LA PRIMA RIVISTA EUROPEA DI SOFTWARE PER PERSONAL COMPUTER

PRIMO NUMERO

PERSONAL SOFTWARE

ANNO 1 N. 1 LUGLIO/AGOSTO L. 3.500

UNA PUBBLICAZIONE DEL GRUPPO EDITORIALE JACKSON

RICORSIVITA' IN BASIC

MASTER MIND E INTELLIGENZA ARTIFICIALE

CONVERSIONI GRAFICHE TRA TRS-80, APPLE E PET/CBM

VENTIQUATTRO MODI PER SCRIVERE UN CICLO

PROGRAMMI PER PET/CBM, APPLE, TRS-80, ATARI, ZX80

Spedizione in Abb. Postale Gruppo III/70

A warner Communications Company

COSA FA UN COMPUTER ATARI CON TUO FIGLIO?

Quali stati compongono gli U.S.A.? Come si dice 'Ci vediamo lunedì' in francese? Atari studia con tuo figlio nel modo più moderno e intelligente. Con Atari lui impara a risolvere problemi di ogni tipo: da come sarebbe andata a Waterloo se lui avesse diretto le operazioni, a come riprodurre un brano musicale, a come risolvere una equazione. E si prepara così alla nuova civiltà del computer.

Non per niente Atari è in dotazione in molte scuole di ogni tipo, come aiuto prezioso sia per gli studenti che per gli insegnanti.

Non avevate ancora pensato di regalare un computer a vostro figlio? Allora, pensate a un Atari. E' facile e veloce, non più grande di una macchina per scrivere e non più costoso di un hi-fi. Basta collegarlo a un qualsiasi apparecchio TV e Atari è pronto per funzionare. A casa vostra, magari anche per giocare con i videogames, ma naturalmente anche nel vostro studio o negozio, officina o laboratorio.

Perchè Atari sa fare molte cose, è il nuovo strumento di lavoro per chi vive la professione in modo moderno e aggiornato. Atari calcola, prevede, consiglia soluzioni, disegna grafici e figure, archivia, fattura... è il collaboratore-amico che vi può dare una mano ogni giorno.

Potete sceglierlo nel più agile modello 400 o, se avete problemi più complessi, nel sofisticato modello 800, fornito del potente sistema gestionale VisiCalc. E potete scegliere anche tra una vastissima gamma di accessori.

Chiedete un Atari in prova al vostro negoziante di fiducia, e dopo un giorno insieme vedrete come è utile, facile e, perchè no, affascinante possedere un Atari.

ATARI®
Computers for people.

DISTRIBUTORE ESCLUSIVO PER L'ITALIA
ADVEICO
CONSUMER DIVISION

ADVEICO S.p.A. - 20124 Milano - Via A. Tadino, 22, Tel. 02/2043281 - 43016 S. Pancrazio, Parma - Via Emilia Ovest, 129, Tel. 0521/998841

ELEDRA 3S S.p.A. Distributore per l'italia

apple computer presenta alcuni dei suoi corsi per PERSONAL COMPUTER:

P1 Introduzione ai Personal Computer

Durata: 1/2 giornata (ore 16,30)
Codice 39005050
Prezzo Lit. 30.000 + IVA 15%

Che cos'è un "personal computer", a cosa serve, cosa si usa e a quali moltepici problemi fornisce la soluzione. Una panoramica introduttiva al fenomeno del "personal computer" con particolare riferimento ai sistemi Apple II e III.

R1 Gestione delle Informazioni con Sistemi Apple Computer

Durata: 1 giornata
Codice 39005350
Prezzo Lit. 70.000 + IVA 15%

Come la massa di informazioni di ogni tipo di attività può essere organizzata, aggiornata e ritrovata in modo semplice e immediato con i personal computer Apple II e III. Una introduzione all'utilizzo dei programmi tipo "data-base" con particolare riferimento ai prodotti VISICALC/VISIPLOT/VISIFILE ecc.

S1 Apple][Plus e Linguaggio BASIC

Durata: 2 giorni
Codice 39006050
Prezzo Lit. 150.000 + IVA 15%

Esame delle caratteristiche base del sistema Apple II Plus e delle sue principali periferiche. Introduzione alle possibilità operative del medesimo con particolare riferimento alla programmazione con il linguaggio BASIC Applesoft e con il sistema operativo a dischetti DOS.

R1 Sistema APPLE /// e Linguaggio Business BASIC

Durata: 2 giorni
Codice 39006500
Prezzo Lit. 200.000 + IVA 15%

Introduzione alle caratteristiche base e alle molteplici possibilità operative del sistema Apple III e delle sue periferiche, incluso il disco rigido Profile.
Presentazione dei programmi applicativi disponibili, del linguaggio di programmazione Business BASIC e del sistema operativo SOS con particolare riferimento al "System Utility" per la definizione delle periferiche collegare al sistema.

W1 Programmazione PASCAL con Sistemi Apple Computer

Durata: 2 giorni
Codice 39006750
Prezzo Lit. 240.000 + IVA 15%

Introduzione ai fondamenti del linguaggio di programmazione strutturata UCSD PASCAL, che permette di ottenere programmi di più rapida esecuzione e più facilmente documentabili e modificabili.
Esposizione dei concetti base del sistema Apple PASCAL con particolare riferimento ai programmai Editor, Compiler, Assembler 6502, Linker e Filer, operanti su Personal Computer Apple II e III.

Per ogni tipo di informazione e per le iscrizioni ai corsi rivolgersi a:

edelektron srl
C.so Sempione, 39 - 20145 MILANO
Tel. (02) 34.93.603-31.85.678-31.85.571-34.90.176

Su AMITALIA il sole splende ALTOS, i nuovi microcomputers "anni luce" avanti. Su tutti.

* **CP/M, MP/M** sono marchi registrati della Digital Research.
* **OASIS** è un marchio registrato della Phase One.

AMITALIA, rappresenta in esclusiva per il mercato italiano una grande famiglia di microcomputers su singola scheda da 8 e 16 bit: gli ALTOS, protagonisti della microinformatica più avanzata, risultati di una tecnologia che viene dal domani per tutte le esigenze di mono e multiutenza di oggi. Microcalcolatori, gli ALTOS, che ricordano e parlano meglio di ogni altro tutte le lingue dell'informatica distribuita.
AMITALIA è anche un'organizzazione leader di distribuzione e assistenza che copre, con personale qualificato e specialistico, l'intero territorio nazionale.
Ma passiamo a conoscerli meglio tecnicamente questi microcomputers "anni luce" avanti su tutti.

ACS 8000 **MICROPROCESSORE 8 BIT** **SUPPORTO DI MEMORIA 8"** **FLOPPY E HARD DISK** **RICOVERO DATI SU CASSETTA MAGNETICA**	da 64 K RAM di memoria a 208 K RAM di memoria Floppy disk singola faccia doppia densità 0,5 MByte Dischi fissi da 10, 20, 40, 80 MByte in linea	Cassetta magnetica per ricovero dati da 17,5 MByte per multiutenza da 1 a 4 terminali Sistemi operativi: *CP/M, *MP/M, *OASIS
ACS 5 **MICROPROCESSORE A 8 BIT** **SUPPORTO DI MEMORIA 5 1/4"** **FLOPPY E HARD DISK**	196 K RAM di memoria Floppy disk doppia faccia doppia densità 1 MByte Dischi fissi da 5, 10, 20 MByte in linea da 1 a 3 terminali per multiutenza Sistemi operativi: *CP/M, *MP/M, *OASIS	Cassetta magnetica per ricovero dati da 17,5 MByte da 1 a 8 terminali per multiutenza Sistemi operativi: *CP/M-86, *MP/M-86, *OASIS-16, XENIX
ACS 8600 **MICROPROCESSORE A 16 BIT** **SUPPORTO DI MEMORIA 8"** **FLOPPY E HARD DISK** **RICOVERO DATI SU CASSETTA MAGNETICA**	da 500 a 1000 K RAM di memoria Floppy disk singola faccia doppia densità 0,5 MByte Dischi fissi da 10, 20, 40, 80 MByte in linea	

ALTOS COMPUTER SYSTEMS

AMITALIA, SAICO, SEGI: tre leader un gruppo, AMMI.

AMITALIA

ADVANCED MICROCOMPUTER ITALIA s.r.l.

20124 Milano - Via Volturno, 46 - Tel. (02) 683985 - 6881946 - 6898015
00142 Roma - Via B. Croce, 97 - Tel. (06) 5410620

Sommario

PERSONAL SOFTWARE

ANNO 1 N. 1 LUGLIO/AGOSTO 1982

RUBRICHE

ARTICOLI

PROGRAMMI

INIZIARE NEL MODO MIGLIORE

Guida al SINCLAIR ZX81 ZX80-Nuova ROM

Pagg. 262
Cod. 318B

L. 16.500
(Abb. L. 11.500)

IL LIBRO
Questa guida, con chiarezza, semplicità espositiva e ricchezza di esemplificazioni, risulta un vero e proprio strumento operativo per tutti coloro che vogliono avvicinarsi all'informatica in generale, e imparare la programmazione in BASIC, in particolare travalicando i tre calcolatori (ZX81, ZX80, ZX80 nuova ROM) a cui fa riferimento. Partendo da quello che è un computer, il lettore impara nei primi sei capitoli a programmare in BASIC, spingendosi, per chi lo vuole, oltre, sino alla programmazione in linguaggio macchina. L'ultimo capitolo riporta parecchi programmi e per ciascuno, vengono fornite, dove possibile, le diverse versioni. Tra l'altro si parlerà di file e di animazione delle figure. Per finire ben otto Appendici, essenziali ed utilissime, tra cui spiccano per interesse le due dedicate ai sistemi operativi dello ZX80, ZX80 nuova ROM e ZX81.

SOMMARIO
Introduzione - Il calcolatore - Installazione del calcolatore - La programmazione - Il linguaggio BASIC - Come operare - Utilizzo della memoria - Linguaggio macchina - Esempi di programmi ---- caratteri del sistema - variabili del sistema - scheda BASIC ZX80 - scheda BASIC ZX80 nuova ROM e ZX81 - errori segnalati dalla macchina - sistema operativo dello ZX80 - sistema operativo dello ZX81 e nuova ROM

Per ordinare il volume utilizzare l'apposito tagliando inserito in fondo alla rivista.

GRUPPO EDITORIALE JACKSON
Divisione Libri

Editoriale

Mauro Boscarol

Software vuol dire programmi?

Questa nuova rivista si chiama Personal Software. Tutti sanno in che senso è usato il termine "personal". Ma software cosa significa? Sfogliando le riviste specializzate si trovano appositi settori "riservati" al software. E in questi settori ci sono programmi, tanti programmi, per le varie macchine, nelle varie versioni, con l'elenco delle variabili, con l'elenco delle routine.

L'idea che ci fa, è che "software" sia qualcosa cui dedicare un settore a parte della rivista. Le cose "vere" sono le macchine, le stampanti, i plotter, le inchieste di mercato, le prove, i testi, la guida all'acquisto. A parte, c'è il software.

E ci si fa una seconda idea: che il software siano i programmi. Programmi quasi esclusivamente in Basic, talvolta qualcosa in linguaggio macchina, ma sapendo che il lettore lo guarda con sospetto.

Questa rivista è nata con la voglia di cambiare. Di far capire che il software non è "a parte". D'altronde tutti lo dicono: nei prossimi anni ci sarà la fame del software. Noi diciamo che i prossimi anni sono adesso, ed iniziamo a parlare solo di software, il nostro unico argomento.

Ed iniziamo a dire che il software non sono solo programmi Basic. Anche noi abbiamo una sezione di programmi Basic per le varie macchine nelle varie versioni. Ma parliamo anche di altro. Parliamo di come si fa il software, di come si scrive, di come si può valutare. Trattiamo il software come un argomento del quale si può diventare "esperti", sul quale ci si può fare una cultura. Pensate possibile essere esperti e colti in "programmi Basic per il CBM 3032"?

Per noi il software è arte della programmazione, intelligenza artificiale, analisi degli algoritmi. Non è troppo complicato. Il nostro linguaggio sarà piano e comprensibile. Non chiuderemo la porta in faccia a nessuno, faremo solo divulgazione culturale in questo particolare campo. E comunque alla fine ci sono sempre i programmi.

In questo primo numero, oltre ai programmi, trovate degli articoli e delle rubriche. Gli uni e le altre sono aperte ai lettori: invitiamo chiunque avesse qualcosa da dire o da scrivere a farlo subito. Le rubriche non hanno dei titolari fissi: di mese in mese pubblicheremo gli interventi che ci sembreranno più in tema, anche se le diverse opinioni potranno essere contrastanti.

Per gli articoli, l'unico tema da rispettare è il software: di base o applicativo, intelligenza artificiale o gestione aziendale, pubblicheremo tutto quello che riterremo interessante per i nostri lettori.

Gli articoli di questo numero toccano temi diversi. *Ventiquattro modi per scrivere un ciclo* è un bell'esempio di come deve venir condotta l'analisi di un algoritmo, che in questo caso è il più semplice che possa esistere: il ciclo appunto. L'articolo sul *Master Mind* rientra nel campo dell'intelligenza artificiale: come si può istruire il calcolatore a giocare una partita contro un "umano", in modo che sia stimolante e competitiva? Qui ci sono alcune semplici regole, e il tutto è una introduzione alle tecniche dell'AI (intelligenza artificiale). Un'altro tipo di tecnica, questa volta di programmazione, è spiegata in dettaglio in *Programmazione ricorsiva in Basic*. Si tratta appunto della ricorsività, implementata in molti linguaggi, per esempio il Pascal, l'Algol e il PL/1. In Basic non c'è, ma la si può fare lo stesso. E la si può applicare per esempio proprio all'AI, nella ricerca di in albero di gioco. Infine un ultimo articolo riguarda le *conversioni grafiche* tra i personal più diffusi. È una raccolta di informazioni, notizie e consigli per effettuare tali conversioni.

Su tutto questo aspettiamo il parere dei lettori. Dal prossimo numero ci sarà anche una rubrica di posta, sempre che i lettori ce lo permettano. Gradiremmo anche ricevere pareri negativi: vogliamo anche noi l'angolo del "Non ci sto!". ■

SCRIVETE UN PROGRAMMA DI SOFTWARE PER L'HOME COMPUTER COL MIGLIOR RAPPORTO PRESTAZIONI/PREZZO: IL TI 99/4A

Texas Instruments Vi offre oggi una splendida occasione. Basta semplicemente scrivere programmi di software.

Se volete contribuire alla veloce crescita del mercato del software per gli Home Computers, possiamo aiutarvi. Eccovi come.

Il computer TI-99/4A è un sofisticato calcolatore progettato non solo per il principiante che desidera la facilità di funzionamento, ma anche per il professionista. Alla base di ciò c'è la riprovata qualità del microprocessore a 16 bit TMS 9900, che lo rende uno dei più potenti e versatili microcomputers.

Il TI-99/4A offre una quantità di prestazioni che è difficile trovare in sistemi similari.

Il TI-99/4A dispone di una RAM interna di 16 k Byte, espandibile a 48 k Byte e ad una capacità combinata RAM/ROM di ben 110 k Byte. Il TI-99/4A è facilmente collegabile ad un qualsiasi normale televisore. Gli si possono inoltre connettere tante altre periferiche: stampante, disk driver, interfaccia RS 232 e sintetizzatore vocale. La sua tastiera professionale permette di scrivere sia con caratteri maiuscoli che con caratteri minuscoli.

Se poi aggiungete le capacità grafiche ad alta risoluzione con 32 caratteri su 24 righe a 16 colori (256 x 192 punti), 3 tonalità su 5 ottave più generatore di effetti sonori, i linguaggi di programmazione in BASIC , UCSD-PASCAL, TI-LOGO e ASSEMBLER, che potete usare da consolle o da periferiche standard, vi accorgerete di quanto il TI-99/4A superi la concorrenza. Specialmente se poi date un' occhiata al suo prezzo di 598.000 lire IVA esclusa.

Per risolvere i vostri problemi, potete usare facilmente la vasta gamma di programmi residenti su moduli «Solid State Software»® della Texas Instruments. Oltre 600 programmi di software sono già disponibili nel mondo.

Desideriamo espandere ulteriormente il software già esistente su moduli «Solid State Software»®, Cassette e dischi, con programmi promettenti, scritti da persone entusiaste. Ecco perché ci rivolgiamo a voi.

Poiché abbiamo riservato a questo programma di sviluppo del software solo un numero limitato di sistemi con TI-99/4A, affrettatevi a spedire oggi stesso il coupon che vi permetterà di conoscere in dettaglio le modalità di partecipazione.

Spedire a Texas Instruments Semiconduttori Italia S.p.A. Divisione Prodotti Elettronici Personali Software Rallye

Casella Postale 1 Cittaducale (Rieti)

Sono interessato al programma per l'Home Computer TI-99/4A della Texas Instruments.

Spedite, per favore, ulteriori informazioni a

Nome

Indirizzo

PSW

TEXAS INSTRUMENTS

Ventiquattro modi per scrivere un ciclo

In effetti sono 124 modi (o anche di più) il che illustra l'infinita varietà di cicli, la più semplice fra le tecniche di programmazione.

di *W.D. Maurer*

Per cominciare, osservate la tavola 1. Ci sono diverse cose relative a questa tavola che sono sorprendenti.

Tutto quello che dobbiamo fare è spostare alcune quantità da un posto ad un altro. Ci sono N quantità, chiamate P(1)...P(N). Dobbiamo spostarle in Q(1)...Q(N), cosa che facciamo, naturalmente, usando una istruzione FOR in Basic, così:

```
10 FOR J=1 TO N
20 Q(J)=P(J)
30 NEXT J
```

che dovrebbero avere un senso per voi anche se non conoscete il Basic (ma a patto che conosciate un po' di inglese). Ma cerchiamo di farlo senza usare l'istruzione FOR, per capire come funzionano i cicli; ed immediatamente ci troviamo davanti a un gran numero di possibilità. La tavola 1 mostra le prime ventiquattro. Sono tutte differenti; sono tutte corte; e fanno tutte esattamente la stessa cosa. (Be', non proprio esattamente la stessa cosa. Alla fine di qualcuna J è uguale a N+1; alla fine di altre J può essere uguale a N o a 1, o a 0. Ma poiché presumibilmente non useremo J dopo aver finito il ciclo, ciò non ha importanza).

Perché ci sono così tanti modi per scrivere un ciclo? Perché ci sono certe cose che possiamo fare in due o più modi.

Possiamo cominciare con Q(1) e andare fino a Q(N) (prima e seconda colonna nella tavola 1), oppure possiamo cominciare con Q(N) e scendere fino a Q(1) (terza e quarta colonna).

Abbiamo tre cose da fare dentro il ciclo: porre Q(J) uguale a P(J); incrementare (o decrementare) J di uno; e fare un test. Queste tre cose possono essere disposte in sei modi: chiamandole U, V e W, possiamo disporle come UVW, UWV, VUW, VWU, WUV o WVU. Ognuna di queste sei combinazioni corrisponde ad una delle sei righe della tavola 1.

Quello che può essere ora sorprendente relativamente alla tavola 1 sono i sottili modi in cui non tutto è perfettamente simmetrico. Per esempio, nell'ultima colonna, la variabile J è talvolta inizializzata a N e talvolta a N+1. Quest'ultimo caso richiede un po' più di tempo (a meno che N non sia veramente una costante, come 10). Nella prima colonna, J è talvolta posto uguale a 1 e talvolta a 0, il che su alcuni computer richiede un po' meno tempo che porlo uguale a 1.

Non ci vuole molto a capire, comunque, che nell'enumerare ventiquattro modi per scrivere un ciclo, abbiamo in realtà solo graffiato la superficie. Quali sono alcune delle altre cose che avremmo potuto fare?

Guardando in cima alla seconda

```
10 J=1
20 Q(J)=P(J)
30 J=J+1
40 IF J<>N+1 GOTO 20

10 J=1
20 Q(J)=P(J)
30 IF J=N GOTO 60
40 J=J+1
50 GOTO20

10 J=0
20 J=J+1
30 Q(J)=P(J)
40 IF J<>N GOTO 20

10 J=0
20 J=J+1
30 IF J=N+1 GOTO 60
40 Q(J)=P(J)
50 GOTO 20

10 J=1
20 IF J=N+1 GOTO 60
30 Q(J)=P(J)
40 J=J+1
50 GOTO 20

10 J=0
20 IF J=N GOTO 60
30 J=J+1
40 Q(J)=P(J)
50 GOTO 20

10 J=1
20 Q(J)=P(J)
30 J=J+1
40 IF J<=N GOTO 20

10 J=1
20 Q(J)=P(J)
30 IF J>=N GOTO 60
40 J=J+1
50 GOTO 20

10 J=0
20 J=J+1
30 Q(J)=P(J)
40 IF J<N GOTO 20

10 J=0
20 J=J+1
30 IF J>N GOTO 60
40 Q(J)=P(J)
50 GOTO 20

10 J=1
20 IF J>N GOTO 60
30 Q(J)=P(J)
40 J=J+1
50 GOTO 20

10 J=0
20 IF J>=N GOTO 60
30 J=J+1
40 Q(J)=P(J)
50 GOTO 20

10 J=N
20 Q(J)=P(J)
30 J=J-1
40 IF J<>0 GOTO 20

10 J=N
20 Q(J)=P(J)
30 IF J=1 GOTO 60
40 J=J-1
50 GOTO 20

10 J=N+1
20 J=J-1
30 Q(J)=P(J)
40 IF J<>1 GOTO 20

10 J=N+1
20 J=J-1
30 IF J=0 GOTO 60
40 Q(J)=P(J)
50 GOTO 20

10 J=N
20 IF J=0 GOTO 60
30 Q(J)=P(J9
40 J=J-1
50 GOTO 20

10 J=N+1
20 IF J=1 GOTO 60
30 J=J-1
40 Q(J)=P(J)
50 GOTO 20

10 J=N
20 Q(J)=P(J)
30 J=J-1
40 IF J>=1 GOTO 20

10 J=N
20 Q(J)=P(J)
30 IF J<=1 GOTO 60
40 J=J-1
50 GOTO 20

10 J=N+1
20 J=J-1
30 Q(J)=P(J)
40 IF J>1 GOTO 20

10 J=N+1
20 J=J-1
30 IF J<=0 GOTO 60
40 Q(J)=P(J)
50 GOTO 20

10 J=N
20 IF J<=0 GOTO 60
30 Q(J)=P(J)
40 J=J-1
50 GOTO 20

10 J=N+1
20 IF J<=1 GOTO 60
30 J=J-1
40 Q(J)=P(J)
50 GOTO 20
```

Tavola 1. *Ventiquattro modi per scrivere un ciclo in Basic. Questi ventiquattro metodi di base possono dar luogo, con una serie di piccoli cambiamenti a 124 diversi tipi di cicli.*

colonna, vediamo che stiamo verificando se J è minore o uguale a N. Avremmo potuto ugualmente verificare se J era minore di N+1. Lo stesso tipo di cambiamento poteva essere fatto nella seconda e quarta colonna, dando luogo a un totale di trentasei modi per scrivere un ciclo.

La maggior parte dei cambi di questo tipo ovviamente peggiora le cose. È chiaro che impiega più tempo un test su N+1 che un test su N, anche se mettiamo qualche nuova variabile come N1 uguale a N+1 prima del ciclo e quindi facciamo il test su N1 all'interno del ciclo. Una possibile eccezione a ciò è quella in cima all'ultima colonna, dove potremmo verificare se J è maggiore di 0. In linguaggio assembler su un sistema che si basa sul 6800 usando dati interi con segno su 8 bit, lo si può fare con una istruzione BGT (salta se è maggiore di zero) direttamente dopo il decremento.

Possiamo considerare la possibilità di porre Q(J+1) uguale a P(J+1) o Q(J−1) a P(J−1) piuttosto che Q(J) uguale a P(J). In alcuni casi questo accelera alcune delle operazioni del ciclo. Per esempio, in cima alla prima colonna, avremmo potuto porre J uguale a 0 e verificare J con N piuttosto che con N+1, se poniamo Q(J+1) uguale a P(J+1). Ambedue questi cambiamenti costituiscono miglioramenti dal punto di vista del tempo.

Può sembrare che ponendo Q(J+1) uguale a P(J+1) ci sia una perdita di tempo rispetto a porre Q(J) uguale a P(J). Ovviamente, non è così, o non dovrebbe essere così su un sistema ben costruito (sebbene possa essere così in alcune versioni di Basic). Una costante additiva o sottrattiva in un indice (come J+1) non deve essere calcolata. Per capire il perché di questo, abbiamo bisogno di alcune nozioni di linguaggio assembler; coloro che conoscono solo il Basic possono saltare il prossimo paragrafo, in cui viene data la spiegazione.

Ogni volta che facciamo riferimento a $t(v)$ per un vettore t e una variabile v, dobbiamo sommare il valore di v all'indirizzo di t. Sull'8080 questo viene fatto esplicitamente; tipicamente si fa un LXI H,t seguito da un DAD D dove il registro DE contiene v (cioè dove il registro E contiene v e il registro D contiene 0) e quindi possiamo fare riferimento a $t(v)$ facendo ADD M o MOV r,M o MOV M,r o quello che è.

in tutta Italia

LE NOSTRE MARCHE

Tandy

BMC

DAI THE MICROCOMPUTER COMPANY

VIC 20

Honeywell

sinclair

ARFON MICRO

SAMSUNG

ATARI®

SONY

GRUPPO EDITORIALE JACKSON

tanta informatica per tanti bit shop

BITSHOP PRIMAVERA è un'organizzazione che cura a livello nazionale una catena di Rivenditori Specializzati e Personalizzati per la vendita di: Personal computer, Stampanti, Floppy Disk, Terminali, Monitors, Calcolatrici Professionali, Giochi Scientifici, Mezzi Didattici per l'informatica.
BITSHOP PRIMAVERA: P.le Massari, 22
20125 MILANO - Tel. 6082255

I NOSTRI SHOP

ALESSANDRIA Via Savonarola, 13
AREZZO Via F. Lippi, 13
BARI Via Capruzzi, 192
BERGAMO Via F. D'Assisi, 5
BOLOGNA Via Brugnoli, 1/A
CAMPOBASSO Via Mons. Il Bologna, 10
CESANO MADERNO Via Ferrini, 6
COMO Via L. Sacco, 3
COSENZA Via dei Mille, 86
FAVRIA CANAVESE C.so Matteotti, 13
GALLARATE Via A. Da Brescia, 2
L'AQUILA Via Strada 85, 2
MESSINA Via Del Vespro, 71
MILANO Galleria Manzoni, 40
MILANO Via Petrella, 6
MILANO Via G. Cantoni, 7
MILANO P.zza Firenze, 4
MILANO Via Altaguardia, 2
MILANO V.le Corsica, 14
NOVARA Via Q. Sella, 32
PADOVA Via Fistomba, 8
PALERMO Via Lamarmora, 82
PARMA Via Borghesi, 16
PAVIA Via C. Battisti, 4 A
PESCARA Via Guelfi, 74
PISA Via XXIV Maggio, 101
PISTOIA V.le Adua, 350
POZZUOLI Via Pergolesi, 13
RIMINI Via Bertola, 75
ROMA Via Cerreto Da Spoleto, 23
ROMA P.zza San Dona di Piave, 14
SONDRIO Via Nazario Sauro, 28
TERAMO Via Martiri Pennesi, 14
TERNI Via P. Gori, 8
TORINO Via Chivasso, 11
TORINO C.so Grosseto, 209
TORINO Via Tripoli, 179
TRIESTE Via F. Severo, 138
VARESE Via Carrobbio, 13
VERONA Via Pontiere, 2
VIAREGGIO Via Volta, 79
VOGHERA P.zza Carducci, 11

IN FASE DI APERTURA

BASSANO DEL GRAPPA
BUSTO ARSIZIO
CAGLIARI
CATANIA
FIRENZE
FROSINONE
GENOVA
LATINA
LECCE
MESTRE
NAPOLI
PERUGIA
RAVENNA
REGGIO EMILIA
RIETI
ROMA
SALERNO
SANREMO
TORINO
UDINE
VENEZIA
VICENZA

con più computer

Sul 6502, questo viene fatto in hardware; si fa un LDA *t*,X o un STA *t*,X o un ADC *t*,X o quello che è, dove il registro X contiene *v*, ma l'hardware somma il registro X all'indirizzo dato nell'istruzione, il che in effetti somma *v* all'indirizzo di *t* in questo caso. Ogni microcomputer ha dettagli leggermente differenti, ma l'idea è la stessa in ogni caso. Ora supponete che desideriamo fare un riferimento a $t(v+k)$, dove k è una costante. Dobbiamo sommare il valore di $v+k$ all'indirizzo di t, il che è lo stesso di sommare il valore di v all'indirizzo di t più k. La questione è che sia k che t sono costanti; il loro valore non cambia durante l'esecuzione del programma. Quindi l'indirizzo di t più k è anche una costante e questa addizione può essere fatta prima che il programma parta. Sull'8080, per esempio, dovremmo solo dire LXI H,$t+k$ invece di LXI H,*t*; si tratta di una singola istruzione il cui campo indirizzo (secondo e terzo byte del codice d'istruzione) contiene la quantità a 16 bit ottenuta sommando k all'indirizzo di t. Sul 6502, possiamo dire LDA $t+k$,X e ancora abbiamo una singola istruzione il cui campo indirizzo contiene l'indirizzo di t più k. Lo stesso funziona, ovviamente, per i riferimenti a $t(v-k)$ invece di $t(v+k)$.

Ad ogni modo, sommando 1 o sottraendo 1 al nostro indice, abbiamo prodotto due nuovi modi di scrivere un ciclo da ognuno di quelli che già avevamo. Ora abbiamo 108 modi per scrivere un ciclo. Naturalmente, in teoria, potremmo porre Q(J+2) uguale a P(J+2) o Q(J−2) uguale a P(J−2), e così via all'infinito. Non c'è nessun limite (eccetto le considerazioni di *overflow* intero) sul numero di nuovi modi di scrivere un ciclo.

Uno di questi, per inciso, ha una certa importanza pratica. Il microcomputer IM6100 (o PDP-8) ha una istruzione di incremento, ma non una di decremento. Inoltre, l'incremento esegue un confronto con 0 (è chiamato ISZ, *increment and skip on* 0). Ciò non corrisponde a niente nella tavola 1; se facciamo il confronto con 0, nella tavola 1, è sempre nel decremento.

Tuttavia, considerate il seguente ciclo:

```
10 J=−N
20 Q(J+N+1) = P(J+N+1)
30 J = J+1
40 IF J< >0 GOTO 20
```

Questa volta non possiamo usare il trucco di cui abbiamo parlato prima a meno che N non sia costante. Tuttavia, se N è una costante, questo ciclo fa la stessa cosa di tutti gli altri cicli di tavola 1; e le sue due ultime istruzioni vengono eseguite, sul IM6100, dalla singola istruzione ISZ J, seguita da un salto all'etichetta 20.

Abbiamo finito con i miglioramenti in velocità nel nostro ciclo? Non del tutto. Considerate il seguente ciclo:

```
10 J=2
20 Q(J−1) = P(J−1)
30 Q(J) = P(J)
40 J = J+2
50 IF J <= N GOTO 20
```

Supponete che N sia 100. Allora, invece di fare questo ciclo cento volte, possiamo farlo solo cinquanta volte. Ogni volta, uguagliamo due elementi del vettore Q ai corrispondenti elementi del vettore P. Il vantaggio è che J=J+2 si fa solo cinquanta volte, J=J+1 si deve fare cento volte nei cicli di tavola 1; e anche l'istruzione IF viene eseguita solo cinquanta volte invece di cento.

Si può accelerare un ciclo facendolo con passo due, ed eseguendo l'assegnazione di due elementi alla volta?

In pratica, il miglioramento qui non è così buono come sembra. L'assegnazione J=J+2 è generalmente più lenta di J=J+1. In effetti, su molti microcomputer, il modo più veloce di fare J=J+2 è di fare J=J+1 due volte. Sul 6502, le due assegnazioni di vettore si possono fare molto elegantemente una dopo l'altra, con

```
LDA P−1,X
STA Q−1,X
LDA P,X
STA Q,X
```

Fino a che il valore di J è nel registro X. Sull'8080, tuttavia, supponendo che l'indirizzo di P(J−1) sia in DE e l'indirizzo di Q(J−1) sia in HL, dovrete fare qualcosa come:

```
LDAX D
MOV M,A
INX D
INX H
```

Per ordinare il volume utilizzare l'apposito tagliando inserito in fondo alla rivista.

Z-80
Pag. 540 **L. 24.000** (Abb. L. 21.600)
Cod. 328D Formato 14,5 x 21

Questi due libri sono stati ideati come testi autonomi e completi per imparare la programmazione in linguaggio Assembler, usando lo Z80 o il 6502 (i microprocessori forse più diffusi).
Scorrevoli da leggere, non richiedono alcuna conoscenza di base, ne di elettronica generale né di programmazione.
Sono stati progettati, infatti, sotto forma di corso che, sistematicamente, passo dopo passo, porta il lettore dai concetti di base fino alle tecniche di programmazione avanzate, al fine di permettergli la realizzazione di programmi sempre più complessi.
L'esposizione progressiva, rigorosamente strutturata, comporta la risoluzione obbligatoria di esercizi attentamente graduati al fine di verificare che si sia veramente capito quanto presentato?
Ben si prestano, perciò, a chi si avvicina per la prima volta ai microprocessori e ne vuole conoscere e capire gli aspetti essenziali di programmazione. Per tutti coloro che già hanno programmato, invece, sarà una vera e propria miniera di informazioni sulle caratteristiche specifiche del microprocessore d'interesse, evidenziandone, nel contempo, vantaggi e svantaggi.

6502
Pag. 384
L. 22.000
(Abb. L. 19.800)
Cod. 503B
Formato 14,5 x 21

La Potenza dei Microprocessori

SOMMARIO
Concetti Fondamentali; Organizzazione Hardware del Microprocessore; Tecniche Fondamentali di Programmazione; Set di Istruzioni; Tecniche di Indirizzamento; Tecniche di Input/Output; Dispositivi di Input/Output; Esempi Applicativi; Strutture dei Dati; Sviluppo del Programma; Conclusioni.

GRUPPO EDITORIALE JACKSON
Divisione Libri

```
LDAX D
MOV M,A
INX D
INX H
```

dove il secondo INX D e INX H predispongono i registri DE e HL per il prossimo ciclo. Quindi l'unico miglioramento qui consiste nell'eseguire meno volte il test alla fine del ciclo.

Ovviamente la stessa idea può essere implementata incrementando la variabile J di 3, di 4, o più ogni volta, sebbene ci sia un corrispondente aumento di memoria occupata dal programma. Un'altra difficoltà con questo schema è che non funziona se J non è multiplo di 2, o dell'incremento. Se questo non è il caso, allora alcuni elementi extra verranno trasferiti da un vettore all'altro, e ciò può causare risultati imprevedibili. La tecnica generale (nota come "srotolare" un ciclo) può tuttavia avere qualche utile applicazione sui grossi computer. Anche su un piccolo sistema è spesso utile, particolarmente quando N è un numero molto piccolo, come 3, scrivere:

```
10 Q(1) = P(1)
20 Q(2) = P(2)
30 Q(3) = P(3)
```

che è migliore di ognuno dei cicli che abbiamo finora discusso.

Un altro miglioramento in velocità dei cicli si ha da un'analisi del caso in cui N=0. Se dobbiamo spostare N quantità, allora, se N è 0, non dobbiamo spostare niente; in altre parole in questo caso non dobbiamo fare proprio niente. Molti dei nostri cicli, tuttavia, o diventano infiniti, o spostano una singola quantità, P(1) o P(N). In particolare questo vale per tutti i cicli della prima o terza riga della tavola 1. La ragione del fatto che ciò da un po' di disturbo è che questi sono gli unici cicli nella tavola 1 che usano tre istruzioni ripetute (le linee 20, 30 e 40). Essi sono quindi i più veloci, poiché tutti gli altri usano quattro istruzioni ripetute (le linee 20, 30, 40 e 50).

In Fortran c'è un errore "istituzionalizzato" nell'esecuzione dei cicli.

Sembrerebbe che avessimo una scelta tra rallentare il ciclo e avere un ciclo che non funziona per N=0. In Fortran, in effetti, la scelta che è stata fatta è quella di non considerare il caso N=0, in favore di un ciclo più veloce. (Si potrebbe chiamare questo un "errore istituzionalizzato" in Fortran.) Dopo tutto, ragionano coloro che lavorano in Fortran, possiamo sempre verificare se N=0 immediatamente prima di partire con il ciclo, se vogliamo evitare questo caso. Ma c'è un modo migliore. Possiamo scrivere:

```
10 J = 1
20 GOTO 50
30 Q(J) = P(J)
40 J = J+1
50 IF J <= N GOTO 30
```

avendo quindi solo tre istruzioni ripetute (le linee 30, 40 e 50) e un ciclo che funziona anche con N=0.

Può sembrare che questo ciclo violi un sacro precetto saltando in mezzo ad un ciclo (alla linea 20). In effetti, tuttavia, il problema connesso con il salto al centro di un ciclo non si applica al caso speciale del salto ad una delle istruzioni di incremento e verifica alla fine del ciclo, a patto di sapere quello che si fa. (Se abbiamo un ciclo FOR in Basic, non possiamo mai saltare dall'esterno del ciclo all'istruzione NEXT alla sua fine; stiamo semplicemente parlando sul modo in cui il ciclo FOR può essere implementato). ■

sinclair

è l'unico sistema completo a 550.000 lire.

ZX81 8 k ROM, 1 k RAM	L. 199.000
ESPANSIONE 16 k RAM	L. 131.000
STAMPANTE ZX	L. 220.000

Prezzi IVA esclusa.

Sinclair, sempre Sinclair: poco più di mezzo milione per un completo sistema di computing.
Guarda, confronta, cerca un'alternativa!
A questo prezzo non trovi neanche un'unità centrale: figuriamoci poi 16 k e la stampante.
Oggi più che mai la chiave che apre le porte dell'informatica per tutti è Sinclair.

REBIT COMPUTER
A DIVISION OF G.B.C.

Per informazioni scrivere a CASELLA POSTALE 10488 - MILANO

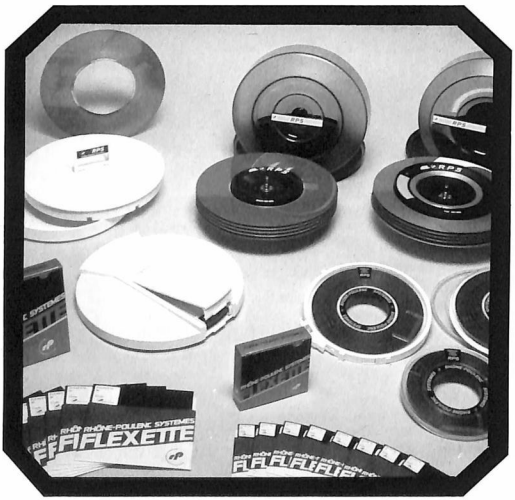
L'ESPERIENZA RHÔNE POULENC AL SERVIZIO DELL'INFORMATICA

RHÔNE-POULENC SYSTEMES

presenta i nuovi prodotti magnetici di fabbricazione

DYPY

filiale comune R.P.S.-DYSAN



I supporti magnetici R.P.S. rispondono alle crescenti esigenze degli utilizzatori
I prodotti R.P.S. sono oggetto di costanti controlli e di tests unitari che li garantiscono REALMENTE "ERROR FREE"

I prodotti di tecnologia DYPY sono utilizzati da tutti i principali fabbricanti di elaboratori, poiché rispettano il loro materiale ed in particolare le testine di lettura, assicurando una manutenzione minima.

Tutti i prodotti informatica fabbricati e commercializzati dalla Soc. RHÔNE - POULENC SYSTEMES sono compatibili con la totalità dei driver esistenti

RHÔNE-POULENC

RHÔNE POULENC ITALIA S.p.A.
Divisione Rhône Poulenc systemes
Via E. Romagnoli, 6 - 20146 MILANO tel. 42461 telex ITARPC 332330

Rhône Poulenc systemes anche per microfilmare, fotocopiare, eliografare.

Martino Giardino

Un piccolo grande problema

Da un po' di tempo circola negli ambienti matematici questo problema, di semplicissima esposizione, ma di soluzione molto difficile.

Prendete un numero (intero, positivo). Se è pari dividetelo per due. Se è dispari, moltiplicatelo per tre, sommategli uno e dividetelo per due. Otterrete un altro numero (intero, positivo).

Riapplicate la stessa procedura, che possiamo riassumere così:

1 Consideriamo un intero positivo x.
2 Se x è pari vai a **3**, altrimenti a **4**.
3 $x \leftarrow x/2$; vai a **2**.
4 $x \leftarrow (3x+1)/2$; vai a **2**.

Prima di porre il problema, applichiamo la procedura un paio di volte. Se partiamo da 12, otteniamo di seguito

$$12 \rightarrow 6 \rightarrow 3 \rightarrow 5 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 1 \rightarrow 2 \ldots$$

cadendo così nel ciclo 1-2 che possiamo indicare così:

$$12 \rightarrow 6 \rightarrow 3 \rightarrow 5 \rightarrow 8 \rightarrow 4 \rightarrow 2 \leftrightarrows 1.$$

Riproviamo partendo da 25: otteniamo

$$25 \rightarrow 38 \rightarrow 19 \rightarrow 29 \rightarrow 44 \rightarrow 22 \rightarrow 11 \rightarrow 17 \rightarrow 26 \rightarrow 13 \rightarrow 20 \rightarrow 10 \rightarrow 5 \rightarrow 8 \rightarrow 4 \rightarrow 2 \leftrightarrows 1.$$

Anche in questo caso siamo caduti nel ciclo 1-2.

Bene, il problema è proprio questo: è vero che partendo da *qualsiasi* numero (intero, positivo) si cade sempre nel ciclo 1-2, oppure partendo da qualche numero particolare non si cade mai in questo ciclo?

Tutti gli esperimenti fatti, anche con computer molto potenti, confermano che si cade nel ciclo 1-2, ma mai nessun matematico è riuscito a dimostrare che ciò è vero per *ogni* numero (intero, positivo). La cosa è tuttavia così verosimile che si ha la convinzione che succeda così. Questa convinzione è detta "congettura di Ulam".

Può comunque darsi che ci sia qualche numero (molto grande) partendo dal quale non si arriva mai al ciclo 1-2: per esempio perché si cresce sempre, o perché si cade in un ciclo diverso.

Con il vostro calcolatore potete fare qualche esperimento.

```
10 REM **********************
20 REM
30 REM CONGETTURA DI ULAM 1
40 REM
50 REM **********************
60 REM
70 INPUT X
80 IF INT(X/2)*2<>X GOTO 110
90 X=X/2
100 GOTO 120
110 X=(3*X+1)/2
120 PRINT X;
130 IF X<>1 GOTO 80
140 PRINT
150 PRINT"CICLO 1-2"
160 GOTO 70
```

Figura 1.

```
10 REM **********************
20 REM
30 REM CONGETTURA DI ULAM 2
40 REM
50 REM **********************
60 REM
70 Y=2
75 Y=Y+1
76 X=Y
80 IF INT(X/2)*2<>X GOTO 110
90 X=X/2
100 GOTO 120
110 X=(3*X+1)/2
120 IF X>=Y GOTO 80
150 PRINT Y "VA IN CICLO 1-2"
160 GOTO 75
```

Figura 3.

Un piccolo grande problema

In figura 1, per iniziare, c'è un programma che semplicemente automatizza la procedura di calcolo della successione. Si dà un numero in input e si può controllare se la successione "cade" nel ciclo 1-2. In figura 2 sono riportati i risultati che si ottengono partendo rispettivamente da 42, 27, 41 e 107.

Il programma implementa esattamente l'algoritmo descritto all'inizio. Nell'istruzione 70 si introduce l'intero di partenza. Nella 80 si controlla se è pari, e in tal caso si va alla 90, oppure dispari, nel qual caso si va alla 110. L'istruzione 120 stampa il nuovo numero e nella 130 si controlla se è 1.

Nella figura 3 è riportato invece un programma di verifica globale. Questo programma si basa sul principio che, se si è verificato che tutti i numeri minori o uguali ad x cadono in un ciclo 1-2, per quanto riguarda $x+1$ basterà ovviamente controllare che nel corso della procedura diventi minore o uguale a x.

Il programma dunque esamina tutti i numeri, uno dopo l'altro, a partire da 3. Nelle istruzioni da 90 a 130 si applica il solito algoritmo. Nell'istruzione 140 si controlla se il numero in esame, $x+1$, è diventato minore o uguale a x.

Il programma non ha termine. In pratica però, a un certo punto x diventerà così grande che la macchina non riuscirà più a rappresentarlo come numero intero. Passerà alla notazione esponenziale, e quindi non ci potrà più essere d'aiuto.

Potete comunque divertirvi a controllare se la congettura di Ulam è verificata per tutti i numeri cui può arrivare il vostro calcolatore. Ma non fatevi troppe illusioni.

```
? 42
21 32 16 8 4 2 1



? 27
41 62 31 47 71 107 161 242 121 182 91 137
206 103 155 233 350 175 263 395 593 890 445
668 334 167 251 377 566 283 425 638 319 479
719 1079 1619 2429 3644 1822 911 1367 2051
3077 4616 2308 1154 577 866 433 650 325 488
244 122 61 92 46 23 35 53 80 40 20 10 5 8 4
2 1
CICLO 1-2



? 41
62 31 47 71 107 161 242 121 182 91 137 206
103 155 233 350 175 263 395 593 890 445 668
334 167 251 377 566 283 425 638 319 479 719
1079 1619 2429 3644 1822 911 1367 2051 3077
4616 2308 1154 577 866 433 650 325 488 244
122 61 92 46 23 35 53 80 40 20 10 5 8 4 2 1
CICLO 1-2



? 107
161 242 121 182 91 137 206 103 155 233 350
263 395 593 890 445 668 334 167 251 377 566
283 425 638 319 479 719 1079 1619 2429 3644
1822 911 1367 2051 3077 4616 2308 1154 577
866 433 650 325 488 244 122 61 92 46 23 35
53 80 40 20 10 5 8 4 2 1
CICLO 1-2
```

Figura 2.

