LA PRIMA RIVISTA EUROPEA DI SOFTWARE PER PERSONAL COMPUTER

ISSN 0392-8896

PERSONAL SOFTWARE

ANNO 3 N.17
APRILE 1984 - L. 4.000

UNA PUBBLICAZIONE DEL GRUPPO EDITORIALE JACKSON

Spedizione in abb. postale Gruppo III/70

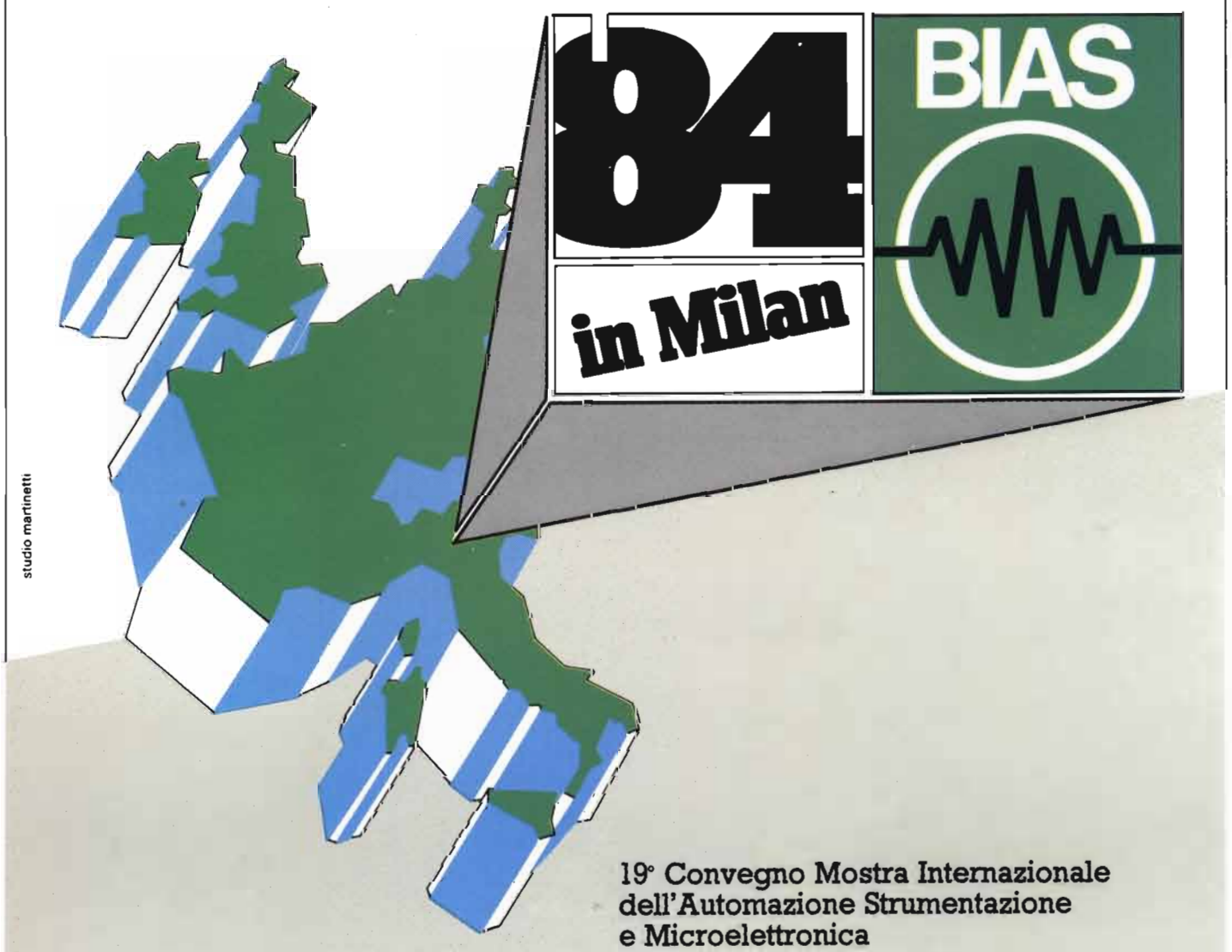
- SPRITE CON IL C 64
- RALLY SAFARI PER SPECTRUM
- SATEL: LO ZX81 CONTROLLA I PIANETI
- LINGUAGGIO MACCHINA PER VIC 20 e C 64
- MACRO ISTRUZIONI CON L'APPLE
- LA BATTAGLIA DEL LAGO GHIACCIATO: UN GIOCO PER PET-CBM

Esposizioni Internazionali dell'Automazione

...1982 Parigi "MESUCORA"... 1983 Düsseldorf "INTERKAMA"

1984 MILANO - B.I.A.S.

Solo il BIAS nel 1984 in Europa presenta l'Automazione e la Microelettronica



studio martinetti

19° Convegno Mostra Internazionale dell'Automazione Strumentazione e Microelettronica

Fiera di Milano
29 novembre - 4 dicembre 1984

- Sistemi e Strumentazione per l'Automazione la regolazione ed il controllo dei processi Robotica, sensori e rilevatori
- Apparecchiature e Strumentazione per laboratorio, collaudo e produzione
- Componentistica, sottoassiemi periferiche ed unità di elaborazione
- Micro, Personal Computer, Software e accessori

E.I.O.M. Ente Italiano Organizzazione Mostre
Segreteria della Mostra
Viale Premuda 2
20129 Milano
tel. (02) 796096/421/635 - telex 334022 CONSEL

in concomitanza con la 8ª RICH e MAC '84

In copertina: L'astrofisica è una scienza in cui è possibile impiegare in modo affascinante i personal computer. Il programma Satel calcola il passaggio dei satelliti artificiali.

ARTICOLI

RUBRICHE

GUIDA

N. 17
APRILE 1984

Personal e home computer

Provando e riprovando

Nicole Bréaud-Pouliquen
La pratica dell'APPLE

Per imparare a usare un calcolatore bisogna... usarlo.
Solo così, ad esempio, è possibile scoprire e sfruttare le immense risorse operative offerte dall'**APPLE**. Provando, riprovando e... leggendo un manuale come questo.

Scritto da un vero esperto, il libro si compone di 3 capitoli fondamentali:

- **Il sistema APPLE II"** dedicato all'hardware e al software
- **"Il BASIC APPLESOFT"** con le istruzioni, i sottoprogrammi, gli operatori aritmetici e logici
- **"Il disegno e la grafica"** con le zone di memoria RAM e le funzioni grafiche.

Il tutto arricchito da numerosi esempi ed esercitazioni con soluzioni: affinchè la pratica abbia l'immediata soddisfazione del riscontro.

130 pagine
Lire 10.000
Codice **341D**

GRUPPO EDITORIALE JACKSON

Attenzione compilare per intero la cedola
ritagliare (o fotocopiare) e spedire in busta chiusa a:
GRUPPO EDITORIALE JACKSON
Divisione Libri
Via Rosellini, 12 - 20124 Milano

CEDOLA DI COMMISSIONE LIBRARIA

VOGLIATE SPEDIRMI

n° copie	codice	Prezzo unitario	Prezzo totale
	341D	**L. 10.000**	

☐ Pagherò contrassegno al postino il prezzo indicato più **L. 2000 per contributo fisso spese di spedizione.**

Condizioni di pagamento con esenzione del contributo spese di spedizione:

☐ Allego assegno della Banca

n° _____

☐ Allego fotocopia del versamento su c/c n. 11666203 a voi intestato

☐ Allego fotocopia di versamento su vaglia postale a voi intestato

Nome
Cognome
Via
Cap Città Prov.
Data Firma

Spazio riservato alle Aziende. Si richiede l'emissione di fattura

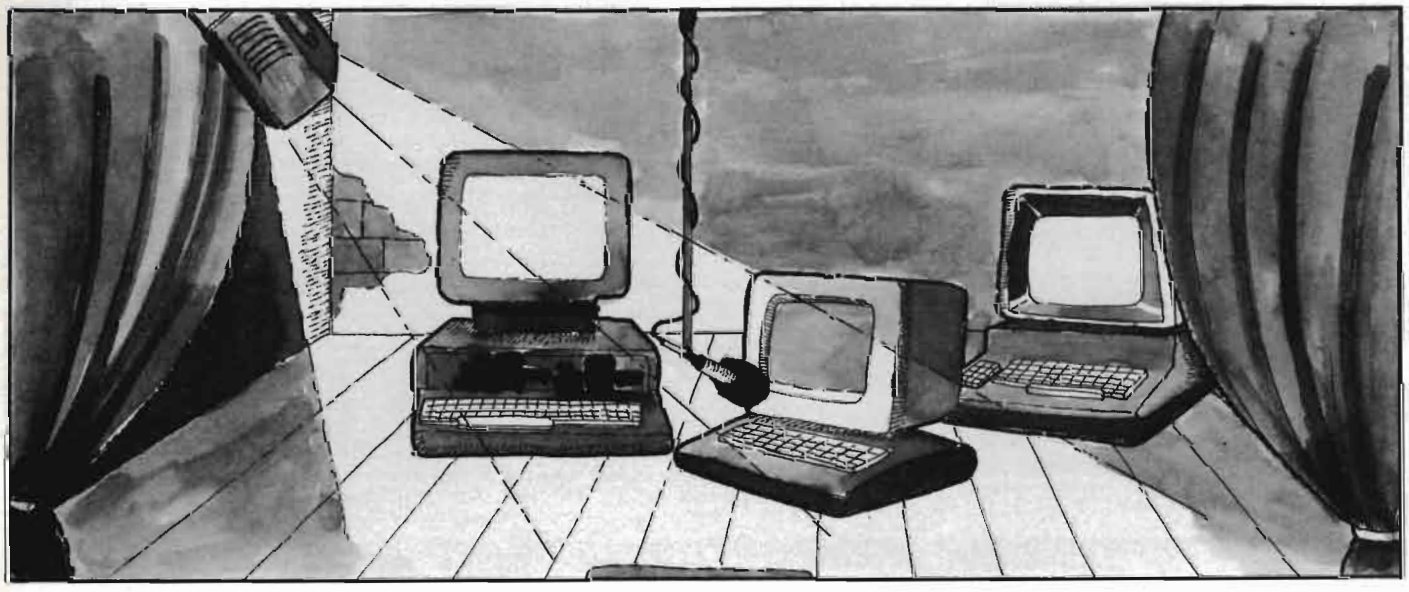
Partita I.V.A. |_|_|_|_|_|_|_|_|_|_|_|

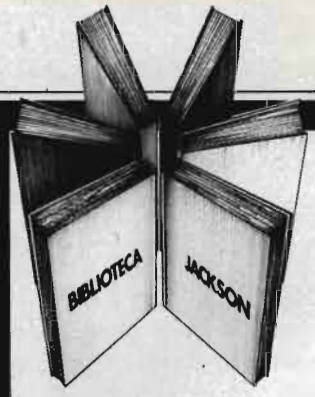
EDITORIALE

Uno sguardo dietro l'angolo

di Riccardo Paolillo

La Cooperativa Teatro di Porta Romana di Milano, ha recentemente messo in scena una commedia dal titolo molto familiare per noi del Gruppo Editoriale Jackson: "Bit".
La vicenda, per raccontarla in due parole, si svolge in una cittadina che vive all'ombra di una grande azienda di informatica, dove il tempo scorre scandito dal succedersi di assunzioni e pensionamenti che determinano un naturale ricambio generazionale. Tutto questo fino a quando subentra la nuova tecnologia a sconvolgere la solita routine e, grazie ai terminali domestici, tutti possono lavorare a casa, anche se anziani o malati.
La conclusione è in chiave ironica: al vecchio ex-dipendente, da tempo pensionato e ripescato grazie alla sua esperienza, viene recapitato un piccolo terminale con il quale lavorare a casa, mentre il giovanissimo successore, ormai inutile per l'azienda, viene paradossalmente collocato in pensione.
Anche se non completamente d'accordo con certe tesi sostenute dall'autore, occorre tuttavia sottolineare come si sia riusciti ad affrontare in modo spiritoso ed efficace un tema non certo semplicissimo. Ma come sarà il futuro telematico? Si possono fare alcune ipotesi, basandosi su quelle che sono le tendenze attuali.
In questo periodo, parallelamente alle ricerche in atto per sviluppare processori sempre più veloci e potenti, si sta lavorando molto nel campo delle telecomunicazioni e delle memorie a basso costo. Uno degli obiettivi è quello di decentralizzare il lavoro per determinate categorie impiegatizie e professionali, rendendo non più necessario il dover raggiungere quotidianamente un ufficio, ma permettendo di lavorare ovunque, anche a casa e con orari variabili.
I benefici ipotizzati sono facilmente immaginabili: diminuzione di strutture non produttive (uffici) e limitazione dei costi dovuti ai tempi di trasferimento del personale. Quindi razionalizzazione di utilizzo di alcuni sistemi, come ad esempio i trasporti, che rischiano il collasso in quanto sollecitati pesantemente negli orari di punta e non sfruttati uniformemente durante tutto l'arco della giornata. Inoltre una migliore utilizzazione del tempo libero individuale: infatti l'orario di lavoro diventerebbe molto più flessibile e personalizzabile a beneficio delle attività ricreative che potrebbero essere molto più dilazionate nell'arco della settimana e non concentrate nelle ore serali e festive.
Il rovescio della medaglia per questi nuovi modelli di vita è costituito principalmente dal rischio che il processo di asocializzazione tuttora in corso, soprattutto nelle grandi città, si accentui ulteriormente e porti a far sì che ognuno viva esclusivamente all'interno del suo micro-cosmo personale o al massimo familiare. Questa è indubbiamente una grande incognita e a seconda di come verranno affrontati questi problemi avremo un modo di vivere nettamente diverso nei prossimi decenni.
Un fatto comunque è certo: la storia ci insegna che non è possibile andare indiscriminatamente contro o addirittura rifiutare nuove tecnologie. Ma occorre sempre fare in modo che le nuove tecniche, con le loro conseguenze sia positive che negative, vedano sempre l'uomo come figura dominante di utilizzatore. ■





Informatica

Esperti a confronto su attualità e prospettive della Computer Grafica

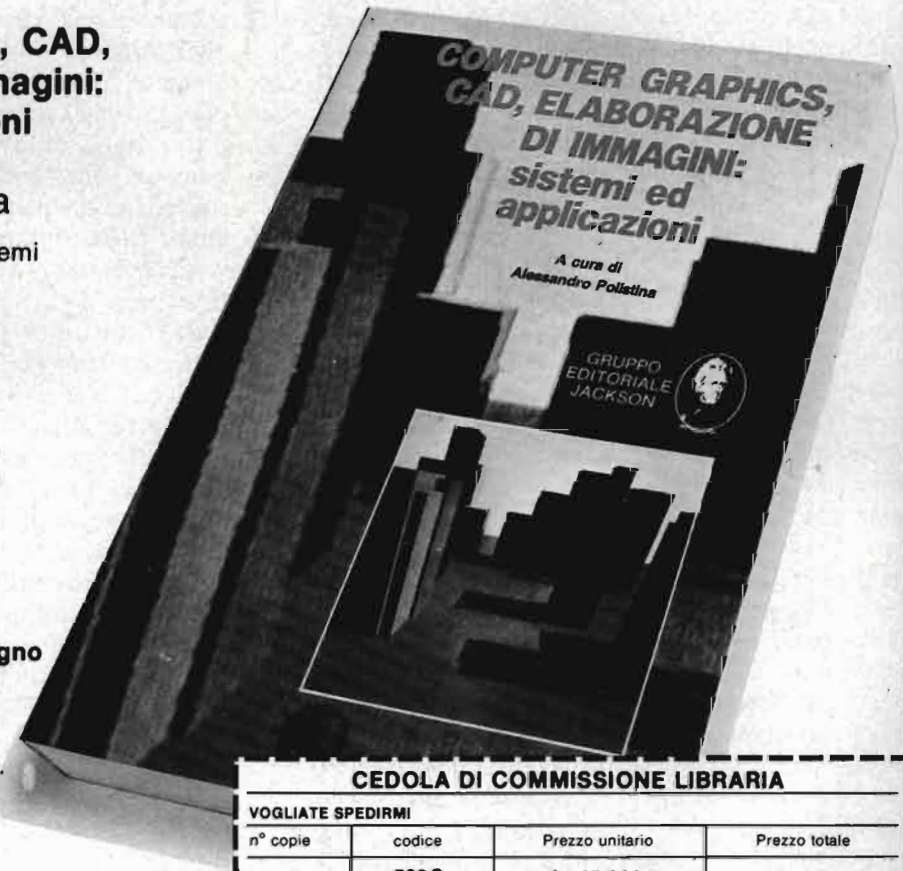
Computer Graphics, CAD, elaborazione di immagini: sistemi e applicazioni

A cura di
Alessandro Polistina

Linguaggi e algoritmi, sistemi grafici, CAD/CAM, didattica e formazione professionale, Computer Graphics e Editoria, modellazione di solidi, CAD in architettura, CAD meccanico, acquisizione e elaborazione di immagini, elaborazione di immagini e scienze biomediche, cartografia e pianificazione editoriale, immagini sintetiche per la televisione....

Tutti gli Atti del 3° Convegno Nazionale AICOGRAPHICS riuniti in un solo volume a disposizione di operatori, sperimentatori, appassionati. 512 pagine, numerosissimi schemi, un'Appendice con 33 illustrazioni a colori..

Lire 45.000
Codice **529C**



CEDOLA DI COMMISSIONE LIBRARIA

VOGLIATE SPEDIRMI

n° copie	codice	Prezzo unitario	Prezzo totale
	529C	**L. 45.000**	

☐ Pagherò contrassegno al postino il prezzo indicato più **L. 2000 per contributo fisso spese di spedizione.**

Condizioni di pagamento con esenzione del contributo spese di spedizione:

☐ Allego assegno della Banca _____

n° _____

☐ Allego fotocopia del versamento su c/c n. 11666203 a voi intestato

☐ Allego fotocopia di versamento su vaglia postale a voi intestato

Nome
Cognome
Via
Cap Città Prov.
Data Firma

Spazio riservato alle Aziende. Si richiede l'emissione di fattura

Partita I.V.A. |_|_|_|_|_|_|_|_|_|_|_|

GRUPPO EDITORIALE JACKSON

Attenzione compilare per intero la cedola
ritagliare (o fotocopiare) e spedire in busta chiusa a:
GRUPPO EDITORIALE JACKSON
Divisione Libri
Via Rosellini, 12 - 20124 Milano

POSTA

Uno Spectrum bollente

Posseggo uno ZX Spectrum nella versione da 48 Kbyte. In proposito vorrei porre alcune domande:
1) Mi hanno detto che lo Spectrum produce una notevole quantità di calore dopo più di tre ore di funzionamento. Come è dunque possibile che il mio Spectrum si surriscaldi dopo solo un quarto d'ora con conseguente degrado dell'immagine e della velocità del computer? (N.B. Il rivenditore mi ha sostituito la macchina 2 volte, pensando a un guasto dovuto al trasporto).
2) È possibile la costruzione o l'adattamento di un joystick allo Spectrum? Se sì, come?
3) È possibile avere informazioni esaurienti sui comandi (e sulla loro utilizzazione): OPEN #; CLOSE #; MOVE; ERASE; CAT?

Vincenzo Romano
Cervino (LE)

Lo Spectrum, a causa delle ridotte dimensioni, è soggetto a un sensibile surriscaldamento anche dopo pochi minuti di funzionamento. Questo fatto, però, non dovrebbe influire sulle prestazioni della macchina e sulla qualità dell'immagine video.
Una delle possibili cause di malfunzionamento potrebbe essere determinata dall'alimentatore, che le consigliamo di controllare.

Per quanto riguarda la possibilità di collegare un joystick, le segnaliamo una delle ultime novità di casa Sinclair: la ZX Interfaccia 2. Questa scheda si collega direttamente allo Spectrum o alla Interfaccia 1 (quella che permette di collegare i famosi Microdrive) e può ospitare speciali cartucce (saranno disponibili soprattutto giochi). Alla Interfaccia 2 possono essere connessi direttamente 2 joystick di tipo standard con attacco a nove poli. Attualmente non è ancora importata, ma pensiamo (e speriamo) che entro breve tempo sarà disponibile anche in Italia ad un prezzo che, sebbene non sia ancora stato fissato, sarà sicuramente molto interessante.

I comandi che lei cita, e quelli che compaiono sotto altri tasti della stessa linea, sono predisposti per un utilizzo con Microdrive.

■ ■ ■

Data base per C 64

Sono un programmatore BASIC dilettante, autodidatta, in possesso da poco tempo di un C 64. Pur riuscendo già a costruire semplici programmi, che mi aiutano a migliorare le mie conoscenze, vorrei utilizzare il mio C 64 anche in maniera più pratica, cioè vorrei formarmi vari archivi memorizzati a cui attingere in caso di bisogno senza avere una grande quantità di fogli e foglietti sparsi. Però non sono ancora riuscito ad impostare un programma che mi permetta di fare questo, sicuramente per la mia scarsa dimestichezza sia con il BASIC che con la macchina.

Quindi con questa mia lettera sono a chiedervi cortesemente, e probabilmente non sono il solo, se e quando avete la possibilità di pubblicare sulla vostra rivista listati di programmi di questo tipo, oppure, se l'argomento è già stato da voi trattato se mi indicate il relativo numero della rivista oppure altre vostre pubblicazioni o meno che trattino questo argomento.

Valentino Terzi
Bologna

Uno dei motivi per cui non abbiamo a tutt'oggi pubblicato un data base per C 64, è costituito dal fatto che un programma di questo tipo richiede necessariamente la disponibilità di almeno una unità a floppy disk per la memorizzazione dei dati.

Infatti, a parte la lentezza in fase di lettura e scrittura che già pregiudica un utilizzo pratico efficiente, il nastro essendo sequenziale non consente determinati inserimenti e aggiornamenti, operazioni fondamentali anche per un data base elementare.

Ora però che il C 64 è così largamente diffuso e molti utenti dispongono di una unità a disco, stiamo studiando un programma che soddisfi questa diffusissima esigenza.

Peraltro invitiamo chiunque avesse già realizzato qualche applicazione interessante a farsi vivo con noi in modo di permettere a tutti di avere disponibile un programma che è sicuramente fra i più utili.

*Infine, soprattutto a titolo di curiosità e per proporre spunti per eventuali conversioni, ricordo che anni fà nel 1980 la nostra rivista consorella "**Bit**" pubblicò in 4 puntate nei numeri 7-8/9-10-11 un potente e completo Personal Data Base per Apple II, che ancora oggi è uno dei programmi più diffusi ed utilizzati per il personal della mela.*

■ ■ ■

Il TI dimenticato

Sono un appassionato di personal computer.

Ho comprato da poche settimane un TI 99/4A, uno degli ultimi rimasti sul mercato: ho notato che, forse proprio per il fatto che il TI 99 non è più in commercio, nelle riviste specializzate come la vostra, di software ce n'è sempre meno.

Poichè è solo da poco che sono alla ricerca di programmi e ho potuto acquistare solo i numeri 10/11, 12,13 e 14 di **Personal Software**, rivista interessantissima sotto ogni aspetto, vi sarei grato se pubblicaste l'elenco dei numeri della rivista in cui avete riportato programmi (o utili informazioni) per il TI 99, magari con i relativi titoli.

Antonio Lo Nardo
Palermo

Rispondendo a lei, cerchiamo di accontentare anche altri lettori che ci hanno fatto richieste analogie.
Per quanto riguarda la nostra rivista, ci sembra che gli articoli dedicati al TI 99 siano rimasti in numero piuttosto costante: in particolare, nei numeri che lei cita, abbiamo pubblicato sempre almeno un articolo e un pezzo nelle rubriche "I segreti dei personal" o "Conversioni".
Nei prossimi mesi pubblicheremo interessanti lavori del nostro collaboratore Sergio Borsani, che ci risulta incontrano il vostro interesse, e anche programmi che altri lettori ci hanno inviato. In particolare Filippo Cerulo (altro nome familiare per i nostri lettori) presenterà un programma per un utilizzo grafico avanzato del Texas.

Per quanto riguarda articoli pubblicati in numeri precedenti le ricordiamo: nel n. 8/9 "Archivio scacchistico", "Programmare grafici" e "PRINT e INPUT in un contesto grafico" per quanto riguarda la rubrica "I segreti dei personal". Per la stessa rubrica nel n. 7 abbiamo pubblicato "INPUT e stampa estesa".

PERSONAL SOFTWARE

Rally Safari

Una appassionante corsa automobilistica con il vostro ZX Spectrum

di *Ivano Parbuono*

La macchina da sempre è una delle cose che più appassiona e coinvolge un po' tutti. C'è chi la usa come mezzo di trasporto, chi per fare gite e chi invece la vede come mezzo di competizione.

Nel campo dei videogiochi non mancano certo le gare automobilistiche e su questa scia viene qui presentato un programma in BASIC che simula un Rally Safari Monaco-Montecarlo-Nairobi. La partenza avviene a Monaco e, attraversando la foresta nera, si deve giungere a Montecarlo ma qui cominciano le difficoltà; il traffico di Montecarlo è caotico e per poter proseguire si devono evitare incidenti e se ci si riesce si può proseguire verso la nostra meta che è il Kenia, più precisamente Nairobi. A questo punto la pista diventa ancora più difficile: la savana è piena di ostacoli, ci sono cactus dappertutto e si incontrano animali come le giraffe e gli elefanti che mettono a dura prova le capacità di scegliere il passaggio giusto per poter giungere al traguardo e vincere così la tanto sospirata quanto meritata coppa, unita ad una trionfale musichetta. Le figure da 1 a 7 visualizzano le varie situazioni del rally. Se il gioco appare troppo difficile per il raggiungimento del traguardo si può intervenire alla linea 4120 per il tratto che va da Montecarlo in Kenia togliendo un carattere grafico, oppure alla linea 5120 per il tratto dell'attraversamento della savana togliendo uno o più caratteri grafici. Buon divertimento.

Figura 1. Rappresentazione della partenza.

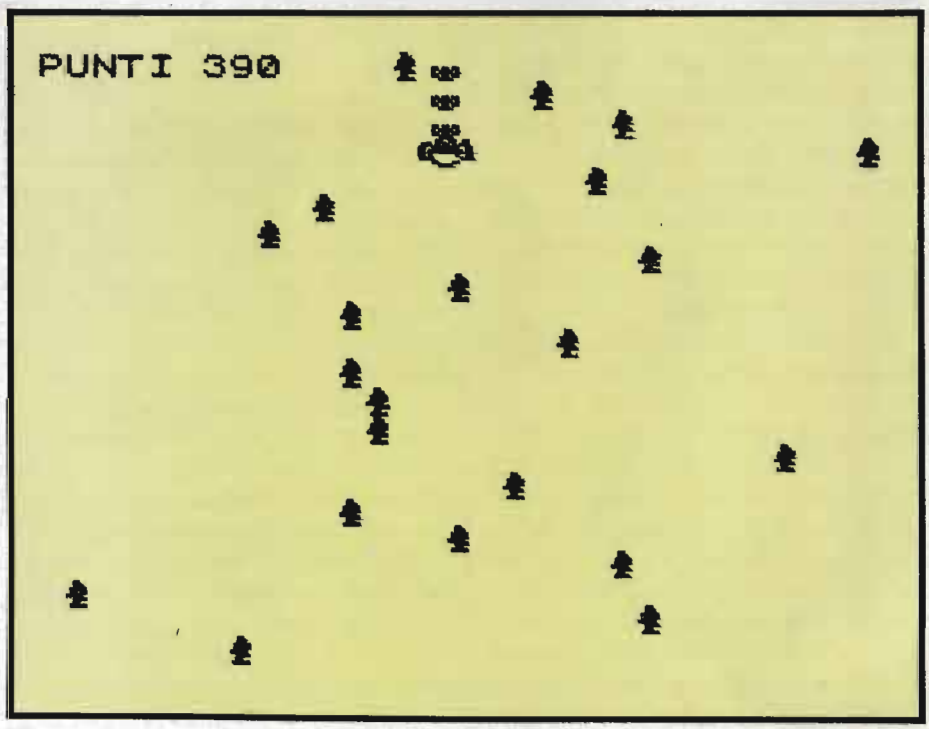


Figura 2. *Attraversamento della foresta nera.*

Il programma

Il programma è riportato nel listato 1 e la figura 8 contiene la lista

Figura 3. *Arrivo a Montecarlo.*

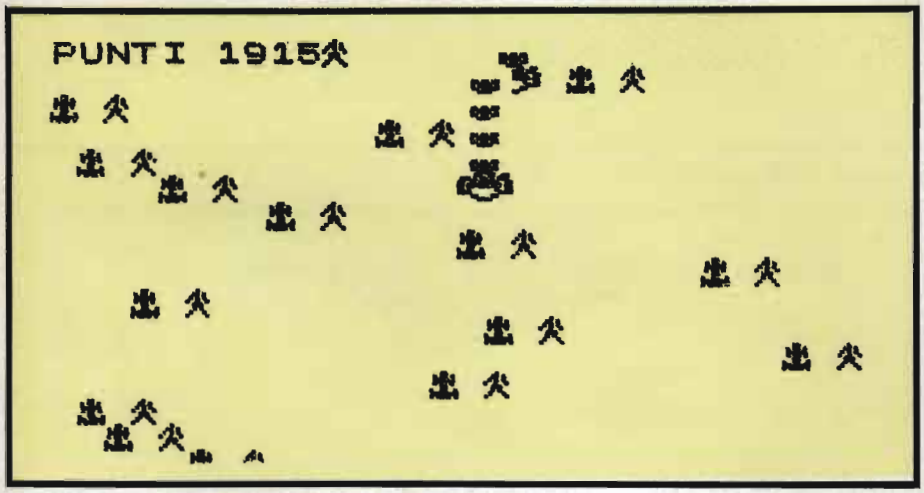


Figura 4. *Attraversamento di Montecarlo.*

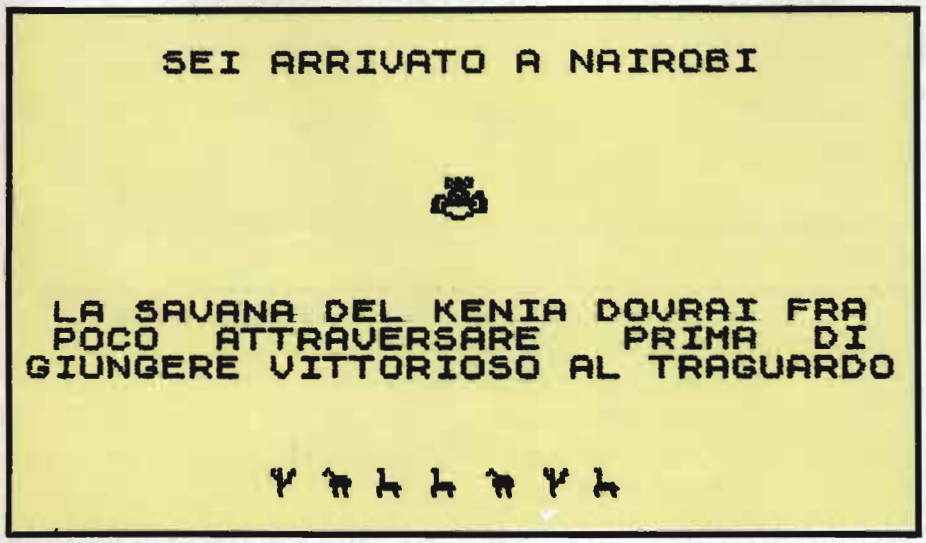


Figura 5. *Arrivo in Kenia.*

dei segni grafici.
La figura 9 è il diagramma a blocchi del programma, che può essere utile a chi volesse studiarne in dettaglio il funzionamento. Il programma inizia subito dalla linea 120 alla 168 con la definizione dei caratteri grafici. Mentre le linee che vanno dalla 188 alla 196 fanno si che appaiano sul video le istruzioni della partenza del Rally Safari.
La linea 200 imposta il tempo della gara, la linea 310 affida ai tasti 5 e 8 il movimento della macchina che a sua volta è rappresentata sul video dalla linea 400.

Le linee 450 e 500 servono a creare sia la foresta che il movimento dello schermo. La linea 550 stampa sul video attimo per attimo il punteggio raggiunto mentre la linea 560 fa diminuire il tempo iniziale da 90 secondi fino a 60. Al raggiungimento di tale tempo, se non si sono creati incidenti durante il percorso, si passa alla linea 4000 che ci avverte che siamo arrivati a Montecarlo.
Se invece durante il percorso si è creato un incidente viene rivelato dalla linea 350 che rimanda il programma alla linea 8000 dove, con un effetto di colori, viene segnalata la fine della gara e stampato sul video il punteggio raggiunto e il massimo punteggio delle gare precedenti. Dopo una breve pausa la macchina viene riposizionata alla partenza pronta per una successiva prova. Dalla linea 4030 alla 5170, anche se con piccole varianti di grafica di punteggio e azzeramento di tempo, si ripete il ciclo qui sopra descritto. Alla linea 5186 viene effettuato un controllo sul tempo che se risulta inferiore a 1 produce l'esecuzione della 5500 che stampa sul video il traguardo creando una piccola pausa e dopo questa, entra in funzione la linea 6000 iniziando a suonare una musichetta per annunciare la vittoria, stampando anche la coppa. Dopo una breve pausa, il programma chiede se si vuole giocare ancora.

■

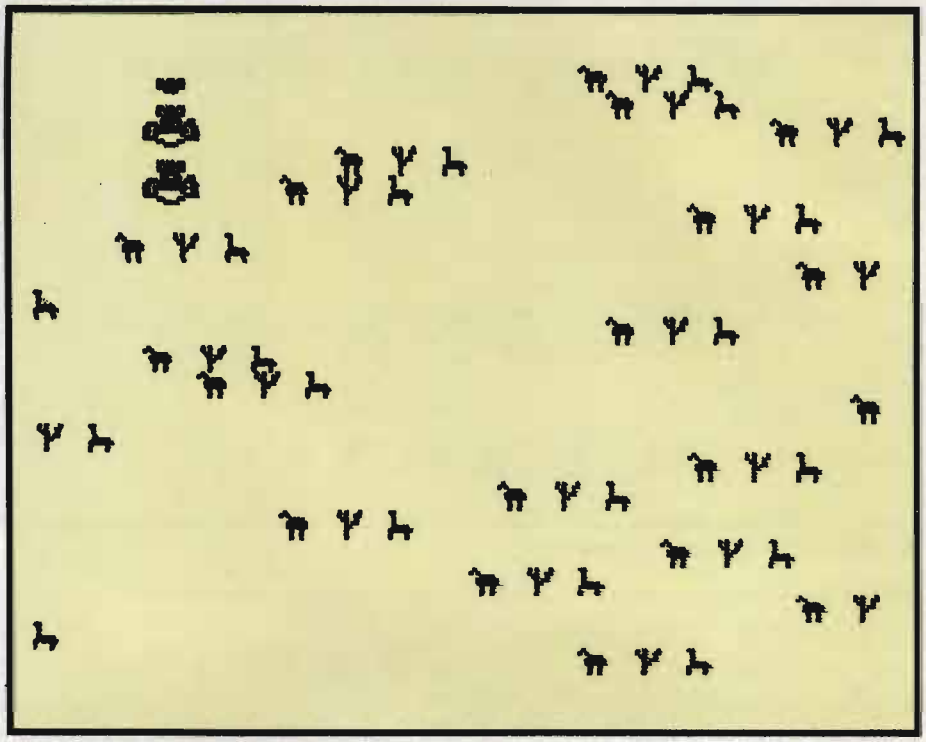


Figura 6. *Attraversamento della Savana.*

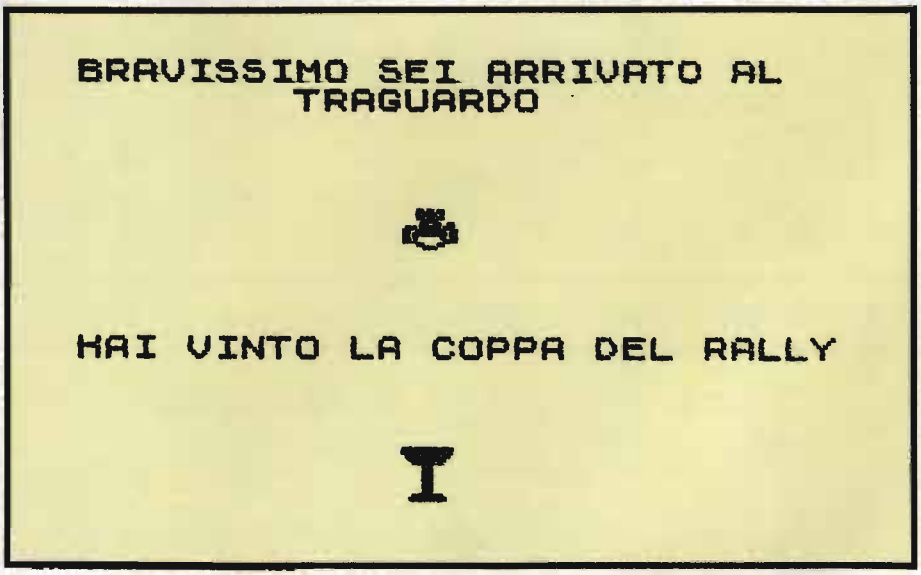


Figura 7. *Consegna della coppa.*

Linea 190 segno grafico C D - E F
Linea 196 segno grafico G
Linea 4016 segno grafico L K
Linea 5020 segno grafico J I H
Linea 6020 segno grafico M O - N P

Figura 8. *Lista dei segni grafici.*

Listato 1. *Programma Rally Safari.*

```
  90 REM  ©   Ivano   Parbuono
           By VERONA 1983
 100 LET hi=0
 120 FOR c=USR "c" TO USR "p"+7
 125 READ user: POKE c,user
 126 NEXT c
 130 DATA 0,0,0,0,13,11,13,3
 135 DATA 0,0,0,0,176,208,176,19
2
 140 DATA 6,117,207,223,208,240,
12,3
 145 DATA 102,174,243,251,19,15,
48,192
 150 DATA 24,28,62,62,124,30,24,
62
 152 DATA 192,64,64,64,124,254,1
32,132
 154 DATA 64,160,30,63,63,18,18,
18
 156 DATA 81,83,50,22,24,16,16,1
6
 158 DATA 24,90,126,90,24,153,25
5,189
 160 DATA 16,52,84,146,56,40,68,
130
 162 DATA 255,255,127,63,31,3,3,
3
 164 DATA 3,3,3,3,3,3,31,63
 166 DATA 254,254,252,248,240,12
8,128,128
 168 DATA 128,128,128,128,128,12
8,240,248
 180 CLS : BEEP .9,7: BEEP .4,7:
 BEEP .3,2: PRINT AT 3,5; INK 1;
"RALLY AUTOMOBILISTICO
      MONACO NAIROBI" : BEEP .6,4
: BEEP .4,2
 190 PRINT AT 6,15; INK 2;"▬";A
T 7,15; INK 2;"◓"
 192 BEEP .9,5: PRINT AT 10,10;
FLASH 1;"M O N A C O";
 194 BEEP .8,3: BEEP .6,12: PRIN
T AT 15,4;"FRA POCO ATTRAVERSERA
I LA": PRINT AT 16,10;"FORESTA N
ERA": BEEP .4,7: BEEP .2,8: BEEP
 .3,1
 196 PRINT AT 18,9; INK 4; FLASH
 1;"♣  ♣♣ ♣♣ ♣  ♣♣": PAUSE 150:
CLS
 200 BORDER 0: PAPER 7: INK 2: C
LS : LET y=2: LET k=16: LET t=90
 210 LET p=0
 300 PRINT AT 2,k;" ": BEEP .015
,1
 310 LET k=k+(INKEY$="8" AND k<3
0)-(INKEY$="5" AND k>0)
 350 IF SCREEN$ (3,k)="" THEN GO
 TO 8000
 352 IF SCREEN$ (3,k+1)="" THEN
GO TO 8000
 400 PRINT AT 2,k; INK 2;"▬";AT
 3,k; INK 2;"◓"
 420 PRINT AT 21,0
 450 PRINT TAB INT (RND*31); INK
 4;"♣"
 500 POKE 23692,10
 550 LET p=p+10: PRINT AT 0,0;"P
UNTI ";p
 560 LET t=t-.2
 570 IF t<60 THEN GO TO 4000
 600 GO TO 300
4012 CLS : BEEP .7,5: BEEP .9,12
: PRINT AT 4,4; FLASH 1;"SEI ARR
IVATO A MONTECALO": PRINT AT 13,
2;"ATTENTO AL TRAFFICO CAOTICO":
 PRINT AT 14,9;"DELLA CITTA'": B
EEP .9,6: BEEP .4,2
4014 PRINT AT 7,15; INK 2;"▬";A
T 8,15; INK 2;"◓"
4016 PRINT AT 18,7; INK 5; FLASH
 1;"♣♣  ♣  ♣♣  ♣♣♣  ♣♣": PAUSE 15
0: CLS
4030 PRINT AT 4,k;" ": BEEP .015
,1
4050 LET k=k+(INKEY$="8" AND k<3
0)-(INKEY$="5" AND k>0)
```

Seguito programma Rally Safari.

```
4060 IF SCREEN$ (5,k)="" THEN GO
 TO 8000
4062 IF SCREEN$ (5,k+1)="" THEN
GO TO 8000
4110 PRINT AT 4,k; INK 2;" ";AT
 5,k; INK 2;"  "
4115 PRINT AT 21,0
4120 PRINT TAB INT (RND*31); INK
 3;"  "
4150 POKE 23692,50
4170 LET p=p+15: PRINT AT 0,0;"P
UNTI ";p
4185 LET t=t-.2
4190 IF t<30 THEN GO TO 5000
4200 GO TO 4030
5000 CLS : BEEP .4,7: BEEP .8,12
: PRINT AT 4,5; FLASH 1;"SEI ARR
IVATO A NAIROBI": BEEP .5,5: PRI
NT AT 13,0;" LA SAVANA DEL KENIA
 DOVRAI FRA  POCO  ATTRAVERSARE
  PRIMA  DI GIUNGERE VITTORIOSO
AL TRAGUARDO": BEEP .9,4: BEEP .
4,5
5010 PRINT AT 8,15; INK 2;" ";A
T 9,15; INK 2;"  "
5020 PRINT AT 19,8; INK 4; FLASH
 1;"  ": PAUSE 150:
 CLS
5030 PRINT AT 4,k;" ": BEEP .015
,3
5050 LET k=k+(INKEY$="8" AND k<3
0)-(INKEY$="5" AND k>0)
5060 IF SCREEN$ (5,k)="" THEN GO
 TO 8000
5062 IF SCREEN$ (5,k+1)="" THEN
GO TO 8000
5110 PRINT AT 4,k; INK 2;" ";AT
 5,k; INK 2;"  "
5115 PRINT AT 21,0
5120 PRINT TAB INT (RND*31); INK
 4;"  "
5150 POKE 23692,50
5170 LET p=p+20: PRINT AT 0,0;"P
UNTI ";p
5185 LET t=t-.2
5186 IF t<1 THEN GO SUB 5500
5190 IF t<0 THEN GO TO 6000
5200 GO TO 5030
5500 PRINT AT 7,7; FLASH 1;"T R
A G U A R D O": PAUSE 225
5600 RETURN
6000 CLS : BEEP .9,4: BEEP .6,3:
 PRINT AT 3,2; FLASH 1;"BRAVISSI
MO SEI ARRIVATO AL ": FLASH 1: P
RINT AT 4,10;"TRAGUARDO": FLASH
0
6005 PRINT AT 8,13; INK 2; FLASH
 1;"  ";AT 9,13; INK 2; FLASH
1;"  ";AT 10,13; INK 2; FLASH
1;"  "
6010 BEEP .5,5: BEEP .8,02: PRIN
T AT 13,2;"HAI VINTO LA COPPA DE
L RALLY": BEEP .4,8
6020 BEEP .7,5: BEEP .4,2: PRINT
 AT 16,13; INK 6; FLASH 1;"  "
;AT 17,13;"  ";AT 18,13;"  "
;AT 19,13;"  ": BEEP .4,6: BEE
P .6,4
6100 GO TO 9000
8002 FOR h=1 TO 100
8005 BORDER RND*7
8010 NEXT h
8020 CLS : PRINT AT 5,4;"HAI TOT
ALIZZATO ";p;" PUNTI"
8025 PRINT AT 9,14; INK 2;"  "
;AT 10,14; INK 2;"  ";AT 11,14
; INK 2;"  "
8026 IF p>hi THEN LET hi=p
8030 PRINT AT 15,4; FLASH 1;"IL
TUO PUNTEGGIO MASSIMO ": PRINT A
T 17,9; FLASH 1;"E' DI ";hi;" PU
NTI"
8050 PAUSE 200
8500 GO TO 200
9000 INPUT "PREMI ENTER PER GIOC
ARE ANCORA"; LINE a$
9100 GO TO 100
```

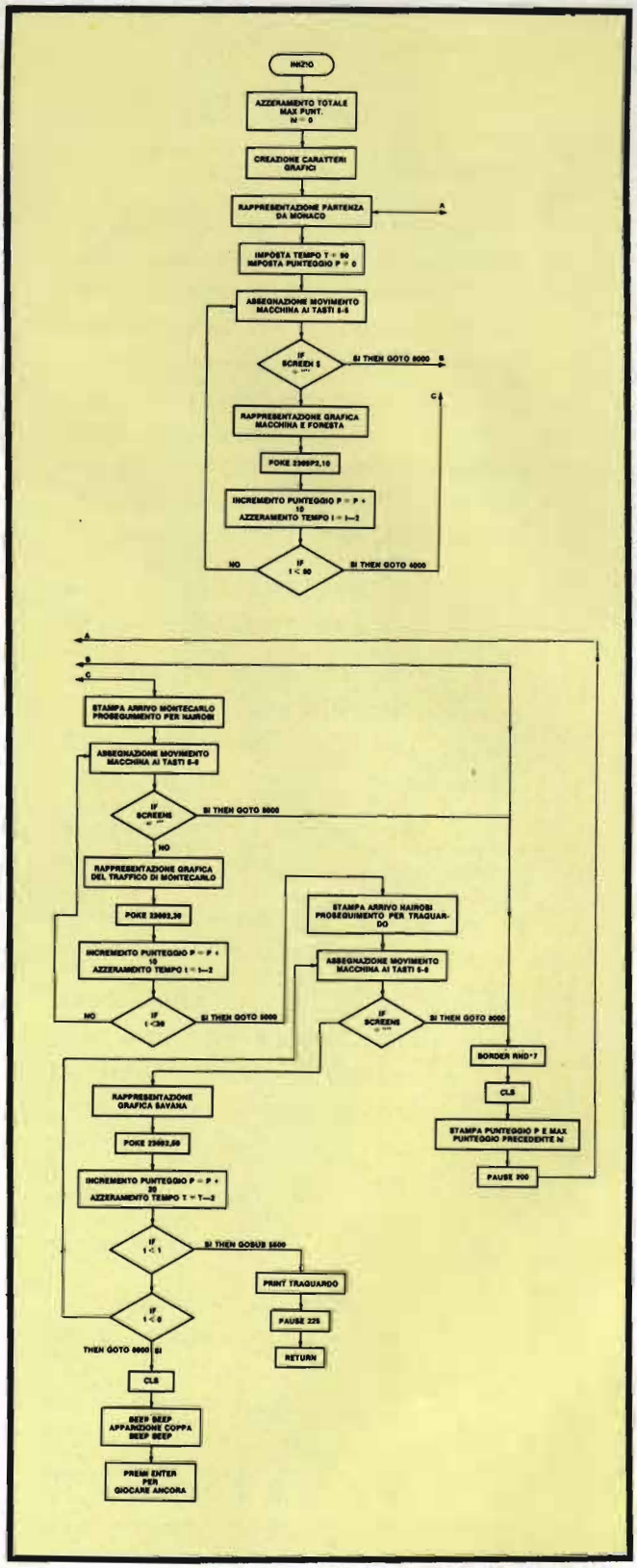


Figura 9. *Diagramma a blocchi del programma Rally Safari.*

2+2=APPLE

Due Riviste famose, specializzate, informatissime

BIT – PERSONAL SOFTWARE

Due volumi preziosi per chi vuole approfondire la conoscenza del suo computer

INTERFACCIAMENTO DELL'APPLE
196 pagine
Cod. **334B**
Lire 14.000

APPLE II Guida all'uso
390 pagine
Cod. **331P**
Lire 26.000

Una sola firma prestigiosa per chi si interessa di informatica e di elettronica

GRUPPO EDITORIALE JACKSON

Attenzione compilare per intero la cedola
ritagliare (o fotocopiare) e spedire in busta chiusa a:
GRUPPO EDITORIALE JACKSON
Divisione Libri
Via Rosellini, 12 - 20124 Milano

COUPON D'INFORMAZIONE

Desidero ricevere un numero omaggio di ☐ BIT - ☐ PERSONAL SOFTWARE insieme a maggiori informazioni sulle condizioni di abbonamento

INVIATEMI CONTRASSEGNO

n° copie	codice	Prezzo unitario	Prezzo totale
	334B	**L. 14.000**	
	331P	**L. 26.000**	
contributo fisso spese di spedizione			L. 2000
Totale			

Nome
Cognome
Via
Cap Città Prov.
Data Firma

Spazio riservato alle Aziende. Si richiede l'emissione di fattura

Partita I.V.A.

PERSONAL SOFTWARE

Music 4.2 per C 64

Per sonorizzare qualsiasi programma senza interromperne l'elaborazione

di *Mirko Gremes*

È ben noto che il CBM 64, grazie all'adozione di un apposito LSI, ha delle capacità sonore molto interessanti. Manipolando adeguatamente i registri di tale integrato, magari con delle routine in linguaggio macchina (per ovvie ragioni di velocità), si possono ottenere dei graziosi motivetti, o dei piacevoli effetti sonori, che inseriti in un qualsiasi programma, lo renderebbero meno monotono e senz'altro più pratico se la sonorizzazione non avenisse in modo casuale.

Music 4.2 non solo permette di riprodurre brani o effetti sonori, ma anche di farlo contemporaneamente ad un qualsiasi altro programma, senza cioè interromperlo o rallentarlo se non in modo praticamente impercettibile.

La routine ha le seguenti caratteristiche:

- possibilità di riprodurre, oltre agli effetti sonori, brani musicali anche polifonici e politonici fino a tre voci;
- possibilità di variare in qualsiasi momento ogni parametro di esecuzione (inviluppo, forma d'onda, volume, velocità), mediante dei codici di controllo;
- possibilità di ripetere parti dei blocchi dati senza doverli riscrivere (ritornello).

A tutto ciò si uniscono i vantaggi dell'uso del linguaggio macchina, che in questo caso sono: velocità di esecuzione anche molto elevate, perfetta contemporaneità di eventuali accordi.

Le note musicali vengono codificate mediante due numeri, di cui il primo indica l'altezza della nota (si veda il manuale ove è riportata la tabella relativa), e deve essere compresa tra "1" (Do # 0) e "89" (Fa 7), e il secondo, la durata della nota espressa in sessantaquattresimi.

Per variare i parametri d'esecuzione, sono disponibili i seguenti codici, che dovranno essere inseriti al posto della nota relativa alla voce per la quale si vuole ottenere il cambiamento:

255 Quando la routine incontra questo dato al posto di un codice nota, si predispone a leggere i 5 dati successivi, che inserirà nei registri del SID col seguente ordine:
- forma d'onda (16,32,64,128),
- duty cycle LO (0/255),
- duty cycle HI (0/15),
- attacco/decadimento (A+D★16),
- sostegno/rilascio (S+R★16).

254 Il dato successivo a questo, viene inserito nel registro del SID che controlla il volume.

253 Il dato seguente indica il ritardo che scandisce la lettura dei dati (1/64); 1 corrisponde alla velocità di esecuzione più elevata, 255 a quella più lenta.

252 Indica la fine del brano.

251 I tre dati successivi indicano rispettivamente quante volte ripetere una parte del blocco dati (1/15), e il numero di dati che devono essere ripetuti (H★256+L).

Non ci si deve comunque preoccupare, in quanto la codifica dei brani musicali avviene automaticamente usando il programma Music Editor presentato il mese scorso.

Descrizione della routine

È possibile suddividere la descrizione di questa routine, in tre parti principali:

1) ricerca e controllo di un brano musicale o effetto sonoro all'interno di un blocco dati;
2) lettura dei dati, loro elaborazione, controllo dei registri del SID;
3) meccanismo che permette di svolgere le operazioni del punto "2", contemporaneamente ad un qualsiasi programma.

Ricerca e controllo

Dall'indirizzo 40550 a 40658, è posta una routine che ricerca e controlla se esiste il blocco dati relativo al numero inserito nella locazione 87.

I dati delle note e i codici di controllo, vengono memorizzati a partire dalla locazione 40368 all'indietro: è possibile così aggiungere quanti dati si vogliono, con l'unico limite dato dallo spazio lasciato libero dal programma BASIC che ospita questa routine musicale.

Ovviamente per evitare deleterie sovrapposizioni, è necessario limitare la memoria usata dal BASIC al di sotto dell'ultimo dato usato da Music 4.2. Ciò non deve per niente impensierire, in quanto l'aggiustamen-

to dei puntatori di fine BASIC avviene automaticamente.
Questa routine di ricerca e controllo, a partire dal dato memorizzato nell'indirizzo più alto, preleva e controlla i dati tramite un contatore all'indietro. Ogni volta che viene incontrato il codice "252" (fine brano), viene decrementata la locazione 87; questo si ripete finché essa non vale 1.

Il contatore usato per il prelievo dati conterrà l'indirizzo dell'ultimo "252" trovato; l'indirizzo successivo (quello più in basso, perché stiamo contando all'indietro), sarà quello ove inizia il brano richiesto. Prima però di passare questo indirizzo alla fase successiva del programma, per evitare spiacevoli blocchi della macchina, viene controllato se effettivamente tale brano esiste. Infatti senza questo controllo non è possibile sapere se all'indirizzo trovato, inizia un brano, o se il codice "252" per ultimo incontrato si riferiva all'ultimo brano memorizzato.
Tale controllo consiste semplicemente nel ricercare il codice "252" di quest'ultimo brano. Se questo codice non viene trovato, vuol dire che il brano non esiste per niente, e si ha un ritorno al BASIC senza che nulla avvenga.
Riassumendo, si tenga presente che:
1) è possibile inserire quanti brani si vogliono, compatibilmente con la memoria lasciata libera abbassando i puntatori di fine BASIC;
2) non importa quale sia la lunghezza del brano (o effetto sonoro) e tantomeno il suo indirizzo di inizio;
3) se si richiede un brano inesistente, la routine se ne accorge e non accetta la richiesta;
4) per ascoltare un brano musicale, basta specificare il numero relativo nella locazione 87 e richiamare la routine con SYS 40550 (POKE 87,X: SYS 40550).
Vorrei a questo punto aprire una parentesi per spiegare, anche se in modo superficiale, il significato della frase "abbassare i puntatori di fine BASIC" o simili, tanto ovvie per gli "addetti ai lavori", quanto invece misteriose per i nuovi e inesperti utenti. Tale concetto non è specifico del 64, ma riferibile a molte macchine oggi in commercio.

Trascurando ora per semplicità il doppio indirizzamento di alcune aree di memoria del 64, possiamo dire che la distribuzione dei due tipi di memoria (RAM e ROM) è la seguente: da 0 a circa 41000 c'è la RAM, oltre e fino a 65535, c'è la ROM e i registri di controllo del VIC 2, del SID, I/O, ecc.
A noi interessa solo la prima parte, cioè la RAM, che viene usata in parte dal sistema operativo, e il rimanente dal BASIC. Due coppie di locazioni (43/44 e 55/56) indicano dove inizia e dove termina l'area di memoria destinata al BASIC. Quando per applicazioni particolari è necessaria ulteriore memoria RAM, modificando le locazioni di inizio e fine (puntatori), si possono creare degli spazi prima e/o dopo l'area BASIC. Se scrivessimo nella coppia di locazioni 55 e 56 il numero 30000, l'interprete BASIC verrebbe informato che deve limitare la memorizzazione di dati BASIC all'indirizzo 30000 anziché 40960. Viene creata così una fascia di 10960 byte disponibili e protetti dalla scrittura di dati BASIC. È ovvio che se scrivessimo a partire dall'indirizzo 30000 dei dati o dei programmi in linguaggio macchina senza limitare l'area BASIC, andremmo ad alterare il delicato formato di memorizzazione del BASIC con conseguente blocco della macchina.

Sono molti i casi in cui è necessaria della memoria RAM liberamente usufruibile, ad esempio per riprogrammare i caratteri, o per creare una memoria video ad alta risoluzione, per programmi in linguaggio macchina, ecc.

