa o viceversa si dovrà usare questo comando (attenti a non convertire una casella dipendente).

Ora potete far girare il programma. Se per qualche motivo vi fermate (p.es. per aver premuto il tasto BREAK), niente problemi, battete GOTO 370 e ricominciate.

Il programma gira su tutti gli ZX 81 e ZX 80 nuova ROM con l'espansione da 16 K.

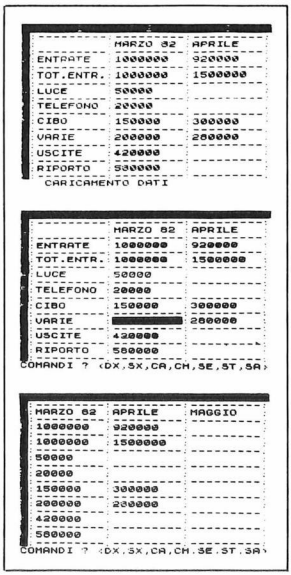


Figura 2. *Esempio di esecuzione del programma Minicalc.*

```
 7 INPUT N
10 DIM A$(N,5)
12 DIM X(N)
30 FOR J=SGN PI TO N
31 PRINT J
35 INPUT A$(J)
40 INPUT X(J)
42 CLS
45 NEXT J
50 GOSUB 150
55 PRINT AT 21,NOT PI;"ORDINI?
";
60 INPUT A
62 IF A>10 THEN GOTO 80
65 PRINT AT 21,NOT PI;A$(A);"=
";
70 INPUT X(A)
75 GOTO 55
80 GOTO 300
150 CLS
155 FOR J=SGN PI TO N
160 PRINT CHR$ (J+156);A$(J);"
";X(J)
170 PRINT
180 NEXT J
190 RETURN
300 GOTO 50
```

Listato 2. *Il programma Minicalc per lo ZX81 senza espansione.*

Minicalc senza espansione

Il folto gruppo di sinclairisti non espansi, in attesa di procurarsi l'espansione di memoria, gradiranno questa versione del Minicalc per la loro macchina.

In questo caso il lavoro non si svolge più su un foglio bidimensionale, ma su uno unidimensionale: in pratica il tabellone è costruito in 10 caselle disposte una sopra l'altra e memorizzate in un vettore di *n* elementi (*n* è il numero fornito in input subito dopo il RUN).

Prima di cominciare bisogna inserire le relazioni, ricordandoci che i dati sono memorizzati nel vettore X, nella riga 300, facendole seguire da un GOTO 5.

La versione per lo ZX senza espansione

Si dà il RUN, si inserisce il numero di caselle e si passa immediatamente all'input di tutte le caselle (prima i commenti, poi i valori).

Il programma stampa la tabella e fa comparire in fondo allo schermo la scritta ORDINI? e resta in attesa di un input numerico che serve per dire al calcolatore quale casella modificare; indi scrive sul fondo il nome della casella e l'input per la modifica. Continuerà a ritornare il messaggio ORDINI? fino a quando non si inserirà un numero superiore a 10. Fatto questo, il programma esce dal ciclo di modifica, aggiorna i risultati leggendo le relazioni della riga 300 e ristampa la tabella aggiornata con la scritta ORDINI? in fondo. ■

La carta del cielo per il PET/CBM

Riceviamo da Nevio Faccini di Padova la conversione del programma "La carta del cielo". Lo ringraziamo e ne approfittiamo per pubblicare una tavola degli aspetti tra i pianeti, necessaria per la comprensione delle "carte" fornite dal programma.

Prima di convertire per il CBM 3032 il programma "La carta del cielo", (*Personal Software* 3) mi sono procurato da un amico che possiede l'Apple il risultato finale di determinati input iniziali e con queste informazioni ho iniziato la conversione.

La parte più difficile è stata quella di disegnare, sullo schermo, il reticolo delle "case", perché nell'Apple questo viene costruito con le istruzioni HTAB e VTAB in cicli FOR-NEXT, istruzioni non presenti nel CBM. Ho risolto il problema usando rispettivamente le POKE 198 e 216, comandate dal SYS 57949. Infatti aggiungendo alla appropriata POKE, il numero di riga o di colonna, si ottiene sullo schermo lo stesso risultato; unica differenza è che il computer della Commodore vuole un'unità in meno:

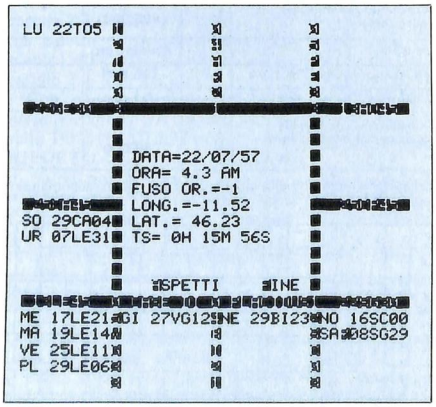


Figura 1. *La carta del cielo usata per il test*

abbreviazione	nome	angolo	approssimazione
CON	congiunzione	0°	7°
OPP	opposizione	180°	7°
TRI	trigono	120°	7°
QUA	quadrato	90°	7°
SES	sestile	60°	5°
SQU	semiquadrato	45°	2°
SSE	sesquadro	135°	2°
QUI	quinconce	150°	2°

Questa tavola, con le abbreviazioni degli aspetti e l'approssimazione usata, completa le tavole apparse a pag. 83-87 di *Personal Software* 3 [*N.d.R.*].

Tavola degli aspetti

se l'Apple adopera HTAB 10, nel CBM dobbiamo mettere POKE 198,9, oppure PRINT TAB(9).

Come si sa il GET del CBM è dinamico e non statico come quello dell'Apple, quindi l'ho dovuto rendere tale, con la classica linea 10 IF GET (A$)=" " THEN 10.

Poi ho apportato qualche personale modifica all'originale, per migliorarne il funzionamento. Notando che per uscire dal programma si doveva usare il tasto dello STOP, per ovviare l'inconveniente e poter uscire, diciamo con più eleganza, al GET di linea 1670, ho aggiunto un'altra condizione con la relativa informazione: linee 390 e 410.

Anche la visualizzazione della data è stata modificata; come si sa, davanti ad ogni numero vi è uno spazio per il segno, che è visibile solo se è il "−" di negativo, la data quindi è abbastanza scombinata e la linea troppo lunga per essere poi cancellata dal ciclo che si trova nelle linee 540 e 550; per questa ragione ho aggiunto le istruzioni della linea 480, usando le variabili: GG$ per il giorno, MM$ per il mese e AA$ per l'anno, facendole poi stampare con le istruzioni della linea 560. Naturalmente ho lasciato le variabili numeriche originali, perché servono per i conteggi.

Per andare sul sicuro, dopo avere inserito il programma nella memoria del vostro computer, ed averlo salvato (non si sa mai) fate il RUN e alla richiesta battete la T per tropicale ed i seguenti dati:

DATA = 22/07/1957
AM/PM = AM
ORA = 04.30
FUSO OR. = −1

LONG. = −11.52
LAT. = 46.23

Subito dopo l'ultimo inserimento comparirà sullo schermo il tempo siderale (TS=), che dovrà essere di 0H 15M 56S. Alla fine dei conteggi, verrà visualizzato un reticolo uguale a quello in figura, con i dati uguali a quelli che si trovano in ogni "casa". Se così non fosse, ricontrollate il listato, perché sicuramente vi è qualche errore eventualmente nelle DATA (linee 100/290). Non aggiungo gli ASPETTI che potrete ottenere premendo i numeri da 0 a 9, perché se il vostro reticolo è uguale a quello in figura, senz'altro anche questi saranno corretti. Per far ripartire il programma premete F.