è in edicola!

In questo numero:

- TI 99/4A, Osborne 1, Rainbow 100
- "Hannover Messe '81 Cebit"
- Bitest HP 125
- Il word-processing
- Wordcraft 80, Wordstar, Apple writer
- I package di W.P.
- Sistema di word-processing in Pascal per Apple
- Lo Scripsit della Tandy
- Apple, Atari, Pet, Sinclair e Vic Club
- CPM/Corner

Una pubblicazione del
GRUPPO EDITORIALE JACKSON

Usare il sistema operativo CP/M

IL LIBRO

Il sistema operativo CP/M è stato progettato per rendere semplice l'uso di un microcomputer. Questo libro vi renderà semplice l'uso del CP/M. (Le versioni esaminate del CP/M sono il CP/M 1.4-il CP/M 2.2. e il nuovo sistema operativo multiutente MP/M) La maggior parte di utenti di microcomputer dovrà, infatti, un giorno o l'altro, fare ricorso al CP/M, disponibile su quasi tutti i computer basati sui microprocessori 8080 e Z80, come pure su certi sistemi utilizzanti il 6502. Il libro, senza presupporre alcuna conoscenza di un calcolatore, inizia con la descrizione, passo-passo delle procedure di inizializzazione del sistema: accensione, inserimento dei dischetti, esecuzione delle più comuni operazioni su file, compresa la duplicazione dei dischetti. Prosegue con il PIP (programma di trasferimento dei file), il DDT (programma di messa a punto) e ED (programma editor). Per entrare sempre più, fornendo numerosi consigli pratici, all'interno del CP/M e delle sue operazioni, al fine di comprenderne appieno le risorse ed eventualmente dare gli strumenti per successive modifiche.

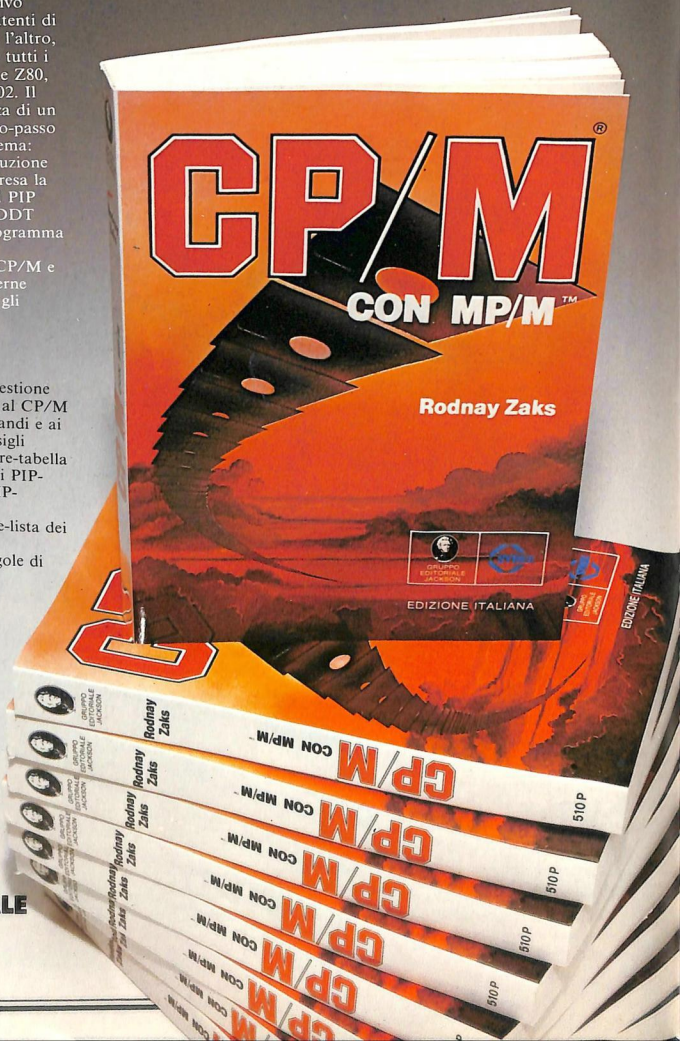
SOMMARIO

Introduzione al CP/M e all'MP/M-Le caratteristiche del CP/M e dell'MP/M-Gestione dei file con PIP-L'uso dell'editor-Dentro al CP/M e all'MP/M-Guida di riferimento ai comandi e ai programmi del CP/M e dell'MP/M-Consigli pratici-Il futuro-messaggi comuni di errore-tabella di controllo di ED-nomi dei dispositivi di PIP-riassunti dei comandi-parole chiave di PIP-parametri di PIP-tasti di controllo per la digitazione dei comandi-tipi di estensione-lista dei materiali-organizzazione della stanza del calcolatore-verifiche in caso di errore-regole di base per la localizzazione dei guasti.

Pagg. 320 Cod. 510P

L.22.000 (Abb. L.19.800)

Per ordinare il volume utilizzare l'apposito tagliando inserito in fondo alla rivista.



GRUPPO EDITORIALE JACKSON
Divisione Libri

Master Mind e intelligenza artificiale

In questo programma il computer usa i principi dell'intelligenza artificiale contro di voi.

di *Duan R. Hope*

Avete mai sentito parlare di *intelligenza artificiale* sfogliando la vostra rivista di computer? Forse vi sarete domandati che cosa fosse e come avreste potuto utilizzarla.

L'intelligenza artificiale (IA) è un sistema realizzato con computer che mostra un comportamento che, se osservato nell'uomo, viene chiamato intelligenza. Le forme più comuni di intelligenza di un sistema IA sono la capacità di risolvere problemi, di apprendere e di riconoscere delle strutture.

Mescolando questi ingredienti, i sistemi IA sono stati progettati per risolvere vari problemi più o meno complessi, come l'analisi degli investimenti, la dimostrazione di teoremi o anche il gioco degli scacchi.

Ho applicato tecniche di IA per la prima volta ad un programma di Master Mind. Avevo già scritto un programma che permetteva ad una persona di giocare a Master Mind: il computer generava un codice e valutava ogni tentativo del giocatore.

Questa versione mi ha presto stufato. Per fare in modo che anche il computer potesse indovinare un codice pensato dal giocatore, ho modificato il programma utilizzando tecniche di IA. Anche se questa versione del Master Mind non è così sofisticata come i sistemi cibernetici di investimento, il computer dimostra con essa un comportamento intelligente. Esso infatti impara a ripetere il processo necessario ad indovinare un codice segreto.

Il programma listato nella prossime pagine è stato realizzato con un TRS-80. È facilmente adattabile ad ogni personal computer perché non usa le caratteristiche grafiche dello schermo.

Il Master Mind

Se non conoscete il Master Mind, si tratta di un gioco di logica deduttiva prodotto dalla Invicta, che si gioca in due persone: il decifratore e il codificatore. I due giocatori si scambiano i ruoli dopo ogni partita.

All'inizio del gioco, il codificatore prepara un codice formato da 4 pioli colorati scelti fra sei gruppi di colore diverso. Il decifratore non può vederlo e deve cercare di indovinarlo facendo dei tentativi che vengono man mano valutati dal codificatore.

Per le sue valutazioni il codificatore userà un piolino nero per ogni piolo del tentativo corretto sia come colore che come posizione, mentre ne userà uno bianco per indicare un piolo corretto come colore ma non come posizione. Il codificatore, tuttavia, non dirà mai a quali pioli del tentativo si riferiscono le sue valutazioni. Vedi esempio 1.

Il codificatore valuta il tentativo dell'esempio 1 con un nero e due bianchi. Se questo fosse il primo tentativo, il decifratore non avrebbe idea di quale fosse il piolo giu-

sto sia come posizione che come colore (quello giallo) e quali due fossero giusti solo come colore (quello rosso e quello blu).

Il codificatore riceve un punto per ogni tentativo del decifratore. Quando tutti e due i giocatori sono stati sia codificatore che decifratore per uno stesso numero prefissato di volte, si confrontano i punteggi: vince ovviamente chi ha più punti.

Analisi mezzi-scopi

La parte più difficile nell'insegnare ad un computer a giocare a Master Mind sta nell'insegnargli ad indovinare un codice segreto. Ma indovinare un codice segreto può essere paragonato a risolvere un problema.

Per fare questo possiamo utilizzare una delle tecniche di soluzione dei problemi dell'IA. Personalmente ho deciso di usare l'analisi *mezzi-scopi* studiata da Newell e Simon.

Nell'analisi mezzi-scopi, gli *scopi* rappresentano la soluzione di un dato problema, e vi ci si riferisce come "stato finale". Lo "stato attuale" dice quanto vicino è il solutore del problema allo stato finale ed è formato da tutte le informazioni accumulate durante la ricerca della soluzione.

Nell'analisi mezzi-scopi è fondamentale il concetto che c'è una differenza misurabile fra lo stato attuale e lo stato finale. La differenza può essere ridotta con una ripetuta applicazione di particolari operazioni ai dati precedentemente acquisiti. Questi sono i *mezzi* nella soluzione del problema.

Nel Master Mind lo stato finale è indovinare il codice. Lo "stato attuale" è dato da tutte le informazioni che il decifratore ha accumulato durante il gioco. Inizialmente lo "stato attuale" è rappresentato solo dalla conoscenza delle regole del Master Mind.

La differenza tra lo stato finale e lo stato attuale è il numero massimo di tentativi che il decifratore potrebbe dover fare per indovinare il codice. Visto che il codice è formato da quattro pioli e ogni piolo potrebbe assumere sei colori diversi, potrebbe essere necessario, per il decifratore, fare 1296 (6×6×6×6) tentativi prima di indovinare il codice. Per ridurre questa differenza, la prima mossa sarà fare un tentativo.

Protocollo verbale e grafi di comportamento al problema

Il *protocollo verbale* è un'altra tecnica di IA che viene utilizzata per scoprire quali ragionamenti vengono fatti dall'uomo per risolvere un certo problema. Per registrare questi ragionamenti, si chiede al solutore del problema di pensare ad alta voce mentre cerca di trovare la soluzione.

I ragionamenti vengono scritti su di un foglio di carta che è chiamato "protocollo verbale". Questo viene analizzato per scoprire le operazioni che il solutore ha fatto per arrivare alla soluzione.

L'analisi viene semplificata usando dei *grafi di comportamento al problema* che sono una rappresentazione del protocollo verbale.

Per spiegare come possono essere utilizzati il protocollo verbale e i grafi vediamo l'esempio di un decifratore che deve fare il suo primo tentativo per indovinare il codice di una partita di Master Mind. Per esemplificare l'esempio, useremo i numeri da 1 a 6, invece dei sei colori, per formare il codice.

Poniamo che il codice sia 4326. Sappiamo già che per prima cosa il decifratore deve fare un tentativo per ridurre la differenza fra stato attuale e stato finale.

Il protocollo verbale del primo tentativo sarà una cosa di questo tipo:

> "Vediamo, devo fare il primo tentativo. Allora, con cosa posso iniziare? Ah sì, potrei iniziare con 1111".
> Il codificatore valuta il tentativo con 0 neri e 0 bianchi.
> Il decifratore continua a pensare: "Nessun bianco e nessun nero, vuol dire che il codice non contiene nessun 1".

Per disegnare il grafo di comportamento al problema, si rappresenta lo stato attuale con un rettangolo. Un'operazione che ci porta ad un nuovo stato sarà rappresentata

	Posizione 1	Posizione 2	Posizione 3	Posizione 4
Codice	blu	rosso	rosso	giallo
Tentativo	rosso	blu	verde	giallo

Esempio 1.

SHARP PC-3201
Il piccolo computer gestionale

INTERNORD

facile come un personal, potente come un mini.

Devi seguire tempestivamente clienti e fornitori, aver sempre aggiornata la situazione crediti e pagamenti, far fronte puntualmente a scadenze e impegni?

Desideri ottenere rapidamente elenchi fatture e ricevute, imputare direttamente dai documenti (senza prima nota) le registrazioni contabili, quadrare giornalmente cassa e fuori cassa, emettere bolle di consegna e fatture, tenere aggiornato il magazzino, fare i bilanci e i conti economici per i controlli budgetari e di fine anno?

Affidati al **PC-3201** della Sharp. Il **PC-3201** è il posto di lavoro EDP che puoi facilmente assegnare alla tua azienda o al tuo ufficio. Perché è dotato di un BASIC interattivo di facile comprensione e di una architettura completa e sofisticata che potrai espandere su misura delle tue esigenze.

Il **PC-3201** possiede infatti una ROM da 32 K e una RAM da 32 K espandibile fino a 112 K, un grande video, memorie a minidischi e una veloce stampante. In questa configurazione il **PC-3201** è alla portata di tutti: il suo prezzo parte infatti da 8.500.000 lire.

Il **PC-3201** è completato dai pacchetti applicativi messi a punto dalla Melchioni Computertime che lo distribuisce in esclusiva per l'Italia e che ne cura l'assistenza anche grazie alla sua rete di Concessionari.

SHARP

MELCHIONI COMPUTERTIME

Concessionari e Rivenditori autorizzati presenti in ogni provincia italiana

Via Fontana, 22 - Milano - Tel. 585.116 - 541.569

SHARP COMPUTERS.

I Nobel dell'informatica.

```
100 REM ************************************************************
110 REM *                                                          *
120 REM *            M A S T E R    M I N D                       *
130 REM *                          DI                              *
140 REM *                  DUANE R. HOPE                           *
150 REM *                                                          *
160 REM ************************************************************
161 REM
162 REM
163 REM ************************************************************
170 REM *                                                          *
180 REM *        PROGRAMMA PRINCIPALE                              *
190 REM *                                                          *
200 REM ************************************************************
210 RANDOM: CLS: DIM GT(20,6)
220 PRINT"   M A S T E R   M I N D":PRINT
230 INPUT"COME TI CHIAMI";NA$
240 PRINT: INPUT"QUANTE PARTITE VUOI FARE";NG
250 IF NG<1 GOTO240
260 PRINT
262 PRINT"ADESSO LANCIO UNA MONETA PER DECIDERE L'ORDINE
    DEL GIOCO"
265 PRINT
270 PRINT"TESTA - SEI IL CODIFICATORE"
280 PRINT"CROCE - SEI IL DECIFRATORE"
290 FOR I=1 TO 1500: NEXT I
300 CT=RND(2)
310 IF CT=1 M1$="TESTA": M2$="CODIFICATORE"
320 IF CT=2 M1$="CROCE": M2$="DECIFRATORE"
330 PRINT M1$;"! TU SEI IL ";M2$: PRINT
350 GC=0: PC=0: PP=0
360 IF CT=1 GOSUB 560 ELSE GOSUB 960
370 IF CT=1 GOSUB 960 ELSE GOSUB 560
380 IF GC=0 GA$="PARTITA" ELSE GA$="PARTITE"
390 PRINT"DOPO";GC+1;GA$;" IL PUNTEGGIO E':"
400 PRINT: PRINT"      "NA$;PP
410 PRINT"      COMPUTER";PC
420 FOR I=1 TO 2500: NEXT I
430 GC=GC+1
440 IF GC<NG GOTO360
450 M1$="PARITA'"
460 PRINT
470 IF PC>PP THEN M1$="HO VINTO IO"
480 IF PC<PP THEN M1$="HAI VINTO TU"
490 PRINT M1$
500 PRINT: INPUT"VUOI FARE UN'ALTRA PARTITA";M1$
510 IF LEFT$(M1$,1)="S" GOTO 240
520 PRINT: PRINT"GRAZIE PER AVER GIOCATO, ";NA$
530 END
540 REM ************************************************************
550 REM *                                                          *
560 REM *        ROUTINE DEL CODIFICATORE                          *
570 REM *                                                          *
580 REM ************************************************************
590 PRINT: PRINT"TOCCA A TE A FARE IL CODICE": PRINT
600 INPUT"BATTI IL CODICE DI 4 CIFRE CHE IO DOVRO' INDOVINARE":
    CO$
605 IF LEN(CO$)<>4 GOTO 600
610 NUM$=CO$: GOSUB 1290
620 IF ER<>0 GOTO 600
630 RESTORE
640 FOR R=1 TO 6
650 FOR C=1 TO 4
660 READ PO(R,C)
670 NEXT C,R
680 C1=1: C2=1: C3=1: C4=0: GU=0
690 GU=GU+1
700 IF GU=1 THEN FOR I=1 TO 4: C(I)=RND(6): NEXT I
710 IF GU=2 THEN C(1)=C1: C(2)=C2: C(3)=C3: C(4)=C4
720 IF GU>2 THEN FOR I=1 TO 4: C(I)=GT(GU-1,I): NEXT I
730 IF GU>1 GOSUB 1690
740 TRY$=""
750 FOR I=1 TO 4: TRY$=TRY$+RIGHT$(STR$(C(I)),1): NEXT I
760 PRINT"PER FAVORE VALUTA IL TENTATIVO NUM.";GU;": ";TRY$
770 INPUT"  QUANTI NERI";BB
780 INPUT"  QUANTI BIANCHI";WW
790 FOR I=1 TO 4: GT(GU,I)=C(I): NEXT I: GT(GU,5)=BB:
    GT(GU,6)=WW
800 NUM$=CO$: GOSUB 1290
810 FOR I=1 TO 4: C(I)=T(I): NEXT I
820 NUM$=TRY$: GOSUB1290
```

Master Mind

da una freccia. Vedi figura 1.

Analizzando il grafo si vede che la prima operazione da fare è il tentativo 1111. Questo ci porta dal primo stato (quando sapevamo solo le regole) al secondo (quando sappiamo che il tentativo è stato valutato con 0 neri e 0 bianchi). Questa operazione ha ridotto la differenza tra lo stato attuale e lo stato finale da 1296 possibili tentativi a 1295. Il grafo (b) della figura 1 mostra la seconda operazione, "Se bianchi + neri = 0 allora i numeri di questo tentativo non possono essere contenuti nel codice", che ci porta ad un successivo stato attuale: il codice non contiene nessun 1. Questa seconda operazione ha ridotto ulteriormente la differenza tra lo stato attuale e lo stato finale a 625 possibilità (5×5×5×5).

Per ridurre ancora la differenza tra stato attuale e stato finale, ho registrato un protocollo verbale mentre cercavo di indovinare svariati codici, ed ho scritto dei grafi di comportamento al problema. Elaborando i grafi, ho trovato che le operazioni potevano essere divise in due gruppi: *riconoscimento di strutture* e *apprendimento*.

Riconoscimento di strutture

Il riconoscimento di strutture è una tecnica di IA usata per trasformare delle informazioni in azioni. Nel riconoscimento di strutture c'è una corrispondenza biunivoca fra le informazioni che si hanno e le azioni che si fanno. Le operazioni di riconoscimento di strutture che ho trovato sono contenute nell'e-

A.E.P. COMPUTERS SYSTEM

via fermi, 40 - 07100 SASSARI - 079/276364

PROPONE

MZ - 80 B

PC - 3201

MZ - 80 A

La famiglia di personal computer più completa attualmente sul mercato per imprenditori, professionisti, tecnici, amministratori.
Vi aspettiamo per fornirvi la più ampia documentazione e le più complete dimostrazioni.

SHARP COMPUTERS

I NOBEL DELL'INFORMATICA

PIEMONTE - GENERAL COMPUTERS - Torino - 011/835156 - COMPDATA - Ivrea - 0125/49069 - OLIVIERI & GOVERNA - Alessandria - 0131/442646 - **LIGURIA** - REM KARD ITALIA - Genova 010/884971 - TECNOSYSTEM - Sanremo - 0184/884794/5 - **LOMBARDIA** SHARCO (di MIGLIORI ROBERTO) - Gavirate - 0332/746526 - ADEL - Brescia - 030/221674 - GAME - Treviglio - 0363/40803 - ENNE COMPUTER - Portichetto di Luisago - 031/920136 - C.E.E. - Vigevano - 0381/81555 - DATA STUDIO (di SERGIO CAVENAGHI) - Burago di Molgora - 039/663736 - LINEA UFFICIO (di ANNUNZIATA ELIO) - Cremona - 0372/24364 - P.G.P. SISTEMA - Milano - 02/2842860 - COMPUTER HOUSE - Monza - 039/362618 **TRE VENEZIE** - SIGMA SYSTEM - Udine - 0432/26992 - INTERSOUND (di COPPETTI FRANCO) - Brunico - 0474/21282 - COMMERCIALE SISTEMI Thiene - 0445/..... - MINI SYSTEM - Bolzano - 0471/32270 - PENTA - Preganziol - 0422/938535 - PINARELLO - Padova - 049/754830 - SYSTEM COPY - 35100 Padova - 049/44982 - PINO ANDREA - Cerea - 0442/82790 - TECNOSYSTEM - Vicenza - 0444/31152 - **EMILIA ROMAGNA - MARCHE - ABRUZZO** - ADRIATICA COMPUTER - Senigallia - 071/62516 - GIMAR SISTEMI - Silvi Marina - 085/932739 - MULTIDATA - Reggio Emilia - 0522/27617 - M.R.P. TECNOSISTEMI - 40069 Zola Predosa - 051/751662 - RODAN & C. - Civitanova Marche - 0733/770386 - ROGANTI F.LLI - Montecassiano - 0733/59231 **TOSCANA** - ELECON - Piombino - 0565/33112 - MNEMO COMPUTERS - Firenze - 055/4378652 - TECNOCOPY - Firenze - 055/352801 - **LAZIO** - EUROCOM - Roma - 06/7574487 - TECNOMEC - Roma - 06/484998 - **CAMPANIA - PUGLIE - CALABRIA** - GENERAL COMPUTERS - Torre Annunziata - 081/8612270 - L. & L. COMPUTERS - Bari - 080/410167 - COMPUTER SUD - Lecce - 0832/42413 - ATLANTIC - Reggio Calabria - 0965/44671 - G.M. MARASCIO COMPUTERLINE - Montauro - 0967/48207 - **SICILIA - SARDEGNA** - SIFI DATA MANAGEMENT - Catania - 095/438178 - A.E.P. COMPUTERS SYSTEM - Sassari - 079/276364 - VIMAR - S. Agata di Militello - 0941/702771.

SHARP

DESIDERO
☐ RICEVERE UNA DOCUMENTAZIONE SULLE SOLUZIONI CON I COMPUTERS SHARP

☐ DISCUTERE IL MIO PROBLEMA SPECIFICO CON UN VOSTRO INCARICATO

NOME _____

SOCIETÀ _____ POSIZIONE _____

INDIRIZZO _____

CITTÀ _____ TEL. _____

sempio 2.

In certi casi, l'uso dei meccanismi di riconoscimento delle strutture porta a ridurre la differenza tra lo stato attuale e lo stato finale. In altri casi, il riconoscimento della struttura deve essere appoggiato dal processo di apprendimento.

Apprendimento

Il processo di apprendimento è una tecnica di IA che modifica gradualmente un sistema di decisione programmato per migliorarne il funzionamento. Le modifiche vengono fatte in base ai risultati ottenuti in precedenza.

Per esempio, ogni tentativo del decifratore, con la valutazione relativa, viene memorizzato. Prima di fare le mosse successive, vengono rivalutati i tentativi precedenti in base al tentativo che sta per essere fatto. Se ogni valutazione data in precedenza corrisponde a quella attuale allora il tentativo potrebbe corrispondere al codice, e verrà giocato. Se invece una qualunque "rivalutazione" è diversa da quella originale, allora il tentativo sarà scartato. Questo è il processo di apprendimento.

Poniamo che il codice sia 4316 e che il primo tentativo sia 1111. Questo sarebbe valutato con 1 nero e 0 bianchi. La differenza fra lo stato attuale e lo stato finale sarebbe ridotta di 1. Tuttavia, usando il processo di apprendimento, possiamo ridurre ulteriormente questa differenza eliminando tutti i tentativi il cui risultato, quando confrontato con 1111 non sia 1 nero e 0 bianchi. Vedi esempio 3.

```
830 GOSUB1480: REM VALUTAZIONE DEL TENTATIVO
840 FOR I=1 TO 4: T(I)=C(I): NEXT I
850 IF BB=B AND WW=W GOTO 880
853 IF B=1 THEN B$="NERO" ELSE B$="NERI"
857 IF W=1 THEN W$="BIANCO" ELSE W$="BIANCHI"
860 PRINT"AVRESTI DOVUTO BATTERE";B;B$;" E";W;W$
870 GT(GU,5)=B: GT(GU,6)=W
880 IF B=4 THEN PP=PP+GU: PRINT"HO INDOVINATO!!":RETURN
890 IF B+W=4 GOSUB 1840
900 IF B+W=0 GOSUB 1990
910 IF B=0 AND W<>0 GOSUB 2110
920 PRINT"STO PENSANDO"
930 GOTO 690
940 DATA 1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5,6,6,6,6
950 RETURN
960 REM ************************************************************
970 REM *                                                          *
980 REM *        ROUTINE DEL DECIFRATORE                           *
990 REM *                                                          *
1000 REM************************************************************
1010 PRINT: PRINT"ADESSO TOCCA A ME A PREPARARE IL CODICE"
1020 FOR I=1 TO 2500: NEXT I
1030 PRINT
1035 PRINT"OA BENE, SONO PRONTO. FAI IL TUO PRIMO TENTATIVO"
1040 PRINT
1050 NO=0
1060 FOR I=1 TO 4: C(I)=RND(6): NEXT I
1070 INPUT"BATTI IL TUO TENTATIVO (4 CIFRE)";TRY$
1075 IF LEN(TRY$)<>4 GOTO 1070
1080 NUM$=TRY$: GOSUB 1290
1090 IF ER<>0 GOTO1070
1100 GOSUB 1480: REM (VALUTAZIONE DEL TENTATIVO)
1110 NO=NO+1
1115 IF B=1 THEN B$="NERO" ELSE B$="NERI"
1117 IF W=1 THEN W$="BIANCO" ELSE W$="BIANCHI"
1120 PRINT"TENTATIVO NUMERO";NO;": ";B;B$;" E";W;W$
1130 IF B<>4 GOTO1070
1140 PRINT"HAI INDOVINATO!!!"
1150 PC=PC+NO
1160 RETURN
1170 REM ************************************************************
1180 REM *                                                          *
1190 REM *    ROUTINE PER TRASFORMARE UNA STRINGA DI QUATTRO        *
1200 REM *    CIFRE NEL VETTORE NUMERICO T(I)                       *
1210 REM *                                                          *
```

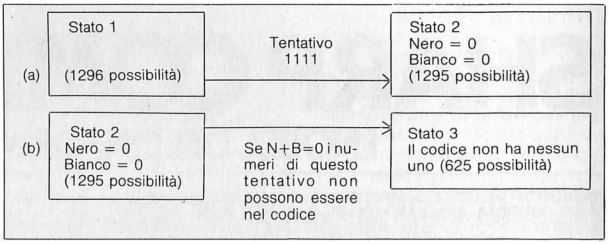


Fig. 1. *Grafo di comportamento al problema.*

Struttura di valutazione	Azione
Neri + bianchi = 0	Non usare questi numeri nei prossimi tentativi.
Neri = 0; bianchi = 0	I numeri di questo tentativo non potranno apparire nella stessa posizione nei prossimi tentativi.
Neri + bianchi = 4	Solo i numeri di questo tentativo sono nel codice, non utilizzarne altri nei prossimi tentativi.
Neri = 4	È stato raggiunto lo stato finale, il codice è stato decifrato.

Esempio 2. Operazioni di riconoscimento strutture.

MNEMO COMPUTERS

via panciatichi, 40/11 - 50127 FIRENZE - 055/4378652

PROPONE

MZ - 80 B

PC - 3201

MZ - 80 A

La famiglia di personal computer più completa attualmente sul mercato per imprenditori, professionisti, tecnici, amministratori.
Vi aspettiamo per fornirvi la più ampia documentazione e le più complete dimostrazioni.

SHARP COMPUTERS

I NOBEL DELL'INFORMATICA

PIEMONTE - GENERAL COMPUTERS - Torino - 011/835156 - COMPDATA - Ivrea - 0125/49069 - OLIVIERI & GOVERNA - Alessandria - 0131/442646 - **LIGURIA** - REM KARD ITALIA - Genova 010/884971 - TECNOSYSTEM - Sanremo - 0184/884794/5 - **LOMBARDIA** SHARCO (di MIGLIORI ROBERTO) - Gavirate - 0332/745526 - ADEL - Brescia - 030/221674 - GAME - Treviglio - 0363/40803 - ENNE COMPUTER - Portichetto di Luisago - 031/920136 - C.E.E. - Vigevano - 0381/81555 - DATA STUDIO (di SERGIO CAVENAGHI) - Burago di Molgora - 039/663736 - LINEA UFFICIO (di ANNUNZIATA ELIO) - Cremona - 0372/24364 - P.G.P. SISTEMA - Milano - 02/2842860 - COMPUTER HOUSE - Monza - 039/362618 **TRE VENEZIE** - SIGMA SYSTEM - Udine - 0432/26992 - INTERSOUND (di COPPETTI FRANCO) - Brunico - 0474/21282 - COMMERCIALE SISTEMI Thiene - 0445/..... - MINI SYSTEM - Bolzano - 0471/32270 - PENTA - Preganziol - 0422/938535 - PINARELLO - Padova - 049/754830 - SYSTEM COPY - 35100 Padova - 049/44982 - PINO ANDREA - Cerea - 0442/82790 - TECNOSYSTEM - Vicenza - 0444/31152 - **EMILIA ROMAGNA - MARCHE - ABRUZZO** - ADRIATICA COMPUTER - Senigallia - 071/62516 - GIMAR SISTEMI - Silvi Marina - 085/932739 - MULTIDATA - Reggio Emilia - 0522/27617 - M.R.P. TECNOSISTEMI - 40069 Zola Predosa - 051/751662 - RODAN & C. - Civitanova Marche - 0733/770386 - ROGANTI F.LLI - Montecassiano - 0733/59231 **TOSCANA** - ELECON - Piombino - 0565/33112 - MNEMO COMPUTERS - Firenze - 055/4378652 - TECNOCOPY - Firenze - 055/352801 - **LAZIO** - EUROCOM - Roma - 06/7574487 - TECNOMEC - Roma - 06/484998 - **CAMPANIA - PUGLIE - CALABRIA** - GENERAL COMPUTERS - Torre Annunziata - 081/8612270 - L. & L. COMPUTERS - Bari - 080/410167 - COMPUTER SUD - Lecce - 0832/42413 - ATLANTIC - Reggio Calabria - 0965/44671 - G.M. MARASCIO COMPUTERLINE - Montauro - 0967/48207 - **SICILIA - SARDEGNA** - SIFI DATA MANAGEMENT - Catania - 095/438178 - A.E.P. COMPUTERS SYSTEM - Sassari - 079/276364 - VIMAR - S. Agata di Militello - 0941/702771.

SHARP

DESIDERO
□ RICEVERE UNA DOCUMENTAZIONE SULLE SOLUZIONI CON I COMPUTERS SHARP

□ DISCUTERE IL MIO PROBLEMA SPECIFICO CON UN VOSTRO INCARICATO

NOME .....

SOCIETÀ ..... POSIZIONE .....

INDIRIZZO .....

CITTÀ ..... TEL. .....

```
1220 REM *       INPUT:  NUM$ STRINGA DI 4 CIFRE                *
1230 REM *       OUTPUT: T(I) VETTORE NUMERICO CONTENENTE       *
1240 REM *                    L'EQUIVALENTE DI NUM$             *
1250 REM *               ER   CODICE D'ERRORE                   *
1260 REM *                       0=NESSUN ERRORE                *
1270 REM *                       1=CIFRA <1 O >6 IN NUM$        *
1280 REM *                                                      *
1290 REM ********************************************************
1300 ER=0
1310 FOR I=1 TO 4
1320 T(I)=VAL(MID$(NUM$,I,1))
1330 IF T(I)<1 OR T(I)>6 THEN ER=1
1340 NEXT I
1350 RETURN
1360 REM ********************************************************
1370 REM *                                                      *
1380 REM *    ROUTINE PER VALUTARE UN TENTATIVO                 *
1390 REM *                                                      *
1400 REM *       INPUT:  T(I) VETTORE NUM. COL TENTATIVO        *
1410 REM *               C(I) VETTORE NUM. COL CODICE           *
1420 REM *                                                      *
1430 REM *       OUTPUT: B    NUMERO DI NERI                    *
1440 REM *               W    NUMERO DI BIANCHI                 *
1450 REM *                                                      *
1460 REM *       NB.  IL CONTENUTO DI T(I) VIENE PERSO          *
1470 REM *                                                      *
1480 REM ********************************************************
1500 B=0: W=0
1510 FOR I=1 TO 4: W(I)=C(I): NEXT I
1520 FOR I=1 TO 4
1530 IF W(I)<>T(I) GOTO 1550
1540 W(I)=-1: T(I)=-2: B=B+1
1550 NEXT I
1560 FOR I=1 TO 4
1570 FOR J=1 TO 4
1580 IF T(I)<>W(J) GOTO 1600
1590 W(J)=-1: W=W+1: GOTO 1610
1600 NEXTJ
1610 NEXTI
1620 RETURN
1630 REM ********************************************************
1640 REM *                                                      *
1650 REM *    ROUTINE PER FARE UN NUOVO TENTATIVO               *
1660 REM *                                                      *
1670 REM *       OUTPUT: C(I) NUOVO TENTATIVO                   *
1680 REM *                                                      *
1690 REM ********************************************************
1700 GOSUB 2220
1710 FOR L=1 TO GU-1
1720 FOR J=1 TO 4
1730 T(J)=GT(L,J)
1740 NEXT J
1750 GOSUB 1480
1760 IF GT(L,5)=B AND GT(L,6)=W GOTO 1780
1770 GOTO 1700
1780 NEXT L
1790 RETURN
1800 REM ********************************************************
1810 REM *                                                      *
1820 REM *    ROUTINE PER ELIMINARE DALLA TABELLA DELLE POS-    *
1830 REM *    SIBILITA' TUTTI I NUMERI TRANNE QUELLI CONTENU-   *
1840 REM *    TI IN QUESTO TENTATIVO.                           *
1845 REM *    DA USARE QUANDO N+B=4                             *
1850 REM *                                                      *
1860 REM ********************************************************
1870 FOR I=1 TO 6: EL(I)=I: NEXT I
1880 FOR I=1 TO 4
```

Se questo tentativo fosse il codice	Allora la valutazione di 1111 sarebbe
1112	3 neri e 0 bianchi
1113	3 neri e 0 bianchi
1114	3 neri e 0 bianchi
1115	3 neri e 0 bianchi
1116	3 neri e 0 bianchi

Esempio 3.

Master Mind

Visto che il codice ha causato 1 nero e 0 bianchi come valutazione al tentativo 1111, nessuno dei tentativi dell'esempio 3 potrebbe essere il codice perché se sostituiti al codice non danno come valutazione 1 nero e 0 bianchi.

In questo caso, le possibilità verrebbero ridotte a 500.

Completamento dell'analisi

Completando la mia analisi del grafo di comportamento al problema, ho capito che sarebbe stato necessario sviluppare un sistema per generare tutti i possibili 1295 codici, altrimenti il computer non sarebbe mai stato in grado di fare dei tentativi. Inoltre questo generatore avrebbe dovuto produrre ogni codice solo una volta.

Il generatore di tentativi avrebbe anche dovuto essere predisposto a funzionare in base ai risultati del riconoscimento delle strutture. Per fare questo ho usato la *tabella delle possibilità* della figura 2.

Tutti i 1296 possibili tentativi possono essere generati da questa tabella. All'inizio i numeri contenuti nelle locazioni W, X, Y e Z sono posti uguali a 1. Poi, per ge-

Numero da usare	Posizione nel tentativo			
	W	X	Y	Z
1	1	1	1	1
2	2	2	2	2
3	3	3	3	3
4	4	4	4	4
5	5	5	5	5
6	6	6	6	6

Figura 2. *Tavola delle possibilità.*

nerare i tentativi successivi, la locazione Z varia da 1 a 6.

Dopo le prime sei generazioni (1111, 1112, 1113, 1114, 1115, 1116) la locazione Z è posta uguale ad 1 e quella alla sua sinistra (la Y) è posta uguale a 2. Quando una valutazione dà 0 neri e 0 bianchi, ogni numero del tentativo può essere cancellato da ogni posizione della tabella delle possibilità ponendo la relativa locazione uguale a 0. Ogni successivo tentativo che contiene almeno uno 0 può essere scartato.

Il modello per risolvere il problema

Le operazioni trovate analizzando il grafo di comportamento possono essere riunite per formare un modello di soluzione del problema. Vedi figura 3. Il modello inizia generando un tentativo (primo passo) che non contraddica le informazioni ricevúte finora (secondo passo). Se un tentativo le contraddicesse, verrebbe eliminato e ne sarebbe generato un altro, e così via, fino a quando ne viene generato uno possibile che verrà considerato come tentativo da giocare (terzo passo).

Dopo aver ricevuto la valutazione (quarto passo) questa viene usata nel meccanismo di riconoscimento delle strutture (quinto passo) per vedere se qualche numero debba essere eliminato dalla tabella delle possibilità. Se la valutazione era 4 neri (sesto passo), il codice è stato indovinato e il problema risolto; altrimenti il modello ritorna al primo passo e viene ripetuto il tutto.

Il listato potrà aiutare a vedere come funziona effettivamente il programma. A questo punto dovreste essere in grado di applicare alcune di queste tecniche di IA ad altre situazioni nelle quali il computer debba risolvere problemi o prendere decisioni. ■

```
1890 EL(GT(GU,I))=0
1900 NEXT I
1910 FOR I=1 TO 6
1920 IF EL(I)<>0 THEN PO(I,1)=0: PO(I,2)=0: PO(I,3)=0:
     PO(I,4)=0
1930 NEXT I
1940 RETURN
1950 REM ************************************************************
1960 REM *                                                          *
1970 REM *     ROUTINE PER ELIMINARE DALLA TABELLA DELLE POS-       *
1980 REM *     SIBILITA' TUTTI I NUMERI CONTENUTI IN QUESTO         *
1990 REM *     TENTATIVO.                                           *
1995 REM *     DA USARE QUANDO N+B=0                                *
2000 REM *                                                          *
2010 REM ************************************************************
2020 FOR I=1 TO 4
2030 FOR J=1 TO 4
2040 PO(GT(GU,I),J)=0
2050 NEXT J,I
2060 RETURN
2070 REM ************************************************************
2080 REM *                                                          *
2090 REM *     ROUTINE PER ELIMINARE DALLA TABELLA DELLE POS-       *
2100 REM *     SIBILITA' QUESTO TENTATIVO.                          *
2110 REM *     DA USARE QUANDO N=0 E B<>0                           *
2120 REM *                                                          *
2130 REM ************************************************************
2140 FOR I=1 TO 4
2150 PO(GT(GU,I),I)=0
2160 NEXT I
2170 RETURN
2180 REM ************************************************************
2190 REM *                                                          *
2200 REM *     ROUTINE PER ESTRARRE UN NUMERO DALLA TABELLA         *
2205 REM *     DELLE POSSIBILITA'.                                  *
2210 REM *                                                          *
2220 REM ************************************************************
2230 GOTO 2360
2240 IF PO(C1,1)<>0 THEN C(1)=PO(C1,1): GOTO 2280
2250 C1=C1+1
2260 IF C1<7 GOTO 2240
2270 PRINT"ERRORE NELLA TABELLA DELLE POSSIBILITA'":END
2280 IF PO(C2,2)<>0 THEN C(2)=PO(C2,2): GOTO 2320
2290 C2=C2+1
2300 IF C2<7 GOTO 2280
2310 C2=1: GOTO 2250
2320 IF PO(C3,3)<>0 THEN C(3)=PO(C3,3): GOTO 2360
2330 C3=C3+1
2340 IF C3<7 GOTO 2320
2350 C3=1: GOTO 2290
2360 C4=C4+1
2370 IF C4<7 GOTO 2390
2380 C4=0: GOTO 2330
2390 IF PO(C4,4)=0 GOTO 2360
2400 C(4)=PO(C4,4)
2410 RETURN
```

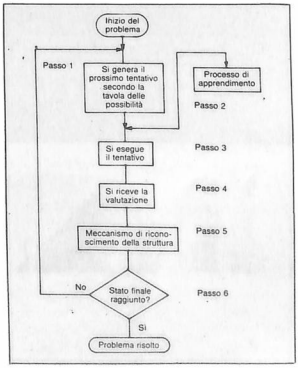


Figura 3. *Diagramma di flusso del modello di risoluzione del problema.*

Eco1:
un collaboratore

vds

VIDEO DISPLAY SYSTEMS

sistemi gestionali dell'ultima generazione.

PRODOTTI E ASSISTITI IN ITALIA

ben preparato

RESPONSABILE RETE DISTRIBUZIONE

DEDO SISTEMI srl

50129 Firenze
Piazza Indipendenza, 13
Tel. 055/474467-486265
Telex 600282 DEDO TL

L'efficienza di un computer dipende dalle periferiche. L'efficienza delle periferiche dipende da SEGI.

Oggi l'informatica è organizzazione efficiente di centro e periferia: computer sì, ma anche periferiche adeguate, funzionali, rispondenti alle esigenze sempre più diversificate. Esigenze di scelte e disponibilità che Segi risolve meglio di ogni altro proprio perché ha un'esperienza più qualificata nel proporre a tutti gli utenti la gamma più completa di periferiche: quanto di meglio offre il mercato internazionale.
Qualità e gamma di prodotti sì,... e soprattutto servizi che Segi, leader nel settore, assicura: affidabilità e assistenza tecnica capillare, effettuata da personale qualificato su tutto il territorio nazionale. Fidati di Segi. È un nome che conta nell'informatica.

LA NUOVA FAMIGLIA DI STAMPANTI EPSON MX - SERIE III

Alle già note e tradizionali prestazioni tecniche della precedente serie (matrice di aghi 9x9, possibilità di trascinamento in modulo continuo o foglio singolo, universalità di interfacciamento, stampa in espanso, compresso e allargato) le nuove stampanti EPSON serie III offrono: grafica implementata, indici e apici, sottolineatura e sul mod. MX 100 velocità di stampa di 100 cps bidirezionali ottimizzati.

AMITALIA, SAICO, SEGI: tre leader.
un gruppo, AMMI.

segi **SERVIZI GENERALI PER L'INFORMATICA** S.P.A.

SEGI - Via Timavo, 12 - 20124 Milano
tel. (02) 6709136 (5 linee ricerca automatica) - Telex 315132 I
SEGI - Via Asmara, 58 - 00199 Roma
tel. (06) 8395766 - Telex 616130 I

Programmazione ricorsiva in Basic

Le funzioni ricorsive si possono programmare anche in Basic: basta creare l'ambiente adatto.

di *Antonio Leal*

A molti utenti di personal computer la nozione di programma "ricorsivo" appare come un argomento elusivo, appartenente alla scienza dei calcolatori più avanzata ed ai linguaggi di programmazione evoluti. Sebbene il Basic non sia stato progettato per supportare subroutine ricorsive, esse possono essere implementate con minimo sforzo e con buoni risultati.

Nella codifica di programmi complessi, può essere risparmiata una grande quantità di tempo. Tuttavia, è necessaria una qualche spiegazione preliminare del concetto di ricorsività.

Il presente articolo descrive alcuni dei concetti più importanti della programmazione ricorsiva e mostra come si possono usare. Gli esempi mostrano come programmare in Basic. Infine, la programmazione ricorsiva è applicata alla ricerca in un albero di gioco.

Funzione

Il concetto di *funzione* è fondamentale in matematica e informatica. In termini di programmazione, una funzione è una subroutine. Tuttavia, essa ha una struttura più ricca rispetto ad una subroutine Basic. Essa consiste in un *nome*, un certo numero di variabili in input dette *argomenti* o *parametri*, un corpo di istruzioni di programma, ed una quantità di output detta *valore* della funzione. Ecco un esempio di funzione scritta in forma matematica

$$f(x,y) = x^2+y^2.$$

La lettera f è il nome della funzione. Le variabili x e y sono gli argomenti. Per poter usare la funzione, bisogna sostituire gli elementi x e y con numeri e dopo il calcolo il risultato (valore) viene ritornato tramite il nome. Per esempio se poniamo $x=3$ e $y=4$ il risultato è 25.

In altre parole f (3,4) ritorna 25 come valore di f.

Una simpatica proprietà dei nomi assegnati alle funzioni è che possono essere richiamati direttamente in altre espressioni. Per esempio, supponiamo che la precedente funzione facesse parte di un programma. Quindi, se scrivessimo

$$b = 2+f(3,4)$$

il valore della funzione (25) sarebbe aggiunto a 2 e a b verrebbe assegnato il valore 27. Quindi, quando la funzione ha un nome, essa può essere attivata semplicemente usando il nome, senza usare GOSUB.

Una funzione può essere scritta anche in forma di subroutine. Il seguente esempio non è possibile in tutti i linguaggi di programmazione, ma è accettato in molti di essi,

Programmazione ricorsiva in Basic

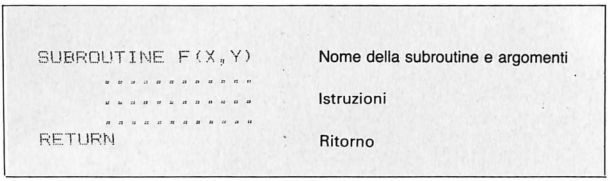


Figura 1.

tra cui il Pascal. Vedi figura 1.

La subroutine ovviamente può essere scritta ovunque all'interno del programma principale ed è attivata chiamandola per nome. Un concetto molto importante relativo alle funzioni è lo "scopo" delle variabili. Le variabili specificate come argomento (X e Y nell'esempio precedente) sono sempre *locali* alla subroutine e perdono il loro significato al di fuori di essa. Per esempio, considerate lo scheletro di programma di figura 2.