Funzionamento simultaneo con altri programmi

Vediamo ora come sia possibile ottenere il funzionamento simultaneo di due programmi, in modo da poter utilizzare il procedimento anche per programmi diversi da quello musicale descritto in questo articolo.
Come si sa, nel 64, un CIA (versione potenziata dei VIA) provvede a generare una richiesta di interruzione (IRQ) ogni sessantesimo di secondo. Questo serve per interrompere momentaneamente il programma principale e per svolgere, tra l'altro, due importanti compiti: uno è incrementare l'orologio interno (accessibile dal BASIC mediante le variabili TI e TI$), l'altro di scandire la tastiera e depositare gli eventuali dati letti nell'apposito buffer.
Ma vediamo l'esatta sequenza degli eventi:
se il flag "I" del registro "Status" (contenuto nel uP) è settato, la richiesta di interruzione viene accolta, il programma in corso viene interrotto, e vengono lette le locazioni 65534 e 65535 che contengono l'indirizzo nel quale iniziano le routine di interrupt.
Nel 64 questo indirizzo è 65352, dove troviamo una "Save routine", che serve a memorizzare tutti i registri della CPU nello STACK, e alla fine un'istruzione di salto indiretto alla locazione 59953 tramite le loca-

zioni (situate in RAM) 788 e 789; ed è proprio da quest'ultimo indirizzo che, in pratica, iniziano le vere e proprie routine di interrupt.
Quando viene chiamata "Music 4.2", modificando il contenuto delle locazioni 788 e 789, viene spostato l'indirizzo di inizio da 59953 a 40932 dove è memorizzata appunto Music 4.2.
Ogni sessantesimo di secondo, viene perciò chiamata questa routine, la quale dopo aver svolto le operazioni di lettura delle note e inserzione dei relativi valori nei registri del SID, rimanda la prosecuzione della elaborazione all'indirizzo originale (59953).
Da tutto questo discorso si può facilmente dedurre che in pratica la routine descritta non funziona contemporaneamente, bensì a brevi intervalli alternativamente al programma principale. Infatti la CPU non può che svolgere un compito per volta; vista però la velocità elevatissima con cui si svolge questa alternanza, l'apparenza è quella di simultaneità.

Music 4.2 usa in pagina zero 12 locazioni di memoria (87, da 163 a 171, 177, 178) normalmente usate per l'RS 232; non è perciò possibile usare il registratore, la stampante o il floppy, contemporaneamente ad essa. Questa è l'unica limitazione che impone la routine, dovuta alla difficoltà di reperire locazioni disponibili in pagina zero.
Anche se questa eventualità è piuttosto relativa rispetto alla sua utilità pratica, occorre precisare che, mentre Music 4.2 sta funzionando, è possibile anche listare o modificare un programma.

Lettura e decodifica dei dati

Senza scendere nei particolari di questa routine, che richiederebbe troppo spazio e cognizioni specifiche sul linguaggio macchina, si può riassumere il funzionamento nel seguente modo: ogni volta che si verifica una richiesta di interrupt, ossia ogni sessantesimo di secondo, e naturalmente se questa parte di routine è abilitata, viene decrementato un contatore; ogni volta che questo arriva a zero, viene effettuato un ciclo di lettura note, e il contatore viene ricaricato con il valore che determina il ritardo voluto. Questo indica in pratica quanti sessantesimi di secondo dura ogni sessantaquattresimo.
Possiamo paragonare questa parte di programma ad un generatore di "Clock" che in un orologio scandisce il tempo.

Ad ogni impulso (quando il contatore arriva a zero), vengono decrementati i tre contatori relativi alle voci; nel caso che uno o più di questi arrivino a zero, vengono letti due nuovi dati: il primo indica il numero della nuova nota, e il secondo, la durata espressa in sessantaquattresimi. Quest'ultimo valore viene caricato nel contatore relativo a quella voce. Il primo dato viene convertito tramite una tabella nei due valori LOFREQ. e HIFREQ. da inserire nei registri del SID. Questa tabella è posta nell'area di memoria che va da 40369 a 40549.

Se al posto di un codice nota viene letto un codice di controllo, i dati successivi vengono inseriti in altri registri come specificato in precedenza.
Tutto questo ciclo si ripete finché non viene incontrato il dato "252" che indica la fine del brano. Viene chiamata a questo punto una subroutine che ripristina le locazioni usate in pagina zero, nonché i puntatori di inizio delle routine di interrupt.

Caricamento della routine

Tra i vari sistemi che si possono usare per caricare dei dati in memoria, ne verrà descritto uno tra i più semplici che richiede l'uso dei floppy.
Su un disco si memorizzerà il programma Music Editor, che serve a trascrivere brani musicali in dati, i quali vengono archiviati, sempre su questo disco, sotto forma di file.
Ancora sullo stesso disco, si salverà il programma BASIC di supporto alla routine Music 4.2, il cui listato appare in queste pagine.
Questo permette di ascoltare e selezionare i brani, effetti o motivetti precedentemente archiviati, che verranno usati per sonorizzare un qualsiasi altro programma.
A questo punto si tolga dal drive questo disco e si inserisca quello su cui è memorizzato il programma da sonorizzare. Su questo ultimo disco verranno automaticamente riversati i dati della routine Music 4.2, i codici dei brani scelti dall'archivio e la tavola di conversione sottoforma, questa volta, di un unico file sequenziale.

All'inizio del programma ospitante si scriveranno le seguenti linee che permettono di trasferire questo file nella memoria del 64:

```
10 OPEN1, 8,2,"I0: MUSIC
   42,S,R": INPUT # 1,T:HI% =
   T/256: LO = T-(256 ★ HI%)
20 POKE51,LO:POKE55,
   LO:POKE56, HI%:POKE52,
   HI%: AA = 40955
30 INPUT # 1,T:IF T < 0 THEN
   CLOSE 1:GOTO 40
35 POKE AA,T:AA=AA-1:
   GOTO 30
40 REM SEGUE
   PROGRAMMA
```

Come già detto, ogni qualvolta si desideri far eseguire un brano, basta scrivere l'istruzione POKE 87,X : SYS 40550, dove X è il numero del brano che si vuole ascoltare.
Il programma BASIC ospitante la routine Music 4.2, per la sua semplicità, non merita commenti, resterà all'operatore l'accortezza di inserire, a partire dalla linea 250, i nomi dei brani inseriti nel file. ■

Music 4.2 per C 64

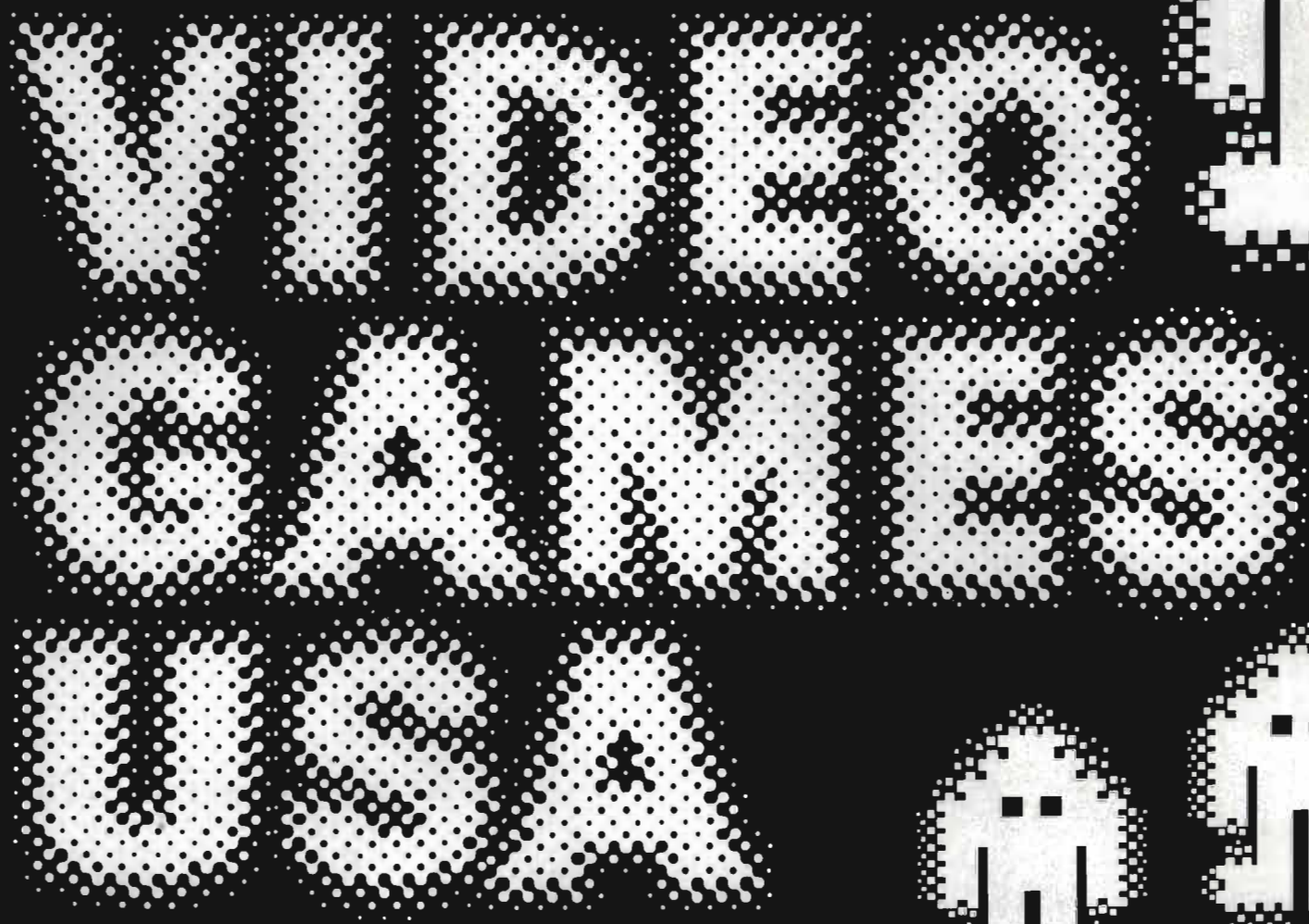
Figura 1. *Il listato Music 4.2.*

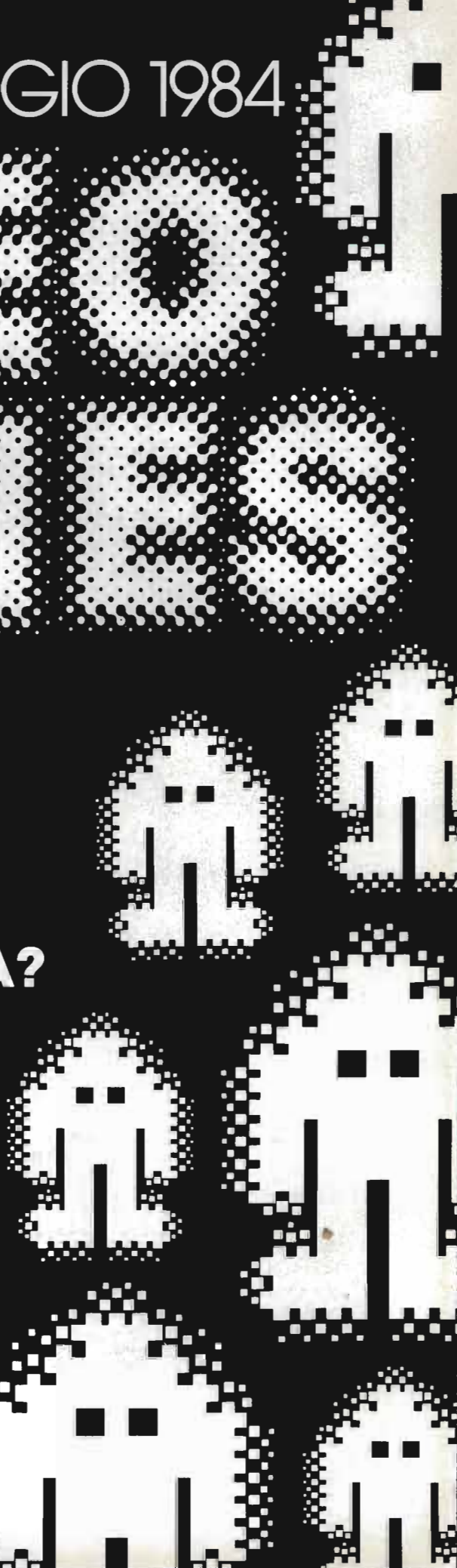
```
1 rem ***************
2 rem *             *
3 rem *  music 4.2   *
4 rem *             *
5 rem *    by       *
6 rem *             *
7 rem * mirko gremes *
8 rem *             *
9 rem ***************
10 poke56,36:poke52,36:poke51,0:poke55,0:dimmo$(30):aa=40368:poke166,0
15 printchr$(8)chr$(14)chr$(156)chr$(147):poke53280,1:poke53281,1:goto1550
17 rem---------------------------------
18 rem  calcola i valori delle note
19 rem---------------------------------
20 r=2↑(1/12):x=440/(29*256+50):fort=1to89:hz=(r↑t)*16.35
30 hh=int(hz/x):hi%=hh/256:lo%=hh-(hi%*256):poke40369+t,lo%:poke40459+t,hi%:next
40 fort=40550to40955:readd:poket,d:next:fort=1to30:readno$(t):next
67 rem---------------------------------
68 rem menu principale
69 rem---------------------------------
70 print"■":printll$:printtab(15)"Menu brani":printll$:print
75 ca=1:fort=0to14:fory=0to1
85 printtab(y*20)"["chr$(ca+63)"]";:printtab(y*20+4)no$(ca);
90 ca=ca+15:next:print:ca=ca-29:next:print
95 printll$:print:printtab(5)"'f1' fine"tab(26)"■sceli■"
97 rem---------------------------------
98 rem dati prog. l.m.
99 rem---------------------------------
100 data216,120,169,234,141,21,3,169,49,141,20,3,88,24,169,0
105 data141,1,212,141,8,212,141,15,212,169,157,133,166,169,176,133
110 data165,165,165,72,165,166,72,160,0,177,165,32,238,159,164,165
115 data196,55,208,15,164,166,196,56,208,9,104,104,169,0,133,165
120 data133,166,96,201,252,208,224,198,87,208,29,104,133,166,104,133
125 data165,162,1,134,164,134,171,134,178,134,168,134,167,169,228,141
130 data20,3,169,159,141,21,3,96,104,104,76,135,158,216,165,167
135 data133,168,162,0,24,214,164,208,23,181,163,157,4,212,160,0
140 data177,165,32,238,159,201,0,208,10,177,165,32,238,159,149,164
145 data76,211,159,201,253,208,10,177,165,32,238,159,133,167,76,228
150 data158,201,255,208,42,177,165,32,238,159,149,163,177,165,32,238
155 data159,157,3,212,177,165,32,238,159,157,2,212,177,165,32,238
160 data159,157,5,212,177,165,32,238,159,157,6,212,76,228,158,201
165 data254,208,11,177,165,32,238,159,141,24,212,76,228,158,201,251
170 data208,69,165,169,41,240,208,13,177,165,32,238,159,105,240,24
175 data133,169,76,104,159,32,238,159,198,169,165,169,201,240,240,26
180 data160,0,177,165,32,238,159,72,177,165,32,238,159,101,165,133
185 data165,104,101,166,133,166,24,76,228,158,32,238,159,32,238,159
190 data169,1,133,169,76,228,158,201,252,208,37,169,0,141,4,212
195 data141,11,212,141,18,212,162,9,149,162,202,208,251,133,177,169
200 data50,133,178,169,234,141,21,3,169,49,141,20,3,76,49,234
205 data168,185,177,157,157,0,212,185,11,158,157,1,212,160,0,177
210 data165,32,238,159,149,164,169,1,21,163,157,4,212,138,105,7
215 data201,21,208,3,76,228,159,170,24,160,0,76,219,158,198,169
220 data208,3,76,211,158,76,49,234,72,198,165,165,165,201,255,208
225 data2,198,166,104,24,96
247 rem---------------------------------
248 rem nomi dei brani
249 rem---------------------------------
250 data "brano 1","brano 2","brano3","brano 4","brano 5"
260 data,,
270 data,,,
280 data,,,
```

Seguito listato Music 4.2.

```
285 data,,,
290 data,,,
295 data,,,,,
500 rem--------------------------------
501 rem  geting
502 rem--------------------------------
503 ifpeek(166)<>0then503
505 wait198,255,0:getf$
508 if f$="■"then 850
510 nr=asc(f$)-63:ifnr<1ornr>30then505
511 ifno$(nr)=""then505
513 ifnr<16thenq=nr*40+160+1024+4:gosub800
516 ifnr>15thenq=(nr-15)*40+160+1024+24:gosub800
518 ifno$(nr)=""then505
520 gosub950:print"■'E' per escludere e 'I' per inserire"
525 fort=0to90:getf$:if f$<>""then540
530 next:poke1945,197:poke1965,73:fort=0to90:getf$
535 if f$<>""then540
537 next:poke1945,69:poke1965,201:goto525
540 if f$="e"thenpoke1945,197:poke1965,73:cb=cb-1:aa=pp:goto70
550 if f$="i"thenpoke1965,201:poke1945,69:goto70
560 goto525
800 rem--------------------------------
801 rem reverse marker
802 rem--------------------------------
810 if q<2024thenfort=0to14:m=peek(f+q):poke(f+q),mor128:next:return
847 rem--------------------------------
848 rem crea il file
849 rem--------------------------------
850 print"■"ll$:print:print"         Inserisci il disco con il"
860 print:print"Programma basic ospitante."
870 print:printll$"■■"
880 printtab(5)"Premi un tasto quando hai finito"
885 print"■■"ll$"■"
890 wait198,255,0:poke198,0
900 open1,8,2,"i0:music 42,s,w":print#1,aa-1:t=40955
910 d%=peek(t):ift=aathenprint#1,-1:close1:printchr$(147):end
915 print#1,d%:t=t-1:goto910
947 rem--------------------------------
948 rem  legqi file brani
949 rem--------------------------------
950 open1,8,2,"i0:"+no$(nr)+",s,r":pp=aa:cb=cb+1
952 input#1,f:pokeaa,f:ift=252thenaa=aa-1:goto960
955 aa=aa-1:goto952
960 close1:fort=163to177:poke f,0:next:poke87,cb:sys40550:return
1547 rem--------------------------------
1548 rem  intestazione
1549 rem--------------------------------
1550 fort=0to38:ll$=ll$+"-":next
1560 fort=0to2:printll$:next:print
1565 print"-- Questo programma serve per creare --":print
1570 print"-- un file contenente i dati di -------":print
1575 print"------------ MUSIC 4.2 ---------------":print
1580 print"----- E' possibile ascoltare i dati----":print
1590 print"-- e scegliere i brani che si vogliono-":print
1600 print"-- inserire nel file. -----------------":print
1605 fort=0to4:printll$:next
1610 print"----------------------ATTENDI----------":printll$:goto 20
```

MILANO 22-26 MAGGIO 1984





PERCHÈ UNA NUOVA DATA?

Per una ragione più che valida: VIDEO GAMES USA entra a far parte di BIT USA, la prestigiosa mostra di home e personal computer americani. E l'edizione '84, arricchita dalla presenza dei videogiochi, sarà più interessante che mai!

NON DIMENTICATE DUNQUE di visitare la sezione Videogiochi di BIT USA 84, dal 22 al 26 maggio, presso il Centro Commerciale Americano.

CENTRO COMMERCIALE AMERICANO
Via Gattamelata 5, 20149 Milano
Tel. (02) 46.96.451 Telex 330208 USIMC-I

La mostra è realizzata in collaborazione con le riviste del **Gruppo Editoriale Jackson.**

PERSONAL SOFTWARE

Dal BASIC al Pascal

— Parte quinta —

Uno schema di programma e un esempio

a cura della *Redazione*

Un programma scritto in Pascal ha una struttura ben definita (come potreste indovinare basandovi su quello che sapete degli scopi e della filosofia di Wirth). L'ordine deve essere seguito esattamente, ma si può omettere qualunque parte se non serve. Ciò che segue è uno schema:

```
PROGRAM NOME (INPUT,OUTPUT,FILE1,FILE2);
LABEL

CONST
  PI=3.141592653;
TYPE
  GIORNODELLASETTIMANA=(LUN,MAR,MER,
                        GIO,VEN,SAB,DOM);
VAR
  GIORNO:GIORNODELLASETTIMANA;
  LAVORATIVI:[LUN..VEN];
  N:INTEGER;
  H,L,O: BOOLEAN;
  FILETTO:ARRAY[1..3,1..3] OF INTEGER;

PROCEDURE NOMEPROC (LISTA PARAMETRI);
  LABEL
  CONST
  TYPE
  VAR
  BEGIN

  END;

BEGIN

  NOMEPROC1;
  NOMEPROC2;
  WRITELN (RISULTATI);
  NOMEPROC3 (PARAMETRI);
  VARIABILE:=FUNCTION (PARAMETRI);

END.
```

Non posso più nascondervi la verità. Il Pascal ha un'istruzione GOTO. Talvolta è più strutturato usare un GOTO per abortire un ciclo quando si scopre un errore, che evitarne l'uso. Parleremo di questo un po' più avanti. Ci sono molte cose di cui dobbiamo parlare, che riguardano la struttura di un programma. Prima di tutto, gli spazi sono ignorati dal compilatore e quindi possono essere usati liberamente per separare le parole, permettendo una più facile lettura. Gli spazi, naturalmente, non possono essere usati in mezzo agli identificatori o comandi. Le righe in bianco sono anch'esse ignorate dal Pascal, e possono essere impiegate per separare gruppi funzionali di istruzioni del programma. I commenti possono trovarsi ovunque sia permesso uno spazio, ma di solito non ha senso usarli in mezzo ad una riga dove c'è un'istruzione.

Le variabili dichiarate all'inizio del programma prima della prima procedura si chiamano *variabili globali*. I riferimenti a queste variabili sono validi ovunque nel programma e nelle procedure. Le variabili dichiarate all'interno di una procedura sono valide solo all'interno di quella procedura e si chiamano *variabili locali*. Un'eccezione alla regola delle variabili globali è che quando una variabile in una procedura è dichiarata con lo stesso nome di una variabile globale, essa diventa una variabile nuova e locale all'interno di quella procedura (a meno che non sia passata a quella procedura come parametro).

Una procedura può essere contenuta dentro un'altra procedura. Le sue variabili sono locali ad essa. Le variabili della procedura esterna le sono disponibili, così come lo sono le variabili globali. Alcuni sistemi non permettono una nidificazione delle procedure. Qualsiasi procedura può chiamare qualsiasi altra procedura che sia stata dichiarata in precedenza. Questa regola elimina i cosiddetti riferimenti in avanti. In verità, ciò è troppo semplificato. Può sorgere una condizione in cui due procedure si devono chiamare (in un programma ricorsivo). In questa situazione, il Pascal ha un modo in cui si può dichiarare parzialmente una procedura (si dà il suo nome e il suo elenco dei parametri) e poi anziché scrivere il resto, si usa FORWARD, per indicare che si dichiarerà dopo. Questo permette i riferimenti in avanti dicendo "Ho intenzione di dichiarare questa procedura più tardi". Potete vederlo come un modo furbo di aggirare i riferimenti in avanti. Wirth fece molto lavoro per strutturare il linguaggio in un modo tale da rendere più facile l'uso del compilatore. Una funzione ha la stessa struttura di una procedura. Le procedure, le funzioni e i programmi sono chiamati blocchi.

Un esempio di programma

Bene, siamo quasi arrivati alla fine dei preliminari. Ci sono naturalmente degli aspetti del Pascal di cui non abbiamo ancora parlato.

Tuttavia, arriva un momento nel quale è più interessante cominciare a programmare, ed è ora. Il programma 6.1 è un gioco chiamato "Reverse". Notate che REVERSE è stato scritto in modo che il numero di numeri nell'elenco è determinato dal valore di N. N

naturalmente non deve superare la dimensione del vettore A che è 20. Incominciamo dall'inizio.

```
(*$R*)

PROGRAM REVERSE (INPUT,OUTPUT);

CONST
  N=9;
```

Ecco il titolo del programma e il numero di numeri. Poichè N non cambierà durante il programma, perchè non farne una costante? Ora, le variabili che ci servono. Il vettore per tenere i numeri è probabilmente la prima che ci viene in mente. Perchè non lo chiamiamo LISTA?

Ci sono due indici, J e K, e per ora, non cambiano i loro nomi. Poi c'è R per l'introduzione del giocatore di quanti numeri sono da rovesciare. Perchè non chiamarlo QUANTI? La variabile Z contiene uno dei numeri del vettore mentre vengono scambiati. TEMP forse è più descrittivo di Z. Il giocatore risponde ad una domanda che gli chiede se vuole giocare un'altra partita o no. Come primo tentativo, usiamo una variabile CHAR RISPOSTA. La variabile T nel programma BASIC conta il numero di mosse o tentativi che occorrono al giocatore per mettere in ordine i numeri. TENTATIVI andrebbe bene. Sospetto che più tardi vorremo usare una variabile BOOLEAN, ma potremo aggiungerla quando ci arriveremo. Ora possiamo scrivere la sezione VAR del programma.

```
VAR
  LISTA:ARRAY[1..20] OF INTEGER; (* LISTA DEI NUMERI *)
  J,K,TEMP:INTEGER;              (* INDICI DI CICLO E DI VETTORE *)
  QUANTI:INTEGER;                (* DA ROVESCIARE *)
  TENTATIVI:INTEGER;
  RISPOSTA:CHAR;
```

Questo è un buon tentativo per tutte le variabili da usare. Per un approccio un po' informale a questo problema, usiamo le sezioni del programma BASIC separate da commenti per le procedure principali da scrivere in Pascal, almeno per iniziare. C'è un problema che bisogna risolvere, allora tanto vale affrontarlo per primo. La maggior parte dei sistemi Pascal non ha una funzione RANDOM, allora dovremo predisporla in modo che fornisca un numero a caso fra 1 e N. Senza approfondire troppo l'argomento, il seguente è un generatore di numeri a caso. Più tardi nel programma, daremo al giocatore la possibilità di "randomizzare" il numero "di base" della funzione.

```
FUNCTION RANDOM(K:INTEGER):INTEGER;

(* UN NUMERO A CASO TRA 1 E K *)

BEGIN
  RAND:=RAND/17;
  RAND:=RAND*19;
  IF RAND>16384 THEN RAND:= RAND-7327;
  RANDOM:=TRUNC(RAND) MOD K +1;
END; (* RANDOM *)
```

Siccome dovremo rendere accessibile la variabile RAND dal nostro programma principale per darle un valore, RAND deve essere una variabile globale e dunque l'aggiungeremo alla nostra sezione VAR.

```
RAND:REAL;
```

La funzione MOD nel Pascal fornisce il resto di una divisione tra interi. RAND MOD N dunque dà un numero fra 0 e N-1. Sommando 1 a questo numero si ottiene il valore fra 1 e N che ci vuole. Una divisione per 17 e una moltiplicazione per 19 produce una serie di numeri casuali con un lungo periodo di ripetizione. Ora che abbiamo messo in funzione RANDOM, la si può usare per generare la lista d'inizio.

```
PROCEDURE CREALISTA;

BEGIN
  FOR K:=1 TO N DO
    LISTA[K]:=K;
  FOR K:= N DOWNTO 2 DO
    BEGIN
      J:=RANDOM(K);
      TEMP:=LISTA[K];
      LISTA[K]:=LISTA[JJ];
      LISTA[JJ]:=TEMP;
    END; (* FOR *)
END; (* CREALISTA *)
```

Vedete come ci siamo liberati del GOTO implicato nel programma BASIC alla linea 250? Questa procedura dapprima crea un elenco 1 ... N. Usa poi un ciclo FOR per assegnare i numeri casuali. Il processo è identico a quello del programma BASIC, compresi i cicli. La THEN 230 è stata eliminata dall'uso di un ciclo REPEAT UNTIL.

Predisponiamo un altro spezzone di programma per fare il rovesciamento dei numeri. Perchè non passiamo il numero di numeri da rovesciare alla procedura come parametro? Anche qui, il processo sarà uguale a quello del programma BASIC.

```
PROCEDURE RIORDINA (NUMERO:INTEGER);

BEGIN
  FOR J:=1 TO NUMERO DIV 2 DO
  BEGIN
    TEMP:=LISTA[J];
    LISTA[J]:=LISTA[NUMERO-J+1];
    LISTA[NUMERO-J+1]:=TEMP;
  END; (* FOR *)
  STAMPALISTA;
END; (* RIORDINA *)
```

Notate che la NUMERO DIV 2 automaticamente produce un risultato intero uguale alla INT(R/2) del programma BASIC. In due posti, il programma BASIC ha GOSUB 610. Questa è la subroutine che stampa l'elenco dei numeri. È abbastanza semplice, ma siccome viene chiamata da due posti nel programma, perchè non metterla nel programma come procedura? In verità, dovrebbe essere chiamata da RIORDINA, quindi dovrebbe procedere RIORDINA, e la

penultima istruzione di RIORDINA dovrebbe essere STAMPA LISTA;.

```
PROCEDURE STAMPALISTA;

BEGIN
  FOR J:=1 TO N DO
    WRITE (LISTA[J]:2);
    WRITELN
  END; (* STAMPALISTA *)
```

Abbastanza semplice, stampa la lista e anche una riga vuota dopo di esso. Poichè lo stampare le istruzioni del gioco è solo una serie di istruzioni WRITE che possono o no essere necessarie, ha senso scrivere una procedura che si chiami ISTRUZIONI.

```
PROCEDURE ISTRUZIONI;
BEGIN
  WRITELN;
  WRITELN('QUESTO E'' IL GIOCO DEL "REVERSE"');
  WRITELN('PER VINCERE DEVI RIORDINARE LA');
  WRITELN('LISTA DEI NUMERI DA 1 A ',N:2);
  WRITELN('IN ORDINE CRESCENTE DA SINISTRA');
  WRITELN('A DESTRA. PER MUOVERE, DEVI DIRMI');
  WRITELN('QUANTI NUMERI (DALLA SINISTRA)');
  WRITELN('DEVO INVERTIRE.');
  WRITELN;
  WRITELN('SE LA LISTA E'':');
  WRITELN;
  WRITELN('2 3 4 5 1 6 7 8 9');
  WRITELN;
  WRITELN('E NE VUOI INVERTIRE 4, IL RISULTATO ');
  WRITELN('SARA'':');
  WRITELN;
  WRITELN('5 4 3 2 1 6 7 8 9');
  WRITELN;
  WRITELN('E SE ORA NE INVERTI 5 HAI VINTO!');
  WRITELN;
  WRITELN('1 2 3 4 5 6 7 8 9');
  WRITELN;
  WRITELN('PER FINIRE, BATTI 0.');
END;
```

Certamente qui non c'è niente di nuovo. È una traduzione diretta della versione BASIC. Non ci rimane molto da fare a parte una procedura GUARDASEHOVINTO. Un ciclo normale FOR funzionerebbe esattamente come nel programma BASIC. Comunque, qui abbiamo una buona opportunità di usare un ciclo WHILE-DO e quindi uscire non appena scopriamo una nonvincita. Questo ridurrà il tempo di elaborazione del programma (sebbene sia una questione accademica discutere della necessità di tale riduzione).

```
PROCEDURE GUARDASEHOVINTO;

BEGIN
  J:=1;
  VINTO:=TRUE;

  WHILE VINTO AND (J<=N) DO

    BEGIN
      IF LISTA [J] <> J THEN VINTO:=FALSE;
      J:=J+1;
    END; (* WHILE *)
END; (* VEDE *)


(* PROGRAMMA PRINCIPALE *)
```

Ora sarà necessario tornare alle dichiarazioni delle VAR globali e aggiungere:

```
VINTO:BOOLEAN;
```

Ora rimane solo il programma principale che chiamerà queste subroutine. Ormai, forse avete notato che il programma non ha alcun GOTO ed è quindi abbastanza ben strutturato così com'è. Alcuni programmatori del Pascal potrebbero pensare che il programma principale che segue sia troppo lungo. Il mio pensiero è che non esiste un altro spezzone abbastanza grande che si possa togliere per farne una procedura a sé. Il programma principale fa partire il tutto e poi rimane in un ciclo esterno per ogni partita giocata e in un ciclo interno per le mosse di una partita. Ho messo alcune spiegazioni nel programma nella forma di commenti per associarle alle varie istruzioni del programma. Come minimo, però, una END dovrebbe sempre avere un commento che indichi di che cosa è la fine. Questo aiuta anche il programmatore. In un programma più complesso dove ci sono tanti cicli interminati che continuano, spesso succede che molti di questi terminano allo stesso punto, e un po' di negligenza può causare o troppo pochi o troppi END. Un commento dopo ciascuno aiuta a tenere le cose in ordine anche per il programmatore. Il programma 6.2 è quello completo con la parte principale compresa.

```
(*$B+*)

PROGRAM REVERSE (INPUT,OUTPUT);

CONST
  N=9;

VAR
  LISTA:ARRAY[1..20] OF INTEGER; (* LISTA DEI NUMERI *)
  J,K,TEMP:INTEGER;              (* INDICI DI CICLO E DI VETTORE *)
  QUANTI:INTEGER;                (* DA ROVESCIARE *)
  TENTATIVI:INTEGER;
  RISPOSTA:CHAR;
  RAND:REAL;
  VINTO:BOOLEAN;

FUNCTION RANDOM(K:INTEGER):INTEGER;

(* UN NUMERO A CASO TRA 1 E K *)

BEGIN
  RAND:=RAND/17;
  RAND:=RAND*19;
  IF RAND>16384 THEN RAND:= RAND-7327;
  RANDOM:=TRUNC(RAND) MOD K +1;
END; (* RANDOM *)

PROCEDURE CREALISTA;

BEGIN
  FOR K:=1 TO N DO
    LISTA[K]:=K;
  FOR K:= N DOWNTO 2 DO
    BEGIN
      J:=RANDOM(K);
      TEMP:=LISTA[K];
      LISTA[K]:=LISTA[J];
      LISTA[J]:=TEMP;
    END; (* FOR *)
END; (* CREALISTA *)

PROCEDURE STAMPALISTA;

BEGIN
  FOR J:=1 TO N DO
    WRITE (LISTA[J]:2);
    WRITELN
  END; (* STAMPALISTA *)

PROCEDURE RIORDINA (NUMERO:INTEGER);

BEGIN
  FOR J:=1 TO NUMERO DIV 2 DO
  BEGIN
    TEMP:=LISTA[J];
    LISTA[J]:=LISTA[NUMERO-J+1];
    LISTA[NUMERO-J+1]:=TEMP;
  END; (* FOR *)
  STAMPALISTA;
END; (* RIORDINA *)

PROCEDURE ISTRUZIONI;

BEGIN
  WRITELN;
  WRITELN('QUESTO E'' IL GIOCO DEL "REVERSE"');
  WRITELN('PER VINCERE DEVI RIORDINARE LA');
  WRITELN('LISTA DEI NUMERI DA 1 A ',N:2);
  WRITELN('IN ORDINE CRESCENTE DA SINISTRA');
  WRITELN('A DESTRA. PER MUOVERE, DEVI DIRMI');
  WRITELN('QUANTI NUMERI (DALLA SINISTRA)');
  WRITELN('DEVO INVERTIRE.');
  WRITELN;
  WRITELN('SE LA LISTA E'':');
  WRITELN;
  WRITELN('2 3 4 5 1 6 7 8 9');
  WRITELN;
  WRITELN('E NE VUOI INVERTIRE 4, IL RISULTATO');
```

PERSONAL SOFTWARE

Dal BASIC al Pascal

```
    WRITELN('SARA'' ');
    WRITELN;
    WRITELN( 5 4 3 2 1 6 7 8 9 );
    WRITELN;
    WRITELN('E SE ORA NE INVERTI 5 HAI VINTO' );
    WRITELN;
    WRITELN('1 2 3 4 5 6 7 8 9');
    WRITELN;
    WRITELN('PER FINIRE, BATTI 0.');
END;

PROCEDURE GUARDASEHOVINTO;
BEGIN
  J:=1;
  VINTO:=TRUE;

  WHILE VINTO AND (J<=N) DO

    BEGIN
      IF LISTA [J] <> J THEN VINTO:=FALSE;
      J:=J+1;
    END; (* WHILE *)
END; (* VEDE *)

(* PROGRAMMA PRINCIPALE *)

BEGIN
  WRITELN;
  WRITELN('QUESTO E'' IL GIOCO DEL REVERSE. );
  WRITELN('DAMMI UN NUMERO TRA 1 E 10000.');
  READLN(RAND);
  WRITELN;
  WRITE('VUOI LE ISTRUZIONI? (S O N) ');
  READLN(RISPOSTA);
  WRITELN;
  IF RISPOSTA='S' THEN ISTRUZIONI;

  REPEAT
    WRITELN;
    CREALISTA;
    TENTATIVI:=0;
    WRITELN('PARTIAMO... LA LISTA E'':');
    WRITELN;
    STAMPALISTA;
    WRITELN;

      REPEAT
        WRITE('QUANTI NE INVERTO? ');
        READLN(QUANTI);

        IF QUANTI<0
          THEN WRITELN('NON POSSO PROPRIO FARLO!');
        IF QUANTI>N
          THEN
            BEGIN
              WRITELN('OOPS! TROPPI! ');
              WRITELN('NE POSSO INVERTIRE AL MASSIMO ',N:2);
            END;

        IF QUANTI IN [1..N] THEN
          BEGIN
            RIORDINA(QUANTI);
            TENTATIVI:=TENTATIVI+1;
            GUARDASEHOVINTO;
          END;
      UNTIL VINTO OR (QUANTI=0);

    IF VINTO THEN WRITELN
      ('HAI VINTO IN ',TENTATIVI:3,' TENTATIVI. );
    WRITELN;
    WRITE('VUOI GIOCARE ANCORA? (S O N) ');
    READLN(RISPOSTA);
    WRITELN;
  UNTIL RISPOSTA='N';
END. (* PROGRAMMA PRINCIPALE *)
```

Conclusione

Eccovi con il primo programma completo in Pascal, e l'abbiamo tradotto da un programma BASIC.

Sia il programma Pascal che quello BASIC sono stati elaborati esattamente come sono scritti qui. È possibile che nessuno dei due funzioni con i vostri particolari sistemi di BASIC e di Pascal. Ho dovuto cambiare la RND(1) nella scrittura di *BASIC Computer Game* a RND(0) per il mio BASIC particolare. La funzione RND nel BASIC non è molto standardizzata, e potreste avere bisogno di usare un altro argomento o una istruzione RANDOMIZE per avere i numeri casuali. Avevo una procedura RANDOM nel programma Pascal che funzionava in un sistema ma non in un altro. Quella data qui funziona in entrambi. ■

Gestione del video tramite PEEK e POKE

POKE, PEEK e lo schermo dell'Apple

di *Claudio Poma*

Leggendo gli interessanti articoli sull'intelligenza artificiale del N. 10-11 di **Personal Software** ho tentato di tradurli per il mio Apple IIe, ma mi sono trovato di fronte al problema dello schermo. Infatti l'organizzazione della memoria di schermo dell'Apple non segue una regolare matrice in relazione alla posizione del cursore sullo schermo, ma ha una organizzazione particolare che non favorisce di certo la gestione del video tramite PEEK e POKE.

Ma veniamo al problema: innanzitutto le locazioni della Pagina 1 di testo e grafica in bassa risoluzione vanno dalla posizione decimale 1024 alla 2047, ci sono quindi più locazioni di memoria di quelle necessarie alla gestione dello schermo, che, come vedremo, non vengono usate dallo schermo.

Altro problema è che il riempimento dello schermo non avviene partendo dalla prima casella in alto a sinistra per giungere all'ultima casella in basso a destra, come avviene in altri home computer, bensì lo schermo è diviso in tre fasce che vengono riempite a turno. Il programma n. 1, illustra questa situazione riempiendo lo schermo con dei caratteri "$".

Il programma n. 2 serve invece per iniziare a scoprire qualcosa sui byte in più prima citati, infatti con lo STEP40 della linea 20 si potrebbero prevedere degli $ incolonnati sulla prima colonna, mentre, una volta fatto girare il programmino, appaiono degli $ in vari punti dello schermo. Che cosa è successo? Sono i byte in più!! Per fortuna il loro inserimento durante il riempimento dello schermo non è casuale, ma segue una regola che, seppur un po' complicata, una volta trovata permette di indirizzare lo schermo dell'Apple con PEEK e POKE.

Il programma n. 3 permette di risolvere il problema: fattolo girare, si vedrà una $ che si sposta sullo schermo saltando le righe seguendo la divisione in fasce prima descritta, inoltre sul fondo a sinistra si vedrà un numero che indica la posizione decimale della memoria schermo occupata in quel momento dalla $. Fino al termine della terza riga, che poi è la 17ma dello schermo, non ci sono problemi, poi attenzione: dall'ultima casella della terza riga alla prima della quarta (che poi è la seconda dello schermo) c'è un "buco" di otto byte e la cosa si ripete all'inizio di ogni nuova riga della prima fascia.

Questa regolarità consente di ricavare una formula in base alla quale ho poi convertito il programma di inseguimento riportato a pag. 45 del n. 10-11 di **Personal Software** senza eccessivi problemi e che permette a chiunque di utilizzare la stessa routine per situazioni più complicate come alcune anche pubblicate sullo stesso numero di **Personal Software**; ma veniamo alla conversione.

Fino alla linea 100 non ci sono cambiamenti. Le 110-111 leggono il carattere dalla tastiera, cioè vedono qual'è l'ultimo tasto premuto. Le linee 120-135 muovono la ★ che è il giocatore, e precisamente S sù, X giù, C sinistra, V destra. Da 140 a 170 non ci sono differenze. L'unica differenza della linea 180 è che qui è permesso al giocatore di toccare i muri e si perde solo quando si è raggiunti dal computer, rappresentato da una X.

E ora veniamo al problema sollevato inizialmente. La linea 181 rimanda ad una subroutine che determina in quale fascia si trova il giocatore. La linea 520 tratta della terza fascia per generalizzare la soluzione anche se tale fascia non è toccata dal semplice gioco del programma. Determinata la fascia vengono assegnati a Q3 il valore della prima locazione di memoria schermo della fascia in questione, e a XX il valore del numero della linea di tale fascia occupata dal giocatore. Fatto questo si ritorna al programma dove alla linea 182 al valore di Q3 vengono aggiunti 8 punti moltiplicati per il numero della linea della fascia (XX) meno 1: questo serve per recuperare gli 8 byte nominati in precedenza, il −1 serve ad annullare aggiunte quando il giocatore si trovi sulla prima linea di ogni fascia in quanto il conteggio per la prima linea di ogni fascia avviene in modo regolare. La linea 183 conclude il conto del valore della locazione di memoria schermo occupata dal giocatore aggiungendo il valore della colonna (Y) e il valore delle locazioni occupate dalle linee precedenti e che non sono quelle poste in posizione superiore sullo schermo, ma quelle che precedono, in valore di posizione schermo, il punto occupato dal giocatore (il 40 rappresenta il numero di posizioni di ogni riga e il 3 è dovuto alle tre fasce in cui è suddiviso lo schermo e che fanno in modo che la seconda riga dello schermo sia, in effetti, la quarta nella memoria schermo e quindi maggiore di 3 volte 40 la posizione in memoria schermo rispetto alla prima riga della stessa fascia.

Le 184-187 fanno tornare il giocatore nella posizione precedente

Gestione del video tramite PEEK e POKE

quando la sua mossa lo ha portato sul muro. Il resto è identico al programma originale salvo un'aggiunta per il record.

Per poter quindi indirizzare lo schermo dell'Apple II tramite PEEK e POKE bisogna determinare la fascia e la riga della fascia occupate dal punto in questione (come da 500-520) e quindi calcolare il punto come dalle linee 182-183. Queste, nel programma, sono state tenute separate per comprendere meglio il loro significato, ma ora possiamo riassumerle in un'unica formula più veloce da trascrivere in ulteriori programmi:

$Q1 = Q3 + 8 \star (XX - 1) + Y + 40 \star ((XX - 1) \star 3)$

che tradotta sta per Valore locazione memoria schermo = VAL. LOC. MEM. SCH. del primo punto della fascia occupata + 8 punti per ogni riga superiore alla prima della fascia + colonna + 40 per 3 per il numero della linea di fascia occupata diminuita di uno.

La formula può essere ulteriormente semplificata raccogliendo il valore (XX−1). Fatto questo si avrà:

$Q1 = Q3 + Y + (XX - 1) \star (40 \star 3 + 8)$ e poi $Q1 = Q3 + Y + 128 \star (XX - 1)$.

Capito il discorso e pronta la formula non dovrebbero esserci problemi per costruire giochi di movimento più complessi: per esempio, questo semplice gioco può essere arricchito con pochi cambiamenti delle prime linee che facciano in modo di far ricoprire tutto lo schermo dal labirinto e il gioco funzionerebbe senza ulteriori cambiamenti, poi si potrebbe fare in modo che l'inseguitore ricostruisca il muro attraverso il quale passa, poi... ad ognuno lo sviluppare tutte le varianti ed espansioni che gli vengono suggerite dalla propria creatività. Arrivederci. ■

```
1  REM    ** PROGRAMMA N.1 **
10  HOME
20  FOR T = 1024 TO 2047
30  POKE T,164
40  NEXT
```

Listato 1. Il programma riempie lo schermo con $ evidenziando la divisione in fasce.

```
1  REM    ** PROGRAMMA N.2 **
10  HOME
20  FOR T = 1024 TO 2047 STEP 40
30  POKE T,164
40  FOR N = 1 TO 300: NEXT
50  NEXT
```

Listato 2. *Perché i $ non sono incolonnati?*

```
1  REM    ** PROGRAMMA N.3 **
10  HOME
20  FOR T = 1024 TO 2047
30  POKE T,164
40  VTAB 20: PRINT T
50  FOR N = 1 TO 300: NEXT
60  POKE T,160
70  NEXT
```

Listato 3. *Osservate bene il conteggio e l'organizzazione di schermo non avrà più segreti.*

```
 5  HOME
 7  SPEED= 255
10  PRINT "##################"
20  FOR C = 1 TO 8
30  PRINT "#    #    #    #"
40  NEXT
50  PRINT "##################"
55  VTAB 3: HTAB 25: PRINT "Record : "RE
60 A = 7
70 B = 14
80 X = 2
90 Y = 2
93 P = 0
95  SPEED= 150
100  VTAB X: HTAB Y: PRINT " ": VTAB A: HTAB B: PRINT " "
110 K =  PEEK ( - 16384)
111 A$ =  CHR$ (K)
120  IF  ASC (A$) = 214 THEN Y = Y + 1
125  IF  ASC (A$) = 195 THEN Y = Y - 1
130  IF  ASC (A$) = 216 THEN X = X + 1
135  IF  ASC (A$) = 211 THEN X = X - 1
140 H =  INT ( RND (1) * 2)
150  IF H = 0 THEN A = A + (X > A) - (X < A)
160  IF H = 1 THEN B = B + (Y > B) - (Y < B)
170  VTAB A: HTAB B: PRINT "X"
180  IF A = X AND B = Y THEN 220
181 NN = 163: GOSUB 500
182 Q1 = Q3 + 8 * (XX - 1)
183 Q2 = Q1 + Y + 40 * ((XX - 1) * 3)
184  IF  PEEK (Q2) = NN AND  ASC (A$) = 214 THEN Y = Y - 1
185  IF  PEEK (Q2) = NN AND  ASC (A$) = 195 THEN Y = Y + 1
186  IF  PEEK (Q2) = NN AND  ASC (A$) = 216 THEN X = X - 1
187  IF  PEEK (Q2) = NN AND  ASC (A$) = 211 THEN X = X + 1
190  VTAB X: HTAB Y: PRINT "*"
200 P = P + 1
210  GOTO 100
220  VTAB 15: PRINT "Punti : "P
225  IF P > RE THEN RE = P
230  PRINT : PRINT "Per continuare premi <RETURN> ";: INPUT Q$
240  GOTO 5
500  IF X =  > 1 AND X =  < 8 THEN Q3 = 1023:XX = X
510  IF X =  > 9 AND X =  < 16 THEN Q3 = 1063:XX = X - 8
520  IF X =  > 17 AND X =  < 24 THEN Q3 = 1103:XX = X - 16
530  RETURN
```

Listato 4. *Conversione per Apple del programma "Giochi di inseguimento" apparso sul numero 10-11 di* **Personal Software.**

PERSONAL SOFTWARE

Impariamo il linguaggio macchina con il VIC e il C 64

— Parte seconda —

Diamo uno sguardo all'interno del micro 6502

di *Alessandro Guida*

Il 6502 e la memoria

Nella prima puntata, abbiamo visto che i numeri vengono trattati secondo la base binaria. Abbiamo inoltre visto che il microprocessore può maneggiare solo numeri composti da otto bit, detti byte, e quindi compresi tra 0 e 255.

Questi numeri possono avere un duplice significato. Essere dei semplici dati numerici (ad esempio il numero di impulsi ricevuti lungo una linea) o essere interpretati come una istruzione, cioè un comando da eseguire.

In ogni caso il microprocessore ha bisogno di una memoria in cui conservare i dati e da cui prelevare le istruzioni da eseguire.

Osservate la figura 1. La memoria è fisicamente costituita da un gruppo di circuiti integrati in grado di mantenere delle informazioni in modo sia permanente (anche a computer spento) sia momentaneo, cioè finché il computer resta acceso o quegli stessi dati cambiati.

La memoria, nel suo insieme, è formata da un certo numero di locazioni, ognuna delle quali contiene un byte di informazione. Queste locazioni non possono essere lette o scritte tutte contemporaneamente, perciò è necessario un "indirizzamento" della memoria. Questo consiste nel fornire agli integrati che la compongono il numero della locazione che vogliamo leggere o scrivere.

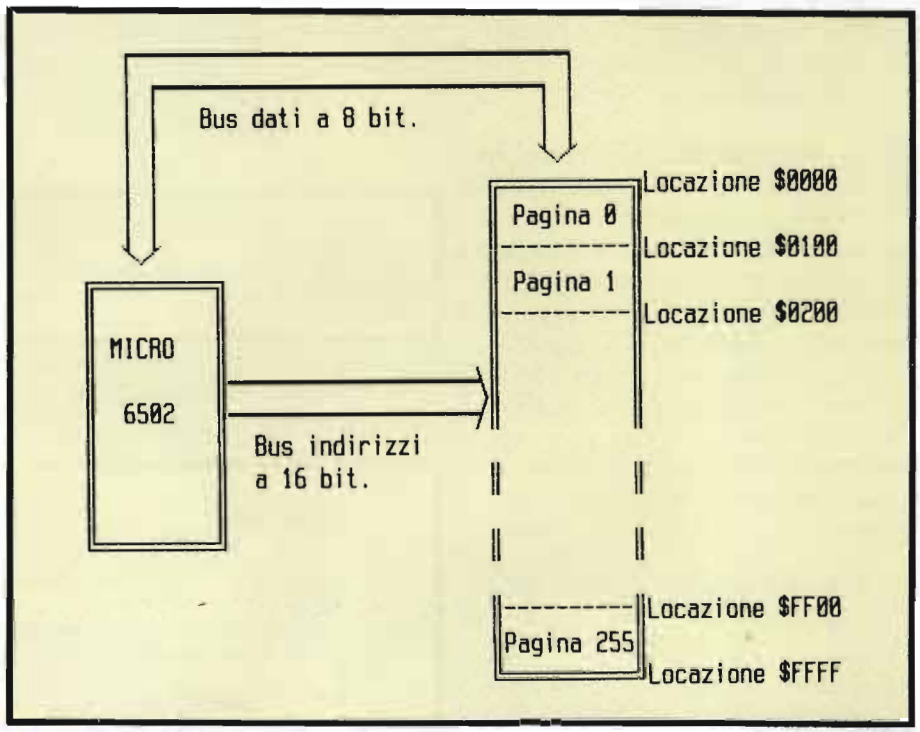


Figura 1. *Il 6502 e la memoria. Notare il bus indirizzi a 16 bit, che permette di indirizzare 65536 locazioni di memoria.*

Il 6502 è dotato di una linea di indirizzamento, chiamata *Bus degli indirizzi*, da 16 bit. È quindi in grado di indirizzare 2 ↑ 16 = 65536 locazioni di memoria.

Per comodità, l'intera memoria viene divisa in 256 pagine di 256 byte ciascuna (256x256=65536). In questa maniera, gli otto bit più significativi dell'indirizzo danno il numero di pagina, mentre i meno significativi la locazione all'interno della pagina. In realtà, questa suddivisione ha anche delle ragioni tecniche, dovute al fatto che all'interno del microprocessore gli indirizzi (come tutti gli altri dati) a 16 bit sono gestiti come due separati da 8 bit. Per questo motivo tutte le operazioni svolte all'interno di una pagina sono più veloci di quelle fatte tra pagine diverse.

Quest'ultime, infatti, obbligano il microprocessore ad ulteriori elaborazioni.

Una volta effettuato l'indirizzamento sarà possibile attraverso il bus dati, connesso alla porta dati del 6502 e all'ingresso/uscita dati della memoria, leggere il contenuto della locazione di memoria indicata o modificarla.

Il meccanismo di indirizzamento della memoria è lo stesso che usiamo normalmente in BASIC con le istruzioni POKE e PEEK. Infatti, anche in BASIC è necessario specificare il numero della locazione nella quale vogliamo leggere o scrivere.

Attenzione: l'avere detto che il microprocessore può accedere a 65536 locazioni di memoria non significa che è sempre possibile anche scrivervi. Infatti, generalmente, una parte della memoria del computer è

occupata dai programmi di sistema, che ne permettono il funzionamento in BASIC.
Questi programmi sono memorizzati su ROM (Read Only Memory = Memoria a sola lettura), che ha il pregio di non cancellarsi allo spegnimento del computer ma, per contro, non permette la scrittura. Vedremo comunque in seguito in quali zone di memoria sarà possibile scrivere i nostri programmi in linguaggio macchina.

I registri del 6502

Per rendere più facile la programmazione in linguaggio macchina è senz'altro importante conoscere, a grandi linee, cosa contiene il microprocessore 6502 (figura 2).
Abbiamo già visto che esiste una linea, attraverso la quale si muovono i dati, chiamata Bus Dati e che, per effettuare l'indirizzamento della memoria, esiste il Bus Indirizzi.
All'interno del microprocessore, a queste due linee sono collegati alcuni registri ed una ALU.

Con registri intendiamo una cella di memoria di otto bit.

Invece il termine ALU sta per Unità Aritmetico Logica.
Essa è rappresentata con una V per indicare la presenza di due entrate ed una sola uscita: il risultato della operazione svolta. Le operazioni che la ALU può svolgere sono molte (sommare, sottrarre, eseguire operazioni booleane, ecc.), ma quella effettivamente eseguita verrà stabilita dal comando ricevuto.

In genere, su un ingresso è presente un dato proveniente dalla memoria e sull'altro uno proveniente da un registro interno.
Vediamo, ora, i vari registri.
Il bus indirizzi fa capo ai registri PC (Program Counter).