I dati relativi agli orari e alle località sono quelli pubblicati su *Personal Software* 3. Ho variato la sigla del segno della Vergine in VG, per distinguerla dalla sigla del pianeta Venere e quella del Sagittario in SG, per distinguerla da Saturno.

```
100 DATA AR,SO,TO,ME,GE,VE,CA,MA,LE,GI,VG,
    SA,BI,UR,SC,NE,SG,PL,CP,LU,AQ,NO,PE,0
110 DATA 1,13,1,19,11,19,21,19,31,19,31,13
    ,31,7,31,1,21,1,11,1,1,1,1,7
120 DATA CON07000,OPP07180,TRI07120,QUA070
    90,SES05060,SQU02045,SSE02135
130 DATA QUI02150
140 DATA 358.47584,35999.0498,-.00015,.016
    751,-.41E-4,0,1.00000013,101.22083
150 DATA 1.71918,.00045,0,0,0,0,0,0,102.27
    938,149472.515,0,.205614,.2E-4,0
160 DATA .387098,28.75375,.37028,.00012,47
    .14594,1.1852,.00017,7.00288,.00186
170 DATA -.1E-4,212.60322,58517.8039,.0012
    9,.00682,-.4E-4,0,.723332
180 DATA 54.38419,.50819,-.00139,75.77985,
    .89985,.00041,3.39363,.001,0,319.5293
190 DATA 19139.8585,.00018,.09331,.9E-4,0,
    1.52369,285.43176,1.06977,.00013
200 DATA 48.78644,.77099,0,1.85033,-.00068
    ,.1E-4,225.32833,3034.69202
210 DATA -.00072,.04833,.00016,0,5.202561,
    273.27754,.59943,.0007,99.44338
220 DATA 1.01053,.00035,1.30874,-.005696,0
    ,175.46622,1221.55147,-.0005
230 DATA .05589,-.00035,0,9.55475,338.3077
    7,1.08522,.00098,112.79039,.873195
240 DATA -.00015,2.49252,-.00392,-.2E-4,72
    .64882,428.37911,.8E-4,.046344
250 DATA -.3E-4,0,19.21814,98.07155,.98577
    ,-.00107,73.4771,.49867,.00131
260 DATA .77246,.00063,.4E-4,37.73067,218.
    46134,-.7E-4,.008997,0,0
270 DATA 30.10957,276.04597,.32564,.00014,
    130.68136,1.09894,.000249,1.77924
280 DATA -.00954,0,229.94722,144.91306,0,.
    24864,0,0,39.51774,113.52139,0,0
290 DATA 108.95444,1.39601,.00031,17.14678
    ,0,0
300 DEF FNS(X)=SIN(X*PI/180):
    DEF FND(X)=X*180/PI:
    T$="SOOOOOOOOOOOOOOOOO"
310 PRINT CHR$(147): PI=3.14159265
320 DEF FNO(X)=SGN(X)*(INT(ABS(X))+(ABS(X)
    -INT(ABS(X)))*100/60): V$="SOOOOOOO"
330 DEF FNU(X)=X-(INT(X/MO)*MO):
    DEF FNW(X)=INT(X*1004+.5)/100:
    DEF FNR(X)=X*PI/180
340 DEF FNY(X)=ATN(SQR(1-X*X)/X)
350 DIM H$(12): DIM R(12,2): DIM H(13):
    DIM Z$(12)
360 DIM C$(12): DIM A$(8): DIM K(12):
    DIM C(12): K$="                "
370 POKE 216,9: POKE 198,11: SYS 57949
380 PRINT CHR$(18)"T"CHR$(146)
    "ROPICALE  ";CHR$(18)"S"CHR$(146)
    "IDERALE"
390 POKE 216,20: SYS 57949:
    PRINT "PER FINIRE PREMI <RETURN>"
400 GOSUB 1670: SD=0: IF G$="S" THEN SD=1
410 IF G$=CHR$(13) THEN PRINT CHR$(147):
    END
420 PRINT CHR$(147)
430 DEF FNX(X)=ATN(X/SQR(1-X*X)): MO=360:
    FOR I=1 TO 12: READ Z$(I): READ C$(I):
    NEXT I
440 FOR I=1 TO 12: FOR J=1 TO 2:
    READ R(I,J): NEXT J: NEXT I:
    FOR I=1 TO 8: READ A$(I): NEXT I
450 POKE 216,8: SYS 57949
460 PRINT TAB(9);:
    INPUT "DATA(GG/MM/AAAA)=";A$:
    D=VAL(MID$(A$,1,2))
470 Y=VAL(MID$(A$,7,5)):
    M=VAL(MID$(A$,4,2)): PRINT TAB(9);:
    INPUT "AM/PM=";F$
480 GG$=LEFT$(A$,3): MM$=MID$(A$,4,3):
    AA$=RIGHT$(A$,2)
490 PRINT TAB(9);: INPUT "ORA(HH.MM)=";F:
    F0=F
500 PRINT TAB(9);: INPUT "FUSO ORARIO=";X:
    X0=X: F=FN O(F)+X
510 PRINT TAB(9);:
    INPUT "LONGITUDINE(GG.MM)=";L5: L0=L5:
    L5=FN O(L5)
520 IF F$="PM" THEN F=F+12
530 PRINT TAB(9);:
    INPUT "LATITUDINE(GG.MM)=";B4: B04=B4:
    B4=FN R(FN O(B4)): X=0
540 LA=B4: FOR Z0=8 TO 15: POKE 198,9:
    POKE 216,Z0: SYS 57949
550 PRINT
    "                              ";
    NEXT Z0: PRINT CHR$(147): PRINT
560 PRINT: PRINT: PRINT: PRINT: PRINT:
    PRINT
570 PRINT TAB(11)"DATA=";GG$;MM$;AA$:
    PRINT TAB(11);"ORA=";F0;F$
580 PRINT TAB(11)"FUSO OR.=";X0
590 PRINT TAB(11);"LONG.=";L05:
    PRINT TAB(11);"LAT.=";B05
600 IM=12*(Y+4800)+M-3:
    J=(2*(IM-INT(IM/12)*12)+7+365*IM)/12
610 J=INT(J)+D+INT(IM/48)-32083:
    IF J<2299171 THEN 630
620 J=J+INT(IM/4800)-INT(IM/1200)+38
630 T=((J-2415020)*F/24-.5)/36525
640 C(11)=FN U((933060-6962911*T+7.5*T*T)/
    3600): IF SD=1 THEN GOSUB 1370
```

(segue)

(segue)

```
650 RA=FN R(FN U((6.646065556+2400.051262*
    T+2.5805E-5*T*T+F)*15-L5))
660 OB=FN R(23.45229444-.0130125*T)
670 MC=ATN(TAN(RA)/COS(OB)):
    IF MC<0 THEN MC=MC+PI
680 IF SIN(RA)<0 THEN MC=MC+PI
690 MC=FN D(MC)
700 X=FN D(RA)/15: Y=INT(X): Z=(X-Y)*60:
    X=INT((Z-INT(Z))*60): Z=INT(Z)
710 PRINT TAB(8);"    TS=";STR$(Y)+
    "H"+STR$(Z)+"M"+STR$(X)+"S":
    GOSUB 1750
720 GOSUB 1030
730 FOR I=1 TO 9: MO=2*PI: GOSUB 960:
    M=FN U(S): GOSUB 960: E=FN D(S): EA=M:
    FOR A=1 TO 5
740 EA=M+E*SIN(EA): NEXT A: READ AU:
    X=AU*(COS(EA)-E):
    Y=AU*SIN(EA)*SQR(1-E*E)
750 GOSUB 990: GOSUB 960: A=A+S:
    GOSUB 970: D=X: X=Y: Y=0: GOSUB 990:
    GOSUB 960: AN=S
760 GOSUB 960: A=A+S: GOSUB 970: ZZ=Y:
    Y=X: X=D: GOSUB 990: A=A+AN:
    IF A<0 THEN A=A+2*PI
770 GOSUB 970: XX=X: YY=Y: GOSUB 900
780 IF I=1 THEN C1=K: C=FN D(FN U(K+PI)):
    M1=R: X1=XX: Y1=YY: Z1=ZZ: X=1:
    GOTO 810
790 W1=(((R^.5+M1^.5)*(M1^.5*R^.5))/(R^1.5
    +M1^1.5))-COS(C1-K)
800 XX=XX-X1: YY=YY-Y1: ZZ=ZZ-Z1:
    GOSUB 900: X=1: IF W1<0 THEN X=-X
810 MO=360: C(I)=FN U(C+SD)*X: NEXT I
820 LL=973563+1732564379*T-4*T*T:
    G=1012395+6189*T:
    G1=1203586+1464852 3*T-37*T*T
830 D=1262655+1602961611*T-5*T*T: M=3600:
    L=(LL-G1)/M: L1=((LL-D)-G)/M
840 F=(LL-(C(11)*M))/M: D=D/M:
    ML=22639.6*FN S(L)-669*FN S(L1)
850 Y=2*D:
    ML=ML-4586.4*FN S(L-Y)+2369.9*FN S(Y)+
    769*FN S(2*L)-206*FN S(L+L1-Y)
860 ML=ML-411.6*FNS(2*F)-212*FNS(2*L-Y)+19
    2*FN S(L+Y)-165*FN S(L1-Y)-125*FNS(D)
870 ML=(LL+ML-165*FN S(L1-Y)+148*FN S(L-L1
    )-110*FN S(L+L1)-55*FN S(2*F-Y))/M
880 C(10)=FN U(ML+SD)
890 C(11)=FN U(C(11)+SD): GOTO 1150
900 X=XX: Y=YY: GOSUB 990: K=A: C=FN D(A):
    RETURN
910 Z3=INT(Z): D=INT(Z3/30)+1:
    Z1=INT(FN W((Z3/30-INT(Z3/30))*30))
920 X$=STR$(Z1):
    IF Z1<10 THEN X$="0"+RIGHT$(X$,1)
930 ZZ$=RIGHT$(STR$(INT((Z-Z2)*60+.5)),2)
940 IF VAL(ZZ$)<10 THEN ZZ$=
    "0"+RIGHT$(ZZ$,1)
950 Z$=RIGHT$(X$,2)+Z$(D)+ZZ$: RETURN
960 READ S,S1,S2: S=FN R(S+S1*T+S2*T*T):
    RETURN
970 IF A=0 THEN A=1.75E-9
980 X=R*COS(A): Y=R*SIN(A): RETURN
990 IF Y=0 THEN Y=1.75E-9
1000 R=(X*X+Y*Y)^.5: A=ATN(Y/X):
     IF A<0 THEN A=A+PI
1010 IF Y<0 THEN A=A+PI
1020 RETURN
1030 POKE 216,1: SYS 57949: FOR I=1 TO 5
1040 PRINT K$;:
     PRINT CHR$(18)MID$(H$(11),I,1)CHR$(146
     );: PRINT K$;
1045 PRINT CHR$(18)MID$(H$(10),I,1)CHR$(146
     );
1050 PRINT K$;:
     PRINT CHR$(18)MID$(H$(9),I,1): NEXT I
1055 PRINT CHR$(18)" "H$(12)
     "  "MID$(H$(11),6);
1060 PRINT K$ MID$(H$(10),6,1) K$ MID$(H$(9
     ),6,1)"  "H$(8)" "
1070 FOR I=1 TO 5: PRINT TAB(9):
     PRINT CHR$(18)" "CHR$(146);:
     PRINT TAB(29)
1075 PRINT CHR$(18)" ": NEXT I
1080 PRINT CHR$(18)" "H$(1)"    ";:
     PRINT TAB(29)
1085 PRINT CHR$(18)"    "H$(7)" "
1090 FOR I=1 TO 5: PRINT TAB(9):
     PRINT CHR$(18)" "CHR$(146);:
     PRINT TAB(29)
1095 PRINT CHR$(18)" ": NEXT I
1100 PRINT CHR$(18)" "H$(2)
     "  " MID$(H$(3),1,1)
     " CASE DI " MID$(H$(4),1,1);
1110 PRINT " PLACIDUS" MID$(H$(5),1,1)
     "  "H$(6)" ": FOR I=1 TO 5: PRINT K$;
1120 PRINT CHR$(18)MID$(H$(3),I+1,1)CHR$(14
     6);: PRINT K$;
1130 PRINT CHR$(18)MID$(H$(4),I+1,1)CHR$(14
     6);: PRINT K$;
1140 PRINT CHR$(18)MID$(H$(5),I+1,1):
     NEXT I: RETURN
1150 FOR I=1 TO 11
1160 K(I)=ABS(C(I)): NEXT I:
     FOR I=1 TO 11-1: FOR J=I+1 TO 11:
     IF K(J)>=K(I)GOTO 1180
1170 K1=K(I): K(I)=K(J): K(J)=K1
1180 NEXT J: NEXT I: A=1: FOR I=1 TO 11:
     FOR J=1 TO 11
1190 IF K(A)=ABS(C(J)) THEN GOSUB 1290:
     H$(A)=C$(J)+R$
1200 NEXT J
1210 A=A+1: NEXT I
1220 H(13)=H(1): FOR I=1 TO 12: HH=H(I+1):
     IF H(I)>H(I+1) THEN HH=H(I+1)+360:
     AB=1
1230 I$=V$: GG=0: G=0
1240 POKE 216,R(I,2): SYS 57949:
     FOR J=1 TO 11: CC=K(J):
     IF AB=1 AND CC<90 THEN CC=CC+360
1250 IF G>4 THEN GG=1
1260 IF CC>H(I) AND CC<HH THEN GOSUB 1320:
     G=G+1
1270 NEXT J: AB=0: NEXT I
1280 GOTO 1400
1290 R$=" ": IF C(J)<0 THEN R$="R"
1300 C(J)=ABS(C(J))
1310 RETURN
1320 Z=K(J): GOSUB 910:
     IF GG=1 THEN PRINT I$TAB(16)STR$(I);:
     GG=0: I$="": GOTO 1360
1330 POKE 198,R(I,1)
1340 PRINT CHR$(157)LEFT$(H$(J),2);
1345 IF RIGHT$(H$(J),1)=
     "R" THEN PRINT CHR$(18)"R"CHR$(146);:
     GOTO 1360
1350 PRINT " ";
1360 PRINT Z$: H$(J)="": RETURN
1370 SD=(259205536*T+2013816)/3600
```