In figura 2, la variabile X usata nel programma principale, non è la stessa X usata nella subroutine. Quando la subroutine è attivata, la X del programma principale (globale) è salvata e viene creata una nuova X (locale) per l'uso all'interno della subroutine. Quando viene restituito il controllo al programma principale, la vecchia X globale ridiventa attiva, conservando il suo valore originale. Quindi, nell'esempio indicato, X avrebbe un valore 3 dopo l'uscita della subroutine anche se la X locale era posta a 4 dentro la subroutine.

Lo scopo delle variabili suggerisce la nozione di "ambiente" di un programma, che è semplicemente una lista di tutte le variabili che sono attualmente visibili e possono essere usate o cambiate. Quando l'esecuzione salta dentro e fuori dalle subroutine, l'ambiente cambia corrispondentemente, e tutto è regolato dal sistema.

Funzioni ricorsive

Una funzione (subroutine) ricorsiva è una funzione che usa se stessa nella propria definizione. Le definizioni ricorsive sono molto utili per descrivere certi tipi di formule matematiche e, come vedremo, per programmare problemi complessi.

Per esempio, il fattoriale può essere descritto sia in forma ricorsiva che in forma non ricorsiva. La più semplice descrizione del fattoriale è la seguente

$$x! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot x.$$

In altri termini, $x!$ è il prodotto di tutti gli interi da 1 a x, supponendo che x sia un intero positivo.

La definizione ricorsiva di $x!$ è questa

$$x! = x\,(x-1)!.$$

Questa definizione usa il fattoriale per definirlo. Essa dice che se si vuole sapere quanto vale $x!$, basta moltiplicare x per $(x-1)!$ e dunque il calcolo deve aspettare. Infatti per calcolare $(x-1)!$ dobbiamo riapplicare la definizione: $(x-1)!$ è semplicemente $x-1$ moltiplicato per $(x-2)!$, e così via.

È naturale che questo tipo di definizione non è utile se non c'è un modo di bloccarla. Ogni funzione ricorsiva deve avere una "regola di stop". La regola è semplicemente il valore iniziale per $x = 1$. La definizione ricorsiva completa deve comprendere il valore iniziale

$$1! = 1$$

$$x! = x\,(x-1)!.$$

Ora sappiamo che quando l'iterazione (ricorsiva) raggiunge 1, il risultato è 1 e la ricorsione si ferma. Ciò permette di "tornare indietro" e si possono fare tutti i calcoli lasciati in sospeso. La figura 3 mostra la procedura per 5!.

Un programma o subroutine per il fattoriale dovrebbe chiamare se stesso dal proprio interno. In altri termini, all'interno della subroutine deve essere usato il suo stesso nome. Tuttavia, prima di poter fare questas chiamata, è necessario un test per vedere se la condizione di stop è stata raggiunta. Vedi figura 4.

Ancora, la forma sopradescritta è una descrizione sommaria di come la subroutine debba essere ristrutturata, e non fa parte di nessun particolare linguaggio. Si noti che F stesso è stato usato nella linea 2. Per una chiamata iniziale di F(5) cioè 5! la subroutine chiamerebbe se stessa 4 volte. Si dice che la ricorsione è a 4 livelli di profondità. Ogni chiamata realizza automaticamente un nuovo ambiente in cui la X locale è differente dalle precedenti chiamate. Quindi, non vanno confuse. Alla fine, all'ultima chiamata, quando X = 1, la condizione di stop è verificata nella linea 1 e viene ritornato un 1 come risul-

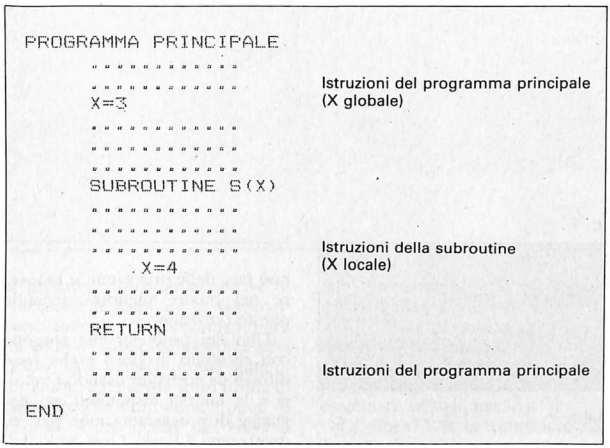


Figura 2.

tato. A questo punto, avvengono tutti i ritorni di subroutine, e i calcoli sospesi vengono completati fino a che la subroutine esce per l'ultima volta con la risposta 5!.

Alcune funzioni ricorsive dipendono non solo dal precedente valore. Per esempio, la successione di Fibonacci dipende dai precedenti due valori. Quindi, devono essere dati due valori iniziali quando si definisce la ricorsione

$$f(1) = 1$$

$$f(2) = 1$$

$$f(x) = f(x-1)+f(x-2)$$

Questa definizione dice che il valore della funzione è la somma dei precedenti due valori e fornisce la seguente successione:

x:	1	2	3	4	5	6	7	8 ...
$f(x)$:	1	1	2	3	5	8	13	21 ...

Ricorsività in Basic

Non vi sono problemi nell'avere una subroutine che chiama se stessa in Basic. Il computer tiene traccia di tutti i punti di ritorno in ordine, ma questo è tutto quello che può fare. Poiché gli argomenti non sono permessi, l'ambiente deve essere esplicitamente creato e salvato dal programmatore. Si può fare ciò dimensionando ogni singola variabile. (Per le variabili già dimensionate occorrerà incrementare la loro dimensioni di 1.) Questa dimensione extra serve per contenere tutti i risultati intermedi della ricorsione e crea spazio per le variabili "locali". Gli elementi di ogni variabile dimensionata corrispondono alla "profondità" della ricorsione e un indice tiene traccia di questa profondità. Ciò significa che ogni volta che viene fatta una chiamata a una subroutine, l'indice deve essere decrementato e, inversamente, ogni volta che si ha un ritorno, deve essere incrementato. Quando l'indice raggiunge 1, sappiamo che deve essere usata la regola di stop. La figura 5 mostra come programmare il fattoriale in Basic usando la ricorsione.

Si supponga che nella linea 30 si dia a X il valore 5. La prima chiamata alla subroutine si ha nella linea 40. Alla linea 100 viene trovata non verificata la condizione di stop, e dunque X viene decrementata (a 4) e la subroutine viene ancora chiamata. Questa volta, tuttavia, la linea di ritorno è la 130 e non la 50 come per la prima chiamata. Il "ciclo" tra le linee 100 e 120 continua fino a che X = 1, e la linea 100 causa un ritorno.

Il ritorno è alla linea 130 e la subroutine comincia a "riavvolgersi". I ritorni successivi "ciclano" tra le linee 130 e 150 ponendo i risultati intermedi del calcolo in successive locazioni di F. Quando la ricorsione è terminata (il che viene trattato automaticamente dal sistema), la subroutine ritorna per l'ultima volta alla linea 50 e viene stampato il risultato finale F(5). (Per osservare il funzionamento della routine, inserite 145 PRINT F(X)).

La ragione per cui X deve essere incrementata nella linea 130 *prima* della definizione ricorsiva della linea 140 è che il risultato del calcolo deve essere messo nella prossima locazione di F. Se le linee fossero invertite, quando viene raggiunto il più basso livello di ricorsione e X = 1, F(X−1) farebbe riferimento a F(0) che non è definito. Le linee 130 e 140 potrebbero essere invertite se la definizione ricorsiva venisse cambiata come segue:

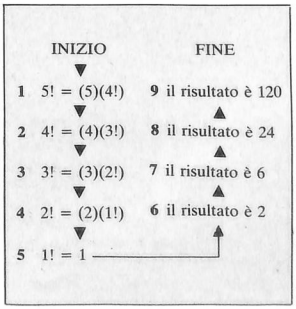


Figura 3.

```
SUBROUTINE F(X)
    1. IF X=1 THEN RETURN 1
    2. ELSE RETURN X*F(X-1)
END
```

Figura 4.

130 F(X+1) = (X+1)*F(X)

140 X = X+1.

Tuttavia alcune versioni di Basic non permettono calcoli dentro le parentesi se sono a sinistra di un segno di uguale. Quindi il primo metodo è migliore.

Scrivere un programma per la successione di Fibonacci non è molto più complicato. La struttura del programma è identica. Tuttavia la regola di stop nella linea 100 deve verificare che X sia minore o uguale a 2 piuttosto che X sia uguale a 1 perché in questo caso ci sono due valori iniziali. Vedi la figura 6.

```
PROGRAMMA PRINCIPALE

10 DIM F(100)
20 F(1)=1
30 INPUT X
40 GOSUB 100
50 PRINT F(X)
60 STOP


SUBROUTINE RICORSIVA

100 IF X=1 THEN RETURN
110 X=X-1
120 GOSUB 100
130 X=X+1
140 F(X)=X*F(X-1)
150 RETURN
```

Figura 5.

Ricerca negli alberi di gioco

Questo paragrafo tratta temi più avanzati e presuppone alcune conoscenze sulla programmazione dei giochi di scacchiera mediante "albero di gioco".

L'uso dei programmi ricorsivi per calcolare semplici funzioni come quelle degli esempi visti non è molto efficiente. Si possono realizzare con cicli più corti, più semplici e più veloci, che occupano meno spazio e sono meno complicati. Tuttavia, quando il problema è più complesso, i programmi ricorsivi possono far risparmiare molto tempo di programmazione e rendere il programma più facile da correggere. Uno degli argomenti in cui le routine ricorsive possono aiutare è la ricerca negli alberi di gioco. Già sapete che i computer possono essere programmati per giocare giochi di scacchiera, costruendo un "albero" delle posizioni sulla scacchiera usando le mosse permesse.

Quindi, ricercando nell'albero le posizioni migliori, il programma può fare delle previsioni, e muovere nel modo migliore, secondo queste previsioni.

Uno dei problemi che sorgono con gli alberi di gioco è che sono difficili da costruire usando i vettori e le matrici disponibili nei linguaggi di programmazione più comuni come il Basic. Ogni nodo dell'albero deve contenere informazioni sulla sua profondità dalla radice, se è un nodo di *max* o di *min*, tutte le informazioni che descrivono la posizione attuale di tastiera, e il suo valore calcolato combinando i valori di tutti i suoi successori.

Inoltre, occorre stabilire un collegamento tra ogni nodo e il suo predecessore, e fra ogni nodo e i suoi successori. Ciò è particolarmente complicato perché non è sempre possibile sapere quanti successori ha un nodo. Occorre scrivere programmi che accedano e modifichino questa struttura di dati ad albero mantenendo tutte le connessioni e separando i dati di un nodo da quelli di un altro.

Le subroutine ricorsive possono eliminare la necessità di considerare alcuni di questi problemi. Usando una subroutine ricorsiva, *non è necessario costruire e mantenere esplicitamente una struttura di dati ad albero*. Può essere difficile crederlo, ma è possibile trarre profitto dal modo in cui le subroutine ricorsive si comportano, per costruire un albero ed eseguire una ricerca nello stesso tempo. Si può fare come segue. Supponiamo che la subroutine stessa sia un nodo nell'albero di gioco. In questo caso, un nodo dell'albero, sarà più che una semplice collezione di dati. Sarà una *procedura*. Il nodo potrà quin-

di generare posizioni successive di scacchiera e chiamare se stesso ricorsivamente. Il collegamento tra un nodo e i suoi predecessori e successori sarà mantenuto dal computer automaticamente sotto forma di chiamate alla subroutine e punti di ritorno. Tutto quello che serve è una attenta costruzione della subroutine in modo che possa all'occorrenza trattare situazioni differenti. Il seguente scheletro di programma è una descrizione di come dovrebbe essere fatta una tale subroutine. Non si tratta di un programma completo, perché viene usato il linguaggio comune per spiegare cosa succede nei vari punti.

I dettagli dipenderanno dal tipo di gioco che si intende programmare. Tuttavia la struttura generale sarà la stessa per tutti i giochi che usano la ricerca ad albero. Vedi figura 7.

Tutti i giochi che richiedono la ricerca ad albero possono essere programmati in modo ricorsivo.

La variabile D è l'indice di ricorsione e rappresenta il livello attuale dell'albero. B(D) è la posizione di scacchiera da analizzare al livello D. V(D) contiene i valori dei nodi, *max* o *min* secondo il tipo di nodo. Il tipo di nodo è contenuto nella variabile M che è posta a 1 per un nodo *max* e a 0 per un nodo *min*. Scrivendo M = 1−M, vengono usati alternativamente i due valori. I(D) viene usato per contare il numero di successori di ogni nodo. Deve essere dimensionato da D per registrare quale successore è da esaminare ad ogni livello.

La linea 1 è la condizione di stop. Se D = 1 è stato raggiunto il fondo dell'albero ed è il momento di chiamare la funzione euristica sulla posizione di scacchiera B(1).

Questa funzione analizza la posizione di scacchiera e ritorna un valore in V(1) che è una misura della sua bontà. Quanto più alto è il numero, migliore è la posizione. La linea 2 pone V(D) a un numero molto alto o molto piccolo, secondo che il nodo sia *min* o *max*. La ragione di ciò sta nel fatto che più avanti si farà un confronto per avere il più grande (o il più piccolo) valore di tutti i successori di un nodo. Questa linea inizializza il valore.

La linea 3 comincia il ciclo che esaminerà tutti i successori del nodo, uno alla volta, e chiamerà la subroutine ricorsiva per ognuno. Si noti che è dimensionato da D così che i successori di nodi e a livelli differenti nell'albero non vengono mescolati. La linea 4 è dove viene effettivamente generato il prossimo successore. Poiché l'indice di ricorsione va verso l'1, il prossimo successore è in D−1. Se il gioco è complesso, ci possono essere molte istruzioni in questo punto e, per contenere tutte le informazioni, può essere necessaria più di una variabile.

Le prossime tre linee predispongono l'ambiente, chiamano la subroutine ricorsivamente e ripristinano l'ambiente nella sua forma originale. Queste linee sono simili a quelle viste negli esami precedenti. D è il livello corrente e M commuta da *max* a *min* e viceversa.

```
10 DIM F(100)
20 F(1)=1
30 F(2)=1
40 INPUT X
50 GOSUB 100
60 PRINT F(X)
70 STOP


100 IF X<=2 THEN RETURN
110 X=X-1
120 GOSUB 100
130 X=X+1
140 F(X)=F(X-1)+F(X-2)
150 RETURN
```

Figura 6.

Nella linea 8 viene aggiornato il valore del nodo corrente. È posto al *max* (o *min*) del suo presente valore e quello del successore che è stato appena analizzato. Quando il ciclo è finito, V(D) contiene il *max* (o *min*) di tutti i successori del nodo corrente e sarà pronto per essere usato dai suoi predecessori. Le linee 9 e 10 completano il ciclo e risalgono di un livello nell'albero.

La caratteristica interessante di questo programma è che, ad ogni momento dell'esecuzione, esiste solo un "cammino" dalla radice. Il programma crea l'albero mentre esegue la ricerca, percorrendo i suoi rami in su e giù, con chiamate e ritorni alle subroutine. Non vi è una esplicita struttura di dati ad albero che deve essere mantenuta dal programmatore. Tutte le infor-

```
1.  IF D=1 THEN poni V(1) al valore della funzione
       euristica su B(1) e RETURN
2.  Poni V(D) ad un numero molto grande se M=0 (MIN)
       o ad un numero molto piccolo se M=1 (MAX)
3.  FOR I(D)=1 TO il numero dei successori di B(D)
    4. Sostituisci B(D-1) con il prossimo successo-
       re di B(D)
    5. D=D-1 e M=1-M (nuovo ambiente)
    6. Chiama la subroutine ricorsivamente
    7. D=D+1 e M=1-M (vecchio ambiente)
    8. Poni V(D) al MAX o al MIN di V(D) secondo M
9.  NEXT I(D)
10. RETURN
```

Figura 8.

mazioni necessarie sono contenute nella subroutine.

Il programma descritto è una traccia della sola parte ricorsiva del programma di gioco totale. Ci sono altre parti principali oltre all'output grafico, ecc. Si tratta di (1) la funzione euristica, (2) il generatore della prossima mossa, e (3) il supervisore. La funzione euristica può essere una subroutine chiamata come nella linea 1 dell'esempio precedente. Il generatore della prossima mossa può essere messo direttamente dentro la subroutine o programmato come una subroutine a sé stante. Questi programmi sono specifici al particolare gioco. Il supervisore, invece, può essere tracciato in generale.

Il supervisore pone la scacchiera nella sua posizione iniziale (o in quella attuale se il gioco è in svolgimento) e chiama la subroutine ricorsiva una volta per ogni possibile mossa. La mossa che risulta avere il massimo valore è quella che sarà giocata dal calcolatore. Il supervisore è, essenzialmente, il nodo più alto dell'albero e, quindi, è un nodo *max*. Esso deve, quindi, chiamare la subroutine ricorsiva con un *min* per farla partire. Fino a che non vi è una situazione di vincita o perdita, il programma continua a giocare.

Questo programma è importante come la subroutine ricorsiva, e le due routine possono essere combinate molto semplicemente. Non dimenticate di salvare il nodo con il valore più alto nella linea 10 altrimenti verrà perso. Ciò non è necessario nella subroutine.

Anche se gli esempi precedenti non sono completi e devono essere dettagliati, mettono in luce l'enorme potenza dei programmi ricorsivi. Man mano che i linguaggi evoluti tipo Pascal diventano più popolari, i programmi ricorsivi saranno più semplici da scrivere e quindi verranno applicati in aree sempre diverse. ■

```
SUPERVISORE

1.  D=livello di profondita' scelto
2.  Costruisci la posizione iniziale della
scacchiera B(D)
3.  M=0 (MIN)
4.  Poni V(D) uguale ad un numero molto piccolo
5.  FOR I(D)=1 TO il numero dei successori di B(D)
    6. Sostituisci B(D-1) con il prossimo successore
       di B(D)
    7. D=D-1
    8. Chiama la subroutine ricorsiva
    9. D=D+1
   10. Poni V(D) al MAX di V(D) e V(D-1)
11. NEXT I(D)
12. Fai la mossa con il piu' alto V(D) e cambia la
scacchiera B(D)
13. Se hai vinto STOP
14. Ricevi la mossa dell'avversario e cambia la
scacchiera B(D)
15. Se hai perso STOP
16. GOTO 4
```

Figura 7.

Osborne 1 lo trovi, subito, alla Microtech.

Osborne 1 è un business computer veramente eccezionale. Perchè è nato da Adam Osborne, che più di chiunque ha scritto di computer. Perchè lo porti dove vuoi, piccolo, leggero, potente.

Ma soprattutto perchè puoi comunicare con gli altri computer; puoi utilizzarlo come sistema word processing, grazie a WORDSTAR®, compreso nel prezzo; o trasformarlo in formidabile sistema di calcolo, per memorizzare modelli, fare previsioni, pianificare budget, grazie a SUPERCALC®, compreso nel prezzo. Oppure, tramite MAILMERGE®, stampare, registrare liste di nominativi e indirizzi.

Non è tutto: Osborne 1 è corredato di potenti linguaggi di programmazione: M BASIC e C BASIC e dispone di accessori per tutte le applicazioni. Non è eccezionale?

Osborne 1 lo trovi in Microtech. Da subito.

Microtech

Microtech Sistemi, Via Bronzetti 20, Milano - Telefono 733.609/740.654

Distributore per l'Italia
IRET
informatica

ELETTRONICA INTEGRATA DIGITALE

di Erbert Taub e Donald Schilling

Non esiste, in lingua italiana, un libro di testo così. Chiaro, completo, moderno, ma anche rigoroso e didattico. Sono alcuni tra gli aggettivi che costituiscono la prerogativa di questo volume.
Per capire l'elettronica digitale bisogna avere delle solide conoscenze sui dispositivi a semiconduttore, soprattutto usati in circuiti di commutazione. E malgrado quest'analisi richieda una notevole complessità matematica, introducendo alcune semplificazioni, è possibile mantenere la trattazione ugualmente rigorosa e ottenere approssimazioni pienamente accettabili.
Come trascurare poi gli amplificatori operazionali, che, se a rigore non rientrerebbero nella materia, però trovano larga applicazione in sistemi completamente digitali. E poi i circuiti integrati, finalmente spiegati e analizzati in tutti i loro aspetti. Dalla vecchia logica resistore-transistor (RTL), funzionale nella sua semplicità all'esemplificazione degli aspetti fondamentali, a quella a simmetria complementare (CMOS).
Questo, però, dopo aver studiato un capitolo che, pur non richiedendo alcuna conoscenza preliminare, va a fondo dei concetti di variabili logiche, di algebra di Boole, di analisi di circuiti logici. E ancora. Via via nei vari capitoli: i flip-flop, i registri, e i contatori (sia sincroni che asincroni), i circuiti logici atti ad eseguire operazioni matematiche, le memorie a semiconduttore (RAM, ROM, EPROM, ...), l'interfacciamento tra segnali analogici e digitali (multiplexer, circuiti sample and hold, ..., convertitori d/a e a/d), i temporizzatori. Tutto con oltre 400 problemi, dai più semplici ai più sofisticati, in cui vengono presentati i circuiti tipici che si trovano nella pratica.
Un testo quindi non solo per gli specialisti e per gli studenti universitari, ma che si adatta magnificamente agli Istituti Tecnici.
Un testo che, speriamo per gli studenti, la scuola non debba scoprire tra alcuni anni.

SOMMARIO

Dispositivi Elettronici Fondamentali; Amplificatori Operazionali e Comparatori; Circuiti Logici; Logica Resistore-Transistore e Logica ad Iniezione Integrata; Logica Diodo-Transistore; Logica Transistore-Transistore, Logica ad Accoppiamento di Emettitore; Porte MOS; I Flip-Flop; Registri e Contatori; Operazioni Aritmetiche; Memorie a Semiconduttore; Interruttori Analogici; Conversione Analogico-Digitale; Circuiti di Temporizzazione; Linee di Trasmissione; Problemi; Alcuni Esempi di Specifiche.

Pag. 740 Formato 16,5x23 Cod. 204A

L. 34.500 (Abb. L. 31.050)

GRUPPO EDITORIALE JACKSON
Divisione Libri

Per ordinare il volume utilizzare l'apposito tagliando inserito in fondo alla rivista.

Lexikon

Da dove vengono i termini dell'informatica

La posizione di predominio degli Stati Uniti nel settore dell'informatica e dell'industria elettronica in generale ha imposto alle diverse lingue (francese, tedesco, italiano) il modello della terminologia anglo-americana. La gran parte dei termini specializzati sono dunque "prestati" all'italiano, e talvolta vengono usati nella grafia originale, mentre altre volte vengono "italianizzati".

L'eccezione di maggior rilievo è proprio la parola *informatica*, apparsa nel lessico italiano nel 1968 o poco prima. La sua diffusione è dovuta al largo uso che ne hanno fatto anche i politici da quando, nel 1968 appunto, un gruppo di esperti nominato dal Ministero del bilancio e della programmazione l'accolse nella redazione del cosiddetto *Progetto 80*, il rapporto preliminare al programma economico nazionale 1971-75.

Ma *informatica* non è una parola di origine italiana. Il termine deriva invece dal francese *informatique* sul quale si è modellato anche l'inglese *informatics* (oggi usato raramente) e il tedesco *Informatik*.

Informatique ha una nascita ben documentata: il termine è stato proposto nei primi anni '60 da Philippe Dreyfus, direttore generale del Centre d'analyse et programmation (CAP) in Francia, in una seduta dell'Association française de calcul et traitement de l'information. Sembra che sia stato costruito partendo da *information* e arrivando a *informatique* sull'esempio di *mathématique*, *physique*, ecc.

Informatica, dunque, derivando dal francese, è l'eccezione che conferma la regola. La quale afferma che la maggior parte dei termini del lessico dell'informatica derivano dall'anglo-americano. Parlavamo in questo senso di "prestiti linguistici". Tali prestiti possono classificarsi in due categorie: quelli che esprimono concetti per cui già esiste un termine nazionale, e quelli che indicano concetti per cui non esiste un termine nazionale corrispondente.

Nel primo gruppo troviamo parole come *computer*; in italiano viene utilizzato assieme a *calcolatore* e *elaboratore*. Tutti e tre i termini hanno oggi ragione di esistere, assieme ad altri come *calcolatrice*, per alcune sfumature di significato che le diversificano.

Nel secondo gruppo troviamo *hardware* e *software*, parole che esprimono concetti non altrimenti esprimibili in italiano, e quindi non traducibili.

Per curiosità, possiamo ricordare che in Francia hanno tentato di farlo, proponendo di volta in volta *matériel* e *programmerie*; *mecanoïde* e *programmoïde*; *mécanectronique* e *intelletronique*.

In italiano, per fortuna, ogni tentativo di tradurre in una sola parola rispettivamente il concetto di hardware e software è caduto nel nulla (alcuni tentativi: *strumentazione* e *apparecchiature* per hardware; *programmeria* e *programmatica* per software). E ora la stessa sorte toccherà a *firmware*, il termine che si pone tra hardware e software e che vuole indicare "software contenuto nell'hardware" (come le informazioni delle ROM).

Nei prossimi numeri di questa rubrica torneremo su questi argomenti, e parleremo in particolare del modo corretto di usare i termini stranieri dell'informatica. ■

L'origine dei file

La parola angloamericana *file*, che si pronuncia "fail" ha in origine molti significati. Tra gli altri: lima, persona astuta, fila, schedario, archivio. Scrive Carlo Rossetti nel libro *I tranelli dell'inglese* (Mondadori): "Come verbo vale alle volte *classificare, mettere agli atti, archiviare* [...] Come sostantivo vale anche *raccolta di documenti, di giornali* [...] Il motivo è che si chiamavano in Inghilterra *files* quelle che i nostri vecchi archivisti chiamavano *filze* [cioè fasci di documenti uniti assieme per essere collocati in archivio] e, cessato l'uso di esse, il verbo *to file* restò come sinonimo di *ordinare, classificare, registrare carte e documenti* in un pubblico ufficio..."

Oggi il termine *file* è entrato nel gergo dell'informatica col significato di *collezione ordinata di record*. In italiano si potrebbe tradurre con *archivio* (mai con *flusso* come recentemente ci è più volte capitato di vedere), ma ha avuto più fortuna il termine originale: il file, i file (al plurale, senza s finale).

SOFT REBIT COMPUTER BANK

A DIVISION OF G.B.C.

PROGRAMMI PER IL SINCLAIR ZX81

Tutti i programmi sottoelencati sono registrati su cassetta.
Se non è specificata la dicitura "1K", necessitano dell'espansione di memoria.
Sono marcate con asterisco le cassette che possono essere usate anche sullo ZX80 con ROM 8K.

CODICE	NOME E DESCRIZIONE	PREZZO VENDITA
TF/0100-01	* **SEI GIOCHI IN INGLESE (1K)** ORBIT - SNIPER - METEORS - LIFE WOLF PACK - GOLF	13.000
TF/0100-02	**GIOCHI EDUCATIVI IN INGLESE** MATEMATICA - OPERAZIONI FRAZIONI DIVERSI GRADI DI DIFFICOLTA'	13.000
TF/0100-03	* **PROGRAMMI GESTIONALI IN INGLESE** AGENDA TELEFONICA - FINANZA PERSONALE - BLOCK NOTES	13.000
TF/0100-04	* **SEI GIOCHI IN INGLESE** LUNAR LANDING - TWENTY ONE - COMBAT SUB STRIKE - COBE BREAKER - MAYDAY	13.000
TF/0100-05	* **GIOCHI EDUCATIVI IN INGLESE (1K)** OPERAZIONI ELEMENTARI PER BAMBINI CON QUATTRO GRADI DI DIFFICOLTA'	13.000
TF/0100-10	**SCACCHI IN INGLESE** SI GIOCA CONTRO IL CALCOLATORE CON DIVERSI GRADI DI DIFFICOLTA'	26.000
TF/0100-11	* **VU-CALC IN INGLESE** POTENTE STRUMENTO DI CALCOLO ADATTO A RISOLVERE DIVERSI PROBLEMI	26.000
TF/0100-12	**FANTASY GAMES IN INGLESE** GIOCHI DI FANTASIA PER TUTTI I GUSTI	26.000
TF/0101-02	* **GIOCO SCACCHI** QUATTRO LIVELLI DIFFICOLTA' - LIBERTA' DI DISPOSIZIONE PEZZI - SOLUZIONE PROBLEMI	26.000
TF/0101-04	**VISIZXCALC** POTENTE STRUMENTO DI CALCOLO ADATTO A RISOLVERE DIVERSI PROBLEMI	26.000
TF/0101-06	**UNDICI GIOCHI (1K)** DIVERTIMENTO E BUONI ESEMPI DI PROGRAMMAZIONE IN BASIC E LINGUAGGIO MACCHINA	17.000
TF/0101-08	**LABIRINTO TRIDIMENSIONALE** DIVERSI LIVELLI DI DIFFICOLTA' PROGRAMMAZIONE DI ALTO LIVELLO CON GRAFICA OTTIMA	17.000
TF/0101-10	**TRE GIOCHI SPECIAL (1K)** USATE IL SINCLAIR COME UN ORGANO VEDETE I BATTERI CHE SI RIPRODUCONO	17.000
TF/0101-12	* **GESTIONE PICCOLI ARCHIVI** GESTIONE COMPLETA DI PICCOLI ARCHIVI	17.000
TF/0101-14	* **SIMULATORE CUBO MAGICO TRIDIMENSIONALE** PER GLI APPASSIONATI DEL CUBO MENO FATICOSO DEL CUBO REALE	17.000
TF/0101-16	* **RISOLUTORE CUBO MAGICO** PER RISOLVERE IL CUBO IN POCO PIU' DI UN MINUTO	17.000
TF/0101-18	**'DEFENDER** UN PO' DI BRIVIDO CON IL SINCLAIR VELOCITA' ECCEZIONALE	17.000
TF/0101-20	**STAR-TREK** MISSIONE GALATTICA CON IMPREVISTI ED EMOZIONI. QUATTRO LIVELLI DIFFICOLTA'	17.000
TF/0101-22	**CENTIPEDE** PROVATE A DISTRUGGERE IL BRUCO CHE SI DIVIDE SE LO COLPITE - BRAVO CHI CI RIESCE!	17.000
TF/0101-24	**ASTEROIDI** UN BUON PASSATEMPO PER VOI E PER I VOSTRI AMICI	17.000
TF/0101-26	**TIRANNOSAURO** PER CHI SI ANNOIA COL LABIRINTO - GRAFICA DINAMICA E TERRORE	17.000
TF/0101-28	**ZUC** GIOCO AFFASCINANTE PER UNO O DUE GIOCATORI NON USATELO TROPPO!	17.000
TF/0102-02	* **SETTE GIOCHI** BIORITMO - 21 - CONTO ALLA ROVESCIA - HAMMURABI - ROULETTE RUSSA - FUGA DAL CASTELLI - METEORITI	22.000
TF/0102-04	* **SETTE GIOCHI** MASTER-MIND - ORBITA - GOLF - BOMBARDAMENTO LANCIA MINE - SOS SOS - CAMMELLO	22.000
TF/0102-06	* **SETTE GIOCHI** ALLUNAGGIO - SLALOM - CACCIA SOTTOMARINA - ALIENI TIRO RAPIDO - ATTACCO MARZIANO - LA GRANDE RAPINA	22.000
TF/0102-08	* **SETTE GIOCHI** SUPER AVVENTURA - SOLITARIO - REVERSE - LABIRINTO ABBATTI IL MURO - GOLF - GIU' DENTRO	22.000
TF/0102-10	* **SETTE GIOCHI** BATTAGLIA NAVALE - BUCHI NERI - ODISSEA - MEMORY ANAGRAMMI - ARMA GIOVIANA - TRENI IN CORSA	22.000
TF/0102-12	* **GESTIONE FINANZIARIA PERSONALE** POSSIBILITA' DI MEMORIZZARE I CONTI SU NASTRO	22.000
TF/0102-14	* **AGENDE:** RUBRICA INDIRIZZI ARRICCHITA - ARCHIVIAZIONE NOTIZIE CON POSSIBILITA' RICERCA	22.000
TF/0102-16	* **MATEMATICA E FISICA** FRAZIONI - STATISTICA - TEMPERATURE PROBLEMI - CONVERSIONI DI BASE	22.000
TF/0102-18	* **MATEMATICA, FISICA E VOCABOLARIO** SOMMARE DIVERTENDOSI - LA BILANCIA - CALCOLO DEI VOLUMI MOLTIPLICAZIONI - VOCABOLI	22.000
TF/0102-20	* **TOOL-KIT** STRUMENTO INDISPENSABILE AD OGNI PROGRAMMATORE CHE VOGLIA AFFINARE LE SUE ABILITA'	22.000

PROGRAMMI PER IL TRS-80 MOD. II

I programmi sottoelencati sono forniti su disco 8".

CODICE	NOME E DESCRIZIONE	PREZZO VENDITA
TF/4502-00	**INVENTORY CONTROL** 3000 ARTICOLI DI MAGAZZINO - 200 FORNITORI S. SCORTA - DIVISIONE IN CLASSI - STATISTICHE	345.000
TF/4506-00	**MAILING LIST** 3000 NOMI E INDIRIZZI IN FORMATO COMPATTO 2000 IN FORMATO ESPANSO - SELEZIONI E STAMPE	140.000
TF/4507-00	**MAILING LIST II (RICHIEDE 2 DISK)** COME IL MAILING LIST MA SE USATO CON LO SCRIPSIT PERMETTE LA STAMPA DI CIRCOLARI SELEZIONATE	210.000
TF/4512-00	**VERSA FILE II** CREATEVI IL VOSTRO SISTEMA DI CLASSIFICAZIONE AUTOMATICA DELLE INFORMAZIONI - FACILE DA USARE	125.000
TF/4511-00	**VISICALC II** SUPERPROGRAMMA CHE GESTISCE COMPLESSE PROIEZIONI E GRANDI QUANTITA' DI DATI PER SIMULAZIONI	420.000
TF/4510-00	**PROFILE II** GESTIONE DI MOLTI DATI CON MOLTI CRITERI DI SELEZIONE - COLLEGAMENTO ALLO SCRIPSIT - STAMPE	340.000
TF/4530-00	**SCRIPSIT II** UNO DEI SISTEMI DI GESTIONE DEI DATI FRA I PIU' POTENTI SUL MERCATO	620.000
TF/4540-00	**STATISTICAL ANALISYS** STATISTICHE - VARIANZE - COVARIANZE - ISTOGRAMMI - CORRELAZIONI - FREQUENZE - ECC.	180.000
TF/4701-00	**FORTRAN** STANDARD ANSI-66 - EDITORE - COMPILATORE - EDITORE DI LINEA - BIBLIOTECA SOTTOPROGRAMMI	520.000
TF/4702-00	**EDITOR/ASSEMBLER** EDITORE - MACROASSEMBLER - EDITORE DI LINEA BIBLIOTECA FORTRAN - TABELLA CORRISPONDENZE	350.000
TF/4703-00	**COBOL** VERSIONE ESPANSA ANSI-74 - ISAM MULTICHIAVE ACCEPT/DISPLAY - DEBUG - MODULO RUN-TIME	520.000
TF/4704-00	**COBOL RUN-TIME** PER L'ESECUZIONE DI PROGRAMMI SCRITTI E COMPILATI COL COBOL COMPILER	600.000
TF/4705-00	**BASIC COMPILER** ISAM MONOCHIAVE - 14 CIFRE DI CALCOLO MODULO RUN-TIME - NON COMPATIBILE COL BASIC INTERPRETE	430.000
TF/4706-00	**BASIC RUN-TIME** PER L'ESECUZIONE DI PROGRAMMI SCRITTI E COMPILATI COL BASIC COMPILER	60.000
TF/4710-00	**TEXT EDITOR** SI PUO' INTEGRARE IN OGNI LINGUAGGIO DEL MOD. 2 RICERCHE E SOSTITUZIONI GLOBALI PIU' ALTRO.	150.000
TF/4714-00	**REFORMATTER (RICHIEDE 2 DISCHI)** SCRITTURA - LETTURA - TRASFERIMENTO DI ARCHIVI TRA DISCHI TRSDOS E DISCHI IMB 3741/3742	450.000

PROGRAMMI PER IL TRS-80 MOD. III VERSIONE DISCO

La minima configurazione per l'uso dei programmi presentati è indicata a fianco del nome.
Tutti i programmi sono in inglese

CODICE	NOME E DESCRIZIONE	PREZZO VENDITA
TF/1508-00	**IN-MEMORY INFORMATION (16K)** CLASSIFICAZIONE DELLE INFORMAZIONI SALVATAGGIO E RICERCA SU DISCO	36.000
TF/1551-00	**DISK MAILING LIST** PIU' POTENTE DELLA VERSIONE SU CASSETTA	70.000
TF/1553-00	**INVENTORY CONTROL (32K 2 DISCHI)** FINO A 1000 ARTICOLI CON RAPPORTI SULLE VENDITE E LE ROTAZIONI DEL MAGAZZINO	170.000
TF/1558-00	**BUSINESS MAILING LIST (32K 2 DISCHI)** FINO A 990 NOMI - CON 48K E 4 DISCHI 2970 NOMI)	170.000
TF/1559-00	**MANUFACTURING INVENTORY CONTROL (32K 2 DISCHI)** GESTIONE DELLE DISTINTE BASE - 20 PRODOTTI FINITI e 1900 MATERIE PRIME PER DISCO	320.000
TF/1562-00	**PROFILE (32K 1 DISCO)** GESTIONE DI ARCHIVI CON RICERCHE MULTIPLE ARCHIVI ACCESSIBILI DA PROGRAMMI UTENTE	135.000
TF/1563-00	SCRIPSIT DISK (32K 1 DISCO) PROCEDURA DI TRATTAMENTO DELLA PAROLA STAMPE MULTIPLE - FACILE EDITING	150.000
TF/1565-00	**MICROFILES (32K 1 DISCO)** VERSIONE SOFISTICATA DEL PROFILE VELOCISSIMO - COMANDI A SINGOLO TASTO	185.000
TF/1567-00	**VISICALC MOD. 3 (32K 1 DISCO)** SUPERPROGRAMMA CHE PERMETTE DI LAVORARE CON PROIEZIONI E MODELLI DI SIMULAZIONE	175.000
TF/1569-00	**VISICALC AVANZATO MOD. 3 (32K 1 DISCO)** UNISCE ALLA POTENZIALITA' DEL VISICALC L'ENORME FLESSIBILITA' DEL MOD. 3	300.000
TF/1603-00	**PERSONAL FINANCE DISK (16K)** FORNITO IN VERSIONE CASSETTA PUO' ESSERE ADATTATO AL DISCO (FINO A 32K 2 DISCHI)	35.000
TF/2010-00	**DISK BASIC COURSE (16K 1 DISCO)** UN GRANDE CORSO SU 4 DISCHI CON TUTTE LE PIU' POTENTI ISTRUZIONI DEL BASIC MOD. 3	60.000
TF/1604-00	**VERSAFILE (32K 1 DISCO)** SCRIVETE QIO' CHE VI VIENE IN MENTE E IL TRS-80 LO RICORDA - CHIEDETEGLIELO!	50.000
TF/2201-00	**FORTRAN (32K 2 DISCHI)** COMPILATORE - EDITORE DI TESTI - EDITORE DI LINEA - LIBRERIA	160.000
TF/2202-00	**EDITOR/ASSEMBLER DISK (32K 2 DISCHI)** ASSEMBLATORE - EDITORE DI TESTI EDITORE DI LINEA - TABELLA DELLE CORRISPONDENZE	160.000
TF/2204-00	**BASIC COMPILER (48K 2 DISCHI)** TUTTA LA POTENZA DEL LINGUAGGIO MACCHINA DAL BASIC - INCOMPATIBILE CON IL BASIC INTERPRETE	280.000

PROGRAMMI PER IL TRS-80 MOD. III VERSIONE CASSETTA

La minima configurazione per l'uso dei programmi presentati è indicata a fianco del nome.
Tutti i programmi sono in inglese

CODICE	NOME E DESCRIZIONE	PREZZO VENDITA
TF/1502-00	**IN-MEMORY PROGRAM (16K)** CLASSIFICAZIONE DELLE INFORMAZIONI SALVATAGGIO E RICERCA	32.000
TF/1503-00	**MAILING LIST (16K)** GESTIONE INDIRIZZI CON STAMPA ETICHETTE - 80 NOMI PER VOLTA OGNI 16K	35.000
TF/1595-00	**SCRIPSIT (16K)** PROGRAMMA COMPLETO DI TRATTAMENTO DEI TESTI - MOLTO POTENTE	120.000
TF/1602-00	**PERSONAL FINANCE (4K)** GESTION ENTRATE E USCITE FAMILIARI GESTIONE BILANCIO MENSILE	30.000
TF/1603-01	**PERSONAL FINANCE DISK (16K)** FORNITO IN VERSIONE CASSETTA PUO' ESSERE ADATTATO AL DISCO (FINO A 32K 2 DISCHI)	35.000
TF/1605-00	**ASTROLOGY (16K)** PRODUZIONE DI OROSCOPI PERSONALI SE COLLEGATO AD UNA STAMPANTE PRODUCE IL QUADRO ASTRALE	50.000
TF/1701-00	**MATHEMATIC COURSE (4K)** INSEGNA AI BAMBINI LE 4 OPERAZIONI	37.000
TF/1702-00	**ALGEBRA COURSE (4K)** IMPARARE L'ALGEBRA È FACILE! - E NON È NECESSARIO ASPETTARE DI FREQUENTARE LE MEDIE!	30.000
TF/1703-00	**STATISTIC COURSE (16K)** PER IMPARARE AGEVOLMENTE E FACILMENTE AD USARE LE TEORIE STATISTICHE - ANCHE PER GRANDI	50.000
TF/1705-00	**ADVANCED STATISTICS (16K)** INTEGRA E COMPLETA IL CORSO DI STATISTICA CON QUALCOSA DI PIU' COMPLESSO	80.000
TF/1706-00	**I.Q. BUILDING (16K)** CALCOLO E MIGLIORAMENTO DEL PROPRIO QUOZIENTE D'INTELLIGENZA TRAMITE SEMPLICI TEST	50.000
TF/1712-00	**SHOW & SPELL (16K)** FACILE CORSO DI GRAMMATICA INGLESE PER BAMBINI	60.000
TF/2000-00	**DEBUG (16K)** PROGRAMMA DI CONTROLLO E DI ESECUZIONE PER PROGRAMMI IN LINGUAGGIO MACCHINA IN MEMORIA	40.000
TF/2001-00	**T-BUG (16K)** CARICA UN PROGRAMMA IN LINGUAGGIO MACCHINA DA CASSETTA E NE PERMETTE IL DEBUG	35.000
TF/2002-00	**EDITOR-ASSEMBLER (16K)** PERMETTE D'INTRODURRE UN PROGRAMMA IN LINGUAGGIO SIMBOLICO ZILOG E DI ASSEMBLARLO	50.000
TF/2003-00	**LEVEL 1 COURSE (4K)** CORSO DI BASIC LIV. 1	30.000
TF/2005-00	**BASIC COURSE LEVEL 2 PT.1 (16K)** CORSO DI BASIC ELEMENTARE - PRIMA PARTE	30.000
TF/2006-00	**BASIC COURSE LEVEL 2 PT. 2 (16K)** CORSO DI BASIC ELEMENTARE - SECONDA PARTE	35.000
TF/2009-00	**TINY PASCAL TAPE (16K)** COMPILATORE DI UN SUBSET DEL LINGUAGGIO PASCAL - POTENZIALITA' MAI VISTA!	38.000

PROGRAMMI PER IL TRS-80 POCKET COMPUTER

Tutti i programmi sono forniti su cassetta e sono in inglese

CODICE	NOME E DESCRIZIONE	PREZZO VENDITA
TF/3511-00	**CIVIL ENGINEERS** PROGRAMMI DI INGEGNERIA - CALCOLO TELAI - SFORZI AI BULLONI - TRAVI INCASTRATE - ECC.	42.500
TF/3513-00	**AVIATION** CALCOLO DEL PIANO DI VOLO - ANGOLO DI DERIVA CONVERSIONI TRA UNITA' DI MISURA - ECC.	42.500
TF/3514-00	**MATH DRILL** ESERCIZI PER GLI SCOLARI DELLE PRIME CLASSI POSSIBILITA' DI INTRODURRE NUOVI PROBLEMI	38.000
TF/3515-00	**GAMES ONE** CANNIBALI E MISSIONARI - NIM - ATTERRAGGIO NELLO SPAZIO - CACCIA AL TESORO - ECC.	38.000
TF/3516-00	**BUSINESS MARKETING** METODO DELLA MEDIA MOBILE PER IL CALCOLO E LA CORREZIONE AUTOMATICA DELLE PREVISIONI - ECC.	35.000
TF/3517-00	**BUSINESS FINANCE** SETTE PROGRAMMI DIFFERENTI PER AIUTARE L'UOMO D'AFFARI - CALCOLI INTERESSI - GIORNI - ECC.	35.000
TF/3518-00	**PERSONAL FINANCE** GESTIONE DEL BILANCIO FAMILIARE - GESTIONE C/C BANCARIO - INTERESSI - CONVERSIONI - ECC.	35.000

PROGRAMMI PER IL TRS-80 COLOR COMPUTER

Tutti i programmi sono distribuiti sotto forma di CARTRIDGE (memoria allo stato solido).
Tutti i programmi sono in inglese.
Tutti i programmi contrassegnati da asterisco richiedono l'uso di joystick.