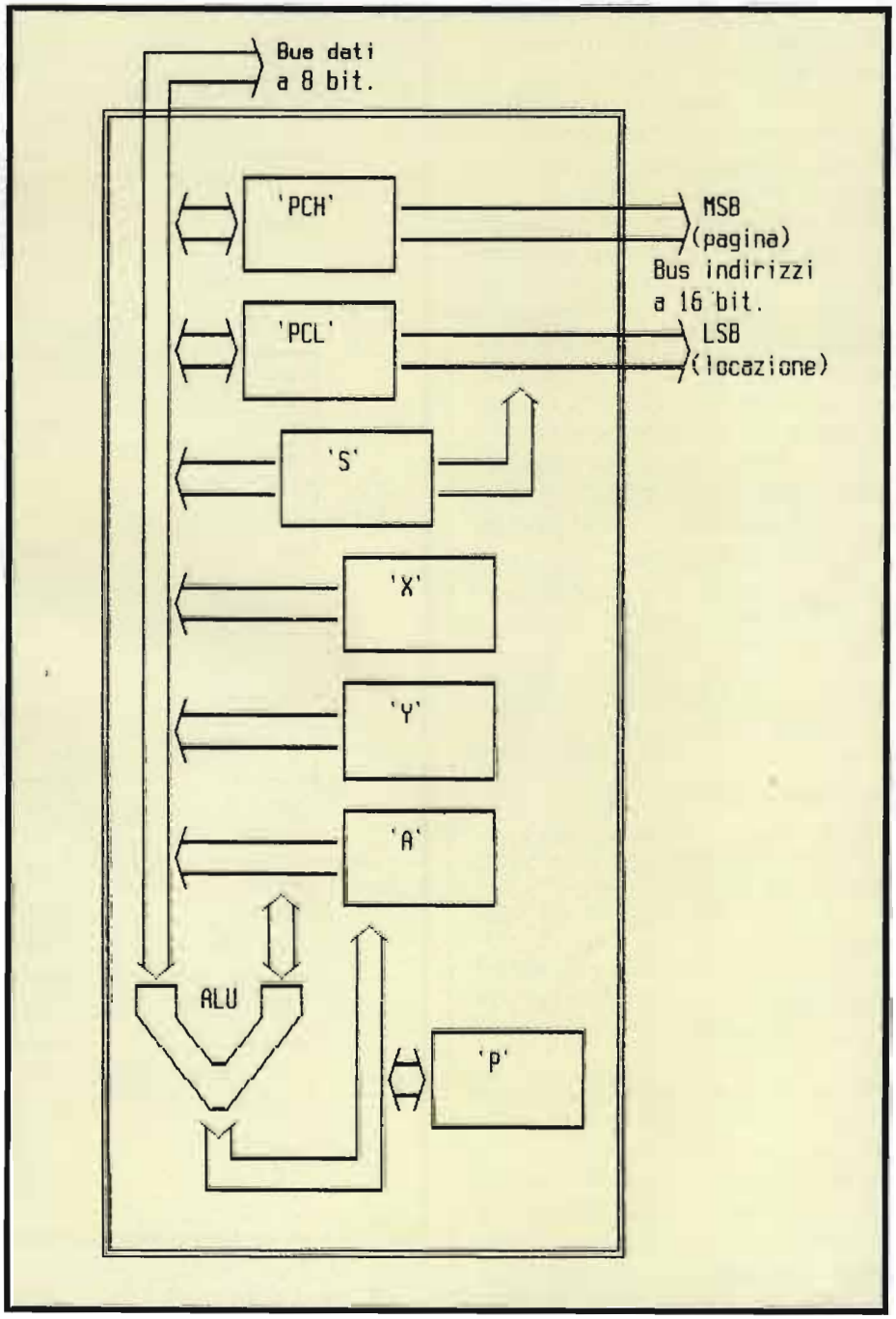


Figura 2. *Rappresentazione schematica del microprocessore 6502. In questa figura sono rappresentati (in maniera molto semplificata) i registri contenuti nel 6502.*

Poiché per gli indirizzi sono necessari 16 bit, mentre tutti i registri del 6502 sono a 8 bit, ne sono stati utilizzati 2. Uno ALTO (H) per il byte più significativo (MSB) e uno BASSO (L) per il byte meno significativo (LSB).

I due registri X e Y sono utilizzati in genere per memorizzare momentaneamente dei dati o come contatori per cicli simili al BASIC FOR-NEXT. Infatti, è possibile incrementarli o decrementarli con una sola istruzione. Sono inoltre utiliz-

zati come puntatori per eseguire degli indirizzamenti indiretti, che vedremo in seguito.
Il registro A, che utilizzeremo molto spesso, è quello utilizzato per le operazioni. In esso viene conservato il risultato proveniente dalla ALU.
P è un registro speciale, in quanto gli otto bit in esso contenuti sono dei FLAG. Ogni bit, quindi, ha un significato diverso. Tra i vari flag ci sono anche il C (Carry) e il V (Overflow) visti nella puntata precedente. Per ultimo vi è il registro S. Questo registro è un puntatore speciale.

Esso viene utilizzato per puntare una locazione di memoria in pagina 1. Le 256 locazioni presenti in pagina 1 sono dette Stack.

Lo Stack

Lo Stack è una zona di memoria gestita con sistema LIFO (Last In-First Out), ossia l'ultimo dato memorizzato è il primo ad essere letto (figura 3).
Per usare una similitudine, possiamo pensare allo Stack come ad una pila di piatti, nella quale ad ogni piatto corrisponde un dato. L'ultimo piatto poggiato sulla pila sarà il primo a poter essere tolto (sempre se non si vuole rovesciarli tutti per terra!).

In realtà il computer opera alla rovescia. I dati, infatti, cominciano ad essere immagazzinati a partire dalla cima dello Stack (locazione 255 in pagina 1). Vengono poi via via aggiunti uno sull'altro, decrementando ogni volta il registro S, puntatore allo Stack. Quando, invece, un dato deve essere prelevato, viene incrementato S e, subito dopo, viene letto il dato indicato dal puntatore S.
Le operazioni di immagazzinamento e prelievo sono dette PUSH e PULL.

Lo Stack e le relative operazioni di PUSH e PULL hanno una notevole importanza, poiché permettono una veloce memorizzazione mo-

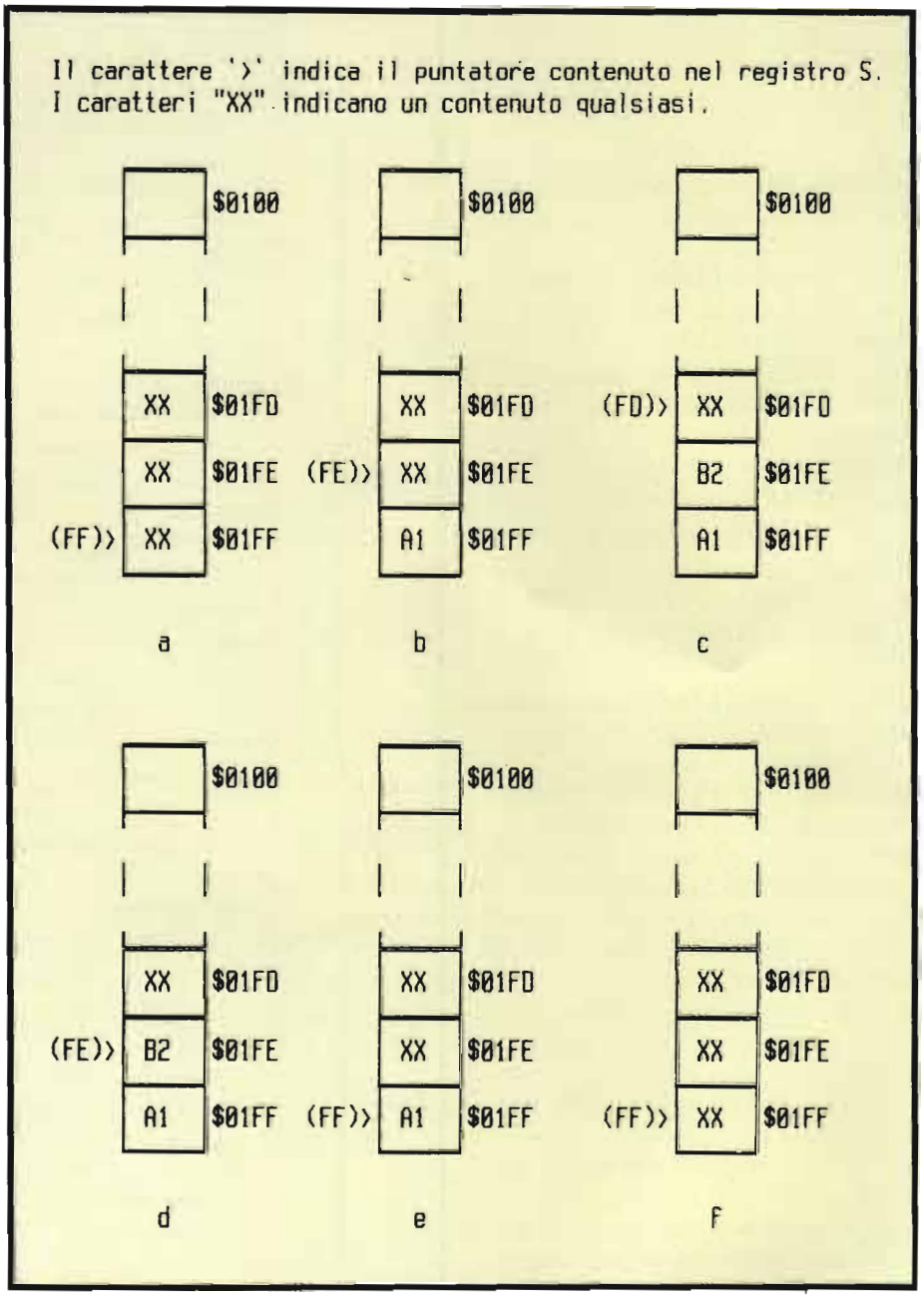


Figura 3. *Funzionamento dello stack.*
a) *La pag. 1 di memoria (stack) all'inizio. Il registro S punta l'inizio dell'area (FF).*
b) *Primo PUSH. Viene memorizzato il dato A1 nella locazione puntata da S, e S viene decrementato.*
c) *Secondo PUSH. Viene memorizzato il dato B2 e S, ancora, decrementato.*
d) *Primo PULL. Viene incrementato S e letto il contenuto della locazione puntata (B2).*
e) *Secondo PULL. Ancora incrementato S (FF), e viene letto A1.*
f) *La situazione dello stack alla fine. Anche se le due locazioni $01FF, $01FE contengono ancora i valori precedentemente memorizzati, questi non hanno più interesse, in quanto verranno sostituiti da nuovi dati nelle successive operazioni di PUSH.*

mentanea dei dati. Vedremo, infatti, che queste due operazioni occupano solo 1 byte di memoria. Scopriremo, anche, che lo Stack è interessato dai salti a subroutine, simili al GOSUB del BASIC.

I flag del 6502

Si è detto che il registro P contiene dei flag (figura 4). Ora vedremo quali sono, ma delle loro funzioni torneremo a parlare in seguito.
N - Negativo. Viene settato quando una operazione svolta dalla ALU dà risultato negativo.
V - Overflow. Quando è settato indica una situazione di overflow nell'ultima operazione effettuata.
B - Break. Indica che il microprocessore ha incontrato il comando BRK.

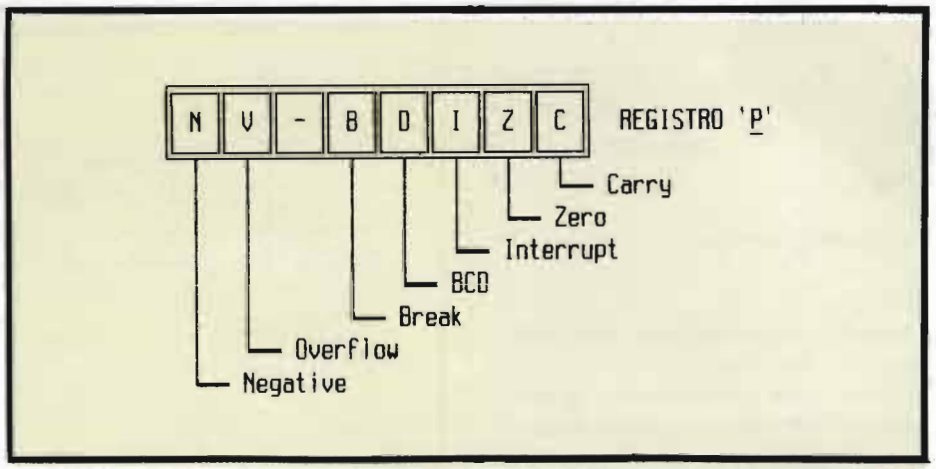


Figura 4. *I 7 flag contenuti nel registro P. A parte il bit 5, ad ogni bit del registro corrisponde un flag. Pertanto ogni flag può avere solo valore 1 (flag attivato) o 0 (disattivato).*

D - Decimale. Se si pone ad 1 questo flag, il 6502 opererà in modo decimale anziché esadecimale. Questo significa che, pur mantenendo la notazione binaria, quattro bit non potranno superare il 9 (1001), dopodiché si avrà il riporto sul quinto bit (esempio 10 = 1000).
Quindi 21, che in binario si scrive 00010101, col flag D attivato diventa 00100001. In pratica, il nibble più significativo anziché essere un multiplo di 16 lo è di 10.
I - Interrupt. Questo bit è legato all'intervento di circuiti esterni al 6502. Lo tratteremo a parte.
Z - Zero. Indica che il risultato di una operazione è uguale a zero.
C - Carry. Segnala che nell'eseguire una addizione o sottrazione vi è sta-

to un riporto oltre l'ottavo bit.

Ciclo di esecuzione delle istruzioni

Vista la struttura interna ed esterna del 6502, è interessante conoscere come, effettivamente, il microprocessore esegue un programma.
Il microprocessore, oltre i registri visti, contiene altri dispositivi che lo rendono operante. Il suo cuore è costituito da un oscillatore, detto clock, che fornisce circa un milione di impulsi al secondo, necessari a sincronizzare e far eseguire le varie istruzioni.
Schematicamente, quando il 6502 è in funzione avviene quanto segue, come illustrato nello schema a blocchi di figura 5.
1) Il clock fa avanzare il registro PC (Contatore di Programma) per puntare la prossima locazione di memoria.
2) Viene inviato alla memoria l'indirizzo e un segnale di READ (leggi) che le comunica di depositare il contenuto della locazione sul Bus Dati.
3) Il dato viene letto sul Bus Dati dal 6502 e messo in uno speciale registro del microprocessore, detto IR (Registro d'Istruzione).
4) Viene interpretato il dato conservato in IR ed eseguita l'operazione relativa.
5) Torna al punto 1.
In realtà, molte istruzioni si compongono di 2 o 3 byte consecutivi, per cui il ciclo descritto viene percorso 2 o 3 volte, prima che l'istruzione sia completa e quindi eseguita.

Naturalmente, questa è una visione molto semplificata di ciò che avviene a livello elettronico nella CPU (Central Processing Unit) 6502, ma è sufficiente per ciò che riguarda la programmazione.
La figura 6 contiene una tabella comparativa dei diversi sistemi di numerazione.
Anche per questa volta abbiamo terminato. Dalla prossima puntata cominceremo a programmare in linguaggio macchina. ■

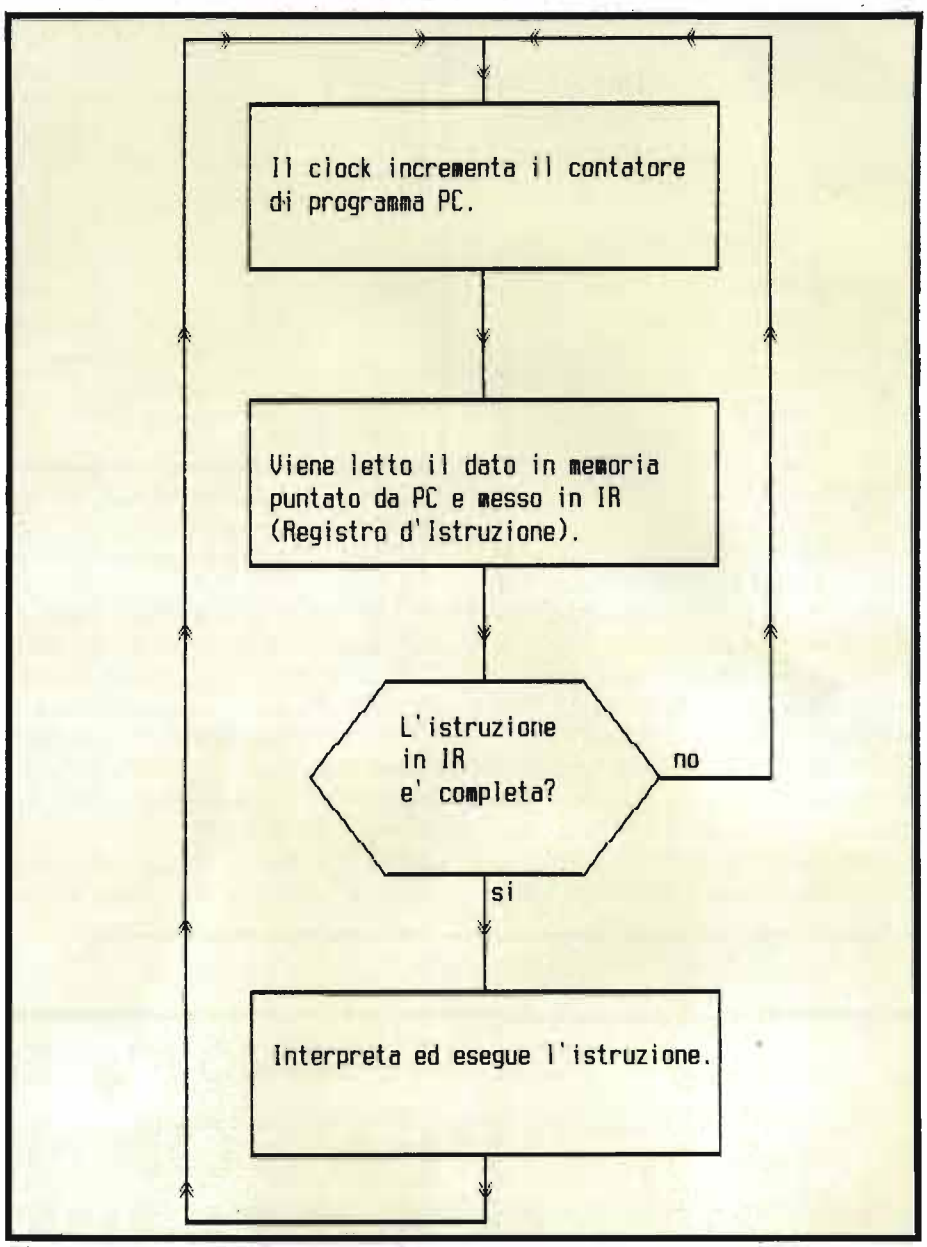


Figura 5. *Schema a blocchi semplificato del ciclo di esecuzione di una istruzione del microprocessore 6502.*

DEC.	ESADEC.	BINARIO (D=0)	BIN.DECIMALE (D=1)
00	00	0000 0000	0000 0000
01	01	0000 0001	0000 0001
..	..	....	....
09	09	0000 1001	0000 1001
10	0A	0000 1010	0001 0000
11	0B	0000 1011	0001 0001
..	..	....	....
15	0F	0000 1111	0001 0101
16	10	0001 0000	0001 0110
..	..	....	....
19	13	0001 0011	0001 1001
20	14	0001 0100	0010 0000
..	..	....	....
30	1E	0001 1110	0011 0000
..	..	....	....
50	32	0011 0010	0101 0000
..	..	....	....
90	5A	0101 1010	1001 0000
..	..	....	....
98	62	0110 0010	1001 1000
99	63	0110 0011	1001 1001
100	64	0110 0100	1 0000 0000

Figura 6. *Tabella comparativa delle numerazioni decimale, esadecimale, binaria, binario-decimale (detta BCD = Binary Coded Decimal).*

MILANO 22-26 MAGGIO 1984

Finalmente insieme.

Un'occasione da non perdere.
Quest'anno a Bit Usa,
la prestigiosa mostra di
Home e Personal Computer americani,
si affianca VIDEO GAME USA.
Un'edizione ancor più ricca e
interessante, Vi aspettiamo perciò
numerosi, dal 22 al 26 Maggio,
presso il
Centro Commerciale Americano.

USA 20

CENTRO COMMERCIALE AMERICANO

Via Gattamelata 5, 20149 Milano
Tel. (02) 46.96.451 Telex 330208 USIMC-I

PERSONAL SOFTWARE

Ordinamenti con lo ZX81

Un sort efficace, ma spesso dimenticato

di *Angelo De Santis*

Il termine "sort" indica il processo di ordinamento degli elementi di un vettore in base al loro valore numerico, in ordine crescente oppure decrescente.

È molto facile trovarsi di fronte alla necessità di ordinare grandi quantità di dati ed è quindi evidente l'utilità di algoritmi di sort veloci ed efficienti.

Molti algoritmi di sort richiedono tempi proporzionali a n ★ log n (dove n è il numero di elementi da ordinare) ed hanno anche la caratteristica di richiedere lunghi e complicati programmi.

In questo articolo viene presentato invece un metodo di sort abbastanza semplice che comporta solo qualche decina di istruzioni in BASIC e richiede un tempo di esecuzione semplicemente proporzionale ad n. Il programma è stato adattato allo ZX81 prendendo spunto da un articolo comparso sulla rivista Byte del Novembre 1983.

Questo tipo di sort si chiama "A calcolo di indirizzo" e il nome deriva dal fatto che esso riserva una memoria maggiore di quella strettamente necessaria, ed ogni elemento va ad essere posto in una locazione di memoria (indirizzo) calcolata a seconda del valore dell'elemento stesso.

Nel programma del listato 1 per ordinare n elementi l'occupazione di memoria è 3.36 ★ n per cui per ordinare 300 elementi viene richiesto ci-

```
   6 SLOW
   7 PRINT "QUANTI NUMERI?"
  10 INPUT N
  20 DIM R(N)
  21 REM *GENERAZIONE A CASO*
  30 FOR I=1 TO N
  50 LET R(I)=INT (65535*RND)-32
767
  60 PRINT I;"=";R(I)
  65 IF I>20 THEN SCROLL
  70 NEXT I
  80 REM *ORDINAMENTO*
  85 FAST
  90 LET I=2.36*N
 100 LET BP=I/65535
 110 DIM A(I+N)
 111 REM *MAIN LOOP*
 120 FOR X=1 TO N
 130 LET XA=X
 140 LET V=1+(32767+R(X))*BP
 150 IF A(V)=0 THEN GOTO 185
 160 IF R(A(V))<=R(XA) THEN GOTO
 175
 165 LET XB=XA
 170 LET XA=A(V)
 173 LET R(V)=XB
 175 LET V=V+1
 180 GOTO 150
 185 LET A(V)=XA
 190 NEXT X
 195 SLOW
 196 PRINT
 200 LET C=1
 210 FOR J=1 TO I+N
 220 IF A(J) THEN GOTO 240
 230 GOTO 270
 240 SCROLL
 250 PRINT C;".";R(A(J))
 260 LET C=C+1
 270 IF C<=N THEN NEXT J
 280 STOP
```

Listato 1. Programma di ordinamento. Le linee 80-190 contengono l'algoritmo vero e proprio e possono essere modificate in base alle necessità reali.

ca 1 Kbyte di memoria (senza considerare la memoria occupata dal programma).

Struttura del programma

7 Richiesta del numero N di elementi da ordinare.
20-70 Si calcolano a caso gli elementi da ordinare.
85-110 Riserva sufficiente spazio in memoria per gli elementi.
120-190 Colloca al proprio posto nel vettore A ciascun elemento.
195-280 Routine di stampa su video degli elementi ordinati.
In pratica la vera e propria routine di sort parte dalla riga 80 in poi.

Riferimenti bibliografici

a) Davidson D., "Address calculation, the forgotten sort", Byte Novembre 1983.
b) Flores I., "Computer sorting", Engewood Cliffs, NJ, Prentice-Hall, 1969.
c) Lorin H., "Sorting and sort systems", Reading, MA, Addison-Wesley, 1975. ■

```
1=3440
2=-4111
3=19397
4=12996
5=-8168
6=-22730
7=-751
8=9249
9=-27195
10=-7941
11=-5705
12=30921
13=25317
14=-1761
15=-964

1.-27195
2.-22730
3.-8168
4.-7941
5.-5705
6.-4111
7.-1761
8.-964
9.-751
10.3440
11.9249
12.12996
13.19397
14.25317
15.30921
```

Listato 2. *Esempio di ordinamento con 15 numeri. Il procedimento richiede meno di un secondo.*

PERSONAL SOFTWARE

La battaglia del lago ghiacciato

— Parte prima —

Da un film famoso un gioco di simulazione per PET CBM

di *Umberto Giovanni Barzaghi*

Prologo

A causa della lunghezza dell'articolo siamo costretti a pubblicarlo in due puntate successive.

Su questo numero compare la descrizione delle caratteristiche del gioco ed alcune note sulla sua implementazione; il listato del programma ed i REMarks allo stesso, compariranno sul prossimo numero di **Personal Software.**

Riprendo, con questo programma, la mia regola non scritta di prendere spunto, per i miei programmi, da opere cinematografiche. I più affezionati lettori di **Bit** ricorderanno programmi come "La 24 ore di Le Mans" (era un film con Steve McQueen), "Gran Prix" (di John Frankenheimer, con Yves Montand e James Gardner) o "'14-'18" (ispirato al film "Ma che bella guerra!" di Sir Richard Attenborough). Per questo programma, lo spunto è invece tratto dal grande cinema russo: l'episodio della battaglia del lago ghiacciato è infatti tratto dall'"Alexander Nevskij" di Sergej Michajlovič Ejzenštejn; l'autore de "La corrazzata Potëmkin" e di "I dieci giorni che sconvolsero il mondo".

In realtà l'idea di trarre da questa famosa pagina della cinematografia un wargame non è originale; ricordo che, una decina di anni fa, uno dei doni previsti per chi si abbonasse alla rivista "Linus", era appunto rappresentato da un boardgame (vale a dire un wargame da tavolo), alla cui realizzazione grafica aveva collaborato il noto disegnatore di fumetti Guido Crepax, ispirato all'episodio in questione.

L'aspetto più interessante, da un punto di vista puramente strategico, della battaglia è dato dalla disparità delle forze in campo. Di fronte, infatti, ai cavalieri in armatura dell'Ordine Teutonico, il principe Alexander Nevskij poteva schierare solo una fanteria, numerosa ma raccogliticcia, composta prevalentemente da contadini. Proprio il fatto di aver sottolineato, nel suo film del 1938, che si avvale, fra l'altro, di splendidi brani musicali, scritti per l'occasione da Sergej Prokof'ev, il grande eroismo del popolo russo nell'affrontare l'invasore germanico, valse a Ejzenštejn, nel Febbraio del 1939, l'Ordine di Lenin.

Fissiamo brevemente il momento storico in cui l'azione si svolge. Siamo nel 1242, dopo la caduta di Kiev, la Russia, sotto l'attacco dei Mongoli ad est, si smembra. Sul fronte occidentale essa è sottoposta alla espansione germanica in Livonia (l'attuale Estonia e Lituania), sorretta dall'Ordine dei Portaspada, cavalieri cattolici di nobili lignaggio, ex crociati in gran parte, fedeli a Papa Innocenzo III°. Ad essi si alleano i cavalieri dell'Ordine Teutonico dei Cavalieri della Croce, rappresentanti dell'autorità militare della Prussia, originariamente, a S. Giovanni d'Acri, in Palestina. Insieme, i due ordini rappresentavano la più grande forza militare dell'epoca, in Europa.

Quando i Teutoni puntavano su Novgorod, una lega fra le città russe minacciate si rivolge ad Alexander, principe del granducato di Suzdalia, detto Nevskij, per aver fermato, in una precedente battaglia sulla Neva, l'avanzata delle orde mongole.

La lega delle città russe non può offrire che un esercito male armato e poco addestrato, quasi interamente composto da truppe a piedi. A questo schieramento ben poco temibile, si contrappongono i Cavalieri dell'Ordine Teutonico, sotto il diretto comando del Gran Maestro dell'Ordine, Von Balk; addestratissimi e magnificamente armati, inferiori solo di numero, ma tutti forniti di armatura e cavalli appositamente addestrati per le battaglie. Il loro punto di forza era rappresentato dalla ferrea disciplina, temprata dalle rigide regole dell'Ordine, che ne facevano una sorta di monaci armati. Ed infatti uno dei pretesti che avevano per la loro espansione territoriale verso est, sostenuta, tra l'altro dalle città che aderivano alla Lega Anseatica, era quella di diffondere la fede cattolica nei paesi slavi, fino allora pagani.

Il principe Nevskij, consapevole della propria inferiorità, decise, con grande acume strategico, di fermare l'avanzata dei cavalieri teutonici sulle rive del lago Cudskoe, al confine tra l'attuale Estonia e l'U.R.S.S. propriamente detta. E, per la precisione, nel punto in cui il lago Cudskoe si congiunge col lago Pskov, uno stretto braccio circondato da paludi in cui i cavalieri teutonici, pesantemente corrazzati, non avrebbero potuto avanzare. Lo scopo del principe era duplice: costringere i cavalieri ad avanzare verso la città di Pskov attraverso lo stretto braccio ghiacciato che collega i due

La battaglia del lago ghiacciato

laghi, in modo che, nello spazio ristretto i teutonici non avessero la possibilità di far valere la propria maggior forza e mobilità e, in secondo luogo, spingere i cavalieri catafratti a correre il rischio di muoversi sulla superficie ghiacciata del lago, per inseguire le forze russe in ritirata.

Come il film illustra, la tattica del principe Alexander Nevskij riuscì appieno. Attirati sul ghiaccio dalla tattica prudenziale delle grandi masse di fanteria russe, i cavalieri dell'Ordine Teutonico, con lel loro pesanti armature, provocarono la rottura della superficie ghiacciata e perirono, in gran numero, nelle acque gelide sottostanti o schiacciati tra i blocchi di ghiaccio spezzati dal loro stesso peso.

Il film, che introduceva la "rivoluzionaria" innovazione di rappresentare i "buoni" (per Ejzenstejn, logicamente, i contadini russi) vestiti di nero ed i "cattivi" vestiti di bianco, prefigurava, nella foggia degli elmi dei cavalieri teutoni, nel loro portamento rigidamente marziale, negli stessi stemmi riprodotti sugli scudi, lo scontro che, di lì a pochi anni, doveva vedere contrapposti i battaglioni dell'armata rossa e l'esercito nazista che, pure, nel 1939 (quando a Ejzenstejn veniva conferito l'Ordine di Lenin) erano divenuti alleati, contro l'indifesa Polonia, con il patto di non aggressione Molotov-Von Ribbentrop.

Dal momento che ho potuto vedere il film per intero una sola volta (è stato trasmesso nel corso di un ciclo su Ejzenstejn che andava in onda il sabato, a tarda ora, se non sbaglio, sulla seconda rete) e, in un paio di occasioni, le pregevoli sequenze della battaglia, mi sono servito, per tutto ciò che riguarda il film, dell'interessante volume "Le-

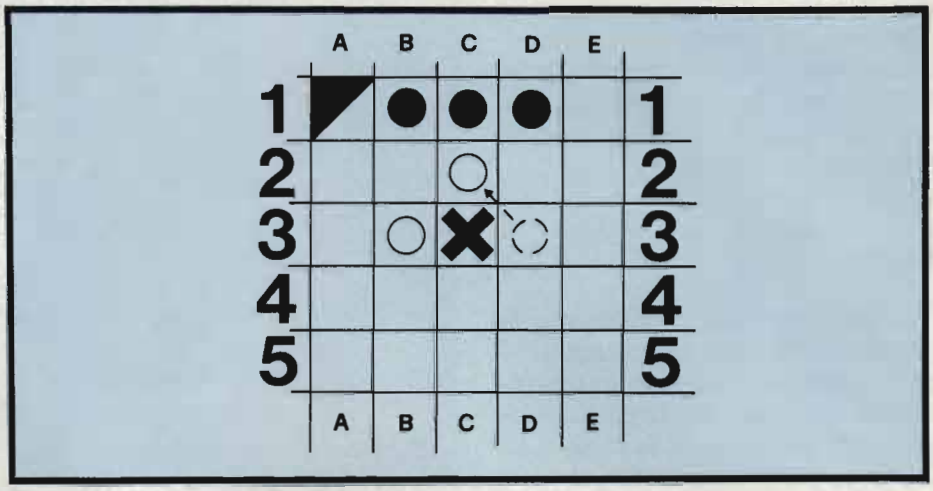


Figura 1. *Le caselle a fondo bianco, compreso quelle su cui si trovano le pedine, rappresentano la superficie ghiacciata del lago, il cui spessore dipende dallo stato iniziale e dal suo successivo deterioramento. La casella a metà bianca e nera in A-1, che rappresenta una parte della riva del lago, è invece considerata alla stessa stregua della terra-ferma.*
La croce in C-3 indica il punto in cui il ghiaccio si è rotto sotto il peso dell'unità che stava occupando la casella, inghiottendo la stessa.
La pedina bianca (unità di cavalleria teutonica) in D-3, muovendo in C-2 provoca lo scontro con le tre unità russe, con l'appoggio dalla pedina tedesca in B-3, che si trova all'interno della zona di controllo della prima pedina, unitamente alle tre unità russe.

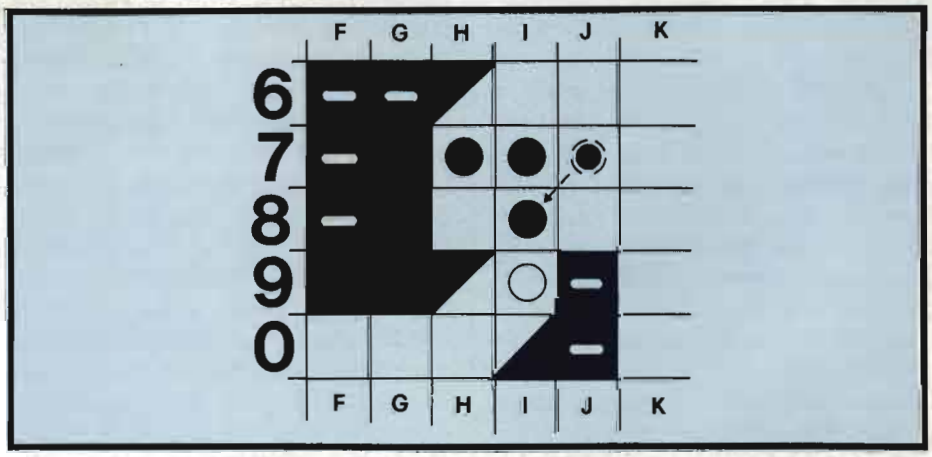


Figura 2. *Le caselle interamente nere o a metà bianche e nere (come in B-9 e I-0) rappresentano zone di terraferma, quelle contenenti un trattino bianco, come in F-6, rappresentano zone paludose, accessibili alle unità russe ma non alle più pesanti unità corazzate teutoniche.*
La pedina russa in J-7, può provocare lo scontro con la singola pedina tedesca in I-8, sia muovendo in J-8, sia muovendo in I-8. Supponendo che tutte le pedine raffigurate siano illese, nel primo caso il rapporto di forze è di 6 a 15, dato dalla somma delle capacità belliche delle due unità russe all'interno della zona di controllo dell'unità mossasi in J-8 (potenza negli scontri per la singola unità russa uguale a 3, moltiplicata per il numero delle unità russe) e dalla potenza dell'unità teutonica isolata (capacità bellica uguale a 15). Muovendo in I-8, invece, l'unità in J-7, coinvolgerebbe nello scontro anche l'unità russa in H-7, facendo salire il rapporto a 9 contro 15.

La battaglia del lago ghiacciato

zioni di regia", di Sergej Michajlovîc Ejzenštejn, pubblicato nella "Piccola biblioteca Einaudi"; mentre per le notizie storiche mi sono servito dell'Enciclopedia di Storia medioevale della Oxford University.

Il gioco

Dopo aver offerto all'utente la possibilità di caricare da nastro i dati necessari per proseguire una partita interrotta, nel caso in cui l'opzione venga rifiutata, viene chiesto all'utente di identificarsi con uno dei due strateghi impegnati nella battaglia. Quindi viene mostrata la cartina del campo di battaglia.

Per ragioni grafiche la cartina è orientata con il nord a sinistra e mostra (da destra a sinistra), l'affluente del lago Pskov, il lago Pskov stesso (ghiacciato), il braccio che collega lo Pskov al Cudskoe (circondato dalle paludi, che sono rappresentate, secondo le convenzioni cartografiche internazionali, da trattini orizzontali), quindi il lago Cudskoe, anch'esso ghiacciato ed il suo effluente.

Il programma provvede, quindi, a generare casualmente lo spessore del ghiaccio in quei tratti del lago rappresentati da caratteri grafici "pieni". Questo spessore varia tra un minimo di 7.5 ed un massimo di 13. L'unità di misura in cui i valori suddetti sono espressi non viene specificata; essa è, comunque, in relazione al peso delle singole unità. Il meccanismo, infatti, che provoca la rottura del ghiaccio è dato dal fatto che il valore che esprime il peso dell'unità che passa su di un tratto ghiacciato, sia superiore al valore numerico che esprime lo spessore del ghiaccio nel punto suddetto.

Terminata la routine precedente, il programma piazza sulla cartina le pedine che rappresentano le unità impegnate nella battaglia. Dato che la documentazione "storica" sulla battaglia è piuttosto limitata, mi sono permesso di stabilire arbitrariamente sia la numerosità dei due schieramenti che la loro posizione.

Per quanto riguarda il primo fattore, esso è stato scelto in modo da avere un rapporto di due a uno tra l'esercito russo e quello germanico.

Gli schieramenti vedevano, inizialmente, venti unità di cavalleria teutoniche affrontare quaranta unità di fanteria russe; ed entrambe le armate erano schierate lungo le rive dello Cudskoe. Dato che la cosa, però lasciava liberi dei varchi lungo la riva del lago Pskov, per gli inserimenti delle unità tedesche, ha preferito portare le unità di cavalleria a trenta (e quelle russe, naturalmente a sessanta), dislocandole, grosso modo, lungo le rive dei due laghi. La necessità di mantenere il rapporto di due a uno è data dal fatto che, in questo modo, nell'ambito di tempo in cui le unità russe compiono tutte una mossa, ogni unità teutonica ne compie due; simulando, così, una velocità di avanzamento per le truppe a cavallo teutoniche, doppia rispetto a quella delle truppe russe a piedi.

Dopo di che la partita ha inizio, alterando le mosse delle varie unità che compongono i due eserciti. L'ordine in cui esse muovono è dato dalla loro posizione sulla scacchiera. Poiché il nodo dell'azione si svolse sull'ampia superficie ghiacciata del lago Cudskoe (la quinta superficie lacustre d'Europa), si è scelto di far muovere le unità a partire da quelle più a sinistra della cartina e di far muovere i russi per primi. Per problemi di coordinamento dei movimenti le unità più vicine alla linea del fronte (per quanto riguarda la sola fanteria russa, che è disposta su più file parallele) muovono prima della retroguardia, il che porta, naturalmente, ad un certo allungamento dello schieramento, che pregiudica la compattezza della formazione russa.

Le prime unità a muovere sono quindi quelle di prima linea dell'ala destra dello schieramento russo, siano esse controllate dal calcolatore o dal giocatore. I due avversari si alternano nelle mosse in base al ritmo descritto precedentemente.

Quando il turno spetta al giocatore, il calcolatore chiede che venga introdotto, tramite il tastierino numerico, un valore da 1 a 9, 5 compreso, che indica in quale direzione si desidera spostare, di una casella, l'unità in questione. La direzione è, come al solito, indicata dalla posizione relativa dell'unità (rappresentata dal tasto "5") rispetto alle altre cifre del tastierino numerico. Così,

La battaglia del lago ghiacciato

premendo "8" indicheremo che desideriamo spostare l'unità di una casella a nord della sua posizione attuale; premendo "3" lo spostamento sarà di una casella a sud ed una a destra, vale a dire uno spostamento a sud-est dell'unità, mentre, nel caso in cui si prema il tasto "5", l'unità rimarrà nella sua posizione attuale.

Dopo ogni mossa il calcolatore provvede a controllare che non si verifichino incongruenze (tentativi di sovrapporre più unità sulla stessa casella) e ad esaminare la situazione creatasi con la nuova mossa.

Così, il calcolatore provvederà ad aggiornare la variabile associata alla vecchia posizione occupata dall'unità mossasi, in modo da simulare un deterioramento dello stato del ghiaccio (nel caso in cui, ovviamente, l'unità si trovasse sulla superficie del lago) in seguito al passaggio dell'unità stessa. Questo deterioramento è proporzionale al peso delle varie unità, e, per la precisione, è pari alla metà del loro peso attuale.

Quindi, un'unità di cavalleria tedesca determina una diminuzione nello spessore del ghiaccio di 4 unità, un cavaliere teutonico appiedato (peso uguale a tre unità), di una unità e mezzo; mentre un'unità di fanteria russa toglie allo spessore del ghiaccio solo mezza unità di misura.

Analogamente il calcolatore provvede a verificare il caso in cui lo spostamento dell'unità l'abbia portata su di una zona ghiacciata, non in grado di reggerne il peso (valore eprimente il peso attuale dell'unità superiore al valore numerico rappresentante lo spessore del ghiaccio nel punto), oppure su di una zona paludosa. Mentre, però, la verifica, nel primo caso viene compiuta per le unità di entrambi gli schieramenti, nel caso delle paludi le unità di fanteria russe, più leggere e prive di corazza, possono avanzare impunemente senza paura di sprofondare.

Come i patiti di boardgame sanno, ogni pedina o "counter" ha associati due o più valori numerici che ne determinano le caratteristiche per capacità belliche, velocità di movimento ecc.. Nel caso di "La battaglia del lago ghiacciato", come abbiamo visto, il fattore velocità è insito nella numerosità stessa dei due schieramenti, che simula una velocità della cavalleria teutonica doppia rispetto a quella della fanteria russa, senza introdurre esplicitamente un fattore velocità (che, peraltro, esiste, come vedremo, sotto forma di han-

La battaglia del lago ghiacciato

dicap provocato dalle conseguenze degli scontri tra più unità, in modo da rallentare le unità ferite o, ad esempio, far muovere i cavalieri tedeschi disarcionati alla stessa velocità delle truppe russe a piedi). Un secondo fattore che abbiamo già visto, piuttosto inusuale per un boardgame, è dato dal peso delle varie unità che è stabilito come segue: presa come unità di misura una pedina russa (quindi una unità di fanteria armata alla leggera e priva di corazza), una unità di cavalleria teutonica completa di corazza e cavalcatura ha un peso otto volte superiore, ed un cavaliere appiedato (ma pur sempre fornito di corazza) ha un peso pari a tre volte quello di una unità di fanteria russa.

Rimangono quindi da considerare i fattori di capacità belliche o di potenza negli scontri. Innanzi tutto ho unificato, come capita ormai sovente i fattori di attacco e di difesa. Ho inoltre stabilito che, nel caso in cui l'esito di uno scontro veda il "ferimento" di una unità, la sua efficienza venga diminuita di una unità e mezza. I valori di partenza sono i seguenti: una unità di cavalleria in perfetto stato ha un valore di potenza negli scontri pari a quindici unità. Nel caso in cui venga ferita una prima volta la sua potenza scende a 13.5; nel caso in cui venga però ferita una seconda volta essa viene considerata "morta". Nel caso in cui una unità di cavalleria venga disarcionata, oltre che il suo peso, (che scende, come visto, a tre) diminuisce anche la sua potenza negli scontri di dodici unità, pertanto, se il cavaliere non era mai stato ferito la sua potenza passa a tre (vale a dire un valore pari a quello di una unità di fanteria russa in perfetta efficienza) mentre se l'unità era già stata ferita in precedenza il suo valore di potenza passa a 1.5. Ovviamente unità appiedate, siano esse teutoniche o russe, nel caso in cui vengano ferite due volte vengono considerate "morte".

Si può forse considerare discutibile la scelta di equiparare una unità di cavalleria disarcionata ad una unità di fanteria illesa. Ma, se la protezione offerta dall'armatura rappresenta un evidente vantaggio in caso di scontri essa rappresenta un non indifferente svantaggio dal punto di vista dell'agilità nei combattimenti e della libertà di movimenti nella marcia, svantaggio tale che è forse eccessivo supporre che un cavaliere disarcionato sarebbe riuscito ad avanzare alla stessa velocità di una unità di fanteria leggera. I due errori, quindi, si compensano.

Il calcolatore provvede anche, dopo ogni mossa a verificare che si siano venute a creare le condizioni necessarie ad uno scontro tra gli opposti schieramenti. Perché le condizioni suddette siano soddisfatte, è necessario che, all'interno della Z.O.C. (Zone of control, zona di controllo) dell'ultima unità mossasi si trovi almeno una unità nemica. La zona di controllo, per ciascuna unità, indipendentemente dall'esercito cui appartiene e dal suo stato attuale, è rappresentata da un quadrato di lato tre unità video, centrato sulla posizione attuale dell'unità attaccante.

La C.R.T. (Combat results table, tabella dei risultati dei combattimenti) è determinata, oltre che da un valore casuale (in luogo dei consueti dadi dei patiti di boardgame) anche dal rapporto di forze esistente, all'interno della suddetta zona di controllo.

Contrariamente ai boardgame tradizionali, ed anche all'esempio di computer wargame da me realizzato per **Bit** ('14-'18, apparso su **Bit** N° 23, anno 4°, Dicembre 1981), ho preferito limitare, anche, lo confesso, per ragioni pratiche, la determinazione del supporto fornito ai due schieramenti coinvolti nello scontro, alla sola Z.O.C. dell'unità responsabile dell'iniziativa di provocare lo scontro, in modo da concedere quindi un vantaggio allo schieramento attaccante, rispetto a chi sta sulla difensiva.

Facciamo un esempio pratico, supponiamo (vedi figura 1) che l'unità teutonica in D-3 cui tocca muovere sia illesa (capacità bellica uguale a 15), mentre la sua compagnia in B-3 sia stata disarcionata in uno scontro precedente (per cui la sua capacità bellica sarà ridotta a 3), mentre delle tre unità russe, due siano nelle loro piene capacità belliche ed una sia già stata ferita (capacità limitata a 1.5 rispetto alle 3 unità originali). Se l'unità in D-3 si muove in C-2 (numero corrispondente del tastierino numerico "7") essa verrà a trovarsi ad avere le tre unità nemiche all'interno della sua Z.O.C e provocherà, quindi, lo scontro. Sommando le capacità belliche delle unità dei due schieramenti all'interno della Z.O.C. dell'unità che si è mossa per ultima otteniamo un valore di 18 unità per lo schieramento tedesco e di 7.5. unità per lo schieramento russo. Con una semplice proporzione, si ottengono i valori percentuali corrispondenti alla ripartizione delle probabilità di esito favorevole per i due eserciti. Con i valori suddetti il rapporto è di 70.58 a 29.42 a favore dello schieramento germanico. Ciò significa che ci sono 70.58 probabilità su cento che l'esito dello scontro sia favorevole alle unità teutoniche e 29.42 a favore delle unità russe. Le probabilità suddette sono poi, a loro volta, suddivise tra i differenti esiti a danno dell'uno o dell'altro dei due schieramenti. Gli esiti suddetti si applicano a tutte le unità di uno stesso schieramento coinvolte nello scontro, siano state esse oppure no a provocarlo direttamente.

Nel caso delle truppe tedesche, le 29.42 probabilità di avere esiti a loro sfavorevoli sono equamente suddivise tra:

— la scomparsa del campo di gioco di entrambe le unità coinvolte, vale a dire la loro "morte";

— il disarcionamento di entrambe le unità (nel caso dell'unità in B-3, essendo quest'ultima già stata disarcionata, essa verrebbe, per questa ragione "ferita", vale a dire, la sua

La prima ... ... l'unica

Enciclopedia di Elettronica e Informatica

Oggi con la 2ª Edizione ...la più aggiornata! In edicola.

Il successo si ripete

E.I. un prezioso strumento di formazione e di aggiornamento
a cui sono abbonati anche migliaia di specialisti, tra cui 4000 quadri FIAT

un orgoglioso primato dell'editoria italiana
alla cui pubblicazione sono interessati editori francesi, tedeschi, svedesi, canadesi, inglesi, sudafricani, portoghesi, spagnoli, australiani, zelandesi, messicani, sudamericani

uno strepitoso successo di lettori:
fino a oggi 6.000.000 di fascicoli venduti

una splendida opera da biblioteca
da 60 fascicoli settimanali
7 volumi - 1680 pagine - 700 foto
2200 illustrazioni a colori

E.I. una prestigiosa collaborazione tra
Learning Center
TEXAS INSTRUMENTS

GRUPPO EDITORIALE JACKSON

La battaglia del lago ghiacciato

capacità bellica scenderebbe a 1.5); — il ferimento di entrambe le unità (in questo caso la capacità bellica delle unità in C-2 scenderebbe a 13.5, mentre quella della B-3 a 1.5).

Per tutti e tre gli esiti le probabilità di avverarsi sarebbero del 9.8%. Nel caso delle truppe russe, non essendoci un esito che preveda, ovviamente il disarcionamento delle unità, per ragioni di simmetria nella determinazione degli esiti, sono previste doppie possibilità di "ferimento" delle unità coinvolte nello scontro. Abbiamo quindi 23.52% di probabilità che le unità in questione siano "uccise" e 47.06% che siano ferite. In quest'ultimo caso, però, poiché una pedina russa è già stata ferita, essa viene considerata "morta" e tolta dal campo di gioco.

Un altro semplice esempio (vedi figura 2), la pedina russa in J-7, provocherebbe lo scontro muovendo in J-8 (tasto corrispondente alla mossa suddetta "2"), poiché la pedina teutonica in I-9 sarebbe all'interno della sua Z.O.C., unitamente all'unità russa in I-7; dalla Z.O.C. stessa rimarrebbe però esclusa l'unità russa in H-7, che, così, non parteciperebbe allo scontro. La somma delle capacità belliche (supponendo che tutte le unità siano al pieno delle loro forze) sarebbe di 6 per lo schieramento russo e di 15 per quello teutonico, il che porterebbe a delle probabilità percentuali di 28.57 a 71.43, a favore dell'unità teutonica.

Muovendo invece in I-8, e coinvolgendo quindi l'unità russa in H-7; la somma delle capacità belliche russe salirebbe a 9, ed il rapporto percentuale, pur rimanendo a favore della più potente unità teutonica (corazzata e a cavallo) sarebbe più vantaggioso per lo schieramento russo: 37.5 a 62.5.

I differenti esiti hanno anche effetti collaterali, oltre quelli, immediati, sulle capacità belliche delle unità coinvolte. Nel caso, infatti in cui una unità di cavalleria venga "disarcionata", essa non continuerà a muoversi alla stessa velocità delle sue compagne a cavallo, ma riceverà un handicap che ne rallenterà le capacità di movimento, eguagliandole a quelle di una unità di fanteria in piena efficienza. Un handicap ancora superiore verrà inflitto alle unità a piedi ferite (siano esse russe o unità teutoniche appiedate), rallentandone ulteriormente l'avanzamento ad una mossa ogni tre di una normale unità germanica a cavallo. Un analogo handicap viene inflitto, ma ha effetto immediato e limitato alle due mosse successive, ad una unità teutonica che venga ferita mentre si trova a cavallo.

Ad intervalli regolari rispetto allo svolgimento della partita (ogni turno completo di mosse per la cavalleria teutonica), il computer provvede a calcolare e mostrare il punteggio parziale dell'incontro. Esso è calcolato in base alla diminuzione di capacità belliche inflitte all'avversario, più un bonus di un punto per ogni unità nemica scomparsa dalla scacchiera di gioco. Da notare che i danni inflitti all'avversario negli scontri vengono sommati a quelli provocati da incidenti senza l'intervento di unità nemiche (come unità sprofondate nel ghiaccio o, per le sole truppe tedesche, impantanate nelle paludi). Le condizioni di vittoria che portano alla conclusione dell'incontro sono, ovviamente, basate sul punteggio suddetto. Esse sono, però, espresse in termini di rapporti di forza.

Le capacità belliche complessive dei due schieramenti, sono inizial-

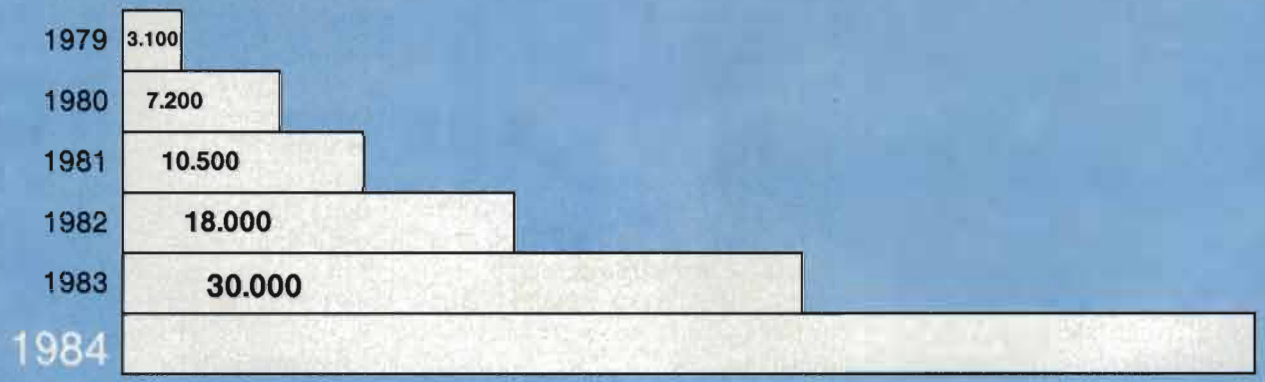
SALONE DELL' INFORMATICA 1984

Quartiere Fiera di Milano

(reception riservata con ingresso da via Gattamelata)

14-18 aprile 1984

L'unica mostra nazionale interamente ed unicamente dedicata all'informatica.
L'appuntamento di primavera scelto dall'utenza.



VISITATORI SALONE DELL'INFORMATICA

6° salone nazionale

per l'elaborazione e trasmissione dei dati e dei messaggi.

Mini - personal - home computer - software - accessori - apparecchiature per l'automazione dell'ufficio - sistemi e servizi di telematica.

Segreteria e informazioni:
E.P.I. - Ente Promozione Informatica
20139 MILANO - Via Marochetti, 27
tel. (02) 56.93.973 - 53.98.267

mente calcolate in ragione della somma delle capacità belliche delle singole unità, più un punto per ogni unità che compone ciascuno schieramento. Con queste regole le capacità belliche della formazione teutonica assommano a 480 punti, distribuite tra trenta diverse unità. Nel caso dell'esercito di Alexander Nevskij, le limitate capacità belliche delle unità di fanteria portano la potenza complessiva dello schieramento russo a 240 punti, distribuiti, però, tra il doppio di unità rispetto allo schieramento teutonico. Ciò significa che, nel caso delle truppe russe, la perdita di una singola unità è meno "costosa" rispetto a quelle teutoniche: se, infatti, una unità di fanteria russa "muore" in uno scontro, la diminuzione di potenza per lo schieramento russo è di 4 punti (tre di capacità belliche ed uno per la perdita dell'unità stessa), nel caso in cui, invece, lo schieramento teutonico perde una unità di cavalleria (anche senza che venga coinvolta in uno scontro, ad esempio solo perché il ghiaccio si rompe), la perdita è di 16 punti (quindici di capacità belliche).

Il rapporto di forze iniziale tra i due eserciti è, come visto, di due a uno in favore dell'esercito teutonico. Le condizioni di vittoria fissate per l'esercito germanico prevedono che il rapporto di forze venga portato a tre a uno, mentre, affinché vinca l'esercito russo, è sufficiente che il rapporto di forze sia pari, poiché questo significherebbe una netta superiorità numerica dell'esercito russo.

È quindi evidente che, nel caso in cui l'attraversamento del lago si rivelasse particolarmente sfortunato per l'esercito tedesco, anche un rapido e fruttuoso aggiramento del lato destro dello schieramento russo non impedirebbe a quest'ultimo di avere, con un po' di pazienza, la meglio.

Il programma

Come detto, l'unica scelta che viene sottoposta al giocatore riguarda la direzione in cui spostare la pedina cui spetta il turno. In realtà, il giocatore ha anche un'altra opzione a disposizione: rispondendo alla domanda, circa quale direzione far prendere all'unità, premendo il tasto "0", il calcolatore provvederà ad interrompere la partita ed a dare al giocatore l'opportunità di salvare una copia della situazione intermedia su nastro. Data la "mole" di dati necessaria, l'operazione suddetta impiega circa sette minuti e valori analoghi per ripristinare la situazione da programma. Dopo di che viene offerta all'utente l'alternativa tra proseguire la partita interrotta o porre fine al programma.

Per quanto riguarda l'algoritmo decisionale, responsabile delle mosse del calcolatore, esso è organizzato, come già in altri wargame da me realizzati (il già citato "'14-'18" o "Abukir 1798", uscito sul numero scorso di **Personal Software**), con un ciclo di valutazione sulle varie mosse disponibili per l'unità cui spetta muovere. Per quanto riguarda i criteri di valutazione suddetti, si è data la precedenza al controllo dello svolgimento degli scontri, in modo da evitare di affrontarli con rapporti di forze inferiori alla parità, il che porta, logicamente e abbastanza verosimilmente, le potenti unità di cavalleria teutoniche ad essere particolarmente aggressive, le più deboli unità di fanteria russa a tenersi sulla difensiva.

Nel caso in cui la mossa non sia influenzata dalla necessità di provocare od evitare lo scontro, la valutazione delle mosse viene determinata in base a considerazioni strategiche (tenere le truppe unite, in modo da avere a disposizione in caso di scontro rapporti di forza favorevoli) e, per così dire, "geografiche", vale a dire riguardanti la posizione dell'unità cui spetta muoversi relativamente alle posizioni strategiche del campo di gioco.

