LA PRIMA RIVISTA EUROPEA DI SOFTWARE PER PERSONAL COMPUTER

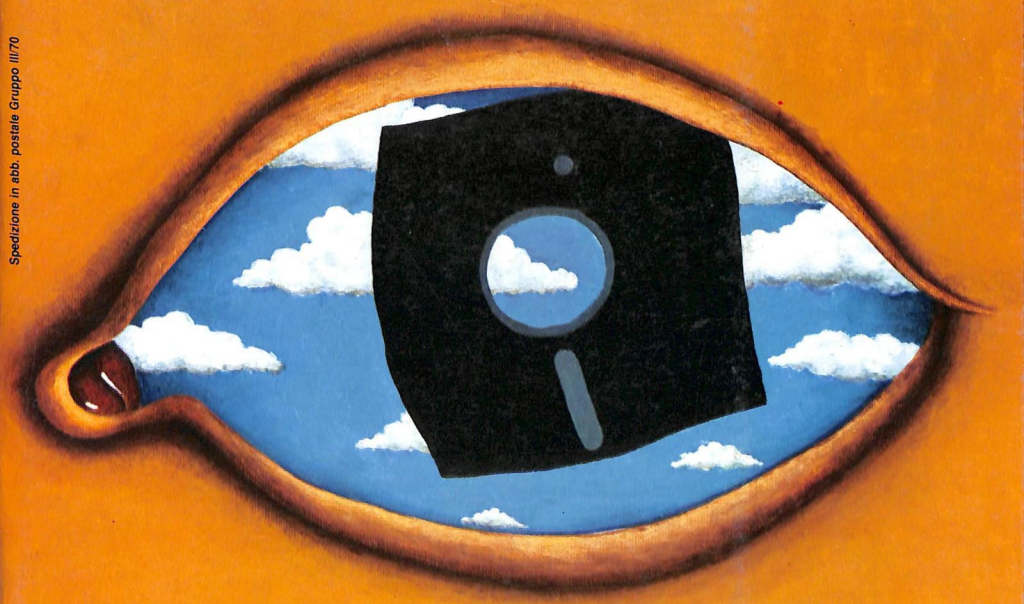
PERSONAL SOFTWARE

in omaggio GUIDA ALLO ZX80-ZX81

ANNO 2 N. 7
GIUGNO 1983 L. 3.500

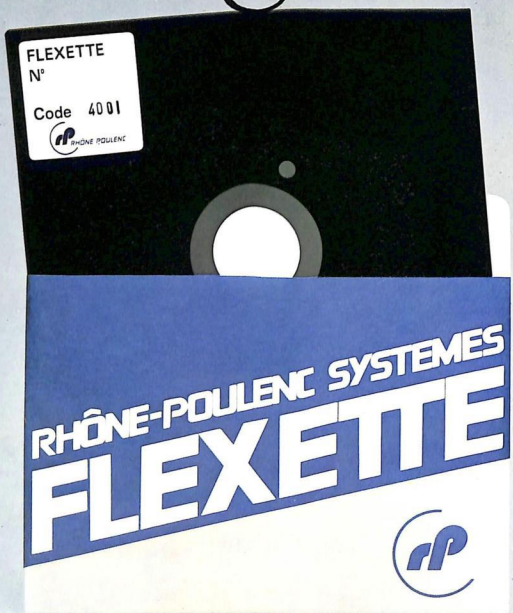
UNA PUBBLICAZIONE DEL GRUPPO EDITORIALE JACKSON

Spedizione in abb. postale Gruppo III/70



- PROJECT ROBOT: UN ROBOTWAR PER LO ZX81
- DATA BASE MODULARE PER L'APPLE
- COME STAMPARE LO SCHERMO DEL VIC
- I SEGRETI DEI PERSONAL: VIC 20, TEXAS 99/4A, NUOVA ELETTRONICA, ZX81, PET/CBM, APPLE
- TRASFORMAZIONE DI COORDINATE TOPOGRAFICHE CON IL TRS-80

leggete tra le righe...



Rispondere alle esigenze sempre maggiori degli utilizzatori di mini e micro computers è la missione che si è fissata RHONE-POULENC SYSTEMES fabbricando FLEXETTE. Grazie alla tecnologia messa in opera nella fabbricazione di questi dischi, l'utilizzatore usufruisce di condizioni ottimali di registrazione. In particolare la certificazione 100% ERROR FREE sia sulle tracce che tra le tracce assicura la conservazione anche in condizione d'impiego marginali e l'intercambiabilità dell'informazione registrata. Gli standards di qualificazione che determinano la certificazione di FLEXETTE si situano ben al di là dei limiti fissati dagli standard industriali. FLEXETTE rimane ERROR FREE anche dopo più di 40 milioni di passaggi sulla stessa traccia. FLEXETTE è riservato agli utilizzatori che ricercano la garanzia di un'alta tecnologia.

Per provare FLEXETTE nella Vostra regione:

RHÔNE POULENC SYSTEMES
settore informatica
concessionari autorizzati

- **MILANO** - **S.D.C.** S.a.S. / Tel. 84.35.593
- **TORINO** - **PROGRAMMA UFFICIO** S.a.S. / Tel. (011) 41.13.565
- **VERONA** - **MIDA** S.r.l. / Tel. (045) 59.05.05
- **FIRENZE** - **C.S.S.** S.n.c. / Tel. (055) 67.96.30
- **PARMA** - **TECNODATA** S.a.S. / Tel. (0521) 25.079
- **ROMA** - **MASSIMO BRENUANI** / Tel. (06) 81.27.665
- **NAPOLI** - **TES. IN** / Tel. (081) 64.31.22
- **BOLZANO** - **DATAPLAN** S.a.S. / Tel. (0471) 47.721-47.056

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

PROGRAMMI DI MATEMATICA E STATISTICA

Leggendo questo libro il lettore potrà formarsi quella logica di base indispensabile per la risoluzione di problemi di matematica e statistica.
Ad ogni programma viene preposta un'esposizione schematica del metodo numerico e delle tecniche di programmazione utilizzate, il diagramma a blocchi relativo all'algoritmo, il listato (anch'esso ottenuto da calcolatore) in cui tra l'altro vengono specificati il tempo e la quantità di memoria impiegati.

Cod. 552D **L. 16.000 Pagg. 228**

INTRODUZIONE AL PASCAL

Il volume, incentrato su numerosissimi esempi che verificano costantemente l'apprendimento del lettore, insegna a conoscere, capire ed usare tutte le particolarità e i vantaggi di questo linguaggio. Nel corso della trattazione vengono ampiamente utilizzate le tecniche di programmazione strutturata, come pure tecniche particolari, quali il trattamento dei file, l'utilizzazione della recursività e il trattamento grafico.

Cod. 516A **L. 30.000 Pagg. 484**

COMPUTER GRAFICA

Si può dire che la computer grafica si pone nel contesto più generale del trattamento dell'informazione, avendo individuato nell'immagine un contenuto informativo che è possibile elaborare.
Quest'opera, con il suo rigore informativo e scientifico, si pone come fondamentale nel carente panorama italiano; inoltre le informazioni e gli spunti contenuti nel testo contribuiranno certamente alla divulgazione ed alla formazione di idee nuove e feconde.

Cod. 519P **L. 29.000 Pagg. 174**

CEDOLA DI COMMISSIONE LIBRARIA

Ritagliare (o fotocopiare) e inviare a
Gruppo Editoriale Jackson Via Rosellini, 12 - 20124 Milano

Nome e Cognome _____

_____

Indirizzo _____

_____

Cap. _____ Città _____ Provincia _____

Partita I.V.A. (indispensabile per le aziende)

Si richiede l'emissione della fattura ☐

Inviatemi i seguenti libri:

Codice Libro	Quantità	Codice Libro	Quantità	Codice Libro	Quantità	Codice Libro	Quantità

☐ Pagherò al postino il prezzo indicato + L 2.000 per contributo fisso spese di spedizione

☐ Allego assegno n° ..... di L. .....

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

☐ Non Abbonato ☐ Abbonato sconto 10% ☐ L'Elettronica ☐ Elettronica Oggi ☐ Automazione Oggi ☐ Elektor ☐ Informatica Oggi ☐ Computerworld ☐ Bit ☐ Personal Software ☐ Strumenti Musicali ☐ Videogiochi

... dalla libreria JACKSON

APPLE II - Guida all'uso

Se possedete un Apple e volete conoscerlo a fondo, se volete comprarlo, o se semplicemente volete imparare la sua programmazione, troverete in questo libro, tutte le risposte, comprese alcune vere "primizie" che vi occorrono per una perfetta operatività del sistema. Conoscerete i vari componenti del sistema e come usarli al meglio. Verrete guidati alla programmazione in BASIC e a usare le caratteristiche grafiche e sonore del sistema. Imparerete a memorizzare su disco sia programmi che archivi dati, come ad inserire un programma scritto in assembler in uno scritto in BASIC.
E poi ancora, tutte le istruzioni e funzioni BASIC e ben 12 appendici veramente basilari.

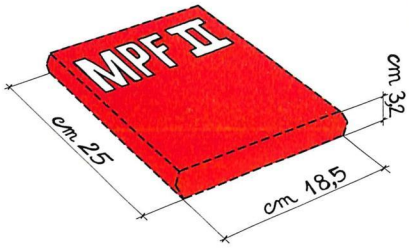
Cod. 331P **L. 26.000 Pagg. 400**

SCONTO 10% agli abbonati

GRUPPO EDITORIALE JACKSON

Divisione Libri

1480 cm³



di MICRO-PROFESSOR MPF II

contengono CPU R6502 - 64 K Bytes di RAM
16 K Bytes di ROM con Interprete Basic Apple Soft

Il MICROPROFESSOR II (MPFII) è un computer unico nel suo genere perché unisce a grandi capacità di memorie residenti **(64 K Bytes di RAM e 16 K Bytes di ROM)** una configurazione di sistema ridottissima.
È veramente portatile.
Le sue minime dimensioni (cm 25 × 18,5 × 3,2) non gli impediscono però di essere un "personal computer" perché oltre ad essere dotato di eccezionali capacità di memoria residenti può essere completato ed allacciato con diverse periferiche.
MPFII diventa così un computer gestionale come altri computer più famosi ed "ingombranti" di lui.
Il modulatore RF e la scheda PALCOLOR residenti vi permetteranno di collegarlo al vostro televisore.
Ecco perché MPFII non è solo "lavoro", ma anche relax.
Insomma un computer idoneo per tutti, dai 7 ai 70 anni di età.
L'ampia disponibilità di software in cassetta, dischi e cartuccia (cartridge) costituisce l'elemento preponderante che lo rende indispensabile come: **SUPPORTO GESTIONALE** (amministrazione, magazzino, acquisti, commerciale, ecc.) per negozi, uffici, aziende. **SUPPORTO SCIENTIFICO PRATICO** per tecnici, professionisti, ricercatori, hobbysti. **SUPPORTO DIDATTICO** per studenti. **SUPPORTO RICREATIVO** (giochi, quiz, ecc.) per tutti.

1) Computer
2) Interfaccia per disk drive
3) Disk drive (slim line)
4) Tastiera esterna

DIGITEK COMPUTER

Ufficio Vendite
Via Marmolada, 9/11 43058 SORBOLO (Parma)
Tel. 0521/69635 Telex 531083

ANNO 2 N. 7 GIUGNO 1983

PERSONAL SOFTWARE

In copertina: l'occhio computerizzato dell'intelligenza artificiale in una illustrazione di Renata Manzato, Studio Morgana.

ARTICOLI

RUBRICHE

Indirizzate tutta la corrispondenza editoriale a

Personal Software, Via Rosellini 12, 20124 Milano

GUIDA

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

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.

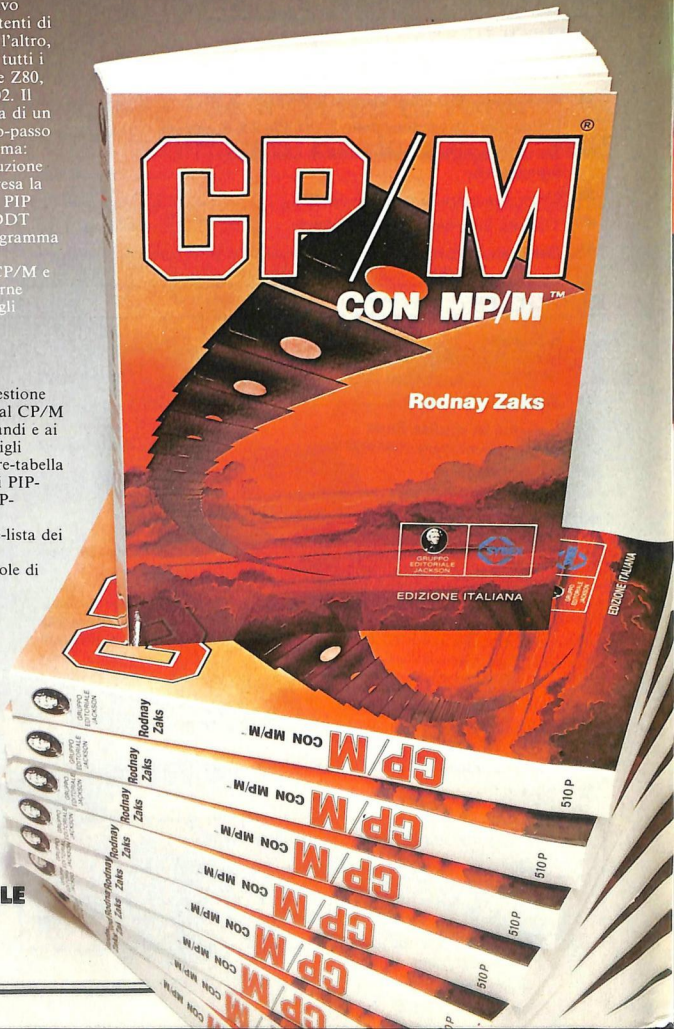
SOMMARIO

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

Alle frontiere della scienza dei computer

Mauro Boscarol

Coloro che fanno professionalmente della ricerca nel campo dell'informatica sono stati educati e hanno cominciato la loro attività in un mondo in cui l'informatica non esisteva. Alcuni tra i meno giovani, addirittura in un mondo in cui non esistevano i calcolatori stessi.

Ciononostante i computer influenzano giorno per giorno l'uomo e la società.

Mentre i progressi nelle scienze fisiche hanno attualmente a che fare con gli "estremi" (il molto piccolo, il molto veloce, il molto grande, il molto distante), la scienza dei computer si trova nel ruolo di studiare il mondo percepibile, quello che creiamo e che controlliamo. Oggi i computer sono dappertutto: nelle case, nelle auto, nelle industrie. Quanto di tutto ciò è dovuto alla scienza dei computer? Forse, per ora, ancora poco: i maggiori progressi sono stati fatti nella fisica dello stato solido, elettronica, analisi numerica e logica matematica. Ciò non deve sorprendere, perché la scienza dei calcolatori ha meno di vent'anni.

La scienza dei calcolatori: cos'è? È stata descritta in diversi modi, ma fondamentalmente le opinioni sono due. Secondo la prima, si tratta dello studio della natura e delle conseguenze dei fenomeni che avvengono nei computer e a causa dei computer. Secondo questo punto di vista, comprende sia parti teoriche che sperimentali. È un misto di ingegneria, fisica, matematica, economia e psicologia. È tuttavia meglio classificabile come attività applicativa, e non scienza nel senso classico.

Un secondo punto di vista pone al centro il concetto di programmazione. Sostenendo che tutte le informazioni che si danno ai computer, che da essi provengono, e che i computer tra loro si scambiano sono costituite da programmi, la scienza dei computer è vista principalmente come lo studio delle proprietà dei programmi astratti, cioè degli algoritmi. Non c'è dubbio che lo studio degli algoritmi sia al centro della scienza dei calcolatori e in questo campo sono stati fatti enormi progressi negli ultimi anni.

I due punti di vista diventano uno solo se si pensa che ciò che succede nei computer e a causa del computer avviene mediante algoritmi. Lo studio delle proprietà degli algoritmi assume dunque un ruolo da protagonista nell'ambito dell'informatica.

Ed è così che la pensano i nostri lettori: gli algoritmi, i programmi, il software stimolano il loro interesse perché li aiutano a capire e a controllare le cose di tutti i giorni.

Personal Software vuole fornire loro una serie di spunti, idee ed argomenti per alimentare questa loro "passione".

In quest'ottica, un argomento molto interessante è trattato in *Project Robot*, un articolo di E. Ferreguti che introduce un nuovo affascinante argomento: i giochi di programmazione. È una sfida per ognuno di voi, e un intelligente esercizio di programmazione.

Un lungo articolo di M. Spero vi illustra le caratteristiche software dello Spectrum, con particolare attenzione alla conversione dei vecchi programmi ZX81.

Con il programma di M. Pelczarski potrete creare un data base davvero efficente per l'Apple.

E infine, la prima puntata di una serie di numerosi articoli permetterà ai programmatori Basic di apprendere i misteri del Pascal. Oltre a questo, le solite rubriche, altri articoli e la sorpresa del mese: la guida al software dello ZX81. ■

E CHI MI AIUTERA' A FAR CRESCERE IL MIO GIRO D'AFFARI?

GGK

IL PERSONAL COMPUTER IBM. IL TUO PICCOLO GRANDE AMICO.

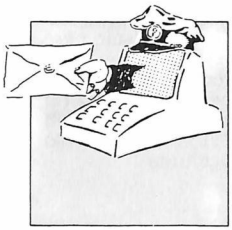
Quando gli affari aumentano, crescono le soddisfazioni, ma cresce anche la mole di lavoro. Senza una perfetta organizzazione, rischi di rimanere intrappolato.
Ma oggi c'è un amico per te, pronto a darti una mano. È il Personal Computer IBM.
Ti aiuta a snellire e risolvere tutti i problemi quotidiani della tua attività. E non solo quelli. Perchè il Personal Computer IBM può ricevere dati, calcolare, gestire l'archivio, il magazzino, la contabilità e i preventivi. E in pochissimo tempo potrai stampare tutto quello che ti serve.
Vedrai, in poche ore diventerete ottimi amici, perchè ragiona come te. Vuoi metterlo alla prova? Vai da un concessionario IBM per il Personal Computer IBM. Il tuo piccolo grande amico ti sta aspettando.

IBM Italia
Distribuzione Prodotti srl

Il Personal Computer IBM contiene un microprocessore a 16 bit e una memoria di utilizzo che raggiunge i 640 Kbyte. E, grazie ai dischi fissi, la capacità massima di memoria del sistema è di 21 Mbyte in linea. Inoltre, puoi facilmente collegarti con un altro Personal Computer IBM, con elaboratori più potenti e con la rete dei Centri Servizi Elaborazione Dati della IBM.

Sistemi operativi: DOS1-DOS2-UCSD-CP/M-86. **Supporti per le comunicazioni:** Supporto per Comunicazioni Asincrone - Supporto per Comunicazioni SDLC - Programma di Emulazione 3101 - Programma di Emulazione 3270.
Programmi applicativi: Corso Autodidattico Interattivo - Gestione Aziendale - EasyWriter (dal 20/5 anche in italiano) - Multiplan (dall'8/6 anche in italiano) - VisiCalc.

POSTA



La guida per gli autori

Possiedo un Texas TI 99/4A e sono molto interessato alla possibilità di inviarvi articoli e programmi sia riguardo a questo computer che di carattere generale. Per questo motivo, vorrei ricevere la vostra "Guida per gli autori" di cui leggo nel numero 4 della vostra rivista. Vi ringrazio per la cortese attenzione.

Giuseppe Devoto
Ferrara

Abbiamo provveduto a spedire a lei, e a tutti coloro che ce l'hanno richiesta, una copia della nostra Guida. Cogliamo l'occasione per invitare chiunque voglia scrivere un articolo, una rubrica o un programma per la nostra rivista, a richiederci la "Guida": sono un paio di fogli dattiloscritti che vi verranno spediti gratuitamente.

Listati più chiari

Spett. Redazione
colgo l'occasione per farvi i miei complimenti. Sono un ragazzo, possessore del personal Apple II, che considero uno dei migliori in commercio. Ho però una obiezione da fare, in particolare riguardo ai listati che appaiono poco leggibili. In secondo luogo, provando il programma pubblicato su *PS 4* "Grafico tridimensionale per Apple II" ho constatato che esso non funziona correttamente. Mi aspetto qualche chiarimento riguardo a questo problema. Un altro suggerimento: vi sarei grato, come penso molti altri lettori, se oltre ai dischetti dei programmi metteste in vendita anche le cassette. Saluti.

Massimo Bonaldo
Roma

È vero: talvolta i listati sono poco chiari: pur stampandoli con un nastro sempre nuovo e su carta bianca, alcune volte il processo fotografico e quindi di stampa, rendono i listati poco leggibili. Sono problemi tecnici che cerchiamo di superare. Così come è un problema tecnico quello delle cassette: anche in questo caso stiamo allestendo le apparecchiature adatte per superarlo. Infine non ci risultano errori nel listato del "Grafici tridimensionali", e dalla sua lettera non comprendiamo dove possa stare l'eventuale problema.

Chi sa programmare in Forth?

Spett. Direzione
innanzi tutto i migliori complimenti per il vostro meraviglioso giornale. A voi stanno a cuore di sicuro i problemi di noi softlettori ed allora eccovene uno. Sono un Vicpatito e ad un certo punto della mia softvita, vista la possibilità offerta dal VIC di sperimentare in modo differente ho acquistato il cartridge per programmare in Forth. Ma, sia il manuale, sia un libro comperato in seguito sono scritti in inglese ed inoltre non sono perfettamente seguibili nell'uso pratico. Il mio unico risultato in Forth è quello di eseguire operazioni aritmetiche in modo immediato e niente altro.

Se un giorno farete una rubrichetta in mezzo a tanta Basicabbondanza del tipo "Linguaggi diversi" ricordatevi anche di Forthincapaci. Vi ringrazio per l'attenzione e rinnovo i complimenti.

Vanni Cecchi
Tresenda (SO)

Uno dei nostri numerosi obiettivi è di parlare di tutti i linguaggi. Proprio in questo numero ha inizio una serie di articoli per passare dal Basic al Pascal. Certamente il Forth non è così diffuso come Basic e Pascal appunto, ma... c'è qualcosa in arrivo.

Perché non un codice a barre?

Spett.le Redazione
seguo la Vs. rivista sin dal primo numero e la ritengo, come impostazione, la più interessante tra tutte le altre del campo.

Da dicembre dell'anno scorso sono in possesso di un Commodore 64. Prima di avere un computer con cui lavorare, leggevo le riviste d'informatica "personal", dilettandomi nella analisi dei programmi che pubblicavano.

Da quando ho il C64, mi ritrovo a consumare ore ed ore nel digitare i vari listati con conseguente affaticamento e perdita d'interesse. Credo che questa sia una esperienza comune a molti appassionati di personal computing.

Senz'altro una soluzione a questo problema è rappresentata dalla vendita di programmi pubblicati, registrati su appositi supporti magnetici. Purtroppo questa soluzione comporta i seguenti svantaggi:

1) Il lasso di tempo che intercorre tra la pubblicazione della rivista

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

Personal Software
Via Rosellini 12
20124 Milano

ECCO CHI TI AIUTERA' AD ANDARE D'AMORE E D'ACCORDO CON IL TUO NUOVO AMICO.

Il tuo concessionario IBM.
Ti aiuterà a ottenere il massimo dal tuo Personal Computer IBM. Ti garantirà un'assistenza puntuale e un servizio all'altezza del nome IBM, che in tutto il mondo significa efficienza e affidabilità. Per una lunga e proficua amicizia fra te e il tuo Personal Computer IBM.

Aosta
INFORMATIQUE SAS -
Av. Du Cons. Des Commis, 16 -
11100 Aosta - Tel. 0165.2242

Bari
PASED SRL - Via Calefati, 134/136 -
70125 Bari - Tel. 080/481488

Belluno
SCP COMPUTER SYSTEM SRL -
Via Feltre, 32 - 32100 Belluno -
Tel. 0437.70826

Bergamo
NUOVA INFORMATICA SAS -
Via Provinciale, 86 - 24021 Albino -
Tel. 035.751784
SELTERING SPA - Via Verdi, 31 -
24100 Bergamo - Tel. 035.248256/7/8

Bologna
ABACO INFORMATICA SAS -
Via Bernini, 1 - 40138 Bologna -
Tel. 051.393274
C.M.B. INFORMATICA SRL -
Via Arcoveggio, 74/10 - 40129 Bologna -
Tel. 051.323594
PALAZZO DONATO - Via Emilia, 23/A -
46026 Imola - Tel. 0542.29195
SYSDATA ITALIA SPA - Via M. D'Azeglio, 58
40123 Bologna - Tel. 051.330021

Bolzano
BOPAM SAS - Via C. Battisti, 32 -
39100 Bolzano - Tel. 0471.30113

Brescia
FIN-ECO SERVICE SRL - Via G. Rosa, 34 -
25100 Brescia - Tel. 030.59055
MICROSELT SRL - Via Cipro, 33 -
25125 Brescia - Tel. 030.220391
SELTERING SPA - Via Cipro, 33 -
25125 Brescia - Tel. 030.220391

Cagliari
C.D.S. SAS - Via Sonnino, 108 -
09100 Cagliari - Tel. 070.650756

Campobasso
PUBLISISTEMI SRL -
Via S. Antonio Abate, 231 -
86100 Campobasso - Tel. 0874.98141

Como
BRUNO SRL - Via Rubini, 5 -
22100 Como - Tel. 031.260538
ZECCA INFORMATICA SPA -
Viale Dante, 14 - 22053 Lecco -
Tel. 0341.373290

Cosenza
CALIÒ SRL - Via N. Serra, 90 -
87100 Cosenza - Tel. 0984.32807

Cuneo
SISTEMI SRL - Via Giolitti, 26 -
12100 Cuneo - Tel. 0171.55475/6

Firenze
C.C.S. SAS - Viale Repubblica, 298 -
50047 Prato - Tel. 0574.580222
SAL DISTRIBUZIONE SRL -
Punto Vendita SESA - Via delle Panche, 65
50100 Firenze - Tel. 055.411635
SESA DISTRIBUZIONE SRL -
Via XI Febbraio, 24 B - 50053 Empoli -
Tel. 0571.72148

Forlì
HARD & SOFT SYSTEMS SRL -
Via Valturio, 43 - 47037 Rimini -
Tel. 0541.773343
I.C.O.T. IMPIANTI SRL - Via Codazzi, 10
47100 Forlì - Tel. 0543.723014

Frosinone
SAIU ELETTRONICA SRL -
Via Vado del Tufo, 85 - 03100 Frosinone
Tel. 0775.83093

Genova
DIFFEL SRL - Via XX Settembre, 31/4 -
16121 Genova - Tel. 010.592431

Lecce
S.V.I.C. SRL - Via V. Emanuele, 121 -
73024 Maglie - Tel. 0836.21604

Lucca
DELPHI SRL - Via Aurelia Sud, 39 -
55049 Viareggio - Tel. 0584.393068

Messina
SICIL FORNITURE SPA - Via Don Blasco, 75
98100 Messina - Tel. 090.2923987

Milano
DATA OPTIMATION SRL - Via Masaccio, 12
20149 Milano - Tel. 02.4987876
DATA PROGRESS SRL -
Via V. Emanuele, 44/A - 20059 Vimercate
Tel. 039.667423
EDICONSULT SRL - Via Rosmini, 3 -
20052 Monza - Tel. 039.389850
ELEDRA 3S SPA - Viale Elvezia, 18 -
20154 Milano - Tel. 02.349751
HOMIC PERSONAL COMPUTER SRL -
Piazza De Angeli, 3 - 20146 Milano -
Tel. 02.4988201
HUGNOT LUIGI LUCIANO -
Via De Togni, 10 - 20123 Milano -
Tel. 02.873190
MICROTECH SRL - Via F.lli Bronzetti, 20 -
20129 Milano - Tel. 02.733609
S.D.I. STUDIO DI INFORMATICA SPA -
Via G. Winckelmann, 1 - 20146 Milano -
Tel. 02.4223305
SIRIO SHOP SRL - Viale Certosa, 148 -
20156 Milano - Tel. 02.3010051
SOFTEC SRL - Viale Mayno, 10 -
20129 Milano - Tel. 02.7491196
TRANSDATA SRL -
Milano Fiori Palazzo E3 Str 1ª -
20094 Assago - Tel. 02.8242460

Modena
DATA SRL - Via B. Peruzzi, 12 -
41012 Carpi - Tel. 059.688090
DATAX SRL - Via Biondo, 6 - 41012 Carpi -
Tel. 059.698355

Napoli
POINTER SRL - Via A. De Gasperi, 45 -
80133 Napoli - Tel. 081.312312

Padova
CERVED ENGINEERING SPA -
C.so Stati Uniti, 14 - 35100 Padova -
Tel. 049.760733

Palermo
SER.COM. ITALIA SRL - Via Sciuti, 180
90144 Palermo - Tel. 091.261041
TESI SRL - Via E. Notarbartolo, 23 -
90141 Palermo - Tel. 091.260549

Pavia
I.T.C. INFORMATICA SRL -
Strada Nuova, 86 - 27100 Pavia -
Tel. 0382.303201
LOGICA INFORMATICA SRL -
Via Montegrappa, 32 - 27029 Vigevano -
Tel. 0381.81888

Perugia
PUCCIUFFICIO SNC -
Via XX Settembre, 148C - 06100 Perugia
Tel. 075.72992

Roma
CERVED SPA - Via Appia Nuova, 696 -
00100 Roma - Tel. 06.7940241
DATAOFFICE SPA - Via Sicilia, 205 -
00187 Roma - Tel. 06.4754568
ELEDRA 3S SPA - Via G. Valmarana, 63
00100 Roma - Tel. 06.8127324
GEDIN SRL - L.go D. De Dominicis, 7 -
00159 Roma - Tel. 06.432183
I.S.E.D. SPA - Via Tiburtina, Km. 12,300 -
00131 Roma - Tel. 06.4125851
JACOROSSI SPA - Via V. Brancati, 64 -
00144 Roma - Tel. 06.54916
SAPES SRL - V.le Tito Livio, 12 -
00136 Roma - Tel. 06.3453536
VALDE ADEL SRL - Piazza S. Anastasia, 3
00186 Roma - Tel. 06.6786648

Salerno
OMNIA SRL - C.so Garibaldi, 47 -
84100 Salerno - Tel. 089.353914

Siena
SILOG SISTEMI LOGICI SRL -
Via Sicilia, 5 - Belverde - 53100 Siena -
Tel. 0577.54085

Terni
DPS SRL - Via Pacinotti, 6 -
05100 Terni - Tel. 0744.58247

Torino
DIVERSIFICATE VENCO SRL -
C.so Matteotti, 32A - 10121 Torino -
Tel. 011.545525
PROGRAMMA SPA - Corso Svizzera, 185
10149 Torino - Tel. 011.746421
SISTEMI SPA - C.so Peschiera, 240 -
10139 Torino - Tel. 011.3358676
SOFTEC SRL - C.so San Maurizio, 79 -
10124 Torino - Tel. 011.8396444

Trento
SIGE SNC - COMPUTER SHOP -
Via Prato, 22 - 38100 Trento -
Tel. 0461.25154

Treviso
EDS SRL - Via S. Pio X, 154 -
31033 Castelfranco Veneto - Tel. 0423.490178
INFORMATICA TRE SRL -
Viale della Repubblica, 19 - 31100 Treviso -
Tel. 0422.65993

Trieste
DITTA MURRI - Via A. Diaz, 24/A -
34123 Trieste - Tel. 040.733253

Varese
ELMEC SPA - Via Sebenico, 12 -
21100 Varese - Tel. 0332.264135

Venezia
COMPUTIME SRL - Piazza Rizzo, 63 -
30027 S. Donà di Piave - Tel. 0421.2548

Vercelli
ANALOG SNC - Via Dionisotti, 18 -
13100 Vercelli - Tel. 0161.61105
CENTRO SERVIZI INFORMATICA
TEOREMA SRL - Via Losano, 9 -
13051 Biella - Tel. 015.24915

Verona
PRAGMA SOFTWARE SRL -
Via Carmelitani Scalzi, 20 - 37100 Verona -
Tel. 045.24629

Vicenza
ALFA DATA SRL - Via Milano, 110 -
36042 Vicenza - Tel. 0445.874199

Viterbo
ITALBYTE SRL - V.le Trento - Pal. Garbini
01100 Viterbo - Tel. 0761.221333

- E per acquisti superiori alle 20 unità puoi anche rivolgerti alle filiali IBM.
- Per ulteriori informazioni sugli indirizzi dei punti di vendita telefona a 02/21752360 oppure 06/54864962.

IBM

IBM Italia
Distribuzione Prodotti srl

e l'arrivo a casa degli eventuali dischetti o cassette ordinate;

2) Il costo reale della rivista, volendo comprare tutti i programmi interessanti, supererebbe facilmente la disponibilità di molti lettori.

Pertanto mi chiedo, anzi Vi chiedo perché non si realizza l'idea venuta alla luce in casa *BYTE* (vedi *BAR CODE LOADER* di Ken Budnick - A Paperbyte Book, 1977 BYTE Pubblications Inc.).

Mi spiego meglio. Vorrei che le riviste (tutte) pubblicassero i programmi con i listati stampati su metà pagina e il relativo BAR CODE sull'altra metà. Un'idea molto semplice che probabilmente nasconde una certa complessità per quanto riguarda la fase realizzativa, ma è senz'altro una meta raggiungibile in breve tempo dati i mezzi e le capacità di cui disponete.

L'attuazione di questa idea si può articolare nei tre punti che vado ad elencarvi:

1) Realizzazione (a cura di *P.S.* o *BIT*) di un lettore di BAR CODE, collegabile ai vari USER PORT, di prezzo accessibile;

2) Driver software su cassetta o disco da fornire insieme al lettore di BAR CODE (specifico per il S.O. indicato dall'acquirente);

3) Driver software (su supporto magnetico o in BAR CODE) per stampante per la generazione di BAR CODE da parte degli utenti con la stessa formattazione usata dalle riviste.

Ritengo che l'attuazione di questa mia proposta possa dare una svolta significativa allo sforzo di "informatizzazione popolare" tanto auspicata dalle riviste del ramo.

Pertanto, mi auguro che terrete conto delle Vs. stesse raccomandazioni, cercando la cooperazione delle altre riviste per concordare uno standard che permetta a tutti di scambiare programmi e idee.

Colgo l'occasione per porgerVi cordiali saluti sperando di aver trovato delle persone oneste e veramente interessate per il bene dell'informatica italiana.

Nicola de Laurentiis
Acerra (NA)

Le sue proposte sono senz'altro interessanti, ma di difficile attuazione. Byte *stesso, dopo l'iniziativa presa nel 1977, ha pubblicato ben pochi listati con il codice a barre. Le difficoltà sono molteplici, e riguardano innanzitutto l'accordo fra tutte le riviste, o almeno fra le maggiori; questi accordi, come saprà, sono difficili talvolta anche tra riviste dello stesso editore. Tra gruppi editoriali diversi poi, non ci si parla neppure.*

Un'altra difficoltà riguarda le attrezzature per stampare i codici a barre in modo affidabile.

Riteniamo tuttavia che questa sia la strada verso la quale ci si dovrà indirizzare: ma è una strada di domani.

Un vocabolario Basic

Spett.le Redazione
ho terminato proprio ora di sfogliare l'ultimo numero della vostra rivista. Devo farvi subito i miei complimenti per l'impegno con cui state cercando e riuscendo a rendere sempre migliore il vostro "prodotto". Io sono un tredicenne in possesso di un TRS-80 Color Computer con Extended; vi scrivo perché purtroppo fino ad ora non ho ancora trovato listati di programmi impostabili direttamente sul mio sopracitato Personal, escludendo naturalmente i programmi e le subroutine di carattere generale. Ho letto quanto rispondevate al sig. Roberto Marchetti e spero che terrete conto anche della mia lettera. Se vi interessa sarei disposto ad inviarvi i programmi che ho già implementato su detta macchina, dovete però considerare che non dispongo di alcuna stampante e che le cassette, come afferma il sig. Ernesto de Bernardis, costano un occhio; di conseguenza per trascrivere i programmi impiego parecchio tempo.

Sarei anche felice (e chi non lo sarebbe?) di ricevere per tali programmi un adeguato compenso, ad esempio un abbonamento alla vostra rivista...

Avrei anche un suggerimento da darvi; potreste in futuro, data la complessità della cosa, proporre un VOCABOLARIO BASIC, mi spiego meglio: come si sa i dialetti Basic utilizzati dai diversi Personal (soprattutto nelle istruzioni grafiche) si differenziano se pur minimamente ebbene voi potreste proporre una stessa istruzione in più traduzioni.

Tale "VOCABOLARIO BASIC" sarebbe assai utile a coloro che spesse volte rinunciano a tradurre un programma scritto per un altro computer per la presenza di istruzioni BASIC di incomprensibile funzione.

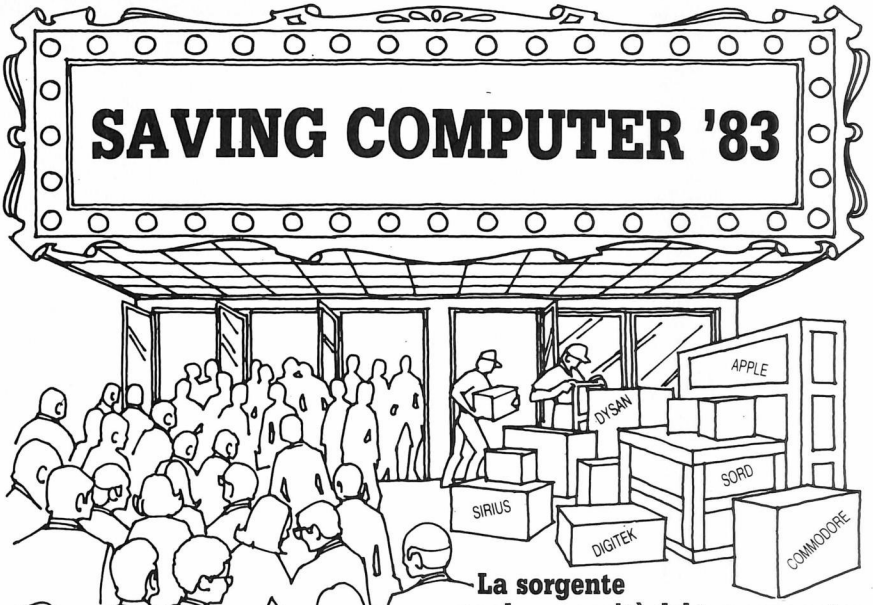
Riconfermandovi i miei complimenti vi porgo i miei saluti e auguri per il futuro.

Riccardo Matteo Giani
Milano

Abbiamo provveduto ad inviarle una copia della "Guida per gli autori" dove, come noterà, si parla anche di questioni economiche.

Per il "Vocabolario Basic" stiamo pubblicando da qualche mese la rubrica "Dizionario di Basic" che ha appunto lo scopo di cui lei parla. Naturalmente, non avendo a disposizione tutti i personal computer, citiamo solo i più diffusi. Ma raccoglieremo tutte le indicazioni dei lettori, che sollecitiamo, e probabilmente pubblicheremo il "Dizionario" completo sotto forma di volume. ■

SAVING COMPUTER '83



La sorgente per le necessità del tuo computer

Nella nostra sala mostra potrai ammirare e provare prodotti come:

- ☐ *stampanti*
- ☐ *floppy disk*
- ☐ *programmi*
- ☐ *biblioteca specializzata*

Le migliori marche di Personal Computer

Disponiamo infatti pronta consegna di APPLE II, APPLE II E, SIRIUS, SORD M 23, AVT COMP 2, VIC 20, VIC 64, ZX81, SPECTRUM, MPF II

Non perdere questa occasione!!!

DISTRIBUTORI ESCLUSIVI DEL FAVOLOSO "THE LAST ONE" PER IL VENETO

Vendita anche per corrispondenza, telefona per le quotazioni, saremo lieti di accontentarti.

SAVING ELETTRONICA

VIA GRAMSCI 40 - MIRANO (VE) - TEL. (041) 432876

Le variabili di sistema nello ZX81

Usate le informazioni della pagina zero per i vostri programmi

di *Bruno Del Medico*

La memoria RAM dello ZX81 o ZX80 nuova ROM (quando parleremo del primo intenderemo sempre anche il secondo) non è un blocco omogeneo ma è composta da diverse aree (vedi fig. 1).

L'area compresa tra gli indirizzi 16384 e 16508, comprende 125 byte che non possono essere utilizzati per memorizzare il programma.

Questa area si chiama *area delle variabili del sistema*, o anche *pagina zero*. In essa sono contenuti ed aggiornati costantemente tutti gli elementi necessari al computer per eseguire correttamente il programma. Per esempio, il numero della linea in esecuzione oppure lo stato del cursore.

Molti considerano persi questi 125 byte, per il fatto di non poterli usare come la rimanente memoria RAM. Si tratta di un errore. Un programmatore abile può utilizzare vantaggiosamente questa area della memoria principalmente in due modi:

(a) leggendo i dati che vi sono contenuti;
(b) modificando gli stessi.

In questo articolo troverete, opportunamente esemplificate, le informazioni necessarie per poter fare con profitto entrambe le cose. Una volta afferrato il principio, potrete sedervi davanti al computer con il manuale aperto alla pagina dell'Appendice B - Variabili del sistema e provare voi stessi ad escogitare diversi possibili usi per questi byte.

Leggere la pagina zero

Un test per le espansioni di memoria

I due byte degli indirizzi 16388 e 16389 contengono un numero, che equivale all'indirizzo del primo byte "non accessibile" nella memoria.

Quando accendete lo ZX81 senza espansione, disponete di 1024 byte di memoria RAM. Questi byte vengono sistemati nelle locazioni che vanno dall'indirizzo 16384 al

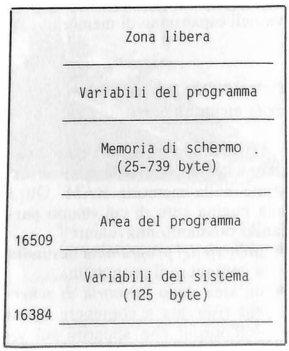


Figura 1. *Memoria RAM dello ZX81.*

17407. Nell'indirizzo 17408 non si potrà memorizzare nessun dato, perché manca il supporto fisico, cioè manca l'espansione di memoria.

L'indirizzo 17408 prende il nome di RAMTOP (punto più alto della RAM).

Se la RAM è di 4 Kb, allora avremo:

1024×4=4096
16384+4096=20480

In questo caso il valore di RAMTOP dovrà essere 20481 e potrà essere letto agli indirizzi 16388 e 16389 con una opportuna istruzione PEEK.

Se invece aggiungiamo l'espansione da 16 Kb, vengono abilitate tutte le locazioni dalla 16384 alla 32767 e il valore di RAMTOP sarà 32768.

Scrivendo l'istruzione

```
PRINT PEEK 16388+256 * PEEK
  16389
```

e premendo NEWLINE sullo schermo apparirà il seguente numero:

17408 se non è inserita alcuna espansione
20481 se è inserita l'espansione da 3 Kb
32768 se è inserita l'espansione da 16 Kb

Se il valore di RAMTOP ottenuto con questo metodo non corrisponde ad uno di quelli specificati qui sopra, c'è qualche cosa che non va nell'espansione di memoria.

Calcolo delle aree di memoria e della memoria libera

Osservando la figura 1 potete vedere che ci sono diverse aree distinte nella memoria RAM. Oltre alla pagina zero di cui stiamo parlando possiamo individuare:

- un'*area del programma* destinata a contenere il programma;
- un'area della *memoria di schermo* riservata a contenere i dati dell'output che apparirà sul video;
- un'*area delle variabili del programma* nella quale vengono immagazzinate le variabili;
- un'area libera.

Ogni linea di programma scritta occupa una certa quantità di memoria in una di queste aree, talvolta anche in più di una. Per esempio, le linee:

```
1  LET A=2
5  LET B=1
10 PRINT A;
     " E MAGGIORE DI ";B
```

occupano spazio in tre aree distinte della memoria, e precisamente:

52 byte nell'area del programma;
25 byte nella memoria di schermo (senza l'espansione);
12 byte nell'area delle variabili del programma.

Questi calcoli però sono noiosi, e del resto il microcomputer tiene sempre aggiornato il conteggio dei byte occupati nelle varie aree di memoria, conservando nella pagina zero queste informazioni.

Possiamo farne una verifica scrivendo sulla tastiera dello ZX81 senza espansione le tre linee di programma 1, 5 e 10 per le quali abbiamo indicato il numero di byte occupati.