CODICE	NOME E DESCRIZIONE	PREZZO VENDITA
TF/3019-00	**ROM DIAGNOSTICA** CONTROLLO DELLA PERFETTA EFFICIENZA DEL VOSTRO CACOLATORE	39.000
TF/3050-00	**SCACCHI** DA ALLENAMENTO, MA ANCHE DA COMBATTIMENTO!	90.000
TF/3051-00	* **QUASAR COMMANDER** RADAR - PILOTA AUTOMATICO - CAMPI DI FORZA DIVERSI LIVELLI DIFFICOLTA'	60.000
TF/3052-00	* **PINBALL** IL CLASSICO GIOCO DEL FLIPPER ORA ANCHE SUL TELEVISORE - DA 1 A 4 GIOCATORI	60.000
TF/3055-00	**CHECKERS** GIOCO DELLA DAMA A DUE LIVELLI DI DIFFICOLTA' PREVEDE LE 3 MOSSE SUCCESSIVE	60.000
TF/3056-00	* **SUPER BUSTOUT** GIOCO RAPIDO PER 1-4 GIOCATORI - SINGOLO O IN EQUIPE - SFONDATE LE LINEE COL PALLONE	60.000
TF/3057-00	* **DINO WARS (16K CONSIGLIATI)** DUE GIOCATORI ALLE PRESE CON I DINOSAURI GRAFICA E SONORO REALISTICI	70.000
TF/3058-00	* **SKILING (16K CONSIGLIATI)** DISCESA SCIISTICA CONTRO IL TEMPO VISTA CON GLI OCCHI DELLO SCIATORE	70.000
TF/3059-00	* **COLOR BACKGAMMON** CLASSICO GIOCO DI SOCIETA' - CONTRO IL CALCOLATORE O UN ALTRO AVVERSARIO	60.000
TF/3060-00	* **SPACE ASSAULT** GLI EXTRATERRESTRI VI INVADONO LO SCHERMO E VI ATTACCANO! - BUONA FORTUNA!	50.000
TF/3061-00	* **ART GALLERY (16K CONSIGLIATI)** CREATE LA VOSTRA GALLERIA DI QUADRI MODERNI - CONSIGLIATI I JOYSTICK	80.000
TF/3063-00	* **PROJECT NEBULA** RESPINGETE GLI INVASORI DELLA VOSTRA GALASSIA - 4 LIVELLI - APPASSIONANTE!	90.000
TF/3103-00	**COLOR FILE** PICCOLO SISTEMA DI GESTIONE PER TANTI ARCHIVI - SI USA COL REGISTRATORE A CASSETTE	60.000
TF/3101-00	**PERSONAL FINANCE** PIANIFICATE IL BUDGET FAMILIARE COMPARATE ENTRATE E USCITE - PREVEDETE IL BILANCIO	90.000

TF/3151-00	* **BINGO MATH** INSEGNA LE 4 OPERAZIONI E IL RICONOSCIMENTO DEI NUMERI - 1-2 GIOCATORI	60.000
TF/3152-00	**TYPING TUTOR** ESERCIZI BASATI SU LETTERE E PAROLE CONTROLLA VELOCITA' - RIFLESSI - ERRORI	60.000
TF/3153-00	**LEARNING LAB** COMBINAZIONE DI LOGICA E TESTI PER INSEGNARE IL COLOR BASIC - ORGANIZZAZIONE E STESURA	80.000
TF/3154-00	**HANDY MAN** CALCOLO DELLE ESATTE NECESSITA' DEL LAVORO DEL BRICOLAGE - MATERIALI - CONSIGLI	60.000

PROGRAMMI PER IL BMC IF 800 MOD. 20

Tutti i programmi sottoelencati sono forniti su disco 5".

CODICE	NOME E DESCRIZIONE	PREZZO VENDITA
TF/2502-00	**FORTRAN-80 (RICHIEDE IL CP/M)** EDITORE - COMPILATORE - EDITORE DI LINEA ANSI-66	800.000
TF/2504-00	**BASIC COMPILER (RICHIEDE IL CP/M)** RENDE PIU' VELOCI I PROGRAMMI IN BASIC INTERPRETE	650.000
TF/2506-00	**MBASIC (RICHIEDE IL CP/M)** BASIC INTERPRETE	300.000
TF/2508-00	**T-MAKER 2 (RICHIEDE IL CP/M)** GESTIONE DI TESTI E ARCHIVI IN COMBINAZIONE CON TUTTI I TIPI DI CALCOLO NUMERICO	700.000
TF/2510-00	**SUPERCALC (RICHIEDE IL CP/M)** IL VOSTRO FOGLIO ELETTRONICO A COLORI CALCOLI E PRVISIONI FINANZIARIE	500.000
TF/2512-00	**WORD STAR (RICHIEDE IL CP/M)** L'ULTIMO E IL PIU' PERFEZIONATO PROGRAMMA PER GESTIONE DI TESTI - PUO' TUTTO!	800.000
TF/2514-00	**WORD INDEX (RICHIEDE IL CP/M)** IN ABBINAMENTO AL WORD STAR PERMETTE LE STAMPE DI MANUALI - INDICE E RIASSUNTI AUTOMATICI	300.000
TF/2516-00	**COBOL-80 (RICHIEDE IL CP/M)** COMPILATORE ANSI-74 - ACCEPT/DISPLAY - EDITORE	1.300.000
TF/2518-00	**DBMS (RICHIEDE IL CP/M)** GESTIONE COMPLETA DI GRANDI ARCHIVI RICERCHE MULTICHIAVE - STAMPE DI TUTTI I TIPI	1.000.000
TF/2520-00	**ARCHIVI (IN OKI-BASIC)** IL DISCO CONTIENE DIVERSI PROGRAMMI DI ARCHIVIO PIU' UN DEMO E UN PROGRAMMA TYPEWRITER	400.000

PROGRAMMI PER IL COMMODORE (LINEA 3000 - 4000 - 8000)

Tutti i programmi sottoelencati sono forniti su disco 5".
Per ogni programma verrà specificato il modello.

CODICE	NOME E DESCRIZIONE	PREZZO VENDITA
TF/1102-00	**FATTURAZIONE MANUALE (3000)** GESTIONE CLIENTI - EMISSIONE FATTURE E TRATTE - SENZA CODIFICA MAGAZZINO	700.000
TF/1104-00	**GESTIONE CONDOMINII (3000)** GESTIONE DI PIU' SCALE - EMISSIONE AUTOMATICA LETTERE - CIRCOLARI - SOLLECITI	800.000
TF/1106-00	**GESTIONE CONDOMINII (4000)** GESTIONE DI PIU' SCALE - EMISSIONE AUTOMATICA LETTERE - CIRCOLARI - SOLLECITI	800.000
TF/1108-00	**GESTIONE COMDOMINII (8000)** GESTIONE DI PIU' SCALE - EMISSIONE AUTOMATICA LETTERE - CIRCOLARI - SOLLECITI	800.000
TF/1110-00	**WORD PROCESSOR (8000)** PROCEDURA COMPLETA DI TRATTAMENTO DEI TESTI PERMETTE CIRCOLARI SELEZIONATE	630.000
TF/1112-00	**ASSEMBLER (3000)** EDITORE - ASSEMBLATORE SIMBOLICO 6502	115.000
TF/1114-00	**PASCAL (3000)** SUBSET UCSD PASCAL - COMPILATORE - EDITORE	115.000
TF/1116-00	**GESTIONE LABORATORI ANALISI MEDICHE (3000)** GESTIONE COMPLETA DI UN LABORATORIO - STAMPA I DOCUMENTI PER GLI ENTI - STATISTICHE	900.000
TF/1118-00	**GESTIONE LABORATORI ANALISI MEDICHE (4000)** GESTIONE COMPLETA DI UN LABORATORIO STAMPA I DOCUMENTI PER GLI ENTI - STATISTICHE	900.000
TF/1120-00	**GESTIONE LABORATORI ANALISI MEDICHE (8000)** GESTIONE COMPLETA DI UN LABORATORIO STAMPA I DOCUMENTI PER GLI ENTI - STATISTICHE	900.000
TF/1122-00	**VISICALC (4000 + ROM AGGIUNTIVA FORNITA)** SUPERPROGRAMMA PER GESTIONE DATI NUMERICI PROIEZIONI - SIMULAZIONI	310.500
TF/1124-00	**VISICALC (8000 + ROM AGGIUNTIVA FORNITA)** SUPERPROGRAMMA PER GESTIONE DATI NUMERICI PROIEZIONI - SIMULAZIONI	310.500
TF/1126-00	**COM-PLUS (8000)** UTILE ACCESSORIO PER SUPERARE LA BARRIERA DELL'INCOMPATIBILITA' TRA I DIVERSI SISTEMI	60.000
TF/1128-00	**WORD-CRAFT (8000 + CHIAVE D'ACCESSO)** ALTRA VERSIONE DI WORD PROCESSOR CON CARATTERISTICHE ADERENTI AD ESIGENZE DIVERSE	632.500
TF/1130-00	**VIGIL (3000)** LINGUAGGIO ORIENTATO ALLA PRODUZIONE DI GIOCHI SONORI E GRAFICI - 9 GIOCHI ESEMPIO FORNITI	120.000

PROGRAMMI PER IL VIC-20 CBM

Tutti i programmi sottoelencati sono registrati su **cassetta.**
Se non specificato, si intende che i programmi funzionano con la memoria in configurazione base.

CODICE	NOME E DESCRIZIONE	PREZZO VENDITA
TF/9402-00	**THE ALIEN WITH JOYSTICK (6K)** PROVATE A CALARVI NEI PANNI DELL'ALIENO!	60.000
TF/9404-00	**AMOK** UN GIOCO DI COMBATTIMENTO E DI VIOLENZA	60.000
TF/9406-00	**THE ALIEN** SIETE L'ALIENO E DOVETE SOPRAVVIVERE!	60.000
TF/9408-00	**3-D MAZE** TROVATE L'USCITA DAL LABIRINTO TRIDIMENSIONALE! DIVERSI LIVELLI DI DIFFICOLTA'	36.000
TF/9410-00	**ALIEN BLITZ (JOYSTICK OPZIONALE)** DISTRUGGETE GLI INVASORI DEL CIELO!	60.000
TF/9412-00	**VICAT** GESTIONE DI UN ARCHIVIO SEQUENZIALE SU CASSETTA	60.000
TF/9300-00	**CASSETTA PROGRAMMI DIMOSTRATIVI** DIMOSTRA LA POTENZA DEL VIC	15.700

PROGRAMMI PER IL VIC-20 CBM

Tutti i programmi sottoelencati sono registrati su **cartridge.**
Se non specificato, si intende che i programmi funzionino con la memoria in configurazione base.

CODICE	NOME E DESCRIZIONE	PREZZO VENDITA
TF/9300-04	**INVASORI SPAZIALI** GRANDE REALISMO - ALTA VELOCITA' NON VI FATE PRENDERE DAL PANICO!	37.000
TF/9300-06	**GARA AUTOMOBILISTICA** PROVATE L'EBREZZA DELLA VELOCITA' E DELLA COMPETIZIONE - 1 o PIU' GIOCATORI	37.000
TF/9300-08	**ATTERRAGGIO SU GIOVE** ESSERE AL COMANDO DI UNA ASTRONAVE NON È SEMPLICE MA QUESTO LO IMPARERETE A VOSTRE SPESE	37.000
TF/9300-10	**GIOCO DEL POKER** ATTENZIONE! - POTRESTE RESTARE POVERI! QUI NON SI TRATTA DI FORTUNA	37.000
TF/9300-12	**IL FANTASMA DI MEZZANOTTE** FUGGITE VIA DALLA CASA INFESTATA DAGLI SPIRITI SE VE NE RIMANE IL TEMPO	37.000
TF/9300-14	**BILANCIO FAMILIARE** PIANIFICATE LE VOSTRE SPESE IN FUNZIONE DELLE ENTRATE GESTITE IL VOSTRO C/C BANCARIO	37.000
TF/9300-16	**APPLICAZIONI MATEMATICHE** UN VALIDO AIUTO TESO AL MIGLIORAMENTO DELLE PROPRIE CAPACITA' DI CALCOLO	37.000
TF/9300-18	**SLOT MACHINE** IL CELEBRE GIOCO D'AZZARDO	37.000
TF/9300-20	**AVENGER** INTERESSANTE GIOCO DI SIMULAZIONE	37.000

PROGRAMMI PER APPLE II

Tutti i programmi sono forniti su disco.
Per ogni programma è indicata la lingua (italiano-inglese) in cui è stato scritto.
Ove non indicato, si intende che i programmi girano sulla configurazione 16K 1 disco.

CODICE	NOME E DESCRIZIONE	PREZZO VENDITA
TF/5502-00	**TOTOCALCIO SISTEMA A CORREZIONE D'ERRORI (It.)** ELABORAZIONE DI SISTEMI RIDOTTI	80.000
TF/5504-00	**TOTOCALCIO CHIAVE ALFA 6 SUPER (It.)** SISTEMA RIDOTTO 10 TRIPLE CON FATTORE DI RIDUZIONE - INDICATO AL SISTEMISTA SERIO	70.000
TF/5506-00	**TOTOCALCIO SISTEMA DERIVATO A ROTAZIONE (It.)** ELABORAZIONE DI UN NUMERO STABILITO DI COLONNE IN BASE AD UN NUMERO CONCORDATO DI ELIMINAZIONI	90.000
TF/5508-00	**APPLE PANIC (Ing.)** LABIRINTO DI SCALE - ALTA RISOLUZIONE GRAFICA AD ALTA VELOCITA'	72.000
TF/5510-00	**ADVENTURES 1/2/3 (Ing.)** ADVENTURELAND - PIRATE'S ADVENTURE - MISSION IMPOSSIBLE	110.000
TF/5512-00	**ADVENTURES 4/5/6 (Ing.)** VODOO CASTLE - THE COUNT - STRANGE ODYSSEY	110.000
TF/5514-00	**ADVENTURES 7/8/9 (Ing.)** MYSTERY - FUN HOUSE - PYRAMID OF DOOM GHOST TOWN	110.000
TF/5516-00	**FLIGHT SIMULATOR (Ing.)** UN REALISTICO SIMULATORE DI VOLO CON VISTA DAL CIELO E DALL'AEREO - ANCHE FASI DI COMBATTIMENTO	60.000
TF/5518-00	**COMPUCUBE (Ing.)** CREARE - RIMESCOLARE - RISOLVERE IL CUBO MAGICO - TRIDIMENSIONALE	72.000
TF/5520-00	**DRAW POKER (Ing.)** IL MIGLIOR PROGRAMMA NEL SUO GENERE	72.000

Nino Tonolli

Proverbi per informatici

"La preparazione di un programma è cosa particolarmente attraente perché può essere vista non solo sotto l'aspetto scientifico, ma anche come esperienza estetica, come il comporre musica o scrivere poesia". Questa frase di D.E. Knuth, uno dei maggiori informatici internazionali, illustra il nostro punto di vista nello scrivere questa rubrica.

Forse tra qualche decina d'anni molto sarà cambiato, ma oggi lo scrivere un programma è ancora una questione che riguarda non solo il razionale ma anche, ed in larga misura, il senso artistico. In altre parole, la disciplina della programmazione non è ancora codificata in regola precise: non si tratta di scienza, ma di un'arte.

Parlare del "buon modo di programmare" significa parlare di esperienza e di buon senso, significa dare consigli più che norme, criteri più che leggi.

Nella prima apparizione di questa rubrica, vogliamo farvi conoscere i cosiddetti "proverbi di Arsac". Jacques Arsac è professore di informatica alla facoltà di Scienze dell'Università di Parigi. Egli distribuisce agli studenti che intraprendono lo studio dell'informatica, una piccola dispensa ciclostilata che contiene diciotto consigli sul modo di programmare, da lui chiamati «proverbi».

I proverbi di Arsac

0. Le cattive abitudini in programmazione si prendono molto presto. Non attendete di esserne schiavi per leggere le raccomandazioni che seguono. Esse vi eviteranno tempo perduto per fastidiose messe a punto e vi agevoleranno nello studio e nel lavoro.

Tuttavia, non prendete queste raccomandazioni per regole assolute. Se sono state chiamate "proverbi", è per rimarcare il loro carattere di "saggezza popolare", buon senso comune, frutto dell'esperienza. Il buon programmatore sa passarvi sopra se ne vale la pena. Rispettatele fino a che la vostra esperienza non vi avrà fatto capire i loro limiti.

1. *Enunciate il problema il più dettagliatamente possibile.* In informatica, non solo nello studio ma specialmente nella vita comune, non partirete mai da un enunciato perfettamente definito. Avrete a che fare con una situazione descritta in italiano corrente, talvolta con delle formule per farvi credere che il problema sia ben posto. Non lasciatevi ingannare. Cominciate a definire il problema da risolvere. In particolare:

- precisate i dati, il loro tipo, il loro campo di variazione, le loro proprietà;
- precisate i risultati che vi si chiedono;
- formulate le relazioni che legano i risultati ai dati.

La scuola non vi ha certo abituato a questo lavoro: vi ha insegnato a risolvere problemi, non a porli. Quello che più si avvicina a questa attività, sono i vecchi problemi di aritmetica della scuola elementare: se una vasca viene riempita da un rubinetto... Non c'erano difficoltà di risoluzione: l'essenziale era porre il problema. Rimettetevi in questa ottica.

2. *Intanto riflettete, programmerete più tardi.* Non precipitatevi a scrivere istruzioni. Questa è veramente l'ultima tappa del processo. Avrete ancora molto da fare:

- studiate l'enunciato del problema; cercate di trovare tutte le ipotesi semplificatrici che vi faciliteranno il lavoro;
- a partire dalle proprietà dei dati fate apparire ciò che già sapete sui risultati. Inutile fare un programma se il risultato può essere calcolato direttamente. Questo lo si sa...
- scegliete un metodo di soluzione (quello che in gergo si chiama algoritmo);
- non vi preoccupate troppo per il momento del numero di operazioni che richiede: non siete voi che lo eseguirete, ma la macchina. Invece interessatevi alla precisione, se fate dei calcoli numerici;
- scegliete una rappresentazione dei dati non a priori, ma in funzione del vostro algoritmo. La rappresentazione deve facilitare l'esecuzione dell'algoritmo.

Proverbi per informatici

È del tutto normale che passiate il 60% del vostro tempo a riflettere, tempo durante il quale non scrivete una sola istruzione. Quelli che sono stati abituati a scrivere subito, a provare il programma sulla macchina, a riflettere soltanto allora sugli errori commessi, vi diranno che perdete il vostro tempo. Sono loro che lo perdono: vi accorgerete di arrivare allo scopo prima di loro, con meno fatica.

3. *Documentate il vostro lavoro.* Non fidatevi della vostra memoria. — Scrivete l'enunciato del problema. Se vi accorgete che è incompleto, cambiatelo. Non fate alcuna modifica che non sia riportata su tutto l'insieme, dall'enunciato del problema alla sua soluzione.

- descrivete con cura l'algoritmo che utilizzate;
- precisate bene quali dati trattate, e in quale campo si devono trovare affinché il programma funzioni;
- dite quali sono i risultati, in che ordine appaiono, cosa significano.

Non attendete d'aver finito per redigere la documentazione. La vostra memoria non è infallibile. Vi perderete voi stessi nelle convenzioni che sarete inevitabilmente condotti ad adottare.

Non dite mai: non è definitivo, lo devo ancora cambiare, quindi non lo scrivo. Scrivetelo, poi lo correggerete se necessario.

4. *Conservate i vostri appunti.* Non si scrive mai un programma di getto, non si stende un enunciato immediatamente senza errori, né un algoritmo senza dimenticare qualche cosa. In programmazione vi è "ritorno all'indietro" senza fine. Mantenete i vostri appunti. Vi capiterà di tornare indietro quando sarete sulla buona strada. Vi capiterà di modificare certi punti e di perdere cammin facendo delle parti corrette.

Conservare i vostri appunti vi faciliterà la ripresa. Allo stesso tempo questi appunti costituiscono la storia della creazione del vostro programma. Vi saranno indispensabili nella fase finale di documentazione.

5. *Fate uno studio discendente.* Tracciate dapprima la soluzione a grandi linee, senza occuparvi dei dettagli. Assicuratevi che questa risolva il problema dato, eventualmente dimostratelo.

A questo livello non parlate del modo in cui i dati saranno effettivamente rappresentati nel programma. Non ha importanza, ed è ancora troppo presto.

Quello che avete fatto non è ancora un programma. Ci saranno delle espressioni in italiano corrente che saranno dettagliate più tardi.

Quando avrete uno schema d'insieme corretto, affrontate le azioni che lo compongono. Per ciascuna, ricominciate lo stesso processo. È il metodo del "divide ed impera": separate il vostro problema in sottoproblemi. Supponete di saperli risolvere, e mostrate come essi permettono di risolvere il problema principale. Dopo di ciò, affrontate i sottoproblemi, e così di seguito.

fino a che raggiungete dei problemi semplici che sapete programmare.

Un esempio: vi si domanda di calcolare il valore di un polinomio. Non dite: bè, so come si fa: devo calcolare le potenze della variabile, moltiplicarle per i coefficienti e accumulare il risultato; faccio un ciclo... NO.

Enunciate il problema. Quale polinomio? Di quale grado? Il grado può essere nullo? Quali sono i valori possibili della variabile? Il calcolo si deve fare per un punto, per un insieme di valori qualunque, o in progressione aritmetica? I coefficienti e i valori della variabile sono interi (nel qual caso non si saranno problemi di precisione, ma rischi di valori troppo grandi) o possono essere reali (attenzione agli errori di arrotondamento).

Immaginate che i coefficienti siano interi, compresi nell'insieme {0, 1, 2, ..., 9} e la variabile abbia il valore 10. Scriverete un programma in questo caso?

Supponete che il problema si enunci così:

- Sia dato un polinomio mediante la successione dei suoi coefficienti, secondo l'ordine del grado discendente, e la successione comprenda almeno un termine (nel caso che contenga un termine, è un polinomio di grado 0, una costante). Questi coefficienti sono reali.

- Sia data una successione di valori della variabile, reale, in ordine qualunque, ma non nullo.

Calcolate, alla precisione della macchina, il valore del polinomio su questa successione di valori della variabile. Si può cominciare a organizzare il lavoro:

1 Leggere la successione dei coefficienti determinandone il numero i; grado $\leftarrow i-1$; posizionarsi all'inizio della successione dei valori della variabile.

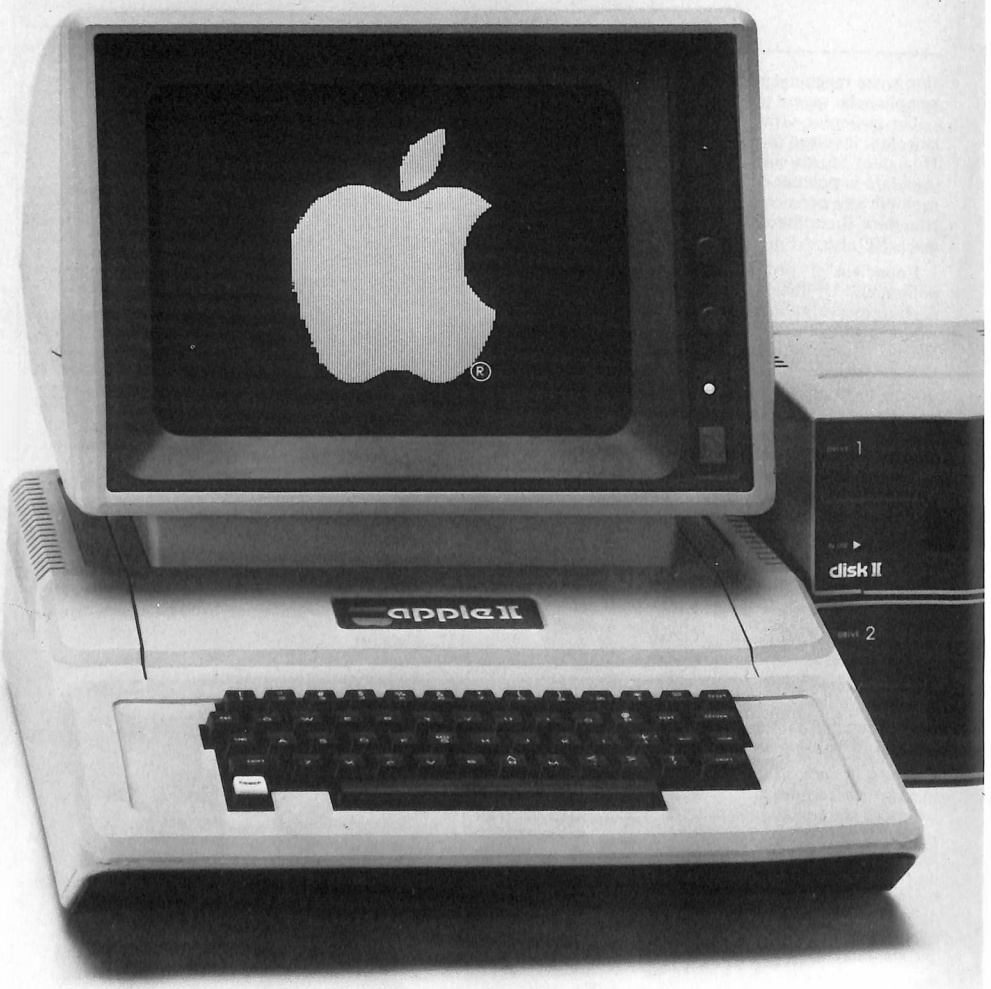
2 Considerare il valore attuale x; calcolare il valore y del polinomio in x; stampare x e y; avanzare nella successione dei valori della variabile; se ci sono ancora valori da trattare andare a **2**, altrimenti stop.

Si ha un buon abbozzo del programma. Alcune difficoltà non sono state ancora risolte, ma ciascuna è stata localizzata. Si potrà ora decidere l'algoritmo di calcolo del valore del polinomio, per esempio il metodo di Horner.

E ancora troppo presto per decidere come rappresentare le successioni dei coefficienti e di valori della variabile. Non se ne sa abbastanza su quello che resta da fare. ■

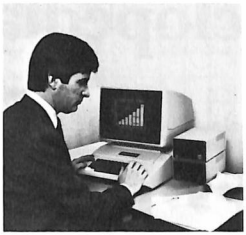
I "proverbi di Arsac" continuano nel prossimo numero.

Dalle un morso



e crescerai.

Per avere successo, occorre crescere.
Per questo tu stai sempre cercando qualcosa che ti aiuti a crescere, a migliorare le tue capacità, a sfidare gli ostacoli.
Apple, il computer personale, ti aiuta a crescere più in fretta e ad andare più lontano di quel che credevi possibile.



In altri termini, un elaboratore personale Apple ti assicura il poterè di mandare ad effetto tutti i pensieri creativi che ti balenano nella mente. Per esempio, prendiamo la previsione. Mentre ogni elaboratore può rielaborare dei risultati per te se tu inserisci una serie di informazioni, tu non sempre hai le ore o le nottate (o non ancora l'accesso ad un elaboratore) per il normale cambiamento.

Quante volte le tue brillanti idee - del tipo "Cosa succede se..." - muoiono perchè tu non puoi metterle in atto? Ora spingi al lavoro un elaboratore personale Apple ed avrai tutte le informazioni/analisi/opzioni che risponderanno al tuo "Cosa succede se..." un minuto dopo che avrai posto la domanda. Immagina che cosa vuol dire!
Dopo che avrai individuato la soluzione migliore (lasciando la noia del lavoro di routine al tuo Apple) sarai in grado di organizzare la massa dei dati e dei loro rapporti reciproci in immagini comprensibili ed operative per sostenere le tue conclusioni. Il tuo Apple ti dà possibilità illimitate: tu non hai che da scegliere quel procedimento grafico che sostenga meglio il tuo punto di vista. Cambi idea? Il tuo Apple risponde più velocemente di quel che immagini alla nuova linea prescelta.

Puoi anche usare il medesimo Apple per redigere e produrre i testi. E puoi usarlo per avere sulla punta delle dita un mucchio di dati importanti, per lavorare in base a notizie del momento (e sarà finita la storia di qualcuno che blocca le tue decisioni infilando nel posto sbagliato i tuoi dati-chiave!).

E questo è solo l'inizio. Gli elaboratori personali Apple sono all'avanguardia per le migliaia dei loro programmi d'applicazione e per le centinaia dei loro accessori, dalla stampatrice alla derivazione telefonica. Così il tuo Apple può crescere alla velocità alla quale cresci tu e non ti lascerà mai indietro nella tua corsa verso quel mobile obiettivo che è il "successo".

Quel che più conta, Apple continua ad immettere sul mercato, sempre migliori e sempre più numerosi, accessori e software pienamente compatibili con quelli già esistenti.
Questo significa nessun pensiero quando decidi di imboccare la "corsia di sorpasso" Apple: sarai sempre in grado di accelerare insieme ad ogni futura innovazione Apple.

Non per nulla oggi ci sono già più di 350.000 Apple in uso. L'elaboratore personale Apple è la via maestra per crescere più in fretta e per andare avanti.
E chi non vuole questo?
Per sapere come Apple può aiutare anche te a crescere, spedisci questo tagliando oggi stesso.

Per saperne di più sui sistemi Apple II e Apple III di personal computer, compilate questo coupon e speditelo in busta chiusa a: Iret Informatica S.p.A. - Via Bovio, 5 - 42100 Reggio Emilia Tel. 0522/32643.

Nome _____ Cognome _____

Società _____ Qualifica _____

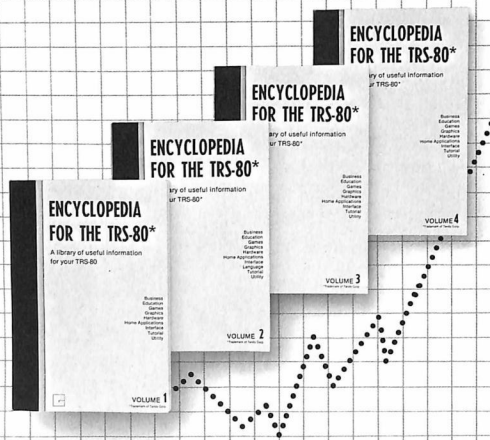
Via _____ n. _____ Città _____

Cap. _____ Tel. _____

apple®

Il computer personale.

Encyclopedia for the TRS-80*
e Encyclopedia Loader



*Encyclopedia for the TRS-80**

Qual è la chiave per avere il massimo dal vostro TRS-80*? No, non sono i dischi, le stampanti o i joystick. È l'informazione. Senza un continuo flusso di informazioni e idee non potete sfruttare la potenzialità del vostro TRS-80*.

La risposta a questa domanda di informazione è l'*Encyclopedia for the TRS-80**, un'opera in dieci volumi che contiene programmi e articoli attentamente scelti per aiutarvi a ricavare il massimo dal vostro personal computer.

Ogni volume dell'*Encyclopedia* comprende circa 300 pagine di formato 15×23 cm, più di venti programmi e cinque articoli di harware e software. I primi quattro volumi sono già disponibili. I prossimi sei verranno pubblicati entro la prima metà dell'82.

* TRS-80 è marchio depositato della divisione Radio Shack della Tandy Corp.

Encyclopedia Loader

Gli editori dell'*Encyclopedia* hanno realizzato una serie di dieci cassette ognuna delle quali riporta i programmi contenuti nel rispettivo volume dell'*Encyclopedia*. Questa serie di cassette è chiamata *Encyclopedia Loader*. Nell'*Encyclopedia for the TRS-80** è contenuta tutta la documentazione necessaria, ma con *Encyclopedia Loader* potete caricare i vostri programmi istantaneamente.

***Encyclopedia for the TRS-80** e Encyclopedia Loader* **sono pubblicati negli Stati Uniti dalla Wayne Greene Books e vengono importati in Italia in versione originale dalla Completo Software, che li propone ad ogni possessore di TRS-80* modello I o III.**

Ogni volume dell'*Encyclopedia for the TRS-80 è offerto a 16.000 lire.**

Ogni cassetta dell'*Encyclopedia Loader* è offerta a 20.000 lire.

Le spese di spedizione sono gratuite.

Spedire a
COMPLETO SOFTWARE
Via Bonporti 32
35100 PADOVA

Voglio ricevere i seguenti volumi di *Encyclopedia for the TRS-80** e i seguenti nastri di *Encyclopedia Loader*:

☐ Pagherò contrassegno
☐ Accludo un assegno
☐ Ho versato l'importo su c/cp 16915357 intestato a Completo Software

vol. 1	☐ Encyclopedia	☐ Loader
vol. 2	☐ Encyclopedia	☐ Loader
vol. 3	☐ Encyclopedia	☐ Loader
vol. 4	☐ Encyclopedia	☐ Loader

Totale lire. . . . . .

Nome e cognome

Indirizzo

Cap. Località

PERSONAL SOFTWARE

Raccolta di routine Basic

Punteggiatura

In questa rubrica raccogliamo le routine Basic che ci sembrano di interesse generale. Ne verrà pubblicata una per ogni numero. Invitiamo i lettori a comunicarci eventuali modifiche che migliorino (per velocità, occupazione di memoria o generalità di applicazione) le routine pubblicate. Questo mese iniziamo con una routine per la punteggiatura dei numeri interi "all'italiana", cioè l'inserimento di un punto di separazione tra le centinaia e le migliaia, tra le centinaia di migliaia e i milioni e così via. La routine inoltre incolonna il numero appoggiandolo a destra in una posizione stabilita.

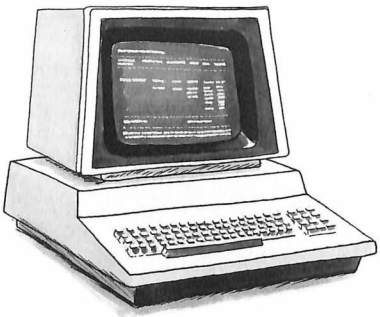
Come regola generale, il listato contiene tutte le informazioni per usare la routine. Le linee 1, 2 e 3 sono un esempio di utilizzazione. La routine inizia alla riga 1000, dove sono specificati scopo, variabili (in input o in output) e vincoli di tali variabili (controllati dalla routine oppure no). Segue quindi il corpo della routine.

```
1 INPUT N,X
2 GOSUB 1000
3 GOTO 1
1000 REM:    SCOPO
1005 REM:    METTE LA PUNTEGGIATURA ALL'ITALIANA NEI NUMERI
1010 REM:    INTERI CON PIU' DI TRE CIFRE E LI INCOLONNA A
1020 REM:    DESTRA IN UNA POSIZIONE DETERMINATA
1030 REM
1040 REM:    VARIABILI
1050 REM:    N    NUMERO DA PUNTEGGIARE (INPUT)
1060 REM:    X    POSIZIONE DI INCOLONNAMENTO (INPUT)
1070 REM:    X$   NUMERO PUNTEGGIATO (OUTPUT)
1080 REM
1081 REM:    VINCOLI
1082 REM:    X DEVE ESSERE MAGGIORE DI    INT(C/3)+C    DOVE
1083 REM:      C SONE LE CIFRE DI N (VINCOLO NON CONTROLLATO)
1084 REM
1085 REM:    **********    INIZIO ROUTINE    **********
1086 REM
1090 N$=STR$(N)
1100 X$=""
1110 L=LEN(N$)-2
1120 IF L>2 GOTO 1150
1130 I=L
1140 GOTO 1180
1150 FOR I=L TO 3 STEP -3
1160 X$="."+MID$(N$,I,3)+X$
1170 NEXT
1180 X$=LEFT$(N$,I+2)+X$
1190 PRINT TAB(X-LEN(X$));X$
1200 RETURN
```

Commodore è alla Homic

Vieni alla Homic, e fatti mostrare un "personal" Commodore: li trovi tutti, dall'eccezionale Vic 20 Colour Computer, che permette di lavorare con 24 colori, produce suoni e musica ed è collegabile con ogni apparecchio televisivo e risolve problemi scolastici, di divertimento e tecnico scientifici, alla Serie CBM destinata a trattare quantità medie e grandi di dati per la gestione della casa, degli studi professionali e delle piccole aziende.

Vieni alla Homic: trovi il meglio.



HOMIC

il più grande centro italiano di microcomputer

Uffici: Piazza De Angeli 3 - Milano - Tel. 4695467-4696040-4984583

Conversioni grafiche tra TRS-80, Apple e PET

Ecco quali sono le capacità grafiche dei personal computer più diffusi, e come si possono convertire i programmi scritti per un calcolatore in modo che possano essere eseguiti su di un altro.

di *Richard Kaplan*

Era attaccato al suo computer già da molte ore: stava battendo la lista del suo gioco preferito. "Convertire questo programma per la mia macchina dovrebbe essere un gioco da ragazzi" pensava. "Dopotutto sono un vecchio professionista della programmazione con l'Apple. Un TRS-80 non sarà poi così differente".

Dopo alcune settimane di infruttuosa programmazione, si arrese al suo computer. Aveva scoperto quanto può essere dura la vita (in questo caso la conversione di un programma) per chi conosca solo il proprio computer.

La conversione grafica è forse uno dei problemi più frustranti con cui un utente di microcomputer abbia a che fare. Ad un programmatore di Apple, l'istruzione

PRINT@1000, A+B

può sembrare un modo di istruire il proprio computer ad aspettare fino alle dieci in punto prima di stampare A+B. Per lo stesso motivo, un programmatore di TRS-80 non sa immaginare il significato dell'istruzione HGR: gli viene solo in mente che possa essere l'abbreviazione di *hungarian*.

Quindi, molto spesso, programmatori estremamente competenti si trovano totalmente fuori strada nel convertire programmi per la propria macchina.

Questo articolo tratta l'Apple II, il TRS-80 modello I e modello III e il PET/CBM. In molti casi è possibile tradurre la grafica di una macchina direttamente su un'altra, proprio come si traduce un linguaggio straniero. Tuttavia, ci sono molte situazioni in cui è abbastanza avventato tentare la traduzione diretta.

A questo punto, il miglior approccio è cominciare con il trovare esattamente il significato di ogni funzione grafica del programma che state traducendo. Diamo dunque un'occhiata alle capacità grafiche del vostro computer.

APPLE

L'Apple produce grafica in tre modi: con le istruzioni PRINT standard e con due speciali modi grafici.

Ogni calcolatore può produrre grafica stampando caratteri sullo schermo. Un semplice istogramma, per esempio, si può facilmente realizzare stampando degli asterischi nella posizione appropriata dello schermo. L'Apple ha due istruzioni che aiutano molto la scrittura di programmi di questa natura e che sono molto utili nel convertire programmi del TRS-80 per l'Apple.

Il primo passo con la grafica mediante PRINT è l'azzeramento dello schermo. Battendo HOME (o eseguendo questa istruzione dall'interno di un programma in Ap-

plesoft) si ottiene l'azzeramento dello schermo. Se avete l'Integer Basic, l'istruzione corretta è CALL−936.

VTAB e HTAB

L'istruzione VTAB controlla la posizione del cursore lungo l'asse Y. Ci sono 24 righe su cui l'Apple può stampare in *text mode*. Battendo VTAB XX, dove XX è un numero tra 1 e 24, si ottiene il posizionamento del cursore in quella linea, senza cancellare alcun carattere. Per esempio, supponete di eseguire

```
FOR I=1 TO 12
PRINT "HELLO"
NEXT
```

Eseguendo le istruzioni

```
VTAB 5
PRINT "CIAO"
```

l'HELLO della quinta riga viene sostituito con CIAO.

Lo stesso principio può essere applicato con la tabulazione orizzontale. Battendo HTAB XX, dove XX è un numero da 1 a 40, si avrà il posizionamento del cursore in orizzontale nella posizione specificata.

HTAB e VTAB possono essere molto utili nel convertire altri programmi per l'Apple, specialmente se usati assieme alle altre funzioni

PEEK(37) contiene un numero da 0 a 23, il cui valore è la posizione verticale del cursore. Questo numero è *uno meno* del valore usato nell'istruzione VTAB. Se il cursore è sulla linea 10 e volete spostarlo in su di una posizione l'istruzione VTAB PEEK(37) fa esattamente questo. HTAB PEEK(36) o HTAB POS(0) fanno la stessa cosa in orizzontale, cioè spostano il cursore indietro di una posizione. Bisogna fare attenzione, tuttavia, di non eseguire HTAB per una posizione inferiore a 1 o superiore a 40, oppure VTAB per una posizione inferiore a 1 o superiore a 24.

Sebbene l'uso di normali istruzioni PRINT sia un mezzo molto primitivo di programmare la grafica, in alcuni casi può essere il metodo migliore e più diretto per convertire un programma. In situazioni in cui sono presenti grafici più complessi, tuttavia, può essere desiderabile usare uno dei metodi grafici dell'Apple.

L'Apple ha due modi grafici. Questi modi permettono di usare fino a 16 colori e alcune potenti istruzioni di *plotting* (tracciamento). L'unico inconveniente nell'usare i modi grafici dell'Apple è che il testo e la grafica non possono essere mescolati nella stessa area di schermo senza enormi sforzi di programmazione. In generale il programmatore dovrà accontentarsi di quattro linee di testo in fondo allo schermo.

Grafica a bassa risoluzione *(lo-res)*

La grafica a bassa risoluzione dell'Apple è molto conveniente per semplici programmi grafici. Si può usare una matrice di 40×40 blocchi grafici con quattro linee di testo sul fondo. In alternativa è possibile avere una matrice 40×48 senza testo. Sono disponibili sedici colori.

Battendo GR (o usandolo dall'interno di un programma) si passa alla grafica a bassa risoluzione (grafica e testo). Lo schermo viene azzerato e le istruzioni PRINT agiscono solo sulle quattro linee inferiori dello schermo.

Se desiderate un'area grafica maggiore (40×48) battete POKE−16302,0. Le quattro linee in fondo allo schermo spariscono e avrete altre otto linee di grafica.

Prima di tracciare qualunque cosa, occorre specificare un colore. Ne sono disponibili sedici. Per assegnare un colore si batte

```
COLOR=X
```

dove X è:

0	nero
1	rosso magenta
2	blu scuro
3	viola
4	verde scuro
5	grigio
6	blu medio
7	blu chiaro
8	marrone
9	arancio
10	grigio
11	rosa
12	verde
13	giallo
14	verde acqua
15	bianco

L'assegnazione di un colore non ha effetto sulla grafica già presente sullo schermo. Solo le istruzioni grafiche eseguite successivamente avranno quel colore. Quindi, eseguendo diverse assegnazioni di colore, si possono avere diversi colori sullo stesso schermo.

Ora arriviamo alla questione fondamentale. Come si traccia un punto? Fondamentalmente, lo schermo Apple funziona come un sistema matematico di coordinate. L'asse X può essere immaginato in cima allo schermo, numerato con le coordinate da 0 a 39. L'asse Y corre parallelo al lato sinistro dello schermo con lo 0 in cima e 39 o 47

```
10 GR                    Grafica a bassa risoluzione
20 COLOR=3               Il colore è viola
30 HLIN 0,39 AT 0        Traccia una linea in cima allo schermo
40 VLIN 0,39 AT 39       Traccia una linea alla destra dello schermo
50 HLIN 0,39 AT 39       Traccia una linea in fondo allo schermo
60 VLIN 0,39 AT 0        Traccia una linea a sinistra dello schermo
```

Fig. 1. *Una cornice in bassa risoluzione per l'Apple.*

in fondo, secondo che avete scelto di usare o no le quattro linee di testo. Dunque il punto 0,0 è in alto a sinistra dello schermo e il punto 39,39 è in basso a destra dello schermo (nel modo grafica-testo).

L'istruzione PLOT traccia un punto specificato. Il suo formato è PLOT X,Y. Quindi PLOT 20,20 traccia un quadrato a distanza 20 dalla sinistra dello schermo e 20 dalla parte superiore. Per cancellare questo punto, specificate il colore 0 (nero) o comunque quello di fondo e ritracciate il punto.

È anche possibile tracciare una linea tra due punti dello schermo. Il comando HLIN X,Y AT Z traccia una linea orizzontale tra le coordinate orizzontali X e Y alla coordinata verticale Z. Quindi l'istruzione HLIN 1,20 AT 10 collega i punti 1,10 e 20,10. VLIN X,Y AT Z fa la stessa cosa per una linea verticale. Quindi il comando VLIN 1,20 AT 10 collega i punti 10,1 e 10,20.

Per un esempio di grafica a bassa risoluzione, vedi il programma di figura 1 che traccia una cornice attorno allo schermo. Infine, l'utente deve conoscere come uscire da questo modo grafico. Semplicemente battendo TEXT: lo schermo si riporta alle solite 24 righe di testo e 40 caratteri per linea.

Grafica ad alta risoluzione *(hi-res)*

La grafica ad alta risoluzione dell'Apple offre alcune fra le migliori capacità grafiche presenti sui microcomputer. Sebbene si possano usare solo otto colori, si può ottenere una risoluzione di 280×192 pixel, permettendo di programmare figure molto dettagliate e una grafica molto espressiva.

Per passare al modo grafico in alta risoluzione si batte HGR. Ciò vi mette a disposizione un reticolo 280×160 con quattro righe di testo in fondo allo schermo. Battendo HGR2 invece di HGR, oppure battendo POKE−16302,0 dopo aver battuto HGR ci si pone in modo grafico a schermo pieno, con una risoluzione di 280×192.

I colori ad alta risoluzione si determinano in maniera simile alla basa risoluzione. HCOLOR= è l'equivalente dell'istruzione COLOR= in bassa risoluzione. Gli otto colori disponibili in alta risoluzione sono:

0	nero
1	verde
2	blu
3	bianco 1
4	nero
5	dipende dalla TV
6	dipende dalla TV
7	bianco 2

Le coordinate in alta risoluzione sono numerate da 0 a 279 lungo l'asse delle X (in cima allo schermo) e da 0 a 159 (HGR) o da 0 a 191 (HGR2) lungo l'asse delle Y.

L'istruzione equivalente a PLOT in alta risoluzione è HPLOT. HPLOT X,Y traccia un punto nella posizione X,Y.

Nella grafica ad alta risoluzione, è possibile tracciare una linea da una posizione qualsiasi ad un'altra qualsiasi, anche in maniera diagonale. L'istruzione HPLOT X,Y TO X,Y o HPLOT X,Y TO X,Y TO X,Y collega i punti indicati tra i TO. Si tratta di una istruzione molto potente, non disponibile in bassa risoluzione.

Per un esempio di grafica in alta risoluzione sull'Apple, si veda il programma in figura 2 che traccia una cornice attorno allo schermo, come nell'esempio precedente.