Pur adottando metodi di valutazione sostanzialmente simili, almeno per quel che riguarda la parte strategica, la routine in questione si è rivelata, per "La battaglia del lago ghiacciato", sensibilmente più lenta, rispetto a quella di uguale funzione inserita in "Abukir 1798". La differenza è, fondamentalmente, data dal fatto che, mentre alle navi impegnate nella battaglia di Abukir non era consentito compiere virate superiori a 45°, nel caso delle unità de "La battaglia del lago ghiacciato" sono ammessi sia bruschi cambi di direzione che ritirate precipitose, la qual cosa rende necessario esaminare tutte e nove le alternative che si pongono ad ogni mossa, contro le tre possibili per "Abukir 1798".

Il computer, tutto sommato, si comporta piuttosto bene, anche se, nella parte del Gran Maestro dell'Ordine Teutonico Von Balk, risente degli inevitabili problemi provocati dalla necessità che le pesanti unità di cavalleria germaniche corrano l'alea dell'attraversamento della superficie ghiacciata del lago, per entrare in contatto con le forze nemiche.

Non ho ritenuto, d'altronde, conveniente inserire un meccanismo che spingesse le unità tedesche a rimanere ferme sulle loro posizioni ad attendere l'attacco delle più deboli unità russe poiché questo sarebbe stato "antistorico" e non conforme con il fatto che i cavalieri dell'Ordine Teutonico rappresentavano, nella battaglia del lago Cudskoe, l'avanguardia della espansione prussiana in oriente.

Epilogo

Queste brevi considerazioni concludono la prima parte dell'articolo. Diamo appuntamento ai lettori, che hanno trovato queste premesse abbastanza interessanti da essere invogliati a conoscere il programma nei suoi particolari, al prossimo numero di **Personal Software**, su cui appariranno il listato del programma ed i commenti allo stesso. ■

è in edicola il nuovo numero

ANTEPRIMA:
OLIVETTI M10

TUTTO FIERE
EDP-USA
SIOA
COMPUTER SHOW

VISICALCOLIAMO
L'IRPEF '84

QUICKCODE:
UN GENERATORE
DI PROGRAMMI
PER DBASE II

CON INSERTI:

SUPER BIT RISERVATO PERSONAL
E
DIGIDATTICA

UNA PUBBLICAZIONE DEL GRUPPO EDITORIALE JACKSON

PERSONAL SOFTWARE

Othello per ZX81

— Parte prima —

Presentiamo in questa prima puntata un famoso gioco implementato su ZX81

di *Angelo Motta*

Hai mai giocato ad Othello? È facile, anzi la pubblicità del gioco dice "è più facile imparare a giocarlo che smettere"! Ed è vero, è un bellissimo gioco di riflessione e strategia.

Si gioca su una scacchiera di 64 caselle. Al centro vengono poste, due pedine bianche e due pedine nere, al turno, ogni giocatore depone una pedina del proprio colore sulla scacchiera in modo da racchiudere almeno una pedina avversaria fra le proprie, se non gli è possibile perde il turno. Le pedine imprigionate diventano dello stesso colore di quelle del giocatore che ha mosso. Quando sono state deposte tutte le 64 pedine, vince chi ha più pedine del proprio colore.

Il programma qui presentato è una modifica più che una conversione di quello apparso su **Personal Software** n. 3 per Atari.

Infatti, per rendere più veloce la risposta del computer, sono state scritte alcune parti in linguaggio macchina, oltre ad altre modifiche. L'impostazione del gioco è rimasta invariata; come per l'Atari non bisogna inserire delle coordinate per effettuare la mossa, ma spostare con i tasti "freccia" una pedina lampeggiante e deporla nella casella desiderata premendo il tasto (0). Inoltre, il computer non è in grado di capire quando il giocatore non può effettuare la mossa; in tal caso basterà premere il tasto (1) per passare la mossa allo ZX.

Byte	Decimale			HEX			Mnemonici	Note
16907	33	0	125	21	00	7D	LD HL, 32000	Carica la mossa nel registro E.
16910	94			5E			LD E, (HL)	
16911	1	8	0	01	08	00	LD BC, 8	
16914	197			C5			PUSH BC	
16915	33	180	65	21	B4	41	LD HL, 16820	Pone in D la 1° direzione.
16918	9			09			ADD HL, BC	
16919	86			56			LD D, (HL)	
16920	123			7B			LD A, E	Somma la direzione alla mossa e la pone in C.
16921	130			88			ADD A, D	
16922	79			4F			LD C, A	
16923	33	255	119	21	FF	77	LD HL, 30719	La casella indicata dalla direzione è occupata dalla pedina avversaria?
16926	9			09			ADD HL, BC	
16927	126			7E			LD A, (HL)	
16928	33	10	125	21	0A	7D	LD HL, 32010	
16931	190			BE			CP (HL)	
16932	32	25		20	19		JR NZ + 25	Se no cambia direzione.
16934	229			E5			PUSH HL	Aggiungi nuovamente la direzione.
16935	121			79			LD A, C	
16936	130			82			ADD A, D	
16937	79			4F			LD C, A	
16938	33	255	119	21	FF	77	LD HL, 30719	La nuova casella è occupata da una pedina avversaria?
16941	9			09			ADD HL, BC	
16942	126			7E			LD A, (HL)	
16943	225			E1			POP HL	
16944	190			BE			CP (HL)	
16945	40	243		28	F3		JR Z − 13	Se si aggiunge nuova direzione la casella è occupata da una pedina propria? Se no cambia direzione. La mossa è valida ritorna al BASIC.
16947	33	20	125	21	14	7D	LD HL, 32020	
16950	190			BE			CP (HL)	
16951	32	6		20	06		JR NZ + 6	
16953	33	30	125	21	1E	7D	LD HL, 32030	
16956	115			73			LD (HL), E	
16957	193			C1			POP BC	
16958	201			C9			RET	
16959	193			C1			POP BC	Carica altra direzione e riprova il controllo.
16960	13			0D			DEC C	
16961	32	207		20	CF		JR NZ − 49	
16963	201			C9			RET	Mossa errata-ritorna

Figura 1. *Routine in linguaggio macchina relativa al controllo della mossa effettuata da entrambi i giocatori. Per inserire i codici usare i listati 2 e 3.*

Il programma presentato deve essere inserito in due fasi successive.

1ª fase: inserire la REM 1 di 450 caratteri e le 77 righe da 100 a 160. A questo punto dare il RUN ed inserire i codici del linguaggio macchina. La REM contiene:

— le variabili fisse per la scelta della mossa dello ZX (locazioni da 16514 a 16613);

— routine per la stampa della scacchiera sul video e per l'inizializzazione delle variabili di controllo gioco in RAMTOP (da 16614 a 16820);

— le variabili per il controllo validità mossa per entrambi i giocatori (da 16820 a 16828);

— la routine per la scelta della mossa di ZX (da 16829 a 16906);

— la routine di controllo validità della mossa (da 16907 a 16953).

Una volta inseriti i codici macchina il programma si fermerà e sul video apparirà la scritta 9/160.

A questo punto occorre, per mettersi al riparo da errori quasi inevitabi-

li, registrare almeno un paio di volte il lavoro fatto.
2ª fase: cancellare le righe da 100 a 160 ed inserire il resto del programma. Si noti che le linee 100 e 110 abbassano RAMTOP di 2 Kbyte. Per il programma sarebbe bastato un abbassamento di un quarto di K; ho voluto aumentare la zona disponibili per lasciare la possibilità di duplicare le variabili di controllo gioco per chi volesse creare un secondo livello di gioco con possibilità di analizzare alcune mosse successive.

Gestione del gioco

Per la gestione del gioco viene usata una matrice di 100 byte nelle locazioni di memoria da 30720 a 30819 (figura 5). Tale matrice corrisponde per le 64 locazioni centrali alla scacchiera dello Othello; le restanti 36 locazioni sono il bordo esterno della scacchiera, indispensabile per il corretto funzionamento del gioco.

La figura 1 mostra la matrice all'inizio del gioco. I byte contenenti il valore 0 corrispondono alle caselle libere, quelli col valore 1 alle caselle occupate dal giocatore, quelli con 2 alle caselle occupate dallo ZX e con 9 il bordo esterno.
Quando si esegue una mossa è possibile imprigionare le pedine avversarie in 8 direzioni. Queste direzioni si riducono a 5 ai bordi e a 3 agli angoli (figura 3).
La funzione del bordo esterno è quella di evitare che nell'esecuzione della routine per il controllo della mossa - più avanti illustrata - e nell'aggiornamento della scacchiera (subroutine BASIC dalla linea 800 alla linea 960) lo ZX fuoriesca dal quadro di gioco con spiacevoli anomalie di funzionamento.

Byte	Decimale			HEX			Mnemonici	Note
16829	33	40	125	21	28	7D	LD HL, 32040	Predispone le variabili nei byte di controllo.
16832	54	1		36	01		LD (HL), 1	
16834	33	10	125	21	0A	7D	LD HL, 32010	
16837	54	1		36	01		LD (HL), 1	
16839	33	20	125	21	14	7D	LD HL, 32020	
16842	54	2		36	02		LD (HL), 2	
16844	33	30	125	21	1E	7D	LD HL, 32030	
16847	54	1		36	01		LD (HL), 1	
16849	01	12	0	01	0C	00	LD BC, 12	Inizio ciclo.
16852	197			C5			PUSH BC	
16853	33	255	119	21	FF	77	LD HL, 30719	Controlla che la casella sia vuota - se occupata passa alla successiva.
16856	9			09			ADD HL, BC	
16857	126			7E			LD A, (HL)	
16858	184			B8			CP B	
16859	32	37		20	25		JR NZ + 37	
16861	33	129	64	21	81	40	LD HL, 16513	Carica il valore della matrice di scelta mossa (vedi figura 2).
16864	229			E5			PUSH HL	
16865	9			09			ADD HL, BC	
16866	126			7E			LD A, (HL)	
16867	33	40	125	21	28	7D	LD HL, 32040	Lo confronta con quello scelto in precedenza (all'inizio 1) e, se minore, scarta la casella e passa alla successiva.
16870	17	0	0	11	00	00	LD DE, 0	
16873	94			5E			LD E, (HL)	
16874	225			E1			POP HL	
16875	25			19			ADD HL, DE	
16876	150			96			SUB (HL)	
16877	250	3	66	FA	03	42	JP M, 16899	
16880	33	0	125	21	00	7D	LD HL, 32000	Controlla la validità della mossa (vedasi figura 3).
16883	113			71			LD (HL), C	
16884	205	11	66	CD	0B	42	CALL 16907	
16887	33	30	125	21	1E	7D	LD HL, 32030	
16890	126			7E			LD A, (HL)	Se non valida passa ad esaminare la successiva.
16891	33	40	125	21	28	7D	LD HL, 32040	
16894	190			BE			CP (HL)	
16895	250	3	66	FA	03	42	JP M, 16899	
16898	119			77			LD (HL), A	
16899	193			C1			POP BC	Seleziona la casella successiva da controllare. Quando le ha passate tutte in rassegna ritorna al BASIC.
16900	12			0C			INC C	
16901	121			79			LD A, C	
16902	254	90		FE	54		CP 90	
16904	200			C8			RET Z	
16905	24	201		18	C9		JR − 55	

Figura 2. *Routine in linguaggio macchina con la quale lo ZX sceglie la mossa al livello 1°. Per inserire i codici usate i listati 2 e 3.*

Routine di controllo della validità della mossa

La mossa è valida se viene imprigionata almeno una pedina avversaria fra una propria già presente sulla scacchiera e quella che si depone. Nella matrice di 10 x 10 le direzioni per muoversi nelle direzioni possibili vengono comodamente indicate con − 11 per alto sinistra, − 10 per in alto, − 9 per in alto a destra, − 1 per a sinistra, + 1 per a destra, + 9 per in basso a sinistra, + 10 per in basso e + 11 per in basso a destra (si veda la figura 4). Queste variabili sono inserite nel programma nella REM iniziale fra le locazioni dal n. 16821 al

Othello per ZX81

n. 16828.
A questo punto occorre aprire una parentesi. Se facciamo:

PRINT PEEK 16821 (NEW LINE)

sul video apparirà il valore 245 anziché − 11 sopra descritto.

Lo stesso vale per le altre locazioni 16822, 16823, 16824 che invece dei valori negativi contengono rispettivamente 246, 247, 255. Questo perché la CPU Z80 memorizza i numeri negativi nella forma "a complemento a due". Per chi non conoscesse la numerazione binaria in complemento a due ne viene fatta menzione nel riquadro.
La routine per il controllo della mossa è scritta interamente in linguaggio macchina ed è illustrata nella figura 2. Quando il giocatore sceglie la mossa, lo ZX, partendo da una delle direzioni sopradescritte, controlla se la casella adiacente è occupata da una pedina avversaria. In caso contrario prova con un'altra direzione.

Se la casella è occupata da una pedina avversaria, controlla, sempre nella medesima direzione che la successiva sia occupata o da una pedina del giocatore, nel qual caso conferma la mossa, o da una pedina avversaria, a fronte della quale continua nella stessa direzione, o da una casella vuota o dal bordo esterno e, in questo caso cambia la direzione di controllo. Se esaurite tutte le otto direzioni non è possibile imprigionare alcuna pedina avversaria, viene segnalata l'invalidità della mossa.

Strategia dello ZX81 al 1° livello

Lo schema di gioco dello ZX al 1° livello è abbastanza semplice.
Ad ogni casella della scacchiera è assegnato un valore (figura 6) in base all'importanza strategica della stessa. Si osserva perciò che mentre la casella d'angolo ha un valore di 9, le adiacenti alla stessa sono valutate 2 ai bordi ed 1 in diagonale.

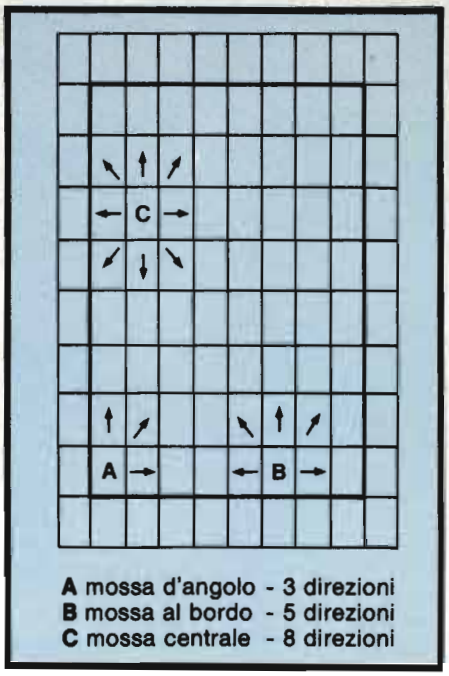


Figura 3. *Tipo di mosse effettuabili e relative direzioni di imprigionamento pedine avversarie.*

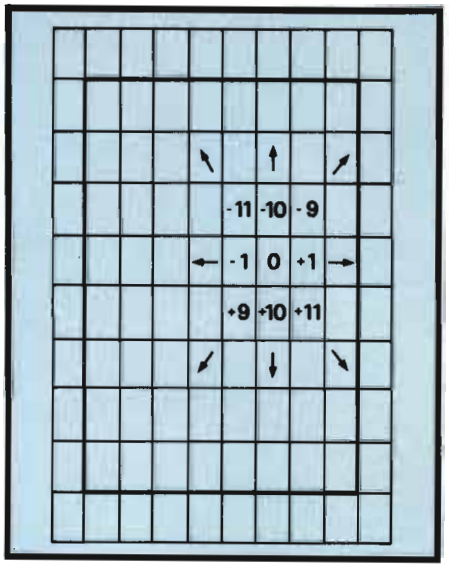


Figura 4. *Variabili utilizzate dallo ZX (e loro direzioni) per il controllo della validità della mossa e per l'aggiornamento della scacchiera.*

Il valore di tale caselle è contenuto nella REM 1 nelle locazioni da 16514 a 16613 e sono comprensive del bordo esterno il cui valore è 0.
Al turno lo ZX sceglie, fra le mosse possibili, quella la cui casella ha il valore strategico più alto.

La routine per la scelta della mossa al 1° livello, scritta anche essa interamente in linguaggio macchina, è illustrata nella figura 1. Inizia con l'assegnazione delle variabili di controllo: 1 nella locazione 32010 (pedina del giocatore), 2 nella locazione 32020 (pedina dello ZX) e 1 in entrambe le locazioni 32030 e 32040 indispensabili la prima per il controllo della validità della mossa e la seconda per la scelta della mossa stessa.
Lo ZX inizia quindi l'esame della scacchiera partendo dalla prima casella in alto a sinistra (la n. 12 - figura 5) e, man mano le passa tutte in rassegna. Se la casella è occupata la scarta e passa alla successiva, altrimenti chiama la subroutine per il controllo della validità della mossa. Se questa non è possibile scarta la casella e continua con la prossima, se valida controlla, il valore strategico della casella (figura 1) con quello della casella il cui numero è contenuto nella locazione 32040 (all'ini-

(30720)	9	9	9	9	9	9	9	9	9
(30730)	9	0	0	0	0	0	0	0	9
(30740)	9	0	0	0	0	0	0	0	9
(30750)	9	0	0	0	0	0	0	0	9
(30760)	9	0	0	0	1	2	0	0	9
(30770)	9	0	0	0	2	1	0	0	9
(30780)	9	0	0	0	0	0	0	0	9
(30790)	9	0	0	0	0	0	0	0	9
(30800)	9	0	0	0	0	0	0	0	9
(30810)	9	9	9	9	9	9	9	9	9

Figura 5. *Matrice contenuta nelle locazioni da 30720 a 30819 contenente le variabili per il controllo del gioco: 1 = pedina del giocatore; 2 = pedina dello ZX; 0 = casella libera; 9 = bordo della matrice.*

(16514)	0	0	0	0	0	0	0	0	0
(16524)	0	9	2	8	6	6	8	2	9	0
(16534)	0	2	1	3	4	4	3	1	2	0
(16544)	0	8	3	7	5	5	7	3	8	0
(16554)	0	6	4	5	0	0	5	4	6	0
(16564)	0	6	4	5	0	0	5	4	6	0
(16574)	0	8	3	7	5	5	7	3	8	0
(16584)	0	2	1	3	4	4	3	1	2	0
(16594)	0	9	2	8	6	6	8	2	9	0
(16604)	0	0	0	0	0	0	0	0	0	0

Figura 6. *Matrice contenuta nelle locazioni da 16514 a 16613 con inserito il valore delle singole caselle della scacchiera, in base al quale lo ZX sceglie la mossa.*

zio della routine 1 corrispondente al bordo esterno il cui valore è 0).

Se tale valore è superiore lo ZX inserisce nella locazione 32040 il numero della casella che sta esaminando come possibile mossa, in caso contrario scarta la casella e continua con la prossima.

Quando sono state passate in rassegna tutte le 64 caselle, nella locazione 32040 lo ZX leggerà il numero della casella da scegliere come mossa. Se il valore della locazione sarà 1, lo ZX farà apparire sullo schermo la scritta "NESSUNA MOSSA" e passerà il turno.

Sul prossimo numero verrà pubblicata un'estensione del programma che consentirà di sfidare lo ZX81 al 2° livello di gioco.

Ultima nota: il programma è predisposto per lo ZX81 o ZX80 nuova ROM + scheda SLOW. È possibile inserire delle pause per farlo funzionare con lo ZX80 senza SLOW, ma non si ottengono dei buoni risultati a causa dell'assenza della pedina lampeggiante che crea confusione sulla scacchiera. ■

Rappresentazione dei numeri binari in complemento a due

Per quanto riguarda la rappresentazione dei numeri binari si rimanda alla tabella "Immagazzinamento di numeri di due byte" apparsa nell'articolo "Linguaggio macchina per ZX81" di Bruno del Medico sul n. 12/13 della rivista.
Tale tabella illustra come è possibile ottenere 256 combinazioni di numeri (dallo 0 a 255) con 8 bit (1 byte). Questi numeri sono però numeri interi positivi.
Per ottenere i numeri negativi si usa la notazione in complemento a due.
Con questa forma i numeri positivi vengono rappresentati sempre allo stesso modo; i numeri negativi si ottengono portando a complemento ad uno (ossia lo 0 diventa 1 e l'1 diventa 0) il numero positivo ed al risultato ottenuto si aggiunge 1.
Prendiamo ad esempio il n. 11 positivo che nella forma binaria è rappresentato:

11 = 0 0 0 0 1 0 1 1

per ottenere lo stesso numero in negativo si dovrà procedere:

numero positivo	0 0 0 0 1 0 1 1	= 11
complemento ad 1	1 1 1 1 0 1 0 0	
+ 1	1	(somma)
numero in complemento a due	1 1 1 1 0 1 0 1	= − 11 (o 245 pos.)

Pertanto, con la notazione in complemento a due in un byte si avranno 127 numeri positivi e 128 numeri negativi.
Un metodo molto più semplice per trovare i numeri negativi, che risulta comodo quando si devono effettuare delle POKE o inserire i codici del linguaggio macchina, è di sottrarre al numero 256 il numero positivo. Esempio: 256 − 11 = 245 che è appunto il corrispondente positivo del − 11 (si veda sopra).
Per chi volesse meglio approfondire lo studio sulla numerazione binaria e sulla programmazione in linguaggio macchina dello Z80, l'autore consiglia il volume "Programmazione dello Z80" di Rodnay Zaks edito dalla Jackson.

Listato 1. *La lunga lista di REM contiene la parte più importante del programma che determina la mossa del computer.*

```
  1 REM
                ??( UNPLOT TAN E£RN
D: "Y 4LN NEW RND7$TAB PAUSE RND
 2 LN NEW RNDY ;LN NEW RNDY
 LN NEW RND7$SN RINKEY$ LN NE
W RND 70 70 70 ( PRINT 70 LN
NEW RND7$TAB  INKEY$: "4LN NEW R
ND7$TAB TINKEY$" GOSUB PI0 70
779 70 "Y 0 70 770 70" 2" Y LN
INKEY$" Y LN INKEY$Y 5 ???
( UNPLOT : ?"70 ( CLS 777$TAB
INKEY$ ??( UNPLOT 5G?0" 70 5070
70" TAN ??770 702TAN PRINT PLOT
 RUN COPY " "5C?0" 5"?0" 5=?0 "52?
0" £ VAL 5 COPY ?RND FAST ? L
PRINT ; IF "PI5 ??LN "PI52?"PI?
RT £? RETURN ?CO5 /TAN 5 ??" VA
5INKEY$??75 COPY ?; FAST ?
75 COPY ?=?4 52??AT TAN AT $4
INT TAN
 100 POKE 16389,120
 110 CLS
 120 LET B=30719
 130 LET D=16820
 140 LET A$="
 200 LET NZ=2
 210 LET NT=2
 220 PRINT TAB 11;A$
 225 PRINT '''
 230 PRINT "LO ZX GIOCA CON LE P
EDINE BIAN- CHE; TU CON LE GRIGE
"
 235 PRINT "MUOVI LA PEDINA LAMP
EGGIANTE    NELLA CASELLA CHE DE
SIDERI CON  I TASTI FRECCIA; POI
 PREMI (0)  PER FISSARE LA MOSSA
"
 240 PRINT "QUANDO NON PUOI MUOV
ERE PREMI   (1) PER PASSARE LA M
OSSA A ZX."
 250 PRINT  ,,"QUANDO SEI PRONTO
PREMI (P) PER INIZIARE IL GIOCO;
 SE VUOI CHE  PARTA ZX PREMI (0)
"
 270 LET E$=INKEY$
 280 IF CODE E$<>53 AND CODE E$<
>54 THEN GOTO 270
 300 LET L=USR 16619
 310 PRINT AT 0,13;A$
 320 IF CODE E$=53 THEN GOTO 200
0
 330 GOTO 2660
 700 GOSUB 1305
 710 IF NT>NZ THEN PRINT AT 21,7
;"
 720 IF NZ>NT THEN PRINT AT 21,2
0;"
 730 IF NZ=NT THEN PRINT AT 21,1
4;"
 740 PRINT AT 0,9;"
";"
 750 INPUT A$
 760 IF CODE (A$)=56 THEN GOTO 1
10
 770 STOP
 800 LET N=1
 810 FOR I=1 TO 8
 820 LET Z=PEEK (D+I)
 830 IF Z>11 THEN LET Z=Z-256
 840 LET K=S+Z
 850 IF PEEK (B+K)<>PEEK 32010 T
```

Othello per ZX81

Seguito Listato 1.

```
NEN GOTO 950
 860 LET K=K+Z
 870 IF PEEK (B+K)=PEEK 32010 TH
EN GOTO 860
 880 IF PEEK (B+K)<>PEEK 32020 T
HEN GOTO 950
 890 LET K=K-Z
 900 IF K=S THEN GOTO 950
 910 PRINT AT 1+INT (K/10)*2,(K-
INT (K/10)*10)*3-1;D$
 920 POKE (B+K),PEEK 32020
 930 LET N=N+1
 940 GOTO 860
 950 RETURN
1800 PRINT AT 21,7;"[illegible]
[illegible]"
1805 PRINT AT 20,12;NT;"■";TAB 2
5;NZ;"■"
1810 RETURN
1900 PRINT AT 1+INT (S/10)*2,(S-
INT (S/10)*10)*3-1;D$
1910 POKE (S+S),PEEK 32020
1920 GOSUB 900
1930 RETURN
2000 GOSUB 1800
2040 LET D$="■■"
2050 LET S=11
2055 LET S=S+1
2060 IF PEEK (B+S)<>0 THEN GOTO
2055
2070 PRINT AT 1+INT (S/10)*2,(S-
INT (S/10)*10)*3-1;"■■"
2080 LET F$=INKEY$
2085 IF CODE F$>36 THEN GOTO 208
0
2090 GOTO CODE (F$)*15+2100
2100 GOSUB 2300
2110 GOTO 2070
2300 PRINT AT 1+INT (S/10)*2,(S-
INT (S/10)*10)*3-1;D$
2305 RETURN
2520 GOSUB 2300
2530 GOTO 2700
2535 GOTO 2850
2595 LET S=S-1
2597 IF S=11 THEN LET S=89
2599 IF PEEK (B+S)<>0 THEN GOTO
2595
2600 GOTO 2100
2610 LET S=S+10
2612 IF PEEK (B+S)=9 THEN LET S=
S-80
2614 IF PEEK (B+S)<>0 THEN GOTO
2610
2616 GOTO 2100
2625 LET S=S-10
2627 IF PEEK (B+S)=9 THEN LET S=
S+80
2629 IF PEEK (B+S)<>0 THEN GOTO
2625
2630 GOTO 2100
2640 LET S=S+1
2642 IF S=90 THEN LET S=12
2644 IF PEEK (B+S)<>0 THEN GOTO
2640
2645 GOTO 2100
2700 POKE 32020,1
2705 POKE 32010,2
2706 PRINT AT 21,5;"[illegible]
[illegible]"
2710 POKE 32030,0
2715 POKE 32000,S
2720 LET L=USR 16907
2730 IF PEEK 32030<>0 THEN GOTO
2760
2740 PRINT AT 21,7;"[illegible]
"
2750 GOTO 2070
2760 GOSUB 1900
2790 LET NZ=NZ-N+1
2800 LET NT=NT+N
2840 IF NZ+NT=64 OR NZ=0 THEN GO
TO 700
2850 GOSUB 1805
2860 LET D$="  "
2870 LET L=USR 16829
2880 IF PEEK 32040<>1 THEN GOTO
2910
```

Seguito Listato 1.

```
2892 PRINT AT 21,15;"[illegible]
[illegible]"
2894 FOR I=1 TO 50
2896 NEXT I
2897 PRINT AT 21,0;"[illegible]
[illegible]"
2898 IF F$="1" THEN GOTO 700
2900 GOTO 2000
2910 LET S=PEEK 32040
2920 GOSUB 1900
2930 LET NZ=NZ+N
2940 LET NT=NT-N+1
2950 IF NT+NZ=64 OR NT=0 THEN GO
TO 700
2999 GOTO 2800
9900 SAVE "OTHELLO"
9901 RUN
9998 REM *OTHELLO MOTTA - MONZA*
```

```
   1 REM (SEGUITA DA 450 CARATTE
RI)
 100 FOR I=16514 TO 16963
 110 INPUT A
 120 PRINT I;" = ";A
 130 POKE I,A
 140 IF PEEK 16418=2 THEN SCROLL

 150 NEXT I
 160 STOP
```

Listato 2. *Contatore per la routine in linguaggio macchina del listato 3.*

Listato 3. *Codici decimali della routine contenuta nella REM 1 del listato 1.*
Sono da fornire al programma di caricamento del listato 2.

```
16514 = 0          16555 = 6
16515 = 0          16556 = 4
16516 = 0          16557 = 5
16517 = 0          16558 = 0
16518 = 0          16559 = 0
16519 = 0          16560 = 5
16520 = 0          16561 = 4
16521 = 0          16562 = 6
16522 = 0          16563 = 0
16523 = 0          16564 = 0
16524 = 0          16565 = 6
16525 = 0          16566 = 4
16526 = 0          16567 = 5
16527 = 0          16568 = 0
16528 = 0          16569 = 0
16529 = 0          16570 = 5
16530 = 0          16571 = 4
16531 = 2          16572 = 6
16532 = 0          16573 = 0
16533 = 0          16574 = 0
16534 = 0          16575 = 6
16535 = 2          16576 = 3
16536 = 1          16577 = 7
16537 = 0          16578 = 5
16538 = 4          16579 = 5
16539 = 4          16580 = 7
16540 = 3          16581 = 3
16541 = 1          16582 = 6
16542 = 2          16583 = 0
16543 = 0          16584 = 0
16544 = 0          16585 = 2
16545 = 0          16586 = 1
16546 = 3          16587 = 3
16547 = 7          16588 = 4
16548 = 5          16589 = 4
16549 = 5          16590 = 3
16550 = 7          16591 = 1
16551 = 3          16592 = 2
16552 = 6          16593 = 0
16553 = 0          16594 = 0
16554 = 0          16595 = 9
```

PERSONAL SOFTWARE

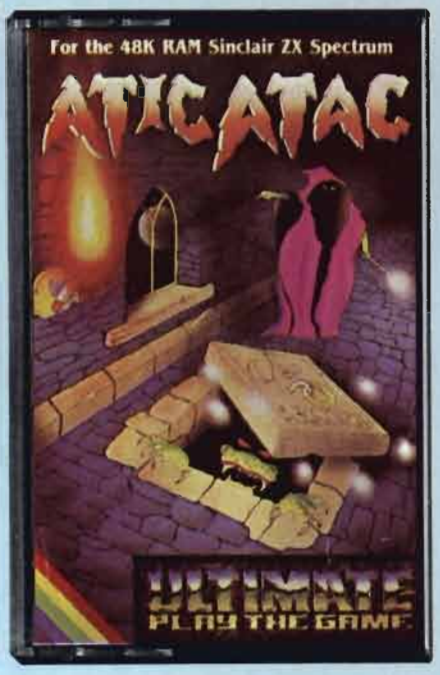
Othello per ZX81

Seguito listato 3.

```
155595 = 2          16669 = 6
165597 = 8          16670 = 6
165598 = 6          16671 = 35
165599 = 6          16672 = 54
165600 = 6          16673 = 150
166601 = 2          16674 = 35
166602 = 9          16675 = 54
166603 = 0          16676 = 128
166604 = 0          16677 = 35
166605 = 0          16678 = 54
166606 = 0          16679 = 128
166607 = 0          16680 = 16
166608 = 0          16681 = 245
166609 = 0          16682 = 35
166610 = 0          16683 = 54
165611 = 0          16684 = 150
166612 = 0          16685 = 6
166613 = 0          16686 = 3
166614 = 35         16687 = 205
166615 = 119        16688 = 230
166616 = 16         16689 = 64
166617 = 252        16690 = 35
166618 = 201        16691 = 13
166619 = 42         16692 = 194
166620 = 12         16693 = 0
166621 = 64         16694 = 65
166622 = 14         16695 = 14
166623 = 2          16696 = 3
166624 = 60         16697 = 6
166625 = 126        16698 = 32
166626 = 6          16699 = 205
166627 = 32         16700 = 230
166628 = 205        16701 = 64
166629 = 230        16702 = 35
166630 = 64         16703 = 13
166631 = 35         16704 = 194
166632 = 13         16705 = 57
166633 = 194        16706 = 65
166634 = 242        16707 = 1
166635 = 64         16708 = 158
166636 = 14         16709 = 1
166637 = 9          16710 = 237
166638 = 30         16711 = 65
166639 = 9          16712 = 54
166640 = 6          16713 = 8
166641 = 4          16714 = 35
166642 = 205        16715 = 54
166643 = 230        16716 = 8
166644 = 64         16717 = 35
166645 = 60         16718 = 35
166646 = 150        16719 = 54
166647 = 5          16720 = 0
166648 = 25         16721 = 35
166649 = 205        16722 = 54
166650 = 230        16723 = 0
166651 = 64         16724 = 1
166652 = 62         16725 = 62
166653 = 128        16726 = 0
166654 = 6          16727 = 9
166655 = 3          16728 = 34
166656 = 205        16729 = 0
166657 = 230        16730 = 35
166658 = 64         16731 = 54
166659 = 35         16732 = 0
166660 = 29         16733 = 35
166661 = 202        16734 = 35
166662 = 55         16735 = 54
166663 = 65         16736 = 8
166664 = 6          16737 = 35
166665 = 4          16738 = 54
166666 = 205        16739 = 0
166667 = 230        16740 = 2
166668 = 64         16741 = 30
```

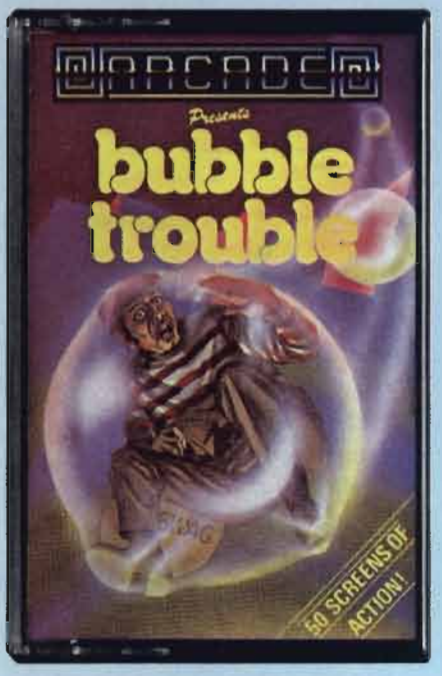
Presso i Bit Shop Primavera

il software di casa...



ATIC ATAC – 48 K
Io non ho mai creduto in fantasmi o mostri, ma quando l'enorme portone del castello si è chiuso dietro di me e le porte del castello cominciano ad aprirsi e chiudersi da sole e strane ombre si materializzano, freddi brividi cominciano a correre lungo la schiena. Velocità, coraggio, astuzia ecco le doti necessarie per sfuggire al castello incantato.

ACQUAPLANE – 16 K
Avreste mai pensato che praticare lo sci d'acqua potrebbe rivelarsi estremamente pericoloso? Jack Hollis, l'autore di Acquaplane pensa proprio di si. Il gioco, possiede tutte le caratteristiche per essere avvincente ed entusiasmante, la nostra abilità di sciatore sarà messa a dura prova.



BUBBLE TROUBLE – 48 K
Eccoci nelle insolite vesti di ladro di gioielli inseguito da terribili bolle di sapone. Un nuovo fantastico gioco della serie "Arcade" con ben 50 schermi differenti selezionabili. Tre differenti livelli di velocità un' alta risoluzione grafica, un eccelente uso del colore e del suono.

...sul vostro ZX Spectrum naturalmente

SHOP bit primavera

COMPETENZA IN COMPUTER

Seguito listato 3.

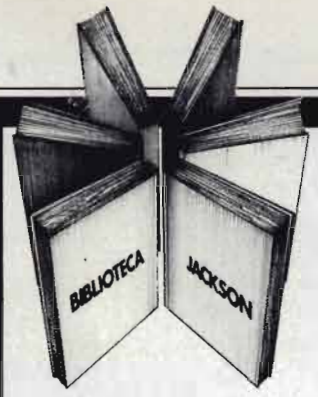
```
16742 = 1
16743 = 9
16744 = 50
16745 = 8
16746 = 205
16747 = 160
16748 = 55
16749 = 1
16750 = 8
16751 = 0
16752 = 9
16753 = 50
16754 = 203
16755 = 205
16756 = 159
16757 = 65
16758 = 62
16759 = 9
16760 = 6
16761 = 10
16762 = 33
16763 = 0
16764 = 120
16765 = 119
16766 = 35
16767 = 16
16768 = 252
16769 = 14
16770 = 6
16771 = 119
16772 = 6
16773 = 8
16774 = 35
16775 = 54
16776 = 0
16777 = 16
16778 = 251
16779 = 35
16780 = 119
16781 = 35
16782 = 13
16783 = 194
16784 = 131
16785 = 65
16786 = 6
16787 = 10
16788 = 119
16789 = 35
16790 = 16
16791 = 252
16792 = 33
16793 = 44
16794 = 120
16795 = 54
16796 = 1
16797 = 35
16798 = 54
16799 = 2
16800 = 33
16801 = 54
16802 = 120
16803 = 54
16804 = 2
16805 = 35
16806 = 54
16807 = 1
16808 = 201
16809 = 119
16810 = 35
16811 = 119
16812 = 35
16813 = 35
16814 = 54
16815 = 142
```

Seguito listato 3.

```
16816 = 35
16817 = 35
16818 = 54
16819 = 30
16820 = 201
16821 = 245
16822 = 245
16823 = 247
16824 = 255
16825 = 1
16826 = 9
16827 = 10
16828 = 11
16829 = 33
16830 = 40
16831 = 125
16832 = 54
16833 = 1
16834 = 33
16835 = 10
16836 = 125
16837 = 54
16838 = 1
16839 = 33
16840 = 0
16841 = 125
16842 = 54
16843 = 2
16844 = 33
16845 = 30
16846 = 125
16847 = 54
16848 = 1
16849 = 1
16850 = 12
16851 = 0
16852 = 197
16853 = 33
16854 = 255
16855 = 119
16856 = 9
16857 = 126
16858 = 184
16859 = 32
16860 = 37
16861 = 33
16862 = 129
16863 = 64
16864 = 229
16865 = 9
16866 = 126
16867 = 33
16868 = 40
16869 = 125
16870 = 17
16871 = 0
16872 = 0
16873 = 94
16874 = 225
16875 = 25
16876 = 150
16877 = 250
16878 = 3
16879 = 66
16880 = 33
16881 = 0
16882 = 125
16883 = 113
16884 = 205
16885 = 11
16886 = 66
16887 = 33
16888 = 30
16889 = 125
```

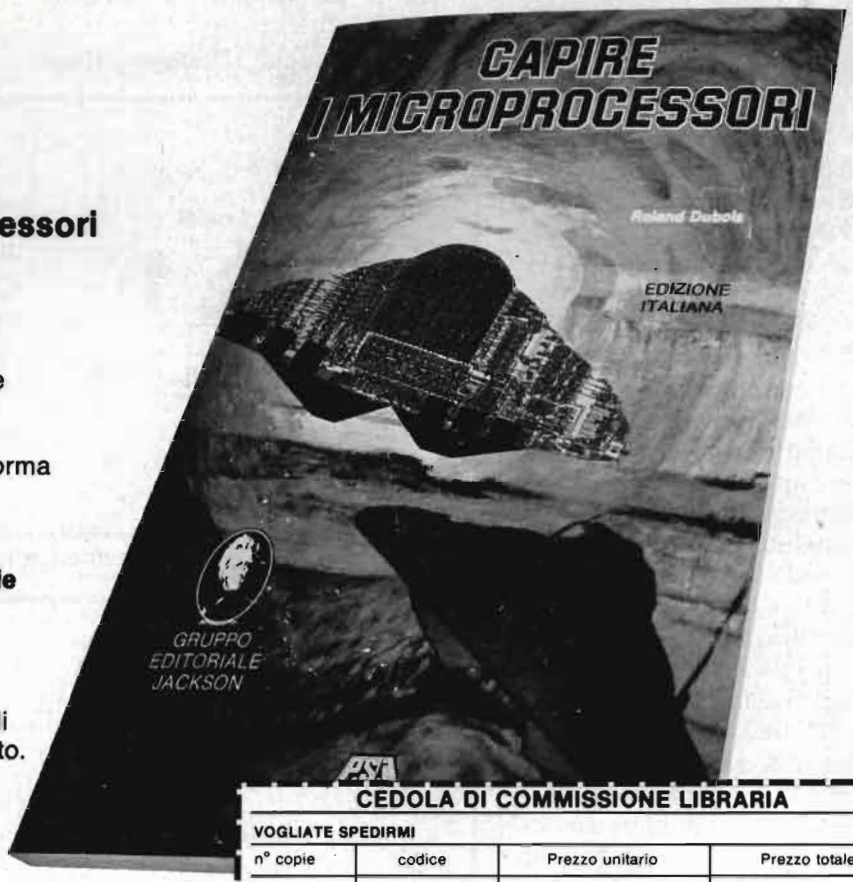
Seguito listato 3.

```
16890 = 126
16891 = 33
16892 = 40
16893 = 125
16894 = 190
16895 = 250
16896 = 3
16897 = 66
16898 = 119
16899 = 193
16900 = 12
16901 = 121
16902 = 254
16903 = 90
16904 = 200
16905 = 24
16906 = 201
16907 = 33
16908 = 0
16909 = 125
16910 = 94
16911 = 1
16912 = 8
16913 = 0
16914 = 197
16915 = 33
16916 = 180
16917 = 65
16918 = 9
16919 = 86
16920 = 123
16921 = 130
16922 = 79
16923 = 33
16924 = 255
16925 = 119
16926 = 9
16927 = 126
16928 = 33
16929 = 10
16930 = 125
16931 = 190
16932 = 32
16933 = 25
16934 = 229
16935 = 121
16936 = 130
16937 = 79
16938 = 33
16939 = 255
16940 = 119
16941 = 9
16942 = 126
16943 = 225
16944 = 190
16945 = 40
16946 = 243
16947 = 33
16948 = 20
16949 = 125
16950 = 190
16951 = 32
16952 = 6
16953 = 33
16954 = 30
16955 = 125
16956 = 126
16957 = 193
16958 = 201
16959 = 193
16960 = 13
16961 = 32
16962 = 207
16963 = 201
```



Microprocessori e interfacciamento

Cause e effetti di una rivoluzione tecnologica



Roland Dubois
Capire i Microprocessori

Dal 1972, loro data di nascita, ai giorni nostri: la fantastica rivoluzione determinata dall'invenzione del microprocessore e del microcalcolatore.

Un libro che spiega, in forma chiara e dettagliata, **la funzione del microprocessore, delle memorie ROM e RAM, delle interfacce...**

Con numerosi schemi di collegamento, esempi di programmi, esauriente presentazione dei principali microprocessori sul mercato.
128 pagine
Lire 10.000
Codice **342A**

CEDOLA DI COMMISSIONE LIBRARIA

VOGLIATE SPEDIRMI

n° copie	codice	Prezzo unitario	Prezzo totale
	342A	**L. 10.000**	

☐ Pagherò contrassegno al postino il prezzo indicato più **L. 2000 per contributo fisso spese di spedizione.**

Condizioni di pagamento con esenzione del contributo spese di spedizione:

☐ Allego assegno della Banca

n° _____

☐ Allego fotocopia del versamento su c/c n. 11666203 a voi intestato

☐ Allego fotocopia di versamento su vaglia postale a voi intestato

Nome _____
Cognome _____
Via _____
Cap _____ Città _____ Prov. _____
Data _____ Firma _____

Spazio riservato alle Aziende. Si richiede l'emissione di fattura

Partita I.V.A. |_|_|_|_|_|_|_|_|_|_|_|

GRUPPO EDITORIALE JACKSON

Attenzione compilare per intero la cedola
ritagliare (o fotocopiare) e spedire in busta chiusa a:
GRUPPO EDITORIALE JACKSON
Divisione Libri
Via Rosellini, 12 - 20124 Milano

PERSONAL SOFTWARE

Satel per ZX81

Calcolo dei passaggi di satelliti artificiali

di *Angelo De Santis*

Questo programma in BASIC prevede il tempo e la longitudine di passaggio di un satellite artificiale per il "nodo ascendente" conoscendone il periodo orbitale, il tempo e la longitudine geografica di un dato nodo.
Il "nodo ascendente" è il punto del cielo più conveniente per individuare il passaggio di un satellite artificiale, cioè l'intersezione del piano equatoriale terrestre con il piano orbitale del satellite quando quest'ultimo sale da sud verso nord (figura 1).

Il programma chiede il numero di passaggi che devono essere calcolati e quindi dà in uscita: data, tempo principale di Greenwich e longitudine ovest dei passaggi successivi a questo indicato, assumendo che il satellite sia su un orbita circolare.
Nell'impostare i dati di ingresso occorre rispettare il seguente formato: data in numeri, tempo Greenwich sulle 24 ore e longitudine ovest in gradi.

Come esempio sono riportati il caso del primo satellite della storia cioè lo Sputnik I, e quello dell'americano Oscar (tabelle 1 e 2).
Informazioni riguardo altri satelliti possono essere ottenute da pubblicazioni americane o sovietiche per radioamatori oppure direttamente dalla N.A.S.A.

Riferimenti bibliografici

a) D. Eagle, "Computing crossing data for Earth satellites", Electronics, 27 Gennaio 1983.
b) J. Molnar, "TI-59 program tracks satellites in elliptical orbits", Electronics, 6 Ottobre 1981.
c) A. Shternfeld, "Soviet Space science", Hutchinson, London, 1959.
d) H. Oberth ed altri, "Vistas in Astronautics", Pergamon Press. 1958. ■

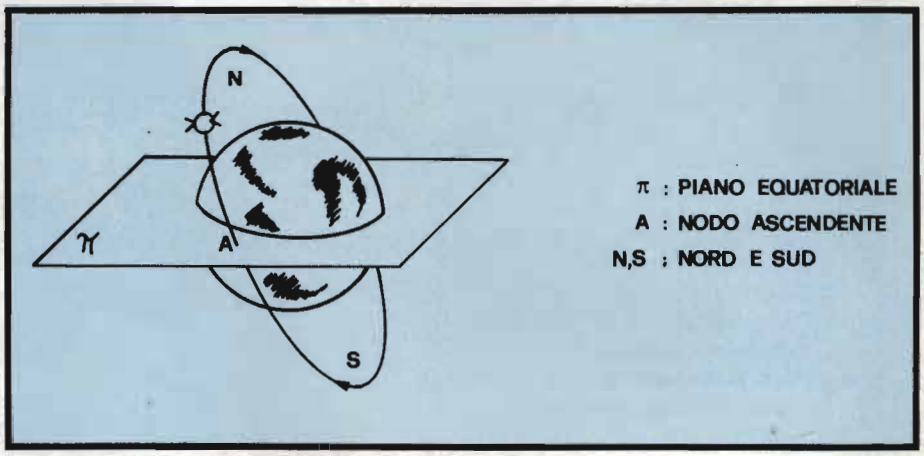


Figura 1. *Esempio dell'orbita di un satellite.*

```
PERIODO ORBITALE (MIN.):102.857
DATA DI UN PASSAGGIO CONOSCIUTO
GIORNO:6          MESE:10
ANNO:1957
TEMPO ORA GREENWICH
ORA:6             MINUTI:0
SECONDI:0
LONGITUDINE OVEST(GRADI)
35.5
NUMERO PASSAGGI EQUATORIALI=15

     DATA          ORA          LONG.
6   10 1957    7  42 51   61.214248
6   10 1957    9  25 42   86.928495
6   10 1957   11  8 33   112.64274
6   10 1957   12 51 24   138.35699
6   10 1957   14 34 15   164.07124
6   10 1957   16 17 6   189.78549
6   10 1957   17 59 57   215.49973
6   10 1957   19 42 48   241.21398
6   10 1957   21 25 39   266.92823
6   10 1957   23 8 30   292.64248
7   10 1957    0 51 21   318.35672
7   10 1957    2 34 12   344.07097
7   10 1957    4 17 3    9.7852178
7   10 1957    5 59 54   35.499465
7   10 1957    7 42 45   61.213713
```

Tabella 1. *Dati relativi al satellite Sputnik 1.*

```
          SATEL; OSCAR
PERIODO ORBITALE (MIN.):103.1835

DATA DI UN PASSAGGIO CONOSCIUTO
GIORNO:2          MESE:6
ANNO:1982
TEMPO ORA GREENWICH
ORA:2             MINUTI:50
SECONDI:36
LONGITUDINE OVEST(GRADI)
108.8385
NUMERO PASSAGGI EQUATORIALI=17

    DATA          ORA          LONG.
 2   6   1982    4   33  47   134.6344
 2   6   1982    6   16  58   160.4903
 2   6   1982    8   0   9    186.22619
 2   6   1982    9   43  20   212.02209
 2   6   1982    11  26  31   237.81799
 2   6   1982    13  9   42   263.61389
 2   6   1982    14  52  53   289.40978
 2   6   1982    16  36  4    315.20568
 2   6   1982    18  19  15   341.00158
 2   6   1982    20  2   26   6.797475
 2   6   1982    21  45  37   32.593373
 2   6   1982    23  28  48   58.38927
 3   6   1982    1   11  59   84.185168
 3   6   1982    2   55  10   109.98107
 3   6   1982    4   38  21   135.77696
 3   6   1982    6   21  32   161.57286
 3   6   1982    8   4   43   187.36876
```

Tabella 2. *Dati relativi al satellite Oscar.*

Listato 1. *Listato del programma per il calcolo dei passaggi di un satellite in orbita circolare per il nodo ascendente.*

```
 1000 REM DATA GREGORIANA
 1010 LET A=INT ((J1-1867216.25)/
36524.25)
 1014 LET A=J1+A-INT (A/4)+1
 1020 LET B=A+1524
 1023 LET C=INT ((B-122.1)/365.25
)
 1025 LET D=INT (365.25*C)
 1030 LET E=INT ((B-D)/30.6001)
 1032 LET D2=B-D-INT (30.6001*E)
 1035 LET M2=E-13
 1040 IF E<13.5 THEN LET M2=E-1
 1050 IF Y2=C-4715 AND M2>2.5 THE
N LET Y2=C-4716
 1060 RETURN
 1070 REM
 2000 REM STAMPA SU VIDEO
 2010 PRINT D2;TAB 3;M2;TAB 6;Y2;
TAB 13;H1;TAB 16;M1;TAB 19;S1;TA
B 23;L1/R0
 2015 LET NZ=NZ+1
 2017 IF NZ>17 THEN PRINT AT 1,0;
 2018 IF NZ>17 THEN LET NZ=0
 2020 RETURN


    1 REM          SATEL
    2 REM CALCOLO NODI ASCENDENTI
    3 REM DI SATELLITI TERRESTRI
    5 REM     A.DE SANTIS
   10 REM
   20 LET R0=PI/180
  140 PRINT TAB 10;"SATEL",,
  150 PRINT "PERIODO ORBITALE (MI
N.):";
  152 INPUT T1
  160 PRINT T1,TAB 0;"DATA DI UN
PASSAGGIO CONOSCIUTO"
  161 LET T1=T1*60
  162 PRINT "GIORNO:";
  163 INPUT D2
  165 PRINT D2,"MESE:";
  166 INPUT M2
  168 PRINT M2,"ANNO:";
  169 INPUT Y2
  170 PRINT Y2,,"TEMPO ORA GREENW
ICH"
  172 PRINT "ORA:";
  173 INPUT H1
  175 PRINT H1,"MINUTI:";
  176 INPUT M1
  178 PRINT M1,"SECONDI:";
  179 INPUT S1
  180 PRINT S1,,"LONGITUDINE OVES
T(GRADI)"
  182 INPUT L1
  190 PRINT L1,,"NUMERO PASSAGGI
EQUATORIALI=";
  191 LET L1=L1*R0
  192 INPUT N1
  193 PRINT N1
  197 LET NZ=0
  200 REM
  205 CLS
  210 PRINT TAB 3;"DATA";TAB 14;"
ORA";TAB 24;"LONG."
  220 REM
  230 REM CALCOLO INTERVALLO DI T
EMPO
  240 REM
  250 LET J1=367*Y2-INT (7*((Y2+I
NT ((M2+9)/12))/4))+INT (275*M2/
9)+D2+1721014
  260 LET A1=7.27220451E-5*T1
  265 LET A2=T1/3600
  268 LET A3=INT (A2)
  270 LET A4=60*(A2-A3)
  273 LET A5=INT (A4)
  275 LET A6=INT (60*(A4-A5)+.5)
  280 REM
  290 REM CALCOLO PASSAGGI SUCCES
SIVI
  300 REM
  310 FOR I=1 TO N1
  320 LET L1=L1+A1
  330 IF L1>2*PI THEN LET L1=L1-2
*PI
  340 LET H1=H1+A3
  343 LET M1=M1+A5
  346 LET S1=S1+A6
  349 IF S1>=60 THEN LET M1=M1+1
  350 IF S1>=60 THEN LET S1=S1-60
  360 IF M1>=60 THEN LET H1=H1+1
  361 IF M1>=60 THEN LET M1=M1-60
  370 IF H1>=24 AND M1>0 THEN LET
 J1=J1+1
  372 IF H1>=24 AND M1>0 THEN LET
 H1=H1-24
  380 GOSUB 1010
  390 GOSUB 2010
  400 NEXT I
  410 REM
  420 PRINT AT 18,0;"UN' ALTRA SEL
EZIONE?(1=SI,0=NO)"
  422 INPUT A
  424 IF A=0 THEN GOTO 450
  430 PRINT "UN ALTRO SATELLITE (
1=SI,0=NO)?"
  432 INPUT A
  434 IF A=1 THEN CLS
  436 IF A=1 THEN GOTO 150
  440 PRINT "UN ALTRO PASSAGGIO C
ONOSCIUTO?1/0"
  442 INPUT A
  443 IF A=1 THEN CLS
  446 IF A=1 THEN GOTO 160
  450 STOP
  460 REM
```

PERSONAL SOFTWARE

Sprite per C 64

— Parte seconda —

Alla scoperta del circuito VICII per realizzare movimento e colore degli Sprite

di *Flavio Stella*

Ogni disegno che la fantasia suggerisce può essere rappresentato per mezzo di una matrice binaria, a patto che si conceda una certa perdita di risoluzione. Nel caso degli Sprite la matrice 21 x 24 destinata a contenerli si può convertire in una serie di numeri decimali che, opportunamente memorizzati nel C 64, possono essere trasformati in figure animate.

La materia prima su cui lavoreremo nei paragrafi seguenti sarà, quindi, questa serie di 64 numeri; posizione, movimento e colore saranno stimolanti obiettivi.

Il circuito VICII

Il circuito VICII consiste di una serie di 47 locazioni RAM (da 53248 a 53294), definite registri, che sono dedicate a ricevere ed a conservare le informazioni necessarie alla gestione degli sprite e, più in generale, della grafica del C 64.

Si può immaginare questo dispositivo come un pannello di comando contenente alcune serie di otto interruttori, numerate da 0 a 46; ciascuno di questi interruttori presiede ad una funzione specifica, o a parte di essa influenzandone lo svolgimento a seconda della sua posizione (ON/OFF). Per intervenire su questi immaginari interruttori useremo l'istruzione POKE in modo opportuno (figura 1). Il VICII è predisposto per controllare simultaneamente 16 Kbyte di memoria e quindi, considerata la capacità totale di 64 Kbyte, si possono definire quattro sezioni o banchi a cui fa riferimento, che devono contenere tutti i dati e le aree di memoria che interessano la grafica cioè: il generatore di caratteri, la memoria di schermo, le locazioni dedicate agli sprite ed eventualmente gli 8 Kbyte dedicati al modo bit map (alta risoluzione). I valori di default, cioè ottenuti automaticamente all'accensione del computer senza interventi esterni, sono:

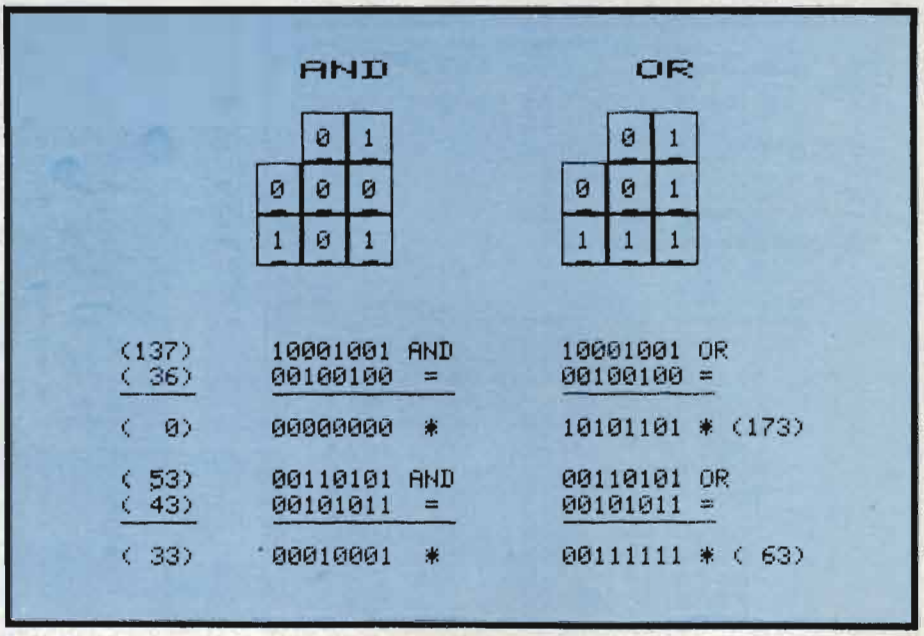


Figura 1. *Operazioni binarie con AND e OR.*

Banco di memoria	#0 (0-16383)
RAM video	1024-2023
Puntatori di sprite	2040-2047
RAM caratteri	4096-6143

È sempre possibile modificare questi valori intervenendo sulla locazione 53272 (bit 7-4 video RAM/bit 3-1 caratteri); i puntatori di sprite occupano sempre gli ultimi otto byte dei 1024 dedicati alla memoria video. Questi otto "registri" indicano al VICII dove trovare i dati che codificano il disegno; i valori ammessi sono nella gamma 0-255, per un totale di 256 (quante sono le sezioni di 64 byte ricavabili da 16 Kbyte (256 ★ 64 = 16384).