Agli indirizzi 16396 e 16397 è contenuto un numero che rappresenta l'indirizzo del primo byte dell'area della memoria di schermo. Si può leggere questo numero con l'istruzione

```
PRINT PEEK 16396+256 PEEK *
  16397
```

Premendo NEWLINE sullo schermo appare

16566

Vecchia o nuova ROM?

Pochi sanno che esistono due versioni della ROM da 8 Kb dello ZX81. La prima versione conteneva alcuni errori abbastanza irrilevanti, relativi ai calcoli di precisione.

Il seguente programma verifica se la ROM del vostro sistema è del tipo vecchio o nuovo. Dopo circa un minuto dal RUN, appare sullo schermo un numero. Se è 854885 allora la ROM appartiene alla prima versione. Se invece il numero è 855106 allora si tratta del tipo nuovo.

```
1 REM NUOVA ROM TIPO A E TIPO B
2 FAST
3 LET B=0
4 FOR C=8191 TO 0 STEP -1
5 LET B=B+PEEK C
6 NEXT C
7 PRINT B
```

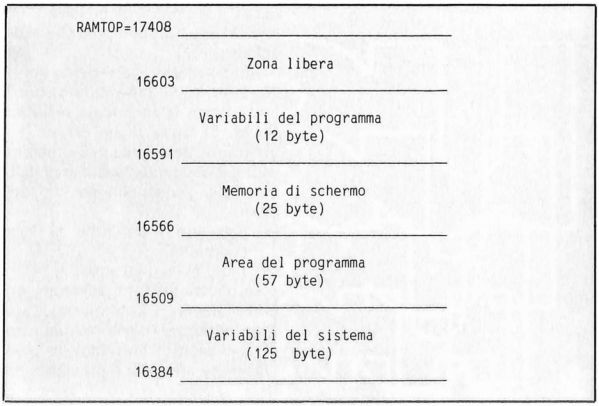


Figura 2. *Memoria RAM all'accensione dello ZX81.*

Tutti i byte compresi tra l'indirizzo 16509 (primo dell'area del programma) e 16566 (primo della memoria di schermo) costituiscono l'area del programma (vedi fig. 2).

16566−16509= 57

Agli indirizzi 16400 e 16401 è contenuto un numero che rappresenta l'indirizzo del primo byte dell'area delle variabili del programma. Per leggere questo numero si può usare l'istruzione

PRINT PEEK 16400+256* PEEK 16401

Premendo NEWLINE sullo schermo appare:

16591

Tutti i byte precedenti a questo, fino al 16566, costituiscono l'area della memoria di schermo:

16591−16566=25

Con una terza istruzione PEEK associata agli indirizzi 16404 e 16405 si può leggere l'indirizzo del primo byte non occupato dall'area delle variabili del programma:

PRINT(PEEK 16404+256* PEEK 16405)−1

Premendo NEWLINE otteniamo il numero

16603

Con la solita sottrazione determiniamo la grandezza in byte dell'area delle variabili del programma:

16603−16591=12

Tuttavia l'informazione più importante che possiamo ottenere è l'indirizzo del primo byte libero.

Scriviamo allora l'istruzione

PRINT PEEK 16404+256* PEEK 16405

Premendo NEWLINE otteniamo:

16604

Poiché non abbiamo inserito espansioni, il valore di RAMTOP è 17408, quindi la quantità di memoria ancora libera sarà data da

17408−16604=804

Comunque questi 804 byte non sono proprio tutti liberi in quanto alcuni di essi vengono utilizzati dallo ZX81 per esigenze particolari. Ricordo inoltre che quando manca l'espansione la memoria di schermo varia da 25 a 793 byte, quando invece è inserita ha un valore di 793 byte fissi.

Modificare la pagina zero

Allunghiamo lo schermo

L'output del Sinclair ZX81 viene rappresentato su uno schermo organizzato in un certo numero di linee e di colonne.

Normalmente vi sono 24 linee e 32 colonne, in pratica però possiamo usare solo 22 linee (dalla 0 alla 21), perché le ultime due in basso sono riservate al computer.

L'indirizzo 16418 contiene il numero di linee riservate. Con l'istruzione

PRINT PEEK 16418

possiamo leggere questo numero, che generalmente è uguale a 2.

Qualche volta potrebbe essere utile poter scrivere anche in queste ultime due linee, per esempio quando i dati da rappresentare hanno un formato superiore alle 22 righe.

Possiamo ottenere ciò sostituendo il 2 contenuto nell'indirizzo 16418 con un altro valore: basterà scrivere il nuovo valore mediante una istruzione POKE:

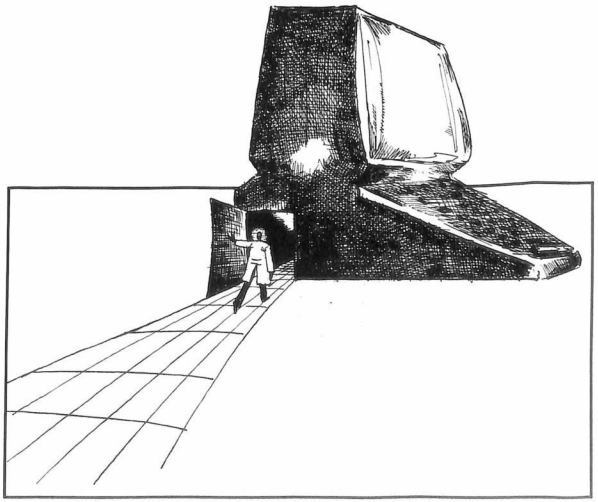
10 POKE 16418,0

Il programma del listato 1 vi convincerà che effettivamente le linee visibili diventano 24.

Ricordo che, con il byte 16418 messo a zero o a uno, non è possi-

```
10 POKE 16418,0
15 FOR W=0 TO 23
20 PRINT AT W,5;
   "QUESTA E LA LINEA ";W
22 PAUSE 25
25 NEXT W
26 STOP
34 CLS
36 GOTO 10
```

Listato 1. *Questo programma utilizza tutte le 24 righe dello schermo dello ZX81.*



```
 1 REM LETTURA DELLA MEMORIA
10 PRINT"INSERISCI L INDIRIZZO DECIMALE"
20 PRINT"DI PARTENZA"
30 INPUT IND
35 CLS
50 POKE 16418,0
55 FOR K=0 TO 23
60 PRINT TAB 6;IND;TAB 14;PEEK IND;TAB 18;CHR$(PEEK IND)
65 LET IND=IND+1
70 NEXT K
75 STOP
80 CLS
90 GOTO 50
```

Listato 2. *Programma che visualizza su schermo gruppi di 24 byte di memoria.*

bile usare istruzioni di tipo INPUT o SCROLL: se lo si fa il sistema va in *crash*.

Per ottenere successive visualizzazioni non avremmo potuto usare perciò nel programma del listato 1 le istruzioni tradizionali:

```
26 PRINT "PER CONTINUARE PREMI NEWLINE"
30 INPUT A$
34 CLS
36 GOTO 10
```

sia perché non c'è più spazio per scrivere il contenuto della stringa della linea 26, sia perché non è possibile usare il comando INPUT. Possiamo invece usare l'istruzione:

```
26 STOP
34 CLS
36 GOTO 10
```

ed otterremo una successiva visualizzazione premendo il tasto CONT. Volendo usare l'istruzione INPUT, dobbiamo rimettere provvisoriamente a 2 il byte 16418 con l'istruzione:

```
POKE 16418,2
```

e poi rimetterlo a zero una volta eseguito l'INPUT.

Nel listato 2 è possibile vedere una applicazione pratica, che consente di leggere la memoria del computer visualizzando sullo schermo il contenuto di 24 byte per volta. Alla richiesta:

INSERISCI L'INDIRIZZO DECIMALE DI PARTENZA

si deve rispondere con un numero da 0 a 65535. Per vedere come viene memorizzato il programma provate ad inserire il numero 16509.

Ricordate che per ottenere successive visualizzazioni dovete premere il tasto CONT.

```
10 REM SCROLL PARZIALE MULTIPLO
20 FOR K=0 TO 19
30 PRINT TAB 10;"LINEA ";K
40 NEXT K
45 FOR W=0 TO 19
48 PRINT AT W,20;W
49 NEXT W
50 PRINT AT 21,0;"SCROLL PARZIALE. QUALE LINEA?"
60 INPUT G
70 POKE 16418,23-G
80 SCROLL
90 POKE 16418,2
100 GOTO 45
```

Listato 3. *Programma che esegue uno scroll parziale da una linea data.*

Scroll parziale

Se lo schermo può essere allungato, può anche essere ristretto modificando il contenuto dell'indirizzo 16418.

Con le istruzioni

```
10 POKE 16418,24
20 PRINT "CIAO"
```

si otterrebbe un segnale di errore, perché alla linea 10 tutte le 24 linee dello schermo sono state riservate al microcomputer.

Naturalmente questo è un caso limite di cui appare abbastanza improbabile un qualche uso pratico.

Invece è possibile utilizzare la possibilità di riservare al microcomputer N linee, facendo eseguire lo scroll dalla linea N+1 in su. Soltanto la parte di disegno dalla linea N in su subisce l'effetto del comando SCROLL, mentre la parte inferiore rimane invariata.

Riposizionando successivamente a valori diversi il byte 16418, diventa possibile eseguire diversi SCROLL che interessano solo una parte dello schermo.

Il listato 3 costituisce un ottimo esempio. Quando l'output viene visualizzato, la colonna più a destra non sembra scorrere perché viene riscritta ogni volta al fine di facilitare il controllo. Le due colonne centrali scorrono partendo dal numero di linea che si può indicare di volta in volta.

Il controllo del tempo

La pagina zero fornisce un utilissimo strumento per il controllo del tempo: infatti gli indirizzi 16436 e 16437 contengono un numero che varia in funzione della quantità di fotogrammi che scorrono sullo schermo.

Il listato 4 è un esempio della utilizzazione di questi due byte.

Con le linee:

```
140 POKE 16436,255
141 POKE 16437,255
```

scriviamo negli indirizzi suddetti il numero 65535.

Da quel momento in poi il numero si decrementa tante volte

```
 5 GOTO 50
10 PRINT"++++++++++++++++++++++++++++++++"
20 PRINT TAB 11;"RIFLESSI"
30 PRINT"++++++++++++++++++++++++++++++++"
40 RETURN
50 GOSUB 10
60 PRINT AT 10,0;"DEVI PREMERE NEWLINE"
61 PRINT"QUANDO APPARE SULLO SCHERMO"
62 PRINT"IL QUADRATINO NERO"
80 PRINT
85 PRINT"TI DARO IL TEMPO DI RISPOSTA"
86 PRINT"IN SECONDI"
87 PRINT AT 21,0;"PER COMINCIARE PREMI N/L"
90 INPUT A$
95 CLS
100 GOSUB 10
110 LET T=0
120 PAUSE INT(RND*500)
121 IF RND>.49 THEN PRINT AT 10,0;"ATTENZIONE: STAI PRONTO"
122 PAUSE INT(RND*20)+50
140 POKE 16436,255
141 POKE 16437,255
142 INPUT A$
150 LET T=T+65535-PEEK 16436-256*PEEK 16437
170 CLS
175 GOSUB 10
180 PRINT AT 10,0;"TEMPO DI RISPOSTA"
182 PRINT TAB 5;T/50;" SECONDI"
190 PRINT AT 21,0;"PER UN ALTRA PROVA PREMI N/L"
200 INPUT A$
210 CLS
220 GOTO 100
```

Listato 4. *Questo programma utilizza le informazioni della pagina zero per il controllo del tempo.*

quanti sono i fotogrammi che scorrono sullo schermo, cioè in ragione di 50 unità per secondo.

Con la linea:

150 LET T=T+65535−PEEK 16436−256*PEEK 16437

si stabilisce che la variabile T (numero dei fotogrammi passati) è uguale a 65535 meno il numero letto nei due indirizzi.

Nella linea 182 dividendo T per il valore fisso di 50 si ottiene il tempo trascorso in secondi.

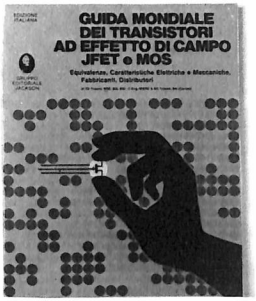
Il programma controlla il tempo impiegato dall'operatore per premere il tasto NEWLINE quando appare sullo schermo il quadratino nero del cursore. Il tempo viene espresso con i decimali in quanto la divisione T/50 della linea 182 non richiede un risultato intero.

In questo programma lo ZX81 cerca di trarre in inganno il giocatore per fargli premere il tasto fuori tempo. ■

SEMICONDUCTORS REPLACEMENT GUIDE

Conoscere subito l'esatto equivalente di un transistore, di un amplificatore operazionale, di un FET, significa per il tecnico, il progettista, l'ingegnere, come pure per l'hobbista, lo studente, il ricercatore, risparmiare tempo, denaro e fatica.

Può darsi però che occorra di un dispositivo conoscere le caratteristiche elettriche e meccaniche, oppure soltanto chi lo produce, o dove reperirlo in tutta sicurezza, oppure riuscire ad identificarne i terminali, o i campi di applicazione. Tutto questo è quanto Vi forniscono queste tre Guide, veramente "mondiali", non solo perché i dispositivi elencati sono europei, americani, giapponesi, inglesi o, persino russi, ma anche nel numero presentato: oltre 20.000 transistori, 5.000 circuiti integrati lineari e 2.700 FET.



GUIDA MONDIALE DEI TRANSISTORI
Oltre **20.000** transistori
Codice 607H - Pagg. 286 - Formato 21 x 26,5
L. 23.000

GUIDA MONDIALE DEGLI AMPLIFICATORI OPERAZIONALI
Oltre **5.000** circuiti integrati lineari
Codice 608H - Pagg. 196 - Formato 21 x 26,5
L. 17.000

GUIDA MONDIALE DEI TRANSISTORI AD EFFETTO DI CAMPO JFET E MOS
Oltre **2.700** FET
Codice 609H - Pagg. 80 - Formato 21 x 26,5
L. 11.500

PREZZO SPECIALE PER LA COLLANA COMPLETA
Codice 610H - **L. 35.000** (abbonati L. 31.500)

GRUPPO EDITORIALE JACKSON
Divisione Libri

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

Project robot: il robotwar per lo ZX81

di *Enrico Ferreguti*

Prendendo spunto dall'articolo "Robotwar, il gioco-soft dell'era informatica", apparso su Bit *36, l'autore ha realizzato questo programma per lo ZX81.*

Si tratta di un gioco di programmazione, il cui obiettivo è programmare *un robot in modo da renderlo autonomo ed in grado di far fronte a qualsiasi situazione. Lo scenario ipotetico è un combattimento tra robot, ognuno programmato in modo diverso, e che si affrontano secondo le istruzioni del loro "computer di bordo".*

Un gioco di programmazione di questo tipo, consiste di due fasi: la prima è dedicata alla stesura del programma di controllo dei robot; la seconda consiste nell'osservazione della battaglia: in questa fase i robot agiscono da soli, e la loro sorte dipende dal programma che li governa.

Questa versione per lo ZX81 naturalmente risente di alcune limitazioni imposte dalla macchina: la ridotta capacità di memoria, l'uso del Basic. Costituisce tuttavia un intelligente esercizio di "arte della programmazione".

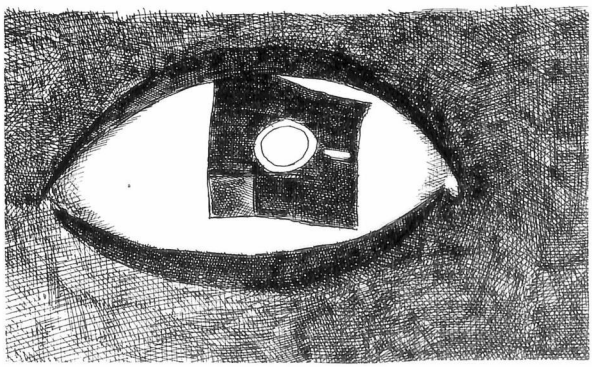
Il programma che proponiamo è un "gioco di programmazione" ispirato a *Robotwar*. Non si tratta del classico gioco d'azione in cui contano i riflessi e l'abilità manuale, bensì di una sfida logica, che impegna la capacità di programmare un robot, mettendolo in condizione di poter affrontare e risolvere ogni situazione.

Possono giocare fino a tre giocatori: dapprima ogni giocatore programmerà il suo robot. Nella seconda fase i tre robot "scenderanno" nell'arena per combattersi.

Il programma comprende due parti: la parte di gestione, e la parte tattico-strategica. Quest'ultima è completamente lasciata al programmatore, che che deve utilizzare le *variabili di stato* e le *routine attive* che la parte di gestione gli mette a disposizione.

Le routine attive

Le routine che il sistema fornisce per il controllo del robot sono tre: la routine LASER per l'offesa, la routine RADAR per la ricerca dell'obiettivo e la routine MUOVI per il movimento del robot. Sono memorizzate come subroutine proprio per mantenere la separazione tra gestione e tattica di cui abbiamo parlato prima; basta infatti chiamare la routine desiderata per ottenere ciò che si vuole senza preoccu-



parsi degli effetti che questa può provocare. Per chiamare una routine bisogna battere la parola chiave GOSUB e quindi il nome come illustrato negli esempi dei programmi di combattimento. Alcune spiegazioni:

1. LASER Chiamando questa routine viene emesso il laser nella direzione dichiarata nella variabile DIR di cui parleremo in seguito, che può danneggiare e distruggere un altro robot avversario. Per distruggere un robot bisogna colpirlo quattro volte: si può modificare questo limite cambiando il 3 in riga 305.

2. RADAR Serve a sondare nella direzione di DIR lo spazio intorno al robot per capire se in quella direzione c'è qualche robot

```
   1 REM ********************
   2 REM *   PROJECT  ROBOT   *
   3 REM *                    *
   4 REM *PERSONAL  SOFTWARE*
   5 REM ********************
  10 LET ZA=0
  11 LET ZB=0
  12 LET ZC=0
  13 LET ZD=0
  14 LET ZE=0
  15 LET ZF=0
  16 LET ZG=0
  17 LET ZH=0
  18 LET ZI=0
  19 LET ZJ=0
  20 LET ZK=0
  21 LET ZL=0
  22 LET ZM=0
  23 GOTO 100
  50 PRINT AT 1,1;"NOME ROBOT"
  51 FOR K=1 TO 3
  52 INPUT Z$(K)
  54 PRINT ,,TAB 1;K;")  ";Z$(K);
  56 LET Z(K,5)=CODE Z$(K,1)
  58 IF Z(K,5)=136 OR Z(K,5)=0 T
HEN LET Z(K,5)=CODE "X"
  60 PRINT "-->";CHR$ Z(K,5)
  62 NEXT K
  64 IF Z(1,5)=Z(2,5) OR Z(1,5)=
Z(3,5) THEN GOTO 80
  66 IF Z(2,5)=Z(1,5) OR Z(2,5)=
Z(3,5) THEN GOTO 80
  68 IF Z(3,5)=Z(1,5) OR Z(3,5)=
Z(2,5) THEN GOTO 80
  70 PRINT AT 13,0;">>PROJECT  R
OBOT<<","*****************",,,"
IN GARA I ROBOT",,,,Z$(1),,Z$(2)
,,Z$(3)
  72 RETURN
  80 PRINT ,,"INIZIALI UGUALI: R
IPETI"
  84 FOR K=1 TO 20
  85 NEXT K
  86 CLS
  87 GOTO 50
 100 DIM Z(3,5)
 105 DIM Z$(3,10)
 110 LET NOLASER=1
 115 LET DISPLAY=1+PEEK 16396+25
6*PEEK 16397
 120 LET N=-33
 125 LET O=1
 130 LET S=33
 135 LET E=-1
 140 LET NO=-32
 145 LET SO=34
 150 LET SE=32
 155 LET NE=-34
 160 FOR K=1 TO 3
 165 LET Z(K,2)=N
 170 LET Z(K,3)=0
 180 NEXT K
 185 GOSUB 50
 190 FOR K=1 TO 3
 195 LET Z(K,1)=DISPLAY+33*INT (
RND*10+1)+INT (RND*10+1)
 200 NEXT K
 210 LET LASER=500
 215 LET RADAR=650
 220 LET MUOVI=750
 250 PRINT AT 12,0;"████████████████████████████████
"
 255 FOR K=11 TO 1 STEP -1
 260 PRINT AT K,0;"█                              █
"
 265 NEXT K
 270 PRINT AT 0,0;"████████████████████████████████
"
 300 FOR R=1 TO 3
 305 IF Z(R,3)>3 THEN GOTO 360
 310 POKE Z(R,1),Z(R,5)
 315 LET DIR=Z(R,2)
 320 LET DANNI=Z(R,3)
 330 LET AZIONI=3
 335 LET Q=Z(R,1)-DISPLAY
 340 LET POSY=INT (Q/33)
 345 LET POSX=Q-POSY*33
 350 GOSUB 6000+R*1000
 355 LET Z(R,2)=DIR
 360 NEXT R
 365 GOTO 300
 500 REM LASER
 505 IF AZIONI=0 THEN RETURN
 510 LET CONT=0
 515 LET AZIONI=AZIONI-1
 520 LET POSLASER=Z(R,1)+DIR
 540 IF PEEK POSLASER<>0 THEN GO
TO 570
 545 POKE POSLASER,CODE "."
 550 LET CONT=CONT+1
 555 LET POSLASER=POSLASER+DIR
 560 POKE POSLASER-DIR,0
 565 GOTO 540
 570 IF PEEK POSLASER<>136 THEN
GOTO 585
 575 LET Z(R,4)=Z(R,4)-CONT
 580 RETURN
 585 LET M=PEEK POSLASER
 590 POKE POSLASER,0
 595 IF M<>Z(1,5) THEN GOTO 610
 600 LET Z(1,3)=Z(1,3)+1
 605 GOTO 575
 610 IF M<>Z(2,5) THEN GOTO 625
 615 LET Z(2,3)=Z(2,3)+1
 620 GOTO 575
 625 LET Z(3,3)=Z(3,3)+1
 630 GOTO 575
 650 REM RADAR
 655 IF AZIONI=0 THEN RETURN
 660 LET CONT=0
 665 LET ECO=0
 670 LET AZIONI=AZIONI-1
 675 LET POSECO=Z(R,1)+DIR
 680 IF PEEK POSECO<>0 THEN GOTO
 700
 685 LET CONT=CONT+1
 690 LET POSECO=POSECO+DIR
 695 GOTO 680
 700 IF PEEK POSECO=136 THEN RET
URN
 705 LET ECO=CONT
 710 RETURN
 750 REM MUOVI
 755 IF AZIONI=0 THEN RETURN
 760 LET AZIONI=AZIONI-1
 765 IF PEEK (Z(R,1)+DIR)<>0 THE
N RETURN
 770 POKE Z(R,1),0
 775 LET Z(R,1)=Z(R,1)+DIR
 780 POKE Z(R,1),Z(R,5)
 785 LET Q=Z(R,1)-DISPLAY
 790 LET POSY=INT (Q/33)
 795 LET POSX=Q-POSY*33
 800 RETURN
5000 REM NELLE RIGHE SEGUENTI
         SI TROVANO I PROG.
         DI COMBATTIMENTO DEI
         VARI ROBOT:
         ROBOT 1)-->RIGA 7000
         ROBOT 2)-->RIGA 8000
         ROBOT 3)-->RIGA 9000
```

Listato 1. *Project Robot.*

nemico. Nel caso la ricerca sia positiva, il sistema fornisce nella variabile speciale di ritorno ECO la distanza tra il robot nemico e il radar.

3. MUOVI La routine fa muovere verso il punto cardinale specificato in DIR il mezzo. Ovviamente se il robot trova il muro o un altro mezzo non può effettuare lo spostamento.

Per impedire un uso troppo prolungato di questi sottoprogrammi, che potrebbe precludere il gioco agli altri robot, si possono chiamare solo tre volte ad ogni lettura del programma di combattimento, assicurato dalla variabile AZIONI. Se si volesse cambiare il numero delle chiamate basta cambiare il valore in riga 330.

Le variabili di stato

Abbiamo quattro variabili di stato utilizzabili dal programma di guida in ogni momento e sono: DIR, DANNI, POSX, POSY.

1. DIR Significa direzione ed indica il verso in cui è orientato il robot. È sicuramente la variabile più importante perché permette al robot di muoversi, sparare, sondare con il radar nella direzione indicata.

Come l'occhio esperto di qualcuno avrà notato, le figure dei robot non sono stampate mediante PRINT AT bensì vengono "pokate" all'interno del display file, questo proprio per consentire un uso veloce ed immediato della DIR. È sufficiente infatti attribuire alla DIR l'iniziale o le iniziali del punto cardinale verso cui si vuole girare il mezzo (p. es. E, NE, S, NO, ecc.). Se si osservano le linee da 120 a 155 si trova la spiegazione di quanto detto, infatti le variabili che rappresentano il punto cardinale contengono i valori che sommati alla locazione dove era stampato il robot lo spostano nella direzione voluta (p. es. −33 fa spostare di una riga in su cioè a nord, −1 sposta ad ovest, 34 sposta a sud-est).

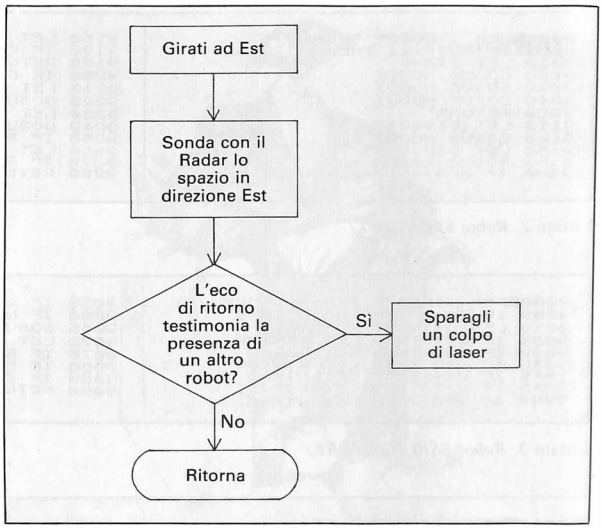
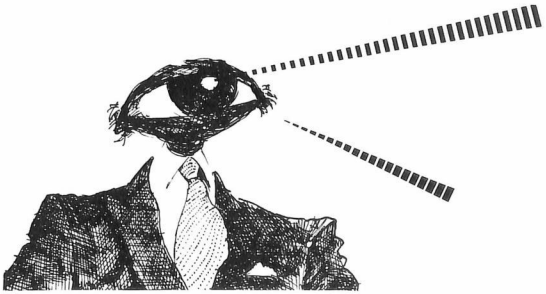


Figura 1.



2. DANNI In questa variabile sono contenuti i danni che il robot ha dovuto sopportare in seguito ad attacchi di altri robot (quando il suo valore arriva a 4 il robot viene eliminato dal gioco). È molto utile per capire se in un certo momento il robot è sotto il tiro dell'avversario.

3 e 4. POSX e POSY Sono usate per la memorizzazione della posizione sull'asse delle X e delle Y del robot (1,1 è la posizione in alto a sinistra mentre 11,11 è quella in basso a destra).

Attenzione: l'unica variabile che si può sia leggere che scrivere è la DIR mentre tutte le altre possono venire solo lette.

I programmi di combattimento

Come in qualsiasi programma scritto in qualsiasi linguaggio, bisogna aver ben chiaro l'obiettivo che ci si propone. In questo caso l'obiettivo è una tattica vincente.

Per esempio potremmo fare un

```
9000 REM >>ROBOT SFONDO<<
9005 IF POSY=11 THEN GOTO 9200
9010 LET DIR=S
9020 GOSUB MUOVI
9030 GOSUB MUOVI
9040 GOSUB MUOVI
9050 RETURN
9110 LET DIR=O
9120 GOSUB MUOVI
9130 LET DIR=N
9140 GOSUB RADAR
9150 IF ECO<>0 THEN GOSUB LASER
9160 LET DIR=O
9170 IF POSX=10 THEN LET DIR=E
9180 RETURN
9200 IF DIR=O THEN GOTO 9100
9210 LET DIR=E
9220 GOSUB MUOVI
9230 LET DIR=N
9240 GOSUB RADAR
9250 IF ECO<>0 THEN GOSUB LASER
9260 LET DIR=E
9270 IF POSX=1 THEN LET DIR=O
9280 RETURN
```

Listato 2. *Robot SFONDO.*

```
9000 REM >>ROBOT ESPLORATORE<<
9005 IF ZA=1 THEN GOTO 9065
9010 LET W=1+INT (RND*8)
9025 IF W=1 THEN LET DIR=NO
9030 IF W=2 THEN LET DIR=O
9035 IF W=3 THEN LET DIR=SO
9040 IF W=4 THEN LET DIR=S
9045 IF W=5 THEN LET DIR=SE
9050 IF W=6 THEN LET DIR=E
9055 IF W=7 THEN LET DIR=NE
9060 IF W=8 THEN LET DIR=N
9065 GOSUB RADAR
9070 LET ZA=0
9075 IF ECO<>0 THEN LET ZA=1
9080 IF ECO<>0 THEN GOSUB LASER
9085 IF ECO=0 THEN GOSUB MUOVI
9090 RETURN
```

Listato 3. *Robot ESPLORATORE.*

```
9000 REM >>ROBOT ANGOLO<<
9005 IF POSX=1 AND POSY=1 THEN G
OTO 9040
9010 LET DIR=E
9015 IF POSX<>1 THEN GOSUB MUOVI
9016 IF POSX<>1 THEN GOSUB MUOVI
9020 LET DIR=N
9025 IF POSY<>1 THEN GOSUB MUOVI
9026 IF POSY<>1 THEN GOSUB MUOVI
9030 RETURN
9040 LET DIR=S
9045 GOSUB RADAR
9050 IF ECO<>0 THEN GOSUB LASER
9055 LET DIR=O
9060 GOSUB RADAR
9065 IF ECO<>0 THEN GOSUB LASER
9070 RETURN
```

Listato 4. *Robot ANGOLO.*

programma per un robot che sta fermo e quando un altro robot gli passa davanti gli spara un colpo di laser. Per organizzare il problema è utile ricorrere al diagramma di flusso di figura 1.

Scriviamo il programma:

1. Bisogna girarsi ad est, quindi basta specificare nella variabile DIR la direzione:

LET DIR=E

2. Il secondo punto consiste nel sondare con il radar nella direzione DIR:

GOSUB RADAR

3. Il risultato dell'indagine compiuta dal radar è contenuto nella variabile ECO che riporta la distanza tra il radar ed il robot, quindi, se c'è un avversario nella fascia est, questa variabile sarà diversa da zero e il sistema verrà informato di sparare con il laser verso questa direzione. Traducendo scriviamo:

```
IF ECO<>0 THEN GOSUB LA-
   SER
```

4. Il nostro programma chiama solo due volte le routine attive, precisamente una volta la routine RADAR e, in caso il confronto sia positivo, la routine LASER. Siamo quindi liberi di ridare il controllo al programma principale con un RETURN. Riassumendo, il programma si presenta così:

```
LET DIR=E
GOSUB RADAR
IF ECO <> 0 THEN GOSUB
   LASER
RETURN
```

Nella memoria del calcolatore c'è posto per 3 programmi per altrettanti robot, a partire dalle righe 7000, 8000, 9000.

Per facilitare la stesura degli algoritmi di combattimento, alle righe 10-22 del programma principa-

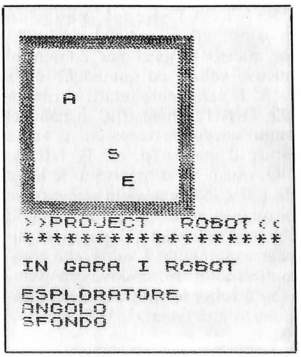


Figura 2.

le vengono inizializzate alcune variabili. Come esempio vengono forniti tre programmi completi per altrettanti robot:

SFONDO: all'inizio della battaglia il robot si muove sull'ultima riga in basso e comincia ad oscillare da un vertice all'altro sparando con il laser ogni volta che un robot gli si para davanti.

ESPLORATORE: si muove verso una posizione casuale e, se in quella direzione c'è un robot, gli spara finché non sparisce dalla sua visuale o viene distrutto.

ANGOLO: all'inizio della battaglia si muove subito nell'angolo in alto a sinistra e se un robot si posiziona sulla prima riga o sulla prima colonna del campo di battaglia gli spara un colpo. È il robot meno vulnerabile, poiché l'unico punto debole è sulla diagonale.

Nella stesura di un programma non ci si deve preoccupare del muro che, essendo indistruttibile, assorbe sia i laser che il radar e non permette l'attraversamento.

Possibili modifiche

Il programma può essere facilmente modificato. Anzitutto si può ampliare il campo di gioco mediante le righe 250-270 e modificando i valori in riga 195, facendo attenzione però ad usare gli stessi caratteri per la cornice del disegno.

Poi, è possibile modificare il numero dei robot cambiando il 3 nelle righe 51-100-101-160-190-300 e il 6000 in riga 350.

Infine, è possibile modificare o aumentare le routine e le variabili di stato.

Un'interessante proposta

L'obiettivo del programma che vi proponiamo è, come avrete capito, quello di stimolare la formulazione di buoni algoritmi in Basic. Usandolo, l'utente affina le tecniche e le tattiche e escogita trucchi che rendano il suo robot più perfetto e potente possibile.

La nostra speranza è che chi utilizzerà questo programma si cimenti nella creazione di uno o più programmi di combattimento che potremo far competere con quelli scritti da noi.

Naturalmente dovrà rispettare alcune regole: sono ammesse solo le istruzioni LET, GOTO, GOSUB, IF, THEN, RND, OR, AND; non può essere né letta né scritta la matrice Z contenente dati fondamentali sui robot come i danni e la locazione di stampa.

Variabili principali

Z(3,5)	Matrice contenente la locazione di stampa, i danni, il carattere del robot e la sua direzione
Z$(3,10)	Vettore contenente i nomi dei robot
DISPLAY	Locazione di inizio del display file
K	Contatore
LASER	Riga di inizio della routine LASER
RADAR	Riga di inizio della routine RADAR
MUOVI	Riga di inizio della routine MUOVI
R	Contatore
DIR	Variabile temporanea contenente la direzione
DANNI	Variabile temporanea contenente i danni
AZIONI	Contiene il numero di volte che è possibile chiamare le routine attive ad ogni lettura del programma utente
POSX	Variabile temporanea contenente la posizione sull'asse delle X del robot
POSY	Variabile temporanea contenente la posizione sull'asse delle Y del robot
CONT	Contatore
POSLASER	Variabile usata dalla routine LASER
ECO	Distanza tra radar e robot individuato
POSECO	Variabile usata dalla routine RADAR. ■

SINCLAIR ZX81

```
D9C5D5E52A0C4016
16CDE7021401007C
1E751CF5F1152862
0E007EBB2001237A
FE013E042801AF5
D96F1F572620D3FB
DBFB1730FBD97EBB
282DE5B7173002C6
8026006F29290917
7EE130012FD94F06
087DCB011F5FDBFB
1F30FB7BD3FB10F1
25D928032318CFD9
7AD3FBDBFB1738FB
D90CCB5923209DE1
18A5D93E04D3FBE1
D1C1D9CD0702C9
FFFFE1DDDDDDE0FF
DFDFC3DDDDDDC3FF
FFFFE3DDDFDFE1FF
FDFDE1DDDDDDE1FF
FFFFE3DDC3DFE1FF
FFF1EFC3EFEFEFFF
FFFFE1DDDDE1FDE3
DFDFC3DDDDDDDDFF
F7FFE7F7F7F7E3FF
FFFBFFFBFBFBDBE7
FFDFDBB7CFD7DBFF
E7F7F7F7F7F7E3FF
FFFFCBD5D5D5D5FF
FFFFC3DDDDDDDDFF
FFFFE3DDDDDDE3FF
FFFFC3DDDDC3DFDF
FFFFC7B7B7C7F5F3
FFFFD3CFDFDFDFFF
FFFFE3DFE3FDE3FF
FFFFDDDDDDDDE3FF
FFFFDDDDEBEBF7FF
FFFFDDD5D5D5EBFF
FFFFDDEBF7EBDDFF
FFFFDDDDDDE1FDE3
FFFFC1FBF7EFC1FF
21001EE511007C01
0002C5EDB0C1E1ED
B021094111307F01
D000EDB021624011
007B016700EDB0CD
230FCDE503
```

Listato 2. *Linee esadecimali da caricare con il programma del listato 1 (ogni linea consiste di 16 cifre esadecimali).*

dovrà fermarsi automaticamente (se non si ferma avete sicuramente sbagliato qualcosa), cancellate tutte le linee da 20 a 80 e inserite 20 LET L=USR 16857.

A questo punto salvate il programma su nastro. Non premete assolutamente RUN. Appena il programma sarà stato salvato battete POKE 16389,123, quindi NEW.

Questo serve ad abbassare la RAMTOP per far posto al programma. Caricate il programma appena salvato e premete RUN, lo schermo impallidirà per circa 2 secondi e apparirà il cursore "K". Sembrerà che il programma si sia cancellato da solo, invece si sarà trasferito in un'altra parte della memoria.

Il programma di listato 3 mostra come usare i caratteri minuscoli. Il comando in linea 130 copia tutto lo schermo trasformando le lettere inverse in caratteri minuscoli.

```
 50  FOR A=0 TO 63
 60  PRINT CHR$ A;" "
 70  NEXT A
 80  PRINT
 90  PRINT
100  FOR A=128 TO 191
110  PRINT CHR$ A;" "
120  NEXT A
130  LET L=USR 31488
```

Listato 3. *Visualizzazione dei caratteri minuscoli.*

Un'altra possibilità del programma è il cambiamento del numero di linee da copiare battendo POKE 31496,X dove X è un numero tra 1 e 24.

È importante ricordare che prima di caricare il programma è necessario abbassare la RAMTOP usando POKE 16389,123 e NEW.■

Guida alla conversione dallo ZX81 allo ZX Spectrum

Tutti i segreti per far funzionare i vecchi programmi ZX81 sul vostro nuovo Spectrum

di *Marcello Spero*

Una vera marea di programmi, routine e listati vari per lo ZX81 è oggi a disposizione di un numero sempre crescente di appassionati. Con la sospirata comparsa sul mercato italiano dello ZX Spectrum, che si propone come continuazione e logico ampliamento della via aperta dallo ZX81, moltissimi si sono domandati se il salto di qualità da una macchina all'altra sarebbe loro costato la rinuncia a tutto il software così faticosamente raccolto. Siamo di fronte alla prospettiva di iniziare da zero o possiamo partire con qualche vantaggio?

Bene, tirate pure un sospiro di sollievo, come l'ho tirato io quando mi sono reso conto che ancora una volta Clive Sinclair ha fatto le cose nel migliore dei modi. Infatti il Basic dello Spectrum si colloca in un felice punto di mezzo fra quello dello ZX81 e i vari Basic "standard".

Questo significa anzitutto che i programmi semplici, che non coinvolgono manipolazioni della memoria o delle variabili di sistema, e non usano un output di tipo grafico, possono essere trascritti così come sono dallo ZX81; d'altra parte gli stessi programmi, con poche modifiche, possono essere portati al livello di funzionalità e strutturazione dei Basic più evoluti.

Lasciando da parte le considerazioni sull'uso della tastiera, che se all'inizio può apparire complesso si rivela presto comodo e piuttosto veloce, passiamo ad esaminare le varie istruzioni e funzioni dei due Basic e le loro eventuali differenze. Ho seguito l'ordine alfabetico per rendere semplice una eventuale ricerca, saltando comunque i casi di completa uguaglianza, nonché quelli che, non avendo alcun corrispondente nello ZX81 (per esempio le istruzioni relative ai colori), possono essere inseriti a piacere. Ricordo che all'accensione lo schermo dello Spectrum si presenta esattamente come quello dello ZX81, cioè a sfondo bianco e caratteri neri, consentendo così a programmi che non richiedono un abbellimento di tipo grafico (programmi di calcolo) di girare tali e quali, senza aggiunta di istruzioni per i colori.

Istruzioni

CLEAR

Usato così com'è non presenta alcuna differenza. Nella forma CLEAR *n* sostituisce la macchinosa procedura di abbassamento della RAMTOP; infatti ponendo in *n* il nuovo indirizzo per la RAMTOP questa viene spostata e resa subito operativa, senza bisogno di effettuare NEW. Questa operazione può quindi essere effettuata da programma, senza che questo debba essere cancellato. Se non è pos-

sibile collocare la RAMTOP dove è stato richiesto, si incorre nell'errore M.

CONTINUE

Può ora essere usato per far ripartire un programma dopo una interruzione: infatti lo schermo non viene cancellato.

DATA, READ, RESTORE

Pur non avendo corrispondente per lo ZX81, queste istruzioni possono essere usate per eliminare lunghe colonne di istruzioni LET, o vettori di dati, nel caso questi siano sempre gli stessi, consentendo l'operazione di CLEAR senza che vadano perduti.

DEF FN

Può vantaggiosamente sostituire quelle subroutine in cui compaiono solo assegnazioni e calcoli (cioè istruzioni LET), nonché condizioni, a patto che dopo il THEN siano contenuti solo altri calcoli o assegnazioni.

Per esempio, una subroutine del tipo:

```
2000 LET r=SQR (s+5)
2010 LET t=INT u
2020 LET v=r+t
2030 IF w>y THEN LET v=r/t
2040 RETURN
```

che oltretutto richiede che nel programma chiamante le variabili si chiamino sempre s, u, w ed y, con la conseguente necessità, nel caso vengano usate variabili diverse, di conversioni del tipo:

```
100 LET a=s
110 LET b=u
120 LET c=w
130 LET d=y
```

può ridursi a:

```
DEF FN v(s,u,w,y)=(SQR
(s+5)+INTu)*(w<=y)+
(SQR(s+5)/INT u)*(w>y)
```

che potrà essere utilizzata anche con variabili diverse, rispettando semplicemente l'ordine in cui dovranno essere sostituite:

```
200 PRINT FN v(a,b,c,d)
```

a verrà utilizzata al posto di *s*, *b* al posto di *u*, e così via; l'istruzione IF è stata trasformata in condizione implicita: vengono eseguiti i calcoli relativi ad ambedue i casi, ciascuno moltiplicato per la sua condizione; quando questa sarà verificata avverrà una moltiplicazione per 1, altrimenti per 0.

Analogo discorso, con le medesime possibilità e limitazioni, può essere fatto nel caso di stringhe; nell'istruzione:

```
DEF FN a$(v$,k$)= ...
```

possono essere contenute tutte le operazioni possibili sulle stringhe, con le relative condizioni.

Gli argomenti di una funzione dovranno sempre concordare con quelli della corrispondente DEF FN per ordine e tipo (numerico, stringa).

IF...THEN...

Niente di nuovo quanto alla sua forma; nel caso di questa istruzione, però, possiamo sfruttare al meglio la possibilità di concatenare più istruzioni sulla stessa riga, fino a riprodurre una situazione del tipo IF...THEN...ELSE.

La cosa funziona in questo modo: se la condizione contenuta nella IF è verificata viene eseguita ovviamente l'istruzione che segue il THEN, e quindi le altre istruzioni eventualmente presenti sulla stessa riga, per proseguire poi con le linee successive; nel caso la condizione non si verifichi, invece, l'esecuzione passa alla linea successiva, anche se più istruzioni sono presenti sulla riga.

Non essendoci praticamente alcun limite al numero di istruzioni concatenabili, possiamo creare strutture piuttosto complesse. Un blocco di istruzioni del tipo:

```
100 IF a>6 THEN GO TO 160
110 FOR i=1 TO 5
120 LET b(i)=a(i)/2
130 PRINT b(i)
140 NEXT i
150 GO TO 200
160 FOR i=6 TO 10
170 LET b(i-5)=a(i)/4
180 NEXT i
200 ecc.
```

può essere trasformato in questo modo:

```
100 IF a>6 THEN FOR i=6 TO
    10: LET b(i-5)=a(i)/4:
    NEXT i: GO TO 200
110 REM else
120 FOR i=1 TO 5
130 LET b(i)=a(i)/2
140 PRINT b(i)
150 NEXT i
160 REM end if
200 ecc.
```

con un guadagno in compattezza, leggibilità a prima vista e velocità per l'eliminazione di un salto, nonché una certa strutturazione, che rende il programma più maneggevole e modificabile. (Naturalmente le linee REM sono eliminabili: sono state aggiunte solo per sottolineare l'analogia con procedure più evolute.)