Anche in questo caso l'istruzione TEXT riporta il computer al normale *text mode*.

```
10 HGR                   Grafica ad alta risoluzione
20 COLOR=1               Il colore è verde
30 HPLOT 0,0 TO 0,159 TO 279,   Collega i quattro angoli dello schermo
         159 TO 279,0 TO 0,0
```

Fig. 2. *Una cornice in alta risoluzione per l'Apple.*

Conversioni grafiche

TRS-80

La grafica del TRS-80 è molto più semplice di quella dell'Apple, sebbene non sia così versatile. Non sono previsti speciali modi grafici. Il testo può essere stampato in una determinata posizione dello schermo (come con le istruzioni VTAB e HTAB dell'Apple) e si può usare la grafica sullo stesso schermo. La risoluzione del TRS-80 può essere paragonata alla bassa risoluzione dell'Apple.

I modelli I e III sono quasi identici: il 95% delle istruzioni TRS-80 possono essere usate sulle due macchine. Per questo motivo parlerò semplicemente di TRS-80 riferendomi ai due modelli. Quando una particolare caratteristica è disponibile solo su un modello, lo specificherò.

Istruzioni PRINT

Il TRS-80, come l'Apple, può produrre grafica mediante le istruzioni PRINT. Tuttavia, il TRS-80 ha una speciale istruzione PRINT@, che rende possibile il riferimento ad ogni posizione dello schermo mediante un semplice numero. Può essere una istruzione molto potente se usata efficacemente.

Lo schermo del TRS-80 è composto di 16 righe ognuna di 64 caratteri, per un totale di 1024 possibili posizioni. Queste posizioni sono numerate da 0 a 1023, con lo 0 in alto a sinistra, 63 alla fine della prima riga, 64 all'inizio della seconda riga, e così via fino a 1023 in basso a destra. La sintassi di questa istruzione è PRINT@ XXXX,... dove XXXX è un numero da 0 a 1023 e... è qualunque espressione valida in una normale istruzione PRINT.

Torniamo agli esempi visti per l'Apple. Prima battete

```
FOR I=1 TO 12
PRINT "HELLO"
NEXT
```

Per sostituire il quinto HELLO con CIAO sull'Apple scrivevamo

```
VTAB 5
PRINT "CIAO".
```

Sul TRS-80 invece il miglior modo per fare la stessa cosa è di identificare il valore numerico della prima locazione della quinta riga dello schermo. A tal scopo possiamo usare la formula (X−1)*64, e scriviamo

```
PRINT@ (5−1)*64, "CIAO"
```

Notate, che il cursore è stato spostato sulla quinta riga dello schermo e quindi la parola READY viene ora scritta su un HELLO. Se non volete che ciò succeda, potete aggiungere

```
PRINT@ (13−1)*64," ";
```

che riporta il cursore sulla tredicesima riga.

CHR$(X) valore di X	Azione
24	Sposta il cursore di un posto a sinistra
25	Sposta il cursore di un posto a destra
26	Sposta il cursore di una riga in basso
27	Sposta il cursore di un riga in alto
28	Sposta il cursore nell'angolo in alto a sinistra

Tav. 1. *Spostamento del cursore con CHR$ sul TRS-80*

Molto spesso, nel convertire la grafica, è conveniente spostare il cursore in su o in giù di un posto senza usare l'istruzione PRINT@. Per esempio quando non si conosce la posizione attuale del cursore, oppure nel convertire un programma PET che usa un metodo speciale per posizionare il cursore. Si può farlo usando la funzione CHR$. Scrivendo PRINT CHR$(X) si ottiene l'azione indicata nella tavola 1.

Caratteri grafici

Il TRS-80 può creare grafica anche scrivendo speciali caratteri grafici. Questi caratteri (vedi figura 3) consistono di tutte le 64 possibili permutazioni on/off di una matrice 2×3 ($2^{2\cdot3}=2^6=64$). Questi caratteri grafici possono essere stampati usando la funzione CHR$. Battendo CHR$(X), dove X è il codice numerico del carattere speciale richiesto, si ottiene la stampa di quel carattere. Questa funzione può essere usata anche assieme alla funzione PRINT@. Inoltre, l'istruzione PRINT STRING$(X,Y) scrive una stringa di X caratteri grafici Y. Quindi l'istruzione PRINT@ 0,STRING$(64,191) stampa una linea orizzontale in cima allo schermo.

Il TRS-80 modello III ha ulteriori 96 caratteri speciali. Sessantaquattro di questi possono essere stampati esattamente come i sessantaquattro visti sopra. Si tratta dei codici 192-255 (vedi figura 4). Tuttavia, c'è una piccola istruzione che deve essere eseguita prima di stampare questi caratteri. Quando si accende il modello III, questi 64 codici rappresentano caratteri di "compressione spazio". PRINT CHR$(192) non stampa alcun spazio, PRINT CHR$(193) stampa uno spazio, fino a PRINT CHR$(255) che stampa 63 spazi. Per sostituire questi caratteri di compressione dello spazio con gli speciali caratteri grafici, battete PRINT CHR$(21). Questa istruzione funziona come un commutatore tra i caratteri di compressione spazio e i caratteri grafici speciali.

Il TRS-80 modello III ha uno speciale insieme di 96 caratteri grafici e un ulteriore insieme di caratteri giapponesi.

Oltre ai 64 caratteri grafici speciali disponibili sul modello III, esiste uno speciale insieme di caratteri giapponesi. I loro codici vanno da 192 a 255 come i caratteri grafici speciali, ma vengono selezionati eseguendo PRINT CHR$(22) dopo aver selezionato l'insieme dei caratteri speciali con l'istruzione PRINT CHR$(21).

DEC	ESA	CODICE Z80	GRAFICA	BASIC TRS-80
128	80	ADD A,B		END
129	81	ADD A,C		FOR
130	82	ADD A,D	80 81 82 83	RESET
131	83	ADD A,E		SET
132	84	ADD A,H		CLS
133	85	ADD A,L	84 85 86 87	CMD
134	86	ADD A,(HL)		RANDOM
135	87	ADD A,A		NEXT
136	88	ADC A,B		DATA
137	89	ADC A,C		INPUT
138	8A	ADC A,D	88 89 8A 8B	DIM
139	8B	ADC A,E		READ
140	8C	ADC A,H		LET
141	8D	ADC A,L	8C 8D 8E 8F	GOTO
142	8E	ADC A,(HL)		RUN
143	8F	ADC A,A		IF
144	90	SUB B		RESTORE
145	91	SUB C	90 91 92 93	GOSUB
146	92	SUB D		RETURN
147	93	SUB E		REM
148	94	SUB H		STOP
149	95	SUB L	94 95 96 97	ELSE
150	96	SUB (HL)		TRON
151	97	SUB A		TROFF
152	98	SBC A,B		DEFSTR
153	99	SBC A,C	98 99 9A 9B	DEFINT
154	9A	SBC A,D		DEFSNG
155	9B	SBC A,E		DEFDBL
156	9C	SBC A,H		LINE
157	9D	SBC A,L	9C 9D 9E 9F	EDIT
158	9E	SBC A,(HL)		ERROR
159	9F	SBC A,A		RESUME
160	A0	AND B		OUT
161	A1	AND C	A0 A1 A2 A3	ON
162	A2	AND D		OPEN
163	A3	AND E		FIELD
164	A4	AND H		GET
165	A5	AND L	A4 A5 A6 A7	PUT
166	A6	AND (HL)		CLOSE
167	A7	AND A		LOAD
168	A8	XOR B		MERGE
169	A9	XOR C	A8 A9 AA AB	NAME
170	AA	XOR D		KILL
171	AB	XOR E		LSET
172	AC	XOR H		RSET
173	AD	XOR L		SAVE
174	AE	XOR (HL)	AC AD AE AF	SYSTEM
175	AF	XOR A		LPRINT
176	B0	OR B		DEF
177	B1	OR C		POKE
178	B2	OR D	B0 B1 B2 B3	PRINT
179	B3	OR E		CONT
180	B4	OR H		LIST
181	B5	OR L		LLIST
182	B6	OR (HL)	B4 B5 B6 B7	DELETE
183	B7	OR A		AUTO
184	B8	CP B		CLEAR
185	B9	CP C		CLOAD
186	BA	CP D		CSAVE
187	BB	CP E	B8 B9 BA BB	NEW
188	BC	CP H		TAB(
189	BD	CP L		TO
190	BE	CP (HL)		FN
191	BF	CP A	BC BD BE BF	USING

Fig. 3. *Caratteri grafici del TRS-80 (codici 128-191)*

Conversioni grafiche

Se siete sorpresi dal numero di caratteri disponibili sul modello III, c'è ancora un'altra sorpresa per voi. Esiste un altro insieme di caratteri sul modello III. Sono i codici 0-31 (vedi figura 4). Tuttavia,

Fig. 4. *Caratteri grafici speciali del TRS-80 modello III (codici 0-31, 192-255)*

```
10 CLS                          Azzera lo schermo
20 FOR X=0 TO 127: SET (X,47):  Traccia una cornice attorno allo schermo
   SET (X,0): NEXT:
   FOR X=0 TO 47: SET(0,X):
   SET(127,X): NEXT
30 PRINT  512, "BATTI ENTER     Messaggio al centro dello schermo
   PER VEDERE I
   CARATTERI SPECIALI"
40 INPUT"";X$                   Attende che venga battuto un tasto
50 CLS                          Azzera lo schermo
60 PRINT CHR$(21)               Seleziona i caratteri grafici
70 FOR I=192 TO 255:            Stampa i caratteri
   PRINT CHR$(I):;"";:
   NEXT
80 INPUT"BATTI ENTER PER        Attende
   VEDERE I CARATTERI
   GIAPPONESI";X$
90 PRINT CHR$(22)               Seleziona i caratteri giapponesi
100 INPUT"BATTI ENTER           Attende
   PER FINIRE";X$
110 PRINT CHR$(22); CHR$(21);:  Seleziona i caratteri standard e azzera
   CLS: END                     lo schermo
```

Fig. 5. *Programma dimostrativo della grafica del TRS-80*

essi sono accessibili solo con istruzioni POKE. Per poter stampare un carattere grafico da 0 a 31, il valore di quel carattere deve essere posto in appropriate locazioni di memoria che la Radio Shack chiama VIDRAM. Questi indirizzi di video partono da 15360 e arrivano a 16383 e sono equivalenti ad una PRINT@indirizzo+15360. Quindi, per scrivere il carattere speciale 10 all'inizio dello schermo, dovete battere POKE 15360,10.

Non abbiamo finito con le capacità grafiche del TRS-80. Ambedue i modelli possono anche tracciare punti sullo schermo. Questi punti possono apparire sullo schermo assieme alle altre caratteristiche grafiche del TRS-80, e anche assieme a testo.

Lo schermo del TRS-80 è diviso in una matrice 128×48, ogni blocco della quale può semplicemente essere posto on o off. I due modelli non prevedono il colore.

L'istruzione SET (X,Y) accende il blocco grafico nella posizione X (orizzontale) e Y (verticale). Il valore di X può essere tra 0 e 127, mentre il valore di Y può essere tra 0 e 47. Una importante differenza tra accendere un blocco grafico e stampare un carattere grafico è che un blocco grafico non provoca lo *scrolling* dello schermo.

L'istruzione RESET, come detto prima, spegne il blocco grafico specificato. La sintassi dell'istruzione è RESET (X,Y) ed ha esattamente gli stessi parametri dell'istruzione SET. Vedi la figura 5 per un esempio che dimostra alcune caratteristiche di base della grafica del TRS-80.

PET

La grafica del PET è molto differente da quella del TRS-80. Sul PET non ci sono modi grafici speciali, né ci si può riferire ad un punto specifico sullo schermo per mezzo di coordinate. Essenzialmente la grafica del PET consiste di istruzioni PRINT standard combinate con caratteri speciali di movimento del cursore. (I caratteri grafici che possono essere stampati si ottengono premendo il tasto SHIFT e un altro tasto. I tasti di movimento del cursore sono specificamente indicati, e talvolta vanno premuti assieme a SHIFT). (Vedi figura 6).

Il PET ha sei caratteri di movimento del cursore. Questi caratteri sono trattati come ogni altro carattere sulla tastiera in quanto possono essere assegnati ad una stringa e stampati. Quando sono stampati,

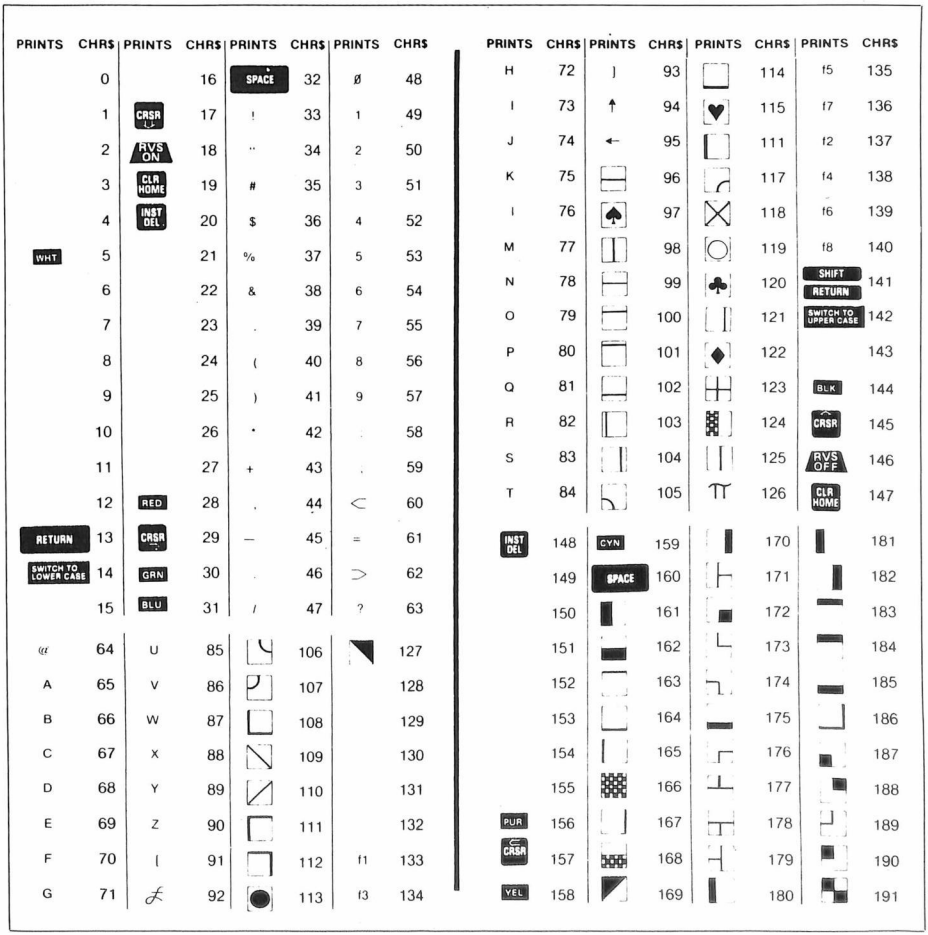


Fig. 6. *Caratteri standard del PET*

appaiono come simboli speciali, diversi da ogni carattere su ogni altro computer.

Il tasto HOME riporta il cursore nell'angolo in alto a sinistra dello schermo. Viene stampato come una S in campo inverso.

Il tasto HOME assieme a SHIFT fa la stessa cosa ma in più azzera il video. Appare sullo schermo come un cuore in campo inverso.

Il tasto CRSR giù/su sposta il cursore in giù di una linea. Appare come una Q in campo inverso. Lo stesso tasto con SHIFT muove il cursore in su. Appare come un cerchio vuoto con un bordo nero.

Il tasto CRSR destra/sinistra sposta il cursore di una posizione a destra. Appare sullo schermo come una parentesi quadra in campo inverso.

Lo stesso tasto con SHIFT sposta il cursore di una posizione a sinistra. Appare sullo schermo come un rettangolo nero con una linea bianca verticale.

Questi sei caratteri di controllo del cursore possono essere trattati esattamente come ogni altro carattere del PET. Per esempio l'istruzione

```
PRINT"(SHIFTHOME)(CRSR giù)
 (CRSR giù) (CRSR giù)HELLO"
```

azzera lo schermo e posiziona la parola HELLO sulla quarta linea.

Il PET ha anche un insieme alternativo di caratteri, che si può selezionare battendo POKE 59468,14. La tastiera in questo modo funziona come una comune tastiera con le maiuscole e le minuscole. Per tornare al set standard di caratteri si esegue la POKE 59468,12.

Il programma in figura 7, sebbene molto semplice, illustra il metodo di base per inserire la grafica in un programma per il PET. Il programma azzera lo schermo, sposta il cursore alla quarta riga e traccia un quadrato.

Conversione dal TRS-80 all'Apple

Nel convertire un programma dal TRS-80 all'Apple, potete usare il modo TEXT, la grafica a bassa risoluzione, o la grafica ad alta risoluzione.

Il modo TEXT deve essere usato quando il programma originale comprende istruzioni PRINT@ o semplicemente istruzioni PRINT, e il testo o la grafica sullo schermo possono essere condensati su 40 colonne. A parte lo schermo più piccolo, l'unico inconveniente nell'usare l'Apple sarà che i caratteri grafici non possono essere usati nel modo TEXT.

L'istruzione che più produce confusione è probabilmente la PRINT@. Tuttavia, si tratta dell'istruzione più facile da convertire. L'istruzione TRS-80 PRINT@X, "QUESTA È UNA PROVA" può essere convertita in tre istruzioni Apple

```
VTAB INT(X/64)+1
HTAB X+1-INT(X/64)*64
PRINT"QUESTA È UNA
      PROVA"
```

Nell'usare questa procedura di conversione, tuttavia, il programmatore deve fare attenzione a che HTAB non superi la colonna 40. Lo schermo del TRS-80 è largo 64 colonne, contrariamente allo schermo dell'Apple che ne ha 40. Se sono necessarie solo 40 colonne, questa procedura è probabilmente la più semplice da usare. Può essere consigliabile, tuttavia, tracciare su carta i risultati delle istruzioni PRINT@ per ottenere risultati più piacevoli.

La conversione più semplice tra TRS-80 e Apple (in modo TEXT) è l'azzeramento dello schermo. Basta sostituire ogni CLS con un HOME.

```
10 PRINT"(SHIFT HOME)(CRSR giù)                Azzera lo schermo e sposta il cursore in giù di quattro righe
   (CRSR giù)(CRSR giù)";
20 PRINT "-----"                               Traccia il lato superiore del quadrato
30 FOR I=1 TO 7: PRINT"-I      I":NEXT        Traccia i lati del quadrato
40 PRINT"-----"                                Traccia il lato inferiore del quadrato
```

Fig. 7. *Programma dimostrativo per la grafica del PET*

La grafica del TRS-80 può essere simulata in alta risoluzione o in bassa risoluzione. Questi metodi forniscono i risultati grafici più piacevoli. Tuttavia se occorre tracciare sullo stesso schermo testo e grafica, deve essere usato il modo TEXT. In questo caso, dovete seguire le istruzioni della grafica a bassa risoluzione sostituendo però HOME a GR (per azzerare lo schermo ma non entrare nel modo grafico).

Su un Apple, la grafica del TRS-80 può essere simulata in alta o in bassa risoluzione.

Le istruzioni PLOT, se usate all'interno del modo TEXT, non tracciano blocchi grafici alle coordinate appropriate, ma pongono invece sullo schermo caratteri standard di testo. I caratteri che vengono stampati possono essere predeterminati, ma ciò va al di là dello scopo di questo articolo.

L'istruzione del TRS-80 CHR$(31) azzera lo schermo dalla posizione del cursore in avanti. Sull'Apple si può simulare con l'istruzione CALL−958.

I caratteri grafici speciali del TRS-80 (compreso l'insieme di caratteri alternativi del modello III) non possono essere facilmente riprodotti sull'Apple. Se avete un programma con testi e caratteri grafici speciali, le sole possibilità sono sostituire i caratteri con quelli standard dell'Apple o usare la grafica ad alta risoluzione e creare un generatore di caratteri, il che, per un programmatore inesperto, è un'impresa molto difficile.

Se nel programma TRS-80 sono usati solo caratteri grafici (istruzione SET) si può usare la grafica a bassa risoluzione dell'Apple. Ma, ricordate, la bassa risoluzione permette al massimo una matrice 40×48 (senza testo) mentre il TRS-80 ha una matrice grafica di 128×48. Tuttavia, se è possibile programmare un particolare applicazione entro questi vincoli, la grafica a bassa risoluzione è preferibile. È più semplice da usare di quella ad alta risoluzione e permette un numero doppio di colori.

Lo schermo del TRS-80 deve però essere "condensato" a 40×48 o 40×40. Una volta fatto questo, la procedura di conversione è molto semplice.

Nel programma del TRS-80, guardate dove inizia la parte grafica. Generalmente a questo punto c'è una istruzione CLS. Sostituitela con GR per azzerare lo schermo e passare alla bassa risoluzione.

Le successive istruzioni PRINT del programma dovranno essere ristrette a quattro linee di testo. Tali linee dovranno essere contigue in fondo allo schermo. Non c'è bisogno di nessuna particolare conversione delle istruzioni PRINT, a meno che non ci sia PRINT@. In questo caso, ricordate che si possono usare solo le righe 21-24 per i testi in grafica a bassa risoluzione.

Se desiderate sostituire le quattro righe di testo con altre otto linee di grafica, eseguite l'istruzione POKE−16302,0. Avrete disponibile una matrice 40×48.

Prima di tracciare i punti bisogna scegliere un colore. (Il colore può essere cambiato in ogni momento nel programma senza cambiare la grafica già tracciata). Si fa ciò con l'istruzione COLOR= (vedi sopra).

Tutte le istruzioni SET devono essere sostituite con istruzioni PLOT. Essenzialmente, SET (X,Y) diventa PLOT X,Y. Non tutti i valori validi per X e Y in una SET sono validi in una PLOT. X in una PLOT non può superare 39, ed Y non può superare 39 o 47 secondo che lo schermo sia tutto riservato alla grafica o ci sia anche testo.

Il passo finale è tradurre l'istruzione RESET del TRS-80. Si fa esattamente come con una SET, ma il colore deve essere scelto uguale al fondo (generalmente nero). Quindi l'esecuzione di una PLOT trasforma il blocco grafico nel suo stato originale.

Grafica ad alta risoluzione

La conversione della grafica del TRS-80 in grafica ad alta risoluzione è più complicata, ma i risultati valgono lo sforzo. L'intero reticolo 128×48 può essere trasferito nello schermo dell'Apple, assieme a tutti i 64 caratteri grafici (codici ASCII da 128 a 191). Anche l'insieme alternativo di caratteri del modello III può essere simulato, sebbene ciò richieda in qualche caso sostanziali sforzi programmatici.

Prima di iniziare a discutere la procedura di conversione, diamo

Conversioni grafiche

un'occhiata più da vicino alle capacità grafiche del TRS-80. Abbiamo detto che lo schermo è 128×48. Ma è realmente così? In effetti, ogni blocco grafico è, per se stesso, una matrice alta tre blocchi e larga due. Questo significa che lo schermo del TRS-80 può essere rappresentato come uno schermo di (128×2)×(48×3) cioè 256×144 blocchi. L'Apple, nella grafica ad alta risoluzione assieme a testo, ha 280×160 blocchi, così che di fatto l'intero schermo del TRS-80 può essere rappresentato nell'Apple.

Se avete seguito fin qui, avrete probabilmente notato che c'è un piccolo problema in questa procedura di conversione. Lo schermo del TRS-80, ricorderete, è composto di 6144 blocchi. La porzione dello schermo dell'Apple che useremo, invece, ne contiene 256×144 cioè 36864. Ciò significa che dovremo tracciare 36864/6144 cioè 6 punti sull'Apple per ogni punto del TRS-80. Ecco come si fa.

Dapprima scegliete il modo grafico appropriato per la vostra applicazione (HGR o HGR2). Generalmente è sufficiente HGR, perché anche con le linee di testo in fondo allo schermo c'è posto sufficiente per sistemare l'intero schermo del TRS-80.

Il prossimo passo è scegliere un colore con l'istruzione HCOLOR=. Si può fare molto semplicemente scegliendo un valore dall'elenco presentato in questo articolo.

Ogni volta che incontrate una istruzione SET(X,Y), deve essere convertita nelle equivalenti istruzioni HPLOT. Le coordinate X e Y dell'istruzione SET possono essere l'Apple. Le coordinate X*2, Y*3 corrispondono al punto in alto a sinistra del reticolo Apple 2×3 di quel punto. Vedi nella tavola 2 la lista dei sei punti sull'Apple che compongono quel punto.

Valore X Apple	Valore Y Apple	Posizione
X*2	Y*3	in alto a sinistra
X*2+1	Y*3	in alto a destra
X*2	Y*3+1	in centro a sinistra
X*2+1	Y*3+1	in centro a destra
X*2	Y*3+2	in basso a sinistra
X*2+1	Y*3+2	in basso a destra

Tav. 2. *Punti sull'Apple che compongono un blocco grafico TRS-80*

Se dovete riempire l'intero blocco, eseguite le istruzioni

```
HPLOT X*2,Y*3 TO X*2,Y*3+2
HPLOT X*2+1,Y*3 TO
X*2+1,Y*3+2
```

È una buona idea scrivere queste istruzioni in una subroutine. Quindi, ogni volta che appare un'istruzione tipo SET(X,Y) nella vostra lista, potete semplicemente sostituirla con le istruzioni

```
X=2: Y=3: GOSUB 10000
```

(se avete scritto la subroutine precedente dalla linea 10000 e avete aggiunto RETURN).

Tracciare uno dei 64 caratteri grafici del TRS-80 è molto semplice. Prima consultare l'elenco per vedere quali dei sei blocchi grafici deve essere tracciato.

Quindi applicate le formule dell'elenco e HPLOT le coordinate appropriate.

Per esempio, supponiamo di voler stampare il carattere 179. Esaminando l'elenco, possiamo vedere che è composto dai seguenti punti della matrice 2×3: in alto a sinistra, in alto a destra, in basso a sinistra e in basso a destra.

Se desideriamo tracciarlo alle coordinate TRS-80 (50,100), dovremo eseguire le seguenti istruzioni:

```
HPLOT 50*2+1,100*3+2
HPLOT 50*2, 100*3+2
HPLOT 50*2+1,100*3
HPLOT 50*2,100*3
```

Conversione dal PET all'Apple

Convertire la grafica del PET per l'Apple può essere eccessivamente frustrante se sono usati gli speciali caratteri grafici del PET.

Produrre questi carattere sull'Apple è spesso confrontabile con il produrre i caratteri speciali del modello III. In molti casi è consigliabile riscrivere l'intero algoritmo di programma in modo che sia più adatto all'Apple.

Se i caratteri grafici usati sul PET sono tali che vi è un carattere simile nell'Apple, la conversione è molto semplice. Lo schermo del PET è composto di 25 righe da 40 caratteri ognuna, e lo schermo dell'Apple contiene 24 righe da 40 caratteri ognuna.

Carattere controllo cursore PET	Istruzione locazione cursore Apple
(HOME)	VTAB 1: HTAB 1
(SHIFT HOME)	HOME
(CRSR giù)	VTAB PEEK(37)+2
(CRSR su)	VTAB PEEK(37)
(CRSR destra)	HTAB PEEK(36)+2
(CRSR sinistra)	HTAB PEEK(36)

Tav. 3. *Corrispondenza tra PET e Apple per il posizionamento del cursore*

Il problema principale nella conversione tra PET e Apple è sostituire con le appropriate istruzioni VTAB e HTAB i caratteri di movimento del cursore del PET. Generalmente lo si può fare direttamente usando la tavola 3.

Ora, supponiamo di avere un programma per un PET che azzera lo schermo e traccia una linea sulla quinta riga dello schermo. Il programma dovrebbe avere una istruzione

```
PRINT"(CLEAR)(CRSR giù)
      (CRSR giù) (CRSR giù)..."
```

La traduzione per l'Apple sarebbe

```
HOME
VTAB PEEK(37)+2
VTAB PEEK(37)+2
VTAB PEEK(37)+2
VTAB PEEK(37)+2
PRINT"..."
```

Si noti che i caratteri di posizionamento del cursore nel PET fanno parte dell'insieme dei caratteri e quindi vengono stampati come elementi di una stringa. L'Apple invece, ha *istruzioni* di movimento del cursore che non possono essere usate all'interno di istruzioni PRINT.

Sia il PET che l'Apple hanno la possibilità di scrivere in modo "inverso". Sul PET ci sono due caratteri che, ancora, devono essere usati all'interno di un PRINT. Sull'Apple ci sono due istruzioni separate per controllare questa funzione. Si veda la tavola 4.

I metodi usati dal PET e dall'Apple per "invertire" il video sono molto simili. Eseguendo l'opportuna istruzione tutto l'output successivo viene visualizzato in modo inverso. C'è però una piccola differenza. L'istruzione INVERSE dell'Apple può essere cancellata *solo* da una istruzione NORMAL. Sul PET, il modo inverso può essere cancellato da un (OFF) (SHIFT e RVS) o da un RETURN.

Supponiamo che il programma PET che state traducendo abbia l'istruzione

```
PRINT"(RVS)QUESTO È IN
      CAMPO INVERSO
      (SHIFT RVS) MENTRE
      QUESTO NO"
```

L'istruzione Apple equivalente è

```
INVERSE PRINT"QUESTO È
    IN CAMPO INVERSO";
NORMAL PRINT"MENTRE
    QUESTO NO"
```

Carattere PET	Istruzioni Apple	Funzione
RVS	INVERSE	Tutti i successivi caratteri vengono stampati in campo inverso
OFF	NORMAL	Annulla ogni istruzione di inversione dei caratteri

Tav. 4. *Corrispondenza tra PET e Apple per la scrittura in campo inverso*

Conversione dall'Apple al TRS-80

Quando possibile, la conversione da Apple a TRS-80 è molto semplice. La maggior parte dell'output scritto può essere trasferito nel TRS-80, ma poiché lo schermo del TRS-80 ha solo sedici righe, mentre quello dell'Apple ne ha ventiquattro, in alcuni casi lo schermo deve essere compresso o modificato in qualche altro modo.

Solo due dei tre "modi" possono essere simulati sul TRS-80. L'output ottenuto con PRINT (VTAB, HTAB, ecc.) può essere convertito facilmente, così come la grafica in bassa risoluzione. Il TRS-80 non ha tuttavia, la capacità di riprodurre la grafica ad alta risoluzione dell'Apple.

Se si deve convertire grafica ad alta risoluzione per il TRS-80 si deve usare la grafica standard del TRS-80, perdendo una parte considerevole di risoluzione.

Conversioni grafiche

L'unico potenziale problema nel convertire un programma Apple per il TRS-80 è nel localizzare l'istruzione PRINT@ equivalente per un dato insieme di locazioni HTAB e VTAB.

Per ogni coppia di valori X e Y, l'istruzione Apple VTAB X: HTAB Y è equivalente all'espressione TRS-80

PRINT@(X*64)+Y−65

Usando questo metodo, tuttavia, vi è il problema che la formula non può essere usata in una situazione in cui X è maggiore di 16. L'unica soluzione a questo problema sta nel ridisegnare lo schermo in modo che vengano usate solo 16 righe di testo.

Un altro elemento nel convertire un programma Apple al TRS-80 è l'azzeramento dello schermo. Semplicemente, si sostituisce ogni HOME del programma Apple con un CLS nel programma TRS-80.

Il TRS-80 non ha la capacità di riprodurre la grafica ad alta risoluzione dell'Apple.

La grafica a bassa risoluzione dell'Apple può essere duplicata molto facilmente sul TRS-80. Poiché la grafica a bassa risoluzione contiene al massimo una matrice 40×48 e lo schermo del TRS-80 ha una matrice 128×48, questa particolare conversione è immediata.

Quando incontrate una istruzione GR, sostituitela con CLS.

Quindi, semplicemente sostituire ogni PLOT X,Y con SET(X,Y). I valori di X e Y non cambiano in questo caso.

Il colore non può essere riprodotto su TRS-80, e dunque l'istruzione COLOR= deve essere ignorata, eccetto quando il colore è messo a 0 o comunque uguale al colore di fondo. In questo caso, le successive istruzioni PLOT devono essere sostituite con istruzioni RESET per cancellare i blocchi grafici alle coordinate opportune.

Caratteri controllo cursore PET	Codice ASCII TRS-80
(HOME)	28
(CRSR giù)	26
(CRSR su)	27
(CRSR destra)	25
(CRSR sinistra)	24

Tav. 5. *Corrispondenza tra PET e TRS-80 per il posizionamento del cursore*

Conversione dal PET al TRS-80

Convertire un programma PET affinché giri su un TRS-80 è simile a convertirlo per girare sull'Apple. Lo schermo di 25 righe del PET viene però ridotto a 16 righe, non a 24. Inoltre, molti dei caratteri speciali PET non hanno un parallelo sul TRS-80. Se non si trova un sostituto adatto, la sola soluzione è tracciare i singoli punti del programma PET ed escogitare un algoritmo per usare le istruzioni SET del TRS-80 o per stampare con PRINT gli speciali caratteri grafici.

Convertendo da PET a TRS-80, probabilmente la maggiore confusione è data dai caratteri di controllo del cursore del PET. Tuttavia, questi caratteri di controllo, hanno equivalenti *diretti* sul TRS-80, come indica la tavola 5.

I codici del TRS-80 vanno usati con la funzione CHR$ e scritti con PRINT. (Assicuratevi di aver messo un punto e virgola dopo la PRINT). Quindi, l'equivalente dell'istruzione PET

```
PRINT"(HOME)(CRSR giù)
      (CRSR destra)TEST"
```

sarà

```
PRINT CHR$(28);CHR$(26);
      CHR$(25); "TEST"
```

Un carattere PET non ha un codice ASCII sul TRS-80: l'azzeramento dello schermo (sul PET è (SHIFT HOME)) deve essere sostituito con CLS sul TRS-80.

Conversione dall'Apple al PET

Convertire un programma dall'Apple al PET è, in molti casi, quasi impossibile. Le capacità grafiche del PET operano in maniera molto differente da quelle di molti altri computer. Il maggior problema in questa conversione è che con il PET non vi è la possibilità di accedere ad una particolare posizione dello schermo direttamente con un insieme di coordinate come le istruzioni Apple VTAB e HTAB o

l'istruzione TRS-80 PRINT@. Il miglior consiglio che si possa dare a coloro che possiedono un PET è di riscrivere la routine grafiche dei loro programmi in modo da riprodurre la grafica in modo efficace.

C'è un modo di creare una subroutine per simulare le istruzioni VTAB e HTAB, ma generalmente non è consigliabile a meno che non si stia usando una grafica relativamente semplice. Può essere usato solo per la grafica con PRINT, quella a bassa e ad alta risoluzione non può essere tradotta direttamente.

Essenzialmente, dobbiamo prima portare il cursore in alto a sinistra (HOME) quindi stampare tanti (CRSR giù) quanto è l'argomento di VTAB meno 1 e tanti (CRSR destra) quanto è l'argomento di HTAB meno 1. Quindi, l'istruzione

```
VTAB 4
HTAB 4
 PRINT"QUESTO È UN TEST"
```

può essere convertita in

```
PRINT"(HOME)(CRSR giù)
      (CRSR giù)(CRSR giù)
      (CRSR destra)(CRSR
      destra)(CRSR destra)
      QUESTO È UN TEST"
```

Con un ciclo FOR-NEXT, questo si può fare in una subroutine. Si tratta di un modo molto primitivo di convertire una istruzione VTAB, ma è una possibilità per programmi grafici relativamente semplici.

L'altra necessaria conversione tra Apple e PET è l'azzeramento dello schermo. Sul PET l'istruzione equivalente a HOME è PRINT"(SHIFT HOME)".

Conversione dal TRS-80 al PET

Convertire un programma da TRS-80 è sostanzialmente la stessa cosa di convertire un programma da Apple a PET. È molto difficile da fare, e i risultati non sono così efficienti se si usa lo stesso algoritmo.

L'istruzione di azzeramento dello schermo su TRS-80 è CLS. Deve essere sostituita con un PRINT"(SHIFT HOME)".

Se è assolutamente necessario convertire un programma TRS-80 per il PET direttamente, lo si può fare. Tuttavia, come per l'Apple, solo il testo può essere trattato direttamente. Questo significa che un programma grafico che usa istruzioni SET non può generalmente venir convertito.

Le istruzioni PRINT@ sono solitamente il mezzo principale di produrre grafica sul TRS-80 senza istruzioni SET. L'indirizzo di PRINT@ deve dapprima essere convertito nelle equivalenti coordinate orizzontale e verticale. Per ogni indirizzo A di PRINT@A, la corrispondente coordinata Y è INT(A/64)+1 e la coordinata X è A+1−INT(A/64)*64.

Una volta calcolate queste coordinate per spostare il cursore, può essere eseguita la procedura sopra descritta per la conversione da Apple a PET.

Le tecniche di conversione grafica qui descritte non esauriscono tutti i possibili metodi di conversione, né è stata descritta ogni istruzione grafica di ogni computer di cui si è parlato. Quello che si è tentato di fare è familiarizzare il lettore con i principi grafici basilari di ogni computer e fornire qualche indicazione su come affrontare il processo di conversione.

La conversione grafica è senza dubbio un obiettivo complesso; una volta compreso come un computer funzioni, la creatività del programmatore influenzerà la sua tecnica di conversione più di ogni altra cosa. ■

PASCAL

IMPARIAMO IL PASCAL

Compattezza, concisione, chiarezza e notevoli potenzialità scientifiche, oltre a prestarsi ottimamente per calcoli gestionali e ad essere usato anche con i microcomputer, sono le caratteristiche che decretano il successo del PASCAL come linguaggio di programmazione. Non vi era però finora un testo che insegnasse a tutti a programmare in PASCAL: o perchè i libri esistenti sono troppo concisi, o troppo semplici, oppure perchè richiedono la conoscenza di altri linguaggi di programmazione, o, non ultimo, perchè in inglese.

Queste sono proprio le lacune che colma il libro un libro di divulgazione, incentrato sull'auto-apprendimento, che non tedia con accademismi non funzionali il lettore riportandolo "a scuola". I capitoli sono il più possibile organici, in modo che la loro consultazione sia semplice ed agevole. Un riassunto di quanto si apprenderà è posto all'inizio e non in fondo al capitolo, perchè il lettore possa subito avere un metro valutativo con cui verificare passo-passo il suo apprendimento. E poi, ci sono consigli, problemi, esercizi affinchè il libro sia "usato" e non letto, perchè occorre sapere come si usa un'istruzione piuttosto che conoscerne le differenze semantiche tra linguaggio e linguaggio. Con un lavoro graduale, partendo senza alcuna conoscenza di programmazione, dopo circa due settimane dovreste conoscere abbastanza bene il PASCAL. Un buon risultato, no?!

Per ordinare il volume utilizzate l'apposito tagliando d'ordine inserito in fondo alla rivista.

Formato 15 x 21

Codice 501A

SOMMARIO

GRUPPO EDITORIALE JACKSON

DIVISIONE LIBRI

Giuseppe Staluppi

Una biblioteca per ottimizzare

L'ottimizzazione di cui si parla qui è quella cosiddetta combinatoria: si ha un insieme di variabili che devono essere determinate in modo da soddisfare un insieme di condizioni rendendo nel contempo massima o minima (ottima) una funzione di tali variabili.

Per risolvere questi problemi si possono usare metodi analitici o metodi evolutivi. In questo manuale è descritta una biblioteca di programmi relativi ai metodi evolutivi più classici.

I programmi, com'è giusto, sono modulari, costituiscono cioè unità facilmente aggregabili ed usabili in base a regole comuni. Il linguaggio utilizzato è il Fortran, ma tutti i programmi sono facilmente convertibili in Basic, per un uso sui personal computer.

Un breve sguardo al contenuto: troviamo algoritmi per la programmazione lineare generale, per quella a numeri interi, per la programmazione dinamica, per i cammini sui grafi e per il PERT e il CPM, oltre a metodi di ricerca generali vincolati e non vincolati.

C. Baldissera, S. Ceri, A. Colorni
Metodi di ottimizzazione e programmi di calcolo
Milano: CLUP, 1981

Ogni libro che riporti biblioteche di programmi è da considerare con estrema attenzione nel panorama editoriale italiano. Questo particolare settore è stato infatti molto trascurato in passato e solo recentemente sono stati pubblicati da alcuni editori testi di software.

Elenco delle routine

FIBON	ricerca dell'ottimo di una funzione di una variabile in un intervallo con il metodo di Fibonacci
AUREO	ricerca dell'ottimo di una funzione di una variabile con il metodo della sezione aurea
BISEC	ricerca dell'ottimo di una funzione di una variabile con il metodo di bisezione
NABLA	ricerca dell'ottimo di una funzione di *n* variabili non lineare senza vincoli con il metodo del gradiente
FLETCH	ricerca dell'ottimo di una funzione di *n* variabili non lineare senza vincoli con il metodo di Fletcher Powel
PATTER	ricerca dell'ottimo di una funzione di *n* variabili non lineare senza vincoli con il metodo *pattern search*
ADRAS	ricerca dell'ottimo di una funzione di *n* variabili non lineare senza vincoli con il metodo della ricerca casuale adattativa
QUADR	ricerca dell'ottimo vincolato per un funzione quadratica di *n* variabili
ZOUTEN	ricerca dell'ottimo vincolato per una funzione non lineare di *n* variabili con il metodo delle direzioni ammissibili
SUMT	ricerca dell'ottimo vincolato per una funzione non lineare di *n* variabili con il metodo della funzione di penalità
SIMPL	programmazione lineare con l'algoritmo del simplesso
REVSIM	algoritmo del simplesso revisionato
TRASPO	problema del trasporto
MAGYAR	problema di assegnamento con metodo "ungherese"
FORFUL	flusso massimo in una rete
TAGLIO	programmazione lineare a numeri interi con l'algoritmo frazionario di Gomory
BALAS	programmazione lineare a variabili binarie con l'algoritmo di enumerazione implicita di Balas
INTEG	programmazione lineare a numeri interi trasformato in variabili binarie
DINAM	programmazione dinamica
GRAFIL	cammini minimi su grafi
GRAFLI	cammini minimi su grafi
GRAFGE	cammini minimi su grafi
PERT	risolve un reticolo di progetti collegati da attività
CPM	risolve un reticolo di progetti collegati da attività

Una biblioteca per ottimizzare

Per soddisfare gli utenti, questi libri devono tuttavia avere alcune particolari caratteristiche: i programmi devono essere facilmente "portabili", cioè la particolare versione del linguaggio utilizzato deve essere facilmente modificabile: meglio di tutto non utilizzare nessuna particolare versione, ma un linguaggio estremamente semplificato, per esempio solo una decina di istruzioni Fortran o Basic.

Una seconda caratteristica è la "modularità". Ogni pezzo, cioè ogni routine, deve essere completamente spiegata e commentata, deve essere chiaro quali sono gli input e quali gli output, quali le limitazioni e quali i controlli.

In terzo luogo i programmi devono essere ben commentati e documentati, con riferimenti precisi per una perfetta comprensione.

Avendo verificato alcuni dei programmi di questo libro, possiamo dire che le tre condizioni che abbiamo citato sono rispettate.

Le traduzioni dei programmi, per chi voglia implementarli su particolari calcolatori, sono particolarmente agevoli, i risultati sono sempre ottimi. Escluse poche eccezioni, questi algoritmi sono trasferibili su personal computer, compatibilmente con la capacità di memoria a disposizione.

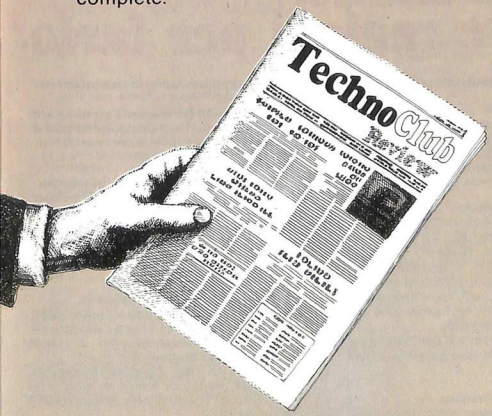
Il libro ha 463 pagine e costa ventimila lire: ma se si considera che vi sono contenuti 24 programmi, e che quindi il costo a programma è inferiore alle mille lire, se ne conclude che si tratta di un investimento che vale la pena di fare.

■

La guida sicura nel labirinto tecnologico.

TechnoClub è l'organizzazione di vendita per corrispondenza del libro tecnico (principalmente elettronica e informatica) nonché del software applicativo.
TechnoClub è anche il tuo consulente, la guida sicura per orientarsi nel labirinto dell'editoria tecnica, lo strumento ed il servizio essenziale per il numero crescente di persone che hanno compreso l'importanza della tecnologia nel mondo odierno.
Libri di base e didattici per imparare a capire; applicativi per realizzare e coltivare il proprio hobby; pratici per risolvere i problemi dell'attività quotidiana; di elevata specializzazione per migliorare il proprio background professionale o culturale. E altri ancora per soddisfare ogni esigenza.
TechnoClub offre solo il meglio della produzione tecnica editoriale. Per questo ha scelto di collaborare con qualificati editori italiani e soprattutto si avvale di un'équipe di professionisti che esamina, seleziona e propone le opere più significative e compiete.

TechnoClub ha instaurato rapporti di collaborazione con i più prestigiosi editori e software-house stranieri, per offrire tempestivamente, già da quest'anno, le opere più innovative in lingua originale e il software più interessante, appena disponibili. Tutti possono aderire al TechnoClub, assicurandosi un servizio garantito, professionale, veloce, unico nel suo genere. Esamina le modalità per diventare Socio e considera i numerosi vantaggi che ne derivano.



TechnoClub

i migliori libri tecnici
e il software a casa vostra.