(segue)

(segue)

```
1380 SD=17.23*SIN(FN R(C(11)))+1.27*SIN(FN
     R(SD))-(5025.64+1.11*T)*T
1390 SD=(SD-84038.27)/3600: RETURN
1400 GOSUB 1540
1410 FOR I=1 TO 11: POKE 216,6:
     POKE 198,13: SYS 57949
1420 PRINT CHR$(18)C$(I)SPC(6)"CENTRO":
     L=L1
1430 FOR J=1 TO 11:
     IF C(I)=C(J) THEN GOTO 1530
1440 X$=CHR$(J+48): IF J=10 THEN X$="*"
1450 IF J=11 THEN X$="0"
1460 POKE 198,11: SYS 57949:
     PRINT CHR$(18)X$CHR$(146);:
     PRINT " "C$(J)" ";
1465 AS=ABS(C(I)-C(J))
1470 IF AS>180 THEN AS=360-AS
1480 FOR K=1 TO 8:
     D=ABS(AS-VAL(RIGHT$(A$(K),3))):
     X$="  "
1490 IF D<VAL(MID$(A$(K),4,2)) THEN PRINT
     "*"LEFT$(A$(K),3);: K=9: X$="":
     GOTO 1510
1500 NEXT K
1510 PRINT X$RIGHT$(
     "  "+STR$(INT(AS+.5)),3);
1520 GOSUB 1630: Z=(Y+.5): GOSUB 910:
     PRINT " "LEFT$(Z$,4)
1530 NEXT J: GOSUB 1540: NEXT I:
     PRINT "S000000000":: GOSUB 1540:
     GOTO 1410
1540 ZX=11: POKE 198,13: POKE 216,17:
     SYS 57949
1545 PRINT CHR$(18)"A"CHR$(146)
     "SPETTI    ";CHR$(18)"F"CHR$(146)"INE"
1550 POKE 216,17: POKE 198,13: SYS 57949
1560 GET G$: IF G$="" THEN 1560
1570 IF ASC(G$)>48 AND ASC(G$)<58 THEN I=VA
     L(G$)-1
1580 IF G$="*" THEN I=9
1590 IF G$="0" THEN I=10
1600 IF G$="F" THEN RUN 300
1610 POKE 216,7: FOR K=1 TO 11: SYS 57949:
     POKE 198,11: SYS 57949:
     PRINT "                "
1620 NEXT K: RETURN
1630 W=0: Y=ABS(C(I)-C(J)):
     IF Y<=180 THEN 1660
1640 W=180: IF (C(I)+C(J))/2<180 THEN 1660
1650 W=-180
1660 Y=(C(I)+C(J))/2+W: RETURN
1670 GET G$: IF G$="" THEN 1670
1680 RETURN
1690 MO=360:
     A1=FN X(SIN(RA)*TAN(B4)*TAN(OB)):
     FOR I=1 TO 12: D=FN U(60+30*I)
1700 A2=D/90-1: KN=1: IF D>=180 THEN KN=-1:
     A2=D/90-3
1710 A3=FN R(FN U(FN D(RA)+D+A2*FN D(A1)))
1720 X=ATN(SIN(A3)/(COS(A3)*COS(OB)-KN*TAN(
     B4)*SIN(OB))): IF X<0 THEN X=X+PI
1730 IF SIN(A3)<0 THEN X=X+PI
1740 Z=FN U(FN D(X)+SD): H(I)=Z: GOSUB 910:
     H$(I)=Z$: NEXT I: RETURN
1750 Y=0: R1=RA+FN R(30): FF=3: GOSUB 1810:
     H(5)=FN U(L+180): HS$="PLACIDUS"
1760 R1=RA+FN R(60): FF=1.5: GOSUB 1810:
     H(6)=FN U(L+180): FF=1: GOSUB 1810:
     H(1)=L
1770 R1=RA+FN R(120): FF=1.5: Y=1:
     GOSUB 1810: H(2)=L: R1=RA+FN R(150):
     FF=3
1780 GOSUB 1810: H(3)=L: H(4)=FN U(MC+180):
     FOR I=1 TO 6: H(I)=FN U(H(I)+SD):
     NEXT I
1790 FOR I=1 TO 6: Z=FN U(H(I)+180):
     H(I+6)=Z: GOSUB 910: H$(I+6)=Z$
1800 Z=H(I): GOSUB 910: H$(I)=Z$: NEXT I:
     RETURN
1810 X=-1: IF Y=1 THEN X=1
1820 FOR I=1 TO 10:
     XX=FN Y(X*SIN(R1)*TAN(OB)*TAN(LA)):
     IF XX<0 THEN XX=XX+PI
1830 R2=RA+(XX/FF):
     IF Y=1 THEN R2=RA+PI-(XX/FF)
1840 R1=R2: NEXT I: L=ATN(TAN(R1)/COS(OB)):
     IF L<0 THEN L=L+PI
1850 IF SIN(R1)<0 THEN L=L+PI
1860 L=FN D(L): RETURN
```

Raccogliamo in questa rubrica le conversioni dei programmi che abbiamo pubblicato nei numeri precedenti, realizzate dai lettori per i loro personal. Invitiamo tutti a contribuire, inviandoci un listato, meglio se anche su supporto magnetico, e una descrizione (un paio di pagine dattiloscritte) dei problemi incontrati nella conversione, e delle tecniche usate per risolverli. Inviate la corrispondenza a

Personal Software
Rubrica "Conversioni"
Via Rosellini 12
20124 Milano

Per finire aggiungo una tabella con le POKE e le SYS da usare con i vari Basic.

		POKE	
	SYS	HTAB	VTAB
Basic 1 (2001)	58843	226	245
Basic 2 (3032)	57949	198	216
Basic 4 (4032)	57471	198	216
Basic 4 (8032)	57455	198	216

I cambiamenti vanno fatti alle linee 370, 390, 450, 540, 1030, 1240, 1410, 1460, 1540, 1550, 1610. ■

Grafici sempre a portata di mano

Un programma che permette di creare una "shape table" con il minimo sforzo

di *Steve Brown*

Uno dei motivi che mi ha spinto a comperare l'Apple è la sua capacità grafica ad alta risoluzione. Ma il metodo di creazione delle tabelle delle figure (*shape table*) proposto dal manuale di Applesoft non mi è sembrato all'altezza. Per ovviare a ciò ho scritto un programma (vedi il listato) di cui elenco alcune caratteristiche.

- Riceve in input semplici istruzioni per il disegno; ad esempio TS (che traccia muovendosi verso sinistra) e NB (che si muove verso il basso senza disegnare).
- Traduce questi comandi nei byte binari necessari per tracciare la curva.
- Stampa i dati necessari per sviluppare la tabella delle figure.
- Memorizza su disco la tabella dei valori per poterla riutilizzare.