INPUT

Ferma restando l'assoluta compatibilità con l'istruzione INPUT dello ZX81, volendo sfruttare le nuove possibilità offerte è possibile convertire una sequenza del tipo:

```
200 PRINT AT 10,0; "nome"
210 INPUT n$
220 PRINT AT 10,15;n$
230 PRINT AT 12,0;"cognome"
240 INPUT c$
250 PRINT AT 12,15; c$
```

in:

```
200 INPUT "nome"; TAB 15; n$
    "cognome"; TAB 15; c$
```

Oltre al guadagno di spazio c'è un altro motivo per preferire un'istruzione di questo tipo: nel caso della sequenza di PRINT e INPUT sarà necessario prima o poi cancellare tutto quello che abbiamo visualizzato, rispettando eventualmente altre parti dello schermo di cui vogliamo mantenere il contenuto; tutto questo avviene automaticamente nel caso dell'istruzione INPUT di nuovo tipo, dove la pressione del tasto "enter" per l'immissione dell'ultimo dato (nel nostro caso il cognome) causa la cancellazione di tutto e solo quello che l'INPUT aveva visualizzato, senza bisogno di altre istruzioni.

INVERSE

Sullo Spectrum è usata per la visualizzazione dei caratteri inversi,

senza dover intervenire sui colori. Nel nostro caso il suo uso è utile solo se ci troviamo di fronte a grandi quantità di caratteri inversi non inframmezzati da caratteri normali: in questa situazione vale la pena di anteporre INVERSE 1, preceduto dal punto e virgola se necessario, alle virgolette di apertura della stringa di caratteri inversi, che scriveremo poi normalmente. L'effetto di una inversione ottenuta in questo modo cessa alla fine della PRINT entro cui è stata fatta; avendo più istruzioni PRINT contenenti solo caratteri inversi può essere utile un'inversione più generale realizzata con una istruzione INVERSE a sé stante, cioè fuori da una PRINT; questa volta l'effetto è permanente, e per farlo cessare occorre una INVERSE 0.

Il metodo più rapido nel caso di caratteri inversi mescolati a caratteri normali (il più frequente) non fa comunque uso della istruzione INVERSE, ed è molto pratico nelle conversioni, rispecchiando quello dello ZX81.

Avete sicuramente notato le diciture "inverse video" e "true video" poste sotto i tasti 3 e 4; al capitolo 16 del manuale, dove sono spiegate, vengono indicate come metodo per creare caratteri inversi specialmente nei listati, ma non molto utili per i programmi: bene, nel nostro caso sono il sistema migliore; battete i caratteri normali, quindi prima di un carattere inverso battete "inverse video" (caps shift e 4): ora siete in modo inverso, e ci rimarrete finché non batterete "true video" (caps shift e 3). Unica avvertenza: passate in "inverse video" solo dopo aver aperto le virgolette, e ripassare in "true video" sempre prima di chiuderle, per non creare strani effetti e perdere l'unico che ci interessa, cioè l'inversione, durante lo svolgimento del programma.

Quando siete in modo inverso, comunque lo abbiate ottenuto, fate attenzione nell'usare il TAB: infatti, a differenza di AT x,y che effettua un salto alla posizione richiesta, TAB y per spostare la posizione di stampa si serve di una serie di spazi, che in modo inverso saranno logicamente inversi e produrranno una "striscia" nera, o comunque del colore dei caratteri; se non si desidera questo è opportuno tornare al modo normale prima di ogni TAB o, meglio, usare AT.

LOAD, SAVE

Molte sono qui le novità; per quanto riguarda LOAD, comunque, la compatibilità resta completa: caricamento da comando, con o senza nome, o da programma, con sovrapposizione e conseguente cancellazione del vecchio programma e dei vecchi dati (esempio: il programma "Tizio" al termine della sua esecuzione chiama il programma "Caio" che gli si sovrappone, e che a sua volta potrà chiamare "Pippo" e così via).

Una modifica è invece necessaria per quei programmi che utilizzavano la possibilità di partire appena caricati; questa si basava infatti su di una sequenza di tipo:

```
999 STOP
1000 SAVE "nome"
1100 RUN (o GO TO n se sono
stati caricati anche dati)
```

La minima modifica possibile è:

```
999 STOP
1000 SAVE "nome" LINE 1100
1100 RUN (o GO TO come prima)
```

Una modifica che sfrutta maggiormente le nuove possibilità può essere:

```
999 STOP
1000 SAVE "nome" LINE n
```

dove *n* è la linea da cui ci interessa che inizi l'esecuzione; l'istruzione LINE effettua l'equivalente di un GO TO, senza quindi cancellare i dati eventualmente caricati col programma.

Una terza possibilità è quella di non includere l'istruzione SAVE nel programma, ma utilizzare SAVE LINE *n* come comando; infatti l'esecuzione automatica avviene ugualmente; unico svantaggio può essere il non avere il nome del programma scritto da nessuna parte nel listato, cosa abbastanza trascurabile visto che questo viene scritto automaticamente nella forma "program: nome" dal sistema all'atto del caricamento.

PAUSE

Per ottenere una pausa illimitata ora la forma è PAUSE 0: i vecchi PAUSE con argomento superiore a 32767 vanno perciò corretti in questo modo.

Visto che ora le istruzioni PAUSE non provocano più i famosi salti di schermo, potrebbe essere utile sostituirli ad eventuali cicli FOR ...NEXT usati solo per introdurre ritardi, nonché trasformare blocchi tipo:

```
50 IF INKEY$ = "" THEN GO
   TO 50
60 LET a = INKEY$
```

in:

```
50 PAUSE 0
60 LET a = INKEY$
```

PLOT

Deve in ogni caso essere modificato; possiamo intervenire con due scopi diversi: sfruttare le possibilità offerte dalla grafica ad alta definizione, o riprodurre con un artificio la grafica a bassa risoluzione dello ZX81. Il primo tipo di conversione, raccomandabile in tutti i casi in cui è possibile, e sempre nel caso di grafici, migliora nettamente la presentazione del programma ma comporta qualche calcolo per il passaggio da un reticolo 64×44 ad uno 256×176; questo può avvenire con un ragionamento logico qualora si conosca bene il tipo di output fornito dal programma, o con l'applicazione di formule matematiche del tipo:

$$nx = \frac{(vx+1)\cdot 256}{64} - 1$$

$$ny = \frac{(vy+1)\cdot 176}{44} - 1$$

dove *nx* e *ny* sono le nuove coordinate che verranno inserite nel programma, mentre *vx* e *vy* quelle vecchie, cioè usate dallo ZX81.

Questo sistema funziona molto bene in alcuni casi, tipicamente nel caso di grafici, dove la forma:

```
100 FOR i = 0 TO 63
110 PLOT 21+21*SIN(i/32*PI)
120 NEXT i
```

diventa:

```
100 FOR i=0 TO 255
110 PLOT 88+88*SIN(i/132)
120 NEXT i
```

Una certa approssimazione viene comunque introdotta, e solo la verifica può suggerire le ulteriori minime modifiche necessarie ad ottimizzare la trasformazione (in questo caso i/132 potrebbe essere sostituito da i/128, per una migliore centratura della curva).

Le situazioni a cui questo metodo non può essere applicato, per esempio perché la forma del disegno creato dal plottaggio si basa sulla forma e dimensioni dei "quadratini" a bassa risoluzione dello ZX81, possono essere risolte ricreando con una apposita subroutine la grafica 64×44. Ecco una forma possibile:

```
1000 LET xx = INT x:
     LET rx=
     xx+2-INT((xx+2)/2)*2:
     LET px=INT(xx/2)
1020 LET yy = INT y:
     LET ry=
     yy+2-INT((yy+2)/2)*2:
     LET py = INT(yy/2)
1030 OVER 1:
     PRINT AT 21-py, px;
1040 IF rx = 0 AND ry = 0
     THEN PRINT CHR$ 136
1050 IF rx = 1 AND ry = 0
     THEN PRINT CHR$ 132
1060 IF rx = 0 AND ry = 1
     THEN PRINT CHR$ 130
1070 IF rx = 1 AND ry = 1
     THEN PRINT CHR$ 120
1080 OVER 0: RETURN
```

dove x ed y devono essere le coordinate così come sono espresse per lo ZX81; alle istruzioni

PLOT ascissa, ordinata

vanno sostituite altrettante

LET x=ascissa: LET y=ordinata: GO SUB 1000.

Infine le istruzioni UNPLOT possono essere convertite nel primo caso in PLOT INVERSE 1;x,y. Nel secondo caso va invece utilizzata la medesima subroutine usata per il PLOT: infatti il modo OVER 1 garantisce la cancellazione del quadratino corrispondente.

Funzioni

CHR$, CODE

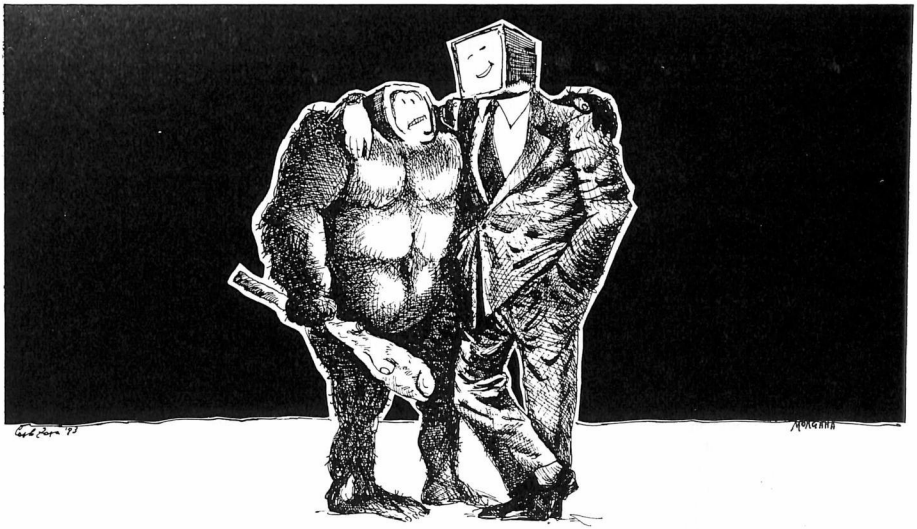
A parte le funzioni di verifica sui caratteri video, le uniche che ci possono interessare sono CHR$ e CODE; queste possono rappresentare un problema nella conversione: infatti non solo sono cambiati i codici relativi ai caratteri e segni speciali, ma i caratteri inversi non esistono più come codici a sé stanti, essendo ottenuti dai relativi caratteri normali.

Tutto questo comporta nel migliore dei casi un lavoro di traduzione dei codici, mentre nel caso disgraziato in cui il programma utilizzi una distinzione fra un carattere ed il suo inverso per mezzo del rispettivo codice, caso per fortuna piuttosto raro, saranno necessarie modifiche piuttosto profonde.

Nel primo caso, e solo se i codici non sono implicati in calcoli di alcun tipo né in confronti, può essere utile la tabella 1, che per ogni codice dello ZX81 riporta il corrispondente per lo Spectrum. Ovviamente sono stati omessi i codici che non compaiono mai come tali in un programma, cioè quelli relativi alle parole chiave ed alle funzioni, a parte CHR$ che può essere trovato come codice.

ZX81 codice n.	ZX Spectrum codice n.	ZX81 codice n.	ZX Spectrum codice n.
0	32	41	68
1	130	42	69
2	129	43	70
3	131	44	71
4	136	45	72
5	138	46	73
6	137	47	74
7	132	48	75
8	non esiste (a)	49	76
9	non esiste (a)	50	77
10	non esiste (a)	51	78
11	34	52	79
12	96	53	80
13	36	54	81
14	58	55	82
15	63	56	83
16	40	57	84
17	41	58	85
18	62	59	86
19	60	60	87
20	61	61	88
21	43	62	89
22	45	63	90
23	42	118	13
24	47	120	14
25	59	128	143
26	44	129	141
27	46	130	142
28	48	131	140
29	49	132	135
30	50	133	133
31	51	134	134
32	52	135	139
33	53	136	non esiste (a)
34	54	137	non esiste (a)
35	55	138	non esiste (a)
36	56	139-191	non esiste (b)
37	57	214	194
38	65	219	199
39	66	220	200
40	67	221	201

Tabella 1.



Per quanto riguarda i codici di cui non esiste l'esatto corrispondente per lo Spectrum, l'ostacolo può essere aggirato senza troppe complicazioni solo se la funzione CHR$ *n* da convertire si trova in una istruzione di output, è cioè destinata ad essere scritta sotto forma del carattere corrispondente sul video o sulla stampante (istruzioni PRINT, LPRINT, INPUT). Limitatamente a questa situazione possiamo distinguere due casi, che vengono richiamati dalla tabella:

(a) *caratteri grafici* comprendenti zone "a scacchiera": il carattere corrispondente viene creato in uno dei caratteri definibili dall'utente, il cui codice (od anche il carattere medesimo) verrà poi inserito come argomento della relativa funzione CHR$ *n*. Nel caso dei caratteri "tutta scacchiera" (8 e 136), ad esempio, si può procedere in questo modo:

```
1000 FOR i=0 TO 6 STEP 2
1010 RESTORE
1020 READ a:
     POKE USR "a"+i,a
1030 READ a:
     POKE USR "a"+i+1,a
1040 NEXT i
1050 DATA BIN 01010101,
     BIN 10101010
```

il medesimo carattere grafico (in questo caso la a, cioè il 144) può dare origine ad ambedue i caratteri (scacchiera con bianco in alto a sinistra o con nero in alto a sinistra) a seconda che sia o meno preceduto da INVERSE 1.

(b) *caratteri inversi*: al vecchio codice va sottratto 128, quindi va cercato sulla tabella il nuovo codice corrispondente al risultato di questa operazione, cui verrà anteposto INVERSE 1; ad esempio, supponiamo di avere a che fare con il codice 165 (9 inverso):

$$165-128 = 137$$

Al codice 137 corrisponde per lo Spectrum il codice 57 quindi INVERSE 1; CHR$ 57 è l'equivalente che cerchiamo. Nel caso seguano caratteri da non invertire, nell'ambito della stessa istruzione occorre naturalmente far cessare l'effetto con INVERSE 0.

Nei casi in cui vi siano espressioni algebriche o confronti contenenti codici, invece, bisognerà studiare nuove relazioni che tengano conto del loro mutato rapporto.

PEEK, POKE

La loro presenza indica una manipolazione della memoria (tipicamente del *display file*, cioè il video) o delle variabili di sistema, cosa questa che, come già detto, rende impossibile una conversione diretta. Cercherò comunque di dare qualche indicazione per coloro che si volessero cimentare, per quanto sia sempre più semplice rifare un programma che modificarlo oltre un certo limite. I due casi più comuni sono appunto l'accesso al display file e alle variabili di sistema, e vanno analizzati separatamente.

(a) *Display file*

L'accesso al display file dello Spectrum per mezzo di PEEK e POKE è, data la sua struttura, talmente complesso da non poter essere preso in considerazione in un caso come questo; dovremo quindi usare forme sostitutive.

Per POKE non c'è scelta: PRINT AT è l'unica soluzione pos-

sibile: d'altronde la maggior velocità dello Spectrum compensa la relativa lentezza di questo tipo di accesso. Il problema sta invece nel riuscire a convertire l'indirizzo di memoria cui punta una POKE in una riga e una colonna del video da dare come argomento ad un PRINT AT; l'unico modo è contare quanto dista l'indirizzo in questione da quello di inizio del display file (indicato dalla variabile di sistema D-File), e quindi dividere per 33 (32 caratteri per riga più "newline" alla fine): il risultato sarà il numero della riga aumentato di uno, e il resto quello della colonna, sempre aumentato di uno.

PEEK può invece essere sostituito dalla funzione SCREEN$, prevista proprio per questo scopo; anche qui lo stesso problema di conversione dell'indirizzo di memoria in un valore di riga e colonna da usare come argomento, e uguale il metodo che può essere usato per far questo. La funzione SCREEN$ riporta, per una data riga e colonna, il carattere trovato e non il suo codice, come avverrebbe con una PEEK: quindi eventualmente una riga del tipo CODE SCREEN$ (x,y) può restituire la compatibilità.

Un problema è dato dai caratteri inversi, che non sono distinti dai caratteri normali; occorre abbinare alla funzione SCREEN$ la funzione ATTR, che riconosce il colore dello sfondo e dei caratteri, secondo lo schema riportato dal manuale, in una linea tipo IF SCREEN$(x,y)="a" AND ATTR(x,y)=7 THEN... (in questo caso viene individuato il carattere "a" inverso).

(b) *Variabili di sistema*

La conversione è sempre piuttosto ardua; comunque si possono distinguere tre diverse situazioni: completa compatibilità, compatibilità del significato della variabile, ma necessità di adattare il programma al suo diverso funzionamento, ed infine alcuni casi in cui la variabile non esiste più, essendo diventato fisso l'indirizzo corrispondente all'interno della RAM.

Per il primo caso è utilizzabile questa tabella, dove per le variabili di sistema più usate vengono forniti gli equivalenti dello Spectrum, che possono essere sostituiti ai vecchi valori senza alcuna modifica al programma

Indirizzo ZX81	Nome	Indirizzo Spectrum
16391	PPC	23621
16394	E-PPC	23625
16398	DF-CC	23684
16400	VARS	23627
16402	DEST	23629
16404	E-LINE	23641
16406	CH-ADD	23645
16408	X-PTR	23647
16419	S-TOP	23660
16421	LAST-K	23560
16425	NXTLIN	23637
16427	OLDPPC	23662
16434	SEED	23670
16438	COORDS	23677
16441	S-POSN	23688

Naturalmente i risultati pratici sono tutti da sperimentare, nel senso che non ci può essere la certezza a priori di un perfetto funzionamento del programma dopo modifiche di questo tipo.

Il secondo caso ci porta decisamente verso una profonda rielaborazione del programma; queste variabili infatti, pur avendo lo stesso scopo delle loro omologhe per lo ZX81, se ne discostano anche di molte nel funzionamento.

16384 ERR-NR 23610
in condizioni normali contiene 255 come nello ZX81, ma i codici di errore sono cambiati.

16388 RAMTOP 23730
nel caso sia stata modificata occorre innanzitutto calcolare di quanto è stata abbassata, sottraendo il valore che le viene assegnato al valore originale per lo ZX81; quindi si sottrae il valore trovato a quello della RAMTOP dello Spectrum (32599 per il 16 K, 65367 per il 48 K), ottenendo il nuovo valore da utilizzare. (Vedi anche la parte relativa a CLEAR.)

16390 MODE 23617
i codici sono cambiati.

16418 DF-SZ 23659
non è possibile ottenere uno schermo di 24 righe a tutti gli effetti: infatti, ponendo questa variabile uguale a zero, possiamo sconfinare con una riga 21 molto lunga nelle due righe sottostanti, ma se tentiamo di utilizzare una PRINT AT 22,0;... otterremo un messaggio di errore, e altrettanto per PLOT.

16436 FRAMES 23672
è ora formata da tre byte anziché i due dello ZX81, e viene incrementata partendo da zero anziché decrementata: tutto da rivedere, quindi.

L'ultimo gruppo comprende le variabili che non esistono più, essendo state sostituite da un indirizzo fisso in RAM. In questi casi bisogna sostituire alla lettura della variabile mediante una PEEK il nuovo indirizzo fisso; ovviamente non esiste la possibilità di cambiare tale indirizzo mediante POKE.

16396 D-FILE ora il display file inizia all'indirizzo fisso 16384

16444 PRBUFF il buffer di stampa inizia all'indirizzo fisso 23296

inoltre le variabili di sistema, che nello ZX81 iniziavano all'indirizzo 16384 partono ora dall'indirizzo 23552 e il programma, che iniziava al 16509 parte ora dall'indirizzo contenuto nella variabile il sistema PROG posta all'indirizzo 23635.

Per concludere, due parole sulla velocità di elaborazione delle due macchine; lo Spectrum è più veloce del 200-250% rispetto allo ZX81 in SLOW, ma è più lento del 5-20% rispetto allo stesso in FAST; questo vuol dire che la maggior parte dei programmi, funzionando interamente in SLOW sullo ZX81, saranno circa tre volte più veloci sullo Spectrum, con eventuale necessità di interporre alcune pause nel caso di programmi di movimento che risultassero troppo rapidi. D'altra parte programmi di calcolo, o che comunque per una parte rilevante della loro durata funzionino in FAST sullo ZX81, avranno sullo Spectrum circa la stessa velocità, arrivando al limite ad essere più lenti, anche se non ho avuto modo di osservare un caso del genere. ■

5

Giorno della settimana e fase della luna

Mauro Boscarol

L'argomento di questo mese è il computo del tempo. Presentiamo un algoritmo di uso generale (il calcolo del giorno giuliano) e due sue applicazioni: la determinazione del giorno della settimana (lunedì, martedì,...) e della fase della luna corrispondenti ad una data fissata.

Normalmente il computo del tempo si fonda su certe unità con l'aiuto delle quali si suddivide lo scorrere del tempo in determinate parti: tali unità sono il giorno, il mese e l'anno.

Il calendario usuale, tuttavia, presenta molte complicazioni per il calcolo automatico, alcune delle quali sono: le diverse lunghezze dei mesi, la presenza degli anni bisestili, i cambiamenti della riforma gregoriana (vedi oltre).

Gli astronomi, che hanno frequentemente necessità di calcoli riguardanti lo scorrere del tempo, superano questi problemi usando un metodo che si basa sulla numerazione progressiva dei giorni.

Il metodo risale al 1581, quando Giuseppe Giusto Scaligero propose di numerare i giorni progressivamente a cominciare dall'1 gennaio 4713 a.C., che viene ad avere il numero d'ordine zero. Questo tipo di computo fu da egli chiamato "giorno giuliano" (J.D., *Julianus dies*) in onore di suo padre Giulio Scaligero.

Per esempio, il 10 giugno 1983 è il giorno giuliano numero 2445496, il giorno dopo ha un numero in più e così via.

Il primo algoritmo che presentiamo, riguarda appunto la conversione di una data espressa con giorno, mese, anno nel relativo giorno giuliano. L'algoritmo è valido per ogni data compresa tra l'1 gennaio 4713 a.C. e il 31 dicembre 9999 d.C.

1 Sia g il giorno, m il mese e a l'anno

2 Se $a < 0$ allora $a=a+1$

3 Calcolare x=INT $((m-3)/12)$

4 Calcolare y=INT$(((m-2) - (x * 12)) * 30.59)$

5 Calcolare JD=INT$((a + x + 4712) * 365.25) + y + g + 29$

6 Se JD $\geq$ 2299170, allora JD = JD+2 − INT(.75 * INT$((a+100+x)/100)$)

Alcune parole di spiegazione dell'algoritmo.

Al passo **1** viene introdotta la data. Se si tratta di una data a.C. bisogna introdurla con il segno negativo. Per non appesantire l'algoritmo, non sono stati scritti i controlli sull'esattezza della data, che comunque sono i seguenti:

- $-4713 \leq a \leq 9999$;
- $1 \leq m \leq 12$;
- $1 \leq g \leq 28$, 29, 30 o 31 secondo m e a;
- se a=1582 e m=10, g non può essere compreso tra 5 e 14.

```
100 INPUT G,M,A
110 GOSUB 240
120 GOSUB 400
130 GOSUB 600
150 PRINT JD,N,F
160 GOTO 100
170 REM ----------------------------
180 REM CALCOLO DEL GIORNO GIULIANO
190 REM ----------------------------
200 REM VARIABILI IN INPUT:
210 REM G - GIORNO
220 REM M - MESE
230 REM A - ANNO
235 REM VARIABILE IN OUTPUT:
236 REM JD - GIORNO GIULIANO
237 REM
240 IF A<0 THEN A=A+1
250 X=INT((M-3)/12)
260 Y=INT(((M-2)-(X*12))*30.59)
270 JD=INT((A+X+4712)*365.25)+Y+G+29
280 IF JD<2299170 THEN RETURN
290 JD=JD-INT(.75*INT((A+100+X)/100))+2
300 RETURN
310 REM ---------------------------------
320 REM CALCOLO DEL GIORNO DELLA SETTIMANA
330 REM ---------------------------------
340 REM VARIABILE IN INPUT:
350 REM JD - GIORNO GIULIANO
360 REM VARIABILE IN OUTPUT:
370 REM N - GIORNO DELLA SETTIMANA
380 REM     (0=LUN, 1=MAR, ..., 6=DOM)
390 REM
400 N=JD-7*INT(JD/7)
410 RETURN
420 REM -----------------------------
430 REM CALCOLO DELLA FASE DELLA LUNA
440 REM -----------------------------
450 REM VARIABILE IN INPUT:
460 REM JD - GIORNO GIULIANO
470 REM VARIABILE IN OUTPUT:
480 REM Z - FASE DELLA LUNA
490 REM     (0=PRIMO QUARTO,   .25=LUNA PIENA
500 REM     .50=ULTIMO QUARTO  .75=LUNA NUOVA)
550 REM
600 Z=JD/29.53059-.55904
610 F=Z-INT(Z)
620 RETURN
```

Quest'ultima limitazione è dovuta alla riforma gregoriana del calendario che, tra le altre cose, stabilì che il giorno dopo il 4 ottobre 1582 si chiamasse 5 ottobre 1582, eliminando quindi dal calendario i 15 giorni compresi tra queste due date.

Al passo **2** viene effettuata una correzione "matematica". Infatti, nella normale numerazione degli anni, manca l'anno zero. All'anno 1 a.C. segue immediatamente l'anno 1 d.C. Questo uso porta facilmente ad errori nei calcoli, e gli astronomi preferiscono chiamare anno 0 l'anno 1 a.C., anno −1 il 2 a.C. e così via. Ciò permette di calcolare immediatamente con una somma algebrica il numero di anni trascorsi tra due date. Per esempio tra la morte di Giulio Cesare, avvenuta il 44 a.C. (anno −43 secondo gli astronomi), e il 1957 sono passati 1957−(−43)=2000 anni.

Ai passi da **3** a **5** viene applicata la formula di calcolo del giorno giuliano (la cui spiegazione è un po' complessa: se a qualcuno interesserà, la pubblicheremo nei prossimi numeri di questa rubrica) e nel passo **6** vengono apportate le opportune correzioni se si tratta di una data posteriore alla riforma gregoriana (15 ottobre 1582).

Dal giorno giuliano è possibile ricavare molte informazioni. Per esempio, per calcolare il numero dei giorni che intercorrono tra due date basta calcolare i rispettivi giorni giuliani e fare la differenza. Altrettanto facile è trovare il giorno della settimana e la fase della luna: riportiamo in dettaglio questi due ultimi calcoli.

Per quanto riguarda il giorno della settimana, è sufficiente dividere il giorno giuliano per 7: se il resto è 0 si tratta di lunedì, se è 1, martedì, eccetera. Per esempio, il 10 giugno 1983 ha giorno giuliano numero 2445496, che diviso per 7 dà 349356 con resto 4. Si tratta dunque di un venerdì.

Per la fase della luna, invece, bisogna dividere il giorno giuliano per 29.53059 e sottrarre 0.55904. La parte frazionaria (cioè quella dopo il punto decimale) del risultato va interpretata così: se è 0 si tratta del primo quarto; 0.25 luna piena; 0.5 ultimo quarto; 0.75 luna nuova; naturalmente sono possibili tutte le situazioni intermedie.

Per esempio, per il 10 giugno 1982, dividendo 2445496 per 29.53059 e sottraendo 0.55904 si ottiene 82811.73831 la cui parte frazionaria è 0.73831: poco prima della luna nuova (cioè in fase di luna calante).

Tenendo conto che, in questa rappresentazione, un giorno corrisponde a 0.03386319, si può calcolare con molta esattezza quanti giorni fa vi era la luna piena o tra quanti giorni ci sarà la luna nuova.

◑	primo quarto	0.00
○	luna piena	0.25
◐	ultimo quarto	0.50
●	luna nuova	0.75

Fasi della luna e corrispondenti valori numerici forniti dall'algoritmo. Il valore 0.034 corrisponde al trascorrere di un giorno. Se quindi per il 10 giugno 1983 il valore numerico è 0.73831, significa che a quella data era passato poco meno di un giorno dalla luna nuova.

Il programma

Il programma segue esattamente gli algoritmi presentati. Dalla linea 100 alla linea 160 vi è un esempio di esecuzione. Le linee da 240 a 290 corrispondono ai passi da **2** a **6** dell'algoritmo principale.

Nella riga 400 viene calcolato il resto della divisione di JD per 7. Nelle righe 600-610 viene calcolata la parte frazionaria della divisione di JD per 29.53059.

-- Data --			Giorno Giuliano	Giorno della settimana	Fase della luna
1	6	1983	2445487	2	.433532715
2	6	1983	2445488	3	.467407227
3	6	1983	2445489	4	.501251221
4	6	1983	2445490	5	.535125732
5	6	1983	2445491	6	.569000244
6	6	1983	2445492	0	.602844238
7	6	1983	2445493	1	.63671875
8	6	1983	2445494	2	.670593262
9	6	1983	2445495	3	.704437256
10	6	1983	2445496	4	.738311768
11	6	1983	2445497	5	.772155762
12	6	1983	2445498	6	.806030273
13	6	1983	2445499	0	.839904785
14	6	1983	2445500	1	.873748779
15	6	1983	2445501	2	.907623291
16	6	1983	2445502	3	.941497803
17	6	1983	2445503	4	.975341797
18	6	1983	2445504	5	9.21630859E-03
19	6	1983	2445505	6	.0430603027
20	6	1983	2445506	0	.0769348145
21	6	1983	2445507	1	.110809326
22	6	1983	2445508	2	.14465332
23	6	1983	2445509	3	.178527832
24	6	1983	2445510	4	.212402344
25	6	1983	2445511	5	.246246338
26	6	1983	2445512	6	.28012085
27	6	1983	2445513	0	.313964844
28	6	1983	2445514	1	.347839355
29	6	1983	2445515	2	.381713867
30	6	1983	2445516	3	.415557861 ■

La trasformazione automatica di coordinate topografiche

Come orientarsi tra coordinate cartesiane e polari, gradi sessagesimali, sessadecimali, centesimali e radianti

di *Bartolo Saccà*

Durante le elaborazioni analitiche, che logicamente seguono qualunque tipo di rilevamento topografico, è frequentissimo, se non inevitabile, imbattersi in un problema di trasformazione di coordinate, sia da polari a cartesiane (caso più frequente), che da cartesiane a polari.

Con l'elaborazione del presente programma (scritto per un TRS-80 mod. III), prendiamo in esame una vasta fascia della casistica che può presentarsi per il caso in discussione, componendone in maniera razionale le soluzioni.

Occorre prima puntualizzare che nella trattazione di problemi topografici è opportuno operare in doppia precisione, avendo l'esperienza diretta dimostrato che solo così la trattazione analitica è adeguata alla precisione della gran parte dei rilevamenti, che spesso si spinge sino al limite del secondo centesimale. Le perdite in termini di tempo e di

Tre sistemi pratici ed uno scientifico per misurare gli angoli

Il sistema tradizionale di misura degli angoli (sistema *sessagesimale*) consiste nel suddividere l'angolo giro in 360 parti uguali (gradi), ogni grado in ulteriore 60 parti (primi) ed ogni primo in 60 secondi. Per esempio con la notazione 68°10′26″ si intendono 68 gradi, 10 primi e 26 secondi.

Tale sistema è tuttavia poco adatto per l'elaborazione automatica, perché i primi e i secondi sono sessantesimi e non sottomultipli decimali dell'unità.

Si è cercato di ovviare al problema introducendo il sistema *sessadecimale*, in cui la frazione di grado non è espressa in primi e secondi ma in frazioni decimali, centesimali, ecc. Per esempio 68°10′26″ sessagesimali sono 68, 173889 gradi sessadecimali.

Un ulteriore miglioramento, particolarmente adatto per il calcolo automatico, si è avuto con l'introduzione del sistema *centesimale*. In questo sistema un angolo giro è suddiviso in 400 gradi, ogni grado in 100 primi e ogni primo in 100 secondi. Per esempio 68°10′26″ sono 75, 748457 gradi centesimali.

Infine, per i calcoli di tipo scientifico, gli angoli vengono misurati in *radianti*. Un radiante è l'angolo al centro corrispondente a un arco di cerchio di lunghezza pari al raggio. Un angolo giro dunque misura 2π radianti e un angolo piatto π radianti.

Nel Basic, l'unico tipo di misura degli angoli implementato è quello in radianti.

capacità di memoria che ne derivano sono così limitate da non costituire un problema per l'utente.

Tornando al problema in esame, e volendo dapprima studiare la trasformazione che chiameremo "diretta" (da coordinate polari a cartesiane), è da considerare il sistema di misura degli angoli con cui si opera: è, infatti, sempre più diffuso, e certamente in futuro diventerà predominante se non unico, il sistema centesimale rispetto al tradizionale sistema sessagesimale (vedi riquadro); oggi, comunque, i due sistemi sembrano equivalersi, ed è quindi sembrato opportuno porre, all'inizio del programma, una opzione, sotto forma di domanda diretta dal monitor, circa il sistema che si è adoperato nel corso del rilevamento.

In entrambi i casi, tuttavia, è indispensabile effettuare la trasformazione degli angoli dal sistema originario al sistema analitico (o radiometrico) essendo soltanto questo il sistema con cui operano tutti i microcomputer che dialogano in Basic. A tal fine, già nella prima riga d'istruzioni (numero di riga 120), viene introdotto il valore di pi greco.

La trasformazione degli angoli viene eseguita mediante le istruzioni della riga 480 se si è operato, nel rilevamento, con il sistema centesimale, mentre il passaggio dal sistema sessagesimale viene eseguito alla riga 470; quest'ultima ipotesi, tuttavia, comporta un iter più elaborato, dovendosi prima trasformare l'angolo stesso da sessagesimale in sessadecimale, non essendo i microcomputer dotati di un automatismo che risolva tale trasformazione (mentre, al contrario, il problema è risolto nella maggior parte delle calcolatrici tascabili): per questo motivo è necessario servirsi della subroutine di cui alla riga 550 e seguenti.

Dai valori sia dell'angolo così trasformato, sia della distanza, con la semplice applicazione di elementari formule trigonometriche, è immediata la trasformazione da coordinate polari in cartesiane, come dalle istruzioni riportate alla riga 500.

```
100 CLS
110 PRINT"                    TRASFORMAZIONE DI COORDINATE"
120 DEFDBL A-Z: P=3.14159265: PG=P/180
130 PRINT @384,"DA CARTESIANE A POLARI BATTI 1"
140 PRINT"DA POLARI A CARTESIANE BATTI 2"
150 INPUT W
160 CLS: F1$="###.#####": F2$="#####.##": F3$="###"
170 ON W GOTO 180,380
180 PRINT"DAMMI LE COORDINATE DEI PUNTI X1,Y1,X2,Y2"
190 INPUT X1,Y1,X2,Y2
200 A=X2-X1: B=Y2-Y1
210 T=ABS(ATN(A/B))
220 IF A>0 AND B>0 THEN 260
230 IF A>0 AND B<0 THEN 270
240 IF A<0 AND B<0 THEN 280
250 T=2*P-T: L=A/SIN(T): GOTO 290
260 L=A/SIN(T): GOTO 290
270 T=P-T: L=A/SIN(T): GOTO 290
280 T=P+T: L=A/SIN(T)
290 T=T/PG: K=T/.9: AA=CINT(T): AB=(T-AA)*60
300 IF AC=-1 AND AD=60 THEN AC=0: AD=0
310 PRINT"AZIMUT=";: PRINT USING F1$;T;
320 PRINT" (";: PRINT USING F1$;K;
330 PRINT")  (";: PRINT USING F3$;AA;AC;AD;
340 PRINT")"
350 PRINT"DISTANZA=";: PRINT USING F2$;L
360 PRINT"-----------------------------------------"
370 PRINT: GOTO 180
380 CLS
390 PRINT"SE L'AZIMUT E' SESSADECIMALE.....BATTI 1"
400 PRINT"SE L'AZIMUT E' CENTESIMALE.....BATTI 2"
410 PRINT"SE L'AZIMUT E' SESSAGESIMALE.....BATTI 3"
420 INPUT AA
430 INPUT"DAMMI L'AZIMUT";T$:T=VAL(T$)
440 INPUT"DAMMI LA DISTANZA";D
450 PRINT"DAMMI LE COORDINATE DEL PRIMO PUNTO": INPUT X1,Y1
460 ON AA GOTO 470,480,490
470 T=T*PG: GOTO 500
480 T=T*.9: GOTO 470
490 AG$=T$: GOSUB 550: T=AR: GOTO 470
500 X2=D*SIN(T)+X1: Y2=D*COS(T)+Y1
510 PRINT"LE COORDINATE DEL SECONDO PUNTO SONO:";
520 PRINT USING F2$;X2;Y2
530 PRINT"-----------------------------------------"
540 PRINT: GOTO 430
550 L=LEN(AG$): FOR L%=1 TO L
560 IF MID$(AG$,L%,1)="." GOTO 570
561 NEXT
562 GR=VAL(AG$): MM=0: SS=0: GOTO 580
570 GR=VAL(LEFT$(AG$,L%-1))
580 MM=VAL(MID$(AG$,L%+1,2))
590 SS=VAL(MID$(AG$,L%+3,2))
600 AR=GR+MM/60+SS/3600
610 RETURN
```

Poiché non sempre il primo vertice coincide con l'origine degli assi, è sembrato opportuno prevedere già note le cartesiane di tale punto: da ciò deriva che le assolute del secondo punto saranno ricavate tramite una somma algebrica dei valori delle coordinate stesse (assolute + parziali).

Tramite le istruzioni di cui alla riga 160, limiteremo la precisione dei risultati alla seconda cifra decimale: ciò appare giustificato dal fatto che generalmente è il metro l'unità di misura adoperata nell'espressione dei valori distanziometrici.

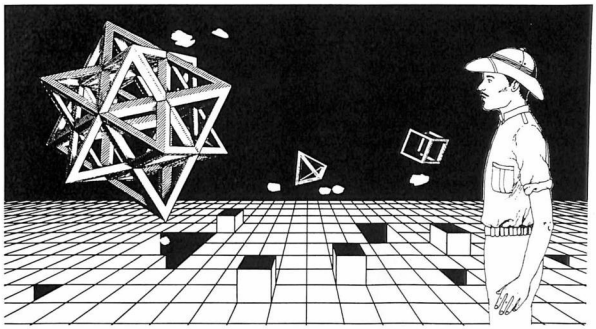
Esaurito il ciclo con le istruzioni di stampa dei risultati finali, il programma prevede la risoluzione di ulteriori problemi simili e dal monitor appare esplicita la richiesta dei rispettivi dati.

Passando ora all'esame del secondo problema (trasformazione da coordinate cartesiane a polari), appare subito evidente la necessità di prendere in considerazione i vari

sottocasi che possono presentarsi in dipendenza dalla reciproca posizione dei due punti dati: in altri termini occorre considerare il quadrante in cui si viene a trovare il secondo punto rispetto ad un sistema di assi secondari parallelo all'originario e con l'origine coincidente con il primo punto: la soluzione, notoriamente, dipende dal segno sia del numeratore, sia del denominatore della formula che serve a ricavare il valore della tangente trigonometrica dell'azimut cercato; dalla riga 200 alla riga 280 vengono esaminate e risolte queste varie possibilità, mentre la corrispondente soluzione dell'azimut è stata realizzata alla riga 290.

È importante osservare che, per la comodità dell'operatore, il programma è stato rifinito in maniera da avere, quale risultato finale dell'azimut, non soltanto il suo valore sessadecimale, utile per eventuali ulteriori elaborazioni, ma anche, inseriti tra parentesi, i valori dello stesso angolo espressi sia nel sistema centesimale che in quello sessagesimale.

Si è ritenuto opportuno non indicare l'unità di misura relativa al valore della distanza, in quanto nel risultato essa è la stessa che si era adoperata nell'input delle coordinate cartesiane; inoltre, poiché nella maggior parte dei casi essa è espressa in metri, la si è arrotondata sino alla seconda cifra decimale, essendo apparso eccessivo prevedere un'opzione anche per l'approssimazione lineare, la quale, tuttavia, può essere inclusa dall'utente con qualche semplice artificio di programmazione.



Come già nella trasformazione "diretta" descritta in precedenza, anche qui è prevista la soluzione di più problemi identici in serie e quindi, risolto un caso, il computer richiede dal monitor i dati per risolvere il prossimo problema.

Questo programma, pur nella sua semplicità, risolve in maniera completa e sufficientemente chiara il problema della trasformazione; data la limitatezza dell'estensione grafica delle risposte, non si è ritenuto opportuno far agire la stampante, potendosi leggere i risultati direttamente sul monitor. ■

SOFTWARE PER ENERGIA SOLARE

ENERGIA SOLARE 1 è il titolo di un package di programmi realizzati da Renato Lazzarin, docente universitario e ricercatore in ingegneria solare.

Il package, che comprende memorizzati i dati climatici e meteorologici di 29 città italiane, permette il calcolo dell'energia utile di un impianto solare a pannelli piani per il riscaldamento ambientale e di acqua sanitaria e l'analisi economica dell'investimento solare.

Il manuale d'uso che accompagna il disco fornisce esaurienti indicazioni per l'esecuzione del programma, consigli sulla valutazione dei parametri, suggerimenti per ulteriori letture, oltre ad un glossario di termini tecnici e a quattro cartine d'Italia utili per la determinazione della località.

I programmi sono disponibili nelle versioni PET, Apple, TRS-80 (su disco) e Basic standard (su listato). Chiedeteci la documentazione completa che abbiamo preparato per gli studi tecnici e professionali inviandoci un biglietto da visita con la dicitura "ENERGIA SOLARE 1".

COMPLETO SOFTWARE

è una divisione della COMPLETO SRL
engineering per l'informatica e l'editoria tecnica e scientifica

Via Bonporti 36 - 35100 Padova - Telefoni 049/661147-661873

Come stampare lo schermo del VIC

Con questi programmi in Basic, assembler e linguaggio macchina potrete stampare i vostri grafici in alta e bassa risoluzione

di *Alessandro Guida*

Prima o poi capita di dover fare una copia dello schermo su carta, sia in bassa risoluzione che in alta. Il problema non è certo di difficile soluzione, a patto che si tengano presenti alcune esigenze come la velocità di esecuzione.

Alta risoluzione

Come si ottiene l'alta risoluzione sul VIC è stato più volte spiegato in altri articoli anche su questa rivista. Qui ci basta ricordare che la mappa video ad alta risoluzione, creata dalla *super expander cartridge*, è organizzata per colonne (fig. 1). Ci sono 20 colonne composte da 160 byte contigui, in ordine crescente dall'alto in basso. Poiché ogni byte è formato da 8 bit, ed ogni bit corrisponde ad un punto, la risoluzione è di 160×160 (20×8) pixel. L'ideale sarebbe poter ricopiare lo schermo su carta byte dopo byte, una colonna per volta. Purtroppo, invece, la stampante del VIC in modo grafico può solo trattare righe alte 7 pixel e lunghe al massimo 480 (fig. 2). È perciò necessario sondare la mappa video nell'ordine con cui verranno inviati i dati alla stampante. Come vengono preparati questi dati è più facile da capire studiando il programma 1 che descrivendo il procedimento a parole. L'unico inconveniente è la velocità (circa 25 minuti per fare una copia).

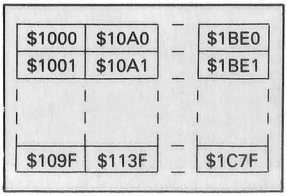


Fig. 1. *Organizzazione mappa video. Ogni locazione indicata contiene un byte, ogni bit del quale è un punto sullo schermo. Se il bit è alto il punto è visibile.*

Il programma 2 è la traduzione del programma 1 in linguaggio macchina, ed è molto più veloce (una copia in circa 2 minuti). La riga 55 serve ad essere sicuri che i puntatori al limite della memoria non vengano più spostati, e può essere eliminata da chi ha una espansione RAM da 8 K o più. Infatti se non si hanno altre espansioni oltre i 3 K necessari per la grafica, l'istruzione GRAPHIC2 sposta i puntatori a 4095($0FFF), cancellando tutto ciò che si trova oltre questa locazione. Se invece si hanno almeno altri 8 K di RAM, viene spostato l'inizio dello spazio riservato al Basic, da 4096($1000) a 8192($2000), lasciando inalterati i puntatori al limite della memoria.