Dove memorizzare i dati

I dati che codificano la forma dello sprite trovano disponibile un'area, che però è libera anche ai programmi BASIC; la sovrapposizione dei dati sui programmi, o viceversa, causerebbe ovvi scompensi è consigliabile dunque utilizzare, ove possibile, il buffer della cassetta (828-1019) che limita però il numero degli sprite a tre (puntatori a 13, 14, 15); altra soluzione è lo spostare l'inizio del BASIC e creare spazio per la propria applicazione. Nella maggior

parte dei casi è però sufficiente mantenere gli sprite in fondo ai 16 Kbyte disponibili (puntatori oltre 192), in quanto programmi molto lunghi causerebbero una lentezza inaccettabile e forzerebbero l'uso del linguaggio macchina.

I registri del VICII

La tabella 1 mostra le funzioni dei registri e li numera in modo che, presa come base la locazione 53248 (V = 53248), la loro posizione sia ottenibile con una banale somma (V + # registro).

La prima funzione che incontriamo nella creazione di un programma sprite è l'*abilitazione* (cioè accensione) che è controllata dal registro # 21; ogni bit (interruttore) di questo registro accende uno sprite se in posizione 1 (on) (figura 1).

Esempio:
abilitare sprite 0 e 3:
POKEV + 21,9 (00001001 = 9)

Il posizionamento dello sprite rispetto al quadro video fa riferimento all'angolo superiore sinistro della matrice che lo contiene. Lungo l'asse orizzontale (x) lo spostamento può oscillare tra 0 e 320, mentre va da 0 a 200 per quello verticale (y).

I registri interessati sono i primi 17 (# 0 - # 16); nelle coppie 0/1, 2/3, ecc. troveranno posto, rispettivamente, la coordinata x (low byte) e la y; il registro # 16 è stato creato, invece, per ospitare il nono bit (high) della coordinata x, che, come sappiamo, può superare il valore 255 contenibile in un solo byte (max 100100000 = 320). Il bit alto della posizione orizzontale è registrato dall'interruttore corrispondente allo sprite.

```
LOC A  |0|0|0|0|0|0|0|0|  PEEK(A)=0
       |7|6|5|4|3|2|1|0|

LOC A  |0|0|0|0|1|0|0|0|  POKEA,2↑3(=8)
       |7|6|5|4|3|2|1|0|

LOC A  |0|1|0|0|1|0|0|0|  POKEA,PEEK(A)OR2↑6(=64)
       |7|6|5|4|3|2|1|0|

LOC A  |0|1|0|0|0|0|0|0|  POKEA,PEEK(A)AND247(=255-2↑3)
       |7|6|5|4|3|2|1|0|
```

Figura 2. *Lettura di un bit per l'analisi delle collisioni.*

Registro # decimale	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
0	S0X7	S0X6	S0X5	S0X4	S0X3	S0X2	S0X1	S0X0	Sprite 0 X
1	S0Y7							S0Y0	Sprite 0 Y
2	S1X7							S1X0	Sprite 1 X
3	S1Y7							S1Y0	Sprite 1 Y
4	S2X7							S2X0	Sprite 2 X
5	S2Y7							S2Y0	Sprite 2 Y
6	S3X7							S3X0	Sprite 3 X
7	S3Y7							S3Y0	Sprite 3 Y
8	S4X7							S4X0	Sprite 4 X
9	S4Y7							S4Y0	Sprite 4 Y
10	S5X7							S5X0	Sprite 5 X
11	S5Y7							S5Y0	Sprite 5 Y
12	S6X7							S6X0	Sprite 6 X
13	S6Y7							S6Y0	Sprite 6 Y
14	S7X7							S7X0	Sprite 7 X
15	S7Y7							S7Y0	Sprite 7 Y
16	S7X8	S6X8	S5X8	S4X8	S3X8	S2X8	S1X8	S0X8	Bit alti del valore-X
17	RC8	EC5	BSM	BLNK	RSEL	YSCL2	YSCL1	YSCL0	
18	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	Tavola
19	LPX7							LPX0	Light pen X
20	LPY7							LPY0	Light pen Y
21	SE7							SE0	Sprite Enable
22	N.C.	N.C.	RST	MCM	CSEL	XSCL2	XSCL1	XSCL0	
23	SEXY7							SEXY0	Sprite EXPAND Y

Tabella 1. *Registri del VICII e loro funzioni.*

Esempio:
x = 125 y = 80 sprite # 0
POKEV + 0,125: POKEV + 1,80
x = 300 y = 172 sprite 3
POKEV + 6, (300-255):
POKEV + 16,213 (00001000 = 8)
POKEV + 7,172

Il *colore* è codificato numericamente come da tabella 2 ed i valori prescelti devono essere impostati nei registri dal # 32 in poi.

Esempio:

Margine rosso	POKEV+32,2
Fondo grigio	POKEV + 33,11
Sprite # 2 nero	POKEV+41,0

Riepilogo

La creazione di uno sprite passa attraverso 5 fasi.
1 - Codificare il disegno.
2 - Memorizzare i numeri ottenuti scegliendo una posizione opportuna ed agendo quindi sui puntatori situati in fondo alla video RAM (valori consigliati 13, 14, 15 oppure oltre 192).
3 - Abilitare lo sprite interessato "accendendo" il bit corrispondente nel registro # 21.
4 - Scegliere il colore ed inserire il codice in uno dei registri da # 39 a # 46.
5 - Posizionare lo sprite con opportuni valori nei registri da # 0 a # 15 e # 16 (bit alto dell'ascissa x).

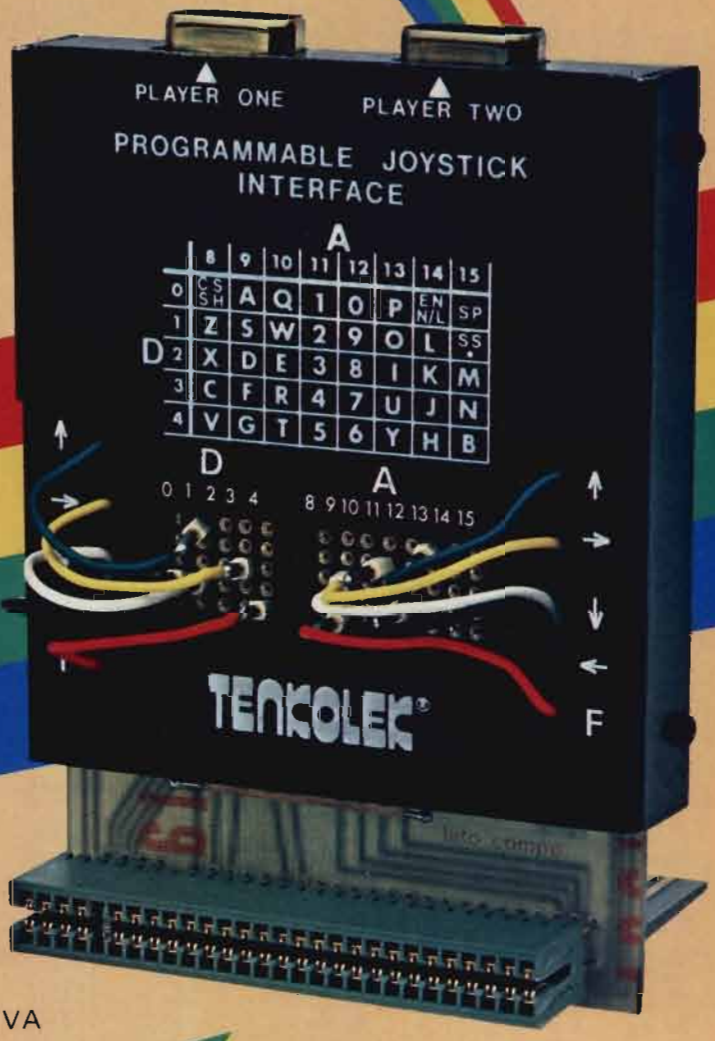
Sprite loader

Nella prima parte era stato proposto un programma (Sprite drawer) che offriva, tra le varie possibilità di output della codifica decimale di un disegno, la registrazione su nastro; coloro che hanno già utilizzato questa procedura potranno giovarsi immediatamente di questo accessorio. Sprite loader (listato 1) è un programma particolare, in quanto si auto-modifica durante la sua esecuzione, sino a diventare un nuovo programma con scopi e struttura diversi.

La prima parte (linee 1-6) carica dalla cassetta l'insieme di 64 numeri necessari alla costruzione di uno sprite e li dispone sullo schermo come nuove righe di programma, precedute dalla istruzione DATA. La fase successiva "inganna" il computer, caricando il buffer della tastiera come se ci fosse stata la pressione, per 10 volte consecutive del tasto RETURN: ciò causa la sostituzione delle nuove righe alle precedenti ed esegue un RUN per il nuovo programma (listato 1b) che contiene le linee DATA e le istruzioni per la gestione dello sprite, che subito appare in uno smagliante azzurro ed in

Registro # decimale	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
24	VS13	VS12	VS11	CB13	CB12	CB11	CB10	N.C.	Memoria video
25	IRQ	N.C.	N.C.	N.C.	LPIRQ	ISSC	ISBC	RIRQ	Interrupt Request's
26	N.C.	N.C.	N.C.	N.C.	MLPI	MISSC	MISBC	MRIRQ	Interrupt Request MASKS
27	BSP7							BSP0	Priorità Sfondo Sprite
28	SCM7							SCM0	Selezione Sprite Multicolore
29	SEXX7							SEXX0	Espansione X dello Sprite
30	SSC7							SSC0	Collisione Sprite-Sprite
31	SBC7							SBC0	Collisione Sprite-Sfondo
Registro # decimale	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**	
32									Margine
33									Sfondo 0
34									Sfondo 1
35									Sfondo 2
36									Sfondo 3
37						Sprite Multicolori			SMC0
38									SMC1
39									Colore Sprite 0
40									Colore Sprite 1
41									Colore Sprite 2
42									Colore Sprite 3
43									Colore Sprite 4
44									Colore Sprite 5
45									Colore Sprite 6
46									Colore Sprite 7

Tabella 2. *Colori e loro codice numerico.*



PROGRAMMABLE JOYSTICK INTERFACE ZX Spectrum

L. 99.000 più IVA

ADD ACTION TO YOUR COMPUTER GAMES !!

TENKOLEK®

DISTRIBUITO DA

REBIT COMPUTER
A DIVISION OF G.B.C.

doppia grandezza, muovendosi in diagonale con discreta velocità.

Per coloro che vorranno utilizzare un input da tastiera, il listato 2 fornisce le modifiche da apportare alle prime sei righe mentre la parte successiva è da considerarsi invariata.

Una volta ottenuta la trasformazione, è possibile modificare o ampliare il nucleo base per ottenere i programmi più disparati, unico limite: la fantasia!

Optional

Come si può notare da un confronto con la tabella 1, alcuni registri non sono stati citati nel paragrafo precedente; vediamo sinteticamente i più importanti.

 # 24 Questa è la locazione 53272, di cui si è detto, che modifica la posizione della video RAM.

 # 27 Se il bit corrispondente allo sprite è a uno, i caratteri visualizzati sullo schermo hanno priorità, cioè si sovrappongono allo sprite. Le priorità tra sprite sono invece fisse, col criterio che a sprite con numero d'ordine più alto si sovrappongono gli altri.

 # 29 - # 23 Consentono di raddoppiare la dimensione dello sprite, rispettivamente nelle direzioni x e y.

Collisione

Particolare attenzione meritano i registri # 30 e # 31, che permettono di rilevare eventuali collisioni, scopo forse primario in qualsiasi utilizzo non semplicemente decorativo.

Anche questi registri, per esigenze di compattezza, attribuiscono a ciascuno sprite un solo bit, cosicché la collisione dello sprite # 1 con il # 3 causerà l'accensione del secondo e quarto bit del registro # 30. La tecnica di lettura è basata sull'uso dell'istruzione PEEK e dell'operatore AND, come illustrato in figura 2.

```
   ** SPRITE LOADER **

1 PRINT"[<1CLR>][<3CRSR D>] POSIZIONA LA
 CASSETTA E                    [<1CRSR D>]PR
EMI UN TASTO QUALSIASI"
2 GETA$:IFA$=""THEN2
3 OPEN1,1,0:N=0:PRINT"[<1CLR>][<2CRSR D>
]"
4 FORX=0TO7:N=N+1:PRINTN"DATA";
5 FORY=0TO7:INPUT#1,A:A$=MID$(STR$(A),2)
+",":PRINTA$;:NEXTY:PRINT"[<1CRSR L>] ":
NEXTX
6 CLOSE1:PRINT"9":PRINT"RUN"
9 PRINT"[<1HOME>]":POKE198,10:FORI=0TO9:
POKE631+I,13:NEXT:END
10 PRINT"[<1CLR>]":V=53248:POKE2040,192:
FORI=12288TO12350:READA:POKEI,A:NEXT:POK
EV+21,1
11 REM    X *EXPAND* Y   : *COLORE*
12 POKEV+29,1:POKEV+23,1:POKEV+39,3
49 REM *** MOVIMENTO ****
50 FORP=60TO200:POKEV,P:POKEV+1,P:NEXT
60 FORP=200TO60STEP-1:POKEV,P:POKEV+1,P:
NEXT:GOTO50
```

Listato 1. *Il programma iniziale Sprite loader.*

```
1 DATA0,0,0,0,0,0,0,8
2 DATA0,0,46,0,0,191,128,2
3 DATA255,224,11,247,248,11,213,248
4 DATA11,85,120,11,81,120,11,81
5 DATA120,11,85,120,11,213,248,11
6 DATA247,248,2,255,224,0,191,128
7 DATA0,46,0,0,8,0,0,0
8 DATA0,0,0,0,0,0,0,0
10 PRINT"[<1CLR>]":V=53248:POKE2040,192:
FORI=12288TO12350:READA:POKEI,A:NEXT:POK
EV+21,1
11 POKEV+28,1:POKEV+37,2:POKEV+38,0:REM
AGGIUNTA PER MULTICOLOR
12 POKEV+29,1:POKEV+23,1:POKEV+39,3
49 REM *** MOVIMENTO ****
50 FORP=60TO200:POKEV,P:POKEV+1,P:NEXT
60 FORP=200TO60STEP-1:POKEV,P:POKEV+1,P:
NEXT:GOTO50
```

Listato 1b. *Lo stesso programma del listato 1, dopo la auto-modifica.*

Nota particolare è che il bit posto a uno vi rimane sino a che non viene letto con una PEEK, che in questo caso legge e cancella, cosicché, dopo

la prima collisione e sino all'azzeramento, qualsiasi interrogazione rischia di essere fuorviante; per alleviare il peso del continuo richiamo di una routine completa di analisi delle collisioni è consigliabile, dopo aver conservato in una variabile il contenuto del registro, passare ai particolari solo se necessario (# 30 o #31 <>0), come mostrato nel listato 3 che propone due esempi da utilizzare come subroutine.

Multicolor

Il C 64 rende possibile la programmazione di sprite a 4 colori; cioè, oltre al colore di fondo ed al colore assegnato allo sprite, è possibile utilizzare altri due colori (multicolor # 1 e # 2).

Questa valorizzazione cromatica sacrifica però la risoluzione, in quanto, per codificare 4 diversi colori, sono impiegati due dei bit della matrice binaria anziché uno (colore/non colore); le combinazioni possibili saranno quindi:

00 - colore di fondo (registro # 33)
01 - multicolor # 1 (registro # 37)
10 - colore animazione (registro # 39 - # 46)
11 - multicolor # 2 (registro # 38).

I 24 punti che compongono le sezioni orizzontali dello sprite saranno forzati a diventare 12 coppie, che potranno assumere uno dei colori prescelti.

Per attivare il modo Multicolor bisognerà "accendere" il bit corrispondente allo sprite, con le tecniche ormai note, all'interno del registro # 28. La selezione dei colori si effettua ponendo nei registri interessati (# 33 / # 37 / # 38 / # 39 - # 46) il valore ricavato dalla tabella 2.

Per costruire sprite multicolori con il programma Sprite drawer, proposto nella prima parte, basterà aggiungere alla riga 30 le istruzioni: POKEV + 28,4: POKEV + 37, (colore 1): POKEV + 38, (colore 2).

```
  MODIFICA
 INPUT DA TASTIERA

1 PRINT"[<1CLR>][<3CRSR D>] PREPARATI AD
  INSERIRE I DATI          [<1CRSR D>] E
PREMI UN TASTO QUALSIASI"
2 GETA$:IFA$=""THEN2
3 N=0:FORI=0TO63:PRINTI"-";:INPUTA(I):NE
XT
4 PRINT"[<1CLR>][<2CRSR D>]":FORX=0TO7:N
=N+1:PRINTN"DATA";
5 FORY=0TO7:A=A(X*8+Y):A$=MID$(STR$(A),2
)+",":PRINTA$;:NEXTY:PRINT"[<1CRSR L>] "
:NEXTX
6 PRINT"9":PRINT"RUN"
```

Listato 2. *La versione con input da tastiera dello Sprite loader.*

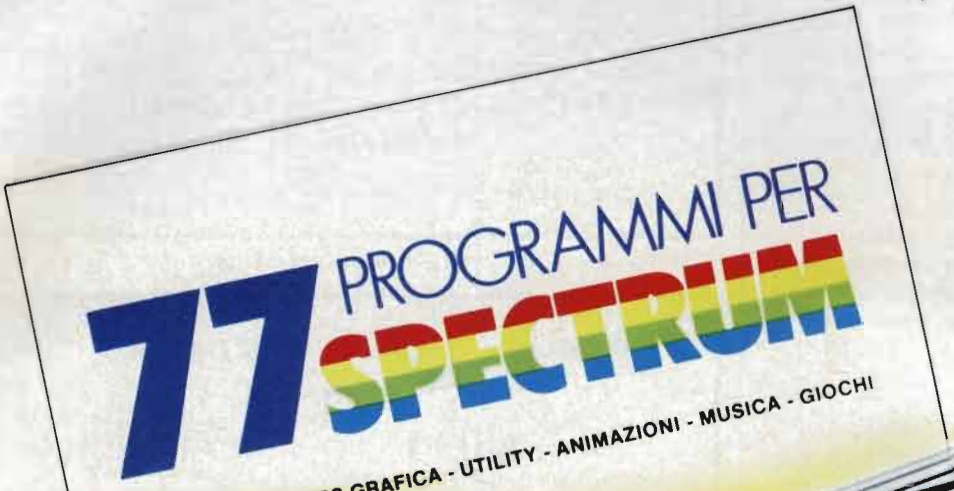
```
**  COLLISIONI  ***

ANALISI  TOTALE

1000 C0=PEEK(V+30):IFC0=0THENRETURN
1010 FORI=0TO7
1020 IF(C0AND2↑I)=2↑ITHENR(I)=1
1025 NEXT
1030 IFR(0)=1THEN *AAAAAA*
1040 IFR(1)=1THEN *BBBBBB*
1050 IFR(2)=1THEN *CCCCCC*
1060 ......................
1100 RETURN


ANALISI  PARZIALE

1000 C0=PEEK(V+30):IFC0=0THENRETURN
1010 IF(C0ANDX)=XTHEN *AAAAA*
1020 IF(C0ANDY)=YTHEN *BBBBB*
1030 IF(C0ANDZ)=ZTHEN *CCCCC*
1100 RETURN
```

Listato 3. *Due esempi di subroutine per verificare le collisioni.*

Conclusione

In questo articolo ho proposto strumenti ed informazioni per creare una base alla programmazione grafica del C 64. Mancano, per completare il quadro, una analisi dettagliata dei caratteri programmabili, del modo Bit map e dello scrolling rallentato. Altro "grande assente" è il linguaggio macchina, che è invece indispensabile per l'applicazione di quanto esposto ai programmi di movimento che coinvolgono l'alta risoluzione con più sprite e spesso con effetti sonori. Mi riprometto di realizzare qualche applicazione concreta, registrando la cronaca ragionata del suo sviluppo, dall'*idea* alla "traduzione" in linguaggio macchina per un ulteriore approfondimento di questo tema. ■

OLTRE L'ORIZZONTE CON LO SPECTRUM



Per ordinare questo libro utilizzate la cedola riportata nella pagina seguente

di Gaetano Marano

GRUPPO EDITORIALE JACKSON

77 PROGRAMMI PER SPECTRUM
150 Pagine, 30 illustrazioni a colori
Cod. 555 A
L. 16000

GRUPPO EDITORIALE JACKSON

E PER LO ZX81...

66 PROGRAMMI PER ZX81 E ZX80 CON NUOVA ROM + HARDWARE

144 Pagine
Cod. **520 D**
L. **12000**

PERSONAL SOFTWARE

Spacman per ZX Spectrum

Un simpatico videogame per tutti gli utenti Sinclair

di *Ivano Parbuono*

Il programma descritto qui di seguito è un classico dei videogiochi. Si tratta di un labirinto con il famoso Spacman che deve mangiare il più possibile, cercando di sfuggire ai tre mostri disseminati nel labirinto stesso. Il movimento non è molto veloce a causa della programmazione in BASIC. Ogni simbolo grafico corrisponde ad un punteggio diverso, non esiste limitazione di tempo per poter mangiare tutto, ma ciò non toglie che sia difficile poter arrivare ai 3715 punti che occorrono per poter cambiare schermo vincendo la partita e sfuggendo ai mostri. Il programma è stato predisposto anche per poter essere giocato con il joystick Kempston.

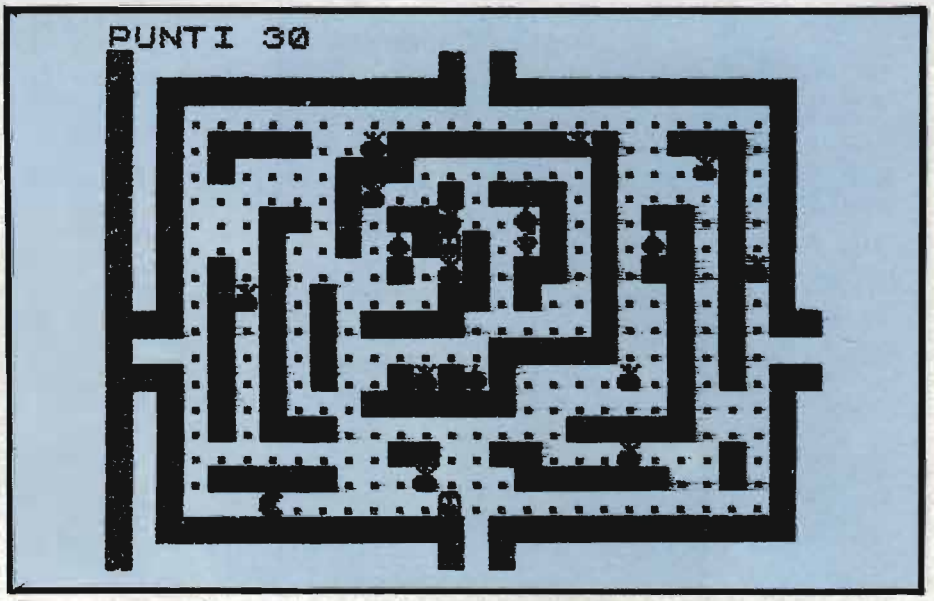


Figura 1. Rappresentazione grafica del gioco sullo schermo.

Analisi del programma

Le linee che vanno dalla N. 7 alla N. 8 si occupano della presentazione del gioco con una piccola pausa. Il gioco inizia con la linea 9 che chiama la subroutine alla linea 1500 per la creazione dei caratteri grafici e memorizza i vari dati con una serie di LET. La linea 20 contiene i comandi per la rappresentazione grafica del labirinto che proseguono fino alla linea 115. La linea 210 controlla continuamente il punteggio e se questo risulta superiore a 3715 rimanda alla linea 3000 dove viene comunicata la vincita della partita.

Le linee che vanno dalla 220 alla 245 fanno si che una volta inserito il joystick il movimento di Spacman risponde a quest'ultimo, mentre quelle che vanno dalla 250 alla 265 permettono il movimento per mezzo dei tasti 5 6 7 8, si noti che Spacman può uscire e rientrare nel labirinto attraverso le porte disposte ai quattro lati. Il nostro Spacman ha una bocca per mangiare e, una volta orientata nella direzione di marcia, le linee 270 - 295 fanno si che prosegua in avanti da solo continuando a mangiare tutto ciò che trova. I vari simboli rappresentati nel labirinto per mezzo delle linee che vanno dalla 350 alla 370 e che una volta inghiottiti dal nostro Spacman a secondo del simbolo, incrementano di 10-50-100 o addirittura 500 il punteggio stampandolo nella parte alta del video, ma attenzione, anzi molta attenzione, Spacman non è solo nel labirinto ci sono anche tre mostri che vogliono prenderlo e sono rappresentati per mezzo delle linee che vanno dalla 375 alla 620 muovendosi in funzione RND, se uno di questi mostri mangia Spacman il gioco termina ed appare la scritta "Hai perso". ■

Linea	Carattere grafico
2000	D
2010	A
2020	B
2030	C
2040	E
2050	R
2060	K
2070	Q
2080	S

Figura 2. *Lista dei simboli grafici.*

Listato 1. *Il programma Spacman.*

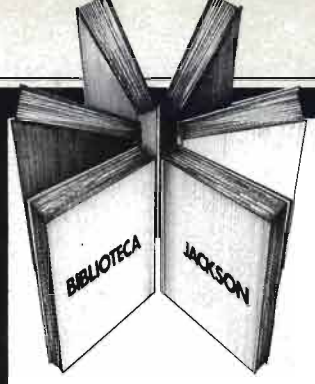
```
   7 FLASH 0: INK 2: PRINT AT 2,
5;"                ": INK 1
: PRINT AT 3,7;"  S P A C M A N
  ": INK 3: PRINT "
                ": PRINT AT 18,0;" SI
 PUO'  GIOCARE  ANCHE  CON
                  JOYSTICK    KEMPSTON": INK 3: P
RINT "
                "
    8 INK 1: PRINT AT 8,3;"
                ": INK 2: PRI
NT AT 9,3;" IDEATO E REALIZZATO
```

Seguito listato Spacman.

```
DR                  PARRUONO IVANO
                 Via A. di CAMBIO 4
                      BY    VERONA ".
INK 1: PRINT "
":": PAUSE 150
   9 GO SUB 1500: LET a=0: LET b
=7: LET c=5: LET d=18: LET e=4:
LET f=15: LET g=14: LET h=18: LE
T i=3: LET S$="": LET P$="": B
ORDER 3: PAPER 3: INK 2: CLS : P
APER 7
  10 FOR l=1 TO 20 STEP 2: BEEP
.1,l: NEXT l: FOR l=20 TO 1 STEP
 -2: BEEP .1,l: NEXT l
  18 DIM T$(20,27)
  20 LET T$(1)="
                              "
  25 LET T$(2)="
"
  30 LET T$(3)="
"
  35 LET T$(4)="
"
  40 LET T$(5)="
"
  45 LET T$(6)="
"
  50 LET T$(7)="
"
  55 LET T$(8)="
"
  60 LET T$(9)="
"
  65 LET T$(10)="
"
  70 LET T$(11)="
"
  75 LET T$(12)="
"
  80 LET T$(13)="
"
  85 LET T$(14)="
"
  90 LET T$(15)="
"
  95 LET T$(16)="
"
 100 LET T$(17)="
"
 105 LET T$(18)="
"
 110 LET T$(19)="
"
 115 LET T$(20)="
"
 150 PRINT ; FOR l=1 TO 20: INK
3: PRINT "";T$(l): NEXT l
 200 LET T$(h,i)=" "
 210 IF a>=3715 THEN CLS : LET a
=0: BORDER 3: PAPER 3: INK 3: CL
S : PAPER 6: GO TO 3000
 220 IF INKEY$="" THEN GO TO 270
 230 IF IN 31=2 THEN LET P$="";
 235 IF IN 31=4 THEN LET P$="";
 240 IF IN 31=8 THEN LET P$="";
 245 IF IN 31=1 THEN LET P$="";
 250 IF INKEY$="5" THEN LET P$="
";
 255 IF INKEY$="6" THEN LET P$="
";
 260 IF INKEY$="7" THEN LET P$="
";
 265 IF INKEY$="8" THEN LET P$="
";
 270 PRINT AT h,i;" "
 280 IF P$="" AND T$(h-1,i)<>"
" THEN LET h=h-1
 285 IF P$="" AND T$(h,i+1)<>"
" THEN LET i=i+1: IF i>26 THEN L
ET i=2
 290 IF P$="" AND T$(h+1,i)<>"
" THEN LET h=h+1: IF h>19 THEN L
ET h=2
 295 IF P$="" AND T$(h,i-1)<>"
" THEN LET i=i-1: IF i<2 THEN LE
T i=26
 300 IF h<2 THEN LET h=20
 310 PRINT AT H,I; INK 0;s$
 350 IF T$(h,i)="." THEN LET a=a
+10: PRINT AT 0,0;"PUNTI ";a: BE
EP .004,54
 360 IF T$(h,i)="" THEN LET a=a
+50: PRINT AT 0,0;"PUNTI ";a: BE
EP .2,5
 365 IF T$(h,i)="" THEN LET a=a
+100: PRINT AT 0,0;"PUNTI ";a: B
EEP .04,35: BEEP .3,5
 370 IF T$(h,i)="" THEN LET a=a
+500: PRINT AT 0,0;"PUNTI ";a: B
EEP .5,7: BEEP .1,9: BEEP .4,9
 375 PRINT AT b,g;T$(b,g): IF IN
T (RND*2)+(B>H) AND T$(b-1,g)<>"
" THEN LET b=b-1
```

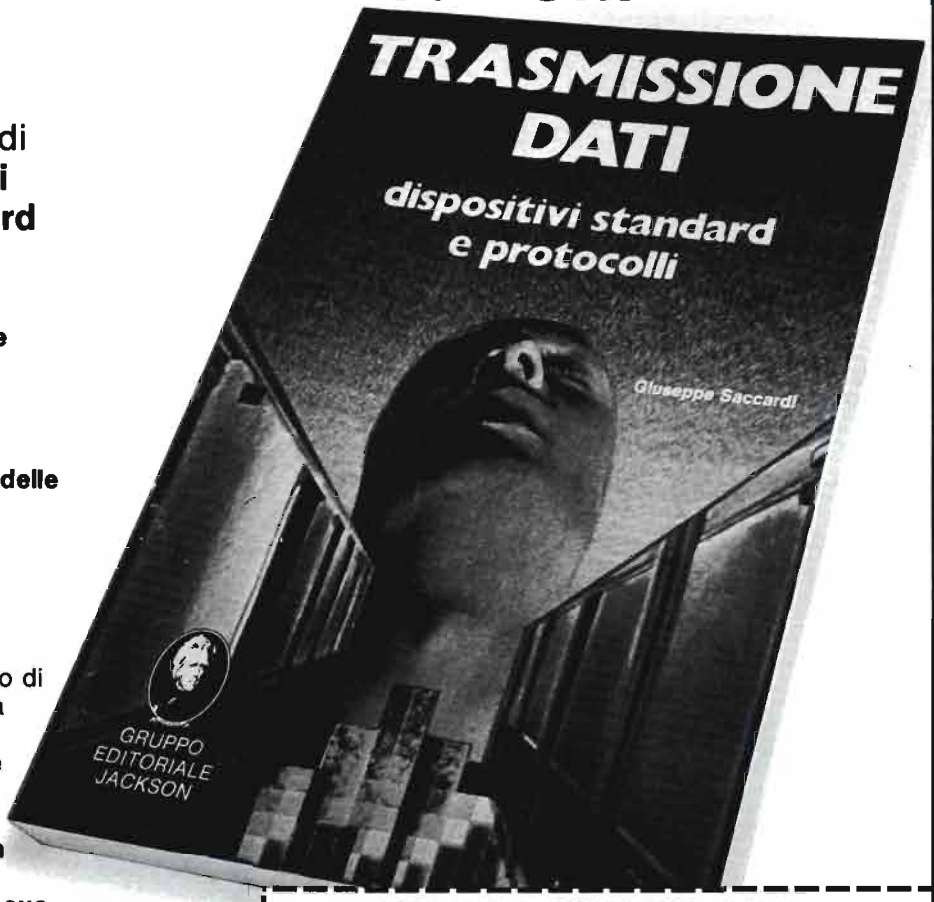
Seguito listato Spacman.

```
 380 IF INT (RND*2)+(B<H) AND T$
(b+1,g)<>"" THEN LET b=b+1
 385 IF INT (RND*2)+(G>I) AND T$
(b,g-1)<>"" THEN LET g=g-1
 390 IF INT (RND*2)+(G<I) AND T$
(b,g+1)<>"" THEN LET g=g+1
 395 PRINT AT b,g; INK 1;""
 600 PRINT AT H,I; INK 0;P$
 510 IF (b=h AND g=i) OR (c=h AN
D e=i) OR (d=h AND f=i) THEN PRI
NT AT h,i: FOR l=-20 TO -40 STEP
 -1,9: BEEP .1,l: NEXT l: FOR a=
1 TO 50: BORDER RND*7: NEXT a: C
LS : FLASH 1: PRINT AT 10,6;"H A
 I  P E R S O !!": PAUSE 0: FLAS
H 0: IF INKEY$<>" " THEN RUN
 520 PRINT AT c,e;T$(c,e): IF IN
T (RND*2)+(C>H) AND T$(c-1,e)<>"
" THEN LET c=c-1
 530 IF INT (RND*2)+(C<H) AND T$
(c+1,e)<>"" THEN LET c=c+1
 540 IF INT (RND*2)+(E>I) AND T$
(c,e-1)<>"" THEN LET e=e-1
 550 IF INT (RND*2)+(E<I) AND T$
(c,e+1)<>"" THEN LET e=e+1
 560 PRINT AT c,e; INK 1;""
 580 PRINT AT d,f;T$(d,f): IF IN
T (RND*2)+(D>H) AND T$(d-1,f)<>"
" THEN LET d=d-1
 590 IF INT (RND*2)+(D<H) AND T$
(d+1,f)<>"" THEN LET d=d+1
 600 IF INT (RND*2)+(F>I) AND T$
(d,f-1)<>"" THEN LET f=f-1
 610 IF INT (RND*2)+(F<I) AND T$
(d,f+1)<>"" THEN LET f=f+1
 620 PRINT AT d,f; INK 2;""
 650 GO TO 200
1000 GO TO 9
1500 DATA 0,BIN 01000010,BIN 111
00111,BIN 11111111,BIN 11111111,
BIN 11111111,BIN 01111110,BIN 00
111100
1510 DATA BIN 00111100,BIN 01111
110,BIN 11111100,BIN 11111000,BI
N 11111000,BIN 11111100,BIN 0111
1110,BIN 00111100
1520 DATA BIN 00111100,BIN 01111
110,BIN 00111111,BIN 00011111,BI
N 00011111,BIN 00111111,BIN 0111
1110,BIN 00111100
1530 DATA BIN 00111100,BIN 01111
110,BIN 11111111,BIN 11111111,BI
N 11111111,BIN 11100111,BIN 0100
0010,0
1540 DATA BIN 01111100,BIN 11111
110,BIN 10010010,BIN 10010010,BI
N 11111110,BIN 11111110,BIN 1111
1110,BIN 10101010
1550 DATA BIN 01010010,BIN 00110
100,BIN 00011000,BIN 01111100,BI
N 11111110,BIN 11111110,BIN 1111
1110,BIN 01111100
1560 DATA BIN 00010000,BIN 00010
000,BIN 00111000,BIN 01111100,BI
N 11111110,BIN 11111110,BIN 0111
1100,BIN 00111000
1570 DATA BIN 00001100,BIN 00010
000,BIN 01111100,BIN 10110110,BI
N 01111100,BIN 00101000,BIN 0001
1000,BIN 00010000
1580 DATA BIN 00111100,BIN 01111
110,BIN 11111111,BIN 11111111,BI
N 11111111,BIN 11111111,BIN 0111
1110,BIN 00111100
1590 DATA 24,60,126,219,219,255,
219,145
1900 RESTORE 1500
2000 FOR f=0 TO 7: READ a: POKE
USR ""+f,a: NEXT f
2010 FOR f=0 TO 7: READ a: POKE
USR ""+f,a: NEXT f
2020 FOR f=0 TO 7: READ a: POKE
USR ""+f,a: NEXT f
2030 FOR f=0 TO 7: READ a: POKE
USR ""+f,a: NEXT f
2040 FOR f=0 TO 7: READ a: POKE
USR ""+f,a: NEXT f
2050 FOR f=0 TO 7: READ a: POKE
USR ""+f,a: NEXT f
2060 FOR f=0 TO 7: READ a: POKE
USR ""+f,a: NEXT f
2070 FOR f=0 TO 7: READ a: POKE
USR ""+f,a: NEXT f
2080 FOR f=0 TO 7: READ a: POKE
USR ""+f,a: NEXT f
2100 RETURN
3000 BEEP 2,1: BEEP 1,2: BEEP .8
,4: BEEP .6,2: CLS : BORDER 2: P
APER 6: FLASH 1: INK 1: PRINT AT
 10,4;"BRAVISSIMO  SEI SFUGGITO"
: PRINT AT 11,12;"AL MOSTRO": PA
USE 150: BEEP .4,4: BEEP .2,35:
BEEP .2,8: BEEP 2,3: BEEP 1,5: F
LASH 0: GO TO 1
```



Informatica

Il calcolatore e le comunicazioni



Giuseppe Saccardi
Trasmissione Dati dispositivi standard e protocolli

Un volume che, senza presupporre conoscenze specifiche del lettore, lo mette al corrente sull'attualità e sulle prospettive della trasmissione elettronica delle informazioni.

Descrive i principali dispositivi attualmente utilizzati; chiarisce le normative e gli standard internazionali; esamina i protocolli che permettono di trasferire informazioni tra calcolatori e terminali; analizza caratteristiche e funzioni di un moderno PABX.

Un libro che traduce in tecnologia, la profezia letteraria di Orwell e del suo "1984".

300 pagine
Lire 23.000
Codice **528P**

CEDOLA DI COMMISSIONE LIBRARIA

VOGLIATE SPEDIRMI

n° copie	codice	Prezzo unitario	Prezzo totale
	528P	**L. 23.000**	

☐ Pagherò contrassegno al postino il prezzo indicato più **L. 2000 per contributo fisso spese di spedizione.**

Condizioni di pagamento con esenzione del contributo spese di spedizione:

☐ Allego assegno della Banca _____

n° _____

☐ Allego fotocopia del versamento su c/c n. 11666203 a voi intestato

☐ Allego fotocopia di versamento su vaglia postale a voi intestato

Nome
Cognome
Via
Cap Città Prov.
Data Firma

Spazio riservato alle Aziende. Si richiede l'emissione di fattura

Partita I.V.A. |_|_|_|_|_|_|_|_|_|_|_|

GRUPPO EDITORIALE JACKSON

Attenzione compilare per intero la cedola
ritagliare (o fotocopiare) e spedire in busta chiusa a:
GRUPPO EDITORIALE JACKSON
Divisione Libri
Via Rosellini, 12 - 20124 Milano

Macro istruzioni per Apple II

Personalizzate e potenziate il vostro BASIC

di *Roberto Brunialti*

Che cosa è una macro

La macro rappresenta nell'ambito della programmazione ciò che gli elementi prefabbricati sono nel campo della edilizia.
Una macro è, infatti, una istruzione simbolica con un nome ed eventuali argomenti, che viene rimpiazzata da una serie di istruzioni di più basso valore simbolico. A ben vedere, una qualsiasi istruzione BASIC è in realtà una macro che viene sostituita, in maniera non permanente, con le corrispondenti istruzioni in linguaggio macchina.

Operando in BASIC può essere di grande utilità disporre di un programma che permetta una facile definizione di una serie di macroistruzioni che vengano poi tradotte in linee BASIC correntemente eseguibili; si tratta in pratica di un preinterprete.
Ma vediamo che cosa significa tutto ciò che abbiamo detto, con un esempio abbastanza significativo.
Nell'uso del calcolatore per applicazioni matematiche, capita frequentemente di dover leggere da dischetto matrici numeriche. Ammettendo che la matrice sia memorizzata in un file sequenziale, con i primi due record che definiscono, rispettivamente, il numero di righe e di colonne e con i dati che seguono ordinati per colonne, un programma di lettura sarà del tipo:

```
10  D$ = CHR$ (4)
20  N$ = "nome file"
30  PRINT D$; "OPEN"; N$
40  PRINT D$; "READ"; N$
50  INPUT NR:INPUT NC
60  DIM A(NR,NC)
70  FOR I = 1 TO NC
80  FOR J = 1 TO NR
90  INPUT A(J,I)
100 NEXT J,I
110 PRINT D$; "CLOSE"; N$
```

Come si vede ben undici righe, irte di virgole e virgolette, con errori di battitura sempre in agguato. Che dire poi se le matrici da leggere sono numerose?
Il nostro programma permette, per l'appunto, di definire una macro di nome LEGGI con i seguenti parametri:
1 — nome della matrice in cui inserire i dati letti;
2 — nome variabile del numero delle righe;
3 — nome variabile del numero delle colonne;
4 — nome del file dati.
Essendo il precedente un esempio, niente vieta di definire diversamente e con altro nome una macro che svolga le stesse funzioni.
Nel nostro caso le nove righe dalla 30 alla 110 possono essere sostituite dalla comoda macro:

```
30  LEGGI A,NR,NC,N$
```

Bello, non è vero?

Memorizzazione dei programmi in un text-file

Nell'Apple II non è possibile memorizzare direttamente un programma sotto forma di testo, successivamente manipolabile. Poiché il nostro programma MACRO, come forse si è già intuito, è essenzialmente un abile manipolatore di stringhe, il primo problema è quello di scrivere su disco e poi rileggere correttamente il programma contenente delle macro, sotto forma di testo.
A pagina 76 del manuale DOS 3.3 è indicato un metodo di scrittura su disco. Ricordo che si tratta di aprire un file sequenziale e scrivere tutto il programma con un comando LIST.

Il problema da risolvere, adottando questo metodo, è quello dell'inserzione di un CR ogni trentatre caratteri.
Poiché CR è anche il carattere di fine record, viene resa indistinguibile la linea completa dai frammenti da questa derivati.
Quando si tenta di leggere una linea di un programma così archiviato, spesso risulta spezzata in più record, a seconda della sua lunghezza.

Ma non è tutto. Se nella linea di programma sono inseriti caratteri separatori (virgole, punti e virgola, due punti), in fase di lettura verrà segnalato un EXTRA IGNORED con troncamento di tutto quanto segue.

Per risolvere questi due problemi per la lettura delle linee è stata usata la istruzione GET che accetta tutti i caratteri ASCII separatori compresi, mentre per la ricostruzione delle linee originali del programma, viene effettuato un controllo incrociato. Infatti se un record non inizia con un numero, sicuramente appartiene (essendone un frammento) alla linea di programma del record precedente. Poiché può accadere che casualmente venga troncata una linea nel mezzo di una equazione numerica, per evitare che il frammento venga considerato come una nuova linea (iniziando con un numero) si verifica che il numero di testa sia superiore a quello dell'ultima linea trattata e inferiore a quello della successiva.

Macro istruzioni per Apple II

70-260	Menu principale e selezione opzioni.
270-350	Opzione 3: master program con chiamata subroutine relative.
360-560	Opzione 3: input del nome programma e vocabolario, lettura programma dal disco.
570-720	Opzione 3: controllo incrociato (vedi testo) per la ricostruzione delle linee del programma.
730-1010	Opzione 3: lettura del file <voc>. CTRL associato ad ogni vocabolario del numero macro. Usa la subroutine 2690 per la lettura dei parametri e delle definizioni delle macro. Scandaglia tutte le righe in cerca delle macro. Se ne trova chiama la sub 1210.
1020-1200	Opzione 3: scrive sul file MACRO.COMODO il programma elaborato, più alcuni comandi per il caricamento e il list automatico, eseguiti mediante un EXEC.
1210-1440	Opzione 3: isola i parametri della macro e ricompone la linea con il testo di definizione.
1450-1980	Opzione 1: definizione delle macro ed aggiornamento del file <voc>.CTRL e del file <voc>.
1990-2190	Opzione 2: trasforma un file programma in file text per permettere alla opzione 3 di leggerlo. Riaggancia, alla fine, il programma MACRO.
2200-2680	Opzione 4: legge il vocabolario mediante la sub 2690 e se incontra la macro che si vuole cancellare non la ricopia sul file MACRO.COMODO. Alla fine aggiorna <voc>.CTRL e, dopo aver cancellato <voc>, mediante un RENAME lo ricrea dal file MACRO.COMODO, aggiornato.
2690-2850	Routine utilizzata per leggere le singole macro dal file <voc>.
2680-2970	Routine di input linea di definizione macro. chiamata dalla opzione 1.
2980-3060	Routine di errore per tentativo lettura file <voc> che risulti vuoto.

Figura 1. *REMarks del listato Macro Program.*

Questo permette di ridurre quasi a zero la probabilità che un frammento venga considerato come una linea di programma BASIC e nell'uso pratico non si sono mai verificati errori.

Il programma

Il programma che presento è di uso assai semplice e piuttosto potente. Si possono definire più macro in una linea di programma e si possono inserire macro anche all'interno di altre.
Unica limitazione: non si deve superare la lunghezza di 251 caratteri per linea di programma dopo la "traduzione".
Questa è una limitazione instrinseca al BASIC dell'Apple le cui linee non possono superare i 255 caratteri.

Il programma si compone di un menu principale che rimanda a quattro opzioni, con compiti distinti che adesso esaminiamo in dettaglio mediante l'esempio riportato all'inizio.

Opzione 1
Definizione nuove macro

È la parte basilare del programma. In questa sede si creano nuove macro mediante la definizione dei loro parametri e la scrittura del testo. Nel nostro caso vogliamo creare la macro LEGGI.
La prima domanda del programma è NOME DEL VOCABOLARIO. Poiché possiamo definire macro indirizzate a problemi specifici, conviene raggrupparle in vocabolari. Questi sono delle raccolte di macro con nome collettivo. Nel nostro caso creiamo un vocabolario di macro 'matematiche' e quindi con poca fantasia risponderemo MAT!

Dobbiamo ora inserire, su richiesta del programma, i quattro parametri:
MACRO-NAME: nome della macro con cui verrà chiamata da programma. Nel nostro caso rispondiamo con LEGGI.
MACRO-ARG: numero degli argomenti della macro, che verranno specificati nella chiamata. Nell'esempio sono i quattro specificati all'inizio.
ARG-SEP: è il segno di separazione fra gli argomenti. Si possono scegliere uno o più caratteri. Normalmente (come nel nostro caso) conviene scegliere la virgola o punto e virgola. Occorre assicurarsi che internamente agli argomenti non compaia il separatore scelto.
ARG-PRE: prefisso con il quale, subito dopo, verranno indicati gli argomenti della definizione vera e propria. Viene richiesto (anche se si potrebbe definire una volta per tutte) per rendere più flessibile il programma. Scriviamo ARG.
A questo punto ci viene chiesto di scrivere il testo della macro che, per quanto sopra detto, non può superare i 251 caratteri (4 caratteri servono per l'inserimento di un 'tappo' dato dalla stringa :REM).
Il testo sarà:

```
PRINT D$; "OPEN"; ARG4:
PRINT D$; "READ"; ARG4:
INPUT ARG2: INPUT ARG3:
DIM ARG1 (ARG2,ARG3): FOR I
= 1 TO ARG3: FOR J = 1 TO
ARG2: INPUT ARG1 (J,I): NEXT:
NEXT: PRINT D$; "CLOSE";
ARG4
```

Bisogna fare molta attenzione alla scrittura del testo. Questo infatti viene accettato mediante la funzione GET perché questa non rifiuta i caratteri separatori che porterebbero a

un troncamento nel caso della funzione INPUT. Dovendo simulare una istruzione che accettasse tutta una linea (comprese le virgole ed altri segni), lasciando le funzioni del cursore, sono dovuto arrivare al compromesso di utilizzare il cursore solo sulla stessa linea, non potendolo utilizzare per 'risalire' linee. Scritto correttamente il testo (rivederlo più volte) si prema due volte il tasto di Return.

Ora su dischetto è presente un vocabolario di nome MAT contenente la sola macro LEGGI.
Dopo aver scritto un programma di prova del tipo:

```
.....
100   INPUT "NOME FILE:"; N$
.....
1000  LEGGI A,R,C,N$
.....
```

e averlo salvato su dischetto sotto il nome di PROVA, lo si trasforma in un file di testo mediante la seconda opzione.

Opzione 2
Scrivi testo programma

È di uso semplicissimo. Basta rispondere alla domanda NOME PROGRAMMA: con PROVA, aspettare qualche secondo e passare alla ultima opzione.

Opzione 3
Traduci programmi macro

Viene richiesto il nome del programma da tradurre (scriviamo PROVA) e il nome del vocabolario (MAT). Dopo aver premuto Return, se non sono stati fatti errori nella scrittura del programma, avremo, dopo breve attesa che ovviamente dipende dalla lunghezza del programma, in memoria centrale il programma pre-interpretato, pronto per l'uso.

Nome	PRO
N. argomenti	6
Sintassi	PRO a1, a2, a3, a4, a5, a6 a1 = 1ª matrice a2 = 2ª matrice a3 = matrice risultato a4 = numero righe a1 a5 = numero colonne a1 a6 = numero colonne a2
Definizione	FOR I = 1 TO a4:FOR J = 1 TO a6:FOR K = 1 TO a5: a3(I,J) = a3(I,J) + a1(I,K)★a2(K,J):NEXT: NEXT: NEXT
Esempio	PRO A, B, C, R1, C1, C2
Nota	Questa macro calcola la matrice prodotto nel modo consueto C = A ★ B
Nome	DET
N. argomenti	3
Sintassi	DET a1, a2, a3, a1 = nome della matrice a2 = numero delle righe o delle colonne a3 = variabile in cui si troverà il risultato
Definizione	a3 = 0:FOR I = 1 TO a2:IF a1(I,I) < >0 THEN NEXT: a3 = 1: FOR I = 1 a2: FOR J = I + 1 TO a2: FOR K = I + 1 TO a2: a1 (J,K) = a1 (J,K) − a1 (I,K) ★ a1 (J,I) /a1 (I,I): NEXT: NEXT: a3 = a3 ★ a1 (I,I): NEXT
Esempio	DET A,3,DT
Nota	La matrice a1 viene alterata nel calcolo. Il determinante risulta zero se nella diagonale è presente uno zero. Per esigenze dell'algoritmo le dimensioni di a1 devono essere aumentate di una unità.
Nome	AREA
N. argomenti	5
Sintassi	AREA a1, a2, a3, a4, a5 a1 = funzione di cui si vuole l'integrale a2 = estremo inferiore di integrazione a3 = estremo superiore di integrazione a4 = numero di iterazioni a5 = variabile in cui si troverà il risultato
Definizione	FOR I = ((a3 − a2)/a4)/2 TO a3 STEP (a3 − a2)/a4: a5 = a5 + a1:NEXT: a5 = a5 ★ (a3 − a2)/a4
Esempio	AREA (3.5 ★ EXP(I)), 2, 5, 100, AC
Nota	Questa macro calcola l'integrale della espressione in a1. L'espressione deve essere funzione della variabile I.

Tavola 1. *Ecco tre macro già definite e pronte da utilizzare.*

Opzione 4
Cancella macro

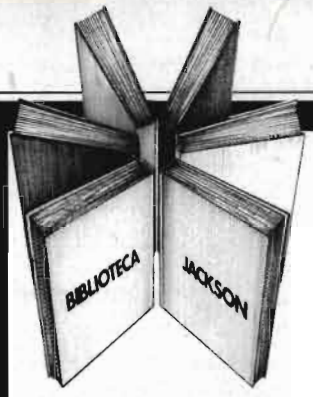
È una opzione da usare con parsimonia, perché tende a manomettere il vocabolario oggetto della cancellazione. Anche questa volta l'uso è semplicissimo. Basta rispondere alle domande poste con il nome della macro e con quello del vocabolario in cui essa risiede.

Ulteriori esempi

Oltre all'esempio sino ad ora considerato (LEGGI), sono presentate nella tavola 1 altre 3 utili macro, pronte per l'uso (ovviamente dopo averle create in un apposito vocabolario).
Lo specchietto riportato è sufficientemente chiaro da non necessitare ulteriori commenti.

Ultime considerazioni

Il programma MACRO genera, oltre ai vocabolari definiti dall'utente, anche un file <voc>. CTRL associato a ciascuno di essi, che ripor-

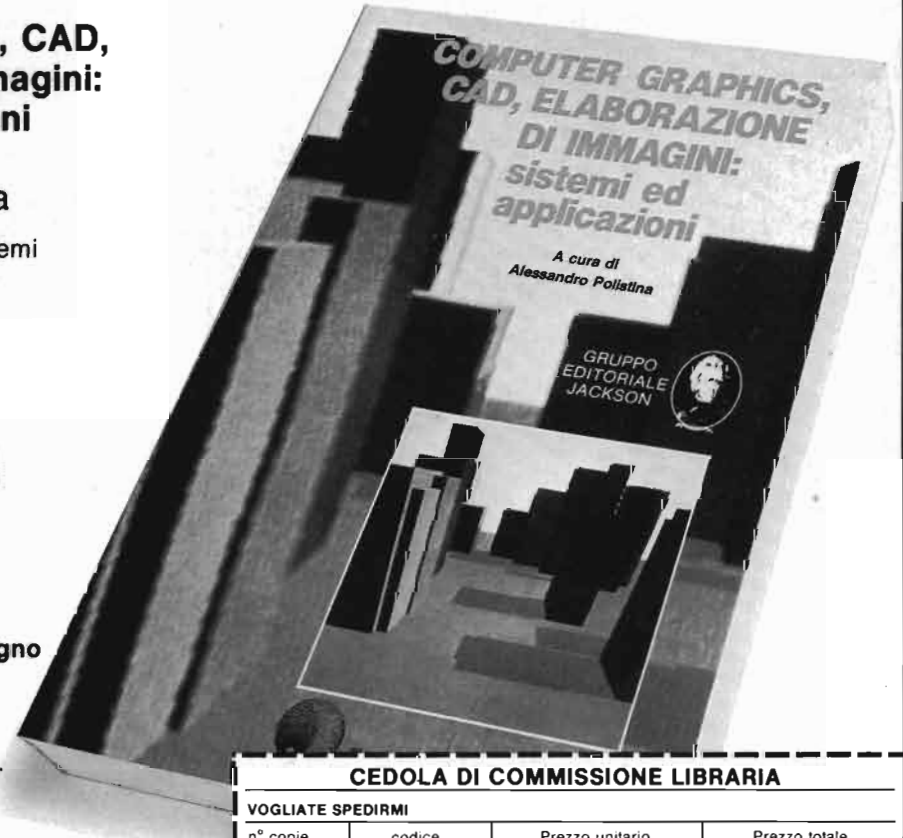


Informatica

Esperti a confronto su attualità e prospettive della Computer Grafica

Computer Graphics, CAD, elaborazione di immagini: sistemi e applicazioni

A cura di
Alessandro Polistina



Linguaggi e algoritmi, sistemi grafici, CAD/CAM, didattica e formazione professionale, Computer Graphics e Editoria, modellazione di solidi, CAD in architettura, CAD meccanico, acquisizione e elaborazione di immagini, elaborazione di immagini e scienze biomediche, cartografia e pianificazione editoriale, immagini sintetiche per la televisione....