Il programma funziona su Apple II Plus 48 K.

Spiegazione delle istruzioni e routine

L'istruzione 100 pone HIMEM a 38000. In questa locazione inizia infatti l'area di lavoro in cui viene creata la tabella delle figure. L'istruzione 110 passa il controllo alla 1830, dove comincia l'inizializzazione del programma. Prima però vengono le due subroutine.

La prima, che va dalla riga 120 alla 260, traduce qualsiasi numero decimale contenuto nella variabile DN nel suo equivalente esadecimale a quattro byte, nella variabile HN$. Le istruzioni dalla 270 alla 470 effettuano l'operazione inversa, cioè convertono il numero esadecimale contenuto in HN$ nel suo equivalente decimale, nella variabile DN.

L'istruzione 1830 pulisce lo schermo, ed è la prima operazione di inizializzazione. Nelle righe 1840 e 1850 si predispongono X$ e Y$, variabili per il controllo della stampante. Le istruzioni 1870 e 1880 preparano l'introduzione della variabile SN, che dà il numero di figure della tabella. Questo numero deve essere compreso tra 1 e 25. Le istruzioni dalla 1960 alla 2010 introducono la variabile VL come stima del numero di istruzioni di *plotting* necessarie per realizzare la figura più grande tra quelle inserite dalla tabella. I valori accettati vanno da 0 a 1000, e 0 ha valore di default 200. Questo numero serve per dimensionare i vettori di lavoro. Nelle istruzioni dalla 2020 alla 2090 viene richiesto il nome della tabella. Le righe dalla 2100 alla 2130 dimensionano i vettori, che hanno i seguenti significati:

HD$ Conversione esadecimale
PV$ Input per i vettori di tracciamento
PL$ Tabella dei comandi di input

PO$ Rappresentazione binaria dei comandi di tracciamento
ID$ Creazione dell'indice della tabella
ID Conversione binaria
BT$ Creazione della tabella

La DATA della riga 2140 è usata dalle istruzioni che vanno dalla riga 2150 alla 2170 per introdurre i valori nei vettori HD$, PL$ e ID. Le istruzioni dalla 2180 alla 2210 convertono in esadecimale il numero di definizioni della tabella e pongono questo valore nell'indice (vettore ID$). Le istruzioni dalla 2230 alla 2260 attendono l'introduzione di una locazione iniziale di memoria per la tabella. Può trattarsi di un numero decimale oppure esadecimale (il numero termina con H, come 1DFCH). Quindi si sceglie la routine per la conversione adatta, e il valore decimale della locazione di partenza è posto nella variabile SXOS ed il suo equivalente esadecimale in SXOS$.

Così termina l'inizializzazione del programma. Poi passiamo all'istruzione 480, dove con un ciclo FOR-NEXT, che si basa sul numero di definizioni della tabella, comincia il programma principale.

Le righe dalla 490 alla 600 incrementano il contatore dei vettori di tracciamento, visualizzano le istruzioni per l'input ed accettano la variabile PV$(NN) come vettore di tracciamento. L'istruzione 610 verifica se il valore introdotto è 00, nel qual caso la definizione in questione è finita. La 620 controlla che il comando di tracciamento non sia stato introdotto alla rovescia (p. es. se al posto di TS è stato introdotto ST, il computer se ne accorgerà e cambierà l'ordine dei caratteri con l'istruzione 630). La riga 640 cambia i comandi di una sola lettera che non tracciano nei loro equivalenti a due lettere.

Nelle istruzioni dalla 650 alla 670 si verifica il vettore PL$ per controllare che sia stato introdotto un comando valido. Se così è stato, il programma salta alla riga 730. Se invece il comando non è valido, le righe dalla 680 alla 700 stampano un messaggio, producono un segnale sonoro d'allarme e ripassano

```
TAVOLA DI FIGURE --- RANOCCHIA: FIGURA NUMERO 1

                          ---------- DATI ---------      - LOCAZIONE -
 LINEA #   BYTE #        BINARI      DECIMALI    ESA      DECIMALE  ESA

     1         5         00110110        54       36        7680    1E00
     2         6         00101101        45       2D        7681    1E01
     3         7         01001100        76       4C        7682    1E02
     4         8         00000110        6        06        7683    1E03
     5         9         00000000        0        00        7684    1E04



LISTA DEI DATI

INIZIO DELLA TAVOLA 1DFC ESA / 7676 DEC

--- LOCAZIONE ---          ---- DATI ----
DECIMALE      ESA          DECIMALI   ESA

7676          1DFC            1       01
7677          1DFD            0       00
7678          1DFE            4       04
7679          1DFF            0       00

MEMORIA NECESSARIA 9 BYTE

LE LOCAZIONI DI MEMORIA E8 E E9 CONTENGONO GLI INDIRIZZI
DI INIZIO DELLA TAVOLA DI FIGURE

LOC E8 HEX / 232 DEC = FC HEX / 252 DEC
LOC E9 ESA / 233 DEC = 1D ESA / 29 DEC


PER COMPLETARE LE OPERAZIONI DEL PROGRAMMA OCCORRE ESEGUIRE

BLOAD BTABLE,A7676

QUINDI

BSAVE RANOCCHIA,A7676,L9


PER USARE LA TAVOLA DI FIGURE IN UN PROGRAMMA BASIC
INSERIRE LE SEGUENTI ISTRUZIONI NEL PROGRAMMA
PRIMA DI INIZIARE AD USARE IL COMANDO 'HGR'.

PRINT CHR$(4);"BLOAD RANOCCHIA"
POKE 232,252
POKE 233,29

OCCORRE PROTEGGERE LE LOCAZIONI DA 7676 (1DFC ESA) A 7684 (1DFCESA)
CON LE OPPORTUNE ISTRUZIONI DI HIMEM E LOMEM
```

Tabella 1. *Lo stampato fornito dal programma.*

il controllo alla riga 650 per un altro input.

L'istruzione 730 elimina qualsiasi messaggio eventualmente rimasto. La riga 740 definisce il binario corrispondente al comando introdotto nel vettore PV$. Con l'istruzione 750 si ritorna all'input del prossimo vettore di tracciamento.

L'istruzione 760 dà inizio alla conversione dei singoli comandi binari nella definizione finale della figura, azzerando l'ultimo elemento nel vettore PV$. Ciò serve ad assicurare che il vettore alla fine della definizione termini nel modo giusto. Le variabili BC e BD vengono azzerate nell'istruzione 760. BC funge da contatore di byte; BD verrà usato più avanti nel programma.

L'istruzione 770 inizia un ciclo FOR-NEXT per leggere il vettore PV$ con salti di tre elementi. La 780 incrementa il contatore di byte. L'istruzione 790 crea il prossimo elemento del vettore BT$ con una rappresentazione a nove caratteri dei prossimi tre vettori di plotting. Se un byte fosse composto da nove bit, il programma potrebbe continuare con i tre elementi di input successivi.

Siccome non è così, il programma deve controllare se il primo carattere è un 1. Se è così, l'istruzione 840 annulla le prime tre posizioni; poi le righe 850 e 860 decrementano il contatore di byte. Così si passa al prossimo byte libero nella tabella.

L'istruzione 870 assegna al byte

binario finale gli otto caratteri più a destra. L'istruzione 880 chiude il ciclo FOR-NEXT. Le righe dalla 890 alla 930 assicurano di nuovo che il byte finale della definizione della figura contenga tutti zeri.

Con le istruzioni dalla riga 940 alla 1010 la routine di stampa comincia a scrivere le intestazioni. Le righe dalla 1020 alla 1220 stampano le singole locazioni di memoria ed i dati associati. L'istruzione 1230 memorizza l'equivalente decimale dei tre vettori di plotting. La 1240 completa il ciclo FOR-NEXT e torna a calcolare i tre vettori di tracciamento successivi.

Le istruzioni 1250 e 1260 spengono la stampante. Le righe dalla 1270 alla 1310 calcolano il valore indice per iniziare la prossima definizione di figura. L'istruzione 1320 completa il ciclo FOR-NEXT iniziato alla riga 480 e torna all'input della prossima definizione di figura.

Le istruzioni dalla 1330 alla 1790 stampano l'indice della tabella dei valori. Le istruzioni dalla 1800 alla 1820 completano il programma archiviando la tabella delle figure come file binario su disco.

C'è un ultimo passo da fare. Poiché la tabella dei valori è registrata con il nome BTABLE alla locazione di memoria 38000, dovete caricarla nella giusta locazione e poi salvarla su disco con il suo vero nome. I comandi per fare questo sono illustrati nell'esecuzione dimostrativa.

La tabella delle figure può ora essere usata in qualsiasi programma. Anche le istruzioni Basic necessarie per caricarla dall'interno di un programma sono presenti nell'esecuzione di prova.