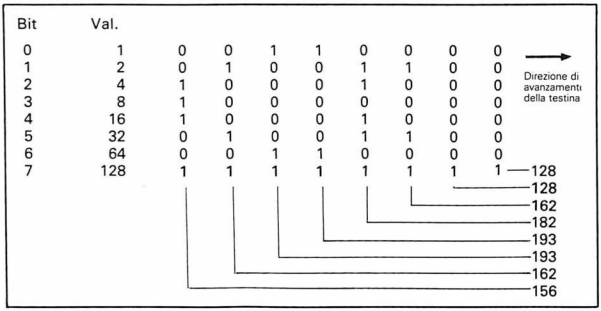


Fig. 2. *Come opera la stampante in modo grafico. Per stampare il simbolo della Commodore come in figura bisogna inviare i seguenti dati: 8,156,162,193,193,182,162,128,128. Il primo 8 seleziona il modo grafico. Gli altri dati definiscono il disegno da stampare.*


```
10 REM************************
20 REM*COPIA  DELLO SCHERMO*
30 REM*IN ALTA  RISOLUZIONE*
40 REM************************
50 REM*AGGIORNAM. PUNTATORI*
55 GRAPHIC2:GRAPHIC0
60 POKE56,PEEK(56)-1:POKE52,PEEK(52)-1
70 S=PEEK(56)*256+PEEK(55)+1:S1=S
80 REM* CARICA DATI IN HEX *
90 READA$:IFA$="*"THENPRINT"□LA ROUTINE SI CHIAMA  ■CON SYS("S1"■)":END
100 A1$=LEFT$(A$,1)
110 N1=ASC(A1$)-48:IFN1>9THENN1=N1-7
120 A1$=RIGHT$(A$,1)
130 N=ASC(A1$)-48:IFN>9THENN=N-7
140 N1=N1*16+N:POKES,N1:S=S+1:GOTO90
150 DATAA9,01,A2,04,A0,00,20,BA,FF,A9,00,20,BD,FF,20
160 DATAC0,FF,A2,01,20,C9,FF,A9,08,20,D2,FF,A9,00,85
170 DATA01,85,62,A9,10,85,02,85,63,A9,14,85,64,A9,08
180 DATA85,65,A0,06,A6,65,B1,62,4A,CA,D0,FC,26,FB,88
190 DATA10,F3,A5,FB,09,80,20,D2,FF,20,D2,FF,EA,EA,EA
200 DATAEA,C6,65,D0,DF,EA,EA,C6,64,F0,10,18,A5,62,69
210 DATAA0,85,62,A5,63,69,00,85,63,18,90,C5,A9,0D,20
220 DATAD2,FF,18,A5,01,69,07,85,01,A5,02,69,00,85,02
230 DATAC9,10,D0,08,A5,01,C9,A1,D0,02,F0,0B,A5,02,85
240 DATA63,A5,01,85,62,18,90,98,A9,0F,20,D2,FF,20,CC
250 DATAFF,A9,01,20,C3,FF,60,EA,*
```

Listato 2. *Programma in linguaggio macchina per l'alta risoluzione.*

```
60000 REM COPIA SU CARTA DELLO
60010 REM SCHERMO IN ALTA RIS.
60020 S=4096:L=20:C=20
60030 REM S=INIZIO MAPPA VIDEO
60040 REM VALORI VALIDI CON LA
60050 REM SUPER EXPANDER CART.
60060 OPEN1,4
60070 PRINT#1,CHR$(8)
60080 FORY=0TOL*8STEP7
60090 A$=""
60100 FORX=0TOC-1
60110 FORI=7TO0STEP-1
60120 B=128
60130 FORRG=0TO6
60140 A=PEEK(S+RG+X*160+Y)AND2↑I
60150 IFATHENB=BOR2↑RG
60160 NEXTRG
60170 A$=A$+CHR$(B)
60180 NEXTI,X
60190 PRINT#1,A$
60200 NEXTY
60210 PRINT#1,CHR$(15)
60220 CLOSE1
```

Listato 1. *Programma Basic per l'alta risoluzione.*

Bassa risoluzione

Fare una copia dello schermo in modo testo è più semplice: basta leggere la mappa video riga per riga e tradurre i valori trovati in codice ASCII, come mostrato in fig. 3. Il programma va bene per qualsiasi schermo, poiché provvede da solo a determinare la locazione d'inizio della mappa, il numero di righe e colonne e il set di caratteri da utilizzare. Quando il controllo torna al Basic è possibile verificare il contenuto della locazione $01 che è 0 se tutto è andato bene, al-

Cod. carattere mappa video	Conversione in ASCII
C<32	C=C+64
31<C<64	C=C
63<C<96	C=C+128
95<C<128	C=C+64
C>127	Carattere in reverse.C=C−128

Fig. 3. *Poiché nella mappa video i caratteri non sono memorizzati in codice ASCII è necessario convertirli prima di inviarli alla stampante. I caratteri con codice >127 prima di essere convertiti vengono diminuiti di 128. Contemporaneamente viene settato un flag che indica che il carattere era in reverse.*

```
B*
   PC  SR AC XR YR SP
.;603E 33 00 63 00 F6
.
.; 0400 LDA #$01
.; 0402 LDX #$04
.; 0404 LDY #$00
.; 0406 JSR $FFBA
.; 0409 LDA #$00
.; 040B JSR $FFBD
.; 040E JSR $FFC0
.; 0411 LDX #$01
.; 0413 JSR $FFC9
.; 0416 LDA #$03
.; 0418 JSR $FFD2
.; 041B LDA #$00
.; 041D STA $01
.; 041F STA $62
.; 0421 LDA #$10
.; 0423 STA $02
.; 0425 STA $63
.; 0427 LDA #$14
.; 0429 STA $64
.; 042B LDA #$08
.; 042D STA $65
.; 042F LDY #$06
.; 0431 LDX $65
.; 0433 LDA ($62),Y
.; 0435 LSR
.; 0436 DEX
.; 0437 BNE $0435
.; 0439 ROL $FB
.; 043B DEY
.; 043C BPL $0431
.; 043E LDA $FB
.; 0440 ORA #$80
.; 0442 JSR $FFD2
.; 0445 JSR $FFD2
.; 0448 NOP
.; 0449 NOP
.; 044A NOP
.; 044B NOP
.; 044C DEC $65
.; 044E BNE $042F
.; 0450 NOP
.; 0451 NOP
.; 0452 DEC $64
.; 0454 BEQ $0466
.; 0456 CLC
.; 0457 LDA $62
.; 0459 ADC #$A0
.; 045B STA $62
.; 045D LDA $63
.; 045F ADC #$00
.; 0461 STA $63
.; 0463 CLC
.; 0464 BCC $042B
.; 0466 LDA #$0D
.; 0468 JSR $FFD2
.; 046B CLC
.; 046C LDA $01
.; 046E ADC #$07
.; 0470 STA $01
.; 0472 LDA $02
.; 0474 ADC #$00
.; 0476 STA $02
.; 0478 CMP #$10
.; 047A BNE $0484
.; 047C LDA $01
.; 047E CMP #$A1
.; 0480 BNE $0484
.; 0482 BEQ $048F
.; 0484 LDA $62
.; 0486 STA $63
.; 0488 LDA $01
.; 048A STA $62
.; 048C CLC
.; 048D BCC $0427
.; 048F LDA #$0F
.; 0491 JSR $FFD2
.; 0494 JSR $FFCC
.; 0497 LDA #$01
.; 0499 JSR $FFC3
.; 049C RTS
.; 049D NOP
.:0400 A9 01 A2 04 A0
.:0405 00 20 BA FF A9
.:040A 00 20 BD FF 20
.:040F C0 FF A2 01 20
.:0414 C9 FF A9 08 20
.:0419 D2 FF A9 00 85
.:041E 01 85 62 A9 10
.:0423 85 02 85 63 A9
.:0428 14 85 64 A9 08
.:042D 85 65 A0 06 A6
.:0432 65 B1 62 4A CA
.:0437 D0 FC 26 FB 88
.:043C 10 F3 A5 FB 09
.:0441 80 20 D2 FF 20
.:0446 D2 FF EA EA EA
.:044B EA C6 65 D0 DF
.:0450 EA EA C6 64 F0
.:0455 10 18 A5 62 69
.:045A A0 85 62 A5 63
.:045F 69 00 85 63 18
.:0464 90 C5 A9 0D 20
.:0469 D2 FF 18 A5 01
.:046E 69 07 85 01 A5
.:0473 02 69 00 85 02
.:0478 C9 10 D0 08 A5
.:047D 01 C9 A1 D0 02
.:0482 F0 0B A5 02 85
.:0487 63 A5 01 85 62
.:048C 18 90 98 A9 0F
.:0491 20 D2 FF 20 CC
.:0496 FF A9 01 20 C3
.:049B FF 60 EA EA EA
```

Listato 3. *Listato in linguaggio macchina del programma per l'alta risoluzione.*

trimenti si ha il codice dell'errore (5=stampante assente; 1=troppi file aperti; 2=file già aperto). In questa maniera non si ha il blocco del programma se, per esempio, la stampante è spenta.

Dopo aver caricato il programma in memoria, basta eseguirlo una volta e poi volendo si può anche cancellarlo. Infine se volete incorporare uno dei due programmi come subroutine di uno più grande la linea 90 va cambiata in:

90 READ A$: IF A$="*" THEN RETURN

La routine in linguaggio macchina va, quindi, chiamata con SYS(S1).

```
10 REM************************
20 REM*COPIA  DELLO SCHERMO*
30 REM*IN BASSA RISOLUZIONE*
40 REM************************
50 REM*AGGIORNAM. PUNTATORI*
60 POKE56,PEEK(56)-1:POKE52,PEEK(52)-1
70 S=PEEK(56)*256+PEEK(55)+1:S1=S
80 REM* CARICA DATI IN HEX *
90 READA$:IFA$="*"THENPRINT"LA ROUTINE SI CHIAMA CON SYS("S1")":END
100 A1$=LEFT$(A$,1)
110 N1=ASC(A1$)-48:IFN1>9THENN1=N1-7
120 A1$=RIGHT$(A$,1)
130 N=ASC(A1$)-48:IFN>9THENN=N-7
140 N1=N1*16+N:POKES,N1:S=S+1:GOTO90
150 DATA78,1D,05,90,A2,04,A0,00,29,0F,F0,02,A0,07,A9
160 DATA01,20,BA,FF,A9,00,20,BD,FF,20,C0,FF,90,10,85
170 DATA01,A9,0F,20,D2,FF,20,CC,FF,A9,01,20,C3,FF,60
180 DATAA2,01,20,C9,FF,B0,E9,A9,00,AE,88,02,85,62,86
190 DATA63,20,ED,FF,86,00,84,01,A0,00,A2,00,A9,0E,20
200 DATAD2,FF,A9,00,85,61,B1,62,C9,80,30,04,29,7F,85
210 DATA61,C9,22,D0,04,A9,27,D0,18,C9,20,10,05,18,69
220 DATA40,D0,0F,C9,40,30,0B,C9,60,10,04,09,80,D0,03
230 DATA18,69,40,85,64,A5,61,F0,05,A9,12,20,D2,FF,A5
240 DATA64,20,D2,FF,A5,61,F0,05,A9,92,20,D2,FF,C8,C4
250 DATA00,30,23,A9,08,20,D2,FF,A9,0D,20,D2,FF,A9,0F
260 DATA20,D2,FF,18,A5,62,65,00,85,62,A5,63,69,00,85
270 DATA63,A0,00,E8,E4,01,F0,0B,20,B7,FF,29,80,F0,85
280 DATAA9,05,D0,02,A9,00,85,01,20,CC,FF,A9,01,20,C3
290 DATAFF,38,60,EA,*
```

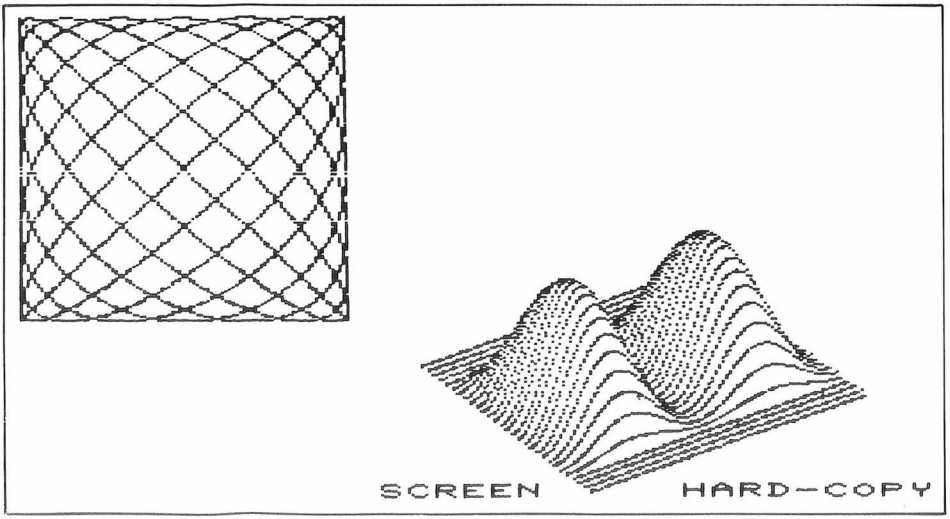
Listato 4. *Programma in linguaggio macchina per la bassa risoluzione.*

```
B*
   PC  SR AC XR YR SP
.;603E 73 00 63 00 F6
.

.; 0400 SEI
.; 0401 LDA $9005
.; 0404 LDX #$04
.; 0406 LDY #$00
.; 0408 AND #$0F
.; 040A BEQ $040E
.; 040C LDY #$07
.; 040E LDA #$01
.; 0410 JSR $FFBA
.; 0413 LDA #$00
.; 0415 JSR $FFBD
.; 0418 JSR $FFC0
.; 041B BCC $042D
.; 041D STA $01
.; 041F LDA #$0F
.; 0421 JSR $FFD2
.; 0424 JSR $FFCC
.; 0427 LDA #$01
.; 0429 JSR $FFC3
.; 042C RTS
.; 042D LDX #$01
.; 042F JSR $FFC9
.; 0432 BCS $041D
.; 0434 LDA #$00
.; 0436 LDX $0288
.; 0439 STA $62
.; 043B STX $63
.; 043D JSR $FFED
.; 0440 STX $00
.; 0442 STY $01
.; 0444 LDY #$00
.; 0446 LDX #$00
.; 0448 LDA #$0E
.; 044A JSR $FFD2
.; 044D LDA #$00
.; 044F STA $61
.; 0451 LDA ($62),Y
.; 0453 CMP #$30
.; 0455 BMI $045B
.; 0457 AND #$7F
.; 0459 STA $61
.; 045B CMP #$22
.; 045D BNE $0463
.; 045F LDA #$27
.; 0461 BNE $047B
.; 0463 CMP #$20
.; 0465 BPL $046C
.; 0467 CLC
.; 0468 ADC #$40
.; 046A BNE $047B
.; 046C CMP #$40
.; 046E BMI $047B
.; 0470 CMP #$60
.; 0472 BPL $0478
.; 0474 ORA #$80
.; 0476 BNE $047B
.; 0478 CLC
.; 0479 ADC #$40
.; 047B STA $64
.; 047D LDA $61
.; 047F BEQ $0486
.; 0481 LDA #$12
.; 0483 JSR $FFD2
.; 0486 LDA $64
.; 0488 JSR $FFD2
.; 048B LDA $61
.; 048D BEQ $0494
.; 048F LDA #$92
.; 0491 JSR $FFD2
.; 0494 INY
.; 0495 CPY $00
.; 0497 BMI $04BC
.; 0499 LDA #$08
.; 049B JSR $FFD2
.; 049E LDA #$0D
.; 04A0 JSR $FFD2
.; 04A3 LDA #$0F
.; 04A5 JSR $FFD2
.; 04A8 CLC
.; 04A9 LDA $62
.; 04AB ADC $00
.; 04AD STA $62
.; 04AF LDA $63
.; 04B1 ADC #$00
.; 04B3 STA $63
.; 04B5 LDY #$00
.; 04B7 INX
.; 04B8 CPX $01
.; 04BA BEQ $04C7
.; 04BC JSR $FFB7
.; 04BF AND #$80
.; 04C1 BEQ $0448
.; 04C3 LDA #$05
.; 04C5 BNE $04C9
.; 04C7 LDA #$00
.; 04C9 STA $01
.; 04CB JSR $FFCC
.; 04CE LDA #$01
.; 04D0 JSR $FFC3
.; 04D3 CLI
.; 04D4 RTS
.; 04D5 NOP
.

.:0400 78 AD 05 90 A2
.:0405 04 A0 00 29 0F
.:040A F0 02 A0 07 A9
.:040F 01 20 BA FF A9
.:0414 00 20 BD FF 20
.:0419 C0 FF 90 10 85
.:041E 01 A9 0F 20 D2
.:0423 FF 20 CC FF A9
.:0428 01 20 C3 FF 60
.:042D A2 01 20 C9 FF
.:0432 B0 E9 A9 00 AE
.:0437 88 02 85 62 86
.:043C 63 20 ED FF 86
.:0441 00 84 01 A0 00
.:0446 A2 00 A9 0E 20
.:044B D2 FF A9 00 85
.:0450 61 B1 62 C9 30
.:0455 30 04 29 7F 85
.:045A 61 C9 22 D0 04
.:045F A9 27 D0 18 C9
.:0464 20 10 05 18 69
.:0469 40 D0 0F C9 40
.:046E 30 0B C9 60 10
.:0473 04 69 80 D0 03
.:0478 18 69 40 85 64
.:047D A5 61 F0 05 A9
.:0482 12 20 D2 FF A5
.:0487 64 20 D2 FF A5
.:048C 61 F0 05 A9 92
.:0491 20 D2 FF C8 C4
.:0496 00 30 23 A9 06
.:049B 20 D2 FF A9 0D
.:04A0 20 D2 FF A9 0F
.:04A5 20 D2 FF 18 A5
.:04AA 62 65 00 85 62
.:04AF A5 63 69 00 85
.:04B4 63 A0 00 E8 E4
.:04B9 01 F0 0B 20 D7
.:04BE FF 29 80 F0 85
.:04C3 A9 05 D0 02 A9
.:04C8 00 85 01 20 CC
.:04CD FF A9 01 20 C3
.:04D2 FF 58 60 EA EA
```

Listato 5. *Listato in linguaggio macchina del programma 3 (bassa risoluzione).*



SINCLAIR ZX80 ZX81

Supplemento al numero 7, giugno 1983

PERSONAL SOFTWARE

UNA PUBBLICAZIONE DEL GRUPPO EDITORIALE JACKSON

ISTRUZIONI BASIC

ISTRUZIONE	DESCRIZIONE
CLEAR	Azzera tutte le variabili di programma.
CLS	Cancella lo schermo.
DIM *var*(*num*)	Definisce il vettore *var* con *num* elementi.
• **DIM *var*(*num*[,*num*...])**	Definisce la matrice *var* con *num* elementi.
FOR *var*=*num*1 TO *num*2	Ciclo fino a NEXT fino a che *var* non supera *num2*.
• **FOR *var*=*num*1 TO *num*2 STEP *incr***	Ciclo fino a NEXT, con passo *incr.*.
GOSUB *linea*	Esegue un salto alla routine nella *linea* fino a RETURN.
GOTO *linea*	Salta alla *linea*.
IF *cond* THEN *azione*	Esegui l'*azione* condizionatamente.
INPUT *var*	Assegna a *var* il valore battuto su tastiera.
LET *var*=*val*	Assegna *val* alla variabile *var*.
LPRINT *arg* [;[,(*arg*...]	Stampa dei dati.
• **LPRINT TAB *num*; *arg* [;[,[TAB...]**	Stampa con i tab.
• **LPRINT AT *linea, col*; arg [;[,[AT...]**	Stampa in posizione.
NEXT *var*	Termine del ciclo FOR; incrementa *var*.
• **PAUSE *disposizioni***	Ferma l'esecuzione per un tempo determinato.
• **PAUSE 4000**	Ferma l'esecuzione fino a che viene premuto un tasto.
• **PLOT (*orizz, vert*)**	Traccia un blocco grafico in questa posizione.
POKE *ind, num*	Memorizza *num* all'*ind*.
PRINT *arg*[;[,[*arg*...]	Visualizza i dati.
• **PRINT TAB *num*; *arg* [;[,[TAB...]**	Visualizza con i tab.
• **PRINT AT *linea, col*; *arg*[;[,[AT...]**	Visualizza in posizione.
RAND *num*	Reinizializza il generatore di numeri casuali.
REM	Commenti.
RETURN	Ritorna da GOSUB.
• **SCROLL**	Esegue lo scroll di una linea verso l'alto dello schermo.
STOP	Interrompe l'esecuzione del programma.
• **UNPLOT (*orizz, vert*)**	Cancella il blocco grafico dalla posizione *orizz, vert.*

Le parentesi non sono richieste per PLOT/UNPLOT

○ = solo per ZX80 • = solo per ZX81.

ORGANIZZAZIONE DELLO SCHERMO

LINEA	POSIZIONE	COORD Y	LINEA	POSIZIONE	COORD Y	LINEA	POSIZIONE	COORD Y
0	0	43 42	8	264	27 26	16	528	11 10
1	33	41 40	9	297	25 24	17	561	9 8
2	66	39 38	10	330	23 22	18	594	7 6
3	99	37 36	11	363	21 20	19	627	5 4
4	132	35 34	12	396	19 18	20	660	3 2
5	165	33 32	13	429	17 16	21	693	1 0
6	198	31 30	14	462	15 14	22	726	
7	231	29 28	15	495	13 12	23	759	

CODICI DI ERRORE

I codici di errore appaiono nella forma: xx/yy
dove : xx è il codice di errore,
e : yy è il numero dell'ultima istruzione eseguita.

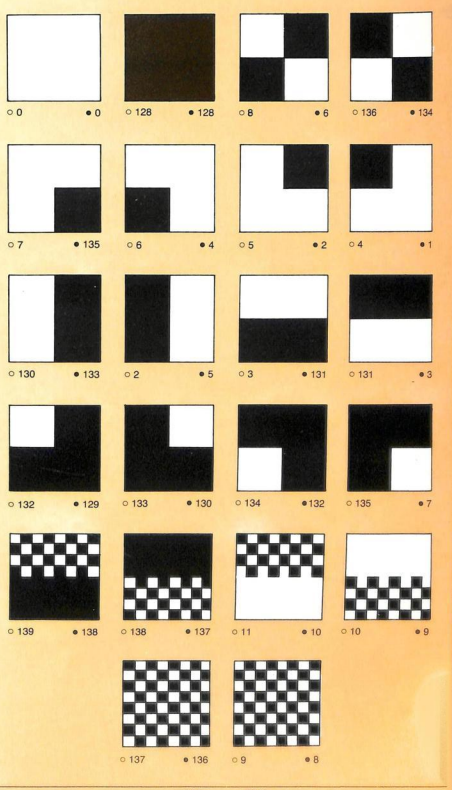
CODICE	SIGNIFICATO
0	Nessun errore.
1	NEXT ha la variabile non valida.
2	Variabile non assegnata, o DIMensionata.
3	Indice errato.
4	Memoria esaurita.
5	Schermo pieno.
6	Numero troppo grande.
7	RETURN prima di GOSUB
8	INPUT tentata in modo comando. Non lecita.
9	Eseguito STOP.
SOLO PER ZX81	
A	Parametro non valido.
B	Intero non valido.
C	Dato non valido in una stringa VAL.
D	Si è premuto BREAK.
E	(non usato).
F	Il nome di SAVE è una stringa nulla. Illecito.

ZX80 MAPPA DI MEMORIA

INDIRIZZO		DESCRIZIONE
DECIMALE	ESA	
16384	4000H	Codice di errore meno uno.
16385	4001H	*Flag di controllo del Basic.
16386	4002H	Attuale numero di istruzione del Basic.
16388	4004H	Indirizzo del cursore ▯ o ▯.
16390	4006H	Numero di istruzione Basic al cursore .
16392	4008H	*Indirizzo delle variabili di programma.
16394	400AH	*Indirizzo della memoria di lavoro (INPUT tastiera).
16396	400CH	*Indirizzo dello schermo alto.
16398	400EH	*Indirizzo dello schermo basso.
16400	4010H	*Indirizzo di fine schermo.
16402	4012H	*Numero di linee dello schermo basso.
16403	4013H	Numero della prima istruzione Basic sullo schermo.
16405	4015H	Indirizzo di ▯ meno 1.
16407	4017H	Numero dell'istruzione a cui continuare.
16409	4019H	Flag di sintassi.
16412	401CH	Seme di numeri casuali.
16416	4020H	Indirizzo del primo carattere della prima variabile nell'ultimo DIM, FOR, INPUT, LET, NEXT.
16418	4020H	Valore dell'ultima variabile o espressione.
16420	4024H	Posizione della linea del successivo carattere di schermo. Da 33 (sinistra) a 2 (destra); 1 = Prima colonna, linea successiva (linea piena). 0 = prima colonna, (fine riga).
16421	4025H	*Attuale linea di schermo (0 = fondo, 23 = cima).
16422	4026H	*Indirizzo del carattere dopo le istruzioni PEEK o POKE
16424	4028H	Area di programma utente.

* Non fare un POKE. Risultati imprevedibili.

CARATTERI GRAFICI



ELENCO DEI CODICI

DEC	ESA	ZX80	ZX81	DEC	ESA	ZX80	ZX81
0	00	SPAZIO	SPAZIO	72	48		
1	01	"	▗	73	49		
2	02	▖	▝	74	4A		
3	03	▄	▐	75	4B		
4	04	▘	▚	76	4C		
5	05	▌	▞	77	4D		
6	06	▛	▜	78	4E		
7	07	▀	▟	79	4F		
8	08	▒	■	80	50		
9	09	▜	▒	81	51		
10	0A	▟	▀	82	52		
11	0B	▐	"	83	53		
12	0C	£	£	84	54		
13	0D	$	$	85	55		
14	0E	:	:	86	56		
15	0F	?	?	87	57		
16	10	(	(	88	58		
17	11	)	)	89	59		
18	12	–	>	90	5A		
19	13	+	<	91	5B		
20	14	*	=	92	5C		
21	15	/	+	93	5D		
22	16	=	–	94	5E		
23	17	>	*	95	5F		
24	18	<	/	96	60		
25	19	;	;	97	61		
26	1A	,	,	98	62		
27	1B	.	.	99	63		
28	1C	0	0	100	64		
29	1D	1	1	101	65		
30	1E	2	2	102	66		
31	1F	3	3	103	67		
32	20	4	4	104	68		
33	21	5	5	105	69		
34	22	6	6	106	6A		
35	23	7	7	107	6B		
36	24	8	8	108	6C		
37	25	9	9	109	6D		
38	26	A	A	110	6E		
39	27	B	B	111	6F		
40	28	C	C	112	70		
41	29	D	D	113	71		◇
42	2A	E	E	114	72		◊
43	2B	F	F	115	73		◊
44	2C	G	G	116	74	HOME	GRAPHICS
45	2D	H	H	117	75	EDIT	EDIT
46	2E	I	I	118	76	NEWLINE	ENTER
47	2F	J	J	119	77	RUBOUT	DELETE
48	30	K	K	120	78		K/L mode
49	31	L	L	121	79		FUNCTION
50	32	M	M	122	7A		
51	33	N	N	123	7B		
52	34	O	O	124	7C		
53	35	P	P	125	7D		
54	36	Q	Q	126	7E		numero
55	37	R	R	127	7F		cursore
56	38	S	S	128	80	■	■
57	39	T	T	129	81	"	▗
58	3A	U	U	130	82	▖	▝
59	3B	V	V	131	83	▄	▐
60	3C	W	W	132	84	▘	▚
61	3D	X	X	133	85	▌	▞
62	3F	Y	Y	134	86	▛	▜
63	3F	Z	Z	135	87	▀	▟
64	40		RND	136	88	▒	■
65	41		INKEY$	137	89	▜	■
66	42		PI	138	8A	▟	■
67	43			139	8B	▐	"
68	44			140	8C	£	£
69	45			141	8D	$	$
70	46			142	8E	:	:
71	47			143	8F	?	?

ELENCO DEI CODICI

(segue)

DEC	ESA	ZX80	ZX81	DEC	ESA	ZX80	ZX81
144	90	(	(	200	C8		COS
145	91	)	)	201	C9		TAN
146	92	–	>	202	CA		ASN
147	93	+	<	203	CB		ACS
148	94	*	=	204	CC		ATN
149	95	/	+	205	CD		LN
150	96	=	–	206	CE		EXP
151	97	>	*	207	CF		INT
152	98	<	/	208	D0		SQR
153	99	;	;	209	D1		SGN
154	9A	,	,	210	D2		ABS
155	9B	.	.	211	D3		PEEK
156	9C	0	0	212	D4	"	USR
157	9D	1	1	213	D5	THEN	STR$
158	9E	2	2	214	D6	TO	CHR$
159	9F	3	3	215	D7	;	NOT
160	A0	4	4	216	D8	,	**
161	A1	5	5	217	D9	)	OR
162	A2	6	6	218	DA	)	AND
163	A3	7	7	219	DB	NOT	<=
164	A4	8	8	220	DC	–	>=
165	A5	9	9	221	DD	+	<>
166	A6	A	A	222	DE	*	THEN
167	A7	B	B	223	DF	/	TO
168	A8	C	C	224	E0	AND	STEP
169	A9	D	D	225	E1	OR	LPRINT
170	AA	E	E	226	E2	**	LLIST
171	AB	F	F	227	E3	=	STOP
172	AC	G	G	228	E4	>	SLOW
173	AD	H	H	229	E5	<	FAST
174	AE	I	I	230	E6	LIST	NEW
175	AF	J	J	231	E7	RET	SCROLL
176	B0	K	K	232	E8	CLS	CONT
177	B1	L	L	233	E9	DIM	DIM
178	B2	M	M	234	EA	SAVE	REM
179	B3	N	N	235	EB	FOR	FOR
180	B4	O	O	236	EC	GOTO	GOTO
181	B5	P	P	237	ED	POKE	GOSUB
182	B6	Q	Q	238	EE	INPUT	INPUT
183	B7	R	R	239	EF	RAND	LOAD
184	B8	S	S	240	F0	LET	LIST
185	B9	T	T	241	F1		LET
186	BA	U	U	242	F2		PAUSE
187	BB	V	V	243	F3	NEXT	NEXT
188	BC	W	W	244	F4	PRINT	POKE
189	BD	X	X	245	F5		PRINT
190	BE	Y	Y	246	F6	NEW	PLOT
191	BF	Z	Z	247	F7	RUN	RUN
192	C0		" "	248	F8	STOP	SAVE
193	C1		AT	249	F9	CONT	RAND
194	C2		TAB	250	FA	IF	IF
195	C3			251	FB	GOSUB	CLS
196	C4		CODE	252	FC	LOAD	UNPLOT
197	C5		VAL	253	GD	CLEAR	CLEAR
198	C6		LEN	254	FE	REM	RETURN
199	C7		SIN	255	FF		COPY

ZX81 MAPPA DI MEMORIA

INDIRIZZO		DESCRIZIONE
DECIMALE	ESA	
0	0000H	ROM del MONITOR.
8192	2000H	Niente. (Usato per la ROM in alcuni dispositivi aggiuntivi).
16384	4000H	Codice di errore meno 1.
16385	4001H	*Flag di controllo del Basic.
16386	4002H	*Indirizzo della successiva istruzione dopo un RETURN.
16388	4004H	Indirizzo dell'ultimo byte disponibile del Basic +1.
16390	4006H	Modo cursore K, L, F, o G
16391	4007H	Numero dell'attuale istruzione Basic.
16393	4009H	Codice di versione ROM (0 = 8K).
16394	400AH	Numero di istruzione Basic al cursore ›.
16396	400CH	*Indirizzo dello schermo.
16398	400EH	Indirizzo della successiva posizione di stampa dello schermo.
16400	4010H	*Indirizzo delle variabili di programma.
16402	4012H	Indirizzo delle aariabili di assegnazione.
16404	4014H	*Indirizzo della memoria di lavoro (INPUT da tastiera).
16406	4016H	*Indirizzo del byte dopo PEEK o POKE.
16408	4018H	Indirizzo di ‹ meno 1.
16410	401AH	*Indirizzo della pila del modo calcolatore.
16412	401CH	*Indirizzo della fine della pila del modo calcolatore.
16414	401EH	Registro B del calcolatore.
16415	401FH	Indirizzo della memoria del calcolatore.
16417	4021H	(non usato).
16418	4022H	*Numero di linee dello schermo basso.
16419	4023H	Numero della prima istruzione Basic sullo schermo.
16421	4025H	Ultimo tasto premuto.
16423	4027H	Stato di rimbalzo sulla tastiera.
16424	4028H	Numero delle righe vuote sopra e sotto i grafici in movimento.
16425	4029H	*Indirizzo della successiva linea di istruzione Basic.
16427	402BH	Numero dell'istruzione a cui continuare.
16429	402DH	Bit del flag di sistema.
16430	403EH	Lunghezza della stringa in assegnazione.
16432	4030H	Indirizzodel successivo argomento nella tavola di sintassi.
16434	4032H	Seme di numeri casuali.
16436	4034H	Contatore del display dello schermo.
16438	4036H	Coordinata X dell'ultimo PLOT.
16439	4037H	Coordinata Y dell'ultimo PLOT.
16440	4038H	BMS dell'indirizzo della successiva posizione "LPRINT".
16441	4039H	*Numero di colonna PRINT.
16442	403AH	*Numero di linea PRINT.
16443	403BH	Bit del flag interno.
16444	403CH	Buffer della stampante.
16477	405DH	Area di memoria ausiliaria del calcolatore.
16507	407BH	(non usato).
16509	407DH	Area di programmazione dell'utente.
17407	43FFH	Fine dei sistemi 1K.
18431	47FFH	Fine dei sistemi 2K.
32767	7FFFH	Fine dei sistemi 16K.

* = non fare POKE. Risultati imprevedibili.

Gli indirizzi da 16393 (4009H) fino a 16508 (407BH) sono sempre "salvati" con il programma.

ZX81 - CHIAMATE A ROUTINE DELLA ROM

PER INTERROGARE LA TASTIERA PIÙ VELOCEMENTE CHE NON INKEY$

ESA	DEC	CODE
CD BB 02	205 187 2	CALL 02BBH
7C	124	LD A,H
C6 02	198 2	ADD A,2
38 09	56 9	JR C,+9
44	68	LD B,H
4D	77	LD C,L
CD BD 07	205 189 7	CALL 07BDH
06 00	6 0	LD B,0
4E	78	LD C,(HL)
D8	216	RET C
01 00 00	1 0 0	LD BC,0
C9	201	RET

PER SPOSTARE IL CURSORE IN UNA RIGA, COLONNA

01 col rig	1 col rig	LD BC, rig col
CD F5 08	205 245 8	CALL 08F5H
C9	201	RET

PER VISUALIZZARE UN CARATTERE SU SCHERMO

3E nn	62 nn	LD A, nn (nn=carattere)
D7	215	RST 0010H
C9	201	RET

PER VISUALIZZARE UNA STRINGA DI CARATTERI SU SCHERMO

11 dd dd	17 dd dd	LD DE ind stringa (prima byte m.s.)
01 dd dd	1 dd dd	LD BC, lung stringa (prima byte m.s.)
CD 6B 0B	205 107 11	CALL 0B6BH
C9	201	RET

PER TRACCIARE (PLOT)

01 xxyy	1 xx yy	LD BC, yyxx
3E 9B	62 155	LD A,9BH
CD B2 0B	205 178 11	CALL 0BB2H
C9	201	RET

PER CANCELLARE (UNPLOT)

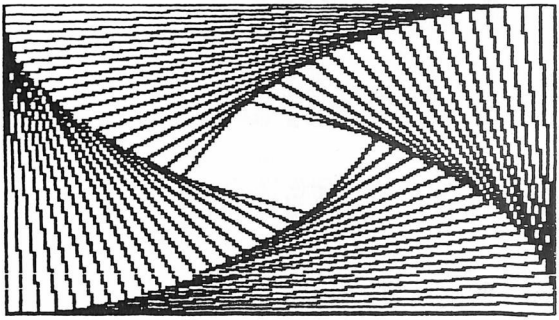
01 xx yy	1 xx yy	LD BC,yyxx
3E A0	62 160	LD A,A0H
CD B2 0B	205 178 11	CALL 0BB2H
C9	201	RET

PER PORRE "FAST"

CD 20 0F	205 32 15	CALL 0F20H
C9	201	RET

PER PORRE "SLOW"

CD 28 0F	205 40 15	CALL 0F28H
C9	201	RET



Kernal

Si notano nei due listati in assembler molte chiamate a subroutine residenti nell'ultima pagina di memoria del 6502. In realtà qui risiede la *Kernal Jump Table*. Kernal è il nome del sistema operativo del VIC 20. Esso contiene tutte le routine di input/output. Per proteggere i programmi scritti utilizzando tali routine da eventuali cambiamenti nell'organizzazione del S.O. da parte della Commodore, è stata prevista una tavola di indirizzamento standardizzata posta nell'ultima pagina di memoria disponibile. Questa tavola contiene solo istruzioni di salto che rimandano alle routine vere e proprie. Questa tavola non verrà mai cambiata (almeno si spera che non lo sia!); così per esempio, la routine CHROUT si chiamerà sempre con JSR $FFD2.

Utilizzarle è molto semplice:

1. Porre i dati nei registri di comunicazione.
2. Chiamare la routine.
3. Elaborare eventuali segnalazioni di errore.

Se vi sono errori il ritorno dalla routine viene effettuato con il carry alto e l'accumulatore contiene il codice dell'errore.

Routine utilizzate

Indir.	Nome	Descrizione
$FFBA	SETLFS	definisce un file. Utilizza i registri A, X, Y. A=num. file; X=num. periferica; Y=indirizzo secondario num. perif.=0–tast.; 1–reg.; 2–RS232; 3–schermo; 4–stamp.; 8–disco.
$FFBD	SETNAM	Assegna il nome ad un file. Registri: A,X,Y. A=lunghezza nome; Y,X=indirizzo inizio caratteri nome.
$FFC0	OPEN	Apre il file definito dalle due routine precedenti.
$FFC9	CHKOUT	Definisce un file canale di output. Reg. X=numero del file aperto.
$FFD2	CHROUT	Invia un carattere in uscita ad un canale già aperto per output. Reg. A=dato da inviare in uscita.
$FFCC	CLRCHN	Annulla tutti i canali aperti e riporta i canali di I/O ai valori di default (inp.-tastiera/ outp.-video).
$FFC3	CLOSE	Chiude un file aperto. Reg. A=num. file da chiudere

Codici degli errori letti nell'accumulatore al ritorno dalla routine

0 Routine abortita con il tasto di STOP
1 Troppi file aperti
2 Si tenta di aprire un file già aperto
3 Si tenta di effettuare operazioni su un file non aperto
5 Periferica assente o spenta
7 Si tenta di inviare in uscita dati su un file aperto per input
9 Numero di periferica sconosciuto o non permesso

La più diffusa rivista italiana di elettronica pratica allarga l'orizzonte e parla anche di personal computer.

Sperimentare, *la più autorevole e diffusa rivista di elettronica pratica, tende a perfezionare i suoi contenuti e ad ampliare l'orizzonte. Oltre alle realizzazioni per gli amatori e gli specialisti di elettronica nei più svariati campi, la rivista, da questo numero, presenterà mensilmente degli articoli dedicati al personal computer, con particolare riguardo al più diffuso di essi:* **il Sinclair.**
Hardware, software, consigli e idee da sviluppare insieme, saranno un contenuto abituale di **Sperimentare.**
Per questo motivo, **Sperimentare** *sarà d'ora in poi la rivista non solo del tecnico elettronico e dell'hobbista, ma anche il mensile dell'utente di personal computer.*
Acquista il numero in edicola con l'inserto **Sinclub.**
Un numero stimolante della rivista senza confronti.

SPERIMENTARE

UNA PUBBLICAZIONE J.C.E.

Avventura nel castello: una *"adventure"*

Difficile descrivere, a chi non li conosce, il fascino dei giochi di avventura, nei quali il computer dialoga con il giocatore (nei panni di un esploratore) descrivendo ambienti e situazioni ed "eseguendo" gli ordini ricevuti. Purtroppo la loro diffusione nel nostro paese è limitata dalla barriera della lingua: inutile dirlo, provengono da oltreoceano.

Qualcosa però comincia a muoversi anche da noi: presentiamo oggi la prova del primo gioco di avventura realizzato in Italia per Apple II: *Avventura nel castello*.

Stabilire dei criteri per giudicare la validità di un gioco non è facile, soprattutto in questo genere. Abbiamo cercato di individuare gli elementi che determinano, alla fine, il divertimento del giocatore ed il successo di un gioco di avventura.

Ambientazione e descrizioni

Avventura nel castello è ambientato in un castello scozzese, nelle vicinanze del lago di Loch Ness (*noblesse oblige*). Le descrizioni degli ambienti, vivide e ricche di particolari, danno realmente, dopo qualche minuto, l'impressione di "vivere" nel castello in prima persona. Se giocate di notte, lo fate a vostro rischio e pericolo.

A volte una lunga descrizione cela un dettaglio fondamentale, a volte invece serve soltanto a distrarre l'attenzione da qualcosa di altrettanto importante. Conviene comunque leggere sempre con molta attenzione, o si rischia di perdere delle "chiavi" essenziali.

Non sempre un dettaglio è immediatamente visibile, per cui è utile guardare gli oggetti o gli arredi presenti nelle varie stanze. Ad esempio: GUARDA IL POZZO potrebbe fornire qualche indicazione utile.

Vocabolario e "intelligenza"

L'impressione che il programma "vi capisca" deriva, più che da elaborate strutture logiche, dal numero di vocaboli e frasi plausibili alle quali viene data una risposta accettabile.

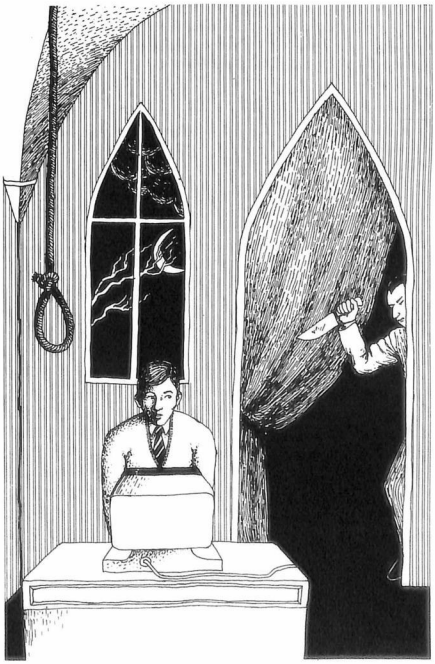
Da questo punto di vista il gioco è molto completo, con un vocabolario ampio e ricco di sinonimi. Ad esempio, per SALTA accetta anche BUTTATI, GETTATI, LANCIATI e così per molti altri vocaboli.

Anche le frasi di default, cioè le risposte ad azioni che non hanno effetto sul gioco, sono molto varie e contribuiscono all'effetto di reale esistenza dell'interlocutore. Ma attenzione: se CERCA UNA SPADA risponde CHI CERCA TROVA, non è detto che si riceva la stessa risposta per tutti gli oggetti!

L'impressione di stare parlando realmente con una persona è frutto anche di un accurato lavoro di analisi logica e sintattica: l'italiano è pieno di trappole linguistiche che l'inglese evita facilmente.

Trama e difficoltà

I giochi di avventura americani sono quasi sempre molto prevedibili: uccidi i mostri, prendi i tesori e torna a casa.



Non così *Avventura nel castello*: per prima cosa, lo scopo del gioco è "semplicemente" di uscire vivo dal castello. Ma la cosa non è molto facile e bisogna passare per un'ingarbugliata serie di problemi, trabocchetti e incontri con altri abitanti del castello (e non è detto che tutti siano nemici). Se poi ci scappa un tesoro, tanto meglio. Io, comunque, non vi ho detto niente.

Le difficoltà richiedono quasi sempre un po' di fantasia per uscirne; alcune ne richiedono un po' di più, ma un indizio si può sempre trovare scritto da qualche parte o nascosto nelle risposte ricevute. Difficilmente la via più ovvia è quella giusta.

Giochi di parole e risposte ironiche sono distribuiti un po' ovunque, celando a volte informazioni utili, a volte semplici prese in giro. A proposito, fate attenzione: è un programma molto suscettibile.

Considerazioni varie

Il tempo di risposta del programma è veramente molto breve, tanto da far pensare che sia scritto in linguaggio macchina, se gli autori non affermassero il contrario (vedi articolo su *Bit* 38). Gli accessi al disco sono limitati al minimo indispensabile ed il gioco ne risulta molto snellito.

