Scroll

di Michele Sciabarra - Agrigento

La routine che vi presento permette di generare una scritta scorrevole PIXEL PER PIXEL che ha particolarità di essere visualizzata "sul bordo" o meglio, nella zona del video dove normalmente si trova il bordo, e dove apparentemente nulla è visualizzabile.

Dico apparentemente perché in realtà è possibile, con un trucco che reputo poco noto, «far sparire» letteralmente i due pezzi di bordo, sopra e sotto l'area testo.

In questa zona non è possibile visualizzare altro oltre gli sprite, eccetto una serie di linee verticali nere che sono, più che altro, un fastidio.

Tempo addietro mi è capitato tra le mani un giochino di origine pirata, non mi vergogno a dirlo. Si trattava di un dimostrativo del videogioco di un «cracker» che lo aveva programmato usando grafica e suoni «importati» da giochi commerciali.

Nella presentazione c'era uno schermo pieno di sprite (ben più di 8), e questo non era certo impossibile, ma ciò che mi lasciò letteralmente a bocca aperta erano delle scritte visualizzate sul bordo.

Fu così che, armato di disassembler, disassemblai il giochino, individuai la routine di raster interrupt responsabile di quel miracolo, e dopo 3-4 giorni passati allo studio di quella sfilza di istruzioni lm pochissimo comprensibili. Ecco le mie conclusioni.

Se si azzera il bit 4 della locazione 53265 (d'ora in poi ometterò di dire «della locazione» e dirò solo «di»), normalmente si ha il blanking del video. Beh, succede che se si attiva il blanking del video quando il raster video è alla linea di scansione 250 ($fa), che è proprio la prima linea del bordo inferiore in condizioni normali (ossia quando lo scrolling fine verticale del video, cioè i bit 2, 1, 0 di 53265 sono 011, cioè ancora lo schermo è centrato verticalmente), il VIC si «dimentica» di disegnare il bordo da quel punto in poi!

Non so perché succeda. Posso solo immaginare che in qualche modo si blocchi il disegno del bordo, lasciando così che il resto del VIC disegni gli sprite. Quello che so è che è necessario riattivare il video PRIMA della linea di scansione $33 (51, la prima del video visualizzabile). Così facendo, con una opportuna routine di interrupt, si ottiene l'area visualizzabile allungata verticalmente a tutto lo schermo, dello stesso colore dello sfondo e libera per sprite vari. Per capire meglio, consideriamo il C64 in condizioni standard (appena acceso) in modo che i registri del VIC siano in condizioni note. Digitate

POKE 53265,31

per usare la possibilità di scrolling fine del VIC, ed abbassare il video di alcuni pixel. Poiché non lo abbiamo fatto coprire dal bordo col modo 24 righe, lo schermo presenta nella parte più alta una fila di locazioni che non fa parte del video e dove è possibile visualizzare caratteri.

Ve ne potete accorgere meglio spostandovi con il cursore in alto. Noterete che tra il quadratino lampeggiante e il bordo c'è un certo spazio ed il cursore non tocca il bordo, come normalmente succede.

In questa striscia i casi sono due: o non c'è niente, ossia c'è lo stesso colore dello sfondo, oppure ci sono delle righine verticali nere (al limite la zona è tutta nera). Questo fatto dipende dal contenuto di un byte, e precisamente l'ultimo del banco di memoria che attualmente il VIC «vede». In altre parole, nelle condizioni in cui adesso ci troviamo, dalla locazione 16383 ($3fff).

Provate ora a dare qualche POKE 16383, n. Vedrete che, a seconda del valore inserito le righine nere cambiano. Precisamente, come è facile intuire, il VIC ripete in ogni riga lasciata libera ogni otto pixel l'immagine dei corrispondenti bit della locazione in questione.

In particolare a noi interessa mettere 0 in questa locazione: infatti, facendo sparire il bordo nel modo che ho appena detto, il fenomeno delle righine nere si ripete anche lì.

Ovviamente l'uso più logico per una cosa del genere è un marchio o una scritta scorrevole. Ed è proprio quest'ultima che ho realizzato.

Il programma, in Assembler, attiva una routine in IRQ che visualizza in basso una scritta scorrevole, utilizzando sette sprite (da 0 a 6) espansi in x.

La routine è locata, con un enorme sforzo di fantasia, a $C000, si attiva ovviamente con SYS 49152 e si disattiva con SYS 49155. Poiché ha bisogno di ridefinire degli sprite si riserva per sprite e puntatori ad essi la zona da $c400 a $c7ff.