Come funziona il programma

Quando si esegue il programma, bisogna fornire il numero di figure presenti nella tabella. Scrivete un numero compreso tra 1 e 25. (Si potrebbe modificare il programma per poter considerare un numero più grande.) Vi sarà poi richiesto l'input di una stima sul numero di

Listato 1. *Programma che crea tabelle di figure ad alta risoluzione con un Apple II Plus 48 K.*

```
100   HIMEM: 38000
110   GOTO 1830
120   D1=0: D2=0: D3=0: D4=0: D5=DN
130   IF DN<16 THEN 210
140   IF DN<256 THEN 200
150   IF DN<4096 THEN 170
160   D1=INT(DN/4096)
170   D5=D5-(D1*4096)
180   D2=INT(D5/256)
190   D5=D5-(D2*256)
200   D3=INT(D5/16)
210   D4=D5-(D3*16)
220   HN$=HD$(D1+1)+HD$(D2+1)+HD$(D3+1)+HD$(D4+1)
230   IF LEN(HN$)=4 THEN 260
240   HN$="0"+HN$
250   GOTO 230
260   RETURN
270   IF LEN(HN$)=4 THEN 300
280   HN$="0"+HN$
290   GOTO 270
300   FOR X=1 TO 16
310   IF MID$(HN$,4,1)=HD$(X) THEN 330
320   NEXT X
330   D4=X-1
340   FOR X=1 TO 16
350   IF MID$(HN$,3,1)=HD$(X) THEN 370
360   NEXT X
370   D3=X-1
380   FOR X=1 TO 16
390   IF MID$(HN$,2,1)=HD$(X) THEN 410
400   NEXT X
410   D2=X-1
420   FOR X=1 TO 16
430   IF MID$(HN$,1,1)=HD$(X) THEN 450
440   NEXT X
450   D1=X-1
460   DN=(D1*4096)+(D2*256)+(D3*16)+D4
470   RETURN
480   FOR DA=1 TO SN
490   NN=0
500   HOME
510   NN=NN+1
520   VTAB 1: HTAB 2: PRINT"TAVOLA DI FIGURE ";SLNAME$: PRINT
      " FIGURA NUMERO ";DA
530   VTAB 4: HTAB 2: PRINT"BATTI IL PROSSIMO VETTORE": PRINT
      " DI TRACCIAMENTO:"
540   VTAB 6: HTAB 2: PRINT"'00' PER TERMINARE QUESTA FIGURA"
550   VTAB 10: HTAB 2: PRINT
      "BATTI 'T' PER TRACCIARE O 'N' PER NON": PRINT
      " TRACCIARE, SEGUITO DALLA"
560   VTAB 12: HTAB 2: PRINT
      "DIREZIONE IN CUI MUOVERE (A,B,S,D)."
570   VTAB 14: HTAB 2: PRINT"(SE E' BATTUTA SOLO LA DIREZIONE":
      PRINT" VIENE ASSUNTO 'NESSUN TRACCIAMENTO')."
580   VTAB 18: HTAB 4: PRINT"PIU' COMANDI DI NON TRACCIAMENTO"
590   VTAB 19: HTAB 4: PRINT"VERSO L'ALTO POSSONO PROVOCARE UN
          ERRORE IN QUESTA DEFINIZIONE DI          FIGURA"
600   VTAB 22: HTAB 2: INPUT PV$(NN)
610   IF PV$(NN)="00" THEN 760
620   IF MID$(PV$(NN),2,1)<>"T" THEN 640
630   PV$(NN)=MID$(PV$(NN),2,1)+MID$(PV$(NN),1,1)
640   IF LEN(PV$(NN))<2 THEN PV$(NN)="N"+PV$(NN)
650   FOR X=1 TO 13
660   IF PV$(NN)=PL$(X) THEN 730
670   NEXT X
680   VTAB 22: HTAB 2: PRINT PV$(NN);" INPUT NON VALIDO"
690   FOR TK=1 TO 50
700   TL=PEEK(-16336)
710   NEXT TK
720   GOTO 600
730   VTAB 22: HTAB 2: PRINT"                                  "
740   PV$(NN)=PO$(X)
750   GOTO 510
760   PV$(NN)="000": BC=0: BD=0
770   FOR RX=1 TO NN STEP 3
780   BC=BC+1
790   AA$=PV$(RX+2)+PV$(RX+1)+PV$(RX)
800   BT$(BC)=AA$
```

(segue)

vettori di tracciamento nella definizione della figura più grande che intendete elaborare.

Qualsiasi numero tra 0 e 1000 va bene. Se si introduce zero, viene assunto il valore 200, che è sufficiente per la maggior parte delle figure semplici.

Poi servirà il nome della tabella delle figure. È permesso qualsiasi nome. Il prompt successivo chiederà una locazione di memoria iniziale per la tabella delle figure. Potete scrivere qualsiasi numero, tenendo conto però che la tabella non venga poi modificata da altri programmi, maschere per video, o altro. Il numero può essere introdotto in notazione decimale (p. es. 7676) o esadecimale, posponendovi una H (p. es. 1DFCH).

Quindi il programma chiede l'introduzione effettiva dei vettori di tracciamento. I comandi per questo sono:

TS Traccia verso sinistra
TD Traccia verso destra
TB Traccia verso il basso
TA Traccia verso l'alto
00 Fine di questa definizione di figura
NS (o S) Spostamento a sinistra (senza tracciare)
ND (o D) Spostamento a destra (senza tracciare)
NB (o B) Spostamento verso il basso (senza tracciare)
NA (o A) Spostamento verso l'alto (senza tracciare)

Questa fase si protrarrà finché non saranno definite tutte le figure. Voglio ancora sottolineare la necessità di evitare il caso in cui un byte risulti formato solo da zeri. Ciò provoca l'interruzione immediata del disegno. La causa principale di questo è l'introduzione di molti comandi di spostamento a penna alta. Vi suggerisco di partire dal punto più alto della curva e di procedere verso il basso. Se siete costretti a muovervi verso l'alto, fatelo con traiettoria a zig zag.

Quando questa operazione sarà terminata, i dati verranno convertiti, stampati e memorizzati su disco come file binario. Quindi dovrete caricare la tabella nella sua locazio-

Listato 1. *(segue).*

```
810   IF LEFT$(BT$(BC),1)="1" THEN 840
820   IF MID$(BT$(BC),2,2)="00" THEN 840
830   GOTO 870
840   BT$(BC)="000"+MID$(BT$(BC),4,6)
850   IF MID$(BT$(BC),4,3)="000" THEN RX=RX-1
860   RX=RX-1
870   BT$(BC)=MID$(BT$(BC),2,8)
880   NEXT RX
890   IF LEN(BT$(BC))=8 THEN 920
900   BT$(BC)="0"+BT$(BC)
910   GOTO 890
920   BC=BC+1
930   BT$(BC)="00000000"
940   PRINT X$;"PR#1"
950   PRINT Y$;"80N"
960   PRINT" "
970   PRINT"TAVOLA DI FIGURE --- ";SLNAME$;": FIGURA NUMERO ";
      DA
980   PRINT" "
990   PRINT"                              ";
995   PRINT"--------------- DATI ---------";
996   PRINT"      - LOCAZIONE -"
1000  PRINT"  LINEA #  BYTE #       BINARI       DECIMALI       ESA
            DECIMALE  ESA"
1010  PRINT" "
1020  FOR CX=1 TO BC
1030  BD=0: BX=0: BY=0
1040  BX$=LEFT$(BT$(CX),4): BY$=RIGHT$(BT$(CX),4)
1050  FOR IQ=1 TO 4
1060  BX=BX+(ID(IQ)*VAL(MID$(BX$,IQ,1)))
1070  BY=BY+(ID(IQ)*VAL(MID$(BY$,IQ,1)))
1080  NEXT IQ
1090  DN=BX
1100  GOSUB 120
1110  BD$=MID$(HN$,4,1)
1120  DN=BY
1130  GOSUB 120
1140  BE$=MID$(HN$,4,1)
1150  BF$=BD$+BE$
1160  HN$=BF$
1170  GOSUB 270
1180  BD=DN
1190  DN=SXOS+CX+IX-1
1200  GOSUB 120
1210  SYO$=HN$
1220  PRINT TAB(6);CX;TAB(9-LEN(STR$(CX)));CX+IX;
      TAB(10-LEN(STR$(CX+IX)));BT$(CX);TAB(9);BD;
      TAB(12-LEN(STR$(BD)));BF$;TAB(8);SXOS+CX+IX-1;
      TAB(11-LEN(STR$(SXOS+CX+IX-1)));SYO$
1230  POKE 38000+CX+IX-1,BD
1240  NEXT CX
1250  PRINT Y$;"40N"
1260  PRINT X$;"PR#0"
1270  DN=IX
1280  GOSUB 120
1290  ID$(3+(DA-1)*2)=RIGHT$(HN$,2)
1300  ID$(4+(DA-1)*2)=LEFT$(HN$,2)
1310  IX=IX+BC
1320  NEXT DA
1330  PRINT X$;"PR#1"
1340  PRINT Y$;"80N"
1350  PRINT" "
1360  PRINT"LISTA DEI DATI"
1370  PRINT" "
1380  PRINT"INIZIO DELLA TAVOLA ";SXOS$;" ESA / ";SXOS;" DEC"
1390  PRINT" "
1400  PRINT"--- LOCAZIONE ---          ---- DATI ----"
1410  PRINT"DECIMALE     ESA          DECIMALI   ESA"
1420  PRINT" "
1430  FOR QX=1 TO(2*SN)+2
1440  HN$=ID$(QX)
1450  GOSUB 270
1460  I1=DN
1470  DN=SXOS+QX-1
1480  GOSUB 120
1490  I2$=HN$
1500  PRINT SXOS+QX-1;TAB(12-LEN(STR$(SXOS+QX-1)));I2$;TAB(15);
      I1;TAB(10-LEN(STR$(I1)));ID$(QX)
1510  POKE 38000+QX-1,I1
1520  NEXT QX
```

(segue)