Le risposte sono molto curate nella "regia": controllo della velocità di presentazione e qualche effetto speciale a sorpresa confermano, se ce ne fosse bisogno, che il programma è stato accuratamente messo a punto e rifinito nei minimi dettagli.

Il finale, anch'esso a sorpresa, è quasi un racconto breve con alterne vicende e conclude degnamente la lunga fatica dell'avventuriero. Sono veramente odiosi quei programmi che, al termine di un lunghissimo gioco, si limitano a stampare il solito "*congratulations, you win!*" (bravo, hai vinto).

Dettaglio non trascurabile, la protezione anti-copia è risultata inattaccabile da *Locksmith*, *Nibbles Away*, *Back-it-up* e tutti i vari *nibble-copier* di nostra conoscenza.

Conclusioni

Per divertimento, qualità e ricchezza di dettaglio *Avventura nel castello* è uno dei migliori giochi disponibili per Apple II.

Speriamo che molti programmatori italiani vogliano seguire l'esempio e contribuire alla diffusione di giochi a basso prezzo che non hanno nulla da invidiare a quelli d'oltre Atlantico. ■

Avventura nel castello è disponibile presso TECHNOCLUB a L. 42.000

Intervista con gli autori di *Avventura nel castello*

Come vi è venuta l'idea di scrivere un gioco di avventura?

Subito dopo aver conosciuto *Apple Adventure*. Ci è piaciuto molto e abbiamo pensato di fare qualcosa del genere in italiano, magari con qualche miglioria.

Migliorie di che genere?

Nel tempo di risposta, prima di tutto. Poi nella presentazione dei messaggi, diciamo pure la regia: se le risposte non sono tutte uguali, il gioco è più "mosso", più divertente.

Il programma vi è costato molto lavoro?

Quasi un anno. Ma bisogna considerare che la stesura è passata per fasi successive: inizialmente doveva essere un piccolo programma dimostrativo contenuto interamente in memoria. Poi aggiungi questo, aggiungi quello, una ciliegia tira l'altra ... ad un certo punto, con le idee molto più chiare, lo abbiamo rifatto completamente da capo. Poi la messa a punto, le routine ausiliarie, i collaudi...

Come si collauda un gioco di avventura?

Con la classica tecnica del "*playtesting*": si prendono degli amici completamente ignari e li si piazza davanti al calcolatore senza il minimo aiuto. È essenziale, in un gioco del genere, vedere le reazioni di molte persone inesperte di fronte alle medesime difficoltà. Certe cose sembrano ovvie a chi ha progettato il gioco, ma rappresentano problemi insormontabili per chi non lo conosce. Il collaudo è durato mesi, ma è risultato fondamentale per la buona riuscita del gioco.

Ci sono "passaggi" particolarmente difficili?

No. O meglio, ogni giocatore supera facilmente alcune difficoltà e si "impianta" in punti che un altro passa facilmente. Anche per questo è meglio giocare in compagnia: da soli può essere frustrante, in tanti è più divertente.

Non avete pensato ad un'avventura grafica?

Avventura nel castello è "parlata" per scelta, ma non escludiamo di presentare prima o poi un gioco del genere anche in hi-res, naturalmente con un'altra vicenda. Per ora, comunque, ci stiamo dedicando a giochi per più persone: è un campo completamente aperto, con possibilità molto interessanti specialmente per quanto riguarda l'interazione tra i giocatori. In giro si è fatto molto poco.

Un data base modulare per l'Apple

Come realizzare un archivio elettronico a scopi generali

di *Mark Pelczarski*

Questo programma è disponibile su floppy disc. Si veda il "Servizio programmi" nelle ultime pagine della rivista.

La più diffusa utilizzazione del calcolatore riguarda la memorizzazione e l'archiviamento dei dati, operazioni generalmente eseguite da programmi "data base". Questo tipo di istruzioni trasformano il calcolatore in un sistema elettronico di archiviazione, per mezzo del quale qualsiasi tipo di dati può venire memorizzato per permettere un successivo richiamo, secondo i diversi modi di memorizzazione.

Il programma che segue è un data base per l'Apple che può manipolare centinaia di informazioni in ogni file.

Passando alla struttura, il programma inizia con una routine principale (linee da 100 a 460) che presenta le varie operazioni possibili. Ad ognuna di queste operazioni corrisponde una subroutine, che agisce come un modulo separato da ogni altra parte del programma (vedi figura 1 per l'organizzazione specifica). Questi moduli sono facilmente modificabili o sostituibili senza peraltro creare problemi per il corretto funzionamento delle altre operazioni: si è così semplificato un eventuale aggiornamento o adattamento del programma alle proprie necessità.

Memorizzazione dei dati

Possiamo raffigurare la specifica organizzazione delle informazioni come una tabella di righe e di colonne, nella quale ogni riga è un record (nel caso di una *mailing list*, per esempio, un record conterrebbe nome, indirizzo, CAP, città) ed ogni colonna ha un nome (testata) sotto cui ci sono le informazioni per ciascun record (nel caso della *mailing list* le testate sarebbero *nome*, *indirizzo*, *CAP*, *città*). Per permettere un uso più efficiente della memoria, nella RAM vengono memorizzati tutti e soli gli elementi di una singola colonna, leggendo da disco le restanti colonne quando necessita una particolare informazione. È proprio per poter leggere su disco qualsiasi record in qualsiasi momento che vengono impiegati i file ad accesso casuale. Infatti, mentre in un file sequenziale si possono leggere o scrivere i record solo in sequenza (se vi serve il decimo record dovete leggere i primi nove), in un file ad accesso casuale, specificando quale record volete leggere, il calcolatore trova automaticamente la posizione richiesta e fornisce il solo record desiderato. Unica limitazione è che, per effettuare il calcolo della posizione, tutti i record devono essere della stessa lunghezza.

Per evitare di mescolare informazioni contenute su dischi (durante l'ordinamento dei record, per esempio), un secondo set di informazioni è conservato nella RAM.

I$ è il vettore che contiene gli

elementi di un campo: P% è un vettore corrispondente nel quale vengono memorizzati i numeri dei record così come compaiono sul disco. In altre parole il primo record della sequenza nella RAM potrebbe essere, dopo un ordinamento o altro, il sedicesimo sul disco: ebbene, P%(1) conterrà il valore 16.

C'è ancora qualche altra osservazione da fare riguardo la memorizzazione. Per prima cosa, ho ottimizzato lo spazio utilizzando l'elemento 0 di ogni matrice: quindi, nella matrice H$, che contiene i nomi dei campi, il nome 1 viene conservato in H$(0). Allo stesso modo, la prima registrazione viene conservata in I$(0): perciò, nell'esempio di cui sopra, sarà P%(0) a contenere il numero 16.

TI$ è il vettore contenente il record che in quel momento viene elaborato in RAM. Quando un record deve essere trascritto su disco, viene letto da TI$. Similmente, quando il record viene caricato da disco, viene direttamente memorizzato in TI$.

NI e NH sono rispettivamente il numero totale di record nel file e il numero di campi nel record. Tenete però conto che il loro valore è in realtà una unità in meno, visto che viene contato lo zero (esempio: se ci sono otto campi, NH assume il valore 7, coprendo i campi da zero a sette).

CH contiene il numero di testata, ovvero il nome del campo attualmente in uso, i cui vari elementi sono conservati in RAM.

Infine B% è il vettore che ricorda la lunghezza di ciascun campo. Così facendo è possibile determinare non solo quale record, ma anche quale campo del record si vuole leggere. L'istruzione Apple che permette di caricare un ipotetico elemento che inizia al quindicesimo byte dell'undicesima posizione è:

```
PRINT D$; "READ nome del file,
   R11, B15"
```

dove D$ è posto uguale a CONTROL-D.

Un ulteriore vantaggio nell'uso di B% è di permettere l'uso di ele-

Listato 1. *Data base.*

```
100   REM DATA BASE
120   D$=CHR$(4): REM CONTROL-D
130   DIM C$(7),C1%(7),C2%(7),F$(5): CH=0
140   HOME: PRINT"(I) CREA UN NUOVO FILE"
150   PRINT"(C) CARICA UN FILE GIA' ESISTENTE ";
160   GET A$: PRINT A$
170   IF A$="C" THEN GOSUB 480: GOTO 200
180   IF A$="I" THEN GOSUB 710: GOTO 200
190   GOTO 160
200   POKE 216,0: HOME: PRINT"(S) SALVA I DATI ATTUALI"
210   PRINT"(P) STAMPA"
220   PRINT"(A) AGGIUNGI UN RECORD"
230   PRINT"(C) CAMBIA UN RECORD"
240   PRINT"(D) CANCELLA UN RECORD"
260   PRINT"(T) SORT"
270   PRINT"(F) CATALOG"
280   PRINT"(N) NUOVO FILE"
290   PRINT"(Q) FINE"
300   PRINT: PRINT NI+1;" RECORD NEL FILE ": PRINT
      "SPAZIO PER ALTRI ";MX-NI-1
310   GET A$: PRINT A$: PRINT
320   IF A$="S" THEN GOSUB 1030: GOTO 200
330   IF A$="P" THEN GOSUB 1150: GOTO 200
340   IF A$="A" THEN GOSUB 1590: GOTO 200
350   IF A$="C" THEN SB=3: GOSUB 2290: GOTO 200
360   IF A$="D" THEN SB=4: FS=1: GOSUB 2290: GOTO 200
370   IF A$="T" THEN GOSUB 2020: GOTO 200
380   IF A$="F" THEN GOSUB 460: GOTO 200
390   IF A$="Q" OR A$="N" THEN 410
400   GOTO 200
410   IF SS=1 THEN 430
420   GOSUB 1030
430   PRINT D$;"CLOSE";F$+".DAT"
440   IF A$="N" THEN CLEAR: GOTO 120
450   END
460   PRINT D$;"CATALOG": GET A$: RETURN
470   REM SUBROUTINE DI CARICAMENTO
480   INPUT"NOME DEL FILE? ";F$
490   ONERR GOTO 630
500   PRINT D$;"OPEN";F$;".HDG"
510   PRINT D$;"READ";F$+".HDG"
520   INPUT NH,NI,MX,LK: PRINT"CI SONO!!"
530   DIM H$(NH),B%(NH+1),I$(MX),P%(MX),TI$(NH)
540   FOR I=0 TO NH: INPUT H$(I),B%(I): NEXT
550   INPUT B%(NH+1)
560   IF NI=-1 THEN 580
570   FOR I=0 TO NI: INPUT P%(I): NEXT
580   PRINT D$;"CLOSE";F$+".HDG"
590   PRINT D$;"OPEN";F$+".DAT,L";B%(NH+1)
600   IF NI=-1 THEN 620
610   GOSUB 650
620   SS=1: RETURN
630   PRINT"IL FILE NON E' SUL DISCO": GET A$: POKE 216,0:
      GOTO 140
640   REM LEGGE I CAMPI DELLA TESTATA CH
650   PRINT: FOR I=0 TO NI
660   PRINT D$;"READ";F$+".DAT,R";P%(I);",B";B%(CH)
670   INPUT I$(I)
680   NEXT
690   PRINT D$: RETURN
700   REM SUBROUTINE DI INIZIALIZZAZIONE
710   INPUT"DAI UN NOME AL FILE: ";F$
720   IF F$="" THEN 710
730   INPUT"QUANTI CAMPI? ";NH
740   IF NH<1 THEN 730
750   NH=NH-1: NI=-1: LK=-1
760   DIM H$(NH),B%(NH+1),TI$(NH): B%(0)=0
770   FOR I=0 TO NH
780   PRINT"NOME DEL CAMPO N.";I+1;: INPUT": ";H$(I)
790   INPUT"LUNGHEZZA MASSIMA:";J
800   B%(I+1)=B%(I)+J+1
810   NEXT I
820   INPUT"QUAL E'IL CAMPO PIU'LUNGO SU CUI FARAI  IL SORT? ";
      J
830   J=J-1: IF J<0 OR J>NH THEN 820
840   J=B%(J+1)-B%(J)-1
850   MX=INT((FRE(0)-2000)/(J+2))
860   DIM I$(MX),P%(MX)
870   PRINT D$;"OPEN";F$+".DAT,L";B%(NH+1)
```

(segue)

menti di lunghezza variabile cosicché nel caso di un'informazione molto corta, come il CAP, nel disco si occuperà meno spazio che non per il nome: non dimentichiamo che in un archivio lo spazio vale oro!

Caricamento o inizializzazione

Dopo avere fatto partire il programma entra in gioco la routine di inizializzazione (linea 700), che permette di partire con un nuovo file, o quella di caricamento (linea 470), che invece legge su disco i dati di un file già esistente.

Nel caso di una inizializzazione è necessario specificare il nome del nuovo file e il numero dei campi per record, seguito dal nome di ciascun campo e dalla rispettiva lunghezza. Queste informazioni vengono salvate su disco in un file separato, il cui nome è quello da voi scelto con l'aggiunta finale di .HDG (*headings*); di questo il calcolatore se ne occupa automaticamente. Nel medesimo file sono conservate anche le informazioni di B%, che vengono conseguentemente rilette ed aggiornate dopo ogni successivo impiego.

La procedura di caricamento comporta quindi la lettura di tutte le informazioni del file "nome.HDG", e pone I$ uguale agli elementi del primo campo del file "nome.DAT", che contiene i dati dell'utente. Un'ulteriore subroutine viene usata per caricare gli elementi dalla testata CH (che all'inizio vale zero), poiché questa funzione sarà necessaria più tardi in diversi punti del programma. Questa routine inizia alla linea 640.

Aggiunta dati, memorizzazione e stampa

Dopo aver creato un file, la prima cosa che vorrete fare sarà inserirci alcuni dati. Scegliendo "A" dal menù, potrete introdurre ogni volta un ulteriore record, per mezzo della subroutine di aggiunta dati (1580). L'input dei nuovi dati viene

Segue Data base.

```
880   SS=0: RETURN
890   REM LEGGI IL RECORD I IN TI$
900   PRINT: R=P%(I)
910   FOR J1=0 TO NH
920   PRINT D$;"READ";F$+".DAT,R";R;",B";B%(J1)
930   INPUT TI$(J1)
940   NEXT
950   PRINT D$: RETURN
960   REM SCRIVI IL RECORD R DA T$
970   PRINT: FOR J1=0 TO NH
980   PRINT D$;"WRITE";F$+".DAT,R";R;",B";B%(J1)
990   PRINT TI$(J1)
1000  NEXT
1010  PRINT D$: RETURN
1020  REM SUBROUTINE DI SCRITTURA
1030  PRINT: ONERR GOTO 1130
1040  PRINT D$;"OPEN";F$+".HDG"
1050  PRINT D$;"WRITE";F$+".HDG"
1060  PRINT NH: PRINT NI: PRINT MX: PRINT LK
1070  FOR I=0 TO NH: PRINT H$(I): PRINT B%(I): NEXT
1080  PRINT B%(NH+1)
1090  IF NI=-1 THEN 1110
1100  FOR I=0 TO NI: PRINT P%(I): NEXT
1110  PRINT D$;"CLOSE";F$+".HDG"
1120  SS=1: RETURN
1130  PRINT"ERRORE DEL DISCO": GET A$: GOTO 200
1140  REM SUBROUTINE DI STAMPA
1150  IF NI=-1 THEN GOSUB 2890: RETURN
1160  PRINT"(S) SPECIFICA FORMATO, (D) STANDARD ";: GET A$:
      PRINT
1170  IF A$="S" THEN GOSUB 2920: FS=2: GOTO 1200
1180  IF A$<>"D" THEN 1160
1190  FS=1
1200  PRINT"(S) SCHERMO, (P) STAMPANTE";: GET A$: PRINT
1210  IF A$="P" THEN SB=2: GOTO 1250
1220  IF A$<>"S" THEN 1200
1230  SB=1: PRINT: PRINT"DOPO OGNI RECORD BATTI <ESC> PER TORNAR
      EAL MENU' - UN TASTO QUALUNQUE": PRINT"PER CONTINUARE"
1240  PRINT: PRINT
1250  PRINT"<BATTI UN TASTO>": GET A$: GOSUB 2300
1260  RETURN
1270  REM STAMPA UN RECORD
1280  IF SB=2 THEN PRINT D$;"PR#1"
1290  ON FS GOSUB 1340,1400
1300  IF SB=2 THEN PRINT D$;"PR#0": GOTO 1320
1310  IF SB<>4 THEN GET A$: IF A$=CHR$(27) THEN RS=1
1320  RETURN
1330  REM STAMPA UN RECORD CON FORMATO DEFAULT
1340  PRINT: PRINT"RECORD ";I+1: PRINT
1350  FOR J=0 TO NH
1360  PRINT H$(J),TI$(J)
1370  NEXT J
1380  RETURN
1390  REM STAMPA UN RECORD CON FORMATO DATO
1400  J=1: T=0: B$=""
1410  J1=VAL(MID$(F$(T),J,1)): J=J+1
1420  IF J1<5 THEN N=VAL(MID$(F$(T),J,2)): J=J+2
1430  ON J1 GOTO 1440,1450,1460,1480,1500,1570
1440  A$=H$(N): GOTO 1540
1450  A$=TI$(N): GOTO 1540
1460  B$=LEFT$(B$,N-1):
      IF LEN(B$)<N-1 THEN FOR J2=LEN(B$) TO N-2: B$=B$+" ":
      NEXT
1470  GOTO 1550
1480  PRINT B$: IF N>1 THEN FOR J2=2 TO N: PRINT: NEXT
1490  B$="": GOTO 1550
1500  IF J>LEN(F$(T)) THEN T=T+1: J=1
1510  J2=J
1520  IF MID$(F$(T),J2,1)<>"!" THEN J2=J2+1: GOTO 1520
1530  A$=MID$(F$(T),J,J2-J): J=J2+1
1540  B$=B$+A$
1550  IF J>LEN(F$(T)) THEN T=T+1: J=1
1560  GOTO 1410
1570  PRINT B$: RETURN
1580  REM SUBROUTINE DI IMMISSIONE
1590  SS=0:: NI=NI+1
1600  FOR J=0 TO NH
1610  GOSUB 1700
1620  NEXT J
1630  IF LK=-1 THEN R=NI: GOTO 1670
```

(segue)

effettuato da una piccola routine alla linea 1690, che richiede le informazioni campo per campo. Dopo l'introduzione dei dati, la registrazione viene trascritta su disco mediante un GOSUB 970, un'ennesima subroutine usatissima che salva T$ nel record R del disco. Alcune istruzioni della routine d'immissione impiegano una variabile LK di cui si parlerà nella sezione che riguarda il cancellamento delle registrazioni.

Una volta che le informazioni si trovano nel data base, per utilizzarle efficacemente serve un qualche metodo per listarle o stamparle. Entra allora in gioco la routine di stampa che serve a listare un record, o più di uno, su video o su stampante, con un formato standard o con un altro che potete definire a piacere. Questa routine è completata da quella di ricerca, in modo che potete selezionare un qualsiasi sottoinsieme di dati. Possono venire specificati per la ricerca fino ad otto criteri, con la possibilità di AND o di OR.

Dopo aver specificato le debite limitazioni (possono anche non esercene), scegliete INIZIO per far partire la stampa.

Come già detto, si può creare un proprio formato mediante la apposita subroutine compresa nel programma: riga per riga, è da specificare se si vogliono dati, titoli, tab, o addirittura stringhe che non dipendono dal file. Queste informazioni di formato vengono salvate su di un file separato con un nome scelto dall'utente, in modo da poter essere richiamate all'occorrenza. Per ritornare all'esempio iniziale, ciò permette, con gli stessi dati, di stampare etichette, fatture, intestazioni di lettere o qualsiasi altro formato possa servire.

I dati vengono automaticamente registrati ogniqualvolta si finisce di usare il programma; nel caso, comunque, si voglia effettuare più di una copia del file, anche nel corso del programma (ad esempio prima di un lungo ordinamento), esiste l'opzione di memorizzazione che salva su disco una copia della testata in uso (con tutti i suoi campi) e

Segue Data base.

```
1640  R=LK
1650  PRINT D$;"READ";F$+".DAT,R";R
1660  INPUT LK: PRINT D$
1670  GOSUB 970: P%(NI)=R: I$(NI)=TI$(CH)
1680  RETURN
1690  REM ACCETTA UN CAMPO
1700  T=B%(J+1)-B%(J)-1
1710  PRINT H$(J);: INPUT" : ";TI$(J)
1720  IF LEN(TI$(J))>T THEN TI$(J)=LEFT$(TI$(J),T)
1730  RETURN
1740  REM SUBROUTINE DI CAMBIAMENTO
1750  PRINT: PRINT"(C) CAMBIA IL DATO": PRINT
      "(K) CONSERVA IL DATO": PRINT
      "(R) CONSERVA IL RESTO DEL RECORD"
1760  PRINT: PRINT"RECORD ";I+1
1770  CS=1: RS=0: FOR J=0 TO NH
1780  PRINT: PRINT H$(J);" : ";TI$(J);"  ";
1790  IF RS=1 THEN PRINT: GOTO 1850
1800  GET A$: IF A$<>"C" AND A$<>"K" AND A$<>"R" THEN 1800
1810  PRINT A$: IF A$="K" THEN 1850
1820  IF A$="R" THEN RS=1: GOTO 1850
1830  GOSUB 1700
1840  CS=0
1850  NEXT J
1860  RS=0
1870  IF CS=0 THEN GOSUB 970: I$(I)=TI$(CH)
1880  RETURN
1890  REM SUBROUTINE DI CANCELLAZIONE
1900  PRINT: PRINT"CANCELLI QUESTO RECORD?";
1910  GET A$: IF A$<>"S" AND A$<>"N" THEN 1910
1920  PRINT A$: IF A$="N" THEN 2000
1930  PRINT D$;"WRITE";F$+".DAT,R";P%(I)
1940  PRINT LK: PRINT D$
1950  LK=P%(I)
1960  FOR I1=I+1 TO NI
1970  I$(I1-1)=I$(I1): P%(I1-1)=P%(I1)
1980  NEXT I1
1990  NI=NI-1: SS=0: I=I-1
2000  RETURN
2010  REM SUBROUTINE DI SORT
2020  IF NI=-1 THEN GOSUB 2890: RETURN
2030  PRINT: FOR J=0 TO NH
2040  PRINT"(";J+1;") ";H$(J)
2050  NEXT J
2060  INPUT"SECONDO QUALE CAMPO ORDINO?";J1
2070  J1=J1-1
2080  IF J1<0 OR J1>NH THEN RETURN
2090  IF J1<>CH THEN CH=J1: GOSUB 650
2100  PRINT"(A) ASCENDENTE, O (D) DISCENDENTE": GET A$
2110  IF A$="A" THEN A=1: GOTO 2140
2120  IF A$="D" THEN A=2: GOTO 2140
2130  GOTO 2100
2140  FOR I=0 TO NI-1
2150  T=I
2160  FOR I1=T+1 TO NI
2170  PRINT I;"  ";I1
2180  ON A GOTO 2190,2210
2190  IF I$(I1)<I$(T) THEN T=I1
2200  GOTO 2220
2210  IF I$(I1)>I$(T) THEN T=I1
2220  NEXT I1
2230  IF T=1 THEN 2260
2240  T$=I$(T): I$(T)=I$(I): I$(I)=T$
2250  J1=P%(T): P%(T)=P%(I): P%(I)=J1
2260  NEXT I
2270  SS=0: RETURN
2280  REM SUBROUTINE DI RICERCA
2290  IF NI=-1 THEN GOSUB 2890: RETURN
2300  I1=0: I2=NI: J=0: C1%(0)=-1: BS=1
2310  HOME: PRINT"CRITERIO DI RICERCA:": PRINT
2320  PRINT"0) NUMERO DI RECORD"
2330  FOR I=0 TO NH: PRINT I+1;") ";H$(I): NEXT I
2340  PRINT: PRINT NH+2;") INIZIO"
2350  VTAB 21: INPUT"SCEGLI:";I: IF I<0 OR I>NH+2 THEN 2350
2360  IF I=NH+2 THEN C1%(J)=-1: GOTO 2490
2370  C1%(J)=I-1
2380  VTAB 22: PRINT"(1) MINORE  (2) UGUALE (3)MAGGIORE  ":
      GET A$: IF A$<"1" OR A$>"3" THEN 2380
2390  C2%(J)=VAL(A$)
2400  VTAB 23: PRINT"CONFRONTATO CON:";: IF C1%(J)=-1 THEN 2430
```

(segue)

del file degli indirizzi indice (P%). I singoli record sono automaticamente trascritti su disco non appena vengono introdotti.

Correzione e cancellazione dati

Una volta introdotte le informazioni in memoria, è importante poterle modificare. Con la routine di cancellazione (1900) si possono rimuovere definitivamente le informazioni dal file. Per cancellare un record il numero di questo (NI) deve venire diminuito di un'unità, ed il buco che si crea riempito spostando di un posto ogni record: è evidente il gran numero di dati da spostare. Ma dato che abbiamo già un vettore (P%) che indica la posizione fisica della registrazione del file su disco, non dobbiamo far altro che cancellare il numero del record dalla lista indice. Se nessun indice punta al "buco", esso sarà ignorato. Purtroppo, nel caso di una massiccia eliminazione di dati, il numero dei "buchi" risulta elevato, causando alla fine uno spreco di spazio sul disco. La soluzione è di tenere una traccia di dove questi buchi si trovano, utilizzandoli appena possibile invece di aggiungere nuovi dati in coda (fisicamente!) al file su disco.

Il mezzo è LK che contiene appunto l'informazione del primo "buco" disponibile che può essere usato per nuovi dati.

Se LK=−1 il file è continuo, e le nuove registrazioni si accodano. Nel momento in cui un buco appare come risultato di una cancellazione, il valore di LK viene depositato nella posizione del record eliminato (che, per noi, ora equivale ad uno spazio vuoto), mentre LK assume il valore della più recente posizione svuotata. Se questo buco viene riutilizzato, il numero contenuto in quella posizione ritorna in LK. Il risultato è una lista concatenata di registrazioni vuote con il buco più recente indicato da LK, ed ogni buco precedente indicato da quello liberatosi subito dopo.

Per correggere o cambiare un record esistente (modificare ad esem-

Segue Data base.

```
2410  INPUT" ";C$(J): J=J+1: IF J>7 THEN 2500
2420  GOTO 2310
2430  INPUT" ";I: IF I<1 OR I>NI+1 THEN 2430
2440  I=I-1
2450  IF C2%(J)=1 THEN I2=I
2460  IF C2%(J)=2 THEN I1=I: I2=I
2470  IF C2%(J)=3 THEN I1=I
2480  GOTO 2310
2490  IF J<2 THEN 2520
2500  VTAB 22: PRINT
      "1) IL DATO DEVE ESAUDIRE TUTTE LE        CONDIZIONI":
      PRINT"2) IL DATO PUO' ESAUDIRE UNA CONDIZIONE": GET A$:
      IF A$<"1" OR A$>"2" THEN 2500
2510  BS=VAL(A$)
2520  RS=0: J1=C1%(0)
2530  DS=0: FOR J=0 TO 7
2540  IF C1%(J)=-1 THEN J=7: GOTO 2560
2550  IF J1<>C1%(J) THEN J1=-2
2560  NEXT
2570  IF J1>-1 AND J1<>CH THEN CH=J1: GOSUB 650
2580  IF J1=-2 THEN DS=1
2590  I=I1-1: FOR I3=I1 TO I2: I=I+1
2600  IF DS=0 THEN TI$(CH)=I$(I): GOTO 2620
2610  GOSUB 900
2620  AS=0: FOR J=0 TO 7
2630  IF C1%(J)=-1 THEN J=7: GOTO 2770
2640  ON C2%(J) GOTO 2650,2670,2720
2650  IF TI$(C1%(J))<=C$(J) THEN 2740
2660  GOTO 2760
2670  IF TI$(C1%(J))=C$(J) THEN 2740
2680  IF RIGHT$(C$(J),1)<>"*" THEN 2760
2690  T=LEN(C$(J))-1: IF LEN(TI$(C1%(J)))<T THEN 2760
2700  IF LEFT$(TI$(C1%(J)),T)=LEFT$(C$(J),T) THEN 2740
2710  GOTO 2760
2720  IF TI$(C1%(J))>=C$(J) THEN 2740
2730  GOTO 2760
2740  IF BS=2 THEN AS=1: J=7
2750  GOTO 2770
2760  IF BS=1 THEN AS=2: J=7
2770  NEXT J
2780  IF AS=0 AND BS=1 THEN 2800
2790  IF AS<>1 THEN 2850
2800  IF DS=0 THEN GOSUB 900
2810  IF SB<>3 THEN GOSUB 1280
2820  IF SB=3 THEN GOSUB 1750
2830  IF SB=4 THEN GOSUB 1900
2840  IF RS=1 THEN I3=I2
2850  NEXT I3
2860  PRINT: PRINT"ECCO TUTTO!!": GET A$: PRINT
2870  RETURN
2880  REM SUBROUTINE DI ERRORE
2890  PRINT"NON CI SONO CAMPI IN MEMORIA."
2900  FOR I=1 TO 1000: NEXT: RETURN
2910  REM FORMATO STAMPA
2920  IF F$(0)="" THEN 2960
2930  PRINT"STESSO FORMATO?";: GET A$: PRINT
2940  IF A$="S" THEN RETURN
2950  IF A$<>"N" THEN 2930
2960  PRINT"(L) CARICA FORMATO, O (C) CREA FORMATO";: GET A$:
      PRINT
2970  IF A$="C" THEN 3080
2980  IF A$<>"L" THEN 2960
2990  ONERR GOTO 3070
3000  INPUT"NOME DEL FORMATO:";A$
3010  PRINT D$;"OPEN";A$+".FMT"
3020  PRINT D$;"READ";A$+".FMT"
3030  INPUT NF
3040  FOR J=0 TO NF: INPUT F$(J): NEXT
3050  PRINT D$;"CLOSE";A$+".FMT"
3060  RETURN
3070  PRINT"FORMATO NON TROVATO": GET A$: GOTO 200
3080  NF=0: J=0: F$(0)=""
3090  HOME: PRINT
      "PARTI DALL'ANGOLO SUPERIORE SINISTRO    E LAVORA SU OGNI
      RIGA."
3100  PRINT
      "1:TITOLO 2:DATO 3:TAB 4:PROSSIMA RIGA   5:STRINGA 6:FINE"
      :INPUT J1
3110  IF J1<1 OR J1>6 THEN 3100
3120  F$(NF)=F$(NF)+STR$(J1): J=J+1
```

(segue)

pio un indirizzo della *mailing list*) il programma utilizza nuovamente la routine di ricerca (sia la stampa che la correzione e la cancellazione usano questa routine per permettere all'utente di scegliere le registrazioni necessarie). La vera routine di correzione parte dalla linea 1740, e permette all'utente di scegliere in ogni record fra il conservare un elemento, cambiarlo o conservare il resto del record stesso. Tutto ciò che succede durante la correzione è un banale rimpiazzamento dei dati.

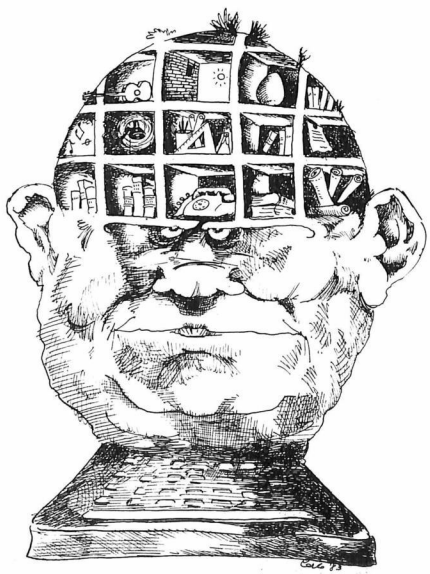
Ordinamento e ricerca

La subroutine di ordinamento può essere considerata il neo di questo programma; per mantenere una relativa semplicità si è scelta una routine di semplice inserzione: essa ricerca fra tutti i campi di una testata quello con il contenuto più piccolo (o più grande), lo mette a capo della lista e ripete l'operazione con gli altri elementi, tirandone fuori il secondo, il terzo, ... Se non altro la lentezza è compensata dalla brevità!

La routine di ricerca, già altre volte citata, visto il suo ruolo di appoggio alle altre, serve all'utente per introdurre fino ad otto criteri di scelta tra i suoi dati. Ogni condizione specifica se (a) l'elemento sotto un certo nome (testata) (b) debba essere minore, maggiore, uguale a (c) uno specifico valore, anche alfanumerico. Se vengono specificate due o più condizioni, c'è anche la possibilità di determinare se si vuole un AND (la registrazione deve soddisfare tutte le condizioni), o un OR (almeno una condizione deve essere soddisfatta). Ogni elemento valido viene memorizzato in C1%, C2% e C$ rispettivamente per le condizioni (a), (b), (c) di cui sopra. Quando una registrazione soddisfa le esigenze della routine di ricerca, viene chiamata la subroutine di partenza (stampa, correzione o cancellazione) mediante il controllo di SB, che indica appunto la funzione desiderata.

Segue Data base.

```
3130  ON J1 GOTO 3140,3140,3170,3170,3210,3270
3140  FOR T=0 TO NH: PRINT T+1;") ";H$(T): NEXT
3150  INPUT"QUALE?";T: T=T-1: IF T<0 OR T>NH THEN 3150
3160  GOTO 3180
3170  INPUT"QUANTI?";T: IF T<1 OR T>99 THEN PRINT" FUORI RANGE."
      :GOTO 3170
3180  A$=STR$(T): IF T<10 THEN A$="0"+A$
3190  F$(NF)=F$(NF)+A$: J=J+2
3200  GOTO 3240
3210  INPUT"STRINGA:";A$: A$=A$+"!"
3220  IF LEN(A$)+J>255 THEN NF=NF+1: J=0: F$(NF)=""
3230  F$(NF)=F$(NF)+A$: J=J+LEN(A$)
3240  IF J>252 THEN NF=NF+1: J=0: F$(NF)=""
3250  IF J>252 THEN NF=NF+1: J=0: F$(NF)=""
3260  GOTO 3100
3270  INPUT"NOME DEL FORMATO:";A$
3280  ONERR GOTO 3340
3290  PRINT D$;"OPEN";A$+".FMT"
3300  PRINT D$;"WRITE";A$+".FMT"
3310  PRINT NF: FOR J=0 TO NF: PRINT F$(J): NEXT
3320  PRINT D$;"CLOSE";A$+".FMT"
3330  RETURN
3340  PRINT"ERRORE DEL DISCO": GET A$: GOSUB 3270
```



A questo punto è necessaria una osservazione riguardante le due routine di ordinamento e ricerca. Questo programma non distingue fra stringhe e dati numerici, per cui tutti i dati sono considerati stringhe alfanumeriche. A ciò si devono gli strani risultati che rivela ad esempio l'ordinamento di numeri come 7, 15, 273. Essendo trattati come alfanumerici, l'ordine finale è 15, 273, 7. L'unica maniera per evitare tale inconveniente è per il momento quello di usare degli zeri di testa nei valori numerici: 007, 015, 273 risulteranno nella giusta sequenza. ■

ZX SPECTRUM

L'uso della tastiera nei programmi di movimento

Marcello Spero

La differenza principale fra il comando INPUT e la funzione INKEY$ sta nel fatto che usando il primo, il computer aspetta pazientemente che voi premiate il tasto ENTER, mentre con la seconda non si fermerà affatto, limitandosi a controllare se qualche tasto è premuto nel momento in cui la incontra. Infatti INPUT è concepito per l'introduzione di dati, numerici o alfanumerici, di qualsiasi lunghezza, con la possibilità di un loro controllo ed eventuale correzione prima di dare il consenso con il tasto ENTER alla loro interpretazione; permette quindi di adeguare la macchina ai nostri tempi di azione, dove INKEY$ pretende che siamo noi ad adeguarci alla macchina. Queste caratteristiche, se da un lato rendono rischiosa e "senza pietà" l'introduzione di dati tramite INKEY$, dall'altro rendono questa funzione estremamente efficace nei programmi di tipo "dinamico", dove cioè una situazione in continua evoluzione può venire aggiornata istantaneamente da tastiera. L'unico controllo che viene effettuato sul carattere "catturato" in questo modo è quello sulla sua reale esistenza; la macchina verifica cioè che il tasto, o l'insieme di tasti nel caso dei caratteri ottenuti con uno degli SHIFT, restituisca uno dei caratteri compresi nel set in memoria; se questo non avviene il computer non tiene conto del carattere: come se non fosse stato premuto alcun tasto. Questo controllo, peraltro utilissimo nei programmi tipo "word processor" o "macchina da scrivere" per esempio, in cui l'involontaria pressione di due tasti potrebbe dare origine ad errori, rende impossibile la combinazione di più tasti dove questa avrebbe senso: tipicamente quando questi rappresentano direzioni di movimento; in questo caso, infatti, oltre a quattro tasti per le direzioni fondamentali occorrono altri quattro tasti per le diagonali, con notevole complicazione d'uso. Tutto questo può essere evitato, rendendo "comprensibile" alla macchina la pressione contemporanea di più tasti, con l'uso della funzione IN. Questa funzione, che insieme al suo stretto parente, il comando OUT, proviene direttamente dall'assembler, ci permette di fare letteralmente "quello che vogliamo" della tastiera: IN e OUT rappresentano un grosso passo avanti del Sinclair Basic dello Spectrum rispetto a quello dello ZX81; permettono infatti la gestione delle eventuali periferiche senza dover ricorrere al linguaggio macchina. A questo proposito occorre aprire una parentesi: possiamo dividere in prima approssimazione un computer in CPU (l'unità centrale di calcolo, il "cervello", che nel nostro caso è un circuito integrato Z80A) e tutto ciò che sta al di fuori della CPU; questi congegni esterni vengono distinti dalla CPU in due categorie, a seconda di come le sono collegati: memoria e periferiche; quindi, così come esistono locazioni di memoria numerate da 0 in avanti, esistono "locazioni per le periferiche", dette I/O *port* (molti traducono "porta", forse "canale di ingresso e uscita" risulta più chiaro, anche se la parola canale è usata in informatica con altri significati) numerate anch'esse da 0 in su.

Essendo la CPU capace di scambiare dati sia con la memoria, che con le periferiche, possiamo chiederle di leggere o scrivere per noi in una locazione di memoria o in un port; per quanto riguarda la memoria questo avviene con le ben note PEEK e POKE, mentre per le periferiche vengono usate IN e OUT, esattamente allo stesso modo; quindi, se con PRINT PEEK 32000 otteniamo il valore conservato nella locazione di memoria 32000, con PRINT IN 32000 otterremo il valore "trasmesso" dalla periferica collegata al port 32000.

Chiusa questa lunga parentesi è importante ricordare che nello ZX Spectrum il video attinge direttamente alla memoria; quindi *è* memoria, nel senso che il contenuto di un certo numero di locazioni, viene visualizzato senza bisogno che sia trasferito, dove invece tutti gli altri congegni (registratore, altoparlante e appunto tastiera) sono visti come periferiche. Sul manuale sono indicati gli indirizzi di tutte le periferiche previste; in particolare alla tastiera sono dedicati 8 port, ciascuno per mezza riga di tasti; la forma binaria del valore riportato dalla lettura di uno di questi port darà degli 1 per i tasti non premuti e degli 0 per quelli premuti, con il bit meno significativo (quello che vale 1) per il tasto più esterno, e avanti fino al quinto bit (quello che vale 16) per il più interno; essendo solo cinque i tasti di ogni mezza riga gli altri tre bit non servono a niente: però ci sono, e sono sempre 1. Quindi se nessun tasto è premuto tutti i port della tastiera daranno il valore 255, ossia il binario 11111111.

In questo modo possiamo fare dei controlli sullo stato della tastiera esattamente come facevamo con INKEY$, con la differenza che se, per esempio, sono premuti sia il tasto 2 che il 4, mentre con INKEY$

ZX SPECTRUM

avremmo ottenuto di ritorno una stringa vuota, qui ci viene restituito il valore 345, cioè 11110101, in cui partendo da destra, cioè dal bit meno significativo, il primo bit è il tasto 1, non premuto, il secondo è il 2 premuto, e così via fino agli ultimi tre bit, che non si considerano.

Ovviamente per ottenere la lettura della mezza riga contenente i tasti che ci interessano dobbiamo inserire nella istruzione Basic il numero del port corrispondente, che in questo caso è il 63486, mentre per esempio nel caso ci interessasse la lettera A sarebbe il 65022; se il controllo dovesse avvenire per tasti posti in semirighe diverse occorrerà una istruzione per ogni semiriga.

```
   5 LET x=0: LET y=0
  10 IF IN 61438=247 THEN LET y=
y+1
  20 IF IN 61438=239 THEN LET y=
y-1
  30 IF IN 61438=251 THEN LET x=
x+1
  40 IF IN 61438=243 THEN LET x=
x+1: LET y=y+1
  50 IF IN 61438=235 THEN LET x=
x+1: LET y=y-1
  60 IF IN 63486=239 THEN LET x=
x-1
  70 PLOT x,y
  80 GO TO 10
 100 LET x=x+(IN 61438=251 OR IN
 61438=243 OR IN 61438=235)-(IN
63486=239): LET y=y+(IN 61438=24
7 OR IN 61438=243)-(IN 61438=239
 OR IN 61438=235): PLOT x,y: GO
TO 100
```

Il listato chiarisce senz'altro meglio questo meccanismo: abbiamo un punto di coordinate (x,y) che possiamo spostare sullo schermo premendo i tasti dei cursori; fin qui niente di particolare, ma questa volta i movimenti diagonali si ottengono con la pressione contemporanea dei due cursori corrispondenti alle componenti della direzione desiderate. Lo stesso risultato è qui ottenuto in due modi diversi: il primo con le linee 10-80, il secondo con la sola linea 100; il primo metodo è più veloce, anche se occupa più spazio, poiché le operazioni relative alle condizioni non verificate non vengono effettuate, mentre nel secondo caso sì; i tempi sono rispettivamente 8.5 secondi e 10.6 secondi per tracciare una linea attraverso l'intero schermo; ambedue i modi richiedono la presenza della linea 5 per inizializzare le variabili delle coordinate.

I valori relativi al controllo di ogni port sono espressi in forma decimale per comodità, ma per maggior chiarezza, o per evitare calcoli potremmo benissimo scrivere, per esempio:

```
10 IF IN 61438=BIN 11110111 THEN LET y=y+1
```

al posto della linea 10; appare così chiaro che ci riferiamo al quarto tasto partendo dall'esterno. Purtroppo, mancando nel Basic il modo di leggere un singolo bit, siamo costretti ad avere un controllo per ogni possibile combinazione di bit che ci interessa, in uno stesso port; è questo il caso delle linee 10-50, dove vengono testate le condizioni di:

linea 10	cursore "su" premuto
linea 20	cursore "giù" premuto
linea 30	cursore "destra" premuto
linea 40	cursori "su" e "destra" premuti
linea 50	cursori "giù" e "destra" premuti

mentre per il cursore "sinistra", essendo questo in un altro port, non serve alcun abbinamento con altre condizioni: infatti volendo realizzare un movimento diagonale verso l'alto e a sinistra, per esempio, saranno verificate contemporaneamente le condizioni di linea 10 e 60, che sono indipendenti l'una dall'altra. Una nota curiosa è che mentre la pressione contemporanea dei cursori "su" e "giù" non provoca il verificarsi di alcuna condizione, quella dei cursori "destra" e "sinistra", proprio per l'appartenenza dei due tasti a port diversi, rende verificate due condizioni, che però si annullano a vicenda: x=x+1 e x=x−1.

Le applicazioni di questo metodo non si limitano certo al campo grafico; comunque un altro uso molto utile per programmi di gioco può essere lo stabilire una priorità fra due o più tasti: per esempio nel caso vengano premuti contemporaneamente i cursori "su" e "giù", tanto per rimanere al nostro caso, possiamo far sì che uno prevalga sull'altro così da non bloccare il movimento; un'altra applicazione potrebbe essere l'attribuire un significato ai tasti SHIFT premuti da soli (es.: CAPS SHIFT per lo spostamento a sinistra, SPACE per quello a destra e SYMBOL SHIFT per sparare...). ■

TEXAS TI 99/4A

Input e stampa estesa

Riceviamo dal sig. Filippo Cerulo di Vitulano (Benevento) queste brevi routine che permettono di superare alcune limitazioni del 99/4A.

Sono un assiduo lettore della vostra rivista, nonché un malato di bittomania cronica. Dopo molti sacrifici, da sei mesi sono in possesso di un TI 99/4A. Propongo dunque alla vostra attenzione un paio di routine di cui da un po' di tempo non riesco a fare a meno in nessun programma.