Cod. IHC01

Cod. IHC02

Cod. IHC03

Cod. IHC04

Cod. IHC05

Cod. IHC06

Cod. IHC07

Cod. IFC04

Cod. IFC05

Cod. IGC01

Cod. IFH02

Cod. IFH04

Cod. IFH07

Cod. IFH08

Cod. IAK03

Cod. IAK04

Cod. IAK05

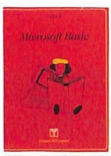
Cod. IAK06

Cod. IAK07

Cod. IAK08

Cod. IAK09

Cod. IHK02



Cod. IHK03

Cod. IHK04

Cod. IFK01

Cod. IFK02

Cod. IFK03

Cod. IFK04

Associati subito. Hai almeno 8 buone ragioni per farlo.

1. **Nessun impegno di acquisto.**
I Soci non sono vincolati all'acquisto di un numero minimo di libri durante il periodo di adesione al **TechnoClub**. Di conseguenza, scelta libera e nessuna imposizione, acquistando quello che si vuole, quando si vuole.

2. **Garanzia.**
I libri proposti dal **TechnoClub** costituiscono sempre la versione originale e più aggiornata delle edizioni in commercio.
Il **TechnoClub** garantisce quindi il contenuto e la veste tipografica originali.

3. **Convenienza certa.**
Il prezzo delle opere offerte ai Soci del **TechnoClub** è inferiore del 10% circa rispetto al prezzo di copertina dell'edizione in commercio. Il risparmio è perciò assicurato.

4. **Consulenza professionale per una scelta sicura.**
La selezione delle opere proposte dal **TechnoClub** è effettuata da un gruppo di esperti dei singoli settori.
Viene in tal modo offerto ai Soci un orientamento sicuro e garantita la massima affidabilità nella scelta.

5. **Informazione costante.**
A tutti i soci del **TechnoClub** viene inviata gratuitamente, ogni tre mesi, la rivista **"TechnoClub Review"**, che presenta l'assortimento, suddiviso per argomento e settore specifico di interesse, dei libri selezionati. Ogni libro viene illustrato con note esplicative che ne chiariscono il contenuto.
Il Socio viene in tal modo facilitato nella scelta, secondo le sue specifiche esigenze.

6. **Aggiornamento continuo.**
"TechnoClub Review" garantisce inoltre l'aggiornamento costante sulle novità editoriali.
Considerando l'evoluzione continua dei settori trattati, i Soci dispongono così di uno strumento efficace per tenersi tempestivamente aggiornati.

7. **Un ulteriore e interessante vantaggio.**
I Soci ricevono anche la tessera **TechnoClub**, un documento strettamente personale che dà diritto a sconti speciali sugli acquisti effettuati presso i negozi convenzionati, indicati sulla rivista **"TechnoClub Review"**.

8. **Praticità e comodità d'acquisto.**
Aderire al **TechnoClub** significa poter scegliere con tranquillità a casa propria consultando semplicemente la rivista **"TechnoClub Review"**.
Garanzia di libri sempre disponibili, nessuna perdita di tempo in lunghe ricerche... e i libri arrivano puntualmente a domicilio.

...e puoi già scegliere tra questi titoli.


Cod. IHC08


Cod. IHC09

Cod. IFC01

Cod. IFC02

Cod. IFC03


Cod. IFH11

Cod. IFH12

Cod. IDH02

Cod. IAK01

Cod. IAK02

Cod. IAK10

Cod. IAK14

Cod. IAK15

Cod. IAK16

Cod. IHK01

Cod. IFK05

Cod. IDK01

Cod. IDK02

Cod. IDK03

Cod. IIK01

32 PROGRAMMI CON IL PET
T. Rugg e P. Feldman - pag. 240, 1981

Trentadue programmi documentati, da eseguire su ogni tipi di PET. Ogni programma si compone di: scopo - come usarlo - esecuzione di prova (con fotografie schermo durante l'esecuzione) - lista del programma, semplici variazioni - routine principali - variabili principali - progetti suggeriti.
Cod. IHC01 **L. 8.500**

32 PROGRAMMI CON L'APPLE
T. Rugg e P. Feldman - pag. 248, 1981

Come sopra, per ogni tipo di Apple.
Cod. IHC02 **L. 8.500**

32 PROGRAMMI CON IL TRS-80
T. Rugg e P. Feldman - pag. 238, 1981

Come sopra, per il TRS-80.
Cod. IHC03 **L. 8.500**

IMPARATE IL BASIC CON IL PET
H.D. Peckham - pag. 245, 1981

Un corso di autoistruzione per chi desidera imparare il BASIC su un PET. Con esempi ed esercizi, partendo dai concetti più semplici, si arriva a programmare il PET, sfruttando tutte le possibilità.
Cod. IHC04 **L. 8.500**

Come diventare socio...

Per diventare Socio è sufficiente scegliere tra queste due semplici possibilità:

A) Versare l'importo di L. 8.000 quale quota di adesione.

B) Effettuare un primo acquisto di libri, per un importo minimo di L. 30.000.
In questo caso non si versa la quota di adesione.
Per acquisti inferiori a L. 30.000 va aggiunta la quota di adesione di L. 8.000.

In ambedue i casi, *il Socio ha diritto a ricevere gratuitamente la rivista* **"TechnoClub Review"** *per ben due anni e la tessera personale* con validità per lo stesso periodo.
Il Socio che nel corso dei due anni di adesione effettuerà acquisti di libri per un importo di almeno L. 60.000 avrà diritto al rinnovo automatico e gratuito dell'iscrizione al **TechnoClub** per un altro anno, conservando quindi tutti i vantaggi esclusivi.

Associati subito.

Spedisci oggi stesso la cedola di adesione

CEDOLA DI ADESIONE da compilare e spedire in busta chiusa a **TechnoClub** - Via Rosellini, 12 - 20124 Milano

Si, aderisco al **TechnoClub** scegliendo la seguente formula:
☐ A) Sola adesione con versamento di L. 8.000
☐ B) Adesione con acquisto dei seguenti libri per un importo totale di L. ..... + L. 1.500 per contributo fisso per spese di spedizione

Cod. .....Cod. .....Cod. .....

Cod. .....Cod. .....Cod. .....

☐ Contanti o francobolli allegati

☐ Assegno allegato n° .....
Banca .....

☐ Ho spedito l'importo a mezzo vaglia postale

☐ Ho versato l'importo sul ccp n° 19445204 intestato a **TechnoClub** - Milano

☐ Pagherò in contrassegno al postino al ricevimento dei volumi (valido solo per la formula B)

Nome .....

Cognome .....

Via .....

Città ..... Cap .....

Cod. Fiscale (per le aziende) .....

Data ..... Firma .....

Sono interessato principalmente a Libri di ...
☐ Elettrotecnica
☐ Elettronica e dispositivi elettronici
☐ Elettronica pratica ed hobbystica
☐ Misure elettroniche
☐ Radioriparazioni - TV Service
☐ Equivalenze dei semiconduttori
☐ Personal computer e calcolatrici
☐ Linguaggi e metodi di programmazione
☐ Informatica
☐ Informatica e organizzazione aziendale
☐ Comunicazioni: elementi e sistemi
☐ Microprocessori
☐ Saggistica elettronica e informatica
☐ Energie alternative
☐ Sistemi di regolazione e controllo
☐ Altri (specificare) .....

.....

Sono interessato anche a libri in lingua originale ...
☐ Inglese ☐ Francese ☐ Tedesco

... Sono interessato a Software per ...
☐ Apple
☐ Atari
☐ Commodore
☐ Sinclair
☐ Tandy Radio Shack
☐ Altri (specificare) .....

.....

...e puoi già scegliere tra questi titoli.

INTERVISTA SUL PERSONAL COMPUTER - HARDWARE
R. Diddav - pag. 244, 1981

Seicento domande e risposte sul mondo dei personal computer.
Questo volume, dedicato all'hardware, contiene un'introduzione, sotto forma di intervista, ai computer in generale e ai microprocessori in particolare.
Cod. IHC05 **L. 8.500**

INTERVISTA SUL PERSONAL COMPUTER - SOFTWARE
R. Diddav - pag. 200, 1981

Centinaia di domande e risposte sul mondo dei personal computer. Questo secondo volume, dedicato al software, contiene un'introduzione, sotto forma di intervista, alla programmazione dei computer in generale e dei microprocessori in particolare.
Cod. IHC06 **L. 8.500**

ASTRONOMIA CON IL CALCOLATORE TASCABILE
A. Jones - pag. 307, 1981

Il libro indica come usare i moderni calcolatori tascabili per risolvere diversi problemi astronomici in tempo minimo. Vengono introdotti metodi sia per logica algebrica che per logica polacca inversa.
Appendici con 57 programmi documentati per calcolatori con e senza schedine magnetiche.
Cod. IHC07 **L. 12.000**

LE SCIENZE CON IL CALCOLATORE TASCABILE
D.R. Green e J. Lewis - pag. 398, 1980

Il libro tratta le varie funzioni disponibili sui calcolatori dimostrando la possibilità di applicazione a numerosi problemi di fisica, chimica, biologia, matematica, ingegneria, con diversi esempi svolti e numerosi problemi presi dalle scienze, che il lettore deve svolgere.
Cod. IHC08 **L. 9.900**

MATEMATICA CON IL CALCOLATORE TASCABILE
P. Henrici - pag. 233, 1980

35 programmi scritti per l'HP-33E e per l'HP-25, che implementano algoritmi di teoria dei numeri, soluzioni di equazioni, teoria della stabilità algebrica, analisi delle serie di potenze, integrazione e funzioni speciali, completati dai diagrammi di flusso e dalle istruzioni operative.
Cod. IHC09 **L. 13.950**

IMPARIAMO A PROGRAMMARE IN BASIC CON IL PET/CBM
R. Bonelli - pag. 180, 1981

Un corso didattico di programmazione che dalle premesse generali arriva alle frasi BASIC, ai concetti di stringhe, al trattamento dei file, all'uso dei floppy disk, alle norme operative, al DOS, sino al linguaggio macchina.
Cod. IFC01 **L. 9.000**

IMPARIAMO A PROGRAMMARE IN BASIC CON IL VIC/CBM
R. Bonelli - pag. 176, 1981

Un manuale che inizia i principianti alla programmazione in BASIC del VIC20, privilegiando l'aspetto pratico, per imparare nel frattempo che cos'è un programma.
Cod. IFC02 **L. 9.900**

GUIDA AL SINCLAIR ZX81 - ZX80 e NUOVA ROM
R. Bonelli - pag. 264, 1982

Vengono confrontati i tre calcolatori, che presentano notevoli differenze nel sistema di gestione e nel BASIC utilizzato, con considerazioni sulla loro potenzialità e approfondimenti su argomenti specifici: trasformazione dei programmi da un calcolatore all'altro, sistema operativo, gestione dei file, linguaggio macchina.
Cod. IFC03 **L. 14.500**

DAI - MANUALE DEL MICROCOMPUTER
R. Bonelli C. Fiorentini - pag. 145, 1981

La prima parte del manuale insegna l'utilizzo del calcolatore DAI, dai primi passi nella programmazione BASIC alla spiegazione delle caratteristiche di questo personal. La seconda parte contiene le specifiche sull'implementazione del linguaggio BASIC sul calcolatore DAI.
Cod. IFC04 **L. 8.000**

INTRODUZIONE AL PERSONAL E BUSINESS COMPUTING
R.Zaks - pag. 224, 1979

Un testo didattico per principianti e per utenti che vogliono ampliare la loro conoscenza.
Un'introduzione ai concetti, alle periferiche e alle tecniche fino ai metodi di valutazione per una scelta oculata.
Cod. IFC05 **L. 12.500**

JUNIOR COMPUTER - Vol. 1
A. Nachtmann e G.H. Nachbar - pag. 178, 1981

Junior Computer è il microelaboratore da autocostruire su un unico circuito stampato. Il sistema base e questo libro sono l'occorrente per l'apprendimento. Prossimamente verranno pubblicati altri volumi relativi all'espandibilità del sistema.
Cod. IGC01 **L. 9.900**

PROGRAMMAZIONE DELLO Z-80
R. Zaks - pag. 530, 1981

Libro ideato come testo autonomo e progettato sotto forma di corso per imparare la programmazione in linguaggio Assembler del microprocessore Z-80: dai concetti di base alle tecniche di programmazione più avanzate, con risoluzione obbligatoria di vari esercizi.
Cod. IFH02 **L. 21.500**

PROGRAMMAZIONE DEL 6502
R. Zaks - pag. 375, 1981

Come sopra per il microprocessore 6502
Cod. IFH04 **L. 19.800**

USARE IL MICROPROCESSORE
G. Giaccaglini - pag. 295, 1981

Un testo per far capire l'utilizzo più razionale del microprocessore, soprattutto in relazione al controllo di impianti e processi, con diversi esempi di simulazione e prospettando problemi di interfacciamento hardware.
Cod. IFH07 **L. 13.500**

APPLICAZIONI DEL 6502
R. Zaks - pag. 214, 1981

Tecniche e programmi per applicazioni tipiche del 6502. I programmi sono, con poche varianti, applicabili direttamente su qualunque microcomputer su scheda basato sul 6502, quali il KYM, il SYM e l'AIM 65 e altri e consentono al lettore alcune realizzazioni pratiche.
Cod. IFH08 **L. 12.000**

INTRODUZIONE AI MICROCOMPUTER VOL. 0 IL LIBRO DEL PRINCIPIANTE
A. Osborne - pag. 240, 1980

Una visione complessiva su calcolatori ed elaboratori, con concetti generali e terminologia di base per capire la tecnologia usata. Vengono illustrate le singole parti del sistema, con le possibilità di espansione e componenti accessori.
Cod. IFH11 **L. 12.500**

INTRODUZIONE AI MICROCOMPUTER VOL. 1 IL LIBRO DEI CONCETTI FONDAMENTALI
A. Osborne - pag. 321, 1980

Il libro presenta la struttura logica fondamentale su cui sono basati i sistemi a microcomputer. Usando i concetti comuni a ogni sistema a microprocessore, viene illustrata l'architettura, la programmazione, le possibilità e l'operatività di un microcomputer, con un set finale ipotetico di istruzioni per la simulazione delle possibili situazioni reali in cui si verrà a trovare con i vari microprocessori.
Cod. IFH12 **L. 14.400**

MICROPROCESSORI 8080 e BUS MMS-8
a cura di L'EMMECI - pag. 160, 1981

Un testo di "consulenza" con caratteristiche didattiche per chi ha già una conoscenza di base su circuiti logici, algebra di Boole e aritmetica binaria. Inteso come manuale commentato del microprocessore 8080 e del bus adottato presso il laboratorio di microcalcolatori del Politecnico di Milano. Si pone come riferimento per chi programma nel linguaggio Assembler 8080.
Cod. IDH02 **L. 5.400**

INTRODUZIONE ALL'APL E ALLE SUE APPLICAZIONI
C. Camerata - pag. 266, 1980

Il linguaggio di programmazione APL si basa su funzioni logico-matematiche di tipo generale. Adatto quindi ad applicazioni in vari settori, è stato implementato su più sistemi per tutti i livelli di elaboratori. Il libro si prefigge lo scopo di fornire agli utenti una conoscenza generale di questo linguaggio.
Cod. IAK01 **L. 10.800**

COME PROGRAMMARE CON IL FORTRAN
P. Ridolfi/H. Coen - pag. 148, 1980

Una guida allo studio del Fortran, concepita con intenti prevalentemente didattici, che pone in grado di scrivere e capire semplici programmi utilizzando questo "linguaggio simbolico", particolarmente adatto per agevolare gli scienziati e i tecnici all'impiego dell'elaboratore elettronico.
Cod. IAK02 **L. 5.400**

IL FORTRAN - TEORIA ED ESERCIZI
P. Ridolfi - pag. 168, 1981

Viene descritto il complesso di regole che costituiscono la grammatica del Fortran, con abbondanti esemplificazioni pratiche, onde acquisire una graduale e completa padronanza del linguaggio. Non si richiede al lettore una conoscenza approfondita degli elaboratori elettronici.
Cod. IAK03 **L. 4.950**

ESERCITAZIONI DI ASSEMBLER
E. Vitale/A. Hernandez - pag. 134, 1981

Una raccolta di 12 esercizi di crescente difficoltà, di ognuno dei quali viene dato l'enunciato, il diagramma a blocchi e la soluzione; quest'ultima costituita dalle liste di compilazioni integrali per ogni problema enunciato.
Cod. IAK04 **L. 4.950**

COME PROGRAMMARE CON IL PL/1
G. Romano - pag. 205, 1982

Il volume presenta le caratteristiche più importanti del linguaggio PL/1, che può definirsi un ibrido tra i linguaggi per utenti scientifici e quelli per utenti commerciali.
Con il ricorso ad esempi esplicativi, si pone come guida per lo studio autodidattico o anche guidato del PL/1.
Cod. IAK05 **L. 7.200**

COME FARE I DIAGRAMMI A BLOCCHI
A.C. Wright - pag. 84, 1981

Con un'abbondante documentazione di esemplificazioni, il libro si propone l'agevole assimilazione della tecnica della diagrammazione a blocchi attraverso numerosi esercizi da svolgere per proprio conto.
Cod. IAK06 **L. 4.500**

L'ASSEMBLER
E. Vitale - pag. 172, 1981

L'opera, rivolta a chi abbia già qualche conoscenza di programmazione, illustra il flessibile linguaggio Assembler, esponendo le varie istruzioni e spiegandone il significato, le modalità d'uso e i casi in cui possono rivelarsi utili o necessarie.
Cod. IAK07 **L. 6.750**

COME PROGRAMMARE CON l'RPG I - RPG II - RPG III
A.C. Wright - pag. 266, 1981

Nella prima parte del testo viene descritto l'RPG I: la sua logica, l'uso di unità periferiche facili da gestire, le maschere di stampa, ecc. Nella seconda e terza parte vengono descritte le istruzioni aggiuntesi nelle più recenti versioni di questo linguaggio: l'RPG II e il compilatore RPG III.
Cod. IAK08 **L. 10.800**

COME PROGRAMMARE CON IL COBOL
G. Sarri - pag. 129, 1982

L'opera espone le regole da seguire per redigere un programma in Cobol, con numerose esemplificazioni e si propone di porre il lettore in grado di scrivere programmi in Cobol e di capirli.
Cod. IAK09 **L. 4.500**

PROGRAMMAZIONE DEGLI ELABORATORI ELETTRONICI
G. Bertoldi - pag. 286, 1982

La prima parte illustra i principi generali dell'elaborazione automatica dei dati, le caratteristiche e il funzionamento degli elaboratori e i problemi di programmazione. La seconda parte è dedicata all'analisi (studio e definizione dei problemi da svolgere); la terza parte illustra la tecnica di stesura dei diagrammi a blocchi. Segue capitolo sul Cobol e uno sui sistemi operativi, più glossario in appendice.
Cod. IAK10 **L. 7.200**

PRINCIPI DI PROGETTAZIONE DEI COMPILATORI
D. Gries - pag. 566, 1980

Il volume descrive le tecniche utilizzabili nella scrittura di compilatori per linguaggi ad alto livello, quali il Fortran ed il PL/1, illustrando teoria ed aspetti pratici connessi con tale scrittura.
Cod. IAK14 **L. 26.100**

IL BASIC - TEORIA ED ESERCIZI
E. Spoletini - pag. 298, 5ª ediz. 1982

Il volume, rivolto anche ai "non addetti ai lavori", presenta le caratteristiche del linguaggio BASIC. L'impostazione didattica si prefigge di far apprendere agevolmente il linguaggio per gradi. Quasi tutti gli esercizi, differenziati come argomenti e grado di difficoltà, sono corredati da diagrammi a blocchi.
Cod. IAK15 **L. 10.800**

IL COBOL - TEORIA ED ESERCIZI
E. Spoletini - pag. 396, 1982

La prima parte dell'opera espone gli elementi per scrivere un programma di media difficoltà; nella seconda parte vengono trattati problemi relativi alle tabelle e ai flussi organizzati in modo non sequenziale. Ogni capitolo è corredato di numerosi esempi ed esercizi.
Cod. IAK16 **L. 13.500**

IL PROGETTO DEI MICROCOMPUTER: SOFTWARE
C.A. Ogdin - pag. 247, 1981

Vengono esposte tecniche aggiornate, tra cui la programmazione strutturata, con diversi esempi di progettazione software che utilizza una notazione simile al Pascal. Viene dato risalto sia al progetto di strutture dati che di procedure strutturate.
Cod. IHK01 **L. 12.150**

PASCAL
P.M. Chirlian - pag. 200, 1981

Questo libro, inteso come manuale di autoistruzione o libro di testo in un corso, per chi non ha esperienza di calcolatori o programmazione, presenta il linguaggio Pascal che permette la "programmazione strutturata". Ogni capitolo si conclude con una serie di esercizi.
Cod. IHK02 **L. 7.650**

MICROSOFT BASIC
K. Knecht - pag. 150, 1981

Un manuale di introduzione al Microsoft BASIC, sorto dall'esigenza di standardizzazione del BASIC per l'implementazione su una varietà di personal computer. Viene dato rilievo alle diverse caratteristiche e forme che il Microsoft Basic può presentare e viene dato particolare risalto alla versione implementata sul TRS-80.
Cod. IHK03 **L. 5.850**

MUSICA CON IL CALCOLATORE
R.C. Zaripov - pag. 169, 1979

Una monografia dedicata al problema della composizione di musica con l'aiuto di calcoli matematico-probabilistici, con rassegna degli studi svolti nel mondo sull'aiuto dei computer per la composizione e l'analisi della musica, oltre alle regole trovate dall'autore per realizzare un modello che simula l'attività di un compositore.
Cod. IHK04 **L. 6.750**

CP/M CON MP/M
R. Zaks - pag. 309, 1982

Il libro si prefigge di rendere agevole l'uso del CP/M (nelle versioni CP/M 1.4 - CP/M 2.2 - sistema operativo multiutente MP/M); il sistema operativo progettato per semplificare l'utilizzo di un microcomputer, disponibile su quasi tutti gli elaboratori basati su microprocessore 8080 e Z80 e su certi sistemi utilizzanti il 6502.
Cod. IFK01 **L. 19.800**

PROGRAMMARE IN ASSEMBLER
A. Pinaud - pag. 153, 1982

Il libro, destinato in particolare a chi già ha una buona conoscenza di un linguaggio evoluto molto semplice come il BASIC, fornisce i rudimenti che consentono di programmare in Assembler, con numerosi esempi pratici. Come Assembler esistente è stato scelto quello dello Z80.
Cod. IFK02 **L. 9.000**

IMPARIAMO IL PASCAL
F. Waldner - pag. 162, 1981

Un libro di divulgazione, incentrato sull'autoapprendimento del linguaggio Pascal, con consigli, problemi.
Un testo da "usare" e non da "leggere", secondo l'intento dichiarato dall'autore.
Cod. IFK03 **L. 9.000**

PASCAL-MANUALE E STANDARD DEL LINGUAGGIO
K. Jensen/N. Wirth - pag. 179, 1981

La prima parte è costituita dal manuale di apprendimento delle regole e dei principi del Pascal (manuale utente); la seconda, dalla descrizione dello standard del linguaggio. Serie di tavole contenenti la definizione del Pascal attraverso il BNF e anche tramite le carte sintattiche.
Cod. IFK04 **L. 9.000**

INTRODUZIONE AL BASIC
P. Le Beux - pag. 314, 1981

Un corso rivolto ai principianti, che illustra tutti gli aspetti del BASIC su differenti sistemi. Con numerosi esempi, il lettore può verificare con immediatezza il reale apprendimento raggiunto.
Cod. IFK05 **L. 16.500**

METODI DI OTTIMIZZAZIONE E PROGRAMMI DI CALCOLO
C. Baldissera/S. Ceri/A. Colorni - pag. 468, 1981

Vengono esposte le caratteristiche di un gruppo di programmi (attualmente esistente presso il Centro di Teoria dei sistemi del CNR), che si compone di alcuni classici algoritmi strutturati per privilegiare la semplicità d'uso, l'uniformità e modularità fra i vari programmi. Il linguaggio usato è il Fortran.
Cod. IDK01 **L. 18.000**

PASCAL - DAL MICROPROCESSORE AL GRANDE ELABORATORE
G. Cioni/S. Crespi Reghizzi/M. Moscarini
pag. 194, 1981

Il testo, rivolto a chi già conosce la programmazione del calcolatore, descrive nella prima parte il linguaggio Pascal standard nelle sue varie parti, con 40 esempi di programmi, collaudati sul calcolatore.
Nella seconda parte, dopo un cenno alle tecniche di compilazione, si presentano le schede tecniche di alcune versioni del Pascal disponibili per elaboratori di vasta diffusione.
Cod. IDK02 **L. 7.200**

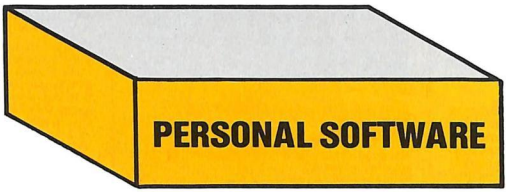
LISP - LINGUAGGIO E METODOLOGIA DI PROGRAMMAZIONE
G. Gini, M. Gini, G. Guida - pag. 204, 1981

Con molti esempi di diversa complessità nel testo vengono illustrati tutti gli aspetti del linguaggio LISP, oggi diffuso anche su mini e personal computer, utilizzato per progettare sistemi di intelligenza artificiale e per la soluzione di problemi di elaborazione non numerica.
Cod. IDK03 **L. 8.100**

PROGRAMMARE IN FORTRAN 375 PROBLEMI RISOLTI
S. Lipschutz/A. Poe - pag. 314, 1980

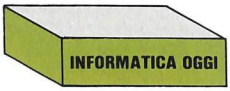
Lo scopo del libro è di presentare il linguaggio Fortran e di insegnare a risolvere dei problemi con esso. Oltre alla sintassi, viene insegnato come scrivere dei programmi Fortran con chiarezza, insistendo sia sulle tecniche che sulle buone abitudini di programmazione. Vengono esaminati i principi più importanti sia del Fortran standard che del Fortran strutturato.
Cod. IIK01 **L. 10.800**

Scegli tra centoventisette combinazioni d'abbonamento. Tutte vantaggiose.



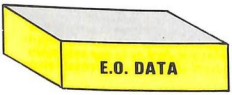
10 numeri
L. 30.000
anzichè L. 35.000
estero L. 48.000

11 numeri
L. 31.000
anzichè L. 38.500
estero L. 49.600

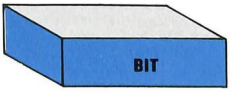


11 numeri
L. 26.500
anzichè L. 33.000
estero L. 42.400

22 numeri
L. 35.000
anzichè L. 44.000
estero L. 56.000



4 numeri
L. 6.500
anzichè L. 8.000
estero L. 10.400



11 numeri
L. 26.000
anzichè L. 33.000
estero L. 42.000

10 numeri
L. 24.000
anzichè L. 30.000
estero L. 38.400

Come calcolare da soli le altre 120 combinazioni di abbonamento ...

abbonamento a 2 riviste. Prezzo Scontato 1ª rivista + Prezzo Scontato 2ª rivista - L. 2.000
abbonamento a 3 riviste. P.S. 1ª + P.S. 2ª + P.S. 3ª - L. 4.000
abbonamento a 4 riviste. P.S. 1ª + P.S. 2ª + P.S. 3ª + P.S. 4ª - L. 8.000
abbonamento a 5 riviste. P.S. 1ª + P.S. 2ª + P.S. 3ª + P.S. 4ª + P.S. 5ª - L. 11.500
abbonamento a 6 riviste. P.S. 1ª + P.S. 2ª + P.S. 3ª + P.S. 4ª + P.S. 5ª + P.S. 6ª - L. 15.000
abbonamento a tutte e 7 le riviste L. 149.000 anzichè L. 226.500 estero L. 239.000

Per l'estero l'importo risultante va aumentato del 60% (moltiplicare 1,6)

Esempio: Informatica Oggi + Personal Software — Italia 26.500 + 30.000 — 2.000 = 54.500 — Estero 54.500 x 1,6 = 87.200

Attenzione: per i versamenti **utilizzate il c.c.p. n° 11666203** intestato a Gruppo Editoriale Jackson - Milano. Oppure inviate un assegno al nostro ufficio abbonamenti.

GRUPPO EDITORIALE JACKSON
SERVIZIO ABBONAMENTI

Gioco del 15

Chi è stato ragazzo nei primi anni sessanta si ricorderà senz'altro di questo gioco, che in quegli anni ebbe una diffusione paragonabile a quella che il cubo di Rubik ha ai giorni nostri.

Come è noto il gioco, che si esegue su di una scacchiera 4×4, consiste nel riordinare su quattro file i numeri dall'uno al quindici, usufruendo per gli spostamenti dell'unica casella libera sulla scacchiera. Si può giocare come "solitario" oppure si possono organizzare sfide tra più giocatori, usufruendo del controllo del calcolatore che, mossa per mossa, aggiornerà i vari "record".

Qui a fianco trovate la versione PET/CBM valida per tutti i modelli a 40 colonne. Più avanti ci sono le versioni per il TRS-80 mod. I e per l'Apple II. ■

commodore

```
10 REM ****************************************
20 REM *                                      *
30 REM *              GIOCO DEL 15            *
40 REM *     VERSIONE PET/CBM 3032 E 4032     *
50 REM *                                      *
60 REM *           PERSONAL SOFTWARE          *
70 REM *                                      *
80 REM ****************************************
90 REM
100 DIM N(4,4)
110 FOR NUM=1 TO 15
120 I=INT(RND(TI)*4+1): J=INT(RND(TI)*4+1)
130 IF N(I,J)<>0 GOTO 120
140 N(I,J)=NUM: NEXT NUM
150 PRINT CHR$(44)
160 GOSUB 650
170 GOSUB 580
180 X=22: GOSUB 800
190 FOR Y=1 TO 40: PRINT" ";: NEXT Y
200 X=23: GOSUB 800
210 FOR Y=1 TO 25: PRINT" ";: NEXT Y
220 X=23: GOSUB 800
230 INPUT"BATTI LA TUA MOSSA";M$
240 IF LEN(M$)<1 GOTO 270
250 IF ASC(M$)=70 THEN END
260 M=VAL(M$): IF M>0 AND M<16 GOTO 300
270 X=22: GOSUB 800
280 PRINT"ERRORE. USA NUMERI DA 1 A 15."
290 GOTO 200
300 X=23: GOSUB 800
310 FOR I=1 TO 4: FOR J=1 TO 4
320 IF N(I,J)=M GOTO 340
330 NEXT: NEXT
340 II=I+1: JJ=J: GOSUB 530
350 IF FLAG=1 GOTO 460
360 II=I-1: JJ=J: GOSUB 530
370 IF FLAG=1 GOTO 460
380 II=I: JJ=J+1: GOSUB 530
390 IF FLAG=1 GOTO 460
400 II=I: JJ=J-1: GOSUB 530
410 IF FLAG=1 GOTO 460
420 X=22: GOSUB 800
430 FOR Y=1 TO 32: PRINT" ";: NEXT Y
440 X=22: GOSUB 800: PRINT"MOSSA NON PERMESSA."
450 GOTO 200
460 N(I,J)=0: N(II,JJ)=M
470 X=22: GOSUB 800
480 MOSSE =MOSSE+1
490 X=4: GOSUB 800: PRINTTAB(34)"MOSSE"
500 PRINTTAB(35)MOSSE
510 X=2+J*4: GOSUB 800: PRINTTAB(7+I*5)"  ";
520 GOTO 170
530 FLAG=0
540 IF II<1 OR II>4 OR JJ<1 OR JJ>4 GOTO 570
550 IF N(II,JJ)<>0 GOTO 570
560 FLAG=1
570 RETURN
580 FOR I=1 TO 4: FOR J=1 TO 4
590 IF N(I,J)=0 GOTO 630
600 X=2+J*4: GOSUB 800
610 PRINTTAB(8+I*5-LEN(STR$(N(I,J)))+1);
620 PRINT N(I,J)
630 NEXT: PRINT: NEXT
640 RETURN
650 PRINT CHR$(147)
660 PRINTTAB(9)"G I O C O   D E L   1 5"
670 FOR RIGA=4 TO 20 STEP 4
680 PRINT CHR$(18);
690 X=RIGA: GOSUB 800: PRINTTAB(10);
700 FOR H=10 TO 30: PRINT" ";: NEXT H
710 PRINT: PRINT: NEXT RIGA: PRINT
720 PRINT CHR$(18);
730 FOR L=5 TO 19: X=L: GOSUB 800
740 PRINTTAB(10)" ";
750 PRINTTAB(15)" ";TAB(20)" ";
760 PRINTTAB(25)" ";TAB(30)" ";
770 NEXT L: PRINT: PRINTTAB(1)
780 X=23: GOSUB 800
790 PRINT"BATTI 'F' PER FINIRE";: RETURN
800 PRINT CHR$(19);
810 FOR Y=1 TO X-1
820 PRINT CHR$(17);
830 NEXT Y: RETURN
```

apple II

```
10 REM ******************************************
20 REM *                                        *
30 REM *              GIOCO DEL 15              *
40 REM *          VERSIONE APPLE II             *
50 REM *                                        *
60 REM *          PERSONAL SOFTWARE             *
70 REM *                                        *
80 REM ******************************************
90 REM
100 DIM N(4,4)
110 FOR NUM=1 TO 15
120 I=INT(RND(1)*4+1): J=INT(RND(1)*4+1)
130 IF N(I,J)<>0 GOTO 120
140 N(I,J)=NUM: NEXT NUM
150 TEXT: HOME
160 GOSUB 500
170 GOSUB 450
190 VTAB 23: INPUT"BATTI LA TUA MOSSA ";M$
200 IF LEN(M$)>1 GOTO 230
210 IF ASC(M$)=69 THEN END
220 M=VAL(M$): IF M<>0 AND M<16 GOTO 250
230 VTAB 22: PRINT"ERRORE. USA I NUMERI DA 1 A 15."
240 GOTO 190
250 VTAB 23: CALL-868
260 FOR I=1 TO 4: FOR J=1 TO 4
270 IF N(I,J)=M GOTO 280: NEXT J,I
280 II=I+1: JJ=J: GOSUB 400: IF FLAG=1 GOTO 330
290 II=I-1: JJ=J: GOSUB 400: IF FLAG=1 GOTO 330
300 II=I: JJ=J+1: GOSUB 400: IF FLAG=1 GOTO 330
310 II=I: JJ=J-1: GOSUB 400: IF FLAG=1 GOTO 330
320 VTAB 22:
    PRINT"MOSSA NON PERMESSA.              ":
    GOTO 190
330 N(I,J)=0: N(II,JJ)=M
340 VTAB 22: CALL-868
350 MOSSE=MOSSE+1
360 VTAB 4: HTAB 24: PRINT"MOSSE"
370 VTAB 5: HTAB 26: PRINT MOSSE
380 VTAB 2+J*4: HTAB 7+I*5: PRINT"  ";
390 GOTO 170
400 FLAG=0
410 IF II<1 OR II>4 OR JJ<1 OR JJ>4 GOTO 440
420 IF N(II,JJ)<>0 GOTO 440
430 FLAG=1
440 RETURN
450 FOR I=1 TO 4: FOR J=1 TO 4
460 IF N(I,J)=0 GOTO 490
470 HTAB 8+I*5-(N(I,J)>9): VTAB 2+J*4
480 PRINT N(I,J)
490 NEXT J,I: RETURN
500 VTAB 1: HTAB 9: PRINT"G I O C O   D E L   15"
510 INVERSE: FOR LINE=4 TO 20 STEP 4
520 VTAB LINE: HTAB 10
530 FOR H=10 TO 30: PRINT" ";: NEXT H,LINE
540 FOR L=5 TO 19: VTAB L: HTAB 10: PRINT" ";
550 HTAB 15: PRINT" ";: HTAB 20: PRINT" ";
560 HTAB 25: PRINT" ";: HTAB 30: PRINT" ";
570 NEXT L: NORMAL: HTAB 1
580 VTAB 24: PRINT"BATTI 'E' PER FINIRE.";: RETURN
```

L'INFORMATICA IN LIGURIA

L'informatica è un settore giovane e quando si è giovani è facile sbagliare.
Affidandovi ad un centro specializzato dove i prodotti sono selezionati e l'assistenza è garantita da personale esperto, potrete risolvere i vostri problemi con le soluzioni più moderne e collaudate.
Nei nostri centri dimostrativi scoprirete che l'utilizzo di un computer è più facile di quanto pensiate, se i programmi rispondono alle vostre esigenze:

GESTIONE ORDINI CLIENTI
GESTIONE ORDINI FORNITORI
BOLLA DI ACCOMPAGNAMENTO
FATTURAZIONE RIEPILOGATIVA
FATTURAZIONE IMMEDIATA
PORTAFOGLIO EFFETTI
ESTRATTO CONTO
CONTABILITA' GENERALE-IVA
CONTABILITA' CLIENTI
CONTABILITA' FORNITORI

GESTIONE DEL PERSONALE
PAGHE PER EDILIZIA
CONTABILITA' SEMPLIFICATA
GESTIONE MAGAZZINO
PARCELLAZIONE PRESTAZIONI
DICHIARAZIONE REDDITI-740
CONTABILITA' FINANZIARIA
GESTIONE ENTI PUBBLICI
PAGHE PER SCUOLE

INGEGNERIA CIVILE E STRUTTURALE
COMPUTI METRICI LEGGE 373
GESTIONE RISTORANTI-ALBERGHI
GESTIONE CARTELLE CLINICHE
AMMINISTRAZIONE CONDOMINI
AGENZIA IMMOBILIARI-ASSICURAZIONI
PREVENTIVAZIONE CANTIERISTICA
E MOLTI ALTRI PROGRAMMI
SETTORI PERSONALIZZABILI

VIENI A SCOPRIRE CHE DIFFERENZA C'È

computer center

ASSOCIATO computer city

GENOVA - C.so Gastaldi, 77R - Tel. 010/300.797
LAVAGNA - C.so Buenos Aires, 125 - Tel. 0185/314.142

L'INFORMATICA SU CUI PUOI CONTARE.

Tandy
Radio Shack

```
10 REM ********************************************
20 REM *                                          *
30 REM *              GIOCO DEL 15                *
40 REM *          VERSIONE TRS-80 MOD. I          *
50 REM *                                          *
60 REM *            PERSONAL SOFTWARE             *
70 REM *                                          *
80 REM ********************************************
90 REM
100 CLEAR 200: DIM N(4,4)
110 FOR NUM=1 TO 15
120 I=RND(4): J=RND(4)
130 IF N(I,J)<>0 GOTO 120
140 N(I,J)=NUM: NEXT NUM
150 CLS
160 GOSUB 490
170 GOSUB 440
180 PRINT@832,STRING$(63," ")
190 PRINT@896,"BATTI LA TUA MOSSA";: INPUT M$
200 IF LEN(M$)<1 GOTO 230
210 IF M$="F" THEN END
220 M=VAL(M$): IF M<>0 AND M<16 GOTO 250
230 PRINT@832,STRING$(63," ");
    PRINT@832,"ERRORE. USA I NUMERI DA 1 A 15."
240 GOTO 190
250 FOR I=1 TO 4: FOR J=1 TO 4
260 IF N(I,J)=M GOTO 280
270 NEXT J,I
280 II=I+1: JJ=J: GOSUB 390: IF FLAG=1 GOTO 330
290 II=I-1: JJ=J: GOSUB 390: IF FLAG=1 GOTO 330
300 II=I: JJ=J+1: GOSUB 390: IF FLAG=1 GOTO 330
310 II=I: JJ=J-1: GOSUB 390: IF FLAG=1 GOTO 330
320 PRINT@832,STRING$(63," ");
    PRINT@832,"MOSSA NON PERMESSA.": GOTO 190
330 N(I,J)=0: N(II,JJ)=M
340 MOSSE=MOSSE+1
350 PRINT@242,"MOSSE";
360 PRINT@307,MOSSE;
370 X=128*(J+1)+18+1*5: PRINT@X,"   ";
380 GOTO 170
390 FLAG=0
400 IF II<1 OR II>4 OR JJ<1 OR JJ>4 GOTO 430
410 IF N(II,JJ)<>0 GOTO 430
420 FLAG=1
430 RETURN
440 FOR I=1 TO 4: FOR J=1 TO 4
450 IF N(I,J)=0  GOTO 470
460 X=128*(J+1)+18+I*5+(N(I,J)>9);
    PRINT@X,N(I,J);
470 NEXT J,I
480 RETURN
490 CLS: PRINT@960,"BATTI 'F' PER FINIRE";
500 PRINT@84,"G I O C O   D E L   1 5"
510 FOR RIGA=10 TO 34 STEP 6: FOR COL=42 TO 82
520 SET(COL,RIGA): NEXT COL,RIGA
530 FOR COL=42 TO 82 STEP 10: FOR RIGA=10 TO 34
540 SET (COL,RIGA): SET(COL+1,RIGA): NEXT RIGA,COL
550 RETURN
```

Una nuova generazione di italiani

PRAKTI 80 - A. VALERI

General Processor Sistema 4

GPS4 è il nome della nuova famiglia di elaboratori General Processor: elaboratori perfetti, nati dalla esperienza della prima azienda italiana costruttrice di piccoli computer.
I GPS4 sono tutti italiani: italiani nel progetto, italiani nella costruzione, italiani nel design, elegante ed essenziale come quello di un'auto sportiva di gran classe. Hanno una tastiera italiana, separata, davanti alla quale ogni dattilografa si trova subito a suo agio perché la Z, la W e la M sono al loro posto e perché, come in una calcolatrice, ci sono i tasti doppio e triplo zero.
E sono italiani anche nella assistenza.
Con i loro 128K RAM minimi (estendibili a oltre 200), due terminali collegabili e con una ineguagliabile biblioteca di software di base ed applicativo, i GPS4 rappresentano lo "status of the art" della moderna miniinformatica, per la quale rappresentano e rappresenteranno negli anni futuri un importante punto di riferimento.
Una raccomandazione: non fatevi influenzare dallo styling: i GPS4 non sono semplicemente i più belli; sono semplicemente i migliori.

Alcuni OEM General Processor

Milano: PGE: 02/28.22.225 - Como e Varese: SIAEMME: 0331/67.96.75 - Alessandria: CID: 0131/34.44.18 - Modena: Data: 059/68.80.90 - Bologna: Computer Systems: 051/79.94.21 - Pistoia: CEIA: 0572/51.611 - Firenze: R2 Data: 055/41.11.42 - Firenze: Aeffe: 055/75.27.89 - Prato: Gerva: 0574/59.26.94 - S. Croce/Arno (PI): Dainelli: 0571/31.805 -Arezzo: Tecem: 0575/28.848 - Arezzo: Etruria Sistemi: 0575/35.59.71 -Livorno: CEDO5: 0586/25.395 - Siena: Tecnocomputer: 0577/74.03.34 -Roma: General Computer: 06/52.84.032 - Latina: Conlax: 0771/22.503 -Napoli: CompuSystems: 081/46.36.02 - Napoli: Tecnodata: 081/24.21.66 - Calabria: Tripodi: 0984/99.21.42 - Spagna (Madrid)/Vimesa: 690.20.29

GENERAL PROCESSOR s.r.l.- elaboratori italiani - Firenze
Tel. 055/43.55.27 - 43.763.88 - Tlx 571034 GENPRO I

Bersaglio

Questo gioco per due persone si spiega da sé: è il classico gioco del bersaglio con le freccette. Ogni giocatore ne ha a disposizione tre alla volta; l'obiettivo è di raggiungere 210 punti prima dell'avversario.