Listato 1. *(segue).*

```
1530  PRINT" "
1540  PRINT"MEMORIA NECESSARIA ";IX;" BYTE "
1550  PRINT" "
1560  PRINT"LE LOCAZIONI DI MEMORIA E8 E E9 CONTEN
      GONO GLI INDIRIZZI"; PRINT
      "DI INIZIO DELLA TAVOLA DI FIGURE"
1570  PRINT" "
1580  HN$="00E8": GOSUB 270: D8=DN
1590  HN$="00E9": GOSUB 270: D7=DN
1600  HN$="00"+MID$(SXOS$,3,2): GOSUB 270: D6=DN
1610  HN$="00"+MID$(SXOS$,1,2): GOSUB 270: D9=DN
1620  PRINT"LOC E8 HEX / ";D8;" DEC = ";
      MID$(SXOS$,3,2);" HEX / ";D6;" DEC"
1630  PRINT"LOC E9 ESA / ";D7;" DEC = ";
      MID$(SXOS$,1,2);" ESA / ";D9;" DEC"
1640  PRINT" ": PRINT" "
1650  PRINT"PER COMPLETARE LE OPERAZIONI DEL PROGR
      AMMA OCCORRE ESEGUIRE"
1660  PRINT" "
1670  PRINT"BLOAD BTABLE,A";SXOS: PRINT
1680  PRINT"QUINDI": PRINT
1690  PRINT"BSAVE ";SLNAME$;",A";SXOS;",L"IX:
      PRINT: PRINT
1700  PRINT"PER USARE LA TAVOLA DI FIGURE IN UN PR
      OGRAMMA BASIC"
1710  PRINT
      "INSERIRE LE SEGUENTI ISTRUZIONI NEL PROGRAM
      MA"
1720  PRINT
      "PRIMA DI INIZIARE AD USARE IL COMANDO 'HGR'
      ."
1730  PRINT" "
1740  PRINT"PRINT CHR$(4);";CHR$(34);"BLOAD ";
      SLNAME$;CHR$(34)
1750  PRINT"POKE 232,";D6
1760  PRINT"POKE 233,";D9
1770  PRINT" "
1775  DN=SXOS+IX-1: GOSUB 120
1780  PRINT"OCCORRE PROTEGGERE LE LOCAZIONI DA ";
      SXOS;" (";SXOS$;" ESA) A ";SXOS+IX-1;" (";
      SXOS$;"ESA)"
1790  PRINT
      "CON LE OPPORTUNE ISTRUZIONI DI HIMEM E LOME
      M"
1800  PRINT X$;"PR#0"
1810  PRINT CHR$(4)"BSAVE BTABLE,A38000,L";IX
1820  END
1830  HOME
1840  X$=CHR$(13)+CHR$(4)
1850  Y$=CHR$(9)
1860  HOME
1870  VTAB 8: PRINT
      "BATTI IL NUMERO DI FIGURE DI QUESTA      TAV
      OLA ";
1880  PRINT"(DA 1 A 25) ";
1890  INPUT SN
1900  IF SN>0 THEN 1920
1910  GOTO 1930
1920  IF SN<26 THEN 1950
1930  VTAB 13: HTAB 10: PRINT SN;
      " NON E' PERMESSO"
1940  GOTO 1890
1950  VTAB 13: HTAB 10: PRINT
      "                    "
1960  HOME
1970  VTAB 9: HTAB 2: PRINT
      "QUANTI VETTORI DI TRACCIAMENTO"
1980  VTAB 10: HTAB 2: PRINT
      "CI SONO APPROSSIMATIVAMENTE NELLA": PRINT
      " FIGURA PIU' GRANDE DA DEFINIRE?"
1990  VTAB 12: HTAB 2: PRINT"(MASSIMO 1000)"
2000  VTAB 14: HTAB 2: PRINT
      "CON '0' VIENE ASSUNTO 200"
2010  VTAB 16: HTAB 2: INPUT VL
2020  IF VL=0 THEN VL=200
2030  IF VL<1001 THEN 2060
2040  VTAB 18: HTAB 2: PRINT VL;
      " E' UN INPUT ERRATO"
2050  GOTO 2010
2060  VTAB 18: HTAB 2: PRINT
      "                    "
2070  HOME
2080  VTAB 8: HTAB 2: PRINT
      "BATTI IL NOME DA ASSEGNARE ": PRINT
      " A QUESTA TAVOLA DI FIGURE"
2090  VTAB 11: HTAB 2: INPUT SLNAME$
2100  DIM HD$(16)
2110  DIM PV$(VL+INT(VL/10)+10+(2*SN))
2120  DIM PL$(13),PO$(13),ID$((SN*2)+4)
2130  DIM ID(4),BT$(INT(VL/2)+INT(VL/10))
2140  DATA 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F,00,000,
      TA,100,TD,101,TB,110,TS,111,NA,000,ND,001,
      NB,010,NS,011,A,000,D,001,B,010,S,011,8,4,2,
      1
2150  FOR X=1 TO 16: READ HD$(X): NEXT X
2160  FOR X=1 TO 13: READ PL$(X): READ PO$(X):
      NEXT X
2170  FOR X=1 TO 4: READ ID(X): NEXT X
2180  DN=SN
2190  GOSUB 120
2200  ID$(1)=MID$(HN$,3,2)
2210  ID$(2)="00"
2220  HOME
2230  VTAB 6: HTAB 2: PRINT
      "BATTI L'INIZIO DELLA LOCAZIONE DI": PRINT
      " MEMORIA DI QUESTA TAVOLA DI FIGURE"
2240  VTAB 10: HTAB 2: PRINT
      "BATTI SOLO IL NUMERO SE E' DECIMALE"
2250  HTAB 2: PRINT
      "O TERMINALO CON UNA 'H' SE ESADECIMALE"
2260  VTAB 14: HTAB 2: INPUT SXOS$
2270  IF RIGHT$(SXOS$,1)="H" THEN 2330
2280  SXOS=VAL(SXOS$)
2290  DN=SXOS
2300  GOSUB 120
2310  SXOS$=HN$
2320  GOTO 2370
2330  SXOS$=MID$(SXOS$,1,LEN(SXOS$)-1)
2340  HN$=SXOS$
2350  GOSUB 270
2360  SXOS=DN
2370  IX=2+(2*SN)
2380  GOTO 480
2390  END
```

ne di memoria e registrarla col nome giusto. Tutti i comandi necessari a questo scopo sono elencati nella stampa di output, assieme alle istruzioni Basic per caricare la tabella dall'interno di un programma.

Vi raccomando di usare carta adatta per la stesura dei grafici; così aumenta la velocità d'esecuzione.

Questo programma non risolve tutti i problemi di alta risoluzione dell'Apple, ma è un utile mezzo per chi ogni tanto deve crearsi delle tabelle di valori. Io l'ho usato per costruire molte tabelle, da quelle relative alle più semplici figure geometriche a quelle per un completo set di caratteri. Modificando alcune dimensioni dei vettori, potete fare qualsiasi grafico. ■

Questo programma è disponibile su disco per l'Apple. Vedete nelle ultime pagine il "Servizio programmi".

PICCOLI ANNUNCI

Commodore VIC 20

Gestione magazzino per VIC 20+drive+printer+8 K RAM, 2400 articoli, altissima velocità di lavoro, stampa di inventari anche con valore. Disco programma L. 50000, disco archivio cadauno L. 25000. Ferruccio Dabene, piazza Bengasi, 27, 10024 Moncalieri (TO), tel. 011-6060253

Cambio per VIC 20 programmi di vario genere e esperienza sulla grafica ad alta risoluzione. Cercasi programmi (max 6 K) per ricetrasmissioni in RTTY con il VIC 20; per ulteriori informazioni scrivete, risponderò a tutti. Roberto Osselladore, via Fausta 136/A int. 5, 30010 Ca Savio (VE)

Commodore PET/CBM

Attenzione! pettomani parmensi: sono un ragazzo di 18 anni e cerco megapatiti (molto gradite le megapatite) per sviluppare software Basic/L.M. Il mio sistema: 3032/3040/4022. Fabio Curati, via Montebello 41, 43100 Parma, tel. 56127

Vendo cassetta contenente 20 giochi americani di cui 3 sonori a L. 26000 anticipate oppure L. 30000 contrassegno. Inoltre listato per word processor a L. 15000 il tutto per il PET, specificare modello. Luigi Menghi, via Maranello 7, 00125 Acilia (Roma), tel. 06-6069235

Sinclair ZX80-ZX81-Spectrum

Vendo programmi per ZX Spectrum 16/48 K a prezzi stracciati (massimo L. 12000 per un programma da 48 K) tra cui i favolosi Scacchi a 10 livelli, Gulpman, 3D e Startrek 48 K. Richiedere elenco con più di cento programmi in continuo aggiornamento, allegando L. 1000 in francobolli, per poter poi scegliere i programmi preferiti (su cassetta o su listato) che verranno poi spediti in un nastro personalizzato. Luigi Mongardi, via Prov. Selice 16/C, 40026 Imola (BO)

Cerco possessori del computer ZX81 per scambio software e programmi. Eventualmente anche qualcuno che conosca già lo ZX Spectrum. Risposta assicurata a tutti. Riccardo Taccia, via Redi, 53, 57100 Livorno

Vendo scambio software per ZX80/81 8 K ROM, 1/16 K RAM. Scrivere inviando elenco o telefonare ore pasti a Sandro Romano, via Adelasia 11, 07046 Porto Torres (SS), tel. 079-514501

Importante! Possessori di ZX81 con espansioni di E.2000 e non, si sta formando un club nella zona Liguria, Genova, il cui scopo principale è uno scambio di informazioni e di esperienze. Larghi contatti anche con altri club fondati in tutta Italia. Compu. ZX Club, Frangioni Luca, P. Giaccone 7, 16100 Genova (Porto)

Sinclair club costituito da utenti ZX per scambio idee e esperienze hardware. L'adesione del costo di L. 18000 dà diritto a ricevere un bollettino trimestrale e a facilitazioni varie. Sinclair Club, via Molino Vecchio 10/F, 40026 Imola (BO)