Da $c800 in poi c'è la scritta scorrevole, memorizzata come sequenza di codici schermo (un ottimo modo di generare la scritta è di scriverla sul video con delle print e poi ricopiare lo schermo da $c800 in poi).

La scritta è terminata da un valore 255, che pertanto è l'unico che non può essere visualizzato.

Un paio di note sulla routine: è in interrupt, e richiama anche la routine

Nota

I codici di controllo nei listati sono riportati in forma «esplicita», in conseguenza dell'impiego della stampante Star NL-10 e relativa interfaccia per Commodore. Ovviamente, nella digitazione del programma è necessario usare i consueti tasti che corrispondono alle indicazioni fra parentesi: ad esempio cursore destro per (RGHT), CTRL-3 per (RED) eccetera.

(CLR)	=	(YEL)	=
(HOME)	=	(RVS)	=
(DOWN)	=	(OFF)	=
(UP)	=	(ORNG)	=
(RGHT)	=	(BRN)	=
(LEFT)	=	(LRED)	=
(BLK)	=	(GRY1)	=
(WHT)	=	(GRY2)	=
(RED)	=	(LGRN)	=
(CYN)	=	(LBLU)	=
(PUR)	=	(GRY3)	=
(GRN)	=	(SWLC)	=
(BLU)	=		

Scroll

```
10 REM ---------------------------------
20 REM SCRITTA SCORREVOLE SUL BORDO
30 REM DELLO SCHERMO DEL COMPUTER NELLA
40 REM PARTE NORMALMENTE NON UTILIZZATA
50 REM DAL TESTO
60 REM ---------------------------------
70 REM
80 FOR I=49152 TO 49758
90 READ A:POKE I,A:NEXT
100 REM
110 REM   CARICHIAMO LA STRINGA 'CIAO'
120 REM   A PARTIRE DA $C800 (DEC51200)
130 REM
140 POKE 51200,ASC("C")
150 POKE 51201,ASC("I")
160 POKE 51202,ASC("A")
170 POKE 51203,ASC("O")
180 POKE 51204,ASC(" ")
190 POKE 51205,255
200 REM
210 SYS 49152
49152 DATA  76, 6,192, 76, 48,194,120,173, 20, 3,141, 97,194,173, 21, 3
49168 DATA 141, 98,194,173, 14,220,141, 99,194,169,157,141, 20, 3,169,192
49184 DATA 141, 21, 3,162, 0,142, 14,220,232,142, 26,208,169,250,141, 18
49200 DATA 208,173, 17,208, 41,127,141,100,194,141, 17,208,169,127,141, 21
49216 DATA 208,141, 29,208,169, 0,141, 23,208,141, 28,208,169, 10,141, 1
49232 DATA 208,141, 3,208,141, 5,208,141, 7,208,141, 9,208,141, 11,208
49248 DATA 141, 13,208,160, 0,162, 16,138,153,248,199,169, 7,153, 39,208
49264 DATA 232,200,192, 8,208,241,169, 18,141,103,194,169, 0,141,255,255
49280 DATA 141, 95,194,141, 96,194,170,157, 0,196,157, 0,197,202,208,247
49296 DATA 173, 0,221,141,101,194,173, 24,208,141,102,194, 96,173, 13,220
49312 DATA 173, 25,208,240, 33,141, 25,208,169, 0,141, 17,208,169,148,141
49328 DATA  0,221,169, 16,141, 24,208,169, 32,141, 18,208,169,204,141, 20
49344 DATA  3,169,192,141, 21, 3,104,168,104,170,104, 64,173, 13,220,173
49360 DATA  25,208,240,100,141, 25,208,173,255, 63, 72,169, 0,141,255, 63
49376 DATA 173,100,194,141, 17,208,173,102,194,141, 24,208,173,101,194,141
49392 DATA  0,221,169,250,141, 18,208,169,157,141, 20, 3,169,192,141, 21
49408 DATA  3,206,103,194,206,103,194,173,103,194,201, 8,240, 17, 32,245
49424 DATA 193,169, 51,205, 18,208,208,251,104,141,255, 63, 76, 49,234,169
49440 DATA  24,141,103,194,165, 1, 72,169, 51,133, 1, 32, 62,193,104,133
49456 DATA  1, 32,245,193,104,141,255, 63,104,168,104,170,104, 64, 32,135
49472 DATA 193,173, 95,194, 72,173, 96,194, 72,160, 6,140,106,194,169, 0
49488 DATA 133,251,169,196,133,252, 32,135,193,160, 0, 32,182,193, 32,135
49504 DATA 193,160, 1, 32,182,193, 32,135,193,160, 2, 32,182,193, 24,165
49520 DATA 251,105, 64,133,251,144, 2,230,252,206,106,194, 16,216,104,141
49536 DATA  96,194,104,141, 95,194, 96, 24,169, 0,109, 95,194,133,253,169
49552 DATA 200,109, 96,194,133,254,160, 0,177,253, 72,200,177,253,201,255
49568 DATA 208, 10,169, 0,141, 95,194,141, 96,194,104, 96,238, 95,194,208
49584 DATA  3,238, 96,194,104, 96,133,253,169, 0,133,254, 24, 6,253, 38
49600 DATA 254, 6,253, 38,254, 6,253, 38,254,165,254, 9,216,133,254,169
49616 DATA  0,141,104,194,140,105,194,172,104,194,177,253,172,105,194,145
49632 DATA 251,238,104,194,238,105,194,238,105,194,238,105,194,173,104,194
49648 DATA 201, 8,208,227, 96,169, 0,170,168,141, 16,208,133,254,173,103
49664 DATA 194,133,253,165,253,153, 0,208,165,254,240, 9,189, 40,194, 13
49680 DATA  16,208,141, 16,208, 24,165,253,105, 48,133,253,144, 2,230,254
49696 DATA 200,200,232,224, 7,208,220, 96, 1, 2, 4, 8, 16, 32, 64,128
49712 DATA 120,169, 0,141, 21,208,141, 26,208,173, 97,194,141, 20, 3,173
49728 DATA  98,194,141, 21, 3,173, 99,194,141, 14,220,173,100,194,141, 17
49744 DATA 208,173,101,194,141, 0,221,173,102,194,141, 24,208, 88, 96, 0
```