Il 99/4A è un ottimo calcolatore, ma, a meno di non comprare il modulo "Extended Basic", ha qualche piccolo limite, però largamente giustificato dal basso prezzo d'acquisto.

I problemi più grandi nascono dall'impossibilità di stampare messaggi e ricevere input se non sull'ultima riga del display, con conseguente scroll verso l'alto di tutto il video.

Confesso che tutto ciò all'inizio mi ha creato dei problemi soprattutto per l'impossibilità di stampare (e cancellare) frasi di commento a figure e grafici prodotti per mezzo dell'ottimo set di istruzioni del TI Basic.

Ma con un po' di buona volontà alla fine ho scritto le due brevi routine che vi propongo e che eliminano tutti i problemi.

Fate girare le due subroutine con il breve programma che accludo per capire bene come funzionano. Alla comparsa del simbolo di prompt (>) battete una qualsiasi stringa. In caso di errore basta premere FCTN3 (ERASE) e ribattere la stringa.

```
2000 REM SBR INPUT ESTESO
2010 K=0
2020 IMP$=""
2030 CALL HCHAR(RIN,CIN+K,62)
2040 CALL KEY(5,COD,ST)
2050 IF ST=0 THEN 2040
2060 IF COD 7 THEN 2090
2070 CALL HCHAR(RIN,CIN,32,K+1)
2080 GOTO 2010
2090 IF COD=13 THEN 2140
2100 IMP$=IMP$&CHR$(COD)
2110 CALL HCHAR(RIN,CIN+K,COD)
2120 K=K+1
2130 IF COD=32 THEN 2030 ELSE 2040
2140 RETURN
```

Listato 1. *Subroutine "input esteso".*

```
100 CALL CLEAR
110 READ R,C,S1$
120 GOSUB 1000
130 READ RIN,CIN
140 GOSUB 2000
150 PRINT" LA STRINGA E' ";IMP$
160 DATA 10,5,BATTI UNA STRINGA,13,5
170 END
```

Listato 2. *Programma principale.*

```
1000 REM STAMPA ESTESA
1010 N=LEN(S1$)
1020 FOR I=1 TO N
1030 S2$=SEG$(S1$,I,1)
1040 V1=ASC(S2$)
1050 CALL HCHAR(R,C-1+I,V1)
1060 NEXT I
1070 RETURN
```

Listato 3. *Subroutine "stampa estesa".*

Non mi dilungo nell'illustrare le singole istruzioni che per i possessori di TI 99/4A saranno abbastanza chiare. Dalla subroutine 1000, sacrificando la chiarezza, possono essere eliminate due istruzioni: i tempi di esecuzione delle due versioni sono però pressoché uguali. Preciso che i due apici nella riga 2020 non rappresentano uno spazio ma una stringa nulla. ■

VIC 20

Routine di interrogazione dei joystick per il VIC 20

Francesco Cordes

La porta di controllo del VIC 20 offre la possibilità di connettere unità esterne di pilotaggio e segnalazione, per verificare informazioni e utilizzarle direttamente nei programmi (ad esempio per cambiare la direzione nei giochi).

Come la tastiera o le altre porte (utente, seriale o cassetta), anche la porta di controllo viene controllata mediante i due chip di interfaccia I/O (VIA 6522), ma, al contrario di quanto avviene per le altre funzioni I/O, non viene supportata dal sistema operativo.

Se quindi si vuole collegare un joystick, per pilotare le direzioni in un programma, lo si può fare solamente mediante una specifica routine di interrogazione dei joystick.

Riportiamo due sottoprogrammi che svolgono questa funzione, uno in assembler ed uno in Basic. Sia il programma in Basic che quello in linguaggio macchina (listato 1 e listato 2) lavorano nello stesso modo: interrogano il joystick e forniscono una specificazione di direzione e l'informazione se è stato premuto il tasto del "fuoco". La routine Basic usa per questo le variabili D (direzione) e F (fuoco) per una successiva verifica, mentre il programma assembler scrive queste informazioni nelle locazioni di memoria 251 (D) e 252 (F).

```
10 POKE 37154,127:REM REGISTRO DI DIREZIONE DATI
20 REM                SULL'OUTPUT  01111111
30 REM VERIFICARE LA DIREZIONE DAL REGISTRO DI OUTPUT
35 REM DELLE PORTE B E A
40 D=(PEEK(37152) AND 128)+(PEEK(37151) AND 28)
50 POKE 37154,255:        REM RIPRISTINARE IL VECCHIO VALORE
60 F=PEEK(37151) AND 32:REM E VERIFICARE IL TASTO DEL FUOCO
```

Listato 1. *Programma in Basic.*

In ambedue i casi, per indicare la direzione scelta, viene utilizzato lo schema di figura 1.

Per il fuoco vi sono solo due stati, rappresentati dai valori 0 (bottone premuto) e 32 (bottone non premuto). L'utilizzazione di questi valori è lasciata poi al programmatore, che inserendo queste routine in un programma può eseguire una veloce verifica della direzione scelta.

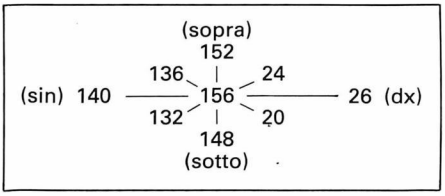
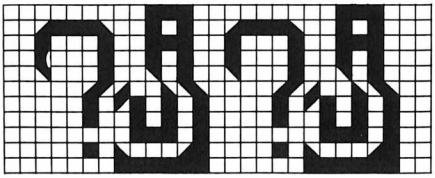


Figura 1. *Schema delle direzioni.*

Per ottenere anche dal Basic la velocità della routine in linguaggio macchina, si è approntata anche una routine di caricamento da Basic (listato 3). Con questo programma di caricamento si può memorizzare il programma in linguaggio macchina in un supporto



```
0000                    DIRECT=$FB
0000                    FIRE=$FC
0000                    DDRB=$9122
0000                    OPRA=$911F
0000                    ORB=$9120
0000                    START=$033C
0000                          *=START
033C  A9 7F             LDA #$7F
033E  8D 22 91          STA DDRB
0341  AD 20 91          LDA ORB
0344  29 80             AND #$80
0346  85 FB             STA DIRECT
0348  A9 FF             LDA #$FF
034A  8D 22 91          STA DDRB
034D  AD 1F 91          LDA OPRA
0350  29 1C             AND #$1C
0352  18                CLC
0353  65 FB             ADC DIRECT
0355  85 FB             STA DIRECT
0357  AD 1F 91          LDA OPRA
035A  29 20             AND #$20
035C  85 FC             STA FIRE
035E  60                RTS
035F                    .END
```

Listato 2. *Programma in assembler.*

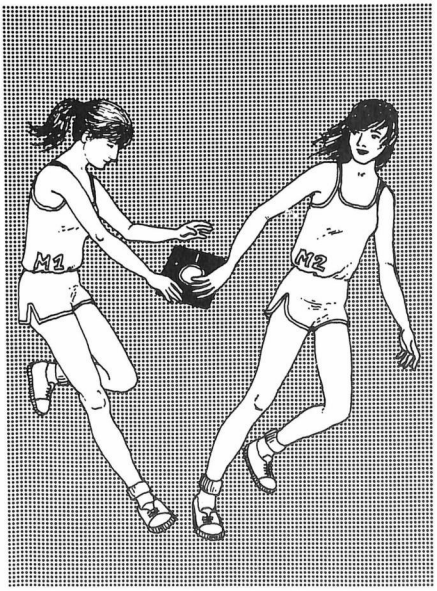
VIC 20

adatto, in questo caso su cassetta, per poterlo poi richiamare con SYS(indirizzo).

I valori con le informazioni del pilotaggio del joystick possono sempre venire letti, e quindi verificati, dopo la chiamata di questo sottoprogramma mediante PEEK dalle locazioni 251 e 253. Si deve fare attenzione ancora a una cosa: è possibile pilotare i joystick solo se i tasti del registratore non sono premuti.

```
20 READ A: REM INDIRIZZO DI PARTENZA
30 FOR I=0 TO 34: READ X: POKE A+I,X: NEXT
40 PRINT "CHIAMATA: SYS ";A
80 REM DATI
90 DATA 828
100 DATA 169,127,141,34,145,173,32,145,41,128,133,251,169,
    255,141,34
110 DATA 145,173,31,145,41,28,24,101,251,133,251,173,31,145,
    41,32
120 DATA 33,252,96
```

Listato 3. *Routine di caricamento da Basic.* ■



COMMODORE PET/CBM

Overlay

Ettore M. Albani

Spesso, sia per motivi di comprensibilità, sia per mancanza di spazio in memoria, è necessario dividere un'intera procedura in più parti.

Questa suddivisione richiede, il più delle volte, di trasferire il contenuto delle variabili da un programma ad un altro. Può essere comodo, per esempio, definire alcune variabili di utilità all'inizio della procedura (come la data odierna, il numero od il nome dei file, ecc.), senza più doverle creare nuovamente all'inizio di ogni programma.

Tale accorgimento, comunemente chiamato "overlay", consente, tra l'altro, di ridurre i tempi necessari al programmatore per sviluppare la procedura, assolvendo così ad uno dei principi base della programmazione strutturata.

Molti computer hanno un Basic che consente di trasferire le variabili da un programma ad un altro. Purtroppo il Basic del CBM non ha questa possibilità, ma con un piccolo espediente si ottiene lo stesso risultato.

La famosa "pagina zero" del CBM (cioè le prime 256 locazioni della memoria RAM) può venire in aiuto. Come è noto, in essa il sistema operativo memorizza, all'atto dell'aaccensione del computer, una serie di contatori e flag, molto utili sia al sistema operativo stesso, sia al programmatore. Molti di questi contatori possono essere modificati. È proprio quello che fa il sistema operativo quando deve aggiornare, per esempio, il numero della linea Basic attualmente in esecuzione, oppure il flag indicante il tipo di variabile in elaborazione (numerica od alfabumerica).

In questo articolo interessano alcune locazioni della pagina zero, in cui sono memorizzati i seguenti puntatori:

Dec.	Esa	Puntatore
42-43	002A-002B	Inizio area variabili
44-45	002C-002D	Fine area variabili
46-47	002E-002F	Fine area vettori

Il sistema operativo pone nelle locazioni 42-43 l'indirizzo del primo byte libero dopo il testo Basic; la locazione corrispondente a questo indirizzo segue, infatti, i tre 00 che indicano la fine del programma. Si può assumere, perciò, che quest'indirizzo invidivui non solo l'inizio delle variabili ma anche la fine del testo Basic.

COMMODORE PET/CBM

Quando il programma genera le variabili, viene aggiornato il contenuto delle locazioni 44-45 e 46-47.

Una considerazione importante ai fini dell'overlay riguarda il comando LOAD. Come si è visto, il caricamento di un nuovo programma provoca l'aggiornamento dei puntatori di inizio e di fine delle variabili: questo accade solamente se il comando è diretto, cioè se viene eseguito battendo LOAD sulla tastiera. Se, viceversa, questo comando è presente all'interno di un programma, l'aggiornamento non avviene, a meno che il programma caricato non occupi più memoria RAM del chiamante. Tutti i contenuti delle variabili possono, perciò, essere trasmessi in questo modo.

Ne consegue che per avere l'overlay è sufficiente che il primo programma caricato sia di lunghezza maggiore o tutt'al più uguale a quella del più lungo dei programmi successivi.

Generalmente, però, accade che il primo programma sia il più corto, in quanto deve svolgere le sole funzioni di inizializzazione della procedura e di richiamo degli altri programmi. Si rende necessario, perciò, allungarlo.

Un metodo molto usato è quello di aggiungere in coda al programma una serie di linee di commento (REM) contenenti asterischi (o qualsiasi altro carattere) fino a raggiungere la lunghezza desiderata. Questa operazione, però, provoca un inutile spreco di memoria.

Un secondo metodo, senz'altro più valido, è quello di variare i puntatori di inizio variabili.

Per fare ciò bisogna calcolare l'occupazione in pagine (1 pagina = 256 byte) del programma più lungo e memorizzare questo numero nelle locazioni 43, 45 e 47 (che sono il byte più significativo di ciascun puntatore).

Ci sono diversi modi per conoscere la lunghezza di un programma: per questa applicazione il metodo più sicuro (e più semplice) è quello di caricare in memoria il programma più lungo ed eseguire:

PRINT PEEK (43)

Sarà così lo stesso sistema operativo ad indicare qual è la pagina di RAM in cui termina il testo Basic. Si ricordi che il programma più lungo può essere trovato confrontando i risultati dell'istruzione "PRINT FRE(0)".

La prima riga del programma chiamante sarà, perciò:

100 POKE 43,XX: POKE 45,XX: POKE 47,XX

dove XX è uguale al numero di pagine occupate dal programma più lungo. Poiché viene utilizzato il byte più significativo di ciascun puntatore, trascurando quello meno significativo, è preferibile, per sicurezza, aggiungere 1 ad XX. Inoltre, si ricordi che l'indirizzo di partenza del testo Basic è $0400 (1024 in decimale), il che corrisponde alla quarta pagina della mappa di memoria: risulta evidente, perciò, che XX conterrà la lunghezza in pagine del programma più 4.

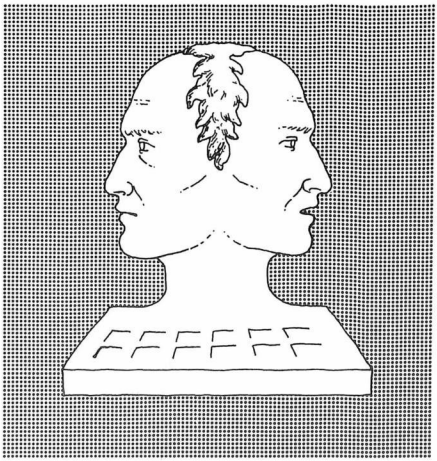
L'esempio che segue indicherà meglio come agire.

Si supponga di avere una procedura composta di cinque programmi della seguente lunghezza:

Nome prog.	PEEK (43)
START	15
GESTIONE 1	32
GESTIONE 2	50
STAMPE	45
SERVIZI	21

Come si vede, il programma più lungo è GESTIONE 2 con l'area delle variabili che inizia in pagina 50, il che corrisponde ad una lunghezza reale del programma di 46 pagine. START è il programma chiamante, la cui prima linea deve contenere i POKE per l'overlay e, nelle successive, l'inizializzazione di tutte le variabili. La prima linea di START sarà quindi:

100 POKE 43,51: POKE 45,51: POKE 47,51



COMMODORE PET/CBM

Precisazioni

Si noti che tutti i programmi in esecuzione durante l'overlay avranno, per il sistema operativo, la stessa lunghezza (nell'esempio precedente occuperanno fino alla 51.ma pagina). Perciò, non è bene eseguire un SAVE dopo aver fermato l'esecuzione di un programma: verrà memorizzato in 47 pagine (51 meno 4) anche un programma lungo in realtà 17 (21 meno 4), come SERVIZI. Si ricordi di caricare sempre il programma in modo diretto prima di modificarlo. In caso di errore esiste, comunque, un sistema per rimediare. Bisogna entrare in modo monitor con il comando SYS 8 e rintracciare le tre locazioni alla fine del programma che contengono 00; l'indirizzo della locazione successiva dovrà essere sostituito a quello presente in 42-43, 44-45, 46-47.

Si ricordi di non definire costanti alfanumeriche nel primo programma chiamante (START nell'esempio precedente), sia in frasi DATA sia in assegnazioni specifiche: quando si passerà da un programma ad un altro il contenuto di queste costanti cambierà. Il motivo è molto semplice e lo si può capire con un esempio.

Generalmente il programma chiamante definisce anche l'insieme delle variabili utilizzate da tutta la procedura. Si supponga che il programma START inizi come nel listato 1.

Il Basic del CBM, ed, in generale, il Basic Microsoft, crea un "vocabolario" delle variabili; insieme al nome di ognuna sarà memorizzato l'indirizzo di RAM in cui si trova il contenuto. In particolare per le variabili alfanumeriche a cui è assegnata una costante alfanumerica (stringa), l'indirizzo punta al primo carattere della stringa interna al testo Basic. Nell'esempio, la locazione in cui è memorizzato il contenuto di K$ è in corrispondenza del carattere "C" di "CYBER HARDWARE & SOFTWARE". È evidente che quando si carica un altro programma in overlay, tale indirizzo rimarrà lo stesso, assegnando così a K$ un contenuto apparentemente strano ma che, in realtà, è una parte del testo Basic attualmente in memoria. Per ovviare a questo inconveniente è sufficiente modificare così la linea 130:

```
130 K$="CYBER HARDWARE & SOFTWARE"
  +""
```

In questo modo viene eseguita un'operazione su stringhe ed il contenuto di K$ sarà trasferito nella posizione di RAM esterna al testo Basic, insieme ai contenuti di tutte le altre variabili alfanumeriche. La stessa operazione deve essere eseguita per le frasi DATA alfanumeriche: dopo l'istruzione READ (variabile stringa) si scriva: (variabile stringa)= (variabile stringa)+"".

```
  1 REM * START
100 POKE 43,51: POKE 45,51: POKE 47,51
110 A=5: B=0: C=0
120 A$="": B$="": C$=""
130 K$="CYBER HARDWARE & SOFTWARE"
140 H$="VIA SCAPACCHIO' 48/B"
150 W$="35030 SELVAZZANO (PD)"
160 T$="TEL. 049/638681"
.... ............
.... ............
.... ............
1000 REM * MENU' PRINCIPALE
.... ............
.... ............
.... ............
```

Listato 1.

```
  1 REM * START
 10 IF W<>0 THEN 1000
100 POKE 43,50: POKE 45,50: POKE 47,50
110 A=5: B=0: C=0: W=1
120 A$="": B$="": C$=""
130 K$="CYBER HARDWARE & SOFTWARE"+""
140 H$="VIA SCAPACCHIO' 48/B"+""
150 W$="35030 SELVAZZANO (PD)"+""
160 T$="TEL. 049/638681"+""
.... ............
.... ............
.... ............
1000 REM * MENU' PRINCIPALE
.... ............
.... ............
.... ............
```

Listato 2.

Un'altra considerazione molto importante riguarda i POKE iniziali del programma START. Questi, come del resto l'inizializzazione delle variabili, devono essere eseguiti soltanto una volta alla partenza della procedura. Il ritorno al programma START da uno degli altri programmi deve prevedere il salto delle linee di assegnazione. Se questo non accade, si provoca la cancellazione di tutte le variabili, poiché viene modificato il contenuto delle locazioni 45 e 47 (fine delle variabili e dei vettori). È necessario, perciò, inserire la linea 10 come controllo prima dei POKE (vedi listato 2). ■

NUOVA ELETTRONICA

Esa e ASCII

Vincenzo Scaffidi

Questo programma vuole essere un piccolo aiuto per tutti coloro che desiderano approfondire la conoscenza del proprio computer.

Esa e ASCII legge dal disco un programma e lo propone sul monitor e sulla stampante in formato esadecimale con accanto, riga per riga, la traduzione ASCII di quanto contenuto.

Si possono leggere tutti i programmi, di utente o di sistema, con l'unica limitazione di DIR/CMD che, pur essendo presente nell'indice del disco, non è un programma vero e proprio.

Per usare il programma è sufficiente rispondere con il nome di ciò che si vuole listare specificando successivamente se si desidera usare la stampante o il solo monitor.

Attenzione: se si fornisce il nome di un programma non presente sul disco, il computer, proprio per come è progettato il DOS, interpreterà la nostra richiesta come il desiderio di iniziare un file random e quindi ne scriverà il nome accanto ai programmi realmente esistenti. Tutto qui, ma la cosa potrebbe essere fonte di confusione, per cui è meglio, qualora ciò avvenisse, cancellare subito il file con il comando KILL.

```
260 CLS:CLEAR 1000:DIM A(256),B$(256):X=1
270 GOSUB 360:OPEN"R",1,F$
280 FIELD 1,255 AS A$,1 AS AA$
290 FOR W=1 TO LOF(1):GET1:NEXTW:GOSUB 640
300 IF X>W THEN 700 ELSE GET1,X
310 FOR Y=1 TO255
320 GOSUB 460
330 NEXT Y:A(Y)=ASC(AA$):GOSUB 490
340 X=X+1:GOSUB 610:GOTO 300
350 '
360 REM ###      Presentazione    ###
370 '
380 PRINT@180,;:UNDERON:REVON:PRINT" ###   HEX - ASCII    ### ";
    :REVOFF:UNDEROFF
390 PRINT@810,"Il programma permette di visualizzare o stampare"
400 PRINTTAB(10)"il contenuto dei programmi presenti su un disco "
410 PRINTTAB(10)"sia in esadecimale che in formato ASCII."
420 PRINT@1690,;:INPUT"A QUALE PROGRAMMA SEI INTERESSATO ";F$
430 PRINT@1770,"DESIDERI ANCHE LA STAMPA SU CARTA ? <S/N>"
440 GOSUB 670:R$=RR$:RETURN
450 '
460 REM ###  LETTURA DEL SINGOLO BYTE  ###
          E TRASFORMAZIONE IN ESA
470 '
480 B$(Y)=MID$(A$,Y,1):A(Y)=ASC(B$(Y))
490 C$=%A(Y)
500 PRINT RIGHT$(C$,2)" ";
510 IF INT(Y/16)=Y/16 OR Y=256 THEN GOSUB 540
520 RETURN
530 '
540 REM ###  STAMPA DEI CARATTERI ASCII ###
550 '
560 PRINT"   ";:FOR Z=Y-15 TO Y
570 IF A(Z)<32 OR A(Z)>126 THEN PRINT ".";:GOTO 590
580 PRINT CHR$(A(Z));
590 NEXT Z:PRINT:PRINTTAB(5);:RETURN
600 '
610 REM ### STAMPA INTESTAZIONE SETTORE ###
620 '
630 GOSUB 650
640 CLS:PRINT@81,;:REVON:UNDERON:PRINT" ## "F$" ## - Settore n." X,
    "   Settori totali:" W;:UNDEROFF:REVOFF:PRINT@245,;:RETURN
650 IF R$="S" THEN LCOPY
660 PRINT@1780,;:BLINKON:PRINT"Premi un tasto per continuare";:
670 RR$=%EYIN$:IFRR$=""THEN 670
680 BLINKOFF:RETURN
690 '
700 REM ### TROVATO EOF ###
710 '
720 CLS
730 CLOSE1:PRINT@1690,"DESIDERI ALTRE LETTURE ? <S/N>"
740 GOSUB 670:IF HTRANS(RR$)="S" THEN RUN
750 CLS:END
```

Variabili principali

A$, AA$	Vengono usate per memorizzare i dati che via via vengono letti dal disco.
B$	Contiene il byte da convertire in esadecimale.
W	Numero dei settori del disco che contengono il programma da leggere.
X,Y, A, RR$	Variabili di comodo.
C$	Valore esadecimale su quattro cifre.
R$	Se contiene S si desidera la stampa su carta.

```
** SYS13/SYS ** - Settore n. 2                    Settori totali: 5

4F 55 54 20 4F 46 20 53 01 82 00 4E 54 52 49 4E   OUT OF S...NTRIN
47 20 53 50 41 43 45 00 53 54 52 49 4E 47 20 54   G SPACE.STRING T
4F 4F 20 4C 4F 4E 47 00 53 54 52 49 4E 47 20 46   OO LONG.STRING F
4F 52 4D 55 4C 41 20 54 4F 4F 20 43 4F 4D 50 4C   ORMULA TOO COMPL
45 58 00 43 41 4E 27 54 20 43 4F 4E 54 49 4E 55   EX.CAN'T CONTINU
45 00 4E 4F 20 52 45 53 55 4D 45 00 52 45 53 55   E.NO RESUME.RESU
4D 45 20 57 49 54 48 4F 55 54 20 45 52 52 4F 52   ME WITHOUT ERROR
00 55 4E 50 52 49 4E 54 41 42 4C 45 20 45 52 52   .UNPRINTABLE ERR
4F 52 00 4D 49 53 53 49 4E 47 20 4F 01 82 80 4E   OR.MISSING O...N
50 45 52 41 4E 44 00 42 41 44 20 46 49 4C 45 20   PERAND.BAD FILE
44 41 54 41 00 44 49 53 4B 20 42 41 53 49 43 20   DATA.DISK BASIC
46 45 41 54 55 52 45 00 55 4E 44 45 46 49 4E 45   FEATURE.UNDEFINE
44 20 55 53 45 52 20 46 55 4E 43 54 49 4F 4E 00   D USER FUNCTION.
46 49 45 4C 44 20 4F 56 45 52 46 4C 4F 57 00 49   FIELD OVERFLOW.I
4E 54 45 52 4E 41 4C 20 45 52 52 4F 52 00 42 41   NTERNAL ERROR.BA
44 20 46 49 4C 45 20 23 00 46 49 4C 45 20 4E 4F   D FILE #.FILE NO




** BASIC/CMD ** - Settore n. 1                    Settori totali: 23

05 06 42 41 53 49 43 42 01 EB 00 4D C3 46 5E C3   ..BASICB...M.F^.
8E 55 C3 49 5E C3 55 56 C3 4C 5E C3 EB 61 C3 31   .U.I^.UV.L^..a.1
62 C3 42 62 C3 2D 5E C3 30 5E C3 33 5E C3 C4 56   b.Bb.-^.0^.3^..V
C3 14 57 C3 49 63 C3 AB 60 C3 7C 62 C3 7B 62 C3   ..W.Ic..`.|b.{b.
6F 60 C3 7B 5F C3 9B 60 C3 BE 06 C3 C0 63 C3 B7   o`.{_..`.....c..
58 C3 E6 60 C3 E5 60 C3 2F 5B C3 44 60 C3 56 57   X..`..`./X.D`.VW
3E BF EF C3 79 56 C3 FC 5F C3 BE 59 C3 33 60 C3   >...yV.._..Y.3`.
7D 5B C3 BC 5B C3 A1 60 C3 7C 57 C3 BE 59 C3 CD   }[..[..`.|W..Y..
59 C3 78 5F C3 15 5A C3 9A 5B C3 99 5B C3 65 5B   Y.x_..Z..[..[.e[
C3 84 57 C3 0D 59 C3 63 5E C3 9C 57 C3 51 5B 4E   ..W..Y.c^..W.Q[N
55 4F 56 41 20 45 4C 45 54 54 52 4F 4E 49 43 41   UOVA ELETTRONICA
20 42 41 53 49 43 20 56 45 52 53 49 4F 4E 20 47    BASIC VERSION G
2E 20 31 2E 30 0D 53 20 42 41 53 49 43 20 4F 52   . 1.0.S BASIC OR
49 47 49 4E 41 4C 4C 59 20 41 55 54 48 4F 52 45   IGINALLY AUTHORE
44 20 41 4E 44 20 43 4F 50 59 52 49 47 48 54 45   D AND COPYRIGHTE
44 20 42 59 0D 01 00 00 52 F5 E5 D7 FE 60 38 07   D BY....R....`8.
```

APPLE II

Applesoft Cataloger

Rosario Amasino

Applesoft Cataloger è un breve programma che consente di operare sul "catalog" di un disco senza che sia necessario battere sulla tastiera in forma completa i comandi DOS. L'immediatezza della situazione e la velocità con cui ci si trova ad operare sono le sue caratteristiche principali. La parte attiva di maggior interesse di questo programma è la routine che acquisisce informazioni sullo stato e sulla natura dei file leggendo quanto è presentato sullo schermo dal programma stesso.

Appena viene lanciato, Applesoft Cataloger presenta un catalog opportunamente modificato del disco su cui si trova, accompagnato da una serie di opzioni per il trattamento dei file in esso contenuti.

Può risultare conveniente inizializzare dischi con questo programma, o comunque caricarlo su dischi in cui ci siano "lavori in corso".

Applesoft Cataloger lavora con qualunque tipo di file.

Cosa fa

In seguito al RUN compare sullo schermo un elenco dei file contenuti nel disco associati ad una lettera tra parentesi quadre; la lettera indica quel file nel corso del programma.

Sulla parte inferiore del video scorre una scritta che indica le operazioni possibili ed il programma attende che l'operatore scelga tra queste battendo un tasto.

A questo punto si può uscire dal programma battendo "5".

Altra possibilità immediatamente disponibile è il RUN di uno dei programmi presentati, che si ottiene premendo la lettera corrispondente.

Per le altre opzioni occorre prima scegliere il trattamento e poi indicare il file su cui deve essere eseguito.

I trattamenti possibili sono LOAD, LOCK, UNLOCK e DELETE. Quando questa scelta è stata effettuata battendo 1, 2, 3 o 4, sulla parte bassa dello schermo compare la richiesta per il nome del file: si tratta ancora di battere la lettera adatta. Un lampeggiamento accompagna la scelta dell'opzione DELETE. Se è stato ordinato un RUN o un LOAD il programma esamina il tipo di file, se sia cioè un file binario, un file Applesoft o un file di testo e formula il comando DOS nella maniera opportuna.

Quando l'ultimo degli input richiesti è stato fornito il programma porta a termine le istruzioni indicate e ritorna il controllo all'utente.

Come funziona

La linea 100 stampa il catalog ed il parametro B controlla a quale riga dello schermo si è giunti. B viene fissato ad un massimo di 22 per la linea 110 che aggiunge sul catalog le parentesi quadre e le letture sostitutive dei nomi.

La linea 120 prepara la scritta da far scorrere sullo schermo, cosa che viene eseguita alla linea 130 che provvede anche a caricare RUN in B$, la stringa che contiene i comandi da eseguire. Interroga inoltre la tastiera (K=PEEK(−16384)) per vedere l'azione che

```
10    REM APPLESOFT CATALOGER
100   TEXT: HOME: D$=CHR$(4): PRINT
      D$"CATALOG": B=PEEK(37)-2:
      IF B>22 THEN B=22
110   T=0: CH=4: FOR CV=0 TO 23: GOSUB 1000:
      IF C<>160 THEN POKE P-1,219:
      POKE P,T+193: POKE P+1,221: T=T+1:
      S=CV
120   NEXT CV: VTAB 24:
      A$=
      "BATTI LA LETTERA PER IL RUN,OPPURE LOA
      D=1,LOCK=2,UNLOCK=3,DELETE=4,FINE=5..."

130   B$="RUN": HTAB 1: PRINT LEFT$(A$,39);:
      A$=MID$(A$,2)+LEFT$(A$,1):
      K=PEEK(-16384):
      IF K<128 THEN FOR K=1 TO 75: NEXT K:
      K=FRE(0): GOTO 130
140   POKE-16368,0: K=K-176:
      IF K<1 OR K>5 THEN 300
200   HTAB 1: CALL-868: IF K=5 THEN END
210   PRINT"BATTI LA LETTERA PER IL ";:
      IF K=1 THEN B$="LOAD"
220   IF K=2 THEN B$="LOCK"
230   IF K=3 THEN B$="UNLOCK"
240   IF K=4 THEN B$="DELETE": FLASH
250   PRINT B$;: CALL-198: NORMAL: GET K$:
      K=ASC(K$)-48
300   IF K<17 OR K>T+16 THEN 130
310   CH=1: CV=S-T+K-16: GOSUB 1000:
      IF C=194 AND(B$="RUN" OR B$="LOAD")
      THEN B$="B"+B$
315   IF C=212 THEN B$="EXEC"
320   FOR CH=6 TO 39: GOSUB 1000:
      B$=B$+CHR$(C): NEXT CH: HTAB 1:
      CALL-868: PRINT B$: PRINT D$;B$
330   END
1000  C1=INT(CV/8): L2=CV-C1*8:
      P=1024+128*L2+40*C1+CH: C=PEEK(P):
      RETURN
```

APPLE II

viene indicata. La linea 140 riabilita la tastiera (POKE−16368,0) e manda a varie routine di controllo del parametro ottenuto.

I comandi DOS vengono eseguiti alla linea 320. Il testo del comando è contenuto in B$. B$ viene formata alle linee 130, 220, 230, 240 ed è integrata se è il caso alle linee 310 e 315.

La linea 310 corregge RUN e LOAD in BRUN e BLOAD se il file di cui si tratta è binario. La linea 315 corregge RUN e LOAD in EXEC se il file di cui si tratta è un file di testo.

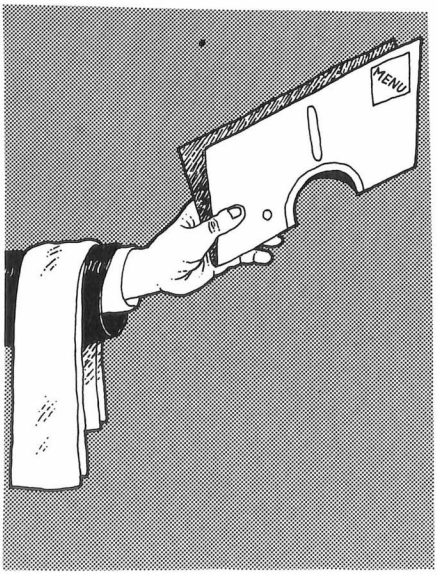
Le informazioni necessarie per decidere queste correzioni sono ottenute dalla lettura dello schermo effettuata dalla routine alla linea 1000.

Questa routine legge nella pagina di testo a bassa risoluzione (1024-2047) ed è usata dal programma per le informazioni sul tipo di file e per la mascheratura del catalog sul video.

CALL−868 allinea il cursore al margine sinistro dello schermo.

CALL−198 produce un "beep".

Il programma occupa 775 byte. ■



SINCLAIR ZX81

Caratteri minuscoli su ZX Printer

Enrico Ferreguti

Quando si confronta la ZX printer con altre stampanti più costose, si nota che la cosa che le manca è la possibilità di stampare caratteri minuscoli e maiuscoli allo stesso tempo. La stampante ZX non può trattare i caratteri minuscoli per il semplice motivo che non sono contenuti nella mappa dei caratteri della ROM da 8 K. Si può risolvere questo problema creando un nuovo set di caratteri e riscrivendo le routine di stampa.

Il programma seguente crea un set di caratteri di 1024 byte, sostituendolo con quello standard di 512 byte contenuto nella ROM.

Il programma non occupa la solita posizione all'inizio dell'area BASIC contenuto in una REM essendo questa una posizione pericolosa in caso di NEW e scomoda qualora si voglia caricare da nastro dei programmi che se ne servono (poiché il programma caricato sostituisce quello precedente).

Si è così provveduto a fornire una routine che sposta il codice macchina e il nuovo set dalla REM in cui viene caricato ad una area sopra alla RAMTOP, così facendo il programma è reso immune da NEW e LOAD.

Caricate il programma del listato 1 e premete RUN. Inserite una ad una le righe esadecimali di listato 2.

Quando avrete inserito tutte le linee il programma

```
10  REM  HEX CHARGER
20  FOR N=16514 TO 16901
30  INPUT I$
40  SCROLL
50  PRINT N,I$
55  FOR J=1 TO LEN I$ STEP 2
60  POKE N, CODE I$(Jj)*16+CODE I
    $ (J+1) −476
62  LET N=N+1
65  NEXT J
70  LET N=N−1
80  NEXT N
```

Listato 1. *Programma per inserire le linee esadecimali del listato 2.*

Dal Basic al Pascal

Prima parte : le variabili

di *Ronald W. Anderson*

Questo è il primo di una serie di articoli particolarmente adatti ai programmatori Basic che vogliono imparare il Pascal.
Questo primo articolo si occupa delle variabili; i prossimi si occuperanno via via di tutti gli altri aspetti dei due linguaggi. Numerosi programmi ed esempi facilitano l'apprendimento della filosofia del Pascal.

Traduzione di *Nancy E. Fischer*

Sia il Basic che il Pascal usano le variabili. Il nome di una variabile è semplicemente un'"etichetta" che definisce dove un valore deve essere memorizzato nella memoria del computer. Le variabili possono essere di diversi tipi. Quasi tutti gli interpreti Basic, di cui si servono attualmente i microcomputer, permettono l'uso dell'aritmetica in virgola mobile. Cioè, i numeri che non sono "interi", possono essere rappresentati come numeri interi più una parte decimale. Ad esempio, π può essere rappresentato come 3.14159265... La prima cifra, 3, è un numero intero e .14159265 è la parte frazionaria o decimale del numero. Anche se il Basic non usa questo termine, i numeri in virgola mobile sono generalmente chiamati *numeri reali* (REAL). Alcuni Basic hanno solo l'aritmetica intera (INTEGER). Molti dei Basic più potenti permettono all'utente di specificare se una variabile è di tipo INTEGER o REAL. Alcuni contrassegnano le variabili intere aggiungendo un % al nome della variabile. A% è perciò una variabile INTEGER. Invece la variabile A è una variabile REAL. Alcuni Basic permettono un'istruzione DEFINT che definisce le variabili elencate come variabili INTEGER.

Il Basic ha un altro tipo di variabile, chiamata stringa. Il Basic usa universalmente il segno $ come identificatore di stringa. Al contrario il Pascal non ha le funzioni di stringa. Esso ha comunque un tipo di variabile che si usa per contenere un carattere, che si chiama CHAR. Ciò è dovuto al fatto che nel periodo in cui il Pascal è stato creato, la maggior parte dei computer erano di tipo "batch", invece che avere l'ingresso di dati interattivi da un terminale, come è più frequente oggi.

Nomi delle variabili

Il Basic permette generalmente l'uso delle lettere dalla A alla Z per le variabili numeriche e da A$ a Z$ per le stringhe. La maggior parte dei Basic permettono più variabili, potendo aggiungere una cifra dopo la lettera. A1, Z9, V3 ecc. sono nomi validi. Molti dei Basic più recenti permettono nomi di variabili con due lettere come PI, WT etc. Queste possono essere molto utili per comunicare il tipo di uso della variabile nel programma. Se avete mai provato a scrivere un lungo programma in Basic, avrete probabilmente desiderato avere a disposizione nomi di variabili più significativi che non da A a Z.

Alcuni dei Basic attualmente usati hanno anche un *precompilatore* che permette l'uso di nomi lunghi. Si tratta di un tentativo di superare le limitazioni del Basic e di rendere possibile al programmatore la produzione di pro-

grammi più leggibili e significativi. Il pre-compilatore permette una versione più "verbosa" del programma con nomi delle variabili più significativi e subroutine con nome, che saranno accettati dall'interprete del Basic. L'uso del pre-compilatore ha lo svantaggio di aggiungere un passaggio tra la scrittura e l'esecuzione del programma. Cioè, il processo di programmazione non è più interattivo e quindi si perde il maggior vantaggio dell'interprete.

Pascal si riferisce ai nomi delle variabili, ai nomi dei file, ai nomi delle funzioni e delle procedure (ogni nome scritto dall'utente) come a "identificatori". Gli identificatori del Pascal possono essere di qualsiasi lunghezza (generalmente è limitata in molte implementazioni, per ragioni pratiche, a circa 32 caratteri). Essi devono comporre un'unica parola. LUNGHEZZA, RAGGIO, IPOTENUSA e PROPORZIONE sono tutti identificatori validi. Le parole concatenate da sottolineature come GIORNI_DELLA_SETTIMANA sono anch'esse valide, ma alcune implementazioni richiedono solo caratteri alfanumerici come GIORNIDELLASETTIMANA.

Altre implementazioni usano internamente solo i primi otto caratteri, e bisogna fare attenzione ad usare solo quelli. Cioè IPOTENUSA_A e IPOTENUSA_B saranno considerati come le stesse variabili in alcune versioni di Pascal in quanto vengono usati solo IPOTENUS, cioè i primi otto caratteri. Usando A_IPOTENUSA e B_IPOTENUSA li si distinguerà sicuramente. Si dovranno perciò porre le parti che servono a distinguere il nome di una variabile lunga all'inizio del nome.

Sintassi e punteggiatura dell'istruzione

Il Pascal usa una punteggiatura che è diversa da quella dei programmi Basic. È necessaria una spiegazione prima di presentare programmi in Pascal. Nel Basic, i due punti ":" sono generalmente usati per separare le istruzioni multiple su una linea. (A volte è usata "/".) Nel Pascal i due punti indicano un'assegnazione di qualche tipo. L'esempio sottostante indica uno degli usi nell'assegnazione di un TYPE (tipo) ad una variabile.

```
B: INTEGER;
```

assegna il tipo INTEGER alla variabile B.

Il Basic fa una piccola distinzione tra l'istruzione di assegnazione del valore di una variabile e il test di eguaglianza. LET A=B assegna il valore di B alla variabile A. IF A=B controlla il valore di A e B per vedere se sono uguali. Sfortunatamente il Basic è troppo tollerante, e LET è opzionale, così si perde la distinzione tra le due diverse azioni. Invece nel Pascal, i due punti determinano la distinzione. A:=B; assegna il valore di B alla variabile A. A=B senza i due punti è usato per il controllo dell'uguaglianza come in IF A=B THEN...

Il Basic e il Pascal usano una tecnica chiamata "passaggio dei parametri" i cui dettagli saranno spiegati più avanti. Per distinguere tra indici di matrice e parametri, il Pascal usa le parentesi quadre [...] per contenere gli indici di matrice come in

```
NUMERO[3,7]:=17;
```

Usa invece le parentesi tonde per le sue normali funzioni matematiche per superare la precedenza della moltiplicazione e della divisione sull'addizione e la sottrazione, e come delimitatori per i parametri. Nell'espressione:

```
A:=SIN (X+3)
```

X+3 è il parametro passato alla funzione SIN, il seno trigonometrico. In Basic i parametri sono passati solo alle funzioni. Nel Pascal sono usati più frequentemente.

Anche nel Pascal si usano commenti (*remarks*). L'uso della parola chiave REM in Basic, in fin dei conti è noioso, non solo per ragioni estetiche. Nel Pascal, un commento a sé stante può essere delimitato in due modi. Lo standard sono le parentesi graffe, ma poiché esse non sono disponibili in tutte le tastiere, possono essere sostituite da (* e *). Questo doppio delimitatore è meno conveniente e se è possibile, è meglio usare le parentesi graffe. Questi simboli, tuttavia, sono più distinguibili visualmente dal contenuto del programma e dal commento che delimitano, che la parola REM.

Nel Pascal, l'uso del punto e virgola è più complesso: basterà qui dire che è usato alla fine dell'istruzione, la quale potrà essere più lunga o più corta di una linea. Le istruzioni non hanno un limite di lunghezza di linea.

Dichiarazione delle variabili

Naturalmente è impossibile presentare in anticipo tutte le caratteristiche del Pascal. Ci saranno alcuni aspetti negli esempi che presenteremo che non saranno ancora stati spiegati. Non preoccupatevi di ciò. La parte dell'esempio che tratta dell'argomento attuale dovrebbe essere chiaramente comprensibile. Ecco alcuni esempi corrispondenti di dichiarazione di variabile:

Basic

```
DEFINT A,B (o A%, B%)
(nessuna)
DIM AR(4)
```

Pascal

```
A,B: INTEGER
NUM: REAL
AR: ARRAY [1..4] OF REAL
```

Abbiamo qui un primo esempio di incremento di flessibilità che determina anche un aumento di complessità. Nel Pascal l'intervallo di un indice di matrice può essere specificato. Nel Basic, la dimensione 3 indica i possibili valori di 0, 1, 2 e 3. (lo zero è usato solo in alcuni Basic). Nel Pascal si devono indicare il limite inferiore e il limite superiore. Anche AR:ARRAY [3..7,9..12] è valido in Pascal. Il Pascal è un buon mezzo per "modellare" il problema reale da risolvere. Supponiamo di dover scrivere un programma per una scuola superiore con un numero di classi da sei a nove. Se abbiamo

una matrice che contiene il numero di studenti per ogni classe, possiamo definirlo come STUDENTI:ARRAY [6..9] OF INTEGER;. L'indice indicherà la classe. Lo spazio per gli indici in questo caso minori di sei, non viene riservato e quindi non viene sprecato.

Variabili indice di matrici

Come nel Basic, anche nel Pascal è possibile usare identificatori di matrice di tipo INTEGER ed inoltre si possono usare identificatori di tipo CHAR (carattere). Supponiamo di dover contare la frequenza delle 26 lettere dell'alfabeto in un testo. Si può definire un vettore LETTERA:ARRAY ['A'..'Z'] OF INTEGER;. L'uso delle lettere A e Z comunica automaticamente al Pascal che gli identificatori devono essere di tipo CHAR. In ogni caso le lettere devono comunque essere contenute tra virgolette per distinguerle dai nomi di variabile. Il contenuto di vettore è definito di tipo INTEGER. Ci si potrebbe domandare quale valore possano avere queste caratteristiche. Questo ci porterebbe un po' lontani, ma comunque, ecco un programma in Basic per leggere un file di caratteri e contare quante volte compare una lettera (vedi listato 1).