Vendo o scambio software per ZX81 1K/16 K. Tanti e fantastici per tutti i gusti ed età. Richiedi l'elenco inviando L. 500 in francobolli. Risposta assicurata. Fausto Lovisolo, piazzale Barsanti 5, 21052 Busto Arsizio (VA), tel. 0331-626543 (ore pasti)

Apple II

Acquisto programmi tecnici per geometri, ingegneri, periti, architetti solo se aperti, soprattutto se con grafica per Apple II e personal computer di larga diffusione. Fornire listino e prezzi. C.T.E. Geom. Pucci Mario, via Centrale Umbra 36, 06038 Spello (PG), tel. 652222

Cerco copia istruzioni del programma "Flight Simulator" per Apple II. Gabriele Scanavino, via M. Grosso 24, 12010 Entracque (Cuneo), tel. 0171-978171

Texas TI99/4A

Scambio programmi di ogni genere ma soprattutto cerco programmi di giochi per TI99/4A. Alessandro De Chiara, via Niccolini 10, 50121 Firenze, tel. 663622

Cerco programmi per il nuovo computer della Texas Instruments TI99/4A per acquistarli o per scambiarli. Luciano Pezza, via 24 maggio 35, 31015 Conegliano (TV), tel. 31124

Diversi

Cambio Visicalc con programmi calcolo cemento armato. Giuseppe Monte, via Toro 6, 20060 Cassina de' Pecchi (MI), tel. 02-9521082

Scambio 2000 francobolli commemorativi mondiali diversi con programmi e giochi Sharp MZ80, VIC 20, Texas, Atari, Activision, Apollo, Hanimex e Philips. I francobolli sono tutti di gran valore! Federico Violin (anni 12), via Breo 58, 35028 Piove di Sacco (PD), tel. 049-5840893

Scambio o vendo programmi per VIC 20 e BBC, scambio anche idee hardware per BBC e posso procurare espansioni di memoria e interfacce. Lucio Rota, via V. Levi 5, 42100 Reggio Emilia, tel. 0522-30155

Hardware

Vendo CBM 4040 floppy disk usato pochissimo. Vendo anche progr. per gestione archivio dati (log. radioamatori, elenchi clienti, telef. biblioteca ecc.) con defin. voci, ricerca veloce dati, aggiorn. ecc. (per 8032). Roberto Vendrame IN3VRR, via Maso della Pieve 72, 39100 Bolzano, tel. 0471-940615 (ore pasti), 41333 (int. 286 ufficio)

Sinclair ZX80 nuova ROM 8 K e 4 K RAM in valigetta completa, "30 programmi con lo ZX 80", "Programmazione dello Z80", Jackson vendo a L. 300000 preferibilmente in zona. Cartucce giochi VIC 20 L. 35000. Fabiano Cattaneo, via Moreschi 75, 22072 Cermenate (CO), tel. 031-771818

Cambio ZX80+16 K con monitor. Compro, vendo, cambio software TI99/4A. Salvatore Sbacchis, via Don G. Minzoni 2/E, 90143 Palermo, tel. 091-547670

Vendo micro N.E. composto da: LX380, LX382, LX383, LX384, LX385, LX387, LX388, LX392. Il tutto chiuso in contenitore a L. 700000. Telefonare ore ufficio al 4388-2677. Vendo inoltre LX386 a L. 100000. Ezio Gazzola, via Gavirate 16, 20148 Milano, tel. 406115

Vendo VIC 20 VCX-1001 con manuali (VIC revealed e Programmer's reference guide) ed una decina di giochi. Il tutto usato pochissimo a L. 500000. Luciano Zaccagni, via Zuccoli 19, 20145 Milano, tel. 6896639 (ore serali)

Compro registratore Commodore (per VIC 20) usato in buono stato. Michele Bergonzoni, via Siepelunga 58, 40141 Bologna, tel. 051-481141

Vendo Atari CX2600 nuovo imballato e perfetto, completo di cartuccia "Basic programming", tastiera per Basic e altre cartucce games con circa 100 giochi. Il tutto a L. 400000+spese spedizione. Piero Discacciati, via Paganini 28/B, 20052 Monza (MI), tel. 039-29412

Vendo Sinclair ZX81, 16 K byte RAM+un libro in francese "La pratique du ZX81"+software L. 400000. Bernard Van Der Stichele, via Pisistrato 50, 00124 Roma (Casalpalocco), tel. 6095986

Vendo ZX81 con espansione da 32 K+programmi vari+alimentatore, cavetti, etc. a L. 380000. Achille Lamma, via Opicina 3, 48100 Ravenna, tel. 0544-420782

Vendo ZX80 N. ROM+slow funzionante L. 170000. Scheda 32 K RAM per ZX80/81 senza integrati L. 35000 completa di zoccoli e compon. passivi. Scambio programmi per ZX80/81. Scrivere o telefonare ore pasti. Armando Pavese, via Cottolengo 59, 13051 Biella (VC), tel. 015-27353

ZX80 trasformato in 81. Slow funzionante + registratore inverse video tutto contenuto in elegante valigetta rigida+tastiera meccanica escludibile con connettore. Tutto a L. 300000. Ore ufficio. Pippo Musumeci, via Gravina 15, 95014 Giarre (CT), tel. 095/931361

Vendo Sinclair ZX81 più espansione 16 K alimentatore giochi vari cassetta L. 350000. Corrado Alba, via Montefollino 2, 00141 Roma, tel. 4751935

Acquisto contanti se vera occasione home computer Atari 800-810-410 anche software. Telefonare per offerta. Gino Piombino, via Marconi 7/24, 39100 Bolzano, tel. 0471-30160

Vendo Sinclair ZX80 corredato di interfaccia a funzione slow, nuova ROM 8 K, cavi, manuali, alimentatore e modifica per inverse video a L. 250000 in blocco. Vendo anche espansione 16 K RAM a L. 125000. Alessandro Bartolini, via 225ma (Monteverde) 6, 63100 Ascoli Piceno, tel. 0736-51412

I lettori che vogliono vendere, comperare o scambiare software, o desiderano dare informazioni possono compilare il tagliando pubblicato in fondo a questa rubrica. Il servizio è gratuito. La redazione si riserva il diritto insindacabile di rifiutare, sospendere o modificare qualsiasi inserzione.

Gli annunci sono riservati ai privati o ai club senza scopo di lucro.

Daremo la precedenza agli annunci che si riferiscono a software, programmi, libri e riviste, club per personal computer.

Un annuncio sarà più efficace se seguirete queste indicazioni:

- La *prima parola* deve essere esplicativa del vostro messaggio: scambio, vendo, compro, cerco... Per renderla più evidente la stamperemo in corsivo.
- Nel testo, riferitevi ad *un solo tipo* di computer (VIC 20, ZX80,...). L'annuncio apparirà sotto la testata relativa a quel computer. Se volete fare un annuncio per due o più computer, compilate due tagliandi.
- Date il vostro *recapito* con esattezza: nome e cognome, via e numero, *cap* e località, provincia, *prefisso* e numero telefonico.

Che cosa ha in più Personal Kid?

PREZZO (IVA escl.)

CPU BOARD 48 K RAM	650.000
Tastiera ASCII con pad numerico esteso e tasti funzionali	210.000
UNITÀ CENTRALE completa di alimentatore, tastiera ASCII dotata di pad numerico esteso e tasti funzionali, contenitore	
Con tastiera incorporata	1.210.000
Con tastiera separata	1.260.000

- Costo Basso
- Lettere minuscole
- Tastiera con pad numerico + i segni delle operazioni
- Repeat automatico
- Set di tasti funzionali per l'esecuzione immediata dei principali comandi
- Completo controllo del cursore
- Zoccolo per memoria EPROM
- Disponibilità del sistema in versione open frame o vestita in più configurazioni

Compatibile Apple

SIPREL s.r.l. Via Di Vittorio, 82 - Zona Ind. Baraccola 60020 - Candia di Ancona
ANCONA TEL. 071/6040305 - MILANO TEL. 02/487930 - BOLOGNA TEL. 051/346013 -
PESCARA TEL. 085/378195 - PISA TEL. 050/575480

Cercasi Concessionari

Vendo Texas TI99/4A+cavo per registratore ancora in garanzia a L. 500000. Telefonare ore pasti a: Gianmaria Gergori, via Volturno 9, 27100 Pavia, tel. 0382-36630

Vendo ROM 8 K+mascherina per ZX80 L. 40000. Interfaccia slow per trasformare ZX80 8 K in ZX81 (ancora da collegare, mai usata) L. 35000. Telefonare per accordi ore pasti. Paolo Ferrami, via Verdi 9, 26011 Casalbuttano, tel. 0374-60259

Vendo sistema Sinclair: ZX81+16 K+stampante+aliment.+manuali. Tutto nuovo (1 mese di vita) per Atari a Natale. Prezzo L. 510000 pagamento anticipato o contrassegno. Carta per stampante + programmi a richiesta. Fabio Berno, via S. Francesco 27, 20010 Pogliano (MI), tel. 02-9341921

Vendo Sinclair ZX80 1 K RAM 4 K ROM completo alimentatore con cavetti, manuali, imballo L. 110000. Telefonare Filippo Salvalaggio, Cannaregio 3541/B, 30121 Venezia, tel. 041-28364, ore pasti.