Qui a fianco vi è la versione per il PET/CBM. Più avanti trovate quella per il TRS-80. ■

commodore

```
10 REM ****************************************
20 REM *                                      *
30 REM *              BERSAGLIO               *
40 REM *      VERSIONE PET/CBM 3032 E 4032    *
50 REM *                                      *
60 REM *          PERSONAL SOFTWARE           *
70 REM *                                      *
80 REM ****************************************
90 REM
100 DIM N1$(20),N2$(20),A$(20)
110 F=230: D=5
120 PRINT CHR$(147)
130 C1=98:C2=214:C3=127:C4=102:C5=160
140 PRINTTAB(15)"BERSAGLIO": PRINT: PRINT
150 INPUT"NOME DEL PRIMO GIOCATORE";N1$
160 INPUT"NOME DEL SECONDO GIOCATORE";N2$
170 PRINT CHR$(147)
180 GOSUB 280
190 S1=0: S2=0: N=1
200 GOSUB 690
210 FOR X=0 TO 39
220 POKE 33567+X,30
230 IF PEEK(151)=59 THEN GOSUB 410
240 POKE 33567+X,32
250 NEXT
260 GOTO 210
270 END
280 FOR H=1 TO 39: FOR F=1 TO 3
290 POKE(32768+H+160+(K-1)*40),32: NEXT: NEXT
300 FOR VL=1 TO 3
310 FOR F=10 TO 13: POKE(32768+F-1+VL*40),C2
320 POKE(32768+F+14+VL*40),C2: NEXT
330 FOR F=14 TO 16: POKE(32768+F-1+VL*40),C3
340 POKE(32768+F+7+VL*40),C3: NEXT
350 FOR F=17 TO 18: POKE(32768+F-1+VL*40),C4
360 POKE(32768+F+2+VL*40),C4: NEXT
370 POKE(32768+18+VL*40),C5
380 NEXT VL
390 FOR H=1 TO 39: POKE(32768+H),C1: NEXT
400 RETURN
410 POKE(32768+X-1+800),32: Y=20
420 IF Y=0 GOTO 470
430 POKE(32768+X-1+(Y-1)*40),30
440 IF PEEK(32768+X-1+(Y-2)*40)<>32 GOTO 470
450 POKE(32768+X-1+(Y-1)*40),32
460 Y=Y-1: GOTO 420
470 N=N+1
480 GOSUB 1010
490 IF TURN=1 GOTO 530
500 IF TURN=0 GOTO 610
510 X=0
520 RETURN
530 IF N=4 THEN TURN=0
540 GOSUB 790
550 S2=S2+HIT
560 IF S2>210 THEN S2=S2-HIT
570 GOSUB 690
580 IF N=4 THEN GOSUB 280
590 IF N=4 THEN N=1
600 GOTO 510
610 IF N=4 THEN TURN=1
620 GOSUB 790
630 S1=S1+HIT
640 IF S1>210 THEN S1=S1-HIT
650 GOSUB 690
660 IF N=4 THEN GOSUB 280
670 IF N=4 THEN N=1
680 GOTO 510
690 W=22: GOSUB1020
700 PRINT N1$"=";S1 TAB(20)N2$"=";S2
710 IF S1=210 OR S2=210 THEN GOSUB 910
720 W=24: GOSUB 1020
730 FOR I=1 TO 39: PRINT " ";: NEXT I
740 W=24: GOSUB 1020
750 PRINT TAB(10)"TOCCA A ";:
760 IF TURN=0 THEN PRINT N1$
770 IF TURN=1 THEN PRINT N2$
780 RETURN
790 HIT=0
800 IY=1
810 IF Y=5 THEN IY=2
820 IF Y=6 THEN IY=3
830 IF Y=7 THEN IY=4
840 IF Y=8 THEN IY=5
850 IF PEEK(32768+X-1+(Y-IY)*40)=C1 THEN HIT=0
860 IF PEEK(32768+X-1+(Y-IY)*40)=C2 THEN HIT=10
870 IF PEEK(32768+X-1+(Y-IY)*40)=C3 THEN HIT=20
880 IF PEEK(32768+X-1+(Y-IY)*40)=C4 THEN HIT=30
890 IF PEEK(32768+X-1+(Y-IY)*40)=C5 THEN HIT=50
900 RETURN
910 FOR S=1 TO 50: GOSUB 1010: NEXT S
920 W=24: GOSUB 1020
930 FOR I=1 TO 39: PRINT" ";: NEXT I
940 W=24: GOSUB 1020
950 INPUT"VOLETE GIOCARE ANCORA";A$
960 IF S2=210 THEN TURN=0
970 IF S1=210 THEN TURN=1
980 IFLEFT$(A$,1)="S"THEN PRINT CHR$(14)
990 GOTO 180
1000 END
1010 RETURN
1020 PRINT CHR$(19);
1030 FOR I=1 TO W-1
1040 PRINT CHR$(17);
1050 NEXT I: RETURN
```

Tandy
Radio Shack

```
10 REM ****************************************
20 REM *                                      *
30 REM *              BERSAGLIO               *
40 REM *        VERSIONE TRS-80 MOD. I        *
50 REM *                                      *
60 REM *          PERSONAL SOFTWARE           *
70 REM *                                      *
80 REM ****************************************
90 REM
120 CLS: CLEAR 100
130 PRINT TAB(15)"B E R S A G L I O": PRINT: PRINT
140 INPUT
    "BATTI IL NOME DEL PRIMO GIOCATORE";N$(1)
150 INPUT
    "BATTI IL NOME DEL SECONDO GIOCATORE";N$(2)
160 TURN=1: S(1)=0: S(2)=0
170 GOSUB 1000
180 PRINT@64,STRING$(64," ");:
    PRINT@64,"TOCCA A ";N$(TURN);
190 FOR N=1 TO 1500: NEXT: FOR N=1 TO 3
210 GOSUB 1500: P(N)=X+64: HIT=0
220 IF X1>14 AND X1<53 THEN HIT=10
230 IF X1>23 AND X1<43 THEN HIT=20
240 IF X1>29 AND X1<37 THEN HIT=30
250 IF X1=33 THEN HIT=50
260 IF (S(TURN)+HIT) <= 210 THEN
    S(TURN)=S(TURN)+HIT
270 PRINT@0,N$(1);"=";S(1);"    ";N$(2);"=";S(2);
280 IF S(TURN)=210 GOTO 2000 ELSE NEXT
290 IF TURN=1 THEN TURN=2 ELSE TURN=1
300 FOR N=1 TO 3
310 PRINT@P(N),CHR$(32);
320 NEXT
330 GOTO 180
1000 CLS: FOR Y=7 TO 14: SET(66,Y): SET(67,Y): NEXT
1005 FOR X=0 TO 127: SET(X,6): NEXT
1010 FOR Y=7 TO 13 STEP 2
1020 FOR X=60 TO 64 STEP 2: SET(X,Y): SET(X+1,Y+1):
     NEXT X
1030 FOR X=69 TO 73 STEP 2: SET(X,Y): SET(X-1,Y+1):
     NEXT X
1040 NEXT Y
1050 FOR Y=8 TO 14 STEP 2
1060 FOR X=49 TO 59 STEP 2: SET(X,Y): NEXT X
1070 FOR X=74 TO 84 STEP 2: SET(X,Y): NEXT X
1080 NEXT Y
1090 FOR Y=7 TO 13 STEP 2
1100 FOR X=30 TO 48 STEP 4
1110 SET(X+2,Y): SET(X,Y+1): NEXT X
1120 FOR X=86 TO 104 STEP 4
1130 SET(X,Y): SET(X+2,Y+1): NEXT X
1140 NEXT Y
1150 RETURN
1500 FOR X=16320 TO 16382
1510 POKE X+1,91: POKE X,32
1520 IF INKEY$=CHR$(91) GOTO 1550 ELSE NEXT
1530 POKE 16383,32
1540 FOR X=1 TO 500: NEXT: GOTO 1500
1550 X=X+1: X1=X-16320: POKE X,32: X=16382: NEXT
1560 X=896+X1
1570 IF PEEK(15360+X)<>32 THEN RETURN
1580 PRINT@X,CHR$(91);: PRINT@X+64,CHR$(32);
1590 X=X-64: GOTO 1570
2000 CLS
2010 PRINT@128,"HA VINTO ";N$(TURN)
2020 PRINT@320,;: INPUT"VOLETE GIOCARE ANCORA";A$
2030 IF LEFT$(A$,1)="S" THEN GOTO 160
```

Orologio digitale

Questo programma simula un orologio digitale, visualizzando sullo schermo le ore, i minuti ed i secondi in grande formato.

Nella riga 100 si inizializzano gli indirizzi relativi al modo musicale. Il modo musicale si abilita infatti con POKE 59467,16; con POKE 59466,Y si decide l'intervallo di frequenza e con POKE 59464,X si decide la frequenza (X e Y variano da 1 a 255). Per chiudere il modo musicale si devono ripetere le tre POKE con argomento 0.

Il modo musicale viene azionato al cambio dell'ora (per esempio dalle 08:59:59 alle 09:00:00) nella riga 260.

L'istruzione F%=FRE(0) della riga 980 serve a compattare le stringhe utilizzate e ad eliminare quelle che non servono più: ogni volta infatti che si trova una variabile stringa, sia essa già definita stringa, sia essa già definita oppure no, essa viene memorizzata. Ciò comporta uno spreco di memoria, occupata da stringhe ridefinite e, pertanto, inutili. Nel nostro caso la variabile alfanumerica T$. L'operazione di compattamento è eseguita automaticamente dal sistema operativo quando non c'è più memoria disponibile. In tal caso, tuttavia l'esecuzione di questa istruzione non può essere relativamente lenta, con un aumento del tempo di esecuzione del programma. Richiamandola periodicamente, si ottimizza l'occupazione di memoria e si migliora la velocità di esecuzione. ■

commodore

```
10 REM ****************************************
20 REM *                                      *
30 REM *          OROLOGIO DIGITALE           *
40 REM *     VERSIONE PET/CBM 3032 E 4032     *
50 REM *                                      *
60 REM *          PERSONAL SOFTWARE           *
70 REM *                                      *
80 REM ****************************************
90 REM
100 U=59464: V=59466: Z=59467
110 A=32314: B=32200: C$=CHR$(209)
120 D$=CHR$(145): D$=D$+D$+D$+D$+D$+D$+D$
122 PRINT CHR$(147)
130 INPUT"BATTI L'ORA (ES. 234158): ";T$
140 IF LEN(T$)  6 GOTO 130
150 PRINT CHR$(147): H$=T$
160 FOR I=0 TO 6: PRINT: NEXT
170 POKE A,87: POKE A-120,87
180 POKE B,87: POKE B-120,87
190 GOSUB 900: T=74
200 D=S: GOSUB 280: T=T-6
210 D=D2: GOSUB 280: T=T-8
220 D=M: GOSUB 280: T=T-6
230 D=DM: GOSUB 280: T=T-8
240 D=H: GOSUB 280: T=T-6
250 D=DH: GOSUB 280
260 IF S=0 AND D2=0 AND M=0 AND DM=0 GOTO 1020
270 GOTO 190
280 ON D GOTO 320,340,410,480,550,620,690,760,830
290 PRINT TAB(T)" "C$C$C$" "
300 FOR I=1 TO 5: PRINT TAB(T)C$"    "C$: NEXT I
310 PRINT TAB(T)" "C$C$C$D$: RETURN
320 FOR I=1 TO 6: PRINT TAB(T)"     "C$: NEXT I
330 PRINT TAB(T)"     "C$D$: RETURN
340 PRINT TAB(T)" "C$C$C$" "
350 PRINT TAB(T)C$"    "C$
360 PRINT TAB(T)"     "C$
370 PRINT TAB(T)"    "C$" "
380 PRINT TAB(T)"   "C$"   "
390 PRINT TAB(T)" "C$"    "
400 PRINT TAB(T)C$C$C$C$C$D$: RETURN
410 PRINT TAB(T)" "C$C$C$" "
420 PRINT TAB(T)C$"    "C$
430 PRINT TAB(T)"     "C$
440 PRINT TAB(T)"  "C$C$" "
450 PRINT TAB(T)"     "C$
460 PRINT TAB(T)C$"    "C$
470 PRINT TAB(T)" "C$C$C$" "D$: RETURN
480 PRINT TAB(T)"    "C$" "
490 PRINT TAB(T)"  "C$C$" "
500 PRINT TAB(T)" "C$" "C$" "
510 PRINT TAB(T)C$"  "C$" "
520 PRINT TAB(T)C$C$C$C$C$
530 PRINT TAB(T)"    "C$" "
540 PRINT TAB(T)"    "C$" "D$: RETURN
550 PRINT TAB(T)C$C$C$C$C$
560 PRINT TAB(T)C$"      "
570 PRINT TAB(T)C$C$C$C$" "
580 PRINT TAB(T)"     "C$
590 PRINT TAB(T)"     "C$
600 PRINT TAB(T)C$"    "C$
610 PRINT TAB(T)" "C$C$C$" "D$:RETURN
620 PRINT TAB(T)"   "C$C$C$
630 PRINT TAB(T)" "C$"    "
640 PRINT TAB(T)C$"    "
650 PRINT TAB(T)C$C$C$C$C$" "
660 PRINT TAB(T)C$"    "C$
670 PRINT TAB(T)C$"    "C$
680 PRINT TAB(T)" "C$C$C$" "D$: RETURN
690 PRINT TAB(T)C$C$C$C$C$C$
700 PRINT TAB(T)C$"    "C$
710 PRINT TAB(T)"    "C$" "
720 PRINT TAB(T)"   "C$"   "
730 PRINT TAB(T)"    "C$"    "
740 PRINT TAB(T)" "C$"    "
750 PRINT TAB(T)" "C$"    "D$: RETURN
760 PRINT TAB(T)" "C$C$C$" "
770 PRINT TAB(T)C$"    "C$
780 PRINT TAB(T)C$"    "C$
790 PRINT TAB(T)" "C$C$C$" "
800 PRINT TAB(T)C$"    "C$
810 PRINT TAB(T)C$"    "C$
```

```
820 PRINT TAB(T)" "C$C$C$" "D$: RETURN
830 PRINT TAB(T)" "C$C$C$" "
840 PRINT TAB(T)C$"    "C$
850 PRINT TAB(T)C$"    "C$
860 PRINT TAB(T)" "C$C$C$C$
870 PRINT TAB(T)"     "C$
880 PRINT TAB(T)"    "C$" "
890 PRINT TAB(T)C$C$C$"  "D$: RETURN
900 IF T%=TI$ GOTO 900
910 T%=TI$: S=VAL(MID$(T$,6,1))
920 D2=VAL(MID$(T$,5,1))
930 M=VAL(MID$(T$,4,1))
940 DM=VAL(MID$(T$,3,1))
950 H=VAL(MID$(T$,2,1))
960 DH=VAL(MID$(T$,1,1))
970 IF DH*10+H=24 THEN H=0: DH=0
980 F%=FRE(0): RETURN
990 POKE Z,16: POKE V,45: POKE U,0: RETURN
1000 POKE Z,0: POKE V,0: POKE U,0: RETURN
1010 FOR T1=0 TO DU: NEXT: RETURN
1020 GOSUB 990: FOR T=1 TO 36: READ F,DU
1030 POKE U,F: GOSUB 1010: NEXT: H=DH*10+H
1040 IF H<=12 AND H<>0 THEN H1=H: GOTO 1060
1050 H1=ABS(H-12)
1060 FOR II=1 TO H1: POKE U,32: GOSUB 1010
1070 POKE U,0: GOSUB 1010: NEXT: GOSUB 1000
1080 RESTORE: GOTO 190
1090 REM    F     D     F     D     F     D
1100 DATA 126,100,112,100,126,100
1110 DATA 142,100,150,100,142,100
1120 DATA 126,100,000,100,169,100
1130 DATA 150,100,142,100,000,100
1140 DATA 150,100,142,100,126,100
1150 DATA 000,100,126,100,112,100
1160 DATA 126,100,142,100,150,100
1170 DATA 142,100,126,100,142,100
1180 DATA 150,100,142,100,126,100
1190 DATA 000,100,169,100,000,100
1200 DATA 126,100,000,100,150,100
1210 DATA 190,100,126,400,000,250
```

Colpo strategico

Colpo strategico è un gioco che richiede logica e memoria. I giocatori tentano di catturare i pezzi dell'avversario muovendo un'armata di 40 pezzi su una scacchiera, in avanti, all'indietro e lateralmente.

Generalmente sono necessari da 30 minuti a un'ora per fare una partita. Man mano che i giocatori acquistano familiarità con il gioco, svilupperanno le loro strategie di posizionamento dell'armata e movimento.

Istruzioni

Obiettivo del gioco è di muovere dei pezzi attraverso un campo di battaglia per catturare la bandiera dell'avversario. Ogni giocatore comincia con una armata di 40 pezzi composta da:

NUM. PEZZI	NOME	GRADO
1	Maresciallo	1
1	Generale	2
2	Colonnello	3
3	Maggiore	4
4	Capitano	5
4	Tenente	6
4	Sergente	7
5	Artificiere	8
8	Scout	9
1	Spia	10
6	Bomba	non eliminabile
1	Bandiera	non eliminabile

Questi pezzi sono in ordine di grado militare: il maresciallo ha il grado più alto, la spia il più basso. Tuttavia, questo è l'unico pezzo che può eliminare il maresciallo dalla tastiera. Le bombe sono pezzi non eliminabili che possono eliminare qualunque pezzo che tenti di "colpirle", escluso l'artificiere che può disinnescarle ed eliminarle. Il gioco si effettua su una scacchiera 10 × 10, che viene continuamente visualizzata sullo schermo durante il gioco. Il primo giocatore posiziona i pezzi sulle quattro righe superiori della scacchiera, il secondo giocatore sulle quattro righe inferiori. Le due righe centrali sono vuote all'inizio del gioco. Quando i due giocatori hanno posizionato il loro esercito, hanno la possibilità di fare qualunque spostamento.

Posizionamento

Il primo giocatore posiziona i pezzi sulle righe 1-4 (la riga 1 è quella posteriore, la 4 quella anteriore). Per posizionare uno dei pezzi eliminabili (eccetto la spia) basta battere il grado, e il quadrato lampeggiante conterrà quel pezzo. Per posizionare una spia, una bomba o una bandiera, battere S, B o F rispettivamente. Il secondo giocatore posiziona i pezzi sulle righe 7-10 (la riga 7 è quella anteriore e la riga 10 quella posteriore).

Mosse

1. I pezzi si possono muovere solo uno spazio alla volta: in avanti, all'indietro o di lato. Non sono possibili mosse diagonali. I giocatori battono le coordinate del pezzo che vogliono spostare. Le coordinate X e Y di un pezzo vengono misurate verticalmente e quindi orizzontalmente (X = verticale, Y = orizzontale).

I pezzi vengono rivelati solo quando è il turno di quel giocatore. L'avversario non può guardare lo schermo quando un giocatore si muove.

2. I due pezzi non possono muoversi al centro della scacchiera.

3. Due pezzi non possono occupare lo stesso spazio, e non sono permessi salti.

4. La bandiera e la bomba non possono essere mosse.

L'unico momento in cui i due giocatori possono guardare lo schermo contemporaneamente è quando un pezzo di un giocatore attacca un altro.

Regole di attacco

1. Non è possibile un attacco in diagonale ma solo sui quattro lati del quadrato adiacente.

2. Un giocatore non può attaccare e muovere allo stesso tempo.

3. Per attaccare seguire le seguenti regole. Quando il computer visualizza DA? Il giocatore batte le coordinate della posizione attuale. Il computer quindi visualizza A? e il giocatore batte le coordinate del pezzo che vuole attaccare (assicurandosi che sia in uno dei quadrati vicini). Il computer risponde CORRETTO? e se si risponde SI il computer controlla che l'attacco sia corretto. Se si risponde NO si ritorna a DA?

4. Il pezzo con il grado minore viene eliminato dalla scacchiera.

5. Se i gradi sono uguali, tutti e due i pezzi sono eliminati dalla scacchiera.

6. Una spia può eliminare un maresciallo se essa

attacca per prima, invece se il maresciallo attacca per primo accade il contrario. Un artificiere può eliminare una bomba.

7. Le bombe non si possono spostare.

Particolarità

1. Se per qualche ragione la visualizzazione dello schermo viene disturbata eventualmente da qualche input sbagliato, un giocatore può battere le coordinate 0,0 sia in risposta a DA? che in risposta a A?

2. Se un giocatore non può muovere al suo turno o desidera passare, basta battere −1, −1 in risposta a DA?

3. Se un giocatore sbaglia a battere le coordinate in risposta a DA? può battere −1, −1 in risposta a A?

4. Quando un giocatore ha introdotto le coordinate iniziali e finali, il computer chiede CORRETTO? e si può rispondere SI o NO. ■

Tandy
Radio Shack

```
10 REM ******************************************
20 REM *                                        *
30 REM *            COLPO STRATEGICO            *
40 REM *         VERSIONE TRS-80 MOD. I         *
50 REM *                                        *
60 REM *          PERSONAL SOFTWARE             *
70 REM *                                        *
80 REM ******************************************
90 REM
100 CLS: DEFINT A-Z: CLEAR 250: PL=1: Z=125
110 DIM SP(2): SP(1)=314: SP(2)=698
120 DIM A(20,10),C(20,10),NU(12),P1(12)
130 FOR T=1 TO 12: READ NU(T): NEXT T
140 DATA 1,1,2,3,4,4,4,5,8,1,1,6
150 PRINT CHR$(23);
160 PRINT@464, "COLPO STRATEGICO";
170 DD$=STRING$(7,191)+CHR$(26)+STRING$(7,24)
180 DE$=DD$+DD$+DD$+STRING$(7,191)
190 ED$=STRING$(7,131)+CHR$(26)+STRING$(7,24)+
    CHR$(199)+CHR$(26)+STRING$(7,24)+CHR$(199)+
    CHR$(26)+STRING$(7,24)+STRING$(7,176)
200 DIM B$(12): B$(1)=CHR$(131)+CHR$(191)+
    CHR$(26)+STRING$(2,24)+CHR$(176)+CHR$(191)+
    CHR$(176)
210 B$(2)=STRING$(2,131)+CHR$(191)+CHR$(26)+
    STRING$(3,24)+CHR$(191)+STRING$(2,179)
220 B$(3)=STRING$(2,131)+CHR$(191)+CHR$(26)+
    STRING$(3,24)+STRING$(2,179)+CHR$(191)
230 B$(4)=CHR$(191)+" "+CHR$(191)+CHR$(26)+
    STRING$(3,24)+STRING$(2,131)+CHR$(191)
240 B$(5)=CHR$(191)+STRING$(2,131)+CHR$(26)+
    STRING$(3,24)+STRING$(2,179)+CHR$(191)
250 B$(6)=CHR$(191)+STRING$(2,131)+CHR$(26)+
    STRING$(3,24)+CHR$(191)+CHR$(179)+CHR$(191)
260 B$(7)=CHR$(143)+CHR$(131)+CHR$(191)+CHR$(26)
    +CHR$(24)+CHR$(191)
270 B$(8)=CHR$(191)+CHR$(179)+CHR$(191)+CHR$(24)
    +STRING$(3,24)+CHR$(191)+CHR$(176)+CHR$(191)
280 B$(9)=CHR$(191)+CHR$(179)+CHR$(191)+CHR$(26)
    +CHR$(24)+CHR$(191)
290 B$(11)=CHR$(191)+STRING$(2,131)+CHR$(26)+
    STRING$(3,24)+CHR$(191)+CHR$(131)
300 B$(10)=CHR$(166)+CHR$(131)+CHR$(132)+
    CHR$(26)+STRING$(3,24)+CHR$(164)+CHR$(179)+
    CHR$(132)
310 B$(12)=CHR$(136)+CHR$(147)+CHR$(26)+
    STRING$(2,24)+CHR$(190)+CHR$(191)+CHR$(189)
320 FOR X=1 TO 10: FOR Y=1 TO 10
330 E(X,Y)=125+Y*5+X*64: NEXT Y,X
340 NL$="12345678901234567891234567"
350 E9=VARPTR(NL$): E8=PEEK(E9+1)+256*PEEK(E9+2)
360 FOR E7=E8 TO E8+26: READ E6: POKE E7,E6
370 NEXT
380 DATA 205,127,10,77,68,62,1,105,211,255,45,32
390 DATA 253,60,105,211,255,45,32,253,13,16,238
400 DATA 175,211,255,201
410 IF PEEK(16396)=201 POKE 16526,PEEK(E9+1):
    POKE 16527,PEEK(E9+2): GOTO 440
420 CMD"T": DEFUSR0=PEEK(E9+1)+256*PEEK(E9+2)
430 POKE 14308,0
440 FOR X=1 TO 4: FOR Y=1 TO 10
450 A(X,Y)=1: A(X+16,Y)=1: NEXT Y,X
460 GOSUB 470: GOTO 930
470 CLS: FOR I=15488 TO 15538: POKE I,176
480 U=USR(800): NEXT
490 FOR I=192 TO 768 STEP 64
500 PRINT@I, "";: FOR II=1 TO 10
510 PRINT CHR$(191); CHR$(196);: U=USR(850)
520 NEXT II: PRINT CHR$(191): NEXT
530 FOR I=16192 TO 16242: POKE I,131
540 U=USR(800): NEXT
550 FOR I=15819 TO 15828: POKE I,191
560 POKE I+64,191: POKE I+20,191: POKE I+84,191
570 U=USR(625): NEXT
580 POKE 15819,184: POKE 15839,184
590 POKE 15820,190: POKE 15840,190
600 POKE 15826,188: POKE 15846,188
610 POKE 15827,176: POKE 15847,176
620 POKE 15887,131: POKE 15884,143
630 POKE 15903,131: POKE 15904,143
640 POKE 15890,143: POKE 15891,131
650 POKE 15910,143: POKE 15911,131
660 T=1: FOR I=897 TO 942 STEP 5
670 PRINT@I, T;: T=T+1: U=USR(1632): NEXT: T=1
680 FOR I=243 TO 819 STEP 64: PRINT@I, T;
690 T=T+1: U=USR(1632): NEXT
700 PP=1: GOSUB 760: PP=2: GOSUB 760: GOTO 810
710 FOR X=1+(PL-1)*10 TO 10+(PL-1)*10
720 FOR Y=1 TO 10
730 IF A(X,Y)=0 GOTO 750 ELSE IF C(X,Y)=10
    THEN PRINT@E(X-AD,Y), " S"; ELSE IF C(X,Y)=
    11 THEN PRINT@E(X-AD,Y), " F"; ELSE IF
    C(X,Y)=12 THEN PRINT@E(X-AD,Y), " #"; ELSE
    PRINT@E(X-AD,Y), C(X,Y);
740 U=USR(1300)
750 NEXT Y,X: RETURN
760 FOR X=1 TO 10: FOR Y=1 TO 10
770 IF A(X+(PP-1)*10,Y)=0 GOTO 800
780 PRINT@E(X,Y), STRING$(2,140);
790 U=USR(1320)
800 NEXT Y,X: RETURN
810 FOR I=15607 TO 15615: POKE I,131
820 POKE I+192,176: POKE I+384,131
830 POKE I+576,176: NEXT
840 FOR I=15607 TO 15799 STEP 64: POKE I,191
850 POKE I+8,191: POKE I+384,191
860 POKE I+392,191: U=USR(8020): NEXT
870 PRINT@183, "NUMERO 1";
880 PRINT@887, "NUMERO 2";: RETURN
890 FOR I=1 TO 500: NEXT: RETURN
900 PRINT@0, CHR$(30);: RETURN
910 FOR I=1 TO 1200: NEXT
920 PRINT@0, CHR$(30);: RETURN
930 FOR PL=1 TO 2: FOR XX=1 TO 12
940 P1(XX)=0: NEXT XX: AD=(PL-1)*10
950 PRINT@0,"POSIZIONAMENTO GIOCATORE"PL;
960 GOSUB 910
970 FOR X=1+(PL-1)*16 TO 4+(PL-1)*16
980 FOR Y=1 TO 10
990 PRINT@0,"CHE PEZZO VUOI IN QUESTO POSTO";
1000 A$=INKEY$
1010 IF A$="S" THEN A=10: GOTO 1060
1020 IF A$="B" THEN A=12: GOTO 1060
1030 IF A$="F" THEN A=11: GOTO 1060
1040 A=VAL(A$)
```

```
1050 IF A=0 THEN PRINT@E(X-AD,Y),"  "::
     PRINT@E(X-AD,Y),STRING$(2,140)::
     GOTO 1000
1060 P=PI(A): IF P=NU(A) GOTO 990
1070 PI(A)=PI(A)+1
1080 IF A$="B" THEN A$="*"
1090 PRINT@E(X-AD,Y)," ";A$;: C(X,Y)=A
1100 U=USR(B020): FOR II=1 TO 10: NEXT II
1110 NEXT Y,X
1120 P=0: P1=0: P2=0: P3=0: P4=0: P5=0: P6=0
1130 P7=0: P8=0: P9=0: P10=0: P11=0: P12=0: PF=0: PP=PL
1140 GOSUB 760
1150 GOSUB 900: NEXT PL
1160 FOR PL=1 TO 2: AD=(PL-1)*10: GOSUB 900
1170 PRINT@0,"GIOCATORE";3-PL;"GIRATI!"
1180 GOSUB 910: GOSUB 710
1190 GOSUB 900: PRINT@0,"VUOI FARE CAMBIAMENTI";
1200 INPUT RR$: IF LEFT$(RR$,1)="N" GOTO 1330
1210 IF LEFT$(RR$,1)<>"S" GOTO 1190
1220 GOSUB 900: PRINT@0,"DA":: INPUT U1,U2
1230 IF U1<1 OR U1>10 OR U1<>INT(U1) GOTO 1220
1240 IF U2<1 OR U2>10 OR U2<>INT(U2) GOTO 1220
1250 U1=U1+(PL-1)*10: IF A(U1,U2)<>1 GOTO 1220
1260 PRINT@13,CHR$(30);: PRINT@13,"A";
1270 INPUT U3,U4
1280 IF U3<1 OR U3>10 OR U3<>INT(U3) GOTO 1260
1290 IF U4<1 OR U4>10 OR U4<>INT(U4) GOTO 1260
1300 U3=U3+(PL-1)*10: IF A(U3,U4)<>1 GOTO 1260
1310 U5=C(U1,U2): C(U1,U2)=C(U3,U4): C(U3,U4)=U5
1320 GOSUB 2520: GOTO 1190
1330 PP=PL: GOSUB 760: NEXT PL
1340 FOR PL=1 TO 2: AD=(PL-1)*10: GOSUB 900
1350 PRINT@0,"TOCCA AL GIOCATRE";PL;
1360 PRINT"(GIOCATORE";3-PL;"GIRATI!)"
1370 GOSUB 910: GOSUB 710
1380 GOSUB 900: PRINT@0,"DA:":: INPUT C1,C2
1390 IF C1=0 AND C2=0 GOTO 1430
1400 IF C1=-1 AND C2=-1 GOTO 2450
1410 IF C1<1 OR C1>10 OR C1<>INT(C1) GOTO 1380
1420 GOTO 1440
1430 CLS: GOSUB 470: GOSUB 710: GOTO 1380
1440 IF C2<1 OR C2>10 OR C2<>INT(C2) GOTO 1380
1450 C1=C1+(PL-1)*10: IF C(C1,C2)=11 GOTO 1380
1460 IF C(C1,C2)=12 GOTO 1380
1470 IF A(C1,C2)=0 GOTO 1380
1480 IF A(C1-AD*2+10,C2)=1 GOTO 1380
1490 PRINT@14,CHR$(210):: PRINT@14,"A";
1500 INPUT C3,C4
1510 IF C3=0 AND C4=0 GOTO 1600
1520 IF C3=-1 AND C4=-1 GOTO 1380
1530 IF C3<1 OR C3>10 OR C3<>INT(C3) GOTO 1490
1540 IF C4<1 OR C4>10 OR C4<>INT(C4) GOTO 1490
1550 C3=C3+(PL-1)*10
1560 Y6=3*INT(E(C3-AD,C4)/64)
1570 X6=2*(E(C3-AD,C4)-INT(Y6/3)*64)
1580 IF POINT(X6,Y6) GOTO 1490
1590 IF C3=C1 AND C4=C2 GOTO 1490 ELSE 1620
1600 CLS: GOSUB 470: GOSUB 710
1610 PRINT@0,"DA";C1-(PL-1)*10;",";C2;: GOTO 1490
1620 IF A(C3,C4)=1 GOTO 1490
1630 IF C3<>C1 AND C4<>C2 GOTO 1490
1640 IF C3<>C1 GOTO 1660
1650 IF C4=C2+1 OR C4=C2-1 GOTO 1670 ELSE 1490
1660 IF C3<>C1+1 AND C3<>C1-1 GOTO 1490
1670 PRINT@35,CHR$(30):: PRINT@35,"GIUSTO";
1680 INPUT RR$: IF LEFT$(RR$,1)="N" GOTO 1380
1690 IF LEFT$(RR$,1)<>"S" GOTO 1670
1700 IF A(C3-AD*2+10,C4)=1 GOTO 1810
1710 FOR I=1 TO 10
1720 IF C(C1,C2)=10 THEN PRINT@E(C3-AD,C4)," S":
     ELSE PRINT@E(C1-AD,C2),C(C1,C2):
1730 GOSUB 2460: PRINT@E(C1-AD,C2),"  ";
1740 GOSUB 2460: NEXT I
1750 A(C1,C2)=0: A(C3,C4)=1
1760 C(C3,C4)=C(C1,C2): C(C1,C2)=0
1770 IF C(C3,C4)=10 THEN PRINT@E(C3-AD,C4)," S";
     ELSE PRINT@E(C3-AD,C4),C(C3,C4);
1780 FOR I=1 TO 10: U=USR(B000): U=USR(25700)
1790 NEXT I: GOSUB 890
1800 PP=PL: GOSUB 760: GOTO 2390
1810 PP=PL: GOSUB 760
1820 IF C(C1,C2)=10 THEN PRINT@E(C1-AD,C2)," S";
     ELSE PRINT@E(C1-AD,C2),C(C1,C2);
1830 GOSUB 900
1840 PRINT@0,"GIOCATORE";3-PL;"GUARDA!!";
1850 GOSUB 910: PRINT@SP(PL),B$(C(C1,C2));
1860 PRINT@SP(3-PL),B$(C(C3-AD*2+10,C4));
1870 GOSUB 890
1880 IF C(C3-AD*2+10,C4)=11 GOTO 2410
1890 IF C(C3-AD*2+10,C4)=12 GOTO 2230
1900 IF C(C3-AD*2+10,C4)<>1 GOTO 1920
1910 IF C(C1,C2)=10 GOTO 1940 ELSE 2030
1920 IF C(C1,C2)=C(C3-AD*2+10,C4) GOTO 2140
1930 IF C(C1,C2)>C(C3-AD*2+10,C4) GOTO 2030
1940 FOR I=1 TO 10: PRINT@SP(3-PL)-66,DE$;
1950 U=USR(9000): GOSUB 2400: PRINT@SP(3-PL)-66,
     ED$;: U=USR(12350): GOSUB 2400: NEXT I
1960 PRINT@0,"GIOCATORE";PL;"VINCE";
1970 PRINT@E(C3-AD,C4),"  ";: GOSUB 910
1980 PRINT@SP(PL)-66,ED$;
1990 A(C3-AD*2+10,C4)=0: C(C3,C4)=C(C1,C2)
2000 PRINT@E(C3-AD,C4),STRING$(2,140);
2010 PRINT@E(C1-AD,C2),"  ";
2020 A(C1,C2)=0: A(C3,C4)=1: GOTO 2390
2030 FOR I=1 TO 10: PRINT@SP(PL)-66,DE$;
2040 U=USR(9000): GOSUB 2400
2050 PRINT@SP(PL)-66,ED$;: U=USR(12350)
2060 GOSUB 2400: NEXT I
2070 PRINT@0,"GIOCATORE";3-PL;"VINCE!!";
2080 PRINT@E(C1-AD,C2),"  ";: GOSUB 910
2090 PRINT@SP(3-PL)-66,ED$;
2100 A(C1-AD*2+10,C2)=0: C(C1,C2)=C(C3,C4)
2130 A(C1,C2)=0: GOTO 2390
2140 FOR I=1 TO 10: PRINT@248,DE$;
2150 PRINT@632,DE$;: U=USR(9000): GOSUB 2400
2160 PRINT@248,ED$;: PRINT@632,ED$;
2170 U=USR(12350): GOSUB 2400: NEXT I
2180 PRINT@0,
     "I GOCATORI SI DISTRUGGONO A VICENDA.";
2190 PRINT@E(C1-AD,C2),"  ";
2200 PRINT@E(C3-AD,C4),"  ";: GOSUB 910
2210 A(C1,C2)=0: C(C1,C2)=0: A(C3-AD*2+10,C4)=0
2220 C(C3-AD*2+10,C4)=0: GOTO 2390
2230 IF C(C1,C2)=8 GOTO 2310
2240 FOR I=1 TO 140: PRINT CHR$(23);
2250 U=USR(400): NEXT I: PRINT CHR$(28);
2260 PRINT@0,"IL PEZZO DEL GIOCTORE";PL;
2270 PRINT "E' SALTATO IN ARIA.";
2280 PRINT@E(C1-AD,C2),"  ";: GOSUB 910
2290 A(C1,C2)=0: C(C1,C2)=0: PRINT@248,ED$;
2300 PRINT@632,ED$;: GOTO 2390
2310 U=USR(0): PRINT@0,"IL GIOCATORE";PL;
2320 PRINT "HA NEUTRALIZZATO LA BOMBA.";
2330 PRINT@E(C3-AD,C4),"  ";: GOSUB 910
2340 PRINT@E(C1-AD,C2),"  ";
2350 A(C3-AD*2+10,C4)=0: A(C3,C4)=1
2360 C(C3,C4)=C(C1,C2): C(C1,C2)=0: A(C1,C2)=0
2370 PRINT@E(C3-AD,C4),STRING$(2,140);
2380 PRINT@248,ED$;: PRINT@632,ED$;
2390 NEXT PL: GOTO 1340
2400 FOR L=1 TO 20: NEXT L: RETURN
2410 CLS: FOR I=1 TO 20
2420 PRINT@486,"IL GIOCATORE";PL;"HA VINTO";
2430 U=USR(24910): PRINT@486,CHR$(215);
2440 U=USR(24900): NEXT: GOTO 2480
2450 PL=3-PL: GOTO 2410
2460 FOR P1=1 TO 20: NEXT P1
2470 U=USR(5470): RETURN
2480 CLS: PRINT CHR$(23)
2490 PRINT@516,"BATTI ENTER PER GIOCARE ANCORA";
2500 IF PEEK(14400)=1 THEN RUN
2510 GOSUB 2460: GOTO 2480
2520 IF C(U1,U2)=10 THEN PRINT@E(U1-AD,U2)," S";
2530 IF C(U1,U2)=11 THEN PRINT@E(U1-AD,U2)," F";
2540 IF C(U1,U2)=12 THEN PRINT@E(U1-AD,U2)," *";
2550 IF C(U1,U2)<10 THEN
     PRINT@E(U1-AD,U2),C(U1,U2);
2560 IF C(U3,U4)=10 THEN PRINT@E(U3-AD,U4)," S";
2570 IF C(U3,U4)=11 THEN PRINT@E(U3-AD,U4)," F";
2580 IF C(U3,U4)=12 THEN PRINT@E(U3-AD,U4)," *";
2590 IF C(U3,U4)<10 THEN
     PRINT@E(U3-AD,U4),C(U3,U4);
2600 RETURN
```

File comandi

Questo programma crea un file di comandi che, quando eseguito, intratterà un colloquio con il sistema operativo, come se i comandi provenissero dalla tastiera, e il computer agirà conseguentemente.

È molto raro che si trovi una segretaria felice nell'avere un computer in ufficio. Ciò è dovuto, in parte, al gran numero di informazioni, apparentemente inutili, che bisogna introdurre nel computer prima che questo cominci a fare qualcosa di interessante. Parliamo di cose come BASIC, HOW MANY FILES?, MEM SIZE? e RUN. Il problema è che l'utente generalmente non comprende l'importanza di queste "banali" informazioni. Bene, questo programma è scritto per introdurre tutte queste informazioni all'accensione del computer, e liberare la segretaria da questo fastidioso lavoro.

Questo è un programma in Disk Basic e non funziona in Basic livello II. Assicuratevi di lasciare liberi almeno 256 byte di memoria usando l'opzione MEM SIZE.

Variabili

EX$ Contiene dei comandi in linguaggio macchina che, quando eseguiti, scrivono il nuovo file comandi su disco con i seguenti comandi:

CMD "S"

*DUMP Nomefile (START=X"####", END=X"####", TRA=X"####").

IX$ Contiene gli input da tastiera da inserire nel file comandi

$ Contiene ogni singolo input da tastiera

ST Inizio decimale del codice macchina

H1$ Inizio esadecimale del codice macchina

Z Fine decimale del codice macchina

Z$ Fine esadecimale del codice macchina

V$ Nome del file

K Distanza dall'indirizzo di inizio

AD Indirizzo di POKE (=ST+K)

V È un indirizzo da convertire nel byte più significativo (BPS) e byte meno significativo (BMS)

V1 BPS di V (usato anche nel calcolo di V3)

V2 BMS di V (usato anche nel calcolo di V3)

H$ Scala numerica esadecimale

K(1)-K(4) Usati per la conversione esadecimale-decimale

B Puntatore alla locazione di EX$

B4 Locazione di EX$

B1-B3 Valori temporanei usati nel calcolo di B4. B1 è usato anche come indirizzo da convertire in PBS e BMS come per V, ma da usare in EX$

C1 BPS di B1

C2 BMS di B1

X$ Valore da inserire in EX$

Il computer richiederà un indirizzo di inizio in esadecimale. Questo indirizzo deve essere nella memoria protetta (più grande del MEM SIZE indicato in precedenza) e deve essere minore o uguale a:

7F00 per le macchine 16 K
BF00 per le macchine 32 K
FF00 per le macchine 48 K

(sottraete 50 byte se pensate di usare il comando BASIC* al prossimo ingresso in Basic).

Infine il computer vi chiederà il nome del file. Se non date un'estensione, verrà usata l'estensione /CMD (la migliore per una esecuzione veloce).

Ora il programma crea un file eseguibile in linguaggio macchina per eseguire questi comandi e trasferirvi in DOS. Per eseguire il file di comandi basta batterne il nome.

Un esempio.

Per scrivere diversi programmi su nastro

```
BASIC
1
48000
CMD"T"
LOAD"PROG1"
CSAVE"A"
LOAD"PROG2"
CSAVE"B"
LOAD"PROG3"
CSAVE"C"
CMD"R"
END
```

Per listare diversi programmi sulla stampante usare una procedura simile, ma con LPRINT.

Routine

130-280

Gli input di tastiera sono posti nella variabile I$; ogni input termina con un ritorno cursore (ENTER) seguito da un carattere NULL (CHR$(0)).

Alla fine della lista viene aggiunto un End of File, CHR$(225).

300-620
Mediante le routine 1400, 1480, 1530 viene messa in memoria una routine in linguaggio macchina, ed aggiunta alla variabile EX$.

630-690
Aggiunge input di tastiera al codice

1120-1180
Istruzione DATA contenenti codice in linguaggio macchina.

1200
Se l'indirizzo di memoria è maggiore di 32767, viene convertito negli appropriati valori negativi per PEEK e POKE.

1220
Converte l'indirizzo B1 nel BMS (C2) e BPS (C1).

1250
Routine d'errore nel caso che la posizione di EX$ sia cambiata (normalmente non dovrebbe capitare).

1260
Fornisce l'indirizzo di inizio e lo converte in decimale.

1370
Preleva i contenuti delle locazioni 40B1H e 40B2H e li usa per costruire V3 (fine della memoria protetta).

1400
Converte l'indirizzo V nel BMS V2 e BPS V1. Questi numeri vengono messi in memoria e aggiunti a EX$ dalla linea 6000. Ciò completa la parte dipendente dalla locazione del linguaggio macchina.

1480
Legge istruzioni in linguaggio macchina e usa la linea 1530 per metterle in memoria e aggiungerle a EX$.

1530
Mette in memoria ASC(X$) e aggiunge X$ a EX$.

1600
Conversione da decimale a esadecimale (per il dump). ■

Tandy Radio Shack

```
10 REM ******************************************
20 REM *                                        *
30 REM *                FILE COMANDI            *
40 REM *          VERSIONE TRS-80 MOD. I        *
50 REM *                                        *
60 REM *            PERSONAL SOFTWARE           *
70 REM *                                        *
80 REM ******************************************
90 REM
100 CLEAR 1000
110 H$="0123456789ABCDEF"
120 K=0
130 CLS: I$=""
140 PRINT TAB(20)"FILE COMANDI"
150 PRINT"
      SCRIVI I COMANDI SECONDO QUESTE REGOLE"
160 PRINT"
      1. BATTI 'ENTER' DOPO OGNI COMANDO
      2. BATTI ':' PER CREARE 10 NULLI (RITARDO)
      3. BATTI ' END ' PER FINIRE"
170 PRINT"
      BATTI LA LISTA DEI COMANDI:"
180 LINEINPUT A$
190 IF A$=":" THEN 230
200 IF A$=" END " THEN 250
210 I$=I$+A$+CHR$(13)+CHR$(0)
220 GOTO 180
230 I$=I$+STRING$(10,0)
240 GOTO 180
250 I$=I$+CHR$(255)
260 PRINT"LISTA COMPLETATA."
270 PRINT"SI STA CREANDO IL FILE COMANDI."
280 PRINT: GOTO 1260
300 EX$=""
310 FOR F=0 TO 8
320 GOSUB 1480
330 NEXT
340 V=ST+62
350 GOSUB 1400
360 FOR F=11 TO 12
370 GOSUB 1480
380 NEXT F
390 V=ST+25
400 GOSUB 1400
410 FOR K=15 TO 28
420 GOSUB 1480
430 NEXT K
440 V=ST+60
450 GOSUB 1400
460 FOR K=31 TO 41
470 GOSUB 1480
480 NEXT K
490 V=ST+60
500 GOSUB 1400
510 FOR K=44 TO 48
520 GOSUB 1480
530 NEXT K
540 V=ST+62
550 GOSUB 1400
560 FOR K=51 TO 59
570 GOSUB 1480
580 NEXT K
590 V=ST+64
600 GOSUB 1400
610 V=ST+62
620 K=K+1: GOSUB 1400
630 L=LEN(I$)
640 FOR I=1 TO L
650 K=K+1
660 X$=MID$(I$,I,1)
670 GOSUB 1530
680 NEXT I
690 Z=ST+255
700 GOSUB 1610
710 LINEINPUT "NOME DEL FILE? ";V$
720 L=INSTR(V$,".")
730 IF L=0 THEN 1770
750 EX$=LEFT$(EX$,64)+STRING$(20,CHR$(0))+"CMD "
    +CHR$(34)+"S"+CHR$(34)+CHR$(13)+CHR$(0)
    +STRING$(20,CHR$(0))
760 EX$=EX$+"DUMP "+V$+" (START=X'"+H1$+"',END=X'"
    +Z$+"',TRA=X'"+H1$+"')"+CHR$(13)+CHR$(0)
    +CHR$(255)
770 MID$(EX$,23,1)=CHR$(201)
```

```
780 B=0: B1=0: B2=0: B3=0: B4=0: C1=0: C2=0: B5=0
785 PRINT"###";EX$
790 B=VARPTR(EX$)
800 B1=B+1: GOSUB 1200
810 B2=PEEK(B1)
820 B1=B+2: GOSUB 1200
830 B3=PEEK(B1)
840 B4=B3*256+B2
850 B1=B4+62
860 GOSUB 1220
870 MID$(EX$,10,2)=CHR$(C2)+CHR$(C1)
880 B1=B4+25: GOSUB 1220
890 MID$(EX$,14,2)=CHR$(C2)+CHR$(C1)
900 B1=B4+60: GOSUB 1220
910 MID$(EX$,30,2)=CHR$(C2)+CHR$(C1)
920 MID$(EX$,43,2)=CHR$(C2)+CHR$(C1)
930 B1=B4+62: GOSUB 1220
940 MID$(EX$,50,2)=CHR$(C2)+CHR$(C1)
950 B1=B4+64: GOSUB 1220
960 MID$(EX$,61,2)=CHR$(C2)+CHR$(C1)
970 B1=VARPTR(EX$)
980 IF B<>B1 THEN 1250
990 B1=B+1: GOSUB 1200
1000 B5=PEEK(B1)
1010 IF B5<>B2 THEN 1250
1020 B1=B+2: GOSUB 1200
1030 B5=PEEK(B1)
1040 IF B5<>B3 THEN 1250
1050 PRINT"PRONTO PER CREARE IL FILE COMANDI ";V$
1060 B1=B4
1070 GOSUB 1200
1090 DEFUSR1=B1
1100 B4=USR1(0)
1110 END
1120 DATA 245,221,229,221,42,22,64,221,34,221,33
1140 DATA 221,34,22,64,221,225,241,195,45,64,221
1150 DATA 229,221,42,221,126,0,254,255,40,9,221
1170 DATA 35,221,34,221,225,201,221,42,221,34,22
1180 DATA 64,221,225,62,0,201,9999
1200 IF B1>32767 THEN B1=B1-65536
1210 RETURN
1220 C1=INT(B1/256)
1230 C2=B1-256*C1
1240 RETURN
1250 PRINT"STRINGA IN SPOSTAMENTO": GOTO 790
1260 LINEINPUT"INDIRIZZO DI PARTENZA IN ESA? ";H1$
1270 IF LEN(H1$)>4 OR LEN(H1$)<1 THEN 1260
1280 ST=0
1290 FOR I=1 TO LEN(H1$)
1300 ST=ST*16
1310 L=INSTR(H$,MID$(H1$,I,1))
1320 IF L=0 THEN 1260
1330 ST=ST+L-1
1340 NEXT I
1350 PRINT"EQUIVALENTE DECIMALE";ST
1360 IF ST<16384 THEN PRINT
     "NON SI PUO' PARTIRE CON UN INDIRIZZO MINORE
     DI 4000 HEX": GOTO 1260
1370 V1=PEEK(&H40B2): V2=PEEK(&H40B1):
     V3=V1*256+V2+2:
     IF ST<V3 THEN 1720
1380 GOTO 300
1400 V1=INT(V/256)
1410 V2=V-V1*256
1420 X$=CHR$(V2)
1430 GOSUB 1530
1440 X$=CHR$(V1): K=K+1
1450 GOSUB 1530
1460 RETURN
1480 READ X
1490 X$=CHR$(X)
1500 GOSUB 1530
1510 RETURN
1530 AD=ST+K
1540 IF K>0 AND K>K+1<>K THEN PRINT"ERRORE": STOP
1550 K=K
1560 IF AD>32767 THEN AD=AD-65536
1570 POKE AD,ASC(X$)
1580 EX$=EX$+X$
1590 RETURN
1600 REM CONVERSIONE DEC/ESA
1610 K(1)=INT(Z/4096)
1620 Z1=Z-K(1)*4096
1630 K(2)=INT(Z1/256)
1640 Z1=Z1-K(2)*256
1650 K(3)=INT(Z1/16)
1660 K(4)=Z1-16*K(3)
1670 Z$=""
1680 FOR I=1 TO 4
1690 Z$=Z$+MID$(H$,K(I)+1,1)
1700 NEXT I
1710 RETURN
1720 PRINT"     NON E' CONSIGLIABILE CREARE UN
     FILE COMANDI SE NON NELL'AREA DI
     MEMORIA PROTETTA. TI SUGGERISCO DI
     PREMERE <BREAK> E INIZIALIZZARE
     IL BASIC CON UNA 'MEM SIZE'
     MINORE DI";ST;"."
1730 PRINT"     ('MEM SIZE' ATTUALE=";V3;")"
1740 INPUT "BATTI 'ENTER' SE VUOI CONTINUARE
     (PERICOLOSO)"; X$
1750 GOTO 1380
1770 L=INSTR(V$,".")
1780 IF L=0 THEN 1810
1790 V$=LEFT$(V$,L-1)+"/CMD"
     +RIGHT$(V$,LEN(V$)-L+1)
1800 GOTO 750
1810 L=INSTR(V$,":")
1820 IF L=0 THEN V$=V$+"/CMD": GOTO 750
1830 GOTO 1790
```

Cuore Matto

Questo programma gira su Atari con almeno 8 K di memoria.