Attenzione, questa è solo una parte del programma. Non potrà essere eseguita senza altre istruzioni che la sostengano: è intesa solamente per illustrare l'uso degli identificatori di tipo CHAR. Questo segmento di programma legge un file di testo precedentemente aperto e conta tutte le A, B etc... Si noti che la linea che si riferisce a EOF cerca la condizione *end of file*: il vostro Basic può implementare diversamente questa funzione. Nella linea 110, LEN(C$) restituisce la lunghezza della linea cioè il numero dei caratteri. La linea 120 usa la funzione MID$ per dare un carattere alla volta. La linea 130 verifica che il carattere si trovi nell'intervallo dalla A alla Z e quindi incrementa un contatore nel vettore in corrispondenza alla lettera. ASC(A$) dà il valore decimale del carattere ASCII. La lettera A ha valore 65. Sottraendo 64 da questo valore si ottiene quindi un numero da 1 (per A) a 26 (per Z); questo numero viene usato come indice per incrementare un contatore nel vettore LE in corrispondenza alla lettera.

In Pascal il programma sarebbe come nel listato 2.

Questo programma è stato di proposito semplificato per scopi illustrativi. Non si è ancora discusso di una serie di cose. Prima di tutto, la dichiarazione di variabile inizia con la parola VAR. L'istruzione IF è un confronto logico che equivale all'istruzione IF C$ ="A" AND C$ ="Z" del Basic. Il programma Pascal non ha GOTO e non sono necessari numeri di linea. C'è inoltre un ciclo REPEAT UNTIL. Esso inizia con REPEAT e finisce con UNTIL. La linea UNTIL dà le condizioni per la fine del ciclo. Se le condizioni non sono verificate, il programma riprende alla linea REPEAT. Il ciclo UNTIL REPEAT può essere "nidificato" come il ciclo FOR-NEXT nel Basic. Come nel Basic, il primo UNTIL termina il REPEAT più interno. Può essere utile, per migliorare la leggibilità del programma, far rientrare le linee tra l'inizio e la fine del ciclo.

Le stringhe nel Pascal

Finora non si è parlato molto delle stringhe nel Pascal. Le manipolazioni di stringa sono il punto forte del Basic e quello debole del Pascal. Il Pascal standard non ha le caratteristiche per maneggiare facilmente una stringa. Per definire una stringa nel Pascal si deve usare un vettore di caratteri, di cui dev'essere poi data la lunghezza: RESPONSE:ARRAY [1..10] OF CHAR;.

Molti implementatori hanno provato a migliorare questo punto debole del Pascal. La maggior parte delle implementazioni del Pascal "si estendono" in quest'area. Non esistendo però degli standard occorre consultare il manuale di ciascuna di esse.

Variabili booleane

Nel Pascal è possibile usare altri tipi di variabili oltre a quelle già discusse. Una di queste è il tipo BOOLEAN. Una variabile booleana può assumere uno dei due possibili valori logici, *vero* (*true*) o *falso* (*false*) (listato 3).

Listato 1.

```
90    DIM LE(26):
      REM IL VETTORE CONTIENE LA FREQUENZA DI OGNI LETTERA
100   READ C$: REM LEGGE UNA LINEA DAL FILE
110   FOR N=1 TO LEN(C$)
120   A$=MID$(C$,N,1): REM PRENDE I CARATTERI UNO ALLA VOLTA
130   IF A$>="A" AND A$<="Z" THEN LE(ASC(A$)-64)=LE(ASC(A$)-64)+
      1
140   NEXT N
150   IF NOT EOF THEN 100:
      REM LEGGE UN'ALTRA LINEA SE NON SI TROVA ALLA FINE DEL
      FILE
```

Listato 2.

```
VAR
   CARATTERE:CHAR;
   LETTERA:ARRAY['A'..'Z'] OF INTEGER;
BEGIN
   REPEAT
      READ(CARATTERE);
      IF CARATTERE IN ['A'..'Z'] THEN LETTERA[CARATTERE]:=
         LETTERA[CARATTERE]+1;
   UNTIL EOF(DATI);
END.
```

Listato 3.

```
VAR
   ALFA:BOOLEAN;
IF CARATTERE IN ['A'..'Z'] THEN ALFA:=TRUE;
IF ALFA THEN LETTERA[CARATTERE]:=LETTERA[CARATTERE]+1;
```

Si noti che non è necessario scrivere IF ALFA=TRUE.

Ciò accelera l'esecuzione di un programma, perché verificare il valore di una variabile booleana risulta più veloce che valutare ripetutamente una complessa espressione logica. Il tempo di esecuzione viene ridotto se la verifica viene fatta una volta, conservando il valore booleano e usandolo successivamente nel programma.

Variabili enumerate

Vediamo ora quelle che il Pascal chiama *variabili scalari*. Si tratta di variabili dichiarate dal programmatore. Per il Pascal, scalare significa enumerato o listato. La parola scalare non è usata nelle dichiarazioni TYPE. Tutte le variabili seguenti sono del tipo scalare. La parola TYPE viene usata per descrivere un tipo di variabile. INTEGER, REAL, CHAR sono tipi di variabili (listato 4).

L'elenco tra parentesi serve ad enumerare tutti i possibili valori che può assumere una variabile di questo tipo. Queste istruzioni definiscono i tipi di dati e non le variabili. Devono essere seguiti da una dichiarazione di variabile come GIORNO: GIORNI-DELLA-SETTIMANA;
per assegnare il tipo GIORNI-DELLA-SETTIMANA alla variabile GIORNO si può dichiarare qualsiasi numero di variabile per qualsiasi TYPE, sia standard che scalare. L'utente può usare come indici di matrice i tipi di dati definiti. TURNSIGNAL:ARRAY [LEFT..RIGHT] OF BOOLEAN; l'istruzione nel programma IF (TURNSIGNAL LEFT)AND (TURNSIGNAL RIGHT)THEN ERROR; può essere usata in un programma e risulterà significativa.

Tipi subrange

Per definire variabili di ogni standard o tipi scalari nel programma, si può usare un subrange di qualche tipo. Lo si è appena fatto con le dichiarazioni di matrice. NUMBERS: [1..7] OF INTEGER; definisce l'indice di vettori come un subrange del tipo INTEGER, il quale può prendere i valori da 1 a 7. GIORNIDELLASETTIMANA: [LUN..GIO]; è una dichiarazione di variabile valida se TYPE GIORNI-DELLASETTIMANA è stato dichiarato come sopra. In altre parole si può usare un subrange di ogni standard o tipi dichiarati dall'utente.

Forse così si inizia a capire qual è la filosofia di Wirth. Il fatto di permettere molti tipi di dati può sembrare all'inizio una seccatura. In realtà ciò, oltre a rendere il programma di più facile comprensione, rende anche più semplice la correzione. Il compilatore è in grado di trovare molti più errori di sintassi. Per esempio, se abbiamo assegnato l'istruzione TURNSIGNAL LEFT :=17, anche se non ci sono errori di sintassi, una variabile booleana non può avere valore 17, e perciò il compilatore otterrebbe un errore, indicante probabilmente un TYPE-MISMATCH.

Con il Pascal, si possono avere tipi di dati chiamati FILE (listato 5).

Si discuteranno i particolari quando parleremo di input/output.

Costanti

C'è un altro tipo di dati che non è variabile. Nel programma del Basic non ci sono meccanismi per le costanti. Si deve assegnare un valore ad una variabile e poi non toccarla. Il Pascal ha una sezione CONST (come la sezione VAR) nella quale si possono dichiarare valori di costanti (listato 6).

Si noti che il tipo di costante è compreso dal Pascal dalla dichiarazione. Ciò se viene usato un intero questo è di tipo INTEGER. Se viene usata una stringa letterale è una stringa costante etc.

Listato 4.

```
TYPE
   GIORNI-DELLA-SETTIMANA=(LUN,MAR,MER,GIO,VEN,SAB,DOM);
   PARTE=(SINISTRA,DESTRA);
   DIREZIONE=(NORD,EST,SUD,OVEST);
```

Listato 5.

```
DATI:FILE OF CHAR;
NUMERI:FILE OF REAL;
```

Listato 6.

```
CONST
   PI=3.14159265;
   N=17;
NOME='NUMERI PRIMI';
```

Listato 7.

```
1000  IF A=B THEN C=0: D=0: E=0: A$="FINE": PRINT
      "IL GIOCO E'
      FINITO"
```

Listato 8.

```
1000  IF A=B THEN C=0: D=0: E=0: A$="FINE"
1010  IF A=B THEN PRINT"IL GIOCO E' FINITO"
```

Listato 9.

```
1000  IF A<>B THEN 1100
1010  C=0: D=0: E=0: A$="FINE"
1020  PRINT"IL GIOCO E' FINITO"
1100  REM IL PROGRAMMA CONTINUA SE A<>B
```

Listato 10.

```
IF A=B THEN
BEGIN
   C=0:D=0;E=0;
   WRITELN('IL GIOCO E FINITO')
END;
```

Lunghezza dell'istruzione

Alcuni Basic permettono di continuare un'istruzione lunga sulla successiva riga o su righe usando caratteri come "/", END o con i terminatori di ciclo che saranno discussi più avanti. Anche se l'uso del ";" può apparentemente creare confusione, in realtà esso è di aiuto in quanto permette l'uso di istruzioni indefinitamente lunghe e l'organizzazione di un'istruzione su linee separate per maggior chiarezza. Si possono avere dei problemi con un IF-THEN nel Basic quando le condizioni vere devono risultare in un numero di azioni (vedi listato 7).

Ciò dà il messaggio ma solo in apparenza poiché la linea è troppo lunga. Si possono dividere le azioni multiple ripetendo il testo come nel listato 8.

Il Basic non funziona sempre allo stesso modo. A volte, come sopra dalla linea 1000 passa direttamente alla linea 1010 se A<>B. Cioè le istruzioni multiple sulla linea 1000 vengono eseguite solo se A=B. Alcuni Basic semplicemente non eseguono l'istruzione C=0 e continuano sulla linea 1000 eseguendo D=0, E=0 e A$="FINE" senza rispettare il testo. Per questi Basic funziona meglio l'istruzione riportata nel listato 9.

In altre parole questa limitazione del Basic vi obbliga ad usare una "logica diversa" e salta la condizione "vero" se trova un "falso".

Il Pascal adotta un metodo più semplice per affrontare questa situazione, che viene definita istruzione composta. È formata da una serie di istruzioni racchiuse tra un BEGIN e una END (vedi listato 10: la stringa nell'istruzione mostrata sopra è un parametro per la procedura WRITE linea).

```
END;
```

La END dell'istruzione composta segnala la fine dell'istruzione, in modo che non sia necessario un ";", anche se la maggior parte delle implementazioni accettano il ";".

Le stringhe letterali nel Pascal non sono un problema. Sono incluse nell'istruzione WRITE tra apici come nell'istruzione PRINT del Basic tra le virgolette. ■

Collisione per il VIC

Alberto Casale di Torino ha convertito "Collisione" per il VIC 20.
La grafica è ben riuscita e il programma particolarmente curato.
Attenzione: questa versione usa i "joystick" e non le "paddle".

Questa è una versione per il VIC 20 del gioco Collisione per l'Apple, apparso su *Personal Software* 3.

Il gioco è concepito per l'uso con il joystick, e consiste nel ripulire il percorso dai puntini verdi del valore di 5 punti, e dalle croci che valgono 10 punti ognuna.

Per evitare di scontrarsi con la macchina del VIC, vi sono quattro punti dove è possibile cambiare di una o due corsie. Tenendo premuto il tasto rosso sul joystick, la vostra velocità raddoppia; occorre però ricordarsi che, vista la particolare gestione del joystick da parte del computer, non è possibile accelerare e contemporaneamente cambiare corsia, fatta eccezione per gli spostamenti verso destra.

Una volta ripulito l'intero quadro, dopo un motivetto sonoro, ne comparirà un altro; a partire dal terzo e per tutti i successivi, la velocità della macchina gestita dal VIC raddoppierà, rendendo più difficile il gioco.

Per quanto riguarda le informazioni sullo schermo, nella prima riga in alto compaiono il vostro punteggio ed il massimo punteggio fino ad allora raggiunto, quest'ultimo, quando venga raggiunto, viene aggiornato contemporaneamente al vostro.

Nella seconda riga compare il numero del quadro che state "pulendo"; al centro dello schermo, infine, sono visualizzate le macchine che vi restano, compresa quella in gioco.

Quando viene dato il "run", il programma richiede la pressione del primo tasto di funzione (f1) per incominciare, se non viene premuto entro un certo tempo, il gioco inizierà automaticamente e, se non vi sono stati interventi da parte vostra, al primo scontro tornerà nella condizione del "run".

Il programma puro e semplice occupa poco meno di 2,5 K, tenendo presente però l'occupazione di memoria delle variabili ed il mezzo kilobyte rubato al BASIC per la definizione dei caratteri, restano liberi circa 200-300 byte per eventuali modifiche.

Principali routine

100 - 120 preparazione dei caratteri
130 - 140 presentazione
150 - 190 blocco principale di controllo ed aggiornamento schermo
200 - 400 gestione del movimento della vostra macchina
410 - 550 gestione del movimento della macchina del VIC
560 - 640 routine cui si accede dopo uno scontro
660 - 810 preparazione dello schermo ed inizializzazione variabili

Variabili usate

SC,HI vostro punteggio e punteggio massimo
J1 e J2 leggono lo stato del joystick
XP,YP,XA,YA coordinate vostre e del VIC
CP,CA,DP,DA cornice e direzione vostra e del VIC
PK,PEEK della locazione di schermo occupata dal VIC
CO è l'offset del colore
CR numero di macchine che vi restano
T e K usate nei loop
Z scopi vari quali la lettura dei DATA e di SGN(CP-CA)

```
100 POKE56,27:POKE55,250
110 FORT=0TO1:RESTORE:FORK=7168+T*16TO7183+T*16:READZ:POKEK,Z:NEXT:NEXT
120 POKE7188,254:POKE7192,0:POKE7199,8:FORT=7424TO7431:POKET,0:NEXT
130 POKE36879,8:PRINT"[illegible] COLLISIONE ":PRINT"[illegible]"TAB(6)"'F1' START"
:GOTO630
140 HI=PEEK(0)*256+PEEK(1)
150 PRINT"[illegible]"SCTAB(14)HI:POKE36875,220:FORT=1TO5:NEXT:POKE36875,0
160 GOSUB200:IFJ1=30THENGOSUB200
170 IFSC>HITHENHI=SC
180 GOSUB410:IFQU>2THENGOSUB410
190 GOTO150
200 POKEFNP(X),32:J1=PEEK(37151)AND63:J2=PEEK(37152):ONDP-1GOTO250,290,330
210 IFXP>9ANDXP<12ANDJ1=58ANDCP<4THENYP=YP-2:CP=CP+1:GOTO230
220 IFXP>9ANDXP<12ANDJ1=54ANDCP>1THENYP=YP+2:CP=CP-1
230 XP=XP+1:IFXP=22-2*CPTHENDP=2
240 GOTO360
250 IFYP<11ANDYP>8ANDJ1=46ANDCP<4THENXP=XP-2:CP=CP+1:GOTO270
260 IFYP<11ANDYP>8ANDJ2=119ANDCP>1THENXP=XP+2:CP=CP-1
270 YP=YP-1:IFYP=2*CP-1THENDP=3
280 GOTO360
290 IFXP<12ANDXP>9ANDJ1=54ANDCP<4THENYP=YP+2:CP=CP+1:GOTO310
300 IFXP<12ANDXP>9ANDJ1=58ANDCP>1THENYP=YP-2:CP=CP-1
310 XP=XP-1:IFXP=2*CP-1THENDP=4
320 GOTO360
330 IFYP>8ANDYP<11ANDJ2=119ANDCP<4THENXP=XP+2:CP=CP+1:GOTO350
340 IFYP>8ANDYP<11ANDJ1=46ANDCP>1THENXP=XP-2:CP=CP-1
```

```
350 YP=YP+1:IFYP=20-2*CPTHENDP=1
360 IFPEEK(FNP(X))>32THENSC=SC+5-(PEEK(FNP(X))=171)*5:PD=PD-1
370 POKEFNP(X),DP-1:POKEFNP(X)+CO,1
380 IFPD=0THENFORT=180TO240STEP5:POKE36876,T:FORK=0TO100:NEXT:NEXT:POKE36876,0:G
OTO670
390 IFFNP(X)=FNA(X)THENZ=0:GOTO560
400 RETURN
410 POKEFNA(X),PK:Z=SGN(CP-CA):ONDA-1GOTO450,480,510
420 IFXA=10ANDCA>0ANDCA<5THENYA=YA+Z*2:CA=CA+Z
430 XA=XA+1:IFXA=22-2*CATHENDA=4
440 GOTO530
450 IFYA=10ANDCA>0ANDCA<5THENXA=XA+Z*2:CA=CA+Z
460 YA=YA-1:IFYA=2*CA-1THENDA=1
470 GOTO530
480 IFXA=11ANDCA>0ANDCA<5THENYA=YA-Z*2:CA=CA+Z
490 XA=XA-1:IFXA=2*CA-1THENDA=2
500 GOTO530
510 IFYA=9ANDCA>0ANDCA<5THENXA=XA-Z*2:CA=CA+Z
520 YA=YA+1:IFYA=20-2*CATHENDA=3
530 PK=PEEK(FNA(X)):IFRND(1)<.08ANDPK=174THENPK=171
540 POKEFNA(X),DA-1:POKEFNA(X)+CO,5:IFFNP(X)=FNA(X)THENZ=1:GOTO560
550 RETURN
560 CR=CR-1:POKE36877,220:FORT=15TO0STEP-1:POKE36878,T:FORK=0TO150:NEXT:NEXT:POK
E36877,0
570 IFSC=175THENPOKE36869,240:SC=0:GOTO130
580 POKE7953+CR,32:POKEFNP(X),32:IFZ=0THENPOKEFNP(X),PK:POKEFNP(X)+CO,5
590 IFCR>0THEN800
600 PRINT"[graphics]";
610 FORT=1TO10:PRINTMID$("GAME  OVER",T,1);:FORZ=0TO150:NEXT:NEXT
620 PRINT:PRINTTAB(6)"[RVS]F1[OFF] STARTS"
630 FORT=0TO1500:IFPEEK(197)<>39THENNEXT
640 POKE0,INT(HI/256):POKE1,HI-(INT(HI/256)*256):CLR:CR=3
660 POKE36869,255
670 PRINT"[graphics]SC="SCTAB(11)"HI="HI:QU=QU+1:PRINT"[RVS]QU
680 PRINT"[RVS]┌──────────────────────┐.....................!";
690 PRINT"!.┌──────  ──────┐.!!.!.................!.!";
700 PRINT"!.!.┌─────  ─────┐.!.!!.!.!.............!.!.!";
710 PRINT"!.!.!.┌──  ──┐.!.!.!!.!.!.!.........!.!.!.!";
720 PRINT"!.!.!.!.┌────┐.!.!.!.!!.  .  .  .ICARS!.  .  .  .!";
730 PRINT"!.  .  .  .![graphics]  [graphics]!.  .  .  .!!.!.!.!.└─────┘.!.!.!.!";
740 PRINT"!.!.!.!.........!.!.!.!!.!.!.!.└──  ──┘.!.!.!";
750 PRINT"!.!.!.!.........!.!.!.!!.!.└──────  ──────┘.!.!";
760 PRINT"!.!.!.!.........!.!.!!.└──────────  ──────────┘.!";
770 PRINT"!.....................!└──────────────────────────┘";:FORT=1TOCR:POKE7952+T,0
:NEXT
780 DATA0,36,36,126,127,126,36,36,8,28,62,28,28,62,28,0
790 DEFFNP(XX)=V+XP+22*YP:DEFFNA(XX)=V+XA+22*YA:V=7724:CO=30720:PD=191
800 XP=1:YP=18:DP=1:CP=1
810 XA=1:YA=17:PK=PEEK(FNA(X)):DA=2:CA=1
820 POKE36878,15:POKE37154,127:GOTO140
```

Torre di Hanoi animata per il VIC

Il signor Massimo Quintini di Viterbo ha preso lo spunto dal programma non grafico relativo al gioco della Torre di Hanoi (PS3) *e ne ha realizzato due versioni per il VIC: una grafica e una in cui chi gioca è l'operatore. I programmi sono ben fatti e l'effetto è gradevole.*

Prendendo lo spunto dall'articolo di Mauro Boscarol "La torre di Hanoi", apparso su *Personal Software* 3 e partendo dal piccolo ma efficace programma relativo, mi è nata l'idea di animare il gioco fornendone però una doppia versione: nella prima l'operatore può, spostando i dischi, tentare di risolvere il gioco (ed in questo caso il VIC dirà alla fine se ha usato o no il minor numero di mosse), nella seconda si lascia fare tutto al computer. Per motivi di spazio (le 23 righe del VIC) ho dovuto limitare il numero massimo di dischi a 6, anche in considerazione del fatto che con 6 dischi occorrono 63 mosse e che un solo disco in più avrebbe portato il numèro minimo di mosse a ben 127. Anche con soli 6 dischi si ha un bel da fare per venirne a capo in sole 63 mosse, però, potendo scegliere il numero dei dischi con i quali cimentarsi, si può arrivare per gradi al numero massimo di 6, prendendo via via confidenza con la strategia del gioco.

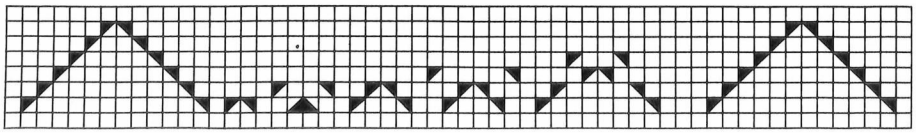
Come tutti i programmi compilati dal vivo e non sulla carta, anche questi avrebbero potuto essere semplificati e resi più leggibili, ma anche così funzionano perfettamente.

```
5 POKE 36879,8: PRINT"■"
6 DIM D$(7): DIM P(4): DIM PP(19)
7 GOSUB 3200
15 PRINT: PRINT: PRINT"NUMERO DISCHI (MAX.6)": INPUT N: NN=N
20 IF N>6 THEN 15
30 FOR S=0 TO3: P(S)=0: NEXT S: FOR S=0 TO 18: PP(S)=0:
   NEXT S: 00=1
40 GOSUB 210
50 I=1
60 J=3
70 GOSUB 100
80 GOTO 870
100 IF N=0 THEN RETURN
110 N=N-1
120 J=6-I-J
130 GOSUB 100
140 J=6-I-J
150 GOSUB 600
160 I=6-I-J
170 GOSUB 100
180 I=6-I-J
190 N=N+1
200 RETURN
210 REM--------------
220 REM* COSTRUZIONE*
240 REM*  PIRAMIDE  *
250 REM--------------
260 PRINT"■": RESTORE: FOR W=0 TO 6
270 READ S$
280 D$(W)=S$: NEXT W
290 DATA" "
300 DATA"      ▞▀▚"
310 DATA"     ▞▀▀▀▚"
320 DATA"    ▞▀▀▀▀▀▚"
330 DATA"   ▞▀▀▀▀▀▀▀▚"
340 DATA"  ▞▀▀▀▀▀▀▀▀▀▚"
350 DATA" ▞▀▀▀▀▀▀▀▀▀▀▀▚"
360 REM
370 REM **************
380 REM
400 QQ$="■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■"
410 P(1)=7: P(2)=14: P(3)=21
420 GOSUB 800
430 SP$="                              "
440 FOR T=N TO 1 STEP-1
450 Q$=MID$(QQ$,1,P(1)): PP(T)=T
460 PRINTQ$;"    ";D$(T)
470 P(1)=P(1)-1
480 NEXT
490 REM
500 REM **************
510 REM
530 P(1)=P(1)+1
540 FOR G=1 TO 1800: NEXT G
550 RETURN
560 REM---------------
570 REM*  MOVIMENTO *
580 REM*    DISCHI   *
590 REM---------------
600 Q$=MID$(QQ$,1,P(I))
620 PRINT Q$;"    ";SP$
630 HH=NN*I
640 IF PP(HH)<>0 AND PP(HH-1)=0 THEN MM=PP(HH): PP(HH)=0:
    GOTO 660
650 HH=HH-1: GOTO 640
660 P(I)=P(I)+1
670 P(J)=P(J)-1
680 Q$=MID$(QQ$,1,P(J))
690 PRINT Q$;"    ";D$(MM)
700 HH=NN*J
720 IF PP(HH)<>0 THEN HH=HH-1: GOTO 720
730 PP(HH)=MM
740 FOR ZZ=1 TO 1000: NEXT ZZ
750 RETURN
760 REM
770 REM**************
780 REM
800 LI$="──────────────────────"
810 FOR KK=1 TO 3
820 Q$=MID$(QQ$,1,P(KK))
830 PRINT Q$;KK;LI$: NEXT
840 P(1)=P(1)-1
850 RETURN
860 Q$=MID$(QQ$,1,P(I))
870 PRINT QQ$;"  NUMERO MOSSE =";2↑N-1: FOR S=1 TO 2500:
    NEXT: GOTO 7
3200 PRINT"■■┌──────────────────────┐"
3210 PRINT"■■│   TORRE DI HANOI     │"
3220 PRINT"■■└──────────────────────┘"
3230 FOR I=1 TO 2000: NEXT
3240 GOTO 15
```

Listato 1. *La Torre di Hanoi: chi gioca è il computer.*

Listato 2. *Con questo programma potete misurare la vostra abilità.*

```
5 POKE36879,8: PRINT"■"
10 DIM PP(19): DIM P(4): DIM S$(7)
20 GOTO 3200
890 REM------------
900 REM*  GIOCATORE *
910 REM------------
1240 RESTORE: FOR S=0 TO 3: P(S)=0: NEXT S: FOR S=0 TO 18:
PP(S)=0: NEXT S: OO=1
1250 INPUT"□NUMERO DISCHI(MAX.6)";N: NN=N
1255 IF N>6 THEN 1250
1260 PRINT"□": FOR W=0 TO 6
1270 READ S$
1280 D$(W)=S$: NEXT W
1290 DATA" "
1300 DATA"      ■1■"
1310 DATA"     ■■2■■"
1320 DATA"    ■■■3■■■"
1330 DATA"   ■■■■4■■■■"
1340 DATA"  ■■■■■5■■■■■"
1350 DATA" ■■■■■■6■■■■■■"
1400 QQ$="■■■■■■■■■■■■■■■■■■■■■■■■■■■■"
1410 P(1)=7: P(2)=14: P(3)=21
1420 GOSUB 1800
1430 SP$="                              "
1440 FOR T=N TO 1 STEP -1
1450 Q$=MID$(QQ$,1,P(1)): PP(T)=T
1460 PRINT Q$;"    ";D$(T)
1470 P(1)=P(1)-1
1480 NEXT
1490 REM
1500 REM **************
1530 P(1)=P(1)+1
1540 GOSUB 2000
1560 REM**************
1570 REM* MOVIMENTO
1580 REM*  DISCHI
1590 REM**************
1600 Q$=MID$(QQ$,1,P(I))
1630 HH=NN*I
1640 IF PP(HH)<>0 AND PP(HH-1)=0 THEN MM=PP(HH): PP(HH)=0:
GOTO 1655
1650 HH=HH-1: GOTO 1640
1655 GOSUB 2170
1660 P(I)=P(I)+1
1670 P(J)=P(J)-1
1680 Q$=MID$(QQ$,1,P(J))
1690 GOSUB 3040
1700 HH=NN*J
1720 IF PP(HH)<>0 THEN HH=HH-1: GOTO 1720
1730 PP(HH)=MM
1740 PRINT"■";OO: OO=OO+1
1745 GOSUB 4000
1750 GOTO 1540
1760 REM
1770 REM**************
1780 REM
1800 LI$="──────────"
1810 FOR KK=1 TO 3
1820 Q$=MID$(QQ$,1,P(KK))
1830 PRINTQ$;KK;LI$: NEXT
1840 P(1)=P(1)-1
1850 RETURN
1880 GOTO 1880
1890 REM
1900 REM----------
1910 REM
2000 PRINT QQ$;: INPUT"   MUOVI!     ";I,J
2010 IF I<3 OR J>3 THEN PRINT QQ$;"   MOSSA NON VALIDA  ":
FOR S=1 TO 1500: NEXT S:GOTO 2130
2020 L1=0: L2=0
2030 LI=N*I
2040 IF PP(LI)=0 THEN PRINT QQ$;"   MOSSA NON VALIDA  ":
FOR S=1 TO 1500: NEXT S: GOTO 2130
2050 IF PP(LI) <> 0 AND PP(LI-1)=0 THEN L1=PP(LI): GOTO 2070
2060 LI=LI-1: GOTO 2050
2070 LJ=N*J
2080 IF PP(LJ)=0 THEN 2120
2090 IF PP(LJ) <> 0 AND PP(LJ-1)=0 THEN L2=PP(LJ): GOTO 2110
2100 LJ=LJ-1: GOTO 2090
2110 IF L1>L2 THEN PRINT QQ$;"   MOSSA NON VALIDA":
FOR LK=0 TO 1500: NEXT LK: GOTO 2130
2120 PRINT QQ$;"                    ": RETURN
2130 PRINT QQ$;"                    ": GOTO 2000
2170 FOR TY=1 TO 5
2180 PRINT Q$;"    ";SP$
2190 FOR YY=0 TO 30: NEXT YY
2200 PRINT Q$;"    ";D$(MM)
2220 FOR YY=0 TO 30: NEXT YY
2240 NEXT TY
2250 PRINT Q$;"    ";SP$: RETURN
3040 FOR TY=1 TO 5
3050 PRINT Q$;"    ";D$(MM)
3055 FOR YY=0 TO 30: NEXT YY
3060 PRINT Q$;"    ";SP$
3070 FOR YY=0 TO 30: NEXT YY
3080 NEXT TY
3090 PRINT Q$;"    ";D$(MM): RETURN
3200 PRINT"□■┌──────────┐"
3210 PRINT"□■│  TORRE DI HANOI  │"
3212 PRINT"□■└──────────┘"
3214 FOR I=1 TO 1000: NEXT
3410 PRINT"□XXXXXXXXXXPREMERE F1"
3440 PRINT"XXNOTA-PER LO SPOSTAMEN-"
3450 PRINT"TO DEI DISCHI INDICARE"
3460 PRINT"LA BASE DI PARTENZA E"
3470 PRINT"QUELLA DI ARRIVO SEPA-"
3480 PRINT"RATE DA UNA VIRGOLA."
3490 PRINT"(ES.: 1,3)"
3500 GET R$: IF R$="■"THEN 890
3520 GOTO 3500
4000 XX=J*NN: CO=0
4020 FOR D=XX TO XX-NN+1 STEP-1
4030 IF PP(D)<>0 THEN CO=CO+1: GOTO 4040
4040 NEXT D
4050 IF CO=NN AND J<>1 THEN 4065
4060 RETURN
4065 IF (2↑NN-1)=OO-1 THEN FOR I=1 TO 900: NEXT I: GOTO 4130
4070 FOR GH=1 TO 2500: NEXT GH
4100 PRINT"□": PRINT: PRINT: PRINT"HAI FATTO■";OO-1;
"■MOSSE": PRINT: PRINT: PRINT
4110 PRINT"(IL NUMERO MINIMO DI  MOSSE, PER";NN;"DISCHI,  E↗■";
2↑NN-1;"■)"
4120 GOTO 4150
4130 PRINT"□": PRINT: PRINT: PRINT: PRINT
4140 PRINT"■■HAI USATO IL MINOR     NUMERO DI MOSSE!"
4150 PRINT: PRINT: PRINT: PRINT: PRINT: PRINT:
PRINT"PER  CONTINUARE  PREMI        ■F1■"
4160 GET HJ$:IF HJ$="■" THEN 3410
4170 GOTO 4160
```



PICCOLI ANNUNCI

Commodore VIC 20

Vendo cambio gioco per su ROM e il cartridge VIC-Graf per (quest'ultimo) L. 35 000 (prezzo di listino L. 112 100 IVA inclusa) causa doppio regalo. La VIC-Graf è completa di manuale e imballo; mai usata. Paolo D'Alessandro, via XXV Aprile 3, 88074 Crotone (CZ), tel. 0962-29417

Vendo per VIC 20 listato programma di gioco di mia invenzione a L. 2 800. Assicuro massima serietà. Specificare se si possiede expander e joystick. Paolo De Felice, via Jannelli 23, 80128 Napoli, tel. 371368

Vendo/cambio software scientifico e giochi per VIC 20. Richiedere elenco a Carlo de Notaristefani, via Schiavonia 1, 40121 Bologna, tel. 051-235812

Vendo programma Totocalcio per VIC 20 exp. 3 K. "Totdefa" riduce sistemi a corr. errore da integrali fino a 14864 col. output su video/tape, velocissimo (machine code scan), con spiegazioni L. 80 000. Vincenzo Lorenzale, via Marzabotto 14, 20094 Corsico (MI), tel. 02-4583226

Vendo o cambio splendide cassette con 20 programmi vari per VIC 20 non espanso a sole L. 20 000. Invio a richiesta listino giochi singoli. Marco Zanchi, via Benaco, 20139 Milano, tel. 02-538191

Dispongo di molti giochi in linguaggio macchina. Sono disponibile a scambi solo per giochi in linguaggio macchina. Marcello Cesi, via Magliana Nuova 178, 00146 Roma, tel. 06-5266009

Vendo programmi su fotocopia per VIC orientati ai giochi e alla grafica-matematica. Prezzo L. 2 000 al listato + L. 500 per spese postali. Massimo Gusso, v.le Felissent 32, 31020 Lancenigo (TV), tel. 0422-62969

Scambio/compro/vendo solo per posta molto software solo su cassetta per VIC 20 anche espanso 16 K. Massima onestà richiesta e garantita. Invio liste gratis in cambio delle vostre, oppure inviate L. 1 200 in francobolli. Giorgio Ferrario, via Adua 1, 21052 Busto Arsizio (VA)

Cambio programmi per il VIC 20. Sono in possesso di circa 300 programmi di tutti i generi. Inviatemi il vostro catalogo + L. 500 in bolli per risposta, risponderò a tutti. Roberto Oselladore, via Fausta 136A, 30010 Ca' Savio (VE), tel. 966923

Per VIC 20 vendo programma audiovisivo per l'apprendimento del codice Morse. Adatto per aspiranti radioamatori. Non occorre espansione. Telefonare ore pomeridiane feriali. Rocco De Micheli, via delle Industrie 32, 73042 Casarano (LE), tel. 0833-331234

Vendo per VIC 20 e Commodore 64 programma di gestione conto corrente e programma di gestione archivi di tracciato autodefinito, entrambi funzionanti sia su cassette che su floppy. Maurizio Galvini, via Montegeneroso 53, 20155 Milano, tel. 320073

Cambio/vendo programmi su cassetta, listati per VIC 20 a prezzi strepitosi!! Listino gratis a richiesta. Carlo Borreo, via G. Berio 34, 18100 Imperia, tel. 0183-21833

Programmi per VIC 20 oltre 60 giochi originali inglesi Amok, Alien, scacchi, Myriad, Defender, ecc. Chiedere lista, vendo/cambio. Massimo Fabrizi, via Isidoro di Carace 47, 00176 Roma, tel. 06-274138

Vendo per VIC 20 cartuccia giochi "Jelly Monsters" della Commodore (cartridge) a L. 37 000 ed interfaccia VCX 1001 per registratore completa di cavi di collegamento a L. 35 000 trattabili. Luca Trabalzini, via dei Termini 11, 53100 Siena, tel. 0577-289011

VIC users Grottaferrata prederebbe contatti con altri users club per scambio programmi. Massima serietà. Assicuriamo risposta a tutti. Vastissima biblioteca software. Cerchiamo anche soci. Gianluca Renna, via S. Andrea 25, 00046 Grottaferrata (Roma)

Scambio/vendo programmi e informazioni su VIC 20. Garantisco risposte a tutti. Inviatemi le vostre liste inizieremo gli scambi. Iolanda Sciutti, via F. Marconi 6, 00168 Roma

Vendo o scambio programmi per VIC 20. Inviare la vostra lista o telefonare. Inoltre vorrei entrare in contatto con VIC-utenti di Udine e dintorni. Jean Pierre Zaccomer, via Lumignacco 83, 33100 Udine, tel. 0432-34329

Vendo traduzione integrale delle istruzioni d'uso cartuccia VIC 1211 "Super espansione" (36 pagine) e cartuccia VIC 1212 "Sussidio al programmatore" (32 pagine) per L. 2 000 ciascuna. Luigi De Negri, via Puggia 22, 16131 Genova

Commodore PET/CBM

Vendo programmi scientifici e giochi per PET 30/4032 su cassetta o listati. Richiedere elenco a ing. Carlo de Notaristefani, via Schiavonia 1, 40121 Bologna, tel. 051-235812

Compro/scambio per PET/CBM 8032 programmi. Ho: RTTY, Gestione condominio, giochi; cerco: CW, giochi, varie. Inoltre scambierei idee con possessori di PET 8032 della mia zona. Antonio (i8YAV) Avagliano, via B. Avallone 103, 84013 Cava dei Tirreni (SA), tel. 089-842153, ore pasti.

Cerco possessori di CBM 64 per scambio idee e programmi. Telefonare dalle ore 22.30 alle 23. Riccardo Fassina, via L. Mangiagalli 5, 20133 Milano, tel. 02-299809

Sinclair ZX80-ZX81-Spectrum

Finalmente anche nella provincia di Venezia si è costituito per gli utenti italiani dello ZX81 un club per risolvere tutti i problemi di soft e di hardware. Con la quota di adesione di L. 15 000 si ha diritto a ricevere, oltre ad un eccezionale bollettino, software per un valore di circa 90 000 lire. Scrivere a Luca Crosara, via Roma 99, 30038 Spinea (VE) o telefonare ore pasti allo 041-994509

Cerco possessori di Spectrum zona Pescara-Chieti per scambio software e eventuale acquisto in comune di soft originale Psion. Egidio Morretti, via R.na Margherita 13, 65010 Carpineto Nora (Pescara), tel. 085-849130

Una banca di software per ZX80-81, istituita da accaniti sinclairisti, è a tua completa disposizione. Per accedervi inviaci un tuo listato inedito con L. 900 in francobolli: riceverai presto 3 programmi secondo le tue preferenze. Samuele Manfrin, via Priv. al Sole 27, 18038 Sanremo (IM)

Spectrum Sinclair, accurata traduzione dei manuali di istruzione vendo a L. 15 000; per lo ZX81 a L. 10 000; spedizione contrassegno. Vendo o scambio programmi per ambedue. Scrivere CP 2301 succ. 12, 34100 Trieste. Walter Radakovic, Galleria 11, 34124 Trieste

Programma PERT per ZX80-81 ottimizzazione progetti sino a 350 attività. Documentazione in italiano con teoria e esempi. Offro a L. 25 000 compresa spedizione. Scrivete per informazioni. Giovanni Servi, via Giovanni XXIII, 41012 Carpi (Modena)

Vendo/cambio programmi per ZX Spectrum. Maurizio Carola, via L. Lilio 109, 00143 Roma, tel. 5917363/5912655

Vendo/cambio programmi per ZX80-81 anche linguaggio macchina: Asterock, Labirinto, Poker da sala giochi. Applicativi: rappresentazione di oggetti in 3D, calcolo matriciale, ecc. Pietro Smedile, C.so Pisani 274, 90129 Palermo, tel. 091-595701

Vendo per ZX81 programma Totocalcio-due fisse + 11 doppie = 132 colonne. L. 10 000 a mezzo vaglia postale. Eventualmente scambio con altri programmi (inviare elenco). Alberto Polano, via D. Chiesa 14, 33038 San Daniele (UD)

Vendo programmi su cassetta e listati per Spectrum e ZX81 a prezzi modici. Giochi, applicazioni scientifiche e domestiche. Inviare francobolli per elenco. Carlo Celi, via Giorgetti 25, 32100 Belluno, tel. 27016

Scambio programmi per ZX81/80 nuova ROM. Cedo programmi per qualsiasi computer in cambio di programmi per ZX81. Vendo espansione 4 K completa di integrati a L. 40 000. Luca Pavan, via Mozart 22, 20021 Bollate (MI)

Programmi per Spectrum oltre 80 giochi originali inglesi ultime novità. Dispongo di programma originale inglese per duplicare programmi protetti. Vendo/cambio. Massimo Fabrizi, via Isidoro di Carace 47, 00176 Roma, tel. 06-274138

Scambio programmi per ZX80/81 16 K RAM. Eseguo montaggi di espansioni e interfacce. Luca Pavan, via Mozart 22, 20021 Bollate (MI)

Apple II

Scambio programmi di qualsiasi tipo per Apple II (16 K). Inviate la vostra lista e io vi risponderò con la mia. Bonifacio Saponaro, via Nicolò Dall'Aria 12, 70100 Bari, tel. 080-227221

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

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

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

Un annuncio sarà più efficace se seguirete queste indicazioni:

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

Software per Apple II. 250 giochi, word processor, utility varie, matematica, grafici (Graphic Magician), dischi di copia (Locksmith 3.1, 4.1, Copy II plus, Back it up) sprotetti. Roberto Pezzano, v.le Augusto 9, 80125 Napoli, tel. 081-619637

Ho 14 anni e vorrei scambiare i miei giochi per Apple II (scacchi, Defender, Risiko, Horizon, Thief e tanti altri fenomenali). Inviatemi la vostra lista, io invierò la mia. Paolo Evangelista, via Leone Leoni 6, 52100 Arezzo, tel. 22489

Cambio software di tutti i tipi per Apple II, vastissima scelta. Inviate la vostra lista a cui farà seguito la mia. Stefano Pagliano, via E. Ciccotti 8, 20161 Milano, tel. 02-6457911

Vendo programmi di calcolo strutture spaziali zone sismiche e di revisione prezzi per Apple II ed Apple IIe. Crescenzio Gallo, via Indipendenza 51, 71041 Carapelle (FG), tel. 0885-95242

Vendo programma scacchi per Apple II originale USA ad altissimo livello, possibilità di arrocco, cambio pedone all'ottava traversa, analisi partita, gioco alla cieca, ecc. 6 livelli, L. 500 000. Piercarlo Gennero-Fariello, via Pateri 15, 10042 Nichelino (TO), tel. 626135

Cambio/vendo programmi per Apple II: gestionali, utilità, giochi. Inviatemi la vostra lista, vi risponderò con la mia. Alessandra Casamassima, via Torre Menegotto 3/6, 16035 Rapallo, tel. 0185/58271

Compro, a scopo informativo, fotocopie manuali softplus, Apple Forth e Compare per Apple, anche separatamente. Contattare ore pasti o scrivere a Lamberto Brunelli, via Pisani Dossi 29, 20134 Milano, tel. 2157480

Scambio Apple II software di ogni tipo. Ho più di 150 programmi (in particolare giochi). Cerco in particolare programma Totocalcio. Assicuro l'invio del mio listato a chi mi contatta. Pietro Gozzi, via Traversetolo 95, 43100 Parma, tel. 0521-44688

Cambio programmi per Apple II. Giancarlo Negri, Galleria Europa scala D, 20081 Abbiategrasso (MI), tel. 02-9465214

Cerco manuale italiano Apple II Locksmith, 80 colonne, Tasc. Sono in possesso e desidero scambiare Ptero, Lastone, Contabilità semplificata o integrata, Legge 373, DOS Tool Kit con programmi Apple. Walter Franceschi, via Binel 4, 11020 Donnas (AO), tel. 0125-82374

Compro e cambio manuali di programmi Apple II. Franco Vandelli, via G.B. Morgagni 32, 20129 Milano, tel. 02-209231

Texas TI 99/4A

Vendo a prezzi irrisori software per TI 99/4A: giochi, matematica, musica, aiuto alla programmazione, ecc. Spedire lettera con francobolli per risposta per listino prezzi. Valerio Zupi, Traversa Bernardo Quaranta 9, 80146 Napoli

Vendo super-software per TI 99/4A, una maxicassetta con più di 20 programmi (L. 30 000) quasi totalmente inediti (fra cui il favoloso Labirinto 3D) ed un'altra cassetta con 10 lezioni di Basic TI (L. 15 000) in inglese. Paolo Benatti, via A. Doria 25, 37138 Verona, tel. 560930

Vendo software di vario genere già in mio possesso per TI 99/4A a prezzi trattabili. Possibilità di realizzo software su richiesta specifica. Paolo Latella, via Papa Benedetto XIII 15, 70124 Bari, tel. 514126

Scambio/vendo per TI 99/4A: Effemeridi, Poker, Vocaboli inglesi, Sviluppo colonne, Invaders e altri. Part time offro lezioni Basic di personal su TI 99/4A. Salvatore Sbacchis, via Don G. Minzoni 2/E, 90143 Palermo

Vendo/scambio software per TI 99/4A per strutture in cemento armato. Ho disponibili programmi per progetto di trave con sagomatura di ferri, tabella pilastri, trave continua anche per la TI 59. Giovanni Malato, via Hajech 2, 20129 Milano, tel. 741952

Vendo programmi di giochi in cassetta assolutamente inediti in Italia per il TI 99/4A. Vendo inoltre Basic per principianti e tutte le cassette della Texas che non avete ancora trovato. Marco Nicoletti, S. Polo 2765/B, 30100 Venezia, tel. 041-707661

Cambio/vendo software in Extended Basic per TI 99/4A. Mandare o richiedere lista. Fabrizio Ganem, via L. Zambarelli 32, 00152 Roma, tel. 5376138

Cerco programmi tecnici e gestionali per Texas TI 99/4A. Gradirei mi inviaste lista programmi e costi. Grazie. Roberto Cristiano, via Sebino 32, 00199 Roma, tel. 06-8453029

Vendo programmi di giochi vari e procedura di riduzione colonne Totocalcio. Programmi anche a richiesta su specifiche chiare. Maurizio Angeloni, via del Comune 27, 00049 Velletri (Roma)

Sharp

Vendo computer M280 Sharp 48 K RAM+Basic, Superbasic, assembler machine language, fotocopie manuali in italiano + stampante Sharp M280P3 slot M280, programmi su cassette perfetto stato, L. 2 000 000. Roberto Ongaro, via Caffaro 13, 00154 Roma, tel. 06-5134421.