Vendo computer Video Genie 3003 16 K di RAM linguaggio Basic esteso livello II del TRS-80. Registratore per caricare programmi incorporato. Dotazione di numerosi programmi su cassette più libri con listati in italiano. L. 659000. Ermenegildo Crippa, via Milano 7/2, 22050 Lomagna (CO), tel. casa 039-58345 oppure 02-741390

Vendo causa passaggio ad altro sistema i seguenti materiali: PET/CBM 3032 L. 900000 e stampante Commodore 4022 a L. 800000. Marco Favuto, corso Adriatico 8, 10129 Torino, tel. 011-586080

Vendo VIC 20 più unità cassette C2N ottimo stato. Solo zona Viterbo o Perugia. Alessandro Cecchetti, via S. Lorenzo 43, 01100 Viterbo, tel. 0761-37454

Vendo TI 99/4A con Space invaders, manipolatori cavi collegamento 2 mesi di vita, perfette condizioni, L. 510000 trattabili causa passaggio sistema superiore. Telefonare ore pasti. Alberto Lo Paro, via del Popolo 2, 19100 La Spezia, tel. 0187-512198

Vendo ZX81+ZX printer+16 K Sinclair+3 rotoli carta a L. 520000. Acquistato ottobre '82 usato poco. Regalo programmi pagati L. 50000. Tel. sabato domenica. Paolo Nappo, via Vernilli Ciommi, 80047 S. Giuseppe Ves.no (NA), tel. 081-8282981

Vendo VIC 20+registratore C2N+superespansione (3 K+grafica)+3 K+ 8 K+supercabinet con 6 slots (per 6 cartridges) + 2 games + programmers'aid + potente word processor "Heswriter" su ROM. Il tutto (del valore di 1700000) a L. 1300000 trattabili. Paolo Barberis, via Indipendenza 57, 19020 Ceparana (SP), tel. 0187-933620

Vendo schede per computer N.E. Z80 LX381 LX82 LX83/4 LX86 completa di 8 K RAM LX388 LX390 tutte perfettamente funzionanti a L. 500000 + programmi vari in regalo. Tel. ore serali. Bruno Palladino, via Risorgimento 2, 20021 Bollate (MI), tel. 02-3502008

Svendo ZX81 + 32 K RAM+inverse video+beeper+alimentatore 2A+manuali italiano-inglese+appunti, fotocopie varie ZX81+sound board+ampl. per sound board con mother board al miglior offerente da L. 350000 in su. Marco Minucci, via Lepanto 111, 80125 Napoli, tel. 619927

Vendo TI59 con stampante PC100C con modulo ing. e 40 schedine imballo originale, manuali, software L. 420000. Marco Minucci, via Lepanto 111, 80125 Napoli, tel. 619927

Vendo VIC lightpen nuovissima (1 mese di vita) causa doppio regalo a L. 90000 Stefano De Santis, via Papa Giovanni XXIII 43, 20091 Bresso (MI), tel. 6108250

Vendo frequenzimetro LX358 di Nuova Elettronica mai usato e perfettamente funzionante. Paolo Geronazzo, via Don Formentini, 21010 Bosco Montegrino (VA), tel. 0332-589739

Studente informatica (4° anno) cerca serie ditte o privati per assemblaggio progetti elettronici a domicilio. Massima serietà e buona esperienza. Telefonare sabato o domenica sera. Paolo Geronazzo, via Don Formentini, 21010 Bosco Montegrino (VA) tel. 0332-589739

Vendo ZX81 costituito da ZX80+slow+8 K ROM, comprato gennaio '83, L. 150000. Massimo Zezza via Quarenghi 45, 20151 Milano, tel. 02-3535489

Cerco espansione di memoria collegabile direttamente al PET 2001 Commodore oppure schema documentato per espansione diretta su circuito stampato. Telefonare al martedì mattina. Francesco Venturelli, via Verdi 252, 41059 Zocca (MO), tel. 059-987909

Vendo VIC 20 completo espansione alta risoluzione+3 K RAM ed espansione 16 K RAM+decine di programmi di giochi, utilities, ingegneria e matematica. Eventualmente anche separati. Telefonare la sera. Daniele Azzaroli, Dorsoduro 203, 30123 Venezia, tel. 041-35675

Vendo tutte le schede del micro Nuova Elettronica a prezzo molto basso. È disponibile anche già montato e nuovo completo di software. Telefonare a Salvatore Ruggiero, via Mich. Schipa 61, 80122 Napoli, tel. 081-665633

PICCOLI ANNUNCI PERSONAL SOFTWARE

Sei un lettore di PERSONAL-SOFTWARE e vuoi entrare in contatto con tutti gli altri lettori per comperare, cambiare o vendere software? Spedisci questo tagliando a Gruppo Editoriale Jackson - Via Rosellini, 12 - 20124 Milano

Nome ..... Cognome .....

Via ..... C.A.P. .....

Città ..... Tel. .....

Servizio programmi

Per alcuni dei programmi pubblicati, *Personal Software* mette a disposizione dischi e nastri già registrati, realizzati in collaborazione con l'autore. Potete ottenerli in contrassegno, pagando direttamente al postino la cifra indicata, spedendo il tagliando pubblicato in fondo alla pagina.

N.	Sistema	Programmi	Supporto	pubblicato in *Personal Software* n.	Prezzo
1	Apple II+	La carta del cielo Collisione	floppy 5″ DOS 3.3	3 pag. 83 3 pag. 93	30.000
2	TRS-80 mod. I	Backgammon	floppy 5″ DOS 2.3	3 pag. 89	25.000
3	PET/CBM 3032/4032	Editor/Assembler in Basic	floppy 5″ 3032/4032+3040/4040	2 pag. 33	40.000
4	Apple II+	Interi in precisione multipla	floppy 5″ DOS 3.3	4 pag. 17	40.000
5	PET/CBM 3032/4032	Gioco del calcio	floppy 5″ 3032/4032+3040/4040	4 pag. 67	25.000
6	Apple II +	Pretty Printer Shape Table	floppy 5″ DOS 3.3	5 pag. 27	30.000

Spedire in busta chiusa a

PERSONAL SOFTWARE
Servizio Programmi
Via Rosellini 12
20124 Milano

Inviatemi i seguenti dischi di *Personal Software*

n. _____

per un totale di lire _____
che pagherò al postino alla consegna del pacco.

_____
Cognome e nome

_____
Indirizzo

_____
Cap., Località

_____
Firma

CHI MI AIUTERA'
A FAR DIVENTARE GRANDE
LA MIA PICCOLA AZIENDA?

GGK

IL PERSONAL COMPUTER IBM IL TUO PICCOLO GRANDE AMICO.

Crescere da soli non è facile: quante volte hai desiderato l'aiuto di un amico?
Ora l'hai trovato: è il Personal Computer IBM.
Vedrai, imparerai da solo a dialogare con lui in poche ore. Darà sempre una risposta immediata ad ogni tua richiesta e, se vuoi, ti collegherà anche ad altri elaboratori.
Potrà aiutarti in tutte le tue attività.
Il Personal Computer IBM, infatti, controlla tutti quei lavori che, in un momento di crescita, rischierebbero di occuparti troppo tempo. Può fare di tutto: riceve dati, analizza, calcola, registra, stampa e, grazie alla sua potente memoria e ai minidischi, ti consente di gestire un'infinità di informazioni.
Ti sarà più facile formulare preventivi e offerte, senza perdere d'occhio il tuo margine di profitto.
Ogni cosa sarà più semplice, con un amico così.
Vuoi conoscerlo meglio? Rivolgiti ai concessionari IBM.
Avrai tutte le informazioni necessarie.

IBM

IBM Italia
Distribuzione Prodotti

IBM Italia
Distribuzione Prodotti

Unità di elaborazione: contiene un microprocessore a 16 bit - ROM di 40 Kbyte - RAM da 64 a 640 Kbyte - una o due unità minidischi oppure una unità minidischi e un disco fisso da 10 Mbyte - architettura aperta - 5 o 8 attacchi per unità periferiche. **Unità di espansione:** contiene uno o due dischi fissi da 10 Mbyte ciascuno (capacità massima del sistema: 21 Mbyte in linea) - 8 attacchi supplementari per unità periferiche. **Tastiera:** ultrapiatta - contiene un microprocessore. **Unità video:** visualizza 25 righe di 80 caratteri con una matrice di 7x9 punti. **Stampante:** grafica bidirezionale - velocità massima 80 cps - 12 tipi di caratteri - densità di stampa da 40 a 132 caratteri per riga. **Sistema operativo DOS:** contiene avanzate funzioni logiche per un'alta flessibilità operativa e una grande facilità d'uso.

Il Personal Computer IBM viene venduto attraverso una rete di Concessionari IBM, costituita a questo scopo da una nuova società, la IBM Italia Distribuzione Prodotti S.r.l. Per acquisti superiori alle 20 unità puoi rivolgerti anche alle filiali IBM. Per informazioni sugli indirizzi dei punti di vendita telefona al 02/21752360 oppure 06/54864962.

Apple continua a crescere.

Apple ha introdotto il concetto di personal in tutto il mondo. E in tutto il mondo Apple cresce. Cresce anche in Italia dove la Iret, che lo importa e ne cura l'assistenza, può oggi annunciare l'esistenza di una rete di vendita di oltre 300 centri specializzati che fanno di Apple il loro cavallo di battaglia.
E naturalmente crescono le vendite di Apple, perché il personal computing conquista piccole aziende, professionisti e privati. È facile prevedere quindi che Apple continuerà a crescere, anche perché l'unica cosa di Apple che non cresce sono i prezzi. (Chiedete l'offerta speciale ai nostri rivenditori).

apple Il Personal Computer

IRET INFORMATICA

Via Bovio, 5 - 42100 Reggio Emilia - Tel. 0522/32643 - TLX 530173 IRETRE