Tutti gli Atti del 3° Convegno Nazionale AICOGRAPHICS riuniti in un solo volume a disposizione di operatori, sperimentatori, appassionati. 512 pagine, numerosissimi schemi, un'Appendice con 33 illustrazioni a colori..

Lire 45.000
Codice **529C**

CEDOLA DI COMMISSIONE LIBRARIA

VOGLIATE SPEDIRMI

n° copie	codice	Prezzo unitario	Prezzo totale
	529C	L. 45.000	

☐ Pagherò contrassegno al postino il prezzo indicato più **L. 2000 per contributo fisso spese di spedizione.**

Condizioni di pagamento con esenzione del contributo spese di spedizione:

☐ Allego assegno della Banca _____

n° _____

☐ Allego fotocopia del versamento su c/c n. 11666203 a voi intestato

☐ Allego fotocopia di versamento su vaglia postale a voi intestato

Nome

Cognome

Via

Cap Città Prov.

Data Firma

Spazio riservato alle Aziende. Si richiede l'emissione di fattura

Partita I.V.A. |_|_|_|_|_|_|_|_|_|_|_|

GRUPPO EDITORIALE JACKSON

Attenzione compilare per intero la cedola
ritagliare (o fotocopiare) e spedire in busta chiusa a:
GRUPPO EDITORIALE JACKSON
Divisione Libri
Via Rosellini, 12 - 20124 Milano

Macro istruzioni per Apple II

ta il numero di macro presenti. Esso viene automaticamente aggiornato dopo ogni attività di definizione o di cancellazione.
Un altro file di sistema è MACRO.COMODO che, come indica il nome, è di uso generico.
Non interferire mai con questi files, se non si vuole incorrere in pericolosi crash del sistema.

Il programma è un pochino complesso, evito perciò una descrizione dettagliata.
Per i più curiosi dovrebbero bastare la fig. 1 e il listato del programma.
Nei listati 2 e 3 viene presentato un programma per dimostrare praticamente il modo di utilizzo della macro.
Altra cosa utile, e che suggerisco caldamente, è quella di appuntarsi la definizione e la sintassi di tutte le macro, risultando tale operazione praticamente indispensabile in caso di molte macro in più vocabolari.

A voi l'uso, semplice nonostante la prima impressione, e la soddisfazione di avere un BASIC personalizzato. E scusate se è poco.

Listato 1. *Il programma per la creazione delle macro.*

```
1  REM **********************
2  REM *    MACRO PROGRAM     *
3  REM *      REL   1.2       *
4  REM *  ----------------    *
5  REM *    BY R.BRUNIALTI    *
6  REM *      ON 12/04/83     *
7  REM **********************
8  REM
9  REM
10 MD = 40:HL = .05:UN = 1
20  DEF  FN M(A) =  INT ((A / MD -
      INT (A / MD)) * MD + HL) *
    UN
30 L$ = "-----------------------------
    ----------------"
40 D$ =  CHR$ (4)
50  DIM PA$(30)
60  DIM NA(30)
70  REM ********************
80  REM   MAIN MENU'
90  REM ********************
100  REM
110  HOME : VTAB 2
120  PRINT L$: PRINT "*** MACRO P
    ROGRAM *** MACRO PROGRAM ***
    ": PRINT L$
130  VTAB 8
140  PRINT "           MENU PRINCI
    PALE"
150  VTAB 10
160  PRINT "       1- DEFINIZIONE
    NUOVE MACRO"
170  PRINT "       2- SCRIVI TESTO
     PROGRAMMA"
180  PRINT "       3- TRADUCI PROG
    RAMMAMACRO"
190  PRINT "       4- CANCELLA MAC
    RO"
200  PRINT "       5- EXIT"
210  PRINT : PRINT : INPUT "SELEZ
    IONA IL NUMERO : ";NN$
220 NN =  VAL (NN$)
230  IF NN < 1 OR NN > 5 THEN  GOTO
    70
240  IF NN = 5 THEN  END
250  ON NN GOSUB 1450,1990,270,22
    00
260  RUN
270  REM ********************
280  REM  OPTION3 MASTER
290  REM ********************
300  REM
310  GOSUB 360
320  GOSUB 570
330  GOSUB 730
340  GOSUB 1020
350  END
360  REM ********************
370  REM     INQUIERY
380  REM ********************
390  REM
400  HOME
410  DIM LN$(600)
420 D$ =  CHR$ (4)
430  PRINT L$: PRINT "OPZIONE 3 *
    ****************************
    ": PRINT L$
440  VTAB 10
450  INPUT "NOME PROGRAMMA MACRO
    :";N$
460  INPUT "NOME DEL VOCABOLARIO
    :";VO$
470 N1$ = N$ + ".COMODO"
480  PRINT D$;"OPEN ";N$
490  PRINT D$;"READ ";N$
500  ONERR  GOTO 540
510 PT = PT + 1
520  GET A$: IF A$ <  > CHR$ (13
    ) THEN LN$(PT) = LN$(PT) + A
    $: GOTO 520
530  GOTO 510
540  PRINT D$;"CLOSE"
550  POKE 216,0
560  GOTO 320
570  REM ********************
580  REM  LINE CONTROL
590  REM ********************
600  REM
610 X =  FRE (0)
620 LN$(PT + 1) = "99999"
630 ANTE =  VAL (LN$(1))
640  FOR I = 1 TO PT
650 CURR =  VAL (LN$(I))
660 DOPO =  VAL (LN$(I + 1))
670  IF CURR = 0 AND (CURR > DOPO
     OR CURR < ANTE) THEN LN$(I)
     = LN$(I - 1) + LN$(I):LN$(I
     - 1) = "":CURR = ANTE
680 ANTE = CURR
690  NEXT
700 NL = PT:PT = 0:FL = 0
710 FL = 0
```

Seguito programma creazione macro.

```
720  RETURN
730  REM ********************
740  REM      SEARCH
750  REM ********************
760  REM
770  PRINT D$;"OPEN";VO$;".CTRL"
780  PRINT D$;"READ";VO$;".CTRL"
790  ONERR  GOTO 2980
800  INPUT NV
810  IF NV = 0 THEN  GOTO 2980
820  PRINT D$;"CLOSE"
830  PRINT D$;"OPEN ";VO$
840  PRINT D$;"READ ";VO$
850  FOR I = 1 TO NV
860  GOSUB 2690
870  FOR K = 1 TO NL
880  IF LN$(K) = "" THEN 980
890 L2 =  LEN (LN$(K))
900 L1 =  LEN (N$)
910 L$ = LN$(K) + "  "
920 TP$ = ""
930  FOR L = 1 TO L2 - L1
940 MI$ =  MID$ (L$,L,L1)
950  IF MI$ = N$ THEN FL = 1: GOSUB
     1240
960  NEXT L
970  IF FL THEN LN$(K) = TP$ +  MID$
     (L$,P1):FL = 0
980  NEXT K
990  NEXT I
1000  PRINT : PRINT D$;"CLOSE"
1010  RETURN
1020  REM ********************
1030  REM  STORE PROGRAM
1040  REM ********************
1050  REM
1060 N1$ = "MACRO.COMODO"
1070  PRINT D$;"OPEN ";N1$
1080  PRINT D$;"DELETE";N1$
1090  PRINT D$;"OPEN ";N1$
1100  PRINT D$;"WRITE";N1$
1110  PRINT "NEW"
1120  FOR I = 1 TO NL
1130  IF LN$(I) <  > "" THEN  PRINT
     LN$(I)
1140  NEXT
1150  PRINT "HOME"
1160  PRINT "LIST"
1170  PRINT "DELETE ";N1$
1180  PRINT "SAVE ";N1$
1190  PRINT D$;"EXEC ";N1$
1200  END
1210  REM ********************
1220  REM      INSERT
1230  REM ********************
1240  REM
1250 LC =  VAL (LN$(K))
1260 LP =  VAL (LN$(K - 1))
1270 LS =  VAL (LN$(K + 1))
1280 TP$ =  LEFT$ (L$,L - 1)
1290 P1 = L + L1
1300  FOR M = 1 TO NA
1310 AR$(M) = ""
1320  FOR N = P1 TO L2
1330 MI$ =  MID$ (L$,N,1)
```

Seguito programma creazione macro.

```
1340  IF MI$ = SEP$ OR MI$ = ":" THEN
     AR$(M) =  MID$ (L$,P1,N - P1
     ):P1 = N + 1: GOTO 1360
1350  NEXT N
1360  IF MI$ = ":" THEN P1 = P1 -
     1
1370  NEXT M
1380  IF AR$(NA) = "" THEN AR$(NA
     ) =  MID$ (L$,P1):P1 = L2 +
     1
1390  FOR M = 1 TO NN
1400 TP$ = TP$ + PAR$(M) + AR$(NA
     (M))
1410  NEXT M
1420 TP$ = TP$ + PAR$(NN + 1)
1430 X =  FRE (0)
1440  RETURN
1450  REM ********************
1460  REM OPTION1
1470  REM ********************
1480  REM
1490  HOME
1500  PRINT L$: PRINT "OPZIONE 1
     ****************************
     *": PRINT L$
1510  PRINT : PRINT L$
1520  INPUT "NOME VOCABOLARIO :";
     VO$
1530  PRINT L$
1540  PRINT "MACRO-NAME        : "
     ;: INPUT "";N$
1550  PRINT "MACRO-ARG.         : "
     ;: INPUT "";NA
1560 SP$ = ""
1570  PRINT "       ARG-SEPAR. : "
     ;
1580  GET A$: PRINT A$;: IF A$ <
      > CHR$ (13) THEN SP$ = SP$
      + A$: GOTO 1580
1590  PRINT "       ARG-PREFIX : "
     ;: INPUT "";PRE$
1600  PRINT L$
1610  PRINT " ENTER MACRO TEXT. N
     OT LONGER THAN 255"
1620  PRINT L$
1630  VTAB 15: HTAB 1
1640 TX$ = ""
1650  GOSUB 2860
1660  VTAB 23: HTAB 1: INPUT "OK?
     ";A$: IF A$ = "N" THEN TX$ =
     "": GOTO 1630
1670 TX$ = TX$ + ":REM"
1680 L1 =  LEN (TX$)
1690 L2 =  LEN (PRE$)
1700 P1 = 1
1710  FOR I = 1 TO L1 - (L2 + 1)
1720 MI$ =  MID$ (TX$,I,L2)
1730  IF MI$ = PRE$ THEN PT = PT +
     1:PART$(PT) =  MID$ (TX$,P1,
     I - P1):P1 = I + L2 + 1:NN =
      VAL ( MID$ (TX$,I + L2,1)):
     PP = PP + 1:NA(PP) = NN
1740  NEXT I
1750 PART$(PT + 1) =  MID$ (TX$,P
     1)
```

Macro istruzioni per Apple II

Seguito programma creazione macro.

```
1760  PRINT D$;"OPEN ";VO$
1770  PRINT D$;"APPEND ";VO$
1780  PRINT D$;"WRITE ";VO$
1790  PRINT N$: PRINT PT: PRINT N
     A: PRINT SP$
1800  FOR I = 1 TO PT
1810  PRINT PA$(I)
1820  PRINT NA(I)
1830  NEXT I
1840  PRINT PART$(PT + 1)
1850  PRINT D$;"OPEN ";VO$;".CTRL
     "
1860  PRINT D$;"READ ";VO$;".CTRL
     "
1870  ONERR  GOTO 1950
1880  INPUT CC
1890  PRINT D$;"OPEN  ";VO$;".CTR
     L"
1900  PRINT D$;"WRITE ";VO$;".CTR
     L"
1910  PRINT CC + 1
1920  POKE 216,0
1930  PRINT D$;"CLOSE"
1940  RUN
1950  PRINT D$;"WRITE ";VO$;".CTR
     L"
1960  PRINT 1
1970  POKE 216,0
1980  GOTO 1930
1990  REM ********************
2000  REM OPTION2
2010  REM ********************
2020  HOME
2030  PRINT L$: PRINT "OPZIONE 2
     ****************************
     *": PRINT L$
2040  VTAB 12
2050  PRINT D$;"OPEN MACRO.COMODO
     "
2060  PRINT D$;"DELETE MACRO.COMO
     DO"
2070  INPUT "NOME PROGRAMMA : ";N
     $
2080  PRINT D$;"OPEN MACRO.COMODO
     "
2090  PRINT D$;"WRITE MACRO.COMOD
     O"
2100  READ A$: PRINT A$
2110  READ A$: PRINT A$;N$
2120  READ A$:A$ = A$ + N$ +  CHR$
     (34): PRINT A$
2130  READ A$:A$ = A$ + N$ +  CHR$
     (34): PRINT A$
2140  READ A$: PRINT A$: READ A$:
      PRINT A$: READ A$: PRINT A$
     ;N$
2150  READ A$: PRINT A$: READ A$:
      PRINT A$
2160  PRINT D$;"EXEC MACRO.COMODO
     "
2170  END
2180  DATA          NEW,LOAD ,1 PR
     INT CHR$(4);"OPEN  ,2PRINT C
     HR$(4);"WRITE  ,"3LIST5,6399
     9",4 END,DELETE, RUN, RUN MA
```

Seguito programma creazione macro.

```
     CRO
2190  RETURN
2200  REM ********************
2210  REM OPTION4
2220  REM ********************
2230  REM
2240  HOME
2250 CM$ = "MACRO.COMODO"
2260  PRINT L$: PRINT "OPZIONE 4
     ****************************
     *": PRINT L$
2270  VTAB 10
2280  INPUT "NOME MACRO DA CANCEL
     LARE :";N1$
2290  INPUT "NOME VOCABOLARIO
          :";VO$
2300  PRINT
2310  INPUT "SICURO ?";A$
2320  IF  LEFT$ (A$,1) = "Y" OR  LEFT$
     (A$,1) = "S" THEN  GOTO 2340

2330  RETURN
2340  PRINT D$;"OPEN";VO$;".CTRL"

2350  PRINT D$;"READ";VO$;".CTRL"
2360  ONERR  GOTO 2980
2370  INPUT NV
2380  IF NV = 0 THEN  GOTO 2980
2390  PRINT D$;"CLOSE"
2400  PRINT D$;"OPEN ";VO$
2410  PRINT D$;"READ ";VO$
2420  PRINT D$;"OPEN ";CM$
2430  PRINT D$;"DELETE ";CM$
2440  PRINT D$;"OPEN ";CM$
2450  FOR I = 1 TO NV
2460  PRINT D$;"READ ";VO$
2470  GOSUB 2690
2480  IF N1$ = N$ THEN  GOTO 2600

2490  PRINT D$;"APPEND ";CM$
2500  PRINT D$;"WRITE ";CM$
2510  PRINT N$
2520  PRINT NN
2530  PRINT NA
2540  PRINT SEP$
2550  FOR K = 1 TO NN
2560  PRINT PA$(K)
2570  PRINT NA(K)
2580  NEXT K
2590  PRINT PA$(NN + 1)
2600  NEXT I
2610  PRINT D$;"CLOSE ";VO$
2620  PRINT D$;"DELETE";VO$
2630  PRINT D$"RENAME MACRO.COMOD
     O,";VO$
2640  PRINT D$;"OPEN ";VO$;".CTRL
     "
2650  PRINT D$;"WRITE";VO$;".CTRL
     "
2660  PRINT NV - 1
2670  PRINT D$;"CLOSE"
2680  RETURN
2690  REM ********************
2700  REM  LEGGI MACRO DEF
2710  REM ********************
```

Seguito programma creazione macro.

```
2720  REM
2730 SEP$ = ""
2740  INPUT N$
2750  INPUT NN
2760  INPUT NA
2770  GET A$: IF A$ <  > CHR$ (1
     3) THEN SEP$ = SEP$ + A$: GOTO
     2770
2780  FOR J = 1 TO NN
2790 PA$(J) = ""
2800  GET A$: IF A$ <  > CHR$ (1
     3) THEN PA$(J) = PA$(J) + A$
     : GOTO 2800
2810  INPUT NA(J)
2820  NEXT J
2830 PA$(NN + 1) = ""
2840  GET A$: IF A$ <  > CHR$ (1
     3) THEN PA$(NN + 1) = PA$(NN
      + 1) + A$: GOTO 2840
2850  RETURN
2860  REM ******************
2870  REM   LINE FEED
2880  REM ******************
2890  REM
2900 B = 768
2910  GET A$: IF A$ =  CHR$ (13) OR
     PT = 250 THEN  GOTO 2950
2920  IF A$ =  CHR$ (21) THEN PT =
     PT + 1: HTAB  FN M(PT): PRINT
      CHR$ ( PEEK (B + PT));: GOTO
     2910
2930  IF A$ =  CHR$ (8) THEN PT =
     PT - 1: IF PT =  > 1 THEN  HTAB
      FN M(PT): PRINT  CHR$ ( PEEK
     (B + PT));: GOTO 2910
2940 PT = PT + 1: POKE B + PT, ASC
     (A$): HTAB  FN M(PT): PRINT
      CHR$ ( PEEK (B + PT));: GOTO
     2910
2950 : FOR I = B + 1 TO PT + B:TX
     $ = TX$ +  CHR$ ( PEEK (I)):
      NEXT
2960 PT = 0:B = 0
2970  RETURN
2980  REM ******************
2990  POKE 216,0
3000  REM ******************
3010  REM
3020  HOME : VTAB 12
3030  PRINT "IL VOCABOLARIO ";VO$
     ;" E' VUOTO"
3040  PRINT "PREMI UN TASTO PER C
     ONTINUARE ";
3050  GET A$: PRINT
3060  RUN
```

Listato 2. *Un programma che contiene i riferimenti alla macro LEGGI, PRO, SHOW, DET, AREA.*

```
 10  DIM A(2,3),B(3,2)
 20  INPUT "MAT1:";N1$
 30  INPUT "MAT2:";N2$
 40 LEGGIA,R1,C1,N1$
 50 LEGGIB,R2,C2,N2$
 55  DIM C(R1 + 1,C2 + 1)
 60 PROA,B,C,R1,C1,C2
 70 SHOWC,R1,C2
 80 DETC,R1,AC
 90  PRINT "RISULTATO=";AC
100 AREA(1 / (1 + I ^ 2)),0,.5,50
     ,AC
105 AC = 0
110  PRINT "INTEGRALE=";AC
```

Seguito Listato 2.

Listato 3. *Lo stesso programma del listato 2, in cui i riferimenti alla macro sono stati sostituiti dalle relative istruzioni BASIC.*

```
 10  DIM A(2,3),B(3,2)
 20  INPUT "MAT1:";N1$
 30  INPUT "MAT2:";N2$
 40  PRINT  CHR$ (4);"OPEN";N1$: PRINT
      CHR$ (4);"READ";N1$: INPUT
     ND: INPUT C1: INPUT R1: FOR
     I = 1 TO C1: FOR J = 1 TO R1
     : INPUT A(J,I): NEXT : NEXT
     : REM
 50  PRINT  CHR$ (4);"OPEN";N2$: PRINT
      CHR$ (4);"READ";N2$: INPUT
     ND: INPUT C2: INPUT R2: FOR
     I = 1 TO C2: FOR J = 1 TO R2
     : INPUT B(J,I): NEXT : NEXT
     : REM
 55  DIM C(R1 + 1,C2 + 1)
 60  FOR I = 1 TO R1: FOR J = 1 TO
     C2: FOR K = 1 TO C1:C(I,J) =
     C(I,J) + A(I,K) * B(K,J): NEXT
     : NEXT : NEXT : REM
 70  FOR I = 1 TO R1: FOR J = 1 TO
     C2: PRINT C(I,J);"  ";: NEXT
     : PRINT : NEXT : REM
 80 AC = 0: FOR I = 1 TO R1: IF C(
     I,I) <  > 0 THEN  NEXT :AC =
     1: FOR I = 1 TO R1: FOR J =
     I + 1 TO R1: FOR K = I + 1 TO
     R1:C(J,K) = C(J,K) - C(I,K) *
     C(J,I) / C(I,I): NEXT : NEXT
     :AC = AC * C(I,I): NEXT : REM

 90  PRINT "RISULTATO=";AC
100  FOR I = ((.5 - 0) / 50) / 2 TO
     .5 STEP (.5 - 0) / 50:AC = A
     C + (1 / (1 + I ^ 2)): NEXT
     :AC = AC * (.5 - 0) / 50: REM

105 AC = 0
110  PRINT "INTEGRALE=";AC
```

SERVIZIO SOFTWARE

PERSONAL SOFTWARE

P.S. propone ai propri lettori i dischi o le cassette dei programmi pubblicati. I programmi, provati e garantiti, sono di immediato utilizzo.

P.S. n°	Programma	Sistema	Prezzo	Codice	Supporto
3	La carta del cielo Collisione	Apple II	30.000	1	Disco
3	Backgammon	TRS-80 Mod. I	25.000	2	Disco
2	Editor/Assembler in BASIC	CBM 3032	40.000	3	Disco
4	Interi in precisione multipla Grafica 3D	Apple II	40.000	4	Disco
4	Gioco del calcio	CBM 3032	25.000	5	Disco
5	Pretty printer Shape table	Apple II	30.000	6	Disco
7	Data base modulare	Apple II	25.000	7	Disco
12-13	Wei-ch'i	CMB 3032	20.000	8	Cassetta
14	Tool-Kit	C 64	35.000	9	Cassetta

Per richiedere i programmi in contrassegno, pagando direttamente al postino la cifra indicata, inviare il seguente tagliando

Spedire in busta chiusa a Gruppo Editoriale Jackson - Via Rosellini, 12 - 20124 Milano

Inviatemi i seguenti nastri e/o dischi con i programmi pubblicati su **P.S.**

GRUPPO EDITORIALE JACKSON

Cod. ☐☐☐☐☐ **a L.** .....

Cod. ☐☐☐☐☐ **a L.** .....

Cod. ☐☐☐☐☐ **a L.** .....

Cod. ☐☐☐☐☐ **a L.** .....

Cognome .....

Nome .....

Indirizzo .....

CAP .....

Città .....

Spese postali (contributo fisso) L. 2.000

TOTALE L. .....

che pagherò al postino alla consegna del pacco.

Firma .....

Il VIC ben temperato

Programma musicale con gestione dei dati separata

di *Filippo Pozzi*

La limitata quantità di memoria disponibile nel VIC in configurazione base impedisce spesso lo sviluppo di programmi, come ad esempio quelli musicali oppure quelli in cui vengono definiti nuovi caratteri, contenenti una discreta quantità di dati.

Un utilizzo più razionale della memoria disponibile permette di ridurre notevolmente la parte riservata ai dati.

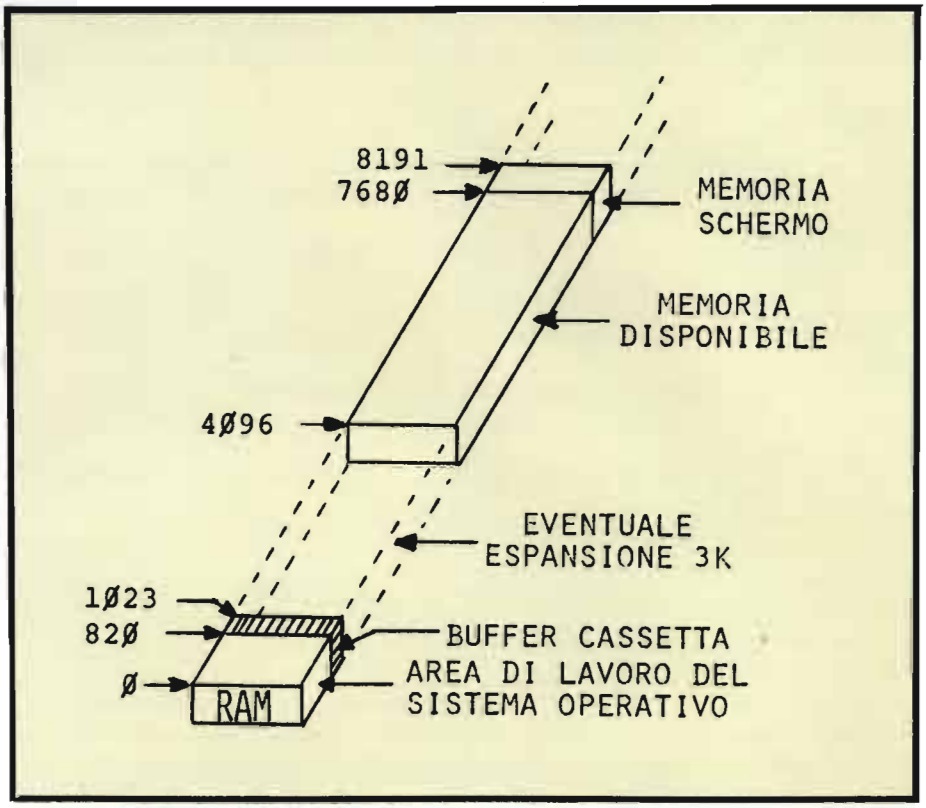


Figura 1. *Rappresentazione dell'area di memoria del VIC dalla locazione 0 alla locazione 8191. È evidenziata l'area normalmente destinata al buffer della cassetta, dove viene inserito il programma caricatore per la ricostruzione del programma principale nell'area della memoria disponibile.*

Il metodo

Nei programmi musicali le varie note sono normalmente affidate ad un'appendice di DATA più o meno lunga ed il BASIC, per gestire questi dati, spreca una notevole quantità di memoria. Per esempio nella linea: 10DATA37, 234, 188 tre soli dati occupano ben 16 byte quando, in teoria, ne potrebbero occupare solo tre.

Perché dunque non gestire in altro modo la memorizzazione dei dati e lasciare al BASIC il resto del programma? La soluzione è relativamente semplice, si tratta di *ricostruire dal di fuori* il programma in oggetto ... ma procediamo con ordine.

Per prima cosa è necessario abbandonare l'area di memoria in cui si opera normalmente, area che va dalla locazione 4096 alla 7679 (i ben noti "3584 BYTE FREE") per spostarsi nel buffer della cassetta (dalla locazione 820 alla 1023), piccola parte della memoria che viene usata dal computer solo quando il registratore è in funzione e che quindi è molto spesso inutilizzata (vedi figura 1).

Per fare ciò si scriva in modo diretto: POKE 43,53: POKE 44,3: POKE 45,55: POKE 46,3: POKE47,55: POKE 48,3: POKE 55,255: POKE 56,3: POKE 820,0 e si prema il RETURN.

Si scriva quindi NEW e si riprema il RETURN. A questo punto si hanno a disposizione i 200 byte del buffer della cassetta (in verità il buffer è limitato a 192 byte) sufficienti per contenere il programma CARICATORE (vedi listato 1). Lo si ricopi attentamente, facendo particolare attenzione al punto e virgola dopo PRINTR; :..nella linea 1. Dopo aver scritto il suddetto programma si dia il RUN e, con estrema attenzione e pazienza, si inseriscano, su richiesta del programma stesso, tutti i dati del listato 3.

In caso di errore nell'introduzione dei dati si prenda nota della locazione in cui è stato inserito il dato errato. Si interrompa il programma e lo si listi come si fa normalmente. Si inserisca quindi detta locazione al posto della prima del ciclo R nella linea 1 del programma CARICA-

TORE (vedi listato 1) e si prema il RETURN; si ridia quindi il RUN, riprendendo così l'inserimento dati dalla locazione del dato errato. Facciamo un esempio per quelli che da tutto questo discorso hanno capito che ... è meglio non sbagliare! Se, per esempio, è stato inserito il dato 25 invece di 215 nella locazione 5126, si premano RUN/STOP e RESTORE insieme per fermare il programma. Dopo averlo LISTato si corregga così la linea 1: 1 FORR = 5126TO6201: ... Dopo il RUN l'inserimento dei dati riprenderà dalla locazione 5126 in cui si inserirà 215, continuando poi normalmente. Lo stesso metodo di correzione vale anche nel caso in cui venga per errore inserito un valore superiore a 255 ottenendo così un ?ILLEGAL QUANTITY ERROR IN 1.

Dopo aver inserito l'ultimo dato il programma chiederà di preparare la cassetta per la successiva memorizzazione su nastro del nuovo programma. Appena pronti si prema F1 (tanto per disturbare un po' i ragni che sotto i tasti-funzione hanno trovato una comoda e tranquilla dimora!) e, seguendo il solito

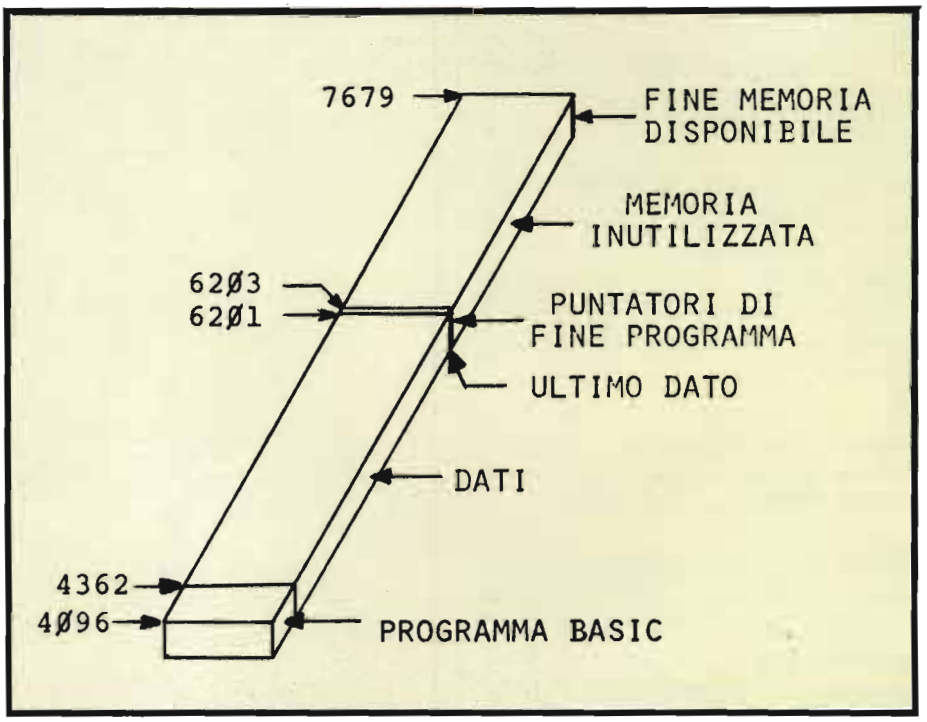


Figura 2. *Particolare dell'area di memoria disponibile dopo aver effettuato il caricamento dei dati. Dei 3584 byte inizialmente disponibili, 267 (dal 4096 al 4362) contengono ora il programma in BASIC, 1839 (dal 4363 al 6201) contengono i dati ed i rimanenti 1478 (dal 6202 al 7679) rimangono inutilizzati. Il posizionamento dei puntatori di fine programma alla locazione 6203 rende la parte dati inscindibile da quella BASIC e il programma può perciò essere "salvato" o "caricato" dalla cassetta come un qualsiasi altro programma.*

```
Programma Caricatore

1 FORR=4096TO6201:PRINTR;:INPUTA:POKER,A:NEXT:PRINT"PREPARARE LA CASSETTA"
2 PRINT"QUINDI PREMERE F1":WAIT198,1:POKE43,1:POKE44,16:POKE45,59
3 POKE46,24:POKE47,59:POKE48,24:POKE55,0:POKE56,30:SAVE
```

Listato 1. *Programma caricatore.*

```
Il VIC ben temperato

1 PRINT"J":V=59765:L=36874:M=L+1:H=M+1:POKEH+2,15:K=4363:T=150
2 A=PEEK(K):B=PEEK(K+1):C=PEEK(K+2):K=K+3:X=RND(8):IFA=2THENT=400
3 POKEL,A:POKEM,B:POKEH,C:IFA=3THENFORT=15TOSTEP-.01:POKEH+2,T:NEXT:END
4 FORR=.TOT:NEXT:POKE8164+X*22,116:POKE38884+X*22,X*6+2:SYSV:GOTO2


VARIANTI PER L'ESECUZIONE DEL PRELUDIO PRIMO

1 PRINT"J":V=59765:L=36874:M=L+1:H=M+1:POKEH+2,15:K=4363:T=100
3 POKEL,0:POKEM,A:POKEH,B:IFA=3THENFORT=15TOSTEP-.01:POKEH+2,T:NEXT:END
```

Listato 2. *Programma "VIC ben temperato".*

PRESS RECORD & PLAY ON TAPE, l'intero programma verrà memorizzato sulla cassetta. Da notare che alla fine dell'operazione comparirà uno strano ?SYNTAX ERROR IN 3, di cui però non si dovrà tener conto.

In conclusione, per mezzo del programma CARICATORE si sono ricostruiti, byte per byte, sia il programma principale (vedi listato 2) che, immediatamente dopo, l'insieme dei dati, ognuno dei quali occupa un solo byte. La chiave di volta sta nella serie di POKE prima del SAVE nel programma CARICATORE. Con essi infatti si inserisce

un *falso* limite del programma: si dà cioè come suo inizio la locazione 4097 e come fine la 6203, quando in effetti il programma terminerebbe alla locazione 4362; i rimanenti byte (dal 4363 al 6201) sono occupati dai dati che così vengono indissolubilmente legati, pur non facendone parte, al programma in BASIC.

Ricapitolando, abbiamo ora un programma di 2106 byte, di cui solo 266 sono riservati al BASIC (vedi figura 2). Lo stesso programma, scritto interamente in BASIC (e quindi con le linee DATA), avrebbe richiesto non meno di 5700 byte e quindi l'aggiunta di un'estensione di memoria.

Il programma

Il programma ora in memoria è quello del listato 2. Non ci si lasci ingannare dal suo aspetto "tarchiato"; esso infatti, come si è appena detto, ha un'invisibile coda di dati. Dopo il RUN il VIC, trasformato in una specie di organo a nastro, eseguirà integralmente la celeberrima Ave Maria che Charles Gounod compose sul Primo Preludio dal "Clavicembalo ben temperato" di J.S.Bach. Nel programma, la variabile T ha funzione di tempo che, alla fine dell'esecuzione (fine linea 2), aumenta notevolmente per dare maggior "respiro" alle ultime battute del brano. L'ultima nota viene infine smorzata dal ciclo della variabile T (qui usata con funzione di volume del suono) nella seconda parte della linea 3.

Alla voce bassa (36874) ed a quella media (36875) è affidata l'esecuzione del Preludio di Bach (con una sola piccola licenza), mentre la voce alta (36876) esegue il "canto" dell'Ave Maria di Gounod. La maggiore durata delle note di questo "canto" è solo apparente, dato che i valori delle note vengono inseriti nelle tre locazioni di memoria ad intervalli uguali.

Listato 3. *I DATA da inserire.*

```
 0, 54, 16, 0, 0, 143, 34, 20, 20, 20, 20, 20, 20, 141, 32, 18, 73, 76, 32, 86,
73, 67, 32, 66, 69, 78, 32, 84, 69, 77, 80, 69, 82, 65, 84, 79, 141, 32, 32, 32,
 32, 32, 18, 68, 73, 32, 70, 46, 80, 79, 90, 90, 73, 0, 112, 16, 1, 0, 153, 34,
147, 34, 58, 89, 178, 53, 57, 55, 54, 53, 58, 76, 178, 51, 54, 56, 55, 52, 58, 7
7, 178, 76, 170, 49, 58, 72, 178, 77, 170, 49, 58, 151, 72, 170, 50, 44, 49, 53,
 58, 75, 178, 52, 51, 54, 51, 58, 84, 178, 49, 53, 48, 0, 165, 16, 2, 0, 65, 178
, 194, 40, 75, 41, 58, 66, 178, 194, 40, 75, 170, 49, 41, 58, 67, 178, 194, 40,
75, 170, 50, 41, 58, 75, 178, 75, 170, 51, 58, 88, 178, 187, 40, 56, 41, 58, 139
, 65, 178, 50, 167, 84, 178, 52, 48, 48, 0, 213, 16, 3, 0, 151, 76, 44, 65, 58,
151, 77, 44, 66, 58, 151, 72, 44, 67, 58, 139, 65, 178, 51, 167, 129, 84, 178, 4
9, 53, 164, 67, 169, 171, 46, 48, 49, 58, 151, 72, 170, 50, 44, 84, 58, 130, 58,
 128, 0, 9, 17, 4, 0, 129, 82, 178, 46, 164, 84, 58, 130, 58, 151, 56, 49, 54, 5
2, 170, 88, 172, 50, 50, 44, 49, 49, 54, 58, 151, 51, 56, 56, 56, 52, 170, 88, 1
72, 50, 50, 44, 88, 172, 54, 170, 50, 58, 158, 89, 58, 137, 50, 0, 0, 0, 0, 195,
 0, 0, 207, 0, 0, 215, 0, 0, 225, 0, 0, 231, 0, 0, 215, 0, 0, 225, 0, 0, 231, 0,
 0, 195, 0, 0, 207, 0, 0, 215, 0, 0, 225, 0, 0, 231, 0, 0, 215, 0, 0, 225, 0, 0,
 231, 0, 0, 195, 0, 0, 201, 0, 0, 219, 0, 0, 228, 0, 0, 232, 0, 0, 219, 0, 0, 22
8, 0, 0, 232, 0, 0, 195, 0, 0, 201, 0, 0, 219, 0, 0, 228, 0, 0, 232, 0, 0, 219,
 0, 0, 228, 0, 0, 232, 0, 0, 191, 0, 0, 201, 0, 0, 215, 0, 0, 228, 0, 0, 232, 0,
 0, 215, 0, 0, 228, 0, 0, 232, 0, 0, 191, 0, 0, 201, 0, 0, 215, 0, 0, 228, 0, 0,
232, 0, 0, 215, 0, 0, 228, 0, 0, 232, 0, 0, 195, 0, 0, 207, 0, 0, 215, 0, 0, 225
, 0, 0, 231, 0, 0, 215, 0, 0, 225, 0, 0, 231, 0, 0, 195, 0, 0, 207, 0, 0, 215, 0
, 0, 225, 0, 0, 231, 0, 0, 215, 0, 0, 225, 0, 0, 231, 0, 0, 195, 207, 0, 207, 20
7, 0, 215, 207, 0, 225, 207, 0, 231, 207, 0, 215, 207, 0, 225, 207, 0, 231, 207,
 0, 195, 207, 0, 207, 207, 0, 215, 207, 0, 225, 207, 0, 231, 207, 0, 215, 207, 0
, 225, 207, 0, 231, 207, 0, 195, 209, 0, 201, 209, 0, 219, 209, 0, 228, 209, 0,
232, 209, 0, 219, 209, 0, 228, 209, 0, 232, 209, 0, 195, 209, 0, 201, 209, 0, 21
9, 209, 0, 228, 209, 0, 232, 0, 0, 219, 0, 0, 228, 209, 0, 232, 209, 0, 191, 215
, 0, 201, 215, 0, 215, 215, 0, 228, 215, 0, 232, 215, 0, 215, 215, 0, 228, 215,
0, 232, 215, 0, 191, 215, 0, 201, 215, 0, 215, 215, 0, 228, 215, 0, 232, 201, 0,
 215, 201, 0, 228, 201, 0, 232, 201, 0, 195, 207, 0, 207, 207, 0, 215, 207, 0, 2
25, 207, 0, 231, 207, 0, 215, 207, 0, 225, 207, 0, 231, 207, 0, 195, 207, 0, 207
, 207, 0, 215, 0, 0, 225, 0, 0, 231, 0, 0, 215, 0, 0, 225, 0, 0, 231, 207, 0, 19
5, 219, 0, 207, 219, 0, 219, 219, 0, 231, 219, 0, 237, 219, 0, 219, 219, 0, 231,
 219, 0, 237, 219, 0, 195, 219, 0, 207, 219, 0, 219, 183, 0, 231, 183, 0, 237, 1
91, 0, 219, 191, 0, 231, 195, 0, 237, 195, 0, 195, 201, 0, 201, 201, 0, 212, 201
, 0, 219, 201, 0, 228, 201, 0, 212, 201, 0, 219, 207, 0, 228, 207, 0, 195, 201,
0, 201, 201, 0, 212, 201, 0, 219, 201, 0, 228, 0, 0, 212, 0, 0, 219, 0, 0, 228,
201, 0, 191, 215, 0, 201, 215, 0, 215, 215, 0, 228, 215, 0, 235, 215, 0, 215, 21
5, 0, 228, 215, 0, 235, 215, 0, 191, 215, 0, 201, 215, 0, 215, 175, 0, 228, 175,
 0, 235, 183, 0, 215, 183, 0, 228, 191, 0, 235, 191, 0, 191, 195, 0, 195, 195, 0
, 207, 195, 0, 215, 195, 0, 225, 195, 0, 207, 195, 0, 215, 201, 0, 225, 201, 0,
191, 195, 0, 195, 195, 0, 207, 195, 0, 215, 195, 0, 225, 0, 0, 207, 0, 0, 215, 0
, 0, 225, 0, 0, 183, 225, 0, 195, 225, 0, 207, 225, 0, 215, 225, 0, 225, 225, 0,
 207, 225, 0, 215, 225, 0, 225, 225, 0, 183, 225, 0, 195, 225, 0, 207, 195, 0, 2
15, 195, 0, 225, 201, 0, 207, 201, 0, 215, 207, 0, 225, 207, 0, 147, 212, 0, 183
, 212, 0, 201, 212, 0, 212, 212, 0, 225, 212, 0, 201, 212, 0, 212, 207, 0, 225,
207, 0, 147, 201, 0, 183, 201, 0, 201, 201, 0, 212, 201, 0, 225, 183, 0, 201, 18
3, 0, 212, 183, 0, 225, 183, 0, 175, 191, 0, 191, 191, 0, 201, 191, 0, 215, 191,
 0, 223, 191, 0, 201, 191, 0, 215, 191, 0, 223, 191, 0, 175, 0, 0, 191, 0, 0, 20
1, 0, 0, 215, 0, 0, 223, 201, 0, 201, 201, 0, 215, 201, 0, 223, 201, 0, 173, 207
, 0, 187, 207, 0, 207, 207, 0, 215, 207, 0, 227, 207, 0, 207, 207, 0, 215, 207,
0, 227, 207, 0, 175, 207, 0, 187, 207, 0, 207, 207, 0, 215, 207, 0, 227, 209, 0,
 207, 209, 0, 215, 215, 0, 227, 215, 0, 163, 219, 0, 183, 219, 0, 201, 219, 0, 2
09, 219, 0, 225, 219, 0, 201, 219, 0, 209, 219, 0, 225, 183, 0, 163, 183, 0,
 183, 183, 0, 201, 183, 0, 209, 183, 0, 225, 0, 0, 201, 0, 0, 209, 0, 0, 225, 20
1, 0, 163, 201, 0, 179, 201, 0, 201, 201, 0, 209, 201, 0, 223, 201, 0, 201, 201,
 0, 209, 201, 0, 223, 201, 0, 163, 201, 0, 179, 201, 0, 201, 201, 0, 209, 201, 0
, 223, 207, 0, 201, 207, 0, 209, 209, 0, 223, 209, 0, 159, 215, 0, 175, 215, 0,
195, 215, 0, 215, 215, 0, 225, 215, 0, 195, 215, 0, 215, 215, 0, 225, 215, 0, 15
9, 175, 0, 175, 175, 0, 195, 175, 0, 215, 175, 0, 225, 0, 0, 195, 0, 0, 215, 0,
 0, 225, 195, 0, 159, 195, 0, 163, 195, 0, 183, 195, 0, 195, 195, 0, 209, 195, 0,
 183, 195, 0, 195, 195, 0, 209, 195, 0, 159, 195, 0, 163, 195, 0, 183, 195, 0, 1
95, 195, 0, 209, 201, 0, 183, 201, 0, 195, 207, 0, 209, 207, 0, 147, 209, 0, 163
, 209, 0, 183, 209, 0, 195, 209, 0, 209, 209, 0, 183, 209, 0, 195, 209, 0, 209,
209, 0, 147, 209, 0, 163, 209, 0, 183, 209, 0, 195, 209, 0, 209, 215, 0, 183, 21
5, 0, 195, 219, 0, 209, 219, 175, 0, 223, 201, 0, 223, 215, 0, 223, 223, 0, 223,
 232, 0, 223, 215, 0, 223, 223, 0, 219, 232, 0, 219, 175, 0, 215, 201, 0, 215, 2
15, 0, 215, 223, 0, 215, 232, 0, 201, 215, 0, 201, 223, 0, 201, 232, 0, 201, 195
, 0, 207, 207, 0, 207, 215, 0, 207, 225, 0, 207, 231, 0, 207, 215, 0, 207, 225,
0, 207, 231, 0, 207, 195, 0, 207, 207, 0, 207, 215, 0, 0, 225, 0, 0, 231, 0, 0,
215, 0, 0, 225, 0, 0, 231, 0, 0, 195, 0, 215, 215, 0, 215, 221, 0, 215, 225, 0,
215, 231, 0, 215, 221, 0, 215, 225, 0, 215, 231, 0, 215, 195, 0, 207, 215, 0, 20
7, 221, 0, 207, 225, 0, 207, 231, 0, 0, 221, 0, 0, 225, 0, 0, 231, 0, 207, 163,
0, 219, 209, 0, 219, 219, 0, 219, 225, 0, 219, 231, 0, 219, 219, 0, 219, 225, 0,
 219, 231, 0, 219, 163, 0, 183, 209, 0, 183, 219, 0, 183, 225, 0, 183, 231, 0, 0
, 219, 0, 0, 225, 0, 0, 231, 0, 0, 167, 0, 219, 195, 0, 219, 219, 0, 219, 225, 0
, 219, 230, 0, 219, 219, 0, 219, 225, 0, 219, 230, 0, 219, 167, 0, 195, 195, 0,
195, 219, 0, 195, 225, 0, 195, 230, 0, 0, 219, 0, 0, 225, 0, 0, 230, 0, 219, 179
, 0, 225, 209, 0, 225, 223, 0, 225, 225, 0, 225, 228, 0, 225, 223, 0, 225, 225,
0, 225, 228, 0, 225, 179, 0, 203, 209, 0, 203, 223, 0, 203, 225, 0, 203, 228, 0,
 0, 223, 0, 0, 225, 0, 0, 228, 0, 223, 175, 0, 225, 209, 0, 225, 215, 0, 225, 22
3, 0, 225, 228, 0, 225, 215, 0, 225, 223, 0, 225, 228, 0, 225, 175, 0, 201, 209,
 0, 201, 215, 0, 0, 223, 0, 0, 228, 0, 201, 215, 0, 201, 223, 0, 201, 228, 0, 20
1, 175, 0, 201, 207, 0, 201, 215, 0, 201, 225, 0, 201, 231, 0, 201, 215, 0, 201,
```

Seguito listato 3.

```
 225, 0, 201, 231, 0, 201, 175, 0, 201, 207, 0, 201, 215, 0, 201, 225, 0, 201, 2
31, 0, 195, 215, 0, 195, 225, 0, 191, 231, 0, 191, 175, 0, 215, 201, 0, 215, 215
, 0, 215, 225, 0, 215, 232, 0, 215, 215, 0, 215, 225, 0, 207, 232, 0, 207, 175,
0, 195, 201, 0, 195, 215, 0, 195, 225, 0, 195, 232, 0, 0, 215, 0, 0, 225, 0, 0,
232, 0, 195, 175, 0, 209, 201, 0, 209, 215, 0, 209, 223, 0, 209, 232, 0, 209, 21
5, 0, 209, 223, 0, 209, 232, 0, 209, 175, 0, 209, 201, 0, 209, 215, 0, 209, 223,
 0, 209, 232, 0, 207, 215, 0, 207, 223, 0, 201, 232, 0, 201, 175, 0, 228, 203, 0
, 228, 219, 0, 228, 225, 0, 228, 234, 0, 228, 219, 0, 228, 225, 0, 223, 234, 0,
223, 175, 0, 215, 203, 0, 215, 219, 0, 215, 225, 0, 215, 234, 0, 0, 219, 0, 0, 2
25, 0, 0, 234, 0, 215, 175, 0, 219, 207, 0, 219, 215, 0, 219, 225, 0, 219, 235,
0, 219, 215, 0, 219, 225, 0, 219, 235, 0, 219, 175, 0, 219, 207, 0, 219, 215, 0,
 219, 225, 0, 219, 235, 0, 223, 215, 0, 223, 225, 0, 225, 235, 0, 225, 175, 0, 2
31, 201, 0, 231, 215, 0, 231, 225, 0, 231, 232, 0, 231, 215, 0, 231, 225, 0, 231
, 232, 0, 231, 175, 0, 231, 201, 0, 231, 215, 0, 225, 225, 0, 225, 232, 0, 215,
215, 0, 215, 225, 0, 207, 232, 0, 207, 175, 0, 201, 201, 0, 201, 215, 0, 201, 22
3, 0, 201, 232, 0, 201, 215, 0, 201, 223, 0, 201, 232, 0, 201, 175, 0, 201, 201,
 0, 201, 215, 0, 219, 223, 0, 219, 232, 0, 223, 215, 0, 223, 223, 0, 219, 232, 0
, 219, 135, 0, 215, 195, 0, 215, 215, 0, 228, 221, 0, 228, 231, 0, 223, 215, 0,
223, 221, 0, 215, 231, 0, 215, 135, 0, 209, 195, 0, 209, 215, 0, 201, 221, 0, 20
1, 231, 0, 191, 215, 0, 191, 221, 0, 175, 231, 0, 175, 135, 0, 195, 195, 0, 195,
 0, 163, 195, 0, 183, 195, 0, 195, 195, 0, 209, 195, 0, 195, 195, 0, 183, 195, 0
, 195, 195, 0, 183, 195, 0, 163, 195, 0, 183, 195, 0, 163, 195, 0, 147, 195, 0,
163, 195, 0, 147, 195, 135, 0, 215, 191, 0, 215, 0, 215, 215, 0, 223, 215, 0, 22
8, 215, 0, 232, 215, 0, 228, 215, 0, 223, 215, 0, 228, 215, 0, 223, 215, 0, 215,
 215, 2, 223, 215, 0, 201, 0, 0, 209, 0, 0, 0, 207, 0, 0, 201, 0, 135, 0, 0, 195, 0
, 0, 0, 207, 0, 0, 215, 0, 3, 225, 0.
```

Volendo ascoltare il solo Preludio (stupendo nella sua apparente semplicità) si apportino al programma le seguenti modifiche:
linea 1: Q = 100 (invece di Q = 150)
linea 3: POKEL, 0: POKEM,A: POKEH,B (invece di POKEL,A: POKEM,B: POKEH,C).
Con opportune variazioni del programma si possono inoltre ottenere numerose variazioni sul tema. Si provi ad esempio a porre Q = 090 (non Q = 90) nella linea 1, K = K + 4 nella linea 2 ed a variare la linea 3 come per l'ascolto del Preludio.
Le ulteriori variazioni dipenderanno unicamente dalla nostra fantasia; si dovrà fare solo attenzione a non variare la lunghezza del programma (ecco perché Q = 090) oppure, variandone la lunghezza, bisognerà aggiornare il valore di K nella linea 1 secondo la formula: K = PEEK(45) + 256 ★ PEEK(46) − 1840. E Bach mi perdoni.

REMarks listato 1

Linea 1 - Viene definita la variabile di ciclo R con valore da 4096 a 6201. Essa viene di volta in volta stampata con il punto interrogativo dell'INPUT (si noti il punto e virgola dopo PRINTR).
Il valore inserito è quindi trasferito, come variabile A, nella locazione R. Alla fine del ciclo viene stampato il messaggio per la preparazione della cassetta.

Linea 2 - Continua il messaggio della linea 1 con la richiesta di premere il tasto-funzione F1 e lo si attende con WAIT198,1.
Segue la serie di POKE necessari per il posizionamento dei puntatori alle varie locazioni:

Puntatori	Funzione	Valori inseriti	Locazione
43-44	Inizio BASIC	1-16	4097
45-46	Inizio variabili	59-24	6203
47-48	Fine variabili	59-24	6203
55-56	Fine memoria	0-30	7680

Linea 3 - Continuano i POKE; alla fine il comando SAVE provoca la comparsa del messaggio PRESS RECORD & PLAY ON TAPE per il "salvataggio" su cassetta del programma.

REMarks listato 2

Linea 1 - Dopo la pulizia dello schermo vengono definite le variabili. Y: inizio della routine di scroll del sistema operativo. L: voce bassa. M: voce media. H: voce alta. Viene portato il volume al massimo (POKEH + 2,15). K: prima locazione di memoria occupata dai dati. T: variabile di ritardo con funzione di tempo.

Linea 2 - Vengono definite le tre variabili (A, B, C) che di volta in volta verranno inserite nelle tre voci, rispettivamente bassa, media ed alta. I valori alle tre voci vengono assegnati "leggendo" (PEEK), per mezzo della variabile K, il contenuto delle locazioni nella parte di memoria riservata ai dati (dalla locazione 4363 alla 6201). Il valore di K viene aumentato di tre unità per poter leggere le successive note. Viene definita la variabile casuale X. Verso la fine del brano, in corrispondenza del valore 2 di A, il tempo T viene aumentato da 150 a 400.

Linea 3 - I valori delle tre variabili vengono inseriti nelle rispettive locazioni di memoria. In corrispondenza dell'ultima nota il valore di A sarà uguale a 3 per cui, con il ciclo di fine riga (dove viene riutilizzata la variabile T ormai inutile come tempo), il volume viene portato lentamente (STEP-.01) a zero ed il programma termina.

Linea 4 - Il ciclo della variabile R serve per creare il tempo (.TOT corrisponde a 0 TO T). Con i due POKE successivi si stampa, in una posizione a caso sull'ultima riga dello schermo, il carattere corrispondente al foro del nastro di carta simulato e lo si colora con un colore scelto a caso. Viene quindi provocato lo scroll dello schermo (SYSY) e si ritorna alla linea 2. ■

PERSONAL SOFTWARE

Missione ufo

Il VIC 20 sfruttato ai limiti

di *Renato Comini*

Supponete di essere un alieno ... e scendete con la vostra astronave verso una metropoli terrestre. Dovete distruggere i grattacieli prima di schiantarvi contro essi e per far questo potete sganciare le vostre bombe, ma dovete anche fare attenzione a non esaurire l'energia. Il programma gira su VIC 20 non espanso con 3,5 Kbyte di RAM ed è un ibrido composto da BASIC e codice macchina. Il gioco dispone di ben nove livelli di difficoltà, indicazione del punteggio più alto, indicazione del carburante, indicazione delle astronavi a disposizione (tramite delle piccole facce che ammiccano in continuazione). Il tutto è poi completato dall'uso di ben 64 caratteri in HI-RES che vengono definiti dall'utente.

Il programma è costituito da 2 differenti parti: questo è servito a poter compattare il tutto nei 3,5 Kbyte. Abbiamo perciò due programmi, il maggiore dei quali in termini di complessità è il primo (figura 1). Questo contiene infatti le specifiche per generare i caratteri in HI-RES e i dati per creare 5 routine in linguaggio macchina che hanno un'importanza fondamentale (suono ecc.).

Tale programma occupa precisamente 3,5 Kbyte e va scritto esattamente come riportato nel list. L'alto numero delle virgole una dopo l'altra rappresenta gli zeri: vanno quindi ricopiate esattamente come sono.

Questo programma deve essere salvato ed accuratamente verificato perché ogni minimo errore può portare a conseguenze irrimediabili riguardo al corretto funzionamento del gioco. Prestare la massima cura nel digitare è essenziale: una virgola in più od in meno ne determina il funzionamento. Alcune linee superano gli 88 caratteri, così per inserirle completamente è necessario usare la forma abbreviata della parola DATA, ovvero D shift A. Come si noterà alcuni dati vengono "pokati" in alcune locazioni strane come quelle libere da 673 a 767.

Tramite l'utilizzo di queste locazioni libere si riesce ad adattare in 3,5 Kbyte di memoria un programma che prenderebbe comodamente più di 4 Kbyte.

Anche il secondo programma (figura 2) va digitato con estrema cautela poiché vengono usate più di 40 variabili ed anche qui compaiono linee al di sopra degli 88 caratteri. Lo schermo viene posto rettangolare, 26 per 19, tramite le istruzioni contenute nella linea 10000. I punteggi vengono visualizzati in modo eccellente. Attenzione, perché una volta lanciato il programma non può più essere fermato (*è quindi fondamentale salvarlo prima di dare il RUN*). Infatti le istruzioni della linea 5 disabilitano i tasti RUN STOP e RESTORE.