TRS-80

Cambio software per TRS-80 mod. 1 (solo cassetta). Inviare lista ed indicare preferenze. Rispondo a tutti. Gradito inizio scambio corrispondenza per notizie ed esperienze. Massimo Politi, V.le G. D'Annunzio 31, 65100 Pescara, tel. 085-690786

CP/M

Cambio software APPLE CP/M. Franco Vandelli, via G.B. Morgagni 32, 20129 Milano, tel. 02/209231.

CP/M User's Group

Si è costituito a Perignano un CP/M User's Group, il cui scopo principale è quello di permettere lo scambio di idee e programmi tra tutti gli utenti di personal computer che utilizzano il CP/M come sistema operativo.

Per accedere alla banca programmi basta essere soci del club ed inviare un disco con i propri programmi per ricevere in cambio i programmi desiderati, scelti tra quelli disponibili al momento. Tutti i programmi presenti in catalogo sono attualmente disponibili in versione 8" singola densità, oppure dove possibile in listato sorgente; attualmente vi sono un centinaio di programmi disponibili che trattano di matematica, finanza, utility, data-base, giochi, linguaggi, ecc.

Inoltre il club è in contatto con analoghe organizzazioni estere per ampliare ancora la banca programmi.

Per ulteriori informazioni ed iscrizioni al club rivolgersi a:

Ceccotti Graziano, via Livornese Est 124, 56030 Perignano (PI), tel. 0587/616046

NOTA È possibile accedere alla banca programmi anche a coloro che possiedono un personal computer senza sistema operativo CP/M ed hanno il registratore come memoria di massa (Apple, Pet, Vic, ZX, Atari, ecc.).

Il tal caso lo scambio dei programmi avviene tramite il listing su carta.

Nel catalogo del Club vi è già del software molto interessante tra cui:
CBASIC - BASIC80 - MBASIC - COBOL80 - FORTRAN - PASCAL - LISP. ASM - BASICO - AUTOPROG - COMPILATORE BASIC80 - SUPERSORT - DATA BASE A CAMPI LIBERI - SELECTOR III - WORD STAR - TEX - GESTIONE DISTINTA BASE - CONTABILITÀ - GENERATORE DI PROGRAMMI IN CBASIC

Triumph Adler

Cerco programmi vari per Alphatronic P2, sia in Basic che in CP/M da permutare con programmi in mio possesso. Telefonare ore serali. Amedeo Di Salvatore, via Dante Alighieri 4, 03100 Frosinone, tel. 0775-857479

Shine

Compro listati per Shine. Sono molto simili ai PET, oppure listati che girino su tutti i personal. Cerco possessori dello Shine per scambio esperienze e software. Telefonare la sera dopo i pasti. Emanuele Abbate, via Garibaldi 129, 97016 Pozzallo (RG), tel. 0932-955341

Nuova Elettronica

Ristretto gruppo di hobbysti del computer Z80 di N.E. ubicato a Bologna cerca futuri soci di tutta Italia scopo scambio programmi giochi, piccole gestioni, scientifici. Luciano Sarego, via Della Pace 168, 40010 Sala Bolognese (BO), tel. 051-451893

Olivetti

Cerco utilizzatori Olivetti M20 per scambio software, idee ed informazioni. Giovanni Marzaro, via Seraglio SX 27, 30031 Dolo (VE), tel. 041-411531

Libri e riviste

Compro numeri arretrati riviste USA/GB. Preferibilmente 80 Micro e Creative Computing. Cerco libri interessanti per TRS-80. Massimo Politi, V.le G. D'Annunzio 31, 65100 Pescara, tel. 085-690786

Vendo: 32 programmi con il PET, Il Nanobook Z80, vol. 1, i primi 5 volumi di N.E., annate di Radio Kit, annate di N.E. Radiorivista dal 1980 ad oggi, The radioamateur handbook 1982, Bit dal n. 1 ad oggi, dal n. 1 al 7 di M.C. Sergio Daraghin, via Bengasi 33/B, 10042 Nichelino (TO), tel. 011-6272087

Diversi

Cerco disperatamente o quasi il listato del gioco "Star Trek" per l'Acorn Atom con consigli per la conversione per il VIC 20. Si trova nella cassetta "Game 4". Grazie. Emilio Desalvo, via Molavecchia, 00061 Anguillara Sabazia (Roma)

Vendo metodo anti-list per Apple, PET/CBM, VIC 20 a L. 10 000. Inoltre vendo o cambio sofrware per suddetti calcolatori. Cerco possessori Shine per scambio idee. Telefonare o scrivere per la lista allegando L. 500. Paolo Marcato, via C. Battisti 3, 35027 Noventa Padovana (PD), tel. 049-502475

Cambio/vendo programmi di utilità, giochi, ecc. su listato per Apple II, PET/CBM, Atari 400/800, VIC 20. Richiedere la lista allegando L. 500 o telefonando a Paolo Marcato, via C. Battisti 3, 35027 Noventa Padovana (PD), tel. 049-502475

Club e gruppi

Cerco utenti di personal computer di qualsiasi tipo nella zona di Bari per creare, anche a livello dilettantistico, un gruppo di studio sulle differenze esistenti nell'utilizzazione e nei linguaggi dei personal. Maurizio Coccorese, via Gentile 108/D, 70126 Bari, tel. 080-491374.

Ci sono mille buone ragioni per comprare il personal computer HP 86.

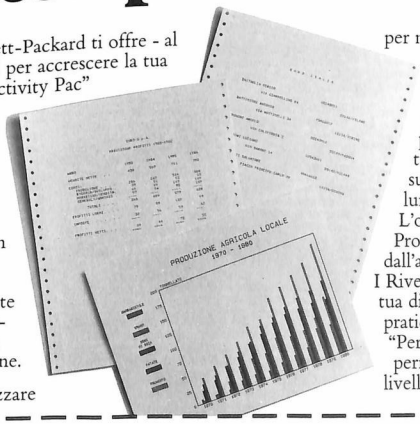
Con tutte queste soluzioni non c'è più spazio per i problemi.

- Soluzioni per gestire tabelle elettroniche
- Soluzioni per lettere, rapporti e memorandum
- Soluzioni per gestire informazioni e banche dati
- Soluzioni per le rappresentazioni grafiche
- Soluzioni per la trasmissione dei dati.

E se queste non bastano, l'HP 86 ti offre anche soluzioni per ingegneria, meccanica, statistica, gestione, finanza ed altre ancora. Quest'ampia scelta di soluzioni e la possibilità di configurare modularmente l'HP 86, ti consentono di avere un sistema in grado di espandersi quando aumentano le tue necessità.

Piú tre ottime ragioni per comprarlo ora.

Fino al 15 luglio 1983 la Hewlett-Packard ti offre - al prezzo di uno - tre *package* fatti per accrescere la tua produttività: il "Personal Productivity Pac" include i *package* VisiCalc® Plus, File/80 e Graphics Presentations.



VisiCalc® Plus è un potente strumento di analisi che ti permette di creare fogli di lavoro e tabelle elettroniche. Cambi un dato, e tutta la tabella viene automaticamente aggiornata. È una risposta meravigliosamente semplice a tutti i "Che cosa succede se..." che incontri nella tua attività di analisi e di pianificazione.

File/80 ti consente di memorizzare e ritrovare rapidamente le tue informazioni, di aggiungere, modificare o cancellare dati e di gestire facilmente i tuoi archivi: il tutto elettronicamente, senza bisogno di schede e schedari.

Graphics Presentations ti consente di produrre, per mezzo di un *plotter*, diagrammi circolari o lineari, istogrammi e pagine di testo multicolori di qualità altamente professionale. E per le tue presentazioni puoi realizzare tutto questo anche direttamente su trasparenti per lavagna luminosa.

L'offerta del "Personal Productivity Pac" è indipendente dall'acquisto del computer.

I Rivenditori Autorizzati HP sono a tua disposizione per dimostrarti praticamente come l'HP 86 e il "Personal Productivity Pac" ti permettano di raggiungere nuovi livelli di produttività nel tuo lavoro.

Per ricevere ulteriori informazioni e il nome del Rivenditore più vicino, telefona allo 02-92369468 o spedisci il coupon alla Hewlett-Packard Italiana C.P. 10190 - 20100 Milano.

Desidero sapere tutto sull'HP86 e il "Personal Productivity Pac".

Nome e Cognome _____

Incarico _____

Società _____

Indirizzo _____ CAP _____

Città _____ Tel. _____

PERSOFT/86 TR

VisiCalc® è un marchio registrato della VisiCorp.

Quando sono i risultati che contano HEWLETT PACKARD

Hardware

Vendo computer schacchistico CHASS CHAMPION POCKET CHESS. 8 livelli di difficoltà, possibilità di impostare problemi o variazioni durante la partita. Usato pochissimo cedo a L. 150 000. Luca Trabalzini, via dei Termini 11, 53100 Siena. tel. 0577/289011

Vendo Apple II e plus 48 K usato pochissimo+manuali+programmi vari. Cambio/vendo programmi per Apple in ambiente tecnico. Telefonare ore pasti. Piero Ceriolo. via Verdi 26bis. 18100 Imperia, tel. 0183-63529

Vendo Commodore VIC 20 + cartridge 16 K RAM + joystick + manuale in italiano (valore complessivo L. 900 000) usato pochi mesi, tutto in ottimo stato a L. 620 000. Regalo inoltre diversi programmi. Andrea Pacelli, via Firenze 205. 65100 Pescara. tel. 085-26380

Vendo ZX81 + alimentatore + cavi + espansione 32 K a L. 400 000 ottime condizioni + regalo cassetta centipede pac-man. Vendo ZX81 completo + espansione 16 K L. 330 000 mai usati (ottimo affare) + cassette (10). Giampaolo Gentili, via Turati 10, 10024 Moncalieri (TO). tel. 011-6407195

Texas TI 99/4A home computer più interfaccia per registratore a cassette. Usato pochissimo ancora in garanzia a causa passaggio ad altro sistema a L. 450 000 trattabili. Telefonare ore pasti. Livio Bucceri, Res. Spiga Milano 2. 20090 Segrate (MI), tel. 02-2130829

Per passaggio a sistema superiore vendo per Apple II Plus: n. 1 interfaccia parallela L. 75 000, n. 1 scheda espansione 16 K RAM L. 120 000, n. 1 scheda 80 car. compatibile 8" L. 125 000 come nuovo. Isabella Bottini. via G. Galilei 681, 18038 Sanremo (IM)

Vendo due personal Kiber 64 K RAM monitor floppy 5" doppia faccia, doppia densità con ampia biblioteca software (CP/M, Basic, Fortran, Data base, Wordstar, ecc.) a prezzo strepitoso. Giuseppe Carnevali, via Coppino 433, 55039 Viareggio, tel. 0584-394059

Vendo Sharp PC1211 completa di manuali con interfaccia per cassette CE-121 (tutto in perfetto stato) L. 250 000. Telefonare ore serali. Alberto Marletta, via C. Fornara 3, 20147 Milano, tel. 4079408

Vendo computer Sinclair ZX80 come nuovo completo di cavi collegamento, registratore e TV alimentatore, manuale originale inglese e italiano, il tutto al prezzo di L. 90 000. Mauro Aredi, via Turati 2, 20093 Cologno Monzese (MI), tel. 2543770

Vendo o scambio Sinclair ZX81 + alimentatore + vari cavetti + libro istruzioni e giochi perfettamente funzionante prezzo affare o scambio per VIC 20 in buone condizioni. Alberto Marchionni, via Pastore 5, 29100 Piacenza, tel. 0523-66123

Vendo HP 41C + tre espansioni + lettore schede + stampante termica + software di matematica applicata (un anno e mezzo di vita) a prezzo da definire. Andrea Arnone, via Masaccio 58, 50132 Firenze, tel. 243164

Vendo Acorn Atom dicembre '82 + alimentatore + 16 K ROM + 12 K RAM + manuali inglese ed italiano. Tutto garantito e funzionante. Inoltre 2 cassette di games originali Atom. Occasione il tutto a L. 650 000 (valore originale L. 814 165). Agostino Bova, via F. Lenci 22, 55049 Viareggio, tel. 0584-92435

Vendo ZX81 + 16 K + alimentatore per stampante + gioco scacchi (16 K) + giochi vari (1 K) + manuale italiano. Garanzia da spedire, tutto a L. 280 000. Alberto Abbadini, via G. Pascoli 6, 25013 Carpenedolo (BS), tel. 030-969310

Vendo ZX81, 32 K RAM, completo di cavi e alimentatore, manuali in italiano e inglese, 2 cassette piene di programmi, tutto montato in fabbrica e poco usato a sole L. 400 000. Alberto Betti, via Laghetto 1, 22050 Imbersago (CO), tel. 039/511563

Vendo, causa partenza militare, ZX81 + 16 K di espansione + 2 manuali + 66 programmi per ZX + cassetta con giochi + stampante per L. 500 000 non trattabili. Stefano De Venuto, via Baveno 25, 10146 Torino, tel. 797088

Vendo ZX81 + 3 manuali + 16 K RAM + tastiera speciale + 101 programmi di cui 39 in linguaggio macchina a L. 400 000 intr. (più 40 listati Basic). Marco Carrubba. via M. Campionesi 29. 20135 Milano, tel. 585294

Vendo pocket computer Sharp PC1500 ottime condizioni (ott. '82) L. 500 000 trattabili o cambio con ZX Spectrum. Inoltre all'eventuale acquirente regalo alcuni listati di giochi e 4 pile. Davide Gessi, via per Volano 49, 44020 Volano (FE), tel. 0533-85188 escluse ore ufficio

Vendo ZX80 con ROM 8 K + 16 K RAM completo di alimentatore, manuali, cavi + 3 cassette programmi + vecchia ROM, tutto a L. 300 000 Giulio Brandaschia, via Lando Landucci 25, 35100 Padova, tel. 755167

Vendo ZX81 con espansione 16 K, alimentatore, cavetti, manuale in italiano e cassette. Telefonare ore serali. Marco Marassi, via Plutarco 12, 20145 Milano, tel. 436982

Vendo ZX81 + espansione 32 K + inverse video - bip per tastiera + manuale inglese/italiano + nanobook Z80 + alimentatore + cassetta programmi + 66 programmi per lo ZX81. Tutto al prezzo sbalorditivo di L. 300 000. Sergio Santoro, via del Gallitello 13, 85100 Potenza

Mother board per VIC 20, 4 slot vendonsi L. 60 000: tasto di reset, interruttori selezione schede, stampato e connettori professionali, alimentata dal VIC (fino 32 K). Per accordi inviare francobollo. Fabrizio Lombardini, via Aloisi 35 (Montenero), 57100 Livorno, tel. 0586-579507

Vendo Sinclair ZX80 8 K ROM + 4 K ROM + relativi manuali + espansione 32 K RAM + relativo alimentatore + 1 cassetta programmi al migliore offerente. Prezzo base L. 300 000. Ideale per chi è alle prime armi. Emilio Triunfo, via Cumana 9, 80126 Napoli, tel. 081-633274

Vendo VIC 20, espansione alta risoluzione, int. registratore, libri, programmi, il tutto nuovissimo a L. 550 000. Gianpietro Remondi, via Don Pelucchi 8, 24021 Albino (BG)

Vendo VIC 20 + unità cassette C2N + espansione 16 K RAM tutto in perfette condizioni a L. 650 000 trattabili. Telefonare ore pasti. Gianmaria Gregori, via Volturno 9, 27100 Pavia, tel. 36630

Vendo VIC 20 + manuale + espansione 16 K + Programmer's Aid + Trislot + riviste + software. Vita: 3 mesi. Occasione L. 650 000. Massima serietà. Telefonare solo se interessati ore 8-13 per accordi. Alessandro Tuveri, via Zorutti 49, 33053 Latisana (UD), tel. 0432-208761

Vendo HP 9845B video grafico, stampante termica + software di ingegneria civile. Fabio Cesaroni, via Camilluccia 589/C, 00135 Roma, tel. 3288308

Vendo ZX81, espansione 16 K RAM, 4 cassette giochi, manuale in italiano. Andrea, Torino, tel. 011-539061

Vendo in blocco ZX80 + 8 K ROM + 16 K RAM + slow + alimentatore + manuale + programmi + cassetta scacchi, in ottime condizioni, a sole L. 300 000. Giuseppe Perona, via Farina 102, 10086 Rivarolo (TO)

Per VIC 20 vendo o cambio oscilloscopio 20 MHz 2 tracce una Homg 4001 con: espansione 40/80 colonne + R9232 (FUL) per VIC 20. Telefonare ore 9-12.30/ 15-19. Luigi Quartieri, via Camaldoli 45, 50054 S. Pierino Furecchio, tel. 0571-20187

Vendo Texas TI 99/4A risoluzione video 192 × 256, suono 5 ottave, 3 tonalità, 16 colori, memoria RAM 16 K, nuovo e mai usato, solo L. 500.000. Stefano Grandesso, Giudecca 173, 30123 Venzia, tel. 041-709079

Vendo Sinclair ZX80, 8 K ROM, manuali in italiano, alimentatore originale, molto software a meno di L. 200 000. Massimiliano Bertini, via Monte Cervino 9, 20052 Monza (MI), tel. 749648

Vendo Commodore 64 con 64 K RAM e suo registratore C2N a L. 950 000. Vendo anche calcolatrice TI59 a L. 150 000 e una TI58C a L. 80 000. Giovanni Cominotto, via Tassini 8, 30173 Mestre (VE), tel. 041-59065

Vendo ZX81, ZX printer, alimentatore, 4 rotoli carta, cavi, manuali, programmi a L. 295 000. Telefonare sabato o domenica pomeriggio. Germana Palmisano, via Vernilli Ciommi 36/A, 80047 S. Giuseppe Ves.no (NA), tel. 081-8281981

Vendo ZX81 + ZX beeper + manuali + cavetti + alimentatore + libro 66 programmi + programmi con imballaggio + 16 K a L. 350 000 + cassette: Tiranno-sauro, Asteroidi, Startrek, Flight Simulation Hires e altri a L. 10 000 l'una. Massimo Rossi Espagnet, via Balduina 114, 00136 Roma, tel. 06-3490023

Vendo utilissimo joystick per ZX81. Potrete "videogiocare" tranquillamente seduti. Allego chiarissime istruzioni e un programma regalo. Tutto eccezionalmente a sole L. 20 000. Massima serietà. Paolo Michetti, via Leone XIII 58, 55043 Lido di Camaiore (LU), tel. 0584-64301

Cerco schemi elettrici per VIC 20 e sue espansioni. Acquisto espansione per alta risoluzione e mother board anche autocostruita o schema della stessa. Umberto Santarlasci, via CR. 44 n. 45, 90100 Palermo

Vendo HP97 con custodia, alimentatore, manuali, carta stampante e schede programmi, L. 500 000 non trattabili. Ore ufficio. Alex Martelli, via Tiberio Imperatore 45, 00145 Roma, tel. 5140606

Vendo/cambio stampante Silentype per Apple II completa di interfaccia (stampa su 80 colonne) a sole L. 550 000. Giorgio Greco, via Garibaldi 53, 97015 Modica (RG), tel. 941168 (martedì e giovedì)

Occasione vendo computer ZX80 Sinclair, causa passaggio a sistema superiore, perfettamente funzionante a L. 95 000 con imballo originale di febbraio '83. Gianlucio Marchetti, via Saloni 53, 30015 Chioggia (VE), tel. 405221

Vendo Apple II 48 K completo di disk drive e monitor Philips, ottime condizioni, ancora in garanzia. Prezzo L. 2 800 000. Eventualmente anche stampante Epson, prezzo a convenire. Il sistema è autonomo e non necessita di altro per un funzionamento a dischi in configurazione minima. Qualsiasi prova. Franco Vandelli, via G.B. Morgagni 32, 20129 Milano, tel. 02/209231

Vendo ZX81 + 16 K + manuale, alimentatore e accessori (usato un mese) per passaggio a sistema superiore a L. 350 000 non trattabili. Telefonare ore 19-20, sabato 9-19. Roberto Garatti, v.le dei Mille 70, 20129 Milano, tel. 02-7382424

Vendo ZX81 + 32 K RAM, alimentatore, cavi, manuali inglesi e italiano, listati programmi, pochi mesi in ottime condizioni, prezzo trattabile. Telefonare 055-474836 ore pasti, 265033 ore ufficio Umberto Torrini, via Bolognese, 57, 50139 Firenze

Vendo VIC 20 usato pochissimo + interfaccia registratore + cassetta software + 2 schede gioco (Radar, Ratrace, Super Lander). Tutto a L. 500 000. Telefonare ore pomeridiane. Mauro Francioni, via Giulio Branca 7, 20147 Milano, tel. 4073751

Occasione DAI 48 K video 64 colonne maiuscole minuscole, grafica alta risoluzione (336×256), 16 colori, sintetizzatore vocale e musicale stereofonico, interfacce per drive, stampante, due registratori cassette, TV color, due paddles tridimensionali, cavi, manuali, programmi giochi, L. 1 300 000. Giulio Cesare Giacobbe, via Finocchiara 46, 16144 Genova, tel. 010-825537

Vendo Sinclair ZX80 nuova ROM 8 K + alimentatore Sinclair + cavi a L. 140 000 a chi lo acquista regalo listati giochi e libro "Come programmare con ZX81/ 80". Vendo inoltre memoria RAM 64 K originale inglese per ZX81 a L. 140 000. Compro programmi per PET/CBM 8 K RAM. Raffaele De Sio, via Settimo Mobilio 17, 84100 Salerno, tel. 089-234828 ore pasti.

Vendo Apple II 48 K più scheda Apple language, Pascal, Fortran, DOS 3.3, data base L. 2000000. Tel. ore 21-22. Pasquale Leone, via G. Santacroce 25, 80129 Napoli, tel. 081-379092

Computer Amico "2000" CPU+interfaccia video+Basic 8 K+alimentatore di potenza+box metallico per altre espansioni+tastiera ASCII per passaggio sistema superiore ceido a L. 800000 trattabili. Antonino De Lorenzo, via Dei Gigli 4/2, 17020 Laigueglia (SV), tel. 0182-499228

VIC 20 causa doppio regalo vendo completo alimentatore ed interfaccia video a L. 520000. Alessandro Taiocchi, via P.pe Eugenio 30, 20155 Milano, tel. 02-3188669

Vendo o cambio Atari 800 + 410P joystick + paddle + star raider + 48 K musik composer L. 2000000 o cambio con Apple 48 K con modulatore TV tutto con tre mesi di scarso utilizzo. Maurizio Battelli, via Roma 100, 61013 Mercatino Conca (PS), tel. 0541-970148

Vendo stampante per HP41C/CV, perfetta, 1 mese di vita, a prezzo davvero interessante. Telefonare ore serali. Amedeo Fasano, via Res. Sagittario, 20090 Segrate (MI), tel. 02-2130331

IL BASIC

PROGRAMMI PRATICI IN BASIC
di Lon POOLE

Il libro è una raccolta di programmi di tipo finanziario, matematico, scientifico e di decisioni manageriali. Ogni programma, orientato alla risoluzione di un problema pratico, è presentato con una breve descrizione iniziale, un campione di esecuzione, il listing BASIC, nonchè, per molti, una sezione in cui sono raccolte possibili variazioni per rendere il programma stesso più rispondente alle necessità personali. I programmi sono stati scritti in un BASIC generale, il che li rende, per la maggior parte, direttamente utilizzabili, senza alcun cambiamento, su molti microcomputer, e sono stati provati usando varie versioni di BASIC.

SOMMARIO

Reddito medio - Valore corrente di un buono del tesoro - Calcolo dell'interesse di obbligazioni - Interesse continuo composto - Regola dell'interesse 78 - Valore netto presente di un investimento - Flusso di cassa non uniforme - Affitto/decisione di acquisto - Analisi degli investimenti sindacali - Scambio di deprezzamento - Ripartizione di quote - Quota interna di ritorno - Amministrazione finanziaria - Analisi di quote di stato finanziario - Partecipazione ai profitti dei contribuenti - Controllo dei libri - Bilancio di casa - Metodo critico Path (CPM) - Pert - Algoritmo di trasporto - Teoria delle code - Analisi di Markov - Analisi non lineare di Breakeven - Analisi con la metrice dei vantaggi - Decisione di Bayes - Quantità economica di un ordine - Quantità economica di una produzione - Teoria della stima statistica.

cod. 550D pag. 200 L. 12.500

INTRODUZIONE AL BASIC

Si tratta di un vero e proprio corso di BASIC. Le caratteristiche che lo hanno fatto scegliere, per questi mini elaboratori sono di essere facile da apprendere ed utilizzare, nonchè di essere un linguaggio interattivo. Se ci sono errori, questi possono subito essere rilevati in maniera tale da poterli correggere.
Facile da leggere e imparare, che con numerosi esempi "testa" subito il reale apprendimento raggiunto dal lettore.
Un testo che si rivolge ai principianti. Infatti in maniera progressiva e pedagogica, senza alcuna necessità di formazione di base sulle tecniche di informatica, illustra, spiega, esemplifica tutti gli aspetti dei linguaggi attualmente disponibili su differenti sistemi, che vanno dal microcalcolatore ai sistemi time-sharing chi ha già acquisito esperienza in altri linguaggi, invece potrà saltare la parte preliminare, di introduzione alla materia, per entrare subito nel vivo del BASIC.
La base dell'informatica; le generalità del linguaggio BASIC; le istruzioni: il trattamento degli elenchi; tabelle, file, sottoprogrammi; i procedimenti grafici e le possibilità offerte; le istruzioni specifiche di alcuni sistemi.

cod. 502A pag. 314 L. 21.000

PROGRAMMARE IN BASIC
di Michel PLOUIN

Come tutte "le lingue viventi", il BASIC viene applicato in realtà a questa o a quella macchina sotto forma di dialetti più o meno particolari. Questo libro si sforza di descrivere in modo metodico il BASIC delle tre macchine più diffuse sul mercato mondiale: Apple, PET, TRS 80, e, naturalmente, i loro derivati.
Ciò faciliterà anche la conversione di programmi scritti, da un determinato personal computer agli altri. Numerosi esempi (programmi verificati attentamente) chiariscono i concetti proposti e sono immediatamente riutilizzabili da i possessori dei sopracitati personal.

SOMMARIO

Introduzione - Le variabili - Funzioni - Logica di svolgimento di un programma - Dialogo con la macchina - Funzioni speciali - Effetti grafici ed altri - Preparazione dei programmi codice ASCII e caratteri speciali - Calcolo binario ed esadecimale - Esempi di programmi.

cod. 513A pag. 94 L. 8.000

COME PROGRAMMARE
di Jean Claude BARBANCE

Il libro insegna a chi programma come deve enunciare e definire correttamente l'idea iniziale, come analizzarla e trasformarla, e come verificare la correttezza della stessa sino a giungere alla stesura di un programma ben documentato, leggibile e facilmente modificabile. Vengono esplicitate tutte le altre fasi intermedie del lavoro: le vie alternative che si presentano e tra cui scegliere, le eventuali estensioni, le prove e le verifiche che occorre fare per ottenere un programma conforme a quanto ci si era proposti. Poichè era necessario appoggiarsi a un linguaggio, si è scelto il BASIC per la sua larga diffusione. I concetti esposti, comunque sono utilizzabili con qualsiasi altro linguaggio. I programmi presentati sono stati tutti provati e girano su computer da 4 a 64K di memoria.

SOMMARIO

Realizzazione dei programmi: le fasi - La definizione degli obiettivi - L'analisi - La codifica e la messa a punto del programma - Presentazione degli esempi - Rappresentazione di un numero decimale mediante una stringa di caratteri altabetici - Il gioco del 421 - La contabilità personale.

cod. 511A pag. 192 L. 12.000

GRUPPO EDITORIALE JACKSON
Divisione Libri

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

Vendo schede computer nuova elettronica: LX384, LX383, LX388, LX385. Telefonare ore pasti. Marco Galliani, via Boggiani 2, 28100 Novara, tel. 453614

Compro personal computer VIC 20 Commodore completo di registratore espansione 16K manuale in italiano. Cerco possibilmente insieme software piccoli archivi giochi utility. Scrivere o telefonare (14-20). Piero Di Giovanni, via Chieti 15, 65100 Pescara, tel. 085-389122

Vendo micro VIC 20 espanso + giochi su cartridge e su cassetta+manuali+modulatore TV+alimentatore nuovi a L. 480000. Telefonare ore 19.30. Bells, via Zanotti 21, 40137 Bologna, tel. 051-303098

Vendo CBM 4032+floppy singolo 2031+stampante Seikosha GP80M+registratore con dischi e cassette il tutto con due mesi di vita e in garanzia. Listino L. 4125000+IVA vendo a L. 3500000+IVA trattabili. Peretti Nicola, via Stelvio 44, 23017 Morbegno (SO), tel. 601401

Vendo SinIciair ZX81 16 K RAM+circa 20 programmi originali tradotti+cavi e manuali acquistato a giugno. L. 350000 trattabili. Telefonare ore pasti. Arvat Riccardo, via Boston 34, 10137 Torino, tel. 011-354030

Vendo ZX81+16 K RAM con manuali inglese italiano+alimentatore e una cassetta con 15 programmi e un flexidisk con programma il tutto per L. 400000, comprato il tutto il 14/10/82. Angelo Vianello, S. Croce 2158, 30100 Venezia, tel. 041-25723

Vendo VIC 20+registratore C2N+manuale in italiano a L. 600000. Stefano Bignoli, via A. Diaz 1, 28074 Ghemme (NO), tel. 0163-840818

Vendo per N.E.Z80 drive Basf 5" 1/4+interfaccia a floppy disk con relativa piattina + alimentatore + N.E-.DOS + documentazione ampia il tutto a L. 600000, solo se veramente interessati, poss. zona Roma e dintorni, ore serali. Roberto Rocchetti, via Casilina 329, 00176 Roma. tel. 06-2776324

Vendo Sharp MZ 80 Kbyte RAM+linguaggi: Basic, Pascal, Assembler (editor+loader+debugger), machine language; completi di manuali e cassette dimostrative, il tutto a L. 2000000. Giorgio Ardenghi, via Ripalta 6/B, 20097 S. Donato Milanese (MI), tel. 02-5279433

Vendo VIC 20+adattatore+registratore+espansione per alta risoluzione+manuale Jackson e VIC revealed, cartuccia Star battle, il tutto scontato del 20% a L. 630124. Telefonare ore ufficio, completa documentazione. Fabrizio Bestetti, via G. Verdi 6, 24040 Canonica d'Adda (BG), tel. 035-883107

Vendo DAI 48 K+manuale inglese e italiano L. 1400000. Piero Marcheselli, via Bari 32/A, 20143 Milano, tel. 02-8134310

Vendo ZX81 ancora da sballare con espansione a 16 K completo di trasformatore e software vario (matematica e game). Tutto a solo 290000 e spese postali a mio carico. Antonio Ventura, via Frusa 18, 50131 Firenze, tel. 588454

Cerco drive per TRS-80 mod. 1, già dispongo di interfaccia di espansione. Massimo Politi, V.le G. D'Annunzio 31, 65100 Pescara, tel. 085-690786

Vendo VIC 20+registratore C2N+espansione 3 K grafica e funzioni programmabili+cassetta gioco dell'Othello. Tutto a L. 688000+gioco Super Lander Jupiter. Bells, via G. Zanotti 21, 40137 Bologna, tel. 051-303098

Vendo Sinclair ZX80 a L. 120000, ricevitore Yaesu FRG7 (da 0,5 a 30 MHz AM-CW-SSB) L. 350000, ricevitore STE ARAC 102 (144/146 e 26/28 MHz AM-FM-SSB) L. 120000. Filippo Salomone, via P. Giuliani 5, 21047 Saronno (VA), tel. 02-9625998

Vendo ZX81+16 K RAM+2 manuali+registratore interfacciato causa regalo Apple. Nuovo L. 550000 vendo a L. 450000 (trattabili) senza registratore L. 400000. Regalo BASF C90 con: Lotto, Dadi, Cavalli, Roulette, Yatzee, ecc. Luca Concone, via Grancini 4, 20145 Milano, tel. 02-4690700

Vendo personal computer Sharp M280K memoria 48 K. P. Paolo Ranocchia, via G. Chiabrera 52, 00145 Roma, tel. 5114554

Vendo ZX81 con 16 K RAM completo cavi alimentatore e manuali a L. 350000+calcolatrice CASIO FX702P programmabile in Basic, acquistata dicembre '82 a L. 220000+registratore Nordmende portatile L. 50000. In blocco L. 600000. Carlo Picca, via C. da Casale 27, 00049 Velletri (Roma), tel. 06-9633525

Vendo VIC 20 (7 mesi di vita)+programmi a L. 480000! Espansione 8 K RAM (nuova) a L. 100000 in blocco a L. 550000! (Solo zona Varese). Telefonare di sabato a Paolo Russo, via Statale 36, 21030 Marchirolo (VA), tel. 722949

Compro interfaccia per Epson per Apple. Offro massimo L. 130000. Inoltre vendo stampante Seikosha+interfaccia per Apple+1 pacco carta+2 nastri a L. 600000 non trattabili. Marco Verdesca, via Res. Archi Milano 2, 20090 Milano, tel. 2138347

PICCOLI ANNUNCI

PERSONAL SOFTWARE

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

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

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

Città ..... Tel. .....

IL BASIC E LA GESTIONE DEI FILE

Il libro si rivolge in modo particolare a chi già conosce il Basic e desidera poter realizzare programmi che prevedano l'uso di file residenti su disco. Dopo aver preso in esame, utilizzando numerosi esempi pratici, le particolarità del Microsoft, si passa alla descrizione delle istruzioni necessarie ad una corretta gestione dei file su disco, sia ad accesso diretto che sequenziale. Una terza parte del libro è infine interamente dedicata alla esposizione dei metodi pratici per l'uso dei file ad accesso diretto e dei data base.

Cod. 515H **L. 11.000 Pagg. 164**

50 ESERCIZI IN BASIC

Una raccolta completa e progressiva di esercizi riguardanti matematica, gestione, ricerca operativa, gioco e statistica. Ciascun esercizio proposto comporta l'enunciazione e l'analisi del problema, la risoluzione mediante flow-chart e commenti, così come un programma che implementa la soluzione, illustrato da semplici esempi rappresentativi.
Questo metodo mette in grado il lettore di verificare passo passo le sue conoscenze e il livello di apprendimento raggiunto.

Cod. 521A **L. 13.000 Pagg. 208**

75 PROGRAMMI IN BASIC PER IL VOSTRO COMPUTER

Il volume raccoglie ben 75 programmi originali scritti in un Basic generico, utilizzabili, salvo poche e semplici modifiche, sulla maggior parte dei personal computer in commercio, a cassetta come a floppy disk. Per ciascuno, dopo una descrizione introduttiva, viene fornito il listing e un campione di esecuzione. Così come sono, i programmi proposti (tutti verificati), costituiscono un valido ausilio per chiunque debba risolvere problemi di matematica, statistica, finanza o, genericamente, di pratica utilità.

Cod. 551D **L. 12.000 Pagg. 196**

GIOCARE IN BASIC

Nei giochi, il lettore può ritrovare tutte quelle situazioni reali di programmazione che gli saranno indispensabili nella comprensione e realizzazione di qualsiasi applicazione interattiva del proprio computer, anche le più sofisticate. Questo senza annoiarsi, ma entrando da subito all'interno della materia per imparare a comprendere il BASIC, il proprio computer e i computer in genere.

Cod. 522A **L. 20.000 Pagg. 324**

... dalla libreria JACKSON

DIVISIONE LIBRI

CEDOLA DI COMMISSIONE LIBRARIA

Ritagliare (o fotocopiare) e inviare a
Gruppo Editoriale Jackson
Via Rosellini, 12 - 20124 Milano

Nome e Cognome _____

_____

Indirizzo _____

_____

Cap. _____ Città _____

Provincia _____

Partita I.V.A. (indispensabile per le aziende)

Si richiede l'emissione della fattura ☐

Inviatemi i seguenti libri:

Codice Libro	Quantità	Codice Libro	Quantità	Codice Libro	Quantità	Codice Libro	Quantità

☐ Pagherò al postino il prezzo indicato + l. 2.000 per contributo fisso spese di spedizione

☐ Allego assegno n° ..... di L. .....

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

☐ Non Abbonato ☐ Abbonato sconto 10% ☐ L'Elettronica ☐ Elettronica Oggi ☐ Automazione Oggi ☐ Elektor ☐ Informatica Oggi ☐ Computerworld ☐ Bit ☐ Personal Software ☐ Strumenti Musicali ☐ Videogiochi

Servizio programmi

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

N.	Sistema	Programmi	Supporto	pubblicato in *Personal Software* n.	Prezzo
1	Apple II+	La carta del cielo Collisione	floppy 5″ DOS 3.3	3 pag. 83 3 pag. 93	30.000
2	TRS-80 mod. I	Backgammon	floppy 5″ DOS 2.3	3 pag. 89	25.000
3	PET/CBM 3032/4032	Editor/Assembler in Basic	floppy 5″ 3032/4032+3040/4040	2 pag. 33	40.000
4	Apple II+	Interi in precisione multipla Grafica 3D	floppy 5″ DOS 3.3	4 pag. 17 4 pag. 47	40.000
5	PET/CBM 3032/4032	Gioco del calcio	floppy 5″ 3032/4032+3040/4040	4 pag. 67	25.000
6	Apple II +	Pretty Printer Shape Table	floppy 5″ DOS 3.3	5 pag. 27 5 pag. 58	30.000
7	Apple II +	Data base modulare	floppy 5″ DOS 3.3	7 pag.	25.000

Spedire in busta chiusa a

PERSONAL SOFTWARE
Servizio Programmi
Via Rosellini 12
20124 Milano

Inviatemi i seguenti dischi di *Personal Software*

n. _____

per un totale + L. 2.000 come contributo fisso_____
spese di spedizione che pagherò al postino alla consegna del pacco.

_____
Cognome e nome

_____
Indirizzo

_____
Cap., Località

_____
Firma

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

Z-80

Pag. 530 **L. 26.000**
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. 390
L. 25.000
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

UNA PUBBLICAZIONE
DEL GRUPPO EDITORIALE JACKSON

PERSONAL SOFTWARE

ANNO 2 N. 7 GIUGNO 1983

DIRETTORE RESPONSABILE: Giampietro Zanga

DIRETTORE: Mauro Boscarol

REDAZIONE: Completo Software - Padova

HANNO COLLABORATO A QUESTO NUMERO:
R. Amasino, E.M. Albani, R.W. Anderson, B. Del Medico, F. Ferrari Bravo, E. Ferreguti, N.E. Fischer, A. Guida, M. Pelczarski, M. Redolfi, B. Saccà, F. Santini, V. Scaffidi, M. Spero.
Copertina e illustrazioni: MORGANA artefatti comunicativi - Padova
Fotocomposizione: Composizioni Grafiche - Padova

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

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

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

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

STAMPA: 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

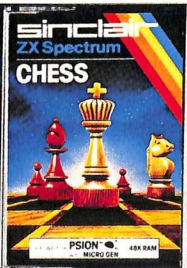
SEDE LEGALE: Via Vincenzo Monti, 15 - 20123 Milano

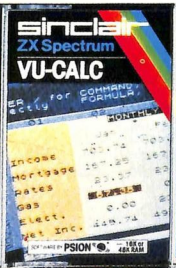
DIREZIONE EDITORIALE: Giampietro Zanga e Paolo Reina
COORDINAMENTO EDITORIALE: Daniele Comboni

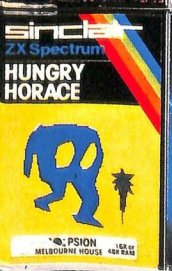
- 16 o 48 kbytes RAM.
- grafica ad alta risoluzione (256x192 punti).
- 8 colori da utilizzare con la più assoluta libertà per testo, sfondo, bordo, in campo diretto o inverso, con due gradi di luminosità, a luce fissa o lampeggiante.
- Tastiera multifunzione con maiuscole, minuscole, simboli grafici,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

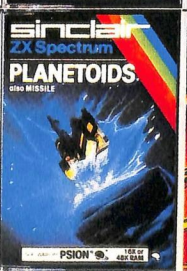
ORA C'E'!

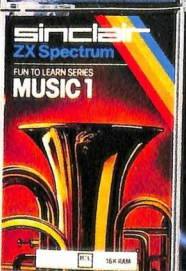
ZX Spectrum

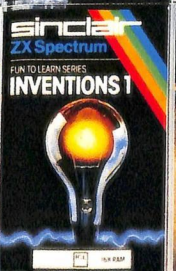


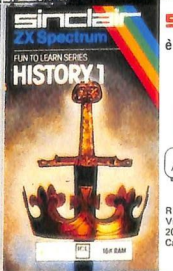












sinclair

è distribuito da

REBIT COMPUTER

A DIVISION OF G.B.C.

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

Harden Italia. Il salto di qualità.

Dal personal computer al professional computer.

Nel quadro di una filosofia aziendale in evoluzione, Harden Italia riconferma la validità della proposta del Sirius 1. Il Sirius 1, con tutta la potenza del suo microprocessore a 16 bit, con 5 MHz, e una memoria centrale che può arrivare 896 KBytes, è uno dei più avanzati della nuova generazione dei Personal.

Oltre ad una enorme capacità di archiviazione dei dati (dai 1240 KBytes del Sirius 1 agli 11.840 KBytes del Sirius 1b) il Sirius può contare su alcune caratteristiche che un tecnico e un professionista non possono non apprezzare: dall'interfacciamento con due porte seriali e una parallela programmabile da software, ai sistemi operativi (MS-DOS della Microsoft e CP/M86 della Digital Research), fino ai linguaggi di alto livello come il BASIC-86 (interprete e compilatore), l'Assembler, il COBOL, il Fortran, il Pascal.

Oltre che sul software vero e proprio (programmi come il Dbase II, il SuperCalc, il Multiplan o l'Harden-text e l'Harden-data) il Sirius 1 si avvale dei così detti "Tool Kits", una serie cioè di utilities compatibili con qualsiasi linguaggio che permettono una stesura dei programmi più facile e più completa come ad esempio l'AutoSort, il FABS, una gestione sofisticata IS, ecc. In più, il Sirius 1 è distribuito e assistito dalla Harden Italia su tutto il territorio nazionale.

Per saperne di più sul Sirius 1, sui suoi programmi o su dove sono i punti di vendita Harden più vicini, chiamare (0372)-63136 oppure (02)-651645: risponde la Harden Italia.

Job Line

HARDEN ITALIA

Harden Italia S.p.A. Direzione generale e uffici commerciali
20121 Milano - via dei Giardini, 4 - tel. (02) 651645
Sede operativa e uffici commerciali
26048 Sospiro (CR) - tel. (0372) 63136 - telex: 3205881

SIRIUS 1 CONFIGURAZIONE BASE
(128 KBYTES RAM, 1240 KBYTES FLOPPY DISC)
DA OGGI L. 6500000