Cuore matto è la versione di un rompicapo noto anche con il nome di "Teaser" o "Shooting Stars". Lo scopo del gioco è di passare da una configurazione in cui compaiono dei cuori di colore scuro che circondano un cuore di colore chiaro, ad un'altra dove dei cuori chiari circondano un cuore scuro. Quando sceglierete di spostare un cuore, altri gruppi di cuori varieranno a seconda della vostra scelta.

Scegliendo un angolo, cambieranno quattro cuori in un quadrato che include quell'angolo. Scegliendo un cuore nel mezzo di un lato della matrice, si invertiranno tutti i cuori di quel lato.

Infine se si sceglie il cuore centrale, verranno invertiti tutti i cuori che stanno su di una croce con al centro il cuore prescelto.

Non è molto difficile risolvere il problema se non ci si preoccupa del tempo che ci si impiega, ma se si cerca di raggiungere la condizione finale nel minor numero di mosse possibile, allora la questione diventa parecchio più complessa. Dagli esperimenti condotti dalla redazione si è giunti alla conclusione che un numero di mosse abbastanza vicino a quello minimo si aggira attorno a 11, anche se non si è sicuri che sia il più basso..

Se in un qualsiasi momento volete ricondurvi alla condizione iniziale, basta premere la lettera "I" (inizio), azione questa che verrà contata come mossa regolare.

Può essere interessante per gli appassionati dell'arte di programmare l'uso estensivo di stringhe per gestire numeri interi (0-255).

Ci sono parecchi buoni motivi per usare questo sistema. Per prima cosa un solo carattere in una stringa richiede molto meno spazio in memoria di un numero (due byte invece di sette). I numeri possono essere prelevati con il sistema che adotta l'Atari di divisione delle stringhe (vedi linea 920). Il confronto fra le righe è anche utile per confrontare gruppi di più numeri alla volta. Notate infatti come sia facile controllare se si ha vinto o perso nelle linee 360-370.

I numeri vengono scritti nelle stringhe come simboli grafici.

Molte linee di programma in «Cuore matto» contengono speciali caratteri grafici che *non* sono mostrati nel listato, ma che è indispensabile inserire se si vuole giocare.

Per scrivere questi caratteri speciali, tenete premuto il tasto CTRL mentre battete le lettere e le virgole segnate qua sotto.

Linea 160: P$ contiene HCJCLCHEJELEHGJGLG
Linea 170: G$ contiene ABDE,ABC,BCEF,ADG, BDEFHCFI,DEGH,GHI,EFHI
Linea 570: QRWRWRE
Linea 580: ARSRSRD
Linea 590: ARSRSRD
Linea 600: ZRXRXRC

Nelle linee 960 e 980 dovrete mettere due cuori (CTRL e VIRGOLA) il primo dopo l'apertura della parentesi e il secondo prima della sua chiusura. Il primo cuore della linea 960 ed il secondo della linea 980 come pure la parola CHIARO nelle linee 350 e 990 devono essere battute con caratteri inversi. Per finire, se volete risparmiare tempo, potete omettere la battitura delle istruzioni che si trovano nelle righe da 940 a 1250 e cancellate le righe 240, 250, 260. ■

ATARI

```
10 REM ***********************************************
20 REM *                                             *
30 REM *                CUORE MATTO                  *
40 REM *              VERSIONE ATARI                 *
50 REM *                                             *
60 REM *          PERSONAL SOFTWARE                  *
70 REM *                                             *
80 REM ***********************************************
90 REM
100 DIM P$(18),B$(48),M$(1),D$(1),H$(9),I$(9)
110 DIM L$(9),S$(9),N$(9)
120 OPEN #1,4,0,"K:"
130 D$=CHR$(200): B$(2)=B$(1,8)
```

```
140 FOR I=1 TO 9: L$(I)=D$: POKE 85+I,I: NEXT I
150 S$=L$: S$(5,5)=R$: P$(5,5)=D$
160 P$="***** VEDI TESTO *****": M$="I"
170 G$="***** VEDI TESTO *****"
180 GRAPHICS 2: POSITION 4,5
190 PRINT #6;"CUORE  MATTO": FOR I=1 TO 5
200 SETCOLOR 0,I,4: SOUND 0,RND(0)*40+10,10,8: FOR J=1 TO 8
210 GOSUB 760: POKE 756,226
220 SOUND 0,RND(0)*40+10,10,8: GOSUB 760
230 POKE 756,224: NEXT I: SOUND 0,0,0,0
240 POKE 752,1: PRINT: PRINT ,"ISTRUZIONI ";
250 PRINT ,"(S/N)";: GET #1,I
260 IF I=83 THEN GOSUB 940
```

```
270 T=0
280 H$=S$: GOSUB 550: F=2: GOSUB 640
290 T=T+1
300 PRINT CHR$(125);"MOSSA N. ";T
310 PRINT"Numero ? (1-9)"
320 GET #1,K: N=K-48: IF N=25 THEN 280
330 IF N<1 OR N>9 THEN 320
340 GOSUB 920: GET #6,H
350 IF H=160 THEN PRINT CHR$(125): PRINT:
    PRINT"SCEGLI UN CUORE CHIARO...": GOSUB 820:
    GOTO 320
360 GOSUB 710: IF H$=L$ THEN 390
370 IF H$=F$ THEN 430
380 GOTO 290
390 FOR S=30 TO 190: SOUND 0,S,10,8: NEXT S
400 SOUND 0,0,0,0: PRINT CHR$(125)
410 PRINT"Niente da fare....": GOSUB 800
420 GOTO 500
430 PRINT CHR$(125): FOR J=0 TO 14
440 POKE 708,4+16*J: SOUND 0,50-2*J,10,8
450 GOSUB 790: POKE 708,40: GOSUB 790
460 NEXT J: SOUND 0,0,0,0
470 POKE 656,1: POKE 657,10: PRINT"ALLEGRIAAA!!!"
480 PRINT"Ci sei riuscito in ";T;" mosse."
490 GOSUB 800
500 POKE 764,255: PRINT CHR$(125): PRINT
510 PRINT"Ti piacerebbe giocare ancora, EH ?"
520 PRINT"(S / N)": GET #1,K: IF K=83 THEN 270
530 PRINT"Grazie infinite per la tua benevolen="
540 PRINT"za.": END
550 GRAPHICS 2: SETCOLOR 4,2,8: SETCOLOR 3,4,10
560 POKE 756,226: POKE 752,1
570 POSITION 7,2: PRINT #6;"# VEDI TESTO #"
580 POSITION 7,4: PRINT #6;"# VEDI TESTO #"
590 POSITION 7,6: PRINT #6;"# VEDI TESTO #"
600 POSITION 7,8: PRINT #6;"# VEDI TESTO #"
610 FOR X=7 TO 13 STEP 2: FOR Y=5 TO 7 STEP 2
620 POSITION X,Y: PRINT #6;CHR$(124);
630 NEXT Y: NEXT X: RETURN
640 FOR N=1 TO 9: GOSUB 920
650 IF F=1 THEN PRINT #6;H$(N,N);
660 IF F=2 THEN PRINT #6;S$(N,N);
670 IF F=3 THEN PRINT #6;F$(N,N);
680 IF F=4 THEN PRINT #6;N$(N,N);
690 IF F=5 THEN PRINT #6;L$(N,N);
700 NEXT N: RETURN
710 FOR L=1 TO G: F=1: GOSUB 840
720 SOUND 0,5*N+50,10,8: F=2: GOSUB 840
730 SOUND 0,5*N+8,10,8
740 NEXT L: SOUND 0,0,0,0: F=3: GOSUB 840: RETURN
750 FOR K=0 TO C: NEXT K: RETURN
760 C=150: GOTO 750
770 C=1000: GOTO 750
780 C=1000: GOTO 750
790 C=30: GOTO 750
800 POKE 764,255
810 PRINT ,"(Premi un tasto qualunque).";
820 IF PEEK(764)=255 THEN 820
830 RETURN
840 SN=N: J=5*(N-1): FOR I=1 TO 5: N=ASC(B$(J+I))
850 IF N=0 THEN 910
860 GOSUB 920: IF F=1 THEN PRINT #6;" ";
870 IF F=2 THEN PRINT #6;H$(N,N);
880 GET #6,Z
890 IF F=3 AND Z=160 THEN POSITION X,Y:
    PRINT #6;B$;: H$(N,N)=B$
900 IF F=3 AND Z=128 THEN POSITION X,Y:
    PRINT #6;D$;: H$(N,N)=D$
910 NEXT I: N=SN: RETURN
920 X=ASC(P$(2*N-1)): Y=ASC(P$(2*N))
930 POSITION X,Y: RETURN
940 H$=S$: GOSUB 550: F=2: GOSUB 640
950 PRINT"Parti da questa configurazione..."
960 PRINT ,"( INTORNO A )": GOSUB 800
970 F=3: GOSUB 640
980 PRINT CHR$(125);
    "Ed arriva a queste...( INTORNO A )"
990 PRINT"Scegliendo i cuori chiari.": GOSUB 800
1000 GRAPHICS 2: F=4: GOSUB 640: POKE 752,1
1010 PRINT"Questi sono i numeri delle posizioni."
1020 PRINT"Da come scegli";
1030 PRINT" cambieranno di colore"
1040 PRINT"diversi gruppi di cuori."
1050 GOSUB 800: F=2: GOSUB 550: GOSUB 640
1060 PRINT"Se scegli la casella centrale"
1070 PRINT"cambieranno colore i cuori"
1080 PRINT"in croce."
1090 GOSUB 770: N=5: GOSUB 710: GOSUB 800
1100 PRINT CHR$(125);
     "Se scegli la casella mediana"
1110 PRINT"di un bordo, l'intero bordo"
1120 PRINT"cambiera' colore."
1130 GOSUB 770: N=8: GOSUB 710: GOSUB 800
1140 PRINT CHR$(125);
     "Se scegli una casella d'angolo"
1150 PRINT"cambiera' colore il quadrato"
1160 PRINT"che comprende quell'angolo"
1170 GOSUB 720: N=7: GOSUB 710: GOSUB 800
1180 PRINT CHR$(125);
     "Se arrivi ad avere tutti i quadrati"
1190 PRINT"scuri PERDI !!!!!": F=5
1200 GOSUB 640: GOSUB 800
1210 PRINT CHR$(125);
     "Premi il pulsante <I> PER RIPARTIRE"
1220 PRINT"dalla posizione iniziale"
1230 PRINT"HAI CAPITO LE ISTRUZIONI ?"
1240 POKE 764,255: GET #1,K: IF K<>83 THEN 940
1250 RETURN
```

Animazione 3D

Questo programma gira su Atari con almeno 8 K di memoria.

Solitamente quando si vogliono scrivere programmi in cui compaiono disegni tridimensionali, non si può pretendere una elevata velocità di elaborazione. La ragione di ciò diviene evidente se si considera il fatto che c'è un terzo asse di cui tener conto.

Invece di calcoli laboriosi per le coordinate lungo gli assi X, Y e Z (posizione orizzontale, verticale e prospettiva spaziale), questo programma mette a fuoco i punti finali di ogni linea usando la funzione DRAWTO.

Questo metodo permette di disegnare velocemente senza perdere l'impressione di tridimensionalità. Il modo migliore per vedere il procedimento all'opera è di dare il RUN al programma.

Il modo con cui sono strutturati i vettori può sembrare all'inizio abbastanza complesso, ma il concetto è piuttosto semplice.

L'idea di cui tener conto è che il computer deve ricordarsi che cosa è stato disegnato e dove (da qui la necessità di usare i vettori).

Come fa il computer a sapere quando deve cominciare a cancellare?

Per cominciare, bisogna scegliere il numero di linee che vogliamo vedere visualizzate sullo schermo ogni volta: poniamo 30.

Ora diamo invece un'occhiata da vicino a quello che fa il ciclo principale.

Il programma passa attraverso le posizioni del vettore incrementando la variabile A, che memorizza la linea visualizzata in quel momento nella posizione del vettore e cancella la variabile R; il che significa che la linea è stata immessa nella locazione successiva del vettore. Dopo aver completato questo ciclo, per esempio quando A è stato incrementato di 30, A ritorna a 0 cosicché il ciclo può riprendere dalla locazione 0.

Fino a che non c'è niente da cancellare durante il primo ciclo (i contenuti sono portati a zero durante l'inizializzazione del programma), è il momento in cui le 30 linee sono visualizzate. Alla trentunesima linea comincia il secondo ciclo. Questo è il momento in cui viene messo un dato nella locazione R in modo che venga cancellata la prima linea sullo schermo. Procedendo, il programma sostituisce la locazione usata per memorizzare la prima linea con le coordinate della linea 32 in modo da non sprecare spazio nel vettore. Questo sistema può essere immaginato come un serpente: una volta raggiunta la lunghezza desiderata (variabile W) si mangia la coda, e la testa prende il posto dell'ultimo segmento.

Pur essendo il programma già soddisfacente in *graphics 8*, non è difficile cambiare la risoluzione effettuando queste modifiche: cancellare le linee 180, 190 e 200; riscrivere le linee 140, 330 e 350 in questo modo:

```
140 GRAPHICS 8+16: A=-1
330 COLOR 0: PLOT X1 (R):
DRAWTO X2(R), Y2(R):
    COLOR 1
350 NEXT COUNTER: GOTO 60
```

Cambiare il numero 159 in 319 nelle linee 380 e 390; cambiare il numero 95 in 191 nelle linee 400 e 410; inserire la linea 145 COLOR 1.

Risultati interessanti si possono ottenere cambiando le costanti numeriche nel programma. Per esempio nelle linee 160-170 cambiando 1.5 in un numero più alto si sparpaglieranno di più le linee. Un'altra modifica possibile è cambiare la lunghezza del ciclo nella linea 220, modificando così la lunghezza dei tempi di cambiamento dei colori.

Ed ora potete lanciarvi ad esplorare il modo della *computer art*. ■

ATARI

```
10 REM ********************************************
20 REM *                                          *
30 REM *              ANIMAZIONE 3D               *
40 REM *              VERSIONE ATARI              *
50 REM *                                          *
60 REM *           PERSONAL SOFTWARE              *
70 REM *                                          *
80 REM ********************************************
90 REM
100 GRAPHICS 0
```

```
110 PRINT"QUANTE LINEE VUOI ?": INPUT W
120 DIM X1(W),X2(W),Y1(W),Y2(W)
130 FOR I=0 TO W: X1(I)=0: X2(I)=0: Y1(I)=0:
    Y2(I)=0: NEXT I
140 GRAPHICS 7+16: A=-1
150 X1(W)=80: X2(W)=100: Y1(W)=40: Y2(W)=50
160 R1=1.5*(INT(7*RND(0))-3):
    R2=1.5*(INT(7*RND(0))-3)
170 R3=1.5*(INT(7*RND(0))-3):
    R4=1.5*(INT(7*RND(0))-3)
180 COLR=INT(3*RND(0))+1
190 IF OLDCOLR=COLR THEN 180
200 COLOR COLR
```

```
210 REM
220 FOR COUNTER=1 TO INT(10*RND(0))+10
230 A=A+1
240 IF A=0 THEN V=W: GOTO 260
250 V=A-1
260 X1(A)=X1(V): X2(A)=X2(V):
    Y1(A)=Y1(V): Y2(A)=Y2(V)
270 X1(A)=X1(A)+R1: X2(A)=X2(A)+R2
280 Y1(A)=Y1(A)+R3: Y2(A)=Y2(A)+R4
290 GOTO 380
300 PLOT X1(A),Y1(A): DRAWTO X2(A),Y2(A)
310 IF A=W THEN R=0: GOTO 330
320 R=A+1
330 COLOR 0: PLOT X1(R),Y1(R):
    DRAWTO X2(R),Y2(R): COLOR COLR
340 IF A=W THEN A=-1
350 NEXT COUNTER: OLDCOLR=COLR: GOTO 160
360 REM
370 REM
380 IF X1(A)>159 OR X1(A)<0 THEN R1=-R1:
    X1(A)=X1(A)+R1
390 IF X2(A)>159 OR X2(A)<0 THEN R2=-R2:
    X2(A)=X2(A)+R2
400 IF Y1(A)>95 OR Y1(A)<0 THEN  R3=-R3:
    Y1(A)=Y1(A)+R3
410 IF Y2(A)>95 OR Y2(A)<0 THEN R4=-R4:
    Y2(A)=Y2(A)+R4
420 GOTO 300
```

Caleidoscopio

Questo programma crea dei disegni molto belli con simmetria ottagonale. Date il RUN al programma, quindi premete il pulsante START sulla destra della tastiera. Nel centro dello schermo comparirà un puntino lampeggiante. Usando il joystick inserito nello slot 1 si può muovere il punto lungo tutta la superficie dello schermo. Se si tiene premuto il pulsante di fuoco, il punto lascerà lungo il suo tragitto una linea colorata. Altre sette immagini speculari di quella linea appariranno sullo schermo, tre di queste dello stesso colore della prima e le rimanenti di colori diversi.

Premendo il pulsante SELECT (posto sotto lo START sulla destra della tastiera), il computer farà da solo dei disegni e non resterà altro da fare che sedersi su qualche sedia comoda e guardare lo schermo in preda a terribili dilemmi artistici. Se poi si volesse riprendere a disegnare con «le proprie mani», basterà riprendere il joystick e muovere la leva. Premendo il pulsante CLEAR verrà cancellato il disegno in modo da poterne fare un nuovo.

Se si mette un joystick nello slot 2, si potrà controllare anche il cambiamento dei colori, premendo in avanti cambieranno i colori dello sfondo, spingendo a sinistra o a destra la leva dello stick cambieranno i colori delle linee. Premendo il pulsante di fuoco durante una di queste operazioni non cambieranno i colori, bensì la luminosità degli stessi. Con otto tonalità per ognuno dei sedici colori, si hanno a disposizione oltre DUE MILIONI di combinazioni possibili!!! ■

ATARI

```
10 REM ********************************************
20 REM *                                          *
30 REM *              CALEIDOSCOPIO               *
40 REM *              VERSIONE ATARI              *
50 REM *                                          *
60 REM *            PERSONAL SOFTWARE             *
70 REM *                                          *
80 REM ********************************************
90 REM
100 GRAPHICS 0: POKE 752,1:
    L=6+PEEK(741)*256*PEEK(742): POSITION 3,4
    PRINT"cAlEiDoScOpIo": POSITION 23,4:
    PRINT" XXXXXXXXXXX"
120 POSITION 6,9: PRINT"PREMI START (JOYSTICK-1)"
130 SETCOLOR 2,2,4: SETCOLOR 4,2,4: SETCOLOR 0,2,8
140 POKE L+4,7: POKE L+5,6: IF PEEK(764)<>7 THEN 140
150 IF PEEK(START)=7 THEN 150
160 X=47: Y=X: GRAPHICS 23: C=3: H1=12: L1=8
170 H2=L1: L2=2: H4=L2: L4=H2: GOSUB 380: R=32:
    P=95: D=127
180 S=STICK(0): GOSUB 200:
    IF PEEK(START)=5 THEN 400
190 GOTO 180
200 PLOT X+A,Y: COLOR 1: PLOT X+A,Y: T=STRIG(0)
210 COLOR C-C*T: GOSUB 440: COLOR 2-T-T: GOSUB 440
220 IF PEEK(KEY)=54 THEN POKE KEY,0: GRAPHICS 23:
    X=47: Y=X: GOTO 380
230 IF S/2=INT(S/2) THEN Y=Y-1+P*(Y=0)
240 IF S=9 OR S=13 OR S=5 THEN Y=Y+1-P*(Y=P)
250 IF S>8 AND S<13 THEN X=X-1+P*(X=0)
260 IF S>4 AND S<9 THEN X=X+1-P*(X=P)
270 V=STICK(1): IF V=15 THEN RETURN
280 W=STRIG(1)*2
290 IF V=14 THEN H4=H4+1-(W=0): L4=L4+2-W
300 IF H4=16 THEN H4=0
310 IF L4=16 THEN L4=0
320 IF V=11 THEN H1=H1+1-(W=0): L1=L1+2-W
330 IF H1=16 THEN H1=0
340 IF L1=16 THEN L1=0
350 IF V=7 THEN H2=H2+1-(W=0): L2=L2+2-W
360 IF H2=16 THEN H2=0
370 IF L2=16 THEN L2=0
380 SETCOLOR 0,0,0: SETCOLOR 1,H1,L1
390 SETCOLOR 2,H2,L2: SETCOLOR 4,H4,L4: RETURN
400 T=0: S=5+INT(RND(0)*10):
    IF STICK(0)<15 THEN 180
410 IF RND(0)<0.1 THEN T=1
420 L=2+RND(0)*10: FOR I=1 TO L: GOSUB 210: NEXT
430 GOTO 400
440 PLOT X+A,Y: PLOT D-X,Y: PLOT D-X,P-Y
450 PLOT X+A,P-Y: Z=X: X=Y: Y=Z: RETURN
```

Odissea nello spazio

Il programma consiste nel ricercare una astronave dispersa in uno spazio cubico di 7 chilometri di lato.

La ricerca viene effettuata mediante l'input di tre coordinate, che rappresentano la x, la y e la z dello spazio tridimensionale.

Dopo l'inserimento di dette coordinate, il programma manda i seguenti messaggi:

VERSO L'ALTO

se la coordinata x della astronave è maggiore di quella inserita.

VERSO IL BASSO

se la coordinata x della astronave è minore di quella inserita.

CON UN ALTRO VALORE DI Y

se la coordinata y della astronave è diversa da quella inserita.

IN AVANTI

se la coordinata z della astronave è maggiore di quella inserita.

INDIETRO

se la coordinata z della astronave è minore di quella inserita.

Il giocatore ha 15 ore per trovare l'astronave; ogni tentativo fallito fa diminuire di un'ora la durata dell'ossigeno a cui è vincolata la vita degli astronauti. ■

sinclair ZX80

```
10 REM ****************************************
20 REM *                                      *
30 REM *        ODISSEA NELLO SPAZIO          *
40 REM *        VERSIONE SINCLAIR ZX80        *
50 REM *                                      *
60 REM *          PERSONAL SOFTWARE           *
70 REM *                                      *
80 REM ****************************************
90 REM
100 PRINT ,"ODISSEA NELLO SPAZIO"
110 PRINT
120 PRINT "HAI 15 ORE PER RITROVARE"
130 PRINT "UN'ASTRONAVE PERSA IN UNO"
140 PRINT "SPAZIO DI 7 KM CUBI."
150 PRINT
160 LET A=RND(7)
170 LET B=RND(7)
180 LET C=RND(7)
190 FOR J=1 TO 15
200 PRINT
210 PRINT "INSERISCI LE TRE COORDINATE X,Y,Z"
220 INPUT D
230 INPUT E
240 INPUT F
250 CLS
260 PRINT, D,E,F
270 IF A=D AND B=E AND C=F THEN GO TO 490
280 PRINT
290 PRINT, "(SHIFT A) COORDINATE ERRATE (SHIFT A)"
300 PRINT
310 PRINT "ORE DI OSSIGENO RIMASTE -"; 15-J
320 PRINT
330 PRINT "MUOVITI";
340 IF A<D THEN PRINT "VERSO L'ALTO";
350 IF A>D THEN PRINT "VERSO IL BASSO";
360 IF NOT B=E THEN
    PRINT "CON UN ALTRO VALORE DI Y ";
370 IF C<F THEN PRINT "IN AVANTI"
380 IF C>F THEN PRINT "INDIETRO"
390 PRINT
400 IF J=14 THEN
    PRINT "PERICOLO - MORTE IMMINENTE"
410 NEXT J
420 CLS
430 PRINT
440 PRINT "DISASTRO - GLI ASTRONAUTI SONO MORTI"
450 PRINT
460 PRINT "L'ASTRONAVE ERA ALLE COORDINATE"
470 PRINT "X=";A;"Y=";B;"Z=";C
480 GO TO 63999
490 PRINT
500 PRINT, "COMPLIMENTI. HAI TROVATO LA"
510 PRINT "ASTRONAVE A ";16-J;" ORE DALLA FINE"
520 PRINT
530 PRINT "UN'ALTRA MISSIONE ?"
540 INPUT H$
550 CLS
560 IF H$="SI" THEN GO TO 120
570 PRINT "ARRIVEDERCI ALLA PROSSIMA MISSIONE."
580 STOP
```

Roulette russa

sinclair ZX80

```
10 REM ******************************************
20 REM *                                        *
30 REM *              ROULETTE RUSSA            *
40 REM *          VERSIONE SINCLAIR ZX80        *
50 REM *                                        *
60 REM *            PERSONAL SOFTWARE           *
70 REM *                                        *
80 REM ******************************************
90 REM
100 PRINT, "ROULETTE RUSSA"
110 PRINT
120 PRINT "HAI UNA PISTOLA CON UN"
130 PRINT "PROIETTILE DENTRO."
140 PRINT "RUOTERAI IL TAMBURO E"
150 PRINT "SPARERAI 10 VOLTE."
160 PRINT "TE LA SENTI DI RISCHIARE?"
170 INPUT A$
180 LET J=0
190 PRINT
200 IF A$="NO" THEN GO TO 440
210 PRINT "PREMI NEW LINE PER SPARARE"
220 INPUT T$
230 CLS
240 PRINT
250 PRINT
260 PRINT
```

Il principio del gioco è semplice: hai una pistola con un solo proiettile, ruoti di tamburo, tiri il grilletto e... BANG o CLICK. ■

```
270 LET J=J+1
280 LET G=RND(6)
290 PRINT, "SPARO NUMERO ";J
300 IF G<6 THEN GO SUB 460
310 IF G=6 THEN GO TO 500
320 IF J=10 THEN GO TO 350
330 IF J<10 THEN GO TO 210
340 CLS
350 PRINT
360 PRINT
370 PRINT, "SEI SOPRAVVISSUTO."
380 PRINT "SE VUOI RISCHIARE ANCORA LA"
390 PRINT "MORTE, BATTI R PER RISCHIARE"
400 PRINT "O F PER FERMARTI."
410 INPUT B$
420 CLS
430 IF B$="R" THEN GO TO 180
440 PRINT "CODARDO...";
450 GOTO 440
460 LET H=RND(2)
470 IF H=1 THEN PRINT "$ CLICK $"
480 IF H=2 THEN PRINT "TAMBURO VUOTO"
490 RETURN
500 PRINT "BANG...";
510 GOTO 500
```


ASSEGNATI ALTRI DUE PERSONAL COMPUTER DEL CONCORSO CAMPAGNA ABBONAMENTI JACKSON 1982

Si sono svolte altre 2 estrazioni del Concorso Campagna Abbonamenti Jackson 1982. Sono risultati vincitori di un personal computer Sinclair ZX-81 i signori:

Scopinaro Francesco
Via Vespri Siciliani, 7 - Roma

La Notte Giuseppe
Via Concordia, 7 - Corsico (MI)

Congratulazioni!

Reverse

Questo programma occupa 1 K di memoria centrale. L'obiettivo del gioco è disporre le nove cifre da 1 a 9, che inizialmente sono disposte a caso, nel loro ordine naturale mediante inversione delle prime N cifre. Ad esempio, se viene stampato

567894321

rispondendo 5 si otterrà

987654321

e rispondendo ora 9, si avrà

123456789

ed il gioco è concluso. ■

sinclair ZX80

```
10 REM ****************************************
20 REM *                                      *
30 REM *              REVERSE                 *
40 REM *        VERSIONE SINCLAIR ZX80        *
50 REM *                                      *
60 REM *          PERSONAL SOFTWARE           *
70 REM *                                      *
80 REM ****************************************
100 PRINT "        REVERSE"
110 PRINT
120 PRINT"PREMI NEW LINE PER INIZIARE"
130 INPUT Y$
140 DIM A(9)
150 FOR I=1 TO 9
160 LET N=RND(9)
170 IF I=1 THEN GO TO 210
180 FOR J=1 TO I-1
190 IF N=A(J) THEN GO TO 160
200 NEXT J
210 LET A(I)=N
220 NEXT I
230 LET C=0
240 CLS
250 FOR I=1 TO 9
260 PRINT A(I);" ";
270 NEXT I
280 PRINT
290 PRINT
300 PRINT"NUMERO DI CIFRE DA INVERTIRE?"
310 INPUT N
320 PRINT N
330 IF N>0 AND N<10 THEN GO TO 360
340 PRINT"ERRORE - RIPROVA"
350 GO TO 250
360 LET K=(N+1)/2
370 FOR I=1 TO K
380 LET J=A(I)
390 LET A(I)=A(N+1-I)
400 LET A(N+1-I)=J
410 NEXT I
420 LET C=C+1
430 FOR I=1 TO 9
440 IF NOT I=A(I) THEN GO TO 240
450 NEXT I
460 PRINT "HAI FATTO ";C;" INVERSIONI"
470 PRINT "UNA ALTRA PARTITA?"
480 INPUT Y$
490 IF CHR$(CODE(Y$))="S" THEN GO TO 150
500 STOP
```

REM come dati

La zona in cui è allocato il programma è l'unica della memoria che viene conservata su nastro magnetico. Per questa ragione, se si vogliono memorizzare dei dati, è necessario inserirli all'interno del programma stesso.

Questi dati ovviamente non devono influire sulla logica del programma e ciò è realizzabile utilizzando l'istruzione REM all'inizio del programma.

È noto che l'indirizzo di inizio del programma è 16424, e che in una linea così organizzata:

nn REM ..... (new line)

il numero di linea occupa sempre due byte ed il REM ne occupa uno, come il new line; inserendo dopo il REM una serie di caratteri fittizi occupanti un byte ciascuno, è possibile riservare una zona di memoria dove successivamente si inseriranno i dati da registrare, carattere per carattere. (Ovviamente il numero dei caratteri fittizi inseriti inizialmente deve essere sufficiente per contenere la quantità di dati prevista.)

Per inserire i dati carattere per carattere è necessaria

l'istruzione:

POKE 16427,p

poiché abbiamo visto che il numero di riga (nn) occupa i due byte e REM ne occupa uno:

16424
16425 nn.
16426 REM
16427 p
16428 ...
.....

Questo programma permette di memorizzare dati per un massimo di 63 caratteri. La prima locazione utilizzabile per la memorizzazione è la 16427; questo byte viene utilizzato come contatore di caratteri inseriti. Inizialmente questo contatore è posto a zero e quindi nella REM iniziale è visualizzato con un blank.

Dopo ogni INPUT viene eseguito un controllo per verificare se l'area rimasta disponibile è sufficiente a contenere il dato stesso.

Alla fine di ogni dato il programma inserisce automaticamente un carattere "tappo" (codice 70) che permette in fase di stampa di separare un dato dall'altro.

Per la memorizzazione di un numero maggiore di dati si devono inserire più caratteri nella REM, modificando opportunamente i controlli necessari. ■

sinclair ZX80

```
100 REM b PP...(63 VOLTE P)...PP
110 PRINT "MEMORIZZAZIONE M O STAMPA S?"
120 INPUT Z$
130 IF Z$="S" THEN GO TO 340
140 PRINT "DATO DA INSERIRE: ";
150 INPUT A$
160 PRINT A$
170 LET C=PEEK(16427)
180 LET F=0
190 LET B$=A$
200 LET B$=TL$(B$)
210 LET F=F+1
220 IF NOT B$="" THEN GO TO 200
230 IF NOT (F+1)>(63-C) THEN GO TO 260
240 PRINT "SPAZIO INSUFFICIENTE"
250 GO TO 110
260 FOR F=1 TO F
270 POKE 16427+C+F,CODE(A$)
280 LET A$=TL$(A$)
290 NEXT F
300 POKE 16427+C+F,148
310 LET C=C+F
320 POKE 16427,C
330 GO TO 110
340 CLS
350 LET C=PEEK(16427)
360 LET F=1
370 IF PEEK(16427+F)=148 THEN PRINT
380 IF F=C THEN GO TO 430
390 IF PEEK(16427+F)=148 THEN GO TO 410
400 PRINT CHR$(PEEK(16427+F));
410 LET F=F+1
420 GO TO 370
430 STOP
```

sinclair ZX80

Calcoli logici

L'uso corretto delle variabili logiche aiuta a scrivere programmi più efficienti. Il manuale dello ZX80 introduce il concetto di calcolo logico con la frase IF, che ha la sintassi

IF (condizione)
THEN (istruzione)

che è equivalente a

IF (variabile logica è vera)
THEN (istruzione

Qualsiasi variabile numerica può essere usata come variabile logica in una fase IF. Se una variabile numerica è zero, sarà considerata come falsa in un calcolo logico, altrimenti sarà considerata vera.

Per esempio:

IF NOT D=0 THEN PRINT D

ha il medesimo effetto di:

IF D THEN PRINT D

Nel primo caso viene creata una variabile logica temporanea che viene posta a zero (falsa) se D=0 oppure a −1 (vera) se D non è uguale a zero. Viene quindi analizzato il valore di tale variabile logica temporanea.

Nel secondo caso, è la variabile D stessa che viene presa come variabile logica da analizzare. Allo stesso modo:

IF NOT D=−1 THE PRINT D

ha lo stesso effetto di

IF NOT D THEN PRINT D

Nel primo caso viene creata una variabile logica temporanea, quindi posta a zero (falsa) se D=−1, oppure a −1 (vera) se D è uguale a 0.

Nel secondo caso se D è uguale a −1 allora NOT D è uguale a zero (falsa), se D è diverso da −1 NOT D è uguale a −1 (vera).

I calcoli logici possono essere introdotti anche in espressioni artimetiche, e, ancora una volta, prende-

ranno il valore zero se sono falsi, oppure −1 se sono veri. Ad esempio:

```
LET S=0
IF R > 15 THEN LET S = 7
```

è equivalente a

```
LET S=−7* (R>15)
```

Nel secondo caso (R>15) prende il valore −1 se è effettivamente R > 15, e così S è posta uguale a 7 = 7* −1. Se invece R < 16 allora (R > 15) prenderà il valore zero, così S sarà posta uguale a 0 = −7*0.

In tal modo, incorporando calcoli logici ed espressioni artimetiche, si possono a volte eliminare degli IF non indispensabili. ■

sinclair ZX80

Libreria di Programmi

Alcuni suggerimenti per crearvi una vostra biblioteca di programmi:

1. Non registrate tutti i programmi su un'unica cassetta; questo, se da una parte consente di raggruppare i programmi in poco spazio, dall'altra vi mette in difficoltà nel caso di una rapida ricerca di un particolare programma, anche se etichettato con una etichetta vocale.
2. Registrate i programmi, uno per ogni facciata, su cassette della durata di pochi minuti, scrivendo chiaramente di che programma si tratta sull'etichetta e sulla copertina della cassetta.
3. Duplicate tutta la vostra libreria; in questo caso potete anche utilizzare una sola cassetta di lunghezza appropriata, ricordandovi però di inserire sempre una etichetta vocale prima di ogni programma e di annotare, se possibile, il numero di giri del registratore a cui inizia il programma.

Questa duplicazione sarà utile in caso di perdita dell'originale o anche di un suo danneggiamento. ■

sinclair ZX80

Occupazione in memoria di una linea di programma

Quando si scrivono programmi per piccoli calcolatori, è importante sfruttare nel modo migliore la capacità della memoria.

Il primo accorgimento da seguire, se si scrive un programma in Basic, è quello di conoscere quanto spazio questo occupa all'interno della RAM.

A questo scopo ricordiamo che:

1. I numeri di linea sono memorizzati in due locazioni eccetto quando seguono GO TO e GO SUB; in questo caso vengono memorizzati simbolo dopo simbolo, un simbolo per locazione;
2. parole chiave e simboli vengono memorizzati uno per locazione;
3. ogni linea termina con 76H (118 in decimale) che occupa una locazione.

Perciò la linea:

```
340IF I < 0 OR I > 6 THEN
    GO TO 510 (new line)
```

occuperà 16 locazioni. Si noti che il numero di linea, 340, è memorizzato in codice binario e perciò occupa due caratteri in ogni caso.

Invece il numero di linea presente dopo il GO TO, in questo caso 510, occupa una locazione per ogni cifra.

Nel caso in cui la memoria risulti insufficiente per un programma, una soluzione può essere quella di rinumerare le linee dello stesso facendo in modo che i numeri di linea presenti dopo i GO TO e i GO SUB occupino meno locazioni (cioè siano composti da un minor numero di cifre). ■

LIBRI PERSONAL SOFTWARE

Vuoi ordinare dei libri? Spedisci questo tagliando a: Gruppo Editoriale Jackson Via Rosellini, 12 - 20124 Milano.

Nome Cognome

Indirizzo

Cap. Città

Codice Fiscale (indispensabile per le aziende)

Inviatemi i seguenti libri:

☐ Pagherò al postino l'importo di L. ..... + L. 1.500 per contributo fisso spese di spedizione

☐ Allego assegno n° ..... di L. .....
(in questo caso la spedizione è gratuita)

Codice Libro	Quantità	Codice Libro	Quantità	Codice Libro	Quantità	Codice Libro	Quantità

☐ Non abbonato ☐ Abbonato Data..... Firma ... .

N.B. È possibile effettuare versamenti anche sul ccp n° 11666203 intestato a: Gruppo Editoriale Jackson - Via Rosellini, 12 - 20124 Milano. In questo caso specificare nell'apposito spazio sul modulo di ccp la causale del versamento e non inviare questo tagliando.

ABBONAMENTI PERSONAL SOFTWARE

Vuoi abbonarti alle riviste Jackson? Spedisci questo tagliando a: Gruppo Editoriale Jackson, Via Rosellini, 12 - 20124 Milano.

☐ Desidero sottoscrivere il seguente abbonamento:

☐ EO L. 31 000	☐ SM L. 24 000	☐ IO + LE L. 59 500	☐ EO + IO + LE + BIT L. 110 500
☐ ED L. 6 500	☐ EO + ED L. 35 500	☐ BIT + PS L. 54 000	☐ EO + IO + LE + BIT + PS L. 137 000
☐ LE L. 35 000	☐ EO + IO L. 64 000	☐ EO + LE + ED L. 68 500	☐ EO + IO + LE + ED + BIT + PS L. 140 000
☐ IO L. 26 500	☐ EO + BIT L. 55 000	☐ EO + IO + BIT L. 79 500	☐ EO + IO + LE + ED + BIT + PS + SM L. 149 000
☐ BIT L. 26 000	☐ EO + LE L. 55 500	☐ EO + IO + LE L. 88 500	
☐ PS L. 30 000	☐ IO + BIT L. 50 500	☐ IO + BIT + PS L. 78 500	☐ ..... L. .....

EO = Elettronica Oggi - ED = E.O. Data - LE = L'Elettronica - IO = Informatica Oggi - Bit - PS = Personal Software - SM = Strumenti Musicali

☐ L'abbonamento dovrà decorrere dal mese di .....

Nome Cognome

Indirizzo

Cap. Città

Codice Fiscale (indispensabile per le aziende)

☐ Pagherò al postino il prezzo indicato al ricevimento del 1° fascicolo

☐ Allego assegno n° ..... di L. .....

☐ Pagherò al ricevimento della vostra fattura (formula riservata alle sole aziende)

Firma _____ Data _____

N.B. È possibile effettuare versamenti anche sul ccp n° 11666203 intestato a Gruppo Editoriale Jackson - Via Rosellini, 12 - 20124 Milano. In questo caso specificare nell'apposito spazio sul modulo di ccp la causale del versamento, e **non inviare** questa cartolina.

UNA PUBBLICAZIONE DEL GRUPPO EDITORIALE JACKSON

PERSONAL SOFTWARE

ANNO 1 N. 1 LUGLIO/AGOSTO 1982

DIRETTORE RESPONSABILE: Giampietro Zanga
DIRETTORE: Mauro Boscarol
REDAZIONE: Completo Software
HANNO COLLABORATO A QUESTO NUMERO: P. Dell'orco, A. Giovannetti, R. Tamagna, F. Petracchi, P. Anelli, W. D. Maurer, D. R. Hope, R. Kaplan, A. Leal, N. Tonolli, G. Staluppi, Lexikon, M. Giordino
Copertina: Maria Elisa Rizzo
Grafica e Impaginazione: Fernanda Arzenton
Fotocomposizione: Composizioni Grafiche - Padova
Traduzioni: B. Pescarolo, L. Tagliarolo, R. Pevarolo
CONTABILITA': Franco Mancini, Roberto Ostelli, Marielia 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
PUBBLICITA': Concessionario per l'Italia e l'Estero Reina s.r.l. Via Washington, 90 - 20146 Milano Tel. (02) 4986066/7/8/9/060 (5 linee r.a.) Telex 316213 REINA I
STAMPA: Arti Grafiche "La Cittadella" S.p.a. Pieve del Cairo (PV)
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

PERSONAL IBM

IBM PERSONAL COMPUTER: Unità di sistema fino a 256 Kbyte, con microprocessore 8088 a 16 bit. 2 unità drive per minifloppy da 160 Kbyte ciascuno. Monitor fosfori verdi 12", 25 righe per 80 col. Stampante. Alimentatore.

Importato e distribuito da

HARD&SOFT SYSTEMS

CENTRI DI DISTRIBUZIONE: HARD & SOFT SYSTEM S.R.L., Via Costantinopoli 50, 47045 Miramare di Rimini, Tel. 0541-31060/759076/759077 ● AVELCO S.R.L., Via Cornigliano 47, Genova, Tel. 010-602994 ● DP INOFSHOP S.N.C., V.le Storchi 10/10A, 41100 Modena, Tel. 059-218821 ● SIRIO SHOP, V.le Certosa 148, Milano, Tel. 02-304713/305778/3080786 ● STUDIO P, Via Paganino Bonafede 2/A, 40139 Bologna, Tel. 051-548080 ● INFODIS S.R.L., Via A. Oriani 10, 04100 Latina, Tel. 0773-491271 ● C.I.S.I.D., Via Pasubio 11, 58100 Grosseto, 0564-414233

Apple cresce.

response

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 200 centri specializzati che fanno di Apple il loro cavallo di battaglia.

Ma cresce anche la gamma Apple. Oltre al già famoso e collaudatissimo Apple II, la Iret presenta Apple III, piú potente e adatto ad usi specialistici. E poi video per ogni esigenza, a fosfori verdi o a colori, stampanti e decine di accessori e programmi.

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.

apple® computer

Distribuzione per l'Italia

IRET® informatica

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