La linea 5 può essere omessa permettendo così di fermare il programma per eventuali modifiche prima di salvarlo definitivamente.

Se avete ora salvato i due programmi uno dietro l'altro non resta che caricare e lanciare il primo il quale alla fine caricherà e lancierà il secondo.

Alla partenza del gioco le istruzioni vengono stampate al centro dello schermo mentre tre verdi testoline sorridono e si accigliano all'unisono al tempo di un frenetico boogie che apprezzerete dall'altoparlante del vostro TV.

Le istruzioni sono molto semplici: la barra dello spazio permette di selezionare il livello di difficoltà del gioco che poi rimane costantemente indicato sullo schermo (in alto in centro); con F1 si fa partire il gioco.

Alla partenza lo schermo viene pulito (con l'uso di una routine in linguaggio macchina) e sotto ai vostri occhi compare una città costituita da grattacieli multicolori. La vostra nave stellare gradatamente perderà quota mentre voi, freneticamente, lancerete bombe sia per distruggere la città, sia per evitare di schiantarvi contro le case più alte. Ogni tasto va bene per sparare, ma il più comodo è forse la barra. Attenzione perché ad ogni sparo diminuisce il carburante. Un altro interessante aspetto del gioco è dato dalla possibilità di poterlo fermare per poi riprenderlo sempre dallo stesso punto. Si ha quindi la possibilità di fare una pausa premendo il primo tasto in alto a sinistra. Per riprendere il gioco basta premere nuovamente lo stesso tasto. In caso di completamento del carburante l'astronave esploderà è lo stesso accadrà schiantandosi contro un grattacielo.

Nel caso di completamento dell'attacco (città distrutta completamente) si avrà invece un bonus di punti dipendente dall'energia avanzata, ed inoltre un piccolo alieno uscirà dall'astronave congratulandosi.

L'altezza dei grattacieli aumenta a seconda degli schermi completati.

Una nuova astronave viene regalata dopo 10000 punti e tale punteggio è anche il minimo da realizzare per l'HI-SCORE.

Le routine in linguaggio macchina sono abbastanza complesse, la prima provvede a cambiare espressione alle piccole faccie (circa due

volte al secondo) alterando dei valori numerici nei byte della generazione caratteri. La seconda routine controlla se viene premuto il tasto per la pausa e, se così, ferma il gioco per riprenderlo poi ad una nuova pressione del medesimo tasto.
La terza routine fa uso di un contatore ed un puntatore ad un listato di note musicali. Il listato delle note è contenuto in 0,5 Kbyte dalla locazione 37888 alla locazione 38399 (nibble colore non usati). Poiché soltanto i primi 4 bit di ogni byte sono utilizzati ogni nota è così scritta in due parti.
I primi 4 bit di ogni nota sono nella lista che parte dal byte 37888 e i secondi 4 sono nella lista che parte dal byte 38144. La nota finale di ogni list ha uno zero. La melodia può essere cambiata cambiando le note, per far questo si utilizzeranno le seguenti linee BASIC:

```
1 FOR I = 0 TO 99 : INPUTN : IF N =−1 THEN END
2 Q=(N AND 240)/16/POKE 37888+I,Q
3 POKE 38144+I,N-Q ★ 16
4 NEXT I
```

Lanciando questo piccolo programma altro non bisogna fare che inserire le note una dietro l'altra. Detto questo è detto tutto, non resta che digitare questo spettacolare programma. ■

Figura 1 - *Il programma numero 1.*

```
 10 PRINT"■":POKE55,0:POKE56,28:CLR
 100 FORZ=7168TO7679:READX:POKEZ,X:NEXT
:FORZ=673TO751:READX:POKEZ,X:NEXT
 200 DATA254,254,146,146,146,146,254,25
4,146,254,146,254,146,254,146,254,254,6
,254,254
 210 DATA254,68,254,254,254,170,170,170
,170,170,170,254,,16,16,56,56,124,254,2
,16,56
 220 DATA56,254,170,254,170,254,16,16,1
6,56,56,254,130,254,16,16,16,56,124,124
,54,254
 240 DATA,108,108,108,108,254,254,254,,
36,60,24,36,60,24,
 250 DATA255,255,136,85,34,255,255,,,,,
,2,142,238,254,,,,,,18,86,246
 260 DATA,,,8,10,94,126,254,,,2,130,138
,218,222,254
 270 DATA,,,,128,146,214,246,,,,128,192
,200,202,238
 280 DATA,,219,146,210,82,219,,,,179,17
0,179,170,171,,,,87,82,114,82,87,
 290 DATA,,117,69,87,85,117,
 300 DATA124,254,198,186,254,214,214,12
4,124,186,198,254,214,214,254,124
 320 DATA,233,137,137,233,137,137,134,,
116,68,68,116,68,68,119
 340 DATA3,15,60,127,201,127,63,8,128,2
24,120,252,38,252,248,32
 360 DATA1,,8,36,129,40,5,,,80,129,40,6
4,26,64,40
 380 DATA0,34,136,6,32,10,,18,64,,32,13
2,32,138,32,
 400 DATA,,,,,,,,,,,,,,,,
 420 DATA153,153,126,24,60,36,66,195,24
,24,126,153,60,36,36,102
 440 DATA3,2,34,2,2,2,34,3
 450 DATA255,,255,255,255,255,,255,255,
,254,254,254,254,,255
 460 DATA255,,252,252,252,252,,255,255,
,248,248,248,248,,255
 470 DATA255,,240,240,240,240,,255,255,
,224,224,224,224,,255
 480 DATA255,,192,192,192,192,,255,255,
,128,128,128,128,,255
 510 DATA255,,,,,,,255,,,154,146,154,14
6,217,
 550 DATA,,180,164,180,164,54,,255,141,
185,141,189,189,189,255
 570 DATA56,108,198,198,198,108,56,,24,
120,24,24,24,24,126,
 580 DATA124,198,6,12,56,96,254,,124,19
8,6,28,6,198,124,
 590 DATA28,60,116,230,254,12,12,,254,1
92,252,6,6,198,124,
 600 DATA60,96,192,252,198,198,124,,254
,198,12,24,48,96,96,
 610 DATA124,198,198,124,198,198,124,,1
24,198,198,126,6,12,120,,,,,32,,,32,
 620 DATA255,136,186,138,232,235,139,25
5,255,140,171,171,139,171,172,255
 640 DATA255,199,95,199,223,95,199,255
 660 DATA255,239,223,129,223,239,255,25
5,192,160,160,192,14,21,21,21
 1010 DATA120,169,174,141,20,3,169,2,14
1,21,3,88,96,230,254,165,254,41,16,74,1
70,160,8,189
 1020 DATA168,28,153,247,28,232,136,208
,246,165,197,201,8,208,37,160,,140,14,1
44,32,159,255
 1030 DATA165,197,201,8,240,247,32,159,
255,165,197,201,8,208,247,32,159,255,16
5,197,201,8
 1040 DATA240,247,160,15,140,14,144,76,
77,1
 2000 FORZ=319TO414:READX:POKEZ,X:NEXT:
FORZ=0TO73:READX,Y:POKE37888+Z,X:POKE38
144+Z,Y:NEXT
 2010 DATA162,,169,32,157,52,30,157,186
,30,202,208,247,96
```

Seguito figura 1.

```
 2020 DATA165,1,240,3,206,12,144
 2030 DATA165,,240,57,198,249,208,53,17
3,10,144,201,127,240,12,230,249
 2035 DATA169,127,141,10,144,141,11,144
 2040 DATA208,34,166,250,230,250,169,15
,61,,148,10,10,10,10,133,248
 2045 DATA169,15,61,,149
 2050 DATA101,248,240,13,141,11,144,141
,10,144,169,7,133,249,76,194,234,169,1,
133,249
 2060 DATA169,,133,250,76,194,234
 3010 DATA12,3,12,3,12,12,12,14,13,7,13
,7,12,12,12,14,12,3,12,3,12,12,12,14,13
,7,13,7
 3015 DATA12,15,13,7,13,1
 3020 DATA13,1,13,8,13,11,14,1,14,1,13,
8,13,11,13,1,13,1,13,8,13,9,14,1,14,1,1
3,9,14,1
 3030 DATA13,7,13,7,13,13,13,15,14,4,14
,4,13,15,13,7,13,1,13,1,13,8,13,10,14,1
,14,1,13,10
 3040 DATA13,1,12,3,12,3,12,12,12,14,13
,7,13,7,12,11,12,13,13,7,13,7,13,1,13,1
,12,15,12
 3050 DATA15,12,9,12,9,,,11,7,11,7,10,1
5,10,15,10,3,10,3,9,15,9,15,,
 3060 POKE198,7:POKE631,76:POKE632,207:
POKE633,13:POKE634,82:POKE635,213:POKE6
36,13


Lista simboli grafici

    10 : 1  SHIFT HOME =CHR$(147)
```

Figura 2. *Il programma numero 2.*

```
 5 POKE37150,2
 10 PRINTCHR$(8):CLR:B=1000:GOTO10000
 20 POKE36879,8:PRINT"■"CHR$(8):CLR:B=5
000:GOTO10000
 50 IFS>=1000ANDC1=0THENC1=1:L=L+1:POKE
38889+L,5:POKE8169+L,31
 60 PRINT"■"TAB(9-LEN(STR$(S)))RIGHT$(
STR$(S),LEN(STR$(S))-1):RETURN
 100 IFM>8095THENRETURN
 103 IFPEEK(K)=JANDO=0THENGOSUB20000:RE
TURN
 105 IFPEEK(8151)=44THEND1=1:POKEE,0:GO
SUB20000:RETURN
 110 IFO=0THENPOKEV-E,240:O=E:N=M+E:POK
EV-2,235:POKEE,E:POKEV-E,0:GOSUB1500:IF
A5=ETHENRETURN
 120 N=N+D:IFPEEK(N)<>ITHENPOKEE,0:POKE
V-2,0:GOTO160
 130 IFPEEK(N-D)<>DANDPEEK(N-D)<>25THEN
POKEN-D,I
 140 POKEN+C,P:POKEN,Q:RETURN
 150 POKEN-D,I:POKEN+C,P:POKEN,Q:RETURN
 160 IFN+D<8122THENPOKEN+D,RND(E)*6+11
 170 IFPEEK(N)<>A2THENPOKEN,I:S=S+2+W
```

Seguito figura 2.

```
 180 IFPEEK(N-D)<>DANDPEEK(N-D)<>25THEN
POKEN-D,I
 190 POKEV-E,130
 200 GOSUB50:O=0:POKEV-E,0:N=A3:RETURN
 1000 POKEM,I:M=M+E:IFM=8107THENA6=E
 1010 POKEC+M,F:POKEH+M,F:POKEM,D-E:POK
EM+E,D:IFPEEK(M+2)=ITHENRETURN
 1020 B1=1:RETURN
 1500 IFPEEK(8151)=44THENA7=1:RETURN
 1510 IFA8=1THEN1530
 1520 A5=A5+1:IFA5>2THENA5=0:RETURN
 1530 POKEA4,PEEK(A4)+1:IFPEEK(A4)=44TH
ENA4=A4-1
 1540 RETURN
 2000 POKEV-2,0
 2005 POKEN,I:POKEV+E,138:POKE0,0:POKEV
-3,0:POKEM,I:POKEM+E,I:M=M+E:FORT=252TO
128STEP-4
 2010 POKEV-E,T:POKEV-T,4:R=RND(1)*15+E
:POKEM+C,R:POKEM+H,R:POKEM,27:R=RND(1)*
15+E
 2020 B1=0:POKEM+E,28:POKEM+C,R:POKEM+H
,R:POKEM,29:POKEM+E,30:NEXT:POKEM,I:POK
EM+E,I
 2025 IFD1<>1THENPOKEM+27,RND(1)*6+11
 2030 POKEV+E,8:FORT=15TO0STEP-.1:POKEV
,T:NEXT:L=L-1:POKE8170+L,I
 2040 O=0:POKE1,0:POKEV-E,0:IFL=0THEN35
00
 2050 FORT=8151TO8167:POKET,36:NEXT:POK
ET,37:A4=8168:FORT=0TO999:NEXT:B1=0:GOT
O7000
 2090 FORT=8151TO8167:POKET,36:NEXT:POK
ET,37:FORT=0TO2000:NEXT:GOTO8000
 3000 IFS>=BTHENB=S
 3010 PRINT"■"TAB(26-LEN(STR$(B)))RIGH
T$(STR$(B),LEN(STR$(B))-1);:RETURN
 3500 FORT=0TO999:NEXT:PRINT"■"TAB(136)
"■,──────────■"TAB(162)"|■■ ■ GAME O
VER |
 3510 PRINT"■■"TAB(188)"└──────────┘"
:POKEV,15:POKEV-3,0:POKEV-4,0:RESTORE:F
ORT=0TO7
 3520 READC,D:POKEV-2,C:FORU=0TO50*D:NE
XT:POKEV-2,0:NEXT:GOTO9000
 3530 DATA225,4,215,2,215,2,219,4,215,8
,223,4,225,4,0,50
 5000 PRINT"■"TAB(163)"■■MISSIONE OK!!"
:POKEM+30723,3:POKE0,0:POKE36874,0:POKE
36875,0
 5010 A8=E:FORT=0TO9:POKEM+3,33:FORU=0T
O99:NEXT:POKEM+3,34:FORU=0TO99:NEXT:NEX
T:A7=0
 5020 POKE36876,240:GOSUB1500:GOSUB50:P
OKE36876,0:S=S+2:IFA7=0THEN5020
 5030 A6=0:A8=0:FORT=0TO999:NEXT:GOTO20
90
 7000 D1=0:M=7732:POKEV,15:POKE0,1:POKE
250,0:POKEX,6
 7010 :GOSUB1000:GOSUB100:GOSUB100:FORT
```

Seguito figura 2.

```
=0TO30:NEXT:IFA6=1THEN5000
 7020 IFB1=1ORD1=1THEN2000
 7030 GOTO7010
 8000 A4=8168:POKE0,0:W=W+1:SYS319:POKE
V,0:POKEY,65:POKEZ,65:IFW>5THENW=5
 8010 FORT=ATOA+21:IFRND(1)<.4-(G/30)-W
/100THENNEXT:GOTO7000
 8020 R1=RND(1)*7+1:R2=RND(1)*6+1+(G/2)
+(W/3):R3=RND(1)*4:R4=RND(1)*5+4
 8030 FORU=TTOT-D*R2STEP-D:POKEC+U,R1:P
OKEU,R3:NEXT:POKEC+U,R1:POKEU,R4:NEXT:G
OTO7000
 9000 PRINT"■";:POKE38414,3:POKE7694,48
+G:PRINT"■R:▙000000  ■-.:▙■  ■ST:▙0000
00";
 9010 FORT=0TO25:PRINT"■■";:NEXT:GOSU
B3000:FORT=0TO17:PRINT:NEXT:PRINT"■■";:
FORT=0TO25
 9020 PRINT"■J";:NEXT:PRINT"WX▙########
**********%■ ■←←←";
 9030 PRINT"■"TAB(85)"■MISSIONE UFO"TA
B(71)"■"TAB(57)"■/■ ■= ST
ART"
 9040 PRINTTAB(45)"■■;<=■ ■=■LIVELLO GI
OCO"TAB(81)"■>■ = PAUSA"
 9050 V=36878:Y=374:Z=385:X=398:D=26:A=
8098:C=30720:POKE650,1:E=1:F=4:H=C+1:I=
32
 9060 K=203:J=64:P=3:0=9:A2=10:A3=9000:
S=0:C1=0
 9100 POKE249,1:POKE250,0:POKE0,1:POKE1
,0:POKEV,15:POKEY,0:POKEZ,0:POKEX,7:SYS
673
 9110 IFPEEK(K)=JTHENPOKE7694,G+48:GOTO
9110
 9120 IFPEEK(K)=39THENL=3:GOTO8000
 9130 IFPEEK(K)=ITHENG=G+E:IFG=10THENG=
E
 9140 FORT=0TO99:NEXT:POKE7694,G+48:GOT
O9110
 10000 G=1:POKE36866,154:POKE36864,9:PO
KE36867,38:POKE36869,255:POKE36879,8:GO
TO9000
 20000 FORT=0TO13:RETURN
```

Lista simboli grafici

```
    20 : 1  SHIFT HOME =CHR$(147)

    60 : 1  HOME =CHR$(19)

  3010 : 1  HOME =CHR$(19)

  3500 : 1  HOME =CHR$(19)
         1  REVERSE =CHR$(18)
         1  HOME =CHR$(19)
         1  SHIFT REVERSE =CHR$(146)
         1  REVERSE =CHR$(18)

  3510 : 1  HOME =CHR$(19)
         1  REVERSE =CHR$(18)
```

Seguito figura 2.

```
 5000 : 1  HOME =CHR$(19)
        1  REVERSE =CHR$(18)

 9000 : 1  SHIFT HOME =CHR$(147)
        1  CRSR← =CHR$(29)

 9010 : 1  REVERSE =CHR$(18)
        1  SHIFT REVERSE =CHR$(146)
        2  CRSR← =CHR$(29)

 9020 : 1  REVERSE =CHR$(18)
        1  SHIFT REVERSE =CHR$(146)

 9030 : 1  HOME =CHR$(19)
        2  REVERSE =CHR$(18)
        1  SHIFT REVERSE =CHR$(146)
        1  REVERSE =CHR$(18)

 9040 : 1  SHIFT REVERSE =CHR$(146)
        1  REVERSE =CHR$(18)
        1  SHIFT REVERSE =CHR$(146)
        1  REVERSE =CHR$(18)
```

SHARP

Alla ricerca del comando nascosto

di Mauro Lenzi

Molti di voi si saranno chiesti come mai la volta scorsa il LOOP del programma era fatto in modo da esaminare solo i caratteri da 1 a 107; la risposta è semplice: perchè al di sopra di quel numero ci sono cose ancora più strane delle precedenti!
Eccovi il programmino della volta scorsa opportunamente modificato per visualizzare i caratteri al disopra del 107 e contemporaneamente stamparli:

```
10 FOR L = 107 TO 227
20 POKE 45135, L
30 AS = " "
40 PAUSE AS; "  "; L
50 LPRINT AS; "  "; L
60 NEXT L
70 END
```

Inizialmente vedrete comparire una lunga fila di emme (M) sulla stampante e contemporaneamente un misterioso simbolo della corrente alternata (~) sul display, ma non scoraggiatevi, perchè presto inizieranno le sorprese.

A partire infatti da L = 125 vedrete sfilare sotto i vostri occhi tutti i comandi utilizzabili dal vostro Sharp PC-1251!!! La cosa più interessante è che però alcuni di questi non sono mai stati dichiarati dalla casa costruttrice.
Tre di questi ci sono ormai più che noti e sono naturalmente:

PEEK	per L = 175
POKE	per L = 206
CALL	per L = 201

Ma osservando bene, possiamo trovarne molti altri:

COM$	per L = 137
ERROR	per L = 150
KEY	per L = 153
SETCOM	per L = 155
ROM	per L = 158
DEBUG	per L = 181
OUTSTAT	per L = 190
INSTAT	per L = 191
OFF	per L = 205

Trovare il modo di utilizzare queste istruzioni non è impresa facile, è probabile anzi che sia impossibile, cioè che esse vengano riconosciute solo come tali, ma che in realtà non possano venire eseguite.

Tuttavia si possono fare alcune congetture, che potrebbero forse essere molto azzardate, ma in ogni caso vi faranno passare lunghe notti insonni ...

Esaminiamole una per una. ERROR quasi sicuramente non è un'istruzione vera e propria e gestibile da tastiera, semplicemente quando viene inviato alla subroutine di stampa il numero 150 viene automaticamente visualizzata la scritta ERROR. Una cosa curiosa è però che questo pseudocomando, come tutti gli altri citati sopra, venga riconosciuto come tale anche se inserito semplicemente da tastiera. Digitiamo:

10 ERROR

Proviamo a fare girare il programma: comparirà ERROR 1 IN 10.

Se ora andiamo a rivedere la linea di programma, il cursore sarà sulla E di ERROR, ma se lo spostiamo di un solo posto a destra, il cursore si posizionerà dopo tutta la scritta, come accade quando scriviamo delle normali istruzioni.

Anche per quanto riguarda DEBUG e OFF non c'è niente da aggiungere, infatti in qualunque forma abbia provato ad inserirli, il risultato è sempre ERROR 1.
Interessante, ma, almeno per ora, irrisolta, è l'utilizzo dell'istruzione SETCOM. Il nome è assai invitante perchè potrebbe stare per "Set Command", tuttavia anche in questo caso non sono riuscito a trovare la giusta sintassi (se esiste).
Le cose vanno un po' meglio per gli altri comandi: innanzitutto, dopo molte prove, ho trovato una possibile sintassi dell'istruzione KEY; essa dovrebbe funzionare in questo modo:

KEY ON o KEY OFF

Se digitiamo infatti una di queste due istruzioni vedremo che il computer "mediterà" per qualche decimo di secondo, facendo comparire la scritta "busy" ed infine ci darà un ERROR 8, cioè un errore di I/O, come se il computer abbia cercato qualche periferica inesistente.

La sintassi di INSTAT (In Status?) dovrebbe invece essere la seguente:

INSTAT n. oppure solo INSTAT

In entrambi i casi si verifica un ERROR 8.
La corrispettiva OUTSTAT (Out Status?) ha invece una sintassi più chiara:

OUTSTAT n. (con n. compreso o uguale fra 0 e 255)

Ne segue ancora una volta un ERROR 8; è tuttavia interessante notare che se n. è maggiore di 255 o minore di 0 si verifica un ERROR 3, equivalente a "number out of range".
Anche ROM, sebbene possa apparire assurdo, ha una sua sintassi: se digitiamo infatti ROM otterremo il solito ERROR 8. (Forse che il computer sia andato a cercare se c'era una memoria ROM interfacciata?).

Possiamo quindi concludere che queste istruzioni sono state forse immesse in previsione di future interfacce o espansioni di memoria, tuttavia tutto è molto vago e dubbioso e sono convinto che ora che ho dato uno spunto, molti di voi troveranno altre cose curiose e impensabili in proposito.

TEXAS TI 99/4A

Print Using in TI BASIC

di Sergio Borsani

Si è già parlato altre volte dei sostanziali vantaggi offerti dal Modulo SSS TI Extended BASIC. Se, tuttavia, molti comandi aggiuntivi sono insostituibili, soprattutto quelli per la creazione ed il controllo degli sprite, altri sono facilmente simulabili con la versione del BASIC residente. Ricordo che in molte riviste, i primi articoli sul TI 99/4A suggerivano un "trucco" per ottenere le funzioni AND e OR usando rispettivamente i comandi IF (A = B) ★ (A = C) THEN ... e IF (A = B + (A = C) THEN ..., dove naturalmente le espressioni tra parentesi sono scritte a puro titolo di esempio. Come queste, altre istruzioni non disponibili in TI BASIC si possono ottenere ugualmente anche se spesso non in modo così semplice come negli esempi sopra citati ma ricorrendo a delle routine più o meno complesse.
L'istruzione Print Using appartiene a questa categoria. La parola chiave Using viene usata per dare un formato alla stampa, sia su video che su stampante, e risulta particolarmente utile quando si debbano tabulare dei dati numerici. Infatti, se si vogliono incolonnare dei numeri con la funzione TAB, essi verranno giustapposti con il margine sinistro e non con il rispetto della posizione delle unità, delle decine e così via.

Se tutti i numeri sono interi, nella stampa, si dovrà rispettare il margine destro. L'istruzione Print Using assolve questo compito in modo molto semplice. Consideriamo innanzitutto la sua sintassi.

```
PRINT USING "######": N
```

In questo caso il numero individuato dalla variabile N viene stampato in un'area riservata di sei caratteri, con la cifra delle unità nell'ultima posizione a destra (le stringhe alfanumeriche vengono invece stampate a partire da sinistra). Se il numero N è composto da più di sei cifre, l'area di stampa riservata viene riempita con sei asterischi ad indicare che il campo non è sufficientemente ampio per contenere il valore indicato. L'istruzione TAB non è disponibile con la Print Using, pertanto chi volesse usarle entrambe dovrà ricorrere a due istruzioni separate consecutive:

```
PRINT TAB(10);:: PRINT USING "######":
123
```

Per ottenere gli stessi risultati in TI BASIC la cosa è abbastanza agevole. È necessario trasformare la variabile numerica in variabile di stringa con la funzione STR$, calcolarne la lunghezza con la funzione LEN ed infine stampare il numero a partire da una colonna prefissata meno la lunghezza calcolata.

A parole può sembrare un po' complicato ma è necessaria una sola istruzione:

```
PRINT TAB (10-LEN(STR$(N)));N
```

ed il numero verrà scritto con la cifra delle unità sulla colonna 10, indipendentemente dalla sua lunghezza. Considerate il listato 1. Sono stati memorizzati dieci numeri nell'istruzione DATA di linea 140 ed il ciclo FOR NEXT seguente li stampa due volte: a sinistra con la semplice istruzione PRINT TAB(4);N; e a destra con la variante un po' più complicata appena suggerita.

L'istruzione Print Using permette anche di definire un formato di stampa comprendente il punto decimale. Ad esempio, si può scrivere:

```
PRINT USING "#####.####":N
```

Se si incolonnano in tal modo più numeri decimali, non verranno stampati con il margine a destra, ma il riferimento sarà dato proprio dal punto decimale. Se, in questo caso, un numero possiede più di quattro cifre decimali, verranno rappresentate solo le prime quattro, con un arrotondamento dell'ultima.
123.456789 risulterà: 123.4568. Se invece è la parte intera del numero ad eccedere le posizioni specificate compariranno gli asterischi. Disponendo del solo TI BASIC, se si volessero incolonnare numeri decimali usando l'espressione prima suggerita, si otterrebbe la giusta posizione del margine destro ma, al variare delle cifre decimali, varierebbe la posizione del punto decimale. In questo caso, per simulare l'opzione Using, non è più sufficiente una sola istruzione TI

Print Using in TI BASIC

BASIC ma è necessario ricorrere ad una brevissima routine che è riportata nel listato 2. Si deve ricavare la parte intera con la funzione INT e stamparla come si è visto in precedenza; poi, se esiste, si scrive la parte decimale. Inoltre i numeri vanno trasformati in stringhe altrimenti tra la parte intera e quella decimale verranno lasciati degli spazi vuoti.

```
100 REM       SIMULAZIONE
110 REM       PRINT USING
120 REM    ******************
130 CALL CLEAR
140 DATA 450,1620,135,24,142,463,2670,85
0,125,12
150 PRINT "  SCRITTURA     SIMULAZIONE"
160 PRINT "   NORMALE      PRINT USING"
170 PRINT
180 CALL HCHAR(23,3,95,28)
190 FOR J=1 TO 10
200 READ N
210 PRINT TAB(4);N;
220 PRINT TAB(22-LEN(STR$(N)));N
230 NEXT J
240 PRINT :::
250 PRINT TAB(7);"<BATTI UN TASTO>"
260 CALL KEY(0,K,S)
270 IF S=0 THEN 260
280 CALL CLEAR
290 END
```

Listato 1. *L'istruzione 220 del listato simula il comando Print Using permettendo di allineare i numeri con il margine destro indipendentemente da quante sono le cifre che li compongono.*

```
100 REM       SIMULAZIONE
110 REM       PRINT USING
120 REM    ******************
130 CALL CLEAR
140 DATA 42.6,162.24,1350,24.145,1.42
150 DATA 2380.2,45.628,365,42.64,2.58
160 PRINT "  SCRITTURA     SIMULAZIONE"
170 PRINT "   NORMALE      PRINT USING"
180 PRINT
190 CALL HCHAR(23,3,95,28)
200 FOR J=1 TO 10
210 READ N
220 PRINT TAB(4);N;
230 IN=INT(N)
240 PRINT TAB(22-LEN(STR$(IN)));STR$(IN);
250 IF N=IN THEN 270
260 PRINT STR$(N-IN);
270 PRINT
280 NEXT J
290 PRINT :::
300 PRINT TAB(7);"<BATTI UN TASTO>"
310 CALL KEY(0,K,S)
320 IF S=0 THEN 310
330 CALL CLEAR
340 END
```

Listato 2. *Se i numeri contengono una parte decimale, l'istruzione precedente non garantisce più il giusto incolonnamento.*
Si rende allora necessaria la breve routine presente nelle linee 230-270. In questo modo si simula l'istruzione PRINT USING "###.##":N e si ottiene l'incolonnamento con riferimento al punto decimale.

In tutto sono necessarie cinque linee di programma. Ricordo al termine che l'istruzione Print Using si può usare anche con la stampante nel formato:

PRINT #4,USING "####.##":N

ma di questo parlerò più diffusamente in occasione della presentazione di un programma.

SINCLAIR ZX SPECTRUM

Eliminare blocchi di programma senza cancellare le linee singolarmente

di Marcello Spero

L'eliminazione di un certo numero di linee da un programma è sempre un'operazione noiosa. Quando poi le linee da cancellare sono molte la cosa può diventare veramente intollerabile.
Le due routine che vediamo questa volta fanno proprio questa operazione, riducendo il nostro compito all'indicazione della prima e dell'ultima linea da eliminare. La prima è tutta in BASIC, quindi di funzionamento facilmente comprensibile ma un po' più lenta della seconda, che invece sfrutta il linguaggio macchina, ed in particolare una routine ROM di cui è bene conoscere l'esistenza, potendo essere utile in molte situazioni.

Procedendo con ordine, iniziamo con l'esaminare il funzionamento della prima routine, che trovate in figura 1.
Partendo dall'indirizzo di inizio dell'area di programma (variabile di sistema PROG all'indirizzo 23635) viene cercata una linea con il primo numero da noi specificato, e il suo indirizzo viene memorizzato nella variabile start. Quindi viene cercata la linea indicata dal secondo numero, l'ultima da cancellare, e il suo indirizzo viene memorizzato nella variabile stop.

A questo punto il programma calcola il numero di byte da eliminare, ne sottrae 4 e lo pone come lunghezza della prima linea da cancellare. In questa situazione cancellare la prima linea vorrà dire cancellare tutte le linee di programma incluse nella sua lunghezza fittizia. La sottrazione dei 4 byte è necessaria poichè la lunghezza di una linea non deve comprendere i primi byte che ne indicano il numero e la lunghezza stessa, che sono appunto 4.
Non resta ora che cancellare, come ci indica il programma, la prima linea, e il gioco è fatto. Un unico avvertimento riguarda i numeri di linea che forniamo al programma: questi devono sempre corrispondere a linee realmente esistenti, pena il non funzionamento della routine.

Passiamo ora alla seconda routine, più rapida e flessibile, che si compone di un programma BASIC di inizializzazione, visibile in figura 2, e di un programmino in linguaggio macchina che svolge materialmente tutti i calcoli e le ricerche.

Come nel caso precedente vengono richiesti i numeri corrispondenti alla prima e ultima linea da eliminare che, dopo alcuni controlli preliminari volti a proteggere il codice macchina da dati errati che porterebbero a risultati disastrosi, sono provvisoriamente immagazzinati nel buffer di tastiera per essere utilizzati dal programma in linguaggio macchina. Viene usato il buffer di tastiera per rendere i dati indipendenti nella loro collocazione dalla posizione in memoria della routine l/m, che in questo modo diviene completamente rilocabile. L'ultima linea del programma provvede quindi a far partire il codice macchina. L'indirizzo della USR sarà quello a cui avrete posto la routine l/m. Colgo l'occasione per una precisazione riguardante l'uso di RANDOMIZE nel lancio dei programmi in linguaggio macchina: potrebbero essere usate allo stesso modo anche PRINT o LET a=, per esempio, ma sarebbero più lente nell'essere eseguite, ed inoltre avrebbero effetti collaterali quali la stampa di un numero o l'impegno di una variabile. Con RANDOMIZE, invece, l'unico effetto è una variazione del contenuto della variabile di sistema SEED, variazione peraltro senza conseguenze. Chiusa la parentesi, torniamo al nostro programma. Sulla parte BASIC si può ancora dire che è stata numerata in modo da poter essere collocata in coda ad un programma ma qualsiasi posizione può andare bene, purchè stiate attenti a non includerla nella cancellazione ...

Passando ad esaminare il linguaggio macchina, vediamo come le operazioni svolte siano sostanzialmente le stesse che venivano eseguite dalla routine precedente. Vengono trovati infatti gli indirizzi della prima e ultima linea, e quindi con una sottrazione si ottiene la lunghezza totale del blocco. Questa viene infine posta come lunghezza della prima linea, che trascinerà in tal modo le altre nella propria cancellazione. La grossa differenza rispetto alla routine precedente sta nel fatto che qui gli indirizzi della linea iniziale e di quella finale non vengono cercati passando in rassegna l'intero programma, dall'inizio fino alla linea voluta, ma in modo diretto. Questo è possibile facendo uso della routine LINE-ADDR, che si trova all'indirizzo esadecimale 196E, in ROM. Se noi chiamiamo LINE-ADDR passandole nel registro HL dello Z80 un numero di linea, questa ci restituirà l'indirizzo di inizio della linea precedente nel registro DE (parliamo sempre di registri dello Z80) e l'indirizzo di inizio della linea stessa in HL. Inoltre, se il numero di linea che le abbiamo passato non corrisponde ad una linea BASIC esistente, in HL ci sarà l'indirizzo di inizio della prima linea esistente il cui numero sia maggiore di quello richiesto.

L'operazione sarà comunque segnalata dal flag di zero dello Z80, che in questo caso sarà posto ad 1, essendo altrimenti a zero.
Pur limitandosi le nostre esigenze al contenuto di

PERSONAL SOFTWARE

Eliminare blocchi di programma

HL, cioè l'indirizzo della linea, tutta questa versatilità ci porta dei vantaggi. Infatti questa routine funziona anche se i numeri di linea che le passiamo non corrispondono a linee BASIC.
Potremo avere due casi: non esiste il numero della prima linea, o non esiste il numero dell'ultima. Nella prima ipotesi l'unica conseguenza sarà il dover cancellare, al termine del programma, non la linea il cui numero avevamo indicato, che non esiste, ma la prima linea successiva. Nella seconda ipotesi, invece, qualche danno può essere prodotto: infatti la cancellazione proseguirà fino alla prima linea realmente esistente di numero successivo a quello che abbiamo fornito, e la cui cancellazione poteva essere o non essere voluta.

La figura 3 riporta il disassemblato del 1/m con qualche notizia più in dettaglio, mentre in figura 4 trovate il codice pronto per l'uso.
Come abbiamo detto è completamente rilocabile, quindi a voi la scelta della posizione in memoria; consiglio comunque di proteggerlo ponendolo oltre la RAMTOP.


```
9000 REM ************************
         *                      *
         *     BLOCK DELETE     *
         *                      *
         ************************

9010 LET x=PEEK 23635+256*PEEK 2
3636
9020 INPUT "prima linea da elimi
nare ";a
9030 INPUT "ultima linea da elim
inare ";b
9040 LET yt=PEEK (x+2)+256*PEEK
(x+3)
9050 IF PEEK x*256+PEEK (x+1)=a
THEN LET start=x
9060 IF PEEK x*256+PEEK (x+1)=b
THEN LET stop=x+yt+4: GO TO 9080
9070 LET x=x+yt+4: GO TO 9040
9080 LET l=stop-start-4
9090 POKE start+2,l-256*INT (l/2
56)
9100 POKE start+3,INT (l/256)
9110 CLS : PRINT AT 10,0;"digita
 ";a;" seguito da ENTER"
```

Figura 1. *Il programma Block Delete.*

```
9000 REM ************************
         *                      *
         *   BLOCK DELETE L/M   *
         *                      *
         ************************

9010 INPUT "prima linea da elimi
nare ";l1
9020 INPUT "ultima linea da elim
inare ";l2
9030 IF l1>l2 OR l1<1 OR l2>9999
 THEN GO TO 9010
9040 POKE 23296,l1-256*INT (l1/2
56)
9060 POKE 23297,INT (l1/256)
9070 POKE 23298,l2-256*INT (l2/2
56)
9080 POKE 23299,INT (l2/256)
9090 RANDOMIZE USR 65001: REM in
dirizzo di inizio della routine
in l/m
9100 PRINT AT 10,0;"digita ";l1;
" seguito da ENTER"
```

Figura 2. *La routine BASIC del programma Block Delete 1/m.*

```
                        trova l'indiriz-
ld hl,(23296)           :zo del del primo
call 6510               :byte della lun-
inc hl                  :ghezza della pri
inc hl                  :ma linea e salva
ld hl,(22298)           :lo stesso per l'
call 6510               :ultima riga, ma
inc hl                  :senza salvarlo.
inc hl                  :
ld e,(hl)               :trova la lunghez
inc hl                  :za dell'ultima
ld d,(hl)               :linea, e quindi
inc hl                  :l'indir. del pri
add hl,de               :mo byte success.
pop de                  :
push de                 :recupera e salva
and a                   :il primo indir.
sbc hl,de               :calcola il n. di
dec hl                  :byte da elimin.
dec hl                  :
ex de,hl                :inserisci il n.
pop hl                  :byte da elimin.
ld(hl),e                :nella lunghezza
inc hl                  :della prima
ld(hl),d                :linea da elimina
ret                     :re.
```

Figura 3. *Disassemblato della routine 1/m.*

```
 42,  0, 91,205,110, 25, 35, 35,
229, 42,  2, 91,205,110, 25, 35,
 35, 94, 35, 86, 35, 25,209,213,
167,237, 82, 43, 43,235,225,115,
 35,114,201
```

Figura 4. *Codice macchina della routine 1/m.*

COMMODORE C 64

Gli "orologi" nel C 64

di Alessandro Guida

C 64 Time of the Day/TOD

Anche se nel manuale d'uso del C 64 non ne viene fatta parola, all'interno del computer sono disponibili ben due orologi con allarme. Questi sono fisicamente alloggiati nei due CIA (Complex Interface Adapter) che gestiscono le operazioni di input/output del 64.
Non bisogna, però, confondere questo orologio con la variazione TI$. Infatti, TI$ (che pure esiste sul 64), è semplicemente un contatore incrementato via software ad ogni interrupt. Per questo motivo è anche poco preciso, infatti, la maggior parte delle routine di I/O rallentano o bloccano del tutto la frequenza degli interrupt causando ritardi nel contatore TI$.
Il TOD, invece, è incrementato dalla frequenza di rete, quindi molto più affidabile.
In figura 1, sono riportati i registri del CIA. Poichè di questi integrati nel 64 ce ne sono due, i rispettivi registri saranno collocati in zone diverse della memoria.
In particolare i registri del primo CIA partono dalla locazione $DC00 (56320) mentre quelli del secondo da $DD00 (56756).

Poichè il funzionamento dei due TOD è assolutamente identico, di seguito non ci riferiremo in particolare agli indirizzi di un CIA o dell'altro ma al numero del registro.
In pratica, poi, il numero del registro andrà sommato alle locazioni di partenza del CIA che si desidera utilizzare.

I registri

L'ora conservata in un CIA è organizzata alla maniera anglosassone. Ci sono, cioè, 12 ore divise in AM e PM. È presente anche una cifra per i decimi di secondo.
L'orario è conservato nei primi quattro registri di figura 1.
È interessante notare che pur essendo questi dei normali registri a 16 bit, contengono i dati in forma BCD (Binary Coded Decimal). In questa maniera è estremamente facile la conversione dei numeri in carattere ASCII.
Un esempio di numerazione binaria secondo il codice RCD è riportato in figura 2.
Nel registro 8 vi sono i decimi di secondo, che occupano solo i primi 4 bit (una cifra da 0 a 9).

		Contenuto bit registri							
Reg	Nome	7	6	5	4	3	2	1	0
8	Decimi secondo	0	0	0	0	D8	D4	D2	D1
9	Secondi	0	S4	S2	S1	s8	s4	s2	s1
A	Minuti	0	M4	M2	M1	m8	m4	m2	m1
B	Ore	0=AM 1=PM	0	0	D1	o8	o4	o2	o1
D	Interrupt (Allarme)	X	X	X	X	X	1=ON 0=OF	X	X
E	Frequenza rete (Hz)	0=60 1=50	X	X	X	X	X	X	X
F	Set orologio/ allarme	0=Or 1=Al	X	X	X	X	X	X	X

Figura 1. *Tabella dei registri del CIA (6526) che riguardano il TOD (orologio). Le X indicano un contenuto senza importanza.*

Nel 9 ci sono i secondi e nel registro $A (10) vi sono i minuti, ambedue organizzati alla stessa maniera. I primi 4 bit sono le unità, gli altri quattro le decine. Naturalmente dei quattro bit più significativi ne sono utilizzati solo tre, giacchè devono contenere al massimo il numero 6.

Infine nel registro B vi è l'ora. I primi 4 bit sono le unità. Il quinto bit è la decina (ricordate che al massimo si arriva a 12), mentre l'ultimo bit contiene 0 se si è di mattina (AM) e 1 se di pomeriggio (PM).
Queste locazioni sono le stesse sia per l'orario corrente sia per l'ora di allarme. Per indicare al CIA se stiamo aggiornando uno o l'altro è necessario mettere a 0 (per l'ora) o a 1 (per l'allarme) il bit 7 del registro F.
Va, anche detto, che l'allarme è possibile solo scriverlo ma non leggerlo. Per cui, anche se il bit 7 è a 1, se si va a leggere nei registri da 8 a B si avrà sempre l'ora del giorno.
Inizialmente aggiorneremo l'ora per cui lo dovremo azzerare.

Per il primo CIA, ad esempio, si avrebbe: 56320 + 15 = 56335, quindi:

```
POKE 56335, PEEK (56335) AND 127
```

Prima di proseguire bisogna settare anche il bit 7 del registro E, che specifica la frequenza di rete. Uno 0, in questo bit, indica una frequenza di 60 Hz, mentre l'1 prepara il CIA a ricevere un segnale a 50 Hz.

Fate attenzione perchè all'accensione del computer questo bit è a zero (60 Hz) mentre in Italia la frequenza di rete è di 50 Hz. Si darà, quindi, anche il seguente comando:

```
POKE 56334, PEEK(56334) OR 128
```

PERSONAL SOFTWARE

Gli "orologi" nel C 64

Per aggiornare i registri contenenti l'ora è necessario seguire un certo ordine di operazioni. Va aggiornato per primo il registro delle ore, poi quello dei minuti e quello dei secondi e, per ultimo, il registro dei decimi di secondo. Questo perchè il CIA nel momento in cui si scrive nel registro dell'ora blocca il conteggio in maniera che alcuni registri non vengano modificati accidentalmente dall'avanzare degli altri. L'orologio riparte, automaticamente, dopo la scrittura dell'ultimo registro (decimi sec.).

La stessa cosa accade quando si va a leggere la locazione dell'ora. Naturalmente in questo caso non viene fermato il conteggio ma solo "congelato" il contenuto dei registri mentre l'orologio interno continua ad avanzare.
I registri tornano ad essere aggiornati, dopo la lettura dei decimi di secondo.

L'allarme

Come abbiamo già anticipato è anche possibile fissare l'orario di allarme. L'allarme va introdotto nella stessa maniera vista per l'ora corrente, naturalmente, dopo aver settato il bit 7 del registro F. Questo va riportato a zero quando si sarà terminata l'operazione.
Fissata l'ora di allarme, il CIA la confronterà continuamente con l'ora del giorno e quando saranno uguali attiverà un'interrupt. Quindi, per poter utilizzare l'allarme è necessario modificare la routine di interrupt originale.

Poichè questo viene attivato, normalmente, 60 volte al secondo e, inoltre, ci possono essere anche altre sorgenti di interruzione, è importante che la nuova routine controlli la provenienza della richiesta di interrupt. Se a causare l'interruzione sarà stato l'allarme proseguirà con le opportune operazioni altrimenti tornerà alla routine di gestione interrupt normale.
Se l'interruzione è stata causata dall'allarme è facile da verificarsi perchè nel registro D il bit 2 sarà settato. Tale registro contiene tutte le possibili fonti di interrupt, interne al CIA, ma noi siamo interessati solo al bit che contiene lo stato dell'allarme.
Il registro D è, in realtà, costituito da due registri: uno a sola lettura e l'altro a sola scrittura. Quello di cui si è parlato prima era quello di lettura. L'altro, invece, serve ad abilitare o disabilitare le varie sorgenti di interrupt.

Di conseguenza, per attivare l'allarme, si dovrà digitare:

POKE 56333,132

mentre per disabilitarlo:

POKE 56333,4

```
10 A=0:FORI=49152TO49327:READJ:POKEI,J:A=A+J:NEXTI
20 IFA<>22193THENPRINT"ERRORE NEI DATA":END
30 PRINT"■■OROLOGIO PER C64■"
40 SYS49152
100 DATA234,234,234,120,173,020,003,141,028,192
105 DATA169,130,141,020,003,173,021,003,141,029
110 DATA192,169,192,141,021,003,088,096,049,234
115 DATA173,024,208,041,240,074,074,133,254,169
120 DATA000,133,253,160,000,173,011,220,072,041
125 DATA127,162,186,032,105,192,173,010,220,032
130 DATA105,192,173,009,220,162,174,032,105,192
135 DATA173,008,220,032,122,192,104,016,003,169
140 DATA144,044,169,129,032,126,192,169,141,145
145 DATA253,169,216,133,254,169,001,145,253,136
150 DATA016,251,108,028,192,072,032,118,192,104
155 DATA032,122,192,138,032,126,192,096,074,074
160 DATA074,074,041,015,009,176,145,253,200,096
165 DATA166,215,224,133,208,007,169,030,141,173
170 DATA192,208,025,224,134,208,007,169,102,141
175 DATA173,192,208,014,224,135,208,014,174,096
180 DATA192,232,138,041,015,141,096,192,169,000
185 DATA133,215,076,30,192,234
200 REM AGGIORNAMENTO ORA
210 PRINT"■■AGGIORNAMENTO ORA■":PRINT
215 POKE56334,PEEK(56334)OR128:REM SELEZIONA CLOCK A 50HZ
220 POKE56335,PEEK(56335)AND127:REM SELEZIONA ORA
230 INPUT"AM O PM";A$
240 A=128:IFASC(A$)=65THENA=0
250 INPUT"ORA    ";A$:IFLEN(A$)>2THENPRINT"□";:GOTO250
260 GOSUB500:IFN>18THENPRINT"□";:GOTO250
270 POKE56331,A+N:REM AGGIORNA ORA
280 INPUT"MINUTI ";A$:IFLEN(A$)>2THENPRINT"□";:GOTO280
290 GOSUB500:IFN>89THENPRINT"□";:GOTO280
300 POKE56330,N:REM AGGIORNA MINUTI
310 INPUT"SECONDI";A$:IFLEN(A$)>2THENPRINT"□";:GOTO310
320 GOSUB 500:IFN>89THENPRINT"□";:GOTO310
330 POKE56329,N:REM AGGIORNA SECONDI
340 PRINT"■PER FAR PARTIRE L'OROLOGIO":PRINT"PREMI UN TASTO"
350 GETA$:IFA$=""THEN350
360 POKE56328,0:REM AGGIORNA DECIMI SEC. E START
380 END
500 IFLEN(A$)=1THENT=0:GOTO520
510 T=VAL(LEFT$(A$,1))
520 U=VAL(RIGHT$(A$,1))
530 N=16*T+U:RETURN
```

Listato 1. *Programma per avere sullo schermo l'orologio interno del C 64.*

Gli "orologi" nel C 64

Decimale	BCD	Decimale	BCD
1	00000001	10	00010000
2	00000010	20	00100000
3	00000011	30	00110000
4	00000100	40	01000000
5	00000101	50	01010000
6	00000110	60	01100000
7	00000111	70	01110000
8	00001000	80	10000000
9	00001001	90	10010000
		99	10011001

Figura 2. *Esempio di numerazione binaria secondo il codice BCD.*

Tornando alla routine di gestione interrupt, in un esempio, quanto detto può essere rappresentato così:

```
C000                    NUOVA ROUTINE IRQ
C000
C000  LDA $DC0D         legge interrupt register
C003  AND &$04          seleziona bit Allarme
C005  BNE $C00A         se è settato salta alla
C007                    routine ALLARME
C007  JMP $EA31         salta routine IRQ originale
C00A
C00A                    ROUTINE ALLARME
```

Il programma

Il listato 1 è una applicazione tipica. Il programma si divide in due parti. La parte in linguaggio macchina si occupa di visualizzare l'ora, sullo schermo in alto a sinistra, ad ogni ciclo di interrupt. La parte in BASIC serve, invece, ad aggiornare l'orario all'inizio. Una volta fatto girare il programma BASIC si può cancellare con un NEW, l'orologio resta comunque in memoria.

Il tasto F1 abilita l'apparizione dell'ora sullo schermo, mentre F3 la disabilita. Con F5 è possibile cambiare il colore dell'orologio.

Se durante l'uso si rendono necessari i tasti funzione per altri usi, il tutto può essere disattivato premendo contemporaneamente i tasti STOP e RESTORE. Per far riapparire l'orologio basterà dare il comando SYS 49152.

■

PICCOLI ANNUNCI

Apple

Cambio software per Apple anche con sistema operativo CPM. Tommaso Tanto - Via Del Sole, 18 - 92019 Sciacca (AG).

Cambio-vendo programmi per Apple. Possiedo vasta biblioteca di software. Rispondo a chiunque invia lista o richieste. Telefonare ore serali. Franco Vittor - Via Grabizio, 35 - 34170 Gorizia - Tel. 0481/81254.

Per Apple 2 + cambio programmi su dischette o listati. Anche C64. Dispongo copie videogiochi 60 BBL ER (Packman) defender, planetoids, space eggs, sabotage. Tasc, o *vendo-cerco* copie programmi - copia. Stefano Malagodi - Comunale, 14 - 44034 Copparo (FE) - Tel. 0532/860196.

Vendo Apple writer IIe con manuale + manuale WPL a L. 100.000 + spese contrassegno. Gianluca Pomponi - Via Raffaello, 5 - 56020 Castel Del Bosco (PI) - Tel. 0571/467053.

Vendo-cambio software per Apple II/IIe con manuali (Apple Writer IIe, visicalc, giochi, etc.). Gianluca Pomponi - Via Raffaello, 5 - 56020 Castel Del Bosco (PI) - Tel. 0571/467053.

Esperto programmatore C.N.R *cambio* programmi di ogni tipo per Apple, sono particolarmente interessato ai package grafici. Scrivere o telefonare a: Alberto Pollastri - Via Di Pratale, 28/B - 56100 Pisa - Tel. 050-20584.

Vendo listati giochi Apple II o cambio con programmi su cassetta o listati per il VIC-20. Giovanni Bellamacina - Via Mazzini, 31 - 20058 Villasanta (MI) - Tel. 039/302576.

Cambio-vendo programmi per Apple II. Dispongo di oltre 500 programmi, tra cui gestionali ingegneria, utilità, giochi. Claudio Citarella - Via Parroco Federico, 41 - 80045 Pompei (NA) - Tel. 081/8632946.

Commodore

Cerco per Commodore 64, software di qualsiasi tipo. Inoltre vorrei contattare utenti CBM64 per Commodore. Veneto Club. Paolo Bigon - Via S. Marco, 177/A - 35020 Padova - Tel. 049/629117.

Costituito il primo Club Nazionale per utenti di Commodore 64 biblioteca di oltre 100 programmi, per informazioni telefonare allo 095/933369 chiedere di Carmelo o allo 0522/30155 chiedere di Lucio. Commodore 64 Exchange Club - C.so Italia, 60 - 95014 Giarre (CT).

Cerco listati programmi per CBM 64, tutte le possibili informazioni sul software e hardware del C 64, possibilmente in italiano, pago bene, *vendo* alla scoperta dello ZX Spectrum L. 18.000 nuovo. Nicola Franchetti - P.zza Libertà, 5 - 35036 Montegrotto T. (PD) - Tel. 794783.

Si *vendono* o si cambiano programmi per Commodore serie 3000-4000 per ricevere ampio elenco inviare L. 1000. Si producono anche programmi su commissione. Scrivere o telefonare ore pasti. Aroldo Bizzarri - Via Pantelleria, 19 - 91100 Trapani - Tel. 0923/20044.

Affaroni, *vendo* computer VIC 20 e Commodore 64 nuovi ancora imballati, rispettivamente a L. 280.000 e L. 550.000, sono comprensivi di registratore, 1 videogioco, manuale. A richiesta anche gli optional. Andrea Di Rienzo - Via Longarone, 10 - 20157 Milano - Tel. 02/3575415.

Vendo software Commodore 64, programmi scientifici, business, didattici, giochi vari, dettagliata documentazione a richiesta. Invio catalogo gratuitamente. Rispondo a tutti. Luciano Antoniucci - Via Goldoni, 7 - 05100 Terni - Tel. 0744/421274.

Cerco possessori Commodore 64 possibilmente zona Monfalcone, Gorizia, per scambio software hardware idee desidero sapere se nella provincia di Gorizia ci sono Club di utenti di Commodore 64. Luca Laconi - Via Don P. Fanin, 26 - 34074 Monfalcone (GO) - Tel. 0481/40958.

Vendo VIC 20 + interf. reg., sup-exp, stamp. VC1515, cart. sargonchess, manuali orig. ingl. e ital. Guida EVM, cassetta giochi, graf. funzioni, con stampa, dis. M. ris. al miglior offerente in blocco o separatamente. Paolo Berbellini - Via Roma, 18 - 63023 Fermo (AP) - Tel. 0734/32136.

Per VIC 20 e 64 *vendo* programmi solo in linguaggio macchina e alcuni super in Basic. Prezzi amichevolissimi. Enzo Piperno - Via Torre Dello Stinco, 45 - 00132 Roma - Tel. 06/6160934.

CBM 64. Sono un analista di una banca di interesse nazionale e mi occupo di titoli-borsa. Ho realizzato un gioco sulla borsa. Possono partecipare 2-8 giocatori. *Vendo-scambio*. Gianni Castellan - Via Cantelli, 5 - 43100 Parma - Tel. 32801.

Cambio/vendo oltre 350 programmi per il Commodore 64 tutti in linguaggio macchina. Maurizio Carola - Via L. Lilio, 109 - 00143 Roma - Tel. 06/5917363.

Compro-cambio-vendo programmi per Commodore 64 di qualsiasi tipo. All'invio della lista (indicare la trattativa che vi è più congeniale); risponderò con la mia. Franco Bruno - Via Giorgio Bratti, 100 - 47023 Cesena (FO) - Tel. 0547/23810.

Commodore 64 programmi di vario genere *vendo* a L. 3000 (sconto sui grandi quantitativi) accetto scambio lista su richiesta. Ferdinando Errichiello - Via Bracco, 59 - 00137 Roma - Tel. 06/8272747 ore pasti.

Per CBM 64 *vendo* "Frogger-64" (su cassetta) "Grandmaster" (scacchi 9 livelli di difficoltà su disco) "Screen-Graphics-64" (24 comandi in più per gestire grafica ad alta risoluzione su disco). Stefano Vandelli - Via Parini, 5 - 41013 Castelfranco E. (MO) - Tel. 059/924259.

Vendo fotocopie perfette di listati di programmi per il Commodore 64. Inviare L. 1000 per il catalogo a: Lauro Michelotti - Via Boboli, 1/A - 51017 Pescia (PT).

Vendo-compro-cambio programmi per Commodore 64 di ogni genere inviare e richiedere lista. Claudio Baraldi - Via San Marone, 9/2 - 41100 Modena - Tel. 059/358649.

Vendo per CBM 64: Simon Basic, RTTY, Hesmon, Pacman, Scacchi, CrazyKong, e molti altri giochi e utility. Per CBM serie 3000 vendo TOOLKIT (ROM da 2 Kbyte e aggiunge 10 istruzioni al Basic). Massimo Cantelli - Via Corso, 40 - 40051 Altedo (BO) - Tel. 051/871270.

Vendo VIC 20 nella sua confezione originaria completo + il manuale d'uso in italiano + registratore a nastro C2N VC1530 in più libro su VIC, il tutto ancora in garanzia usato pochissimo a L. 400.000. Telefonare dalle ore 14,30 alle 15,30 e chiedere di Giancarlo oppure scrivere. Giancarlo Bighi - Via Mortara, 86 - 44100 Ferrara - Tel. 0532/29759.

Vendo stampante Commodore 4022 per PET/CBM4032/8032 o VIC 20 o 64, perfetta, 1 mese di vita, prezzo davvero interessante. Telefonare ore serali. Roberto Gens - Via Ponte Romano, 90 - 11027 St. Vincent (AO) - Tel. 0166/37782.

Vendo-compro-cambio software per Commodore 64 cerco inoltre utilizzatori di questo computer zona Parma e Provincia per scambio idee e programmi telefonare ore 19 - 19,30. Fabrizio Parenti - Via Prampolini, 5 - 43100 Parma - Tel. 0521/72924.

CBM 64 *vendo* programmi di utility giochi libri desidero inolte entrare in contatto con conoscitori linguaggio macchina per scambio informazioni sul sistema operativo. Marcello Cesi - Via Magliana Nuova, 178 sc. B/22 - 00146 Roma - Tel. 06/5266009.

Vendo Cabinet espansione + registratore C2N per VIC 20 come nuovi L. 150.000 non trattabili telefonare ore pasti. Luigi Bertinelli - Via Pallera, 4 - 10027 Moncalieri (TO) - tel. 011/6470696.

Vendo Commodore 64 + registratore C2N + una cartuccia con gioco + la guida di riferimento per il programmatore + un altro libro + tanti programmi su cassetta a L. 500.000. Augusto Amato - Via Rodolfo Morandi, 3 - 00139 Roma - Tel. 8185465.

Cambio-vendo moltissimi giochi per VIC 20 (anche in linguaggio macchina). A prezzi clamorosi !!!! Amedeo De Franco - Via Mazzini, 277 - 55049 Viareggio (LU) - Tel. 0584/32134.

Per far funzionare il vostro nuovo VIC 20 avete bisogno di programmi. Ne ho di eccezionali. Scrivetemi e vi manderò l'elenco. Andrea Bartolini - V.le A. Volta. 106 - 50131 Firenze - Tel. 575224.

Cerchiamo possessori Commodore 64 per organizzare club zona Napoli. Inoltre vendiamo, compriamo, scambiamo software su cassetta o disco. Inviare o richiedere lista dettagliata. Corrispondere con: Gennaro Di Giaimo - P.zza Tanucci, 2 - 80046 San Giorgio a Cremano (NA) - Tel. 081/489639.

Desidero contattare possessori Commodore 64 per *scambio* programmi e notizie utili. Esiste già qualche "Commodore 64 Club?". Elio Antonucci - Via Faenza, 11 - 40139 Bologna.

Vendo-cambio programmi per VIC 20 su cassetta richiedere elenco inviando L. 400 in francobolli o propria lista. Renzo Radollovich - Via Marco Polo - 34074 Monfalcone (GO) - Tel. 0481/42192.

Eccezionale! *Vendo* programmi VIC 20, software dilettanti, commerciale, scientifico, tra gli altri: legge di Ohm, anagrammi, test ed altri per avere gratis la lista, scrivere o telefonare a: Vincenzo Musicò - Via Paolo Blandino, 12 - 98100 Messina - Tel. 090/2938626.

PICCOLI ANNUNCI

Cerco qualcuno che mi regali programmi per VIC 20. Spedire in abbondanza a Barozzi Massimo - Via Bettini, 50 - Rovereto (TN) - Tel. 35215, Grazie.

Vendo-cambio software per Commodore 64 - PET 2001 - 3032 e 8032 - utility e anche giochi spedisco lista a chi invia busta affrancata oppure telefonare nelle ore serali dopo le 21. Francesco Venturelli - Via Repubblica, 193 - 41059 Zocca (MO) - Tel. 059/987909.

Scambio-vendo programmi VIC 20 in linguaggio macchina ed cartridge su nastro. Vendo Sargon chess ed Road Race a L. 20000 l'uno. Cerco inoltre sistemi per duplicare i prog. protetti. Scrivere a Luca Mansutti - Via M. Grappa, 1 - 33100 Udine.

Invio cassetta con più di 20 giochi per VIC 20 o 64, anche in L.M., a coloro che mi inviano la loro, con 10 programmi non copiati da riviste! Annuncio sempre valido. Assicuro serietà e risposta! Carlo Dalle Luche - Via Rovigo, 18/1 - 39100 Bolzano.

Vendo calc-result advanced (in italiano) originale. Potente Spread-Sheet (tipo Sicalc). Lire 200.000 (prezzo norm. L. 400.000) per Commodore 64 all'acquirente regalo Simons Basic. Alessandro Fogar - Via Venezia, 26 - 34073 Grado - Tel. 0481/768655.

Vendo VIC 20 + 3K super Expander + 8K + C2N datasette unit + manager e +joystick Commodore + 6 programmi in LM (fra i quali Skyhawk e Shadofax) + alla scoperta del VIC 20 al miglior offerente. Telefonare ore pasti. Alberto Gaffuri - V.le A. Doria, 48/A - 20124 Milano - Tel. 02/6704057.

Vendo per VIC 20 programmi di utilità e video games, 30 programmi su cassetta in blocco L. 29.000. Armando Mazza - Via Settembrini, 96 - 70053 Canosa (BA) - Tel. 0883/64050.

Vendo VIC 20 come nuovo con espansione 8K motherboard due cartucce e vastissimo software tra cui Jelly Monsters istruzioni in italiano a L. 450.000 trattabili (valore 650.000). Alberto Scotti - Via Frassinetti, 7 - 20148 Milano - Tel. 02/4033929.

Cambio software per VIC 64 preferibilmente disco. Inviare listati o telefonare. Fernando Forner - Via Valperga Caluso, 21 - 10225 Torino - Tel. 011/6566538.

Vendo per CBM 64 Simon' BASIC su nastro aggiunge 114 comandi al tuo computer completo di istruzioni in italiano e di un programma dimostrativo. Telefonare ore pasti (chiedere di Carlo). Carlo Ferrari - Via G. Martinelli - 35044 Montagnana (PD) - Tel. 0429/82469.

Vendo per PET CBM oltre 600 programmi, giochi, utilità, IVA, fatture, mailinglist, tutti su nastro chiedere elenco inviando L. 1000 per la risposta. Sandro Biondi - Via Canova, 14/1 - 07026 Olbia - Tel. 0789/50938.

Vendo-scambio programmi per Commodore 64, Basic 4.0, Pisk copy e giochi vari. Scrivere o telefonare a: Max Maneschi - Via G. Bonanno, 67 - 90143 Palermo - Tel. 091/264184.

Compro per Commodore 64 "Guida al C 64" in italiano, programmi, giochi (cassette), listati. Spedire elenco con prezzo e specifica. Giancarlo Testi - Via F.P. Da Cherso, 4 - 00143 Roma.

Vendo VIC 20 + registratore + espansione 8K + 30 giochi su cassetta + 2 cartridge + guida al VIC 20 + cavetti e manuale. Tutto funzionante (4 mesi di vita) a L. 500.000 trattabili. Walter Bianchini - Via Dei Ciclamini, 11 - 20147 Milano - Tel. 4151007.

64isti *scambiano* programmi di ogni genere su cassetta o listato. Disponibili interessanti games. Inviare lista programmi a: Walter Giacosa - Via Barletta, 117 - 10136 Torino - Tel. 011/324978.

Cerco software per il VIC 20 sia in cassette che in listati. Spedirò altrettanti programmi a tutti coloro che me li invieranno. Gian Piero Gavi - Via Terre Bianche, 8 - 18100 Imperia - Tel. 0183/63981.

Vendo-scambio programmi per Commodore 64 molti in L.M. originali americani chiedere lista allegando L. 300 in francobolli o telefonare alla sera. Nicola Cattafesta - Via Roselli, 18 - 46047 Porto Mantovano (MN) - Tel. 0376/398072.

Commodore 64: *Cerco* software di termotecnica, impianti riscaldamento/condizionamento, ventilazione, energia solare, reti idriche. Cerco inoltre programma o circuito per tradurre CW e RTTV. Dino Fornaciari - Villaggio Dante, 30 - 52100 Arezzo - Tel. 0575/351451 (ore pasti).

Vendo-scambio programmi per Commodore 64 disponibili: Tool 64, Simon's, Basic (su cassetta) Frogger, Crazy kong, Calcio, Scramble, Rox 64 e molti altri, richiedere lista allegando L. 300 per spese postali a: Carmelo Cutuli - C.so Italia, 60 - 95014 Giarre (CT).

Cerco ad Arezzo utenti Commodore 64 per scambio esperienze. Cerco inoltre softare di termotecnica per Commodore 64 o Sharp PC 1211. Dino Fornaciari - Villaggio Dante, 30 - 52100 Arezzo - Tel. 0575/351451.

Vendo per VIC 20 molti programmi su cassetta, giochi utility e altro anche in L.M. espanso a L. 600 l'uno. Richiedere liste gratuite. Sono disposto anche per cambi. Antonio Liuni - Parco dei Principi, 50 - 70010 Casamassima (BA) - Tel. 080/671708.

Vendo C 64 Simons'Basic L. 50000, Calcio L. 25000, Pacman L. 15000, Skirace L. 15000, Pool Biliard L. 15000, Fort Apocalips L. 20000, Shanus L. 20000, Scramble L. 20000, Gridder L. 15000, Motormania L. 15000, Scacchi L. 40000, Frogger L. 20000, conti correnti L. 30000. Giacom Natali - Via S. D'Acquisto, 19 - 62010 Petriolo (MC) - Tel. 55201.

Se possiedi un VIC 20 non puoi non far parte dell'eden software club! Scrivici, riceverai dettagliate spiegazioni ed un interessante bollettino omaggio. Per informazioni: Rinaldo Denti - Via Bellane, 4 - 10025 Pino Torinese (TO).

Per i vecchi e nuovi possessori del prestigioso VIC 20 e 64 *vendo* in cassetta di alta qualità 6 videogiochi originali Commodore inglesi e italiani poseggo vasta gamma di giochi 20.000 per cassetta. Diego Gelsomino - Via Cerrignano, 2 - 20142 Milano - Tel. 575444.

Vendo-compro-cambio programmi per CBM 64 sia su nastro che su disco. Scrivete a: Luigi Beviglia - Casella Postale 41 - 21052 Busto Arsizio (VA).

Vendo-cambio giochi e programmi su cassetta per Commodore 64 solo residenti in Torino. Telefonare ore pomeridiane 14-17 esclusi giorni festivi a Claudio. Claudio Genova - Via S. Ambrogio, 9 - 10139 Torino - Tel. 711809.

È nato il nuovo computer club Commodore ogni livello a chi interessa *compro-vendo-scambio* esperienza curiosità tecnica o voglia apportare esperienze e dialogo scrivere a: Spartaco Della Spora - Via P. Savi, 218 - 55049 Viareggio.

Cerco possessori Commodore 64 per scambio di programmi. Cerco manuali e qualsiasi altro materiale riguardante questo computer e il suo modo di operare. Posseggo anche diversi giochi interessanti. Gilberto Frioli - Via G. Mazzini, 32 - 60035 Jesi (AN) - Tel. 0731/541792.

Cerco possessori di Commodore CBM 64 per scambio programmi e idee richiedete la lista e/o inviate la vostra. Rispondo a tutti. Stefano Belluzzi - Via Giacomo Matteotti, 179 - 46025 Poggio Rusco (MN).

Vendo espansione di memoria 8K per VIC 20 a L. 50000. Gioco scacchi Sargon II a L. 30000. Telefonare ore 14.00-14.30, o scrivere. Lorenzo Longagna - Via Piave, 20-7 - 17100 Savona - Tel. 019/25322.

Per Commodore 64 *vendo* più di 500 programmi in LM ED Utility. Amodio Tortora - Via Muratori, 44 - 57100 Livorno - Tel. 500422.

Vendo per VIC 20 base e Exp 8K, splendidi giochi. Tipo "Arcade" su nastro. Chiedere elenco prezzi gratuito. Lailes Lorenzale - Via Marzabotto, 14 - 20094 Corsico (MI) - Tel. 4583226.

Per CBM 64 *vendo-cambio* vasta gamma giochi ed utilityes. Richiedere ed inviare liste. Disponibili sia su cassetta che su disco. Telefonare ore serali a: Paolo Raimondo - C.so Vittorio Emanuele, 71 - 10128 Torino - Tel. 011/545625.

Vendo 20 programmi per TI99/4A dispongo di Hammurabi, Peg Jump, Poker, Othello, Market simulation, ecc. Il tutto a sole L. 25000. Vendita contrassegno richiedere elenco dettagliato a: Paolo Rosi - C.so Italia, 34 - 41058 Vignola (MO).

Amici vichinghi *vendo* programmi a prezzi eccezionali, siete principianti? Mandatemi il vostro indirizzo vi spedirò una cassetta con 30 giochi divertenti in contrassegno a L. 15.000 SCRIVETEMI!!! Gianluca Burattelli - Via Di Vittorio, 2 - 58022 Follonica (GR) - Tel. 40903.

Vendo per VIC 20 i seguenti giochi su cassetta: Medioevo, Blitz, Sci, Invaders, Follet, Amok, Frog, Vic-rescue, Bowling. (quasi tutti per Joystick) tutto a L. 20.000. Telefonare ore pasti. Alberto Artioli - Via M. Marzabotto - 0425 S.M. Maddalena (RO) - Tel. 0425/757672.

Cambio-vendo programmi per CBM 64 The Last one - PET Speed - Hesmomd alta risoluzione magazzino Condominio Word Vari - PET Emulator Simon's Basic, Bilancio familiare e tanti fantastici giochi originali. Augusto Bernardini - Via Valle Verde, 5 - 05100 Terni - Tel. 56870/47148.

Per CBM 64 *compro* qualunque tipo di programmi. L'annuncio è valido sempre (anche per acquisto libri). Inviate le vostre liste a: Leonardo Lombardi - Via Largo A. Banfi, 4 - 50018 Scandicci (FI) - Tel. 256640.

PICCOLI ANNUNCI

Vendo-cambio software su cassetta per Commodore 64: Utility, Videogames, ecc. Stefano Lomurno - Via Giambellino, 10 - 20146 Milano - Tel. 4235248.

Eccezionale offerta CBM 64 *vendo* programmi come "Simon's, Screen Graphics, Data Base" e giochi: Atomo, Alto Medioevo, Calcio, Scacchi, Frogger, Atlantic City. Tutto con testi esplicativi. Luca Montalbano - Via Malagrida, 14 - 65100 Pescara - Tel. 085/34196.

Cambio-vendo programmi per Commodore serie 3000, 4000 per ricevere ampio elenco inviare L. 2000. Si producono anche programmi su commissione scrivere o telefonare ore pasti o dopo le 21.00. Aroldo Bizzarri - Via Pantelleria, 19 - 91100 Trapani - Tel. 0923/20044.

Compro-cambio-vendo programmi per Commodore 64, di qualsiasi tipo. All'invio della lista (indicare la trattativa che vi è più congeniale), risponderà con la mia. Franco Bruno - Via Giorgio Bratti, 100 - 47023 Cesena (FO) - Tel. 0547/23810.

Vendo software per CBM 64, soprattutto giochi, se siete interessati richiedete la lista dei programmi a: John Ceresi - Via Mazzini - 18016 S. Bartolomeo Mare (IM).

Per VIC 20 *vendo-scambio* software su cassetta dispongo di circa 100 programmi di cui alcuni fantastici in LM. Es. Gridrunner, Abductor, Alien Blitz, Blitz a prezzi stracciati. Mandatemi le liste vostre o/e chiedete la mia gratis. Leonardo Buzzano - V.le Cairoli, 111/B - 31100 Treviso - Tel. 0422/260808.

Vendo VIC 20 + registratore C2N + espansione 8KB + software 40 colonne + altri programmi e cavetti L. 250.000 + postali. Alberto Del Bianco - Via Trasimeno, 7 - 52100 Arezzo - Tel. 300692.

Vendo computer Commodore CBM 4016, 16 K, monitor 11" fosfori verdi incluso, + registratore C2N + manuale Basic 4.0 + manuale Jackson. Il tutto a L. 650.000. Telefonare dopo le ore 20.00 Sergio Giannesi - Via Ovada, 19 - 20142 Milano - Tel. 02/810868.

Vendo VIC 20 + cavi e trasformatore + 2 cartucce (scacchi, Hesmon monitor in linguaggio macchina) + registratore C2N +Joystick + cassetta programmi + manuale + 1 libro L. 400.000. Solo zona Roma. Davide Petrarca - Via Epicarmo, 21/23 - 00125 AxA Roma - Tel. 06/6062238.

Vendo o cambio programmi per VIC 20, cerco inoltre programmi in linguaggio macchina.
Leonardo Pecchi - Via SS Per Correggio, 22 - 41012 Carpi (MO) - Tel. 059/664219.

Cambio listati di programmi per Commodore 64: fra i tanti giochi possiedo listato in linguaggio macchina di un Toolkit. Rispondo a tutti. Giulio Frigo - Via Balai, 78/C - 07046 Porto Torres (SS).

Commodore 64 *vendiamo*. Sviluppiamo programmi disponibile vasta biblioteca di programmi molti con istruzioni in italiano listino a richiesta allegare francobolli per la risposta. Marco De Nittis - Via Varese, 22 - 71016 S. Severo (FG).

Cerco possessori di Commodore 64 per scambio idee e programmi sono interessato tra l'altro al Simon's Basic su cassetta che scambierei con altri programmi o acquisterei se a prezzo modico. Inviare liste. Paolo Di Mauro - Via Bertieri, 1 - 20146 Milano - Tel. 471803.

Compro-vendo programmi di ogni tipo per Commodore CBM 64. Scrivere a Paolo Bigon - Via S. Marco, 177/A - 35020 Padova - Tel. 049/629117 (ore 21.00).

Vendo per CBM 64 software giochi di vario tipo come: Frogger, Pac-Man, su cassetta: per informazioni scrivete e riceverete una lista con i prezzi oppure telefonate. Preferisco contattare in zona. Alessandro Palma - Via A. Longo, 50 - 80127 Napoli - Tel. 081/656305.

Vendo bellissimi programmi di giochi su cassetta per VIC 20 inviare 400 lire in francobolli per la lista e i prezzi regalo inoltre ai primi 5 - programmi vari . scrivetemi. Gerardo Ventura - Via Regina Elena, 82 - 65100 Pescara.

Vendo bellissimo programma di simulazione spaziale per CBM 64. Potrete esplorare una parte di galassia alla ricerca di stelle instabili, 22K chiedere informazioni con francobollo 400 lire a: Stefano Zattini - Via Sforza, 33 - 47100 Forlì - Tel. 26271.

Vendo VIC 20 in perfette condizioni + superxpander + alimentatore + modulatore + numerosi programmi in Basic + Libro VIC revelead + videogame alien + registratore e 1 game a L. 500.000. Telefonare ore pasti (12-14) o (19-21). Lorenzo Battaglioni - Via Zanaboni, 87 - 44038 Pontelagoscuro (FE) - Tel. 0532/461717.

Vendo corso Basic prima parte per VIC 64 interamente in Italiano comprendente 2 cassette e 1 manuale a L. 30.000 (metà del prezzo di vendita) tratto solo con Bari disposto a dimostrazioni. Nicola Lavopa - Via T. Cardarelli, 22 - 70125 Bari - Tel. 472842.

Cerco-scambio programmi in linguaggio macchina per VIC 20 e CBM 64 richiedere la lista. Cerco vichinghi nelle tre venezie e oltre per apertura di Club VIC. Roberto Oselladore - Passo San Boldo, 35/2 - 30030 Favaro Veneto (VE) - Tel. 041/631106.

Vendo-scambio cartucce (visible solar sistem) per VIC 20. Nuovissima (1 mese) ancora nell'incarto originale a L. 30.000 trattabili. Compro registratore Commodore a basso prezzo. Enzo Capia - Via Almitelli, 1 - 80011 Acerra (NA) - Tel. 081/885...

Per CBM 64 *vendo-scambio* interessanti Utility e giochi disco o cassetta per informazioni: Ernesto Perini - Via Petrella, 15 - 21052 Busto Arsizio (VA) - Tel. 0331/635025.

Per VIC 20 ho una nastroteca di circa 100 programmi (videogiochi, didattica, gestionali, giochi di società, grafici, per la casa, ecc.) che vendo a prezzi da sballo. Ricco e dettagliato indice inviando L. 1000, anche in francobolli a: Giovanni Vermiglio - V.le Friuli, 27 - 10015 Ivrea (TO) - Tel. 0125/252170.

Compro-vendo-cambio programmi per CBM 64, soprattutto su disco. Emilio Di Lello - Via Giotto, 3 - 64026 Roseto D'Abruzzo (TE) - Tel. 085/8992146.

Vendo-cambio-acquisto software per CBM 64. Inviate vostra lista rispondo a tutti a stretto giro di posta. Ho circa 300 (trecento) programmi. Annuncio sempre valido. Giancarlo Giuliani - P.za Sacro Cuore, 26 - 65100 Pescara - Tel. 085/26593.

Vendo favolosi giochi di tutti i generi: sportivi, Arcade games, classici, di avventura, spazio, e di azione. Prezzo variabile dalle L. 5.000 fino a 15.000. Solo per Commodore 64. Luca Vecchi - Via Valeriani, 39/2 - 40134 Bologna - Tel. 051/412652.

CBM 64 *scambio-vendo-compro* programmi di vario genere. Dispongo di ottimi giochi, ma anche di stupendi programmi di utilità. Richiedere la lista inviando la vostra. Massima serietà. Vincenzo Sena - Via Ferrovia, 26 - 80033 Cicciano (NA).

Vendo cartucce (cartridge) per VIC 20 "Alien", "Avenger", "Radar Rat Race" in perfetto stato, usate pochissimo, 1 mese di vita, funzionanti a L. 35.000/40.000. Vero affare. Luca Maccari - Via Pismonte, 5 - 20139 Milano - Tel. 02/5391991.

Vendo per VIC 20 i migliori giochi sul mercato. Stupendi LM ed anche ottimi Basic ed inediti americani. Tutto a prezzi adeguati. Scrivete per ricevere il fornito elenco gratuito. Federico Guerrieri - Via Ugo Foscolo, 14 - 50124 Firenze - Tel. 055/700635.

Se possiedi un VIC 20 non puoi non far parte dell'eden software club, il club delle idee, delle amicizie, degli scambi, scrivici. Riceverai un interessante bollettino omaggio. Rinaldo Denti - Via Bellane, 4 - 10025 Pino Torinese (TO).

Commodore 64 *vendo* programmi, cerco manuali di istruzione sia sul CBM 64 che di programmi acquisto in contanti o software di adeguato valore. Grassi Giancarlo - Via Vasto, 81 - 46044 Goito (MN) - Tel. 0376/607239.

Vendo-scambio i seguenti programmi per C 64: Basic 4.0, Pet Speed 80 colonne, Mon Easy Script, Screen Graphics, Simon's Basic, Disk Copy, Disk Backup, Videogames, Totosistemi e altri, Telefonare o scrivere. Saverio Pensabene - Lungomare C. Colombo, 3544 - 90149 Addaura (PA) - Tel. 091/451425.

Sinclair

Attenzione! Per Spectrum 16/48 K *vendo* bellissimi programmi registrati perfettamente e collaudati sia giochi (The Hobbit, Scacchi) che utilità, (Vufile, Vucalc). Telefonare ore 15 o scrivere. Omaggi su quant. Marco Moretti - P.zza Dei Giudici, 2 - 50122 Firenze - Tel. 055/296115.

Vendo ZX81 compreso di cavetti, libro istruzioni + espan. memoria 16 K RAM + libro 66 programmi usato pochissimo cedo tutto a L. 200.000. Francesco Castellan - C.so Vittorio Emanuele, 188 - 67100 L'Aquila - Tel. 21223.

Cambio-compro-vendo programmi ZX Spectrum inviate le vostre liste oppure chiedete la mia risposta assicurata (oltre 300 programmi) originali inglesi. Bruno Mautone - Via Trentino, 74 - 80145 Napoli - Tel. 081/7540707.

Se vuoi divertirti col tuo ZX Spectrum 16 o 48 K hai trovato l'occasione giusta non è uno dei soliti annunci perchè questa è veramente un'occasione 300 programmi tutti originali su cassette di marca. Enrico Tadini - Via Mameli, 29 - 16035 Rapallo (GE) - Tel. 0185/60935 (sera).

Vendo ZX-81 16K RAM alimentatore. Moltissimi software anche inediti L. 200.000. Alessandro Giolitti - Via G. Fabroni - 50134 Firenze - Tel. 473810.

PICCOLI ANNUNCI

Eccezionale *vendo* ZX-81 ricarrozzato ed esteticamente imbattibile montato in contenitore metallico nero con tastiera professionale, espansione 64K RAM + ZX Printer a sole L. 450.000 pensateci!!! Gianni Arcieri - Via Nazionale, 168 - 64020 Ripattoni (TE) - Tel. 0861/610493.

Vendo ZX Spectrum con espansione a 48 K + interfaccia seriale RS232 Centronics + manuale a L. 500.000. In omaggio software. Andrea Tomat - Via Fermi, 23 - 20094 Corsico (MI) - Tel. 4401277.

ZX Spectrum *vendo-cambio* oltre 350 programmi originali inglesi in continuo aggiornamento scrivere o telefonare per catalogo. Akis Fletsios - Via Ferriera, 11 - 40132 Bologna - Tel. 051/341952.

Cambio-vendo programmi per ZX Spectrum 16/48 Kb, ho decine di programmi di ogni tipo. Stefano Sansavini - Via De Macchi, 18 - 50100 Firenze - Tel. 652321.

Vendo programmi per Spectrum in listati o su cassetta (L. 2000 e 8000 rispettivamente). Dispongo anche di listati per Commodore, Atari, Apple ed altri a L. 3000. Programmi anche su richiesta. Daniele Antonio Iavarone - Via Torino, 9 - 81022 Casagiove (CE) - Tel. 0823/468607.

Cerco possessori ZX Spectrum per scambio programmi. Inviate la vostra lista ed io invierò a voi la mia. Risposta assicurata. Alfredo Trifiletti - Via Fiume, 20/A - 71100 Foggia - Tel. 0881/75385.

Richiedere catalogo giochi Spectrum inviando L. 500 per spese postali. Indicare chiaramente mittente. Prezzi eccezionali. Francesco Imbesi - Via Deledda, 9 - 17025 Loano (SV).

Vendo ZX-81 con alimentatore + cavetti + espansione 32 K RAM + libro programmi + manuale + 3 cassette giochi applicativi. Il tutto a L. 250.000 telefonare ore 13/14. Davide Longobardi - Via Giovannida Milano, 7 - Milano - Tel. 7388675.

Vendo fantastici programmi per ZX Spectrum originali a L. 4000. Scrivere o telefonare per ricevere listino. Risposta assicurata. Guido Montacchini - Via Tiepolo, 7 - 10126 Torino - Tel. 677670.

Straordinario Sinclair su nastro! Spectrum: 20+20+20 progr. (3 nastri diversi). ZX-81: 50+50 progr. 1K, 35+35 prog. 16 K, 20 maxiprog. 16 K. Ogni nastro L. 7000 solo se soddisfatti dopo averli provati per 10 giorni richiedili a: Vincenza Avena - Via Garibaldi - 04016 Sabaudia (LT).

Vendo ZX-81 32 K RAM, tastiera premente con repeat e beep, bus, cavi e alimentatore, 2 manuali in italiano, documentazione hardware e software, programmi in LM registrati. Tutto L. 240.000 irriducibili. Mauro Masci - Via M. Odescalchi, 2 - 00152 Roma - Tel. 06/530951.

Vendo causa passaggio sistema superiore, computer ZX-81 come nuovo (maggio '83) + cavetti + 16 KRAM + manuale + software il tutto perfettamente funzionante e imballo originali a L. 300.000. Scrivi subito a: Maurizio Della Sala - Casella Postale 84016 - Pagani (SA).

Scambio programmi Spectrum 48 K tratto solo con privati in Milano. Telefonare ore serali. Guerrino Neroni - Via Malakoff, 20 - 20094 Corsico (MI) - Tel. 02/4471898.

Vendo programmi ZX Spectrum, Splat, Missile Defence, Fruit Machine, Jungle Trouble, Cookie e molti altri. Catalogo a richiesta L. 1000. Stefano Nocilli - Via Truscolana, 224 - 00181 Roma.

Vendo programmi per Spectrum in Ling. macchina su cassetta 7 giochi e 4 utility 16-48 K ottima grafica. Eccezionale prezzo L. 8000. Telefonare dalle 20-21 ottima serietà istruzioni incluse. Giacomo Caviglione - C.so Ferrari, 181/3 - 17011 Albisola Capo (SV) - Tel. 019/42966.

Sinclair Spectrum disponendo di un notevole archivio software *vendo* a L. 10.000 cassette con 5 giochi a scelta. Ore pasti. Pivano Parbuono - Via A. di Cambio, 4 - 37138 Verona - Tel. 045/568649.

Compro listati o cassette per ZX Spectrum 48 K e TI 99/4A extended riguardanti il programma di compusteria e di ragioneria per il triennio dell'istituto tecnico commerciale. Angelo Bottini - Via Dante 31/5 - 18039 Ventimiglia - Tel. 357902.

Vendo-scambio programmi di giochi ed utilità per Spectrum 16/K e 48/K. Vasto assortimento e garanzia di massima soddisfazione. Richiedi il catalogo inviando il francobollo per la risposta e l'eventuale tuo elenco.
Francesco Giorgio - Via Quilico - 10018 Pavone Canavese.

Vendo programmi per ZX 81 Spectrum VIC 20 TI 99/4A C 64 a L. 1000 cad. Telefonare dopo ore 20.00. Fabrizio Cattaneo - C.so Sempione, 63 - Milano - Tel. 386037.

ZX-80 8 K ROM 16 K RAM manuali vari completo di cavi cassette e valigetta + cassetta scacchi, videogiochi Philips video Pak G 7000+10 cass. + cass musica. Solo genova e provincia tel. ore pasti. Alberto Schmuckher - C.so Torino 26/14, 16129 Genova - Tel. 010/584292.

Cerco traduzione del programma VU3D della Sinclair, per TI 99/4A, lauta ricompensa, il programma può essere fornito per qualsiasi configurazione pel il TI 99/4A. Salvatore Imparato - Via Banca Naz. Lavoro - 80027 Frattamaggiore (NA) - Tel. 081/8801301 - 8801450.

Cambio-vendo programmi per ZX Spectrum (giochi utility linguaggi ecc.) L. 4000 cad. Chiedere listino gratuito con oltre 250 programmi anche telefonicamente (ore pasti). Mario Bontempi - Via Valle, 7 - 25087 Salò (BS) - Tel. 0365/40637.

Vendo-scambio fantastici programmi per ZX-81 16 K in LM come 3D defender scacchi invader e tanti altri a prezzi bassissimi. Inviare bollo L. 400 per contributo spese postali. Paolo De Cupis - Via Dei Gladioli, 2 - 00048 Nettuno (Roma).

Vendo-cambio software su cassetta per ZX Spectrum. Emmanuele Nerantzulis - Via Gramsci, 35 - 20037 Paderno Dugnano (MI).

Spectrum soft club. L'iscrizione è gratuita ed aperta a tutti. I soci hanno libero accesso ai programmi del club e contribuiscono ad aumentare il numero. Per maggiori informazioni scrivere a: Marino Marinanza - Via F.B. Rastrelli, 102 - 00128 Roma - Tel. 06/5203292.

Vendo Sinclair ZX-81 + espansione 16 K + alimentatore con manuali in inglese ed italiano L. 170.000 + cartridge word processing a L. 80.000. Roberto Turolla - Via Gavirate, 19 - 20148 Milano - Tel. 4074518.

Vendo ZX-81 usato per un breve periodo con manuali italiano e inglese e alcune cassette per memoria base a L. 90.000. Telefonare ore serali. Alessio Nigrisoli - Via 8° Strada, 23 - 20090 S. Felice (MI) - Tel. 02/7531517.

Vendo-cambio software per ZX Spectrum 16-48 K; dispongo di utility in assembler per vari impieghi. Cerco Vufile-e-Vucalc. Inoltre vendo prog. per C 64 anche in linguaggio Fourth. Ferruccio Zamuner - Via G. Di Vittorio, 22 - 10023 Chieri (TO) - Tel. 011/942282.

Vendo ZX Spectrum dicembre 1983 garanzia da timbrare con 7 programmi (Jetpac - Pool - Scacchi - Arcadia - Orazio va a sciare - 3D Tanx - Galaxians) L. 320.000 intrattabili. Telefonare ore cena tranne sabato. Leone Maurizio - Via Pian Delle Mele, 16 - 65100 Pescara - Tel. 085/414706.

Cerco listati programmi per Sinclair Spectrum. Siano essi di giochi o utilità. Disponibile a sostenere spese di fotocopiatura e spese postali. Inviarli a: Domenico Garofalo - Via Panebianco 5° Strada Co-op. Frate Umile - 87100 Cosenza.

Vendo in perfette condizioni ZX-81 + 16 K RAM + alimentatore e cavetti + manuale in italiano + software (cassette e listati) a L. 130.000. Paolo Gravellini - Via C. Colombo, 22 - 20049 Concorezzo (MI) - Tel. 039/640812.

Vendo programmi su nastro. Spectrum: 20 + 20 + 20 progr. (3 nastri diversi). ZX81: 50 + 50 prog. 1K, 35 + 35 16 K, 20 maxi programmi 16 K. Ogni nastro L. 7000, contrassegno più 2000. Richiederli a: Bruno Del Medico - Via Torino, 72 - 04016 Sabaudia.

Vendo software Spectrum su cassetta - The hobbit VU 3D, simulatore di volo 48 K, monitor and disassembler. L. 12.000 cad. L. 40.000 in blocco. Roberto Ghezzi - Via Volontari del Sangue, 202 - 20099 Sesto San Giovanni (MI) - Tel. 02/2485511.

Vendo computer ZX-81 con alim. al migliore offerente. A chi lo desidera vendo anche il manuale originale in inglese del ZX-81 e 5 cassette magnetiche e 11 dispense del linguaggio Basic. Edoardo Triscari - Via G. Ripamonti, 16 - 20136 Milano - Tel. 571979.

Vendo-cambio software su cassetta per ZX Spectrum. Giovanni De Stefani - Via Capodistria, 18 - 31100 Treviso -Tel. 261383.

Vendo ZX80 completo di alimentatore e manuale L. 60.000. Vendo terminale Olivetti DE 521 con due unità a cassette L. 300.000. Adriano Scapini - V.le Sicilia, 34 - 37138 Verona - Tel. 045/572740 (ore pasti).

Vendo oltre 100 programmi molto recenti per ZX Spectrum 16 K (L. 5000) e 48 K (L. 7000) in linguaggio macchina. Giorgio Gatteschi - Via Roma, 430/0 - 50012 Bagno a Ripoli (FI) - Tel. 055-631312.

Vendo tutti i programmi per ZX Spectrum sia 16K che 48K sono più di 200 programmi. Venduti in cassette di buona marca 10.000 per ogni programma sia 16K che 48K. Su ordinazione tutti i programmi Apple. Enrico Tadini - Via Mameli, 29 - 16035 Rapallo (GE) - Tel. 0185/60935.

PICCOLI ANNUNCI

Per ZX Spectrum *vendo* penna ottica L. 40.000. Vendo inoltre più di 200 programmi a prezzi molto bassi. Oltre a molti giochi ho programmi di ingegneria, utilità ecc. chiedere elenco. Carlo Celi - Via Giorgetti, 25 - 32100 Belluno - Tel. 0437/27016.

Cerco possessori di Spectrum di Pavia e provincia per scambio programmi. Vorrei fondare anche un Sinclair club in Pavia. Matteo Longhini - Via San Giovannino, 5 - 27100 Pavia - Tel. 38179.

Gruppo di hobbisti esperti di informatica ha fondato "Soft bank" per Spectrum: circa 400 programmi selezionati e subito pronti, libri, interfaccia Joy, espansioni 80 K a innesto, catalogo L. 800. Luigi Roberto Callegari - Via Alcide De Gasperi, 47 - 21040 Sumirago (VA) - Tel. 0331/909183.

Spectrum software per radioamatori *vendo* oscar 10 Tracking ORB satelliti Russi e americani Moontracking Shuttle etc. 14 programmi a L. 15000 comprese spese postali. Gianni Matteini - Via C. Pavese, 20 - 47041 Bellaria (FO) - Tel. 0541/44292.

Help *cercasi* espansione 32 K per ZX-81 in cambio do gioco televisivo della soundic TVC e centralina luci psichedeliche per informazioni telefonare dopo le 8.30 e chiedere di Nicola. Nicola Amato - Via Alessandro Manzoni, 5 - 28037 Domodossola (NO) - Tel. 0324/44918.

Vendo-cambio oltre 100 programmi per ZX Spectrum 16/48 K a L. 10.000 tutti originali inglesi fra i quali: manic miner, bugaboo, Jet pac, chequered flag, ecc. Per richiesta della lista inviare L. 1000 in francobolli oppure telefonare dopo le 20.00 a: Massimiliano Marconi - Via Cantagallo, 64/0 - 50047 Prato (FI) - Tel. 0574/460716.

Hai uno ZX-81 ad un solo K di RAM potenziato a 16 K con sole L. 85.000 prova poi l'espansione con una vera cassetta il Tirannosauro e con sole 100.000 ti farai un regalo coi fiocchi. Scrivi subito. Nicola Mazzariello - Via Veneria Reale, 40 - 20030 Villaggio SNIA (MI) - Tel. 0362/522476.

Soft club Sinclair *vende* software per ZX81 con e senza espansione su listato o cassetta - oltre 150 programmi - inviare L. 600 in francobolli per ricevere lista programmi a: Soft Club Sinclair di: Antonio Nicosia - Via Galatea, 13 - 93100 Caltanisetta - Tel. 33270.

Scambio-vendo programmi per ZX Spectrum 16/48 K dispongo di molti titoli. Scrivete e vi manderò la mia lista. Risposta assicurata massima serietà. Alessandro Augello - V.le Del Fante, 56 - 90146 Palermo - Tel. 501740.

Se hai un Sinclair sei nostro amico. Scrivici o telefona disponiamo di tutti i programmi Spectrum, libri, adesivi, bollettino a prezzi fantastici! Indirizzare al "Gruppo utilizzatori computer Sinclair c/o". Roberto Chimenti - Via Luigi Rizzo, 18 - 80124 Napoli - Tel. 081/617368.

Compro listati di programmi per ZX81 girabili ad 1K di memoria programmi di qualsiasi genere vorrei corrispondere con possessori di ZX-81. Dante Bosia - Via Italia, 8 - 20076 Maleo - Tel. 0377/58168.

Vendo-cambio software per ZX Spectrum 16-48 K, dispongo di Utility in assembler per vari impieghi. Cerco Vufile e Vucalc. Inoltre vendo progr. per C 64 anche in linguaggio "Fourth". Ferruccio Zamuner - Via G. di Vittorio, 22 - 10023 Chierti (TO) - Tel. 011/9422829.

Vendo ZX Spectrum completo di cavetti per il registratore, alimentatore, manuale di istruzioni in italiano + 2 programmi in cassetta, a L. 350.000. Walter Lubatti - Via Sestriere, 53/2 - 10024 Moncalieri (TO) - Tel. 011/6066975.

Vendo a L. 5000 ciascuno numerosi programmi in linguaggio macchina per ZX Spectrum 16 o 48K chiedere elenco aggiornato e maggiori dettagli a: Cosimo Dibello - Via Perugini, 19 - 70043 Monopoli (BA) - Tel. 080/746126.

Causa passaggio sistema superiore *vendo* ZX-81 con alimentatore e cavi + 2 manuali uno inglese ed uno italiano + vari listati a L. 90.000 usato pochissimo, ancora in imballo originale. Roberto Poletti - Via F. Cilea, 14 - 50054 Fucecchio (FI) - Tel. 0571/22906.

Vendo 25 giochi originali inglesi su cassetta per ZX Spectrum a L. 50.000 + spese (spedizione - eventuali fotocopie di manuali). Dispongo inoltre di software di ogni tipo. Chiedere lista con 200 titoli a: Nico Dialessandro - Via C. Alcide De Gasperi, 413/D - 70125 Bari - Tel. 412470.

Cambio-vendo software Spectrum 16/48 K. Buon prezzo inviatemi vostro elenco per scambio vi invierò il mio. Oppure richiedete la mia lista per acquisti allegando L. 400 in francobolli. Erminio Benzoni - Via Del Cipresso, 4 - 22050 Perledo (CO) - Tel. 0341/830026 (ore serali).

Sinclair su nastro. Spectrum 20+20+20 progr. (3 nastri diversi). ZX81: 50+50 progr. 1K; 35+35 progr. 16 K; 20 maxi prog. 16 K. Incredibile ogni nastro L. 7000. Volendo pagare al postino aggiungere L. 2000. Bruno Del Medico - Via Torino, 72 - 04016 Sabaudia (LT).

Spectrum *vendo* programmi gioco e utilità risposta garantita invio elenco gratuitamente scrivere a: Antonio Sfriso - Via Salomone, 7 - 30173 Mestre (VE) - Tel. 041/972887 cena.

Vendo programmi per ZX Spectrum a prezzi bassi. Scrivere o telefonare (di sera) per ricevere gratis l'elenco. Stefano Calcaterra - Via Marconi, 34/2 - 40122 Bologna - Tel. 051/521063.

Vendo programmi Spectrum 16/48 K originali inglesi a max L. 12.000 per i migliori da 48 K. Richiedere elenco inviando francobollo a: Maurizio Leone - Via Gaio Melisso, 16 - 00175 Roma - Tel. 7662671.

Texas

Texas TI99/4A + alimentatore + collegamento TV + cavo interfaccia registratore + corso Basic in cassetta + modulo scacchi + coppia Joystick + software vario a L. 320.000 trattabili (!!) *Vendo*. Telefonare ore pasti. Fabio Scaffidi - V.le Casiraghi, 34 - 20099 Sesto S. Giovanni (MI) - Tel. 248756.

Per passaggio a sistema superiore *vendo* Texas TI-99/4A completo di manuale, alimentatore, modulatore. Pal tutto originale a L. 350.000 (+ 1 cassetta). Per informazioni rivolgersi a: Maurizio Levoni - Via Medaglie D'oro, 46 - 41100 Modena.

Vendo TI 99/4A + gioco TI invaders + gioco di Basic x TI 99 + manuale d'uso e tutti i cavi speciali L. 330.000 trattabili. Luca Cislaghi - Via Melloni, 10 - 20129 Milano - Tel. 02/708307.

Per TI 99 offro numerosi programmi giochi archivio e funzionamento dei file - matematica (derivate integrali zeri funzione calcolo equazioni) il tutto a prezzi popolari francobolli per la lista. Daniele Catalfamo - Via Guido Reni 219/5 - 10137 Torino - Tel. 305093.

Acquisto dietro rimborso spese ed eventuale equo compenso, copia corretta listato "Factor Foe" per Texas TI 99/4A. Raffaele Ferrigno - Via Andrea D'Isernia, 4 - 80122 Napoli - Tel. 081/681571 - 685720.

Vendo programmi per il TI 99/4A da L. 2000 in su. Sono disponibili anche programmi in extended Basic. Richiedere lista a: Raffaele Avino - Via Lepanto - 80045 Pompei (NA) - Tel. 081/8632802.

Vendo-scambio programmi per il computer della Texas Instruments TI 99/4A. Disponibili solo su cassetta. Per informazioni telefonare di pomeriggio. Luca Maggioni - Via XXV Aprile, 29 - 20090 Segrate (MI) - Tel. 2131056.

Vendo per Texas TI 99/4A a sole L. 3000 ottimi programmi registrati su cassetta. Luciano Iapichino - Via Avigliana, 16 - 10098 Rivoli (TO) - Tel. 011/9533290.

Scambiamo-vendiamo software per il TI 99/4A: giochi, matematica, ingegneria (anche in extended). Richiedete la lista di oltre 500 programmi inviando bollo L. 500. Massima serietà e celerità. TI 99 IT Users' club - Via Mascarella 104/9 - 40126 Bologna - Tel. 051/224310 (Lunedì pomeriggio).

Texas TI 58C poco usata imballo originale con manuali e alimentatore *vendo* L. 100.000 scrivere o telefonare ore pasti. Valentino Bilardi - Via Stampa, 2 - 10010 Settimo Vittone (TO) - Tel. 0125/758356.

Vendo per possessori TI 99/4A modulo SSS "blackjack & poker" a L. 35.000. Ignazio Marturano - Via Pisanelli, 17 - 74100 Taranto.

Cerco schema elettrico e descrizione tecnica del TI 99/4A. Pietro Viani - Via 5 Maggio, 11 - 20157 Milano.

Vendo programmi su cassetta per TI 99/4A in Basic. 10 programmi a sole L. 10.000 escluse le spedizioni. Telefonare solo dalle 17.00 alle 18.00 esclusi sabato e festivi. Osvaldo Danzi - Via Tito Livio, 3 - 80122 Napoli - Tel. 7697702.

Vendo programmi per il TI 99/4A. Sono disponibili inoltre programmi per l'extended Basic fra cui un interessante word processing richiedere lista a: Andrea Barbieri - Via Livorno, 12/A - 35100 Padova.

Vendo software per TI 99/4A. Disponibili in Basic o Extended Basic, in vendita a L. 6000 comprese cassetta & spedizione. Romano Perico - Via Geroni, 2 - 24025 Gazzaniga (BG) - Tel. 035-711993.

Cerco cassetta TI-99 Adventure + pirate a un prezzo trattabile sulle L. 25.000. Guelfi Lionello - Via Vasto, 1 - 20097 S. Donato Mil. (MI) - Tel.5275022.

Vendo programmi per TI 99/4A (molti giochi, utility, ecc.) in TI Basic e in Extended Basic. Inviare un francobollo per ricevere dettagliato listino prezzi. Stefano Dominioni - Via N. Tommaseo, 18 - 21100 Varese - Tel. 0332/229909.

PICCOLI ANNUNCI

Varie

AIUTO!!!! Da alcuni mesi ho a disposizione un personal SHARP MZ 80 B e purtroppo nessuno ne parla mai; tutta la mia riconoscenza a chi volesse darmi consigli su tutto quel che riguarda programmazione BASIC, listati, libri in generale o specifici per lo SHARP MZ 80 B. Giovanni Caligaris - Via Campile, 64 - 13062 Candelo - Tel. 015/53226.

Cercasi programmi di ingegneria civile ("373", travi,...). Solamente su listati oppure su cassette per Sharp MZ 731 o per VIC 20 (non protetti). Scrivere per accordi. Gianluca Renna - Via S. Andrea, 25 - 00046 Grottaferrata (RM).

Acquisto numeri arretrati di computer! e pubblicazioni in lingua inglese riguardanti l'Apple II (manuali, riviste ecc.). Rispondo a tutti. Non mi interessano fotocopie. Scrivere a: Stefano Dellabiancia - P.zza Douhet, 30 - 62016 Porto Potenza Picena - Tel. 0733/688191.

Sirius *vendosi* manuali in italiano, anche su dischetto (Basic C86, CP/M-86, MS-DOS, CIS-COBOL, ecc). Scrivere o telefonare a: Studio EDP-Informatica - Via S. Pio X, 50 - 31031 Caerano S. Marco (TV) - Tel. 0423/858201.

Vendo Newbrain mod. AD completo di cavetti TV e registratore + trasformatore + manuali inglese e italiano + manuali "beginner's guide programs" e cassetta originali a L. 700.000 trattabili. Bedin Giacomo - Via Novara, 37 - 20029 Turbigo (MI) - Tel. 0331/899071.

Olivetti BCS 2030 con software di base mai usato cedo per L. 12.000.000 più 21 rate da L. 343.000. Roberto Napoli - Via Bravetta - 00164 Roma - Tel. 6254611.

Cambio Atari VCS ancora in garanzia con 5 cassette tra cui River Raid MS PAC-MAN, Laser Blast, Combat star Voyager con ZX Spectrum o Commodore C64 o vendo a L. 320.000 telefonare ore 13,30 in poi. Massimo Pecorari - Via Quasimodo, 23 - 62012 Civitanova Marche (MC) - Tel. 0733/74023.

Per Atari 400-800 *vendo* due favolosi giochi: battaglia aeronavale e black-hole, presentati con successo al computer play 83 e ripresi da RAI2 Tandem. Una cassetta L. 60.000 - Cerco Ataristi Bologna. Cristiano Fanucci - Via Spartaco, 29 - 40138 Bologna - Tel. 051/533343.

Vendo enciclopedia EST completamente aggiornata prezzo bomba L. 800.000 una base sicura per addentrarsi nei meandri della scienza e dei computers. (valore commerciale L. 1.200.000). Riccardo Pucher - Via Roma, 174/A - 30038 Spinea (VE) - Tel. 041/991987.

Cambio software Sharp MZ80A/K specie giochi. Dispongo indirizzi poke SP5025 SA5510 utilities subroutines modifiche Basic. Rispondo anche a chi non ha programmi da scambiare. Solo per lettera. Letizia Bizzarri - Via Lago Isoletta, 31 - 65100 Pescara.

Vendo videogioco intellivision in perfetto stato più tre cassette (star-strike, poker; beauty and the best della imagic) tutto a L. 300.000 o permuto con computer T.I. 99/4A. Paola Pinto - Via Oberdan, 86 - 73100 Lecce - Tel. 0832/33312.

Vendo Osborne 1 completo di manuale in italiano e programmi inusato. Pino Mariani - Via Nenni, 2 - 02014 Ozieri - Tel. 079/786131.

Svendo il primo volume dell'enciclopedia "BASIC" (Curcio ed.) più i primi sette fascicoli del secondo volume. Il 1° volume corredato di copertina è da rilegare. Tutto L. 30.000. Frediano Frumento - Via Gramsci 44/10 - 17047 Vado Ligure (SV) - Tel. 019/884239.

Vendo per Micro Z80 NE - configurato con scheda grafica (LX529) e almeno un drive - dischetto 5" completamente riempito da 14 nuovi programmi di gioco e grafici. Il tutto a L. 20.000 + 8000 per dischetto e postali, in contrassegno. Federico Venier - Via Venezia, 120 - 33170 Pordenone - Tel. 0434/42500.

Vendo-cambio programmi di ingegneria (Kani 130, package zona sismica, ecc. Data base ecc.) a prezzi super convenienti o cambio con equivalenti. Scrivere o telefonare solo per Sharp MZ80B. Stefano Lazzaro - Via Molte Sabotino, 2 - 35141 Padova - Tel. 049/22675.

Cerco possessori Dragon 32 per scambio programmi e idee. Luca Benetti - Via S. Andrea, 16/A - 19100 La Spezia - Tel. 27372.

BASIC club per *scambio* software, opinioni, consigli e tutto senza pagare una lira. Per informazioni scrivere a De Rensis Nicola Club - Via M. Buonarrotti, 28 - 51100 Pistoia - Tel. 27197.

Computer MPF II compatibile Apple, con pochi mesi di vita, completo di tastiera esterna. Manuale di programmazione e d'uso. Con 5 programmi in omaggio. *Vendo* a L. 800.000 Telefonare ore serali. Tel. 02/585633.

Vendo software ingegneria civile per Triump adler alphtronic P2 e PC - torsione - forze sismiche - telai sino a 110 nodi - aree acciaio - ecc. Per informazioni ing. Cesare Ambrosini - Via Cavallo, 85/B - 60019 Senigallia (AN) - Tel. 071/64989.

Vendo monitor 12" Kyber a fosfori verdi a L. 120.000. Vendo inoltre monitor colore 14" Hantares con interfaccia per Apple II a L. 560.000 non trattabili. Walter Franceschi - Via Binel, 4 - 11020 Donnas (AO) - Tel. 0125/82374.

Occasionissima! *Vendo* consolle per videogiochi Philips (tastiera elettronica) + una cassetta (asteroidi) a sole L. 125.000 (imballaggio e spese spedizione comprese). Telefonare ore serali. Mauro Faccin - Via Giuseppe Cesare Abba - 17024 Finale Ligure - Tel. 019/690586.

Vero affare *svendo* computer Sharp MZ80K memoria 48K video registratore interfaccia stampante cassetta BASIC a sole L. 1.000.000. Antonio Attard - Via Riva del Garda, 27-3 - 39100 Bolzano - Tel. 0471/45470.

Vendo Sharp PC1500 + plotter 4 colori CE-150 + espansione 4K + trasformatore + manuali telefonare tra le ore 17 e le ore 19 L. 750.000 trattabili. Simone Besana - Via St. San Felice, 84/7 - 10025 Pino + SE. (TO) - Tel. 011/842730.

Vendo giochi per Sharp MZ-80 A, se siete interessati scrivete per richiedere lista a: John Ceresi - Via Mazzini - 18016 S. Bartolomeo Al Mare.

Intellivision + 18 cassette + tastiera Keiboard in garanzia vendesi a L. 850.000 - eventualmente anche separatamente. Telefonare dopo ore 20. Andrea Virgilli - Via Vetulonia, 1 - 40139 Bologna - Tel. 547547.

Acquisto-scambio-vendo programmi (utility-games) per computer Atari 400-800. Luigi Servolini - Via La Spezia, 81 - 00182 Roma - Tel. 06/384488 - 7581219.

Per Atari 400-800 *vendo-scambio* software originale americano utility-games solo su disco per VCS Atari vendo 24 cassette videogiochi a L. 800.000. Vittorio Di Paola - Via S. Dino Isol. 8, 18 - 80058 Torre Annunziata (NA) - Tel. 081/8614223.

CEDOLA DI COMMISSIONE LIBRARIA PERSONAL SOFTWARE

Da inviare a Gruppo Editoriale Jackson - Via Rosellini, 12 - 20124 Milano

Nome 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. .....

☐ Non abbonato ☐ Abbonato sconto 20% ☐ l'Elettronica ☐ Elettronica Oggi ☐ Automazione Oggi ☐ Elektor ☐ Informatica Oggi ☐ Computerworld ☐ Bit ☐ Personal Software ☐ Strumenti Musicali ☐ Videogiochi

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

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. .....

UNA PUBBLICAZIONE DEL GRUPPO EDITORIALE JACKSON

PERSONAL SOFTWARE

ANNO 3 N. 17 APRILE 1984

DIRETTORE RESPONSABILE: Giampietro Zanga

REDAZIONE: Lorenzo Barrile

COORDINAMENTO: Riccardo Paolillo

CONSULENZA: Giacomo Bortone

HANNO COLLABORATO A QUESTO NUMERO:
I. Parbuono, M. Gremes, C. Poma, A. Guida, A. De Santis, U. Barzaghi, A. Motta, F. Stella, R. Brunialti, F. Pozzi, R. Comini, M. Lenzi, S. Borsani, M. Spero.

UFFICIO ABBONAMENTI: Tel. 6880951/2/3/4/5

GRAFICA E IMPAGINAZIONE: Luigi Chiesa

USA: G.E.J. Publishing Group, Inc. a Subsidiary of - Gruppo Editoriale Jackson 1143 Quince Avenue - 94087 Sunnyvale (CA) Tel. (408) 7730103 - Telex: 4995972

GRAN BRETAGNA: G.E.J. Publishing L.t.d. 187 Oxford Street London WIR1AJ Tel. (01) 4392931 - Telex: (051) 21248

PUBBLICITA': 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

PUBBLICITA' PER USA E CANADA: International Media Marketing 16704 Marquardt Avenue, P.O. Box 1217 Cerritos (CA) 90701 - (213) 926-9552

FOTOCOMPOSIZIONE: LINEACOMP S.r.l. Via Rosellini, 12 - 20124 Milano

STAMPA: Reweba - Brescia

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

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

Spedizione in abbonamento postale Gruppo III/70
Prezzo della rivista L. 4.000. Numero arretrato L. 8.000.
Abbonamento annuo (10 numeri) L. 28.000; per l'Estero L. 44.800
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.

Il Gruppo Editoriale Jackson è iscritto nel registro Nazionale della stampa al n. 117 - Vol. 2 - Foglio 129 in data 17-8-1982

GRUPPO EDITORIALE JACKSON S.r.l.

DIREZIONE, REDAZIONE, AMMINISTRAZIONE: Via Rosellini, 12 - 20124 Milano - Telefoni: 68.80.951/2/3/4/5

SEDE LEGALE: Via G. Pozzone, 5 - 20121 Milano

DIREZIONE EDITORIALE: Giampietro Zanga e Paolo Reina

COORDINAMENTO EDITORIALE: Daniele Comboni

DIREZIONE AMMINISTRATIVA: Giuliano Di Chiano

ce l'hai la supergaranzia?

La Rebit Computer, distributrice per l'Italia dei prodotti SINCLAIR, ha messo a punto una nuova **sùpergaranzia** che ti darà i seguenti vantaggi:

1° Prezzo ridotto nell'acquisto dell'interfaccia programmabile.

2° Tessera sconto sull'acquisto dei programmi.

3° Tariffa ridotta per l'abbonamento a "Sperimentare con il Computer"

4° Libro sulle interfacce e sui microdrives.

Un risparmio di oltre 70.000 lire.

REBIT COMPUTER
A DIVISION OF G.B.C.

molto di più di una garanzia!


ENTE FIERE SCANDIANO (RE)
5ª MOSTRA
DELL'ELETTRONICA
E TELECOMUNICAZIONI
SCANDIANO (RE)
28 APRILE - 1 MAGGIO 1984
TELEFONO 0522/857436