standard, anche se il sincronismo di interrupt adesso è generato dal VIC.

Inoltre poiché una volta ogni quattro interrupt la routine standard non viene chiamata, io non mi fiderei tanto delle TI$ e TI. Inoltre, poiché il VIC vede soltanto 16K alla volta, non potendo mettere gli sprite nei primi 16K per non lasciare troppo poco spazio per i programmi Basic, ho risolto selezionando il banco 3 (da $c000 a $ffff) ogni volta che il raster ha superato la linea di raster $0fa e rimetto tutto a posto quando il raster arriva a $020.

Ora, poiché il banco video ritorna ad essere il numero zero, appaiono nelle righe di raster tra la $020 e la $033, la prima dell'area visualizzabile. Compaiono le «simpatiche» righine verticali nere, in quanto immagine non più di $ffff ma di $3fff, che presumibilmente è proprio nel bel mezzo del programma Basic, se è abbastanza lungo. Non possiamo rimettere il video a posto dopo $032 perché, se no, la routine di scrolling degli sprite impiega troppo tempo e, quando finisce, il raster ha già superato il valore per il prossimo interrupt, perde un sincronismo di quadro e la scritta si mette a lampeggiare. Soluzione: salviamo la locazione $3fff per un momento, la azzeriamo, aspettiamo che il resto della routine finisca il suo lavoro e poi rimettiamo tutto a posto. A proposito: non usate le locazioni in pagina zero da $fb a $fe perché la routine ne fa un uso intenso (e non le rimette a posto...).

Il programma è scritto in Assembler standard, non usa macro e solo una direttiva .BYT per ovvi motivi.

Labyrinth

di Ciro d'Urso, Roma

Metodi di sicurezza

Cos'è l'intelligenza? Efficienza, creatività, originalità quindi: capacità generalizzatrice, autocontrollo consapevole. Ma quando si può dire che una unità (biologica o meccanica) è intelligente? O meglio, quando un determinato comportamento si può definire intelligente?

Noi ogni giorno abbiamo a che fare con l'intelligenza e ne riconosciamo intuitivamente gli effetto e, del resto, riusciamo a distinguere tra comportamento sostanzialmente intelligente e apparentemente intelligente, per il quale, una volta scoperte le regole, le leggi, le istruzioni che lo determinano, non parliamo più di intelligenza ma di qualità molto più concrete quali abilità o destrezza. Però troviamo difficile intenderci in modo preciso su ciò che intuitivamente riconosciamo. Il termine «intelligenza» è così evanescente e indeterminato che crea infinite interpretazioni senza peraltro poterle giustificare.

Non è in questa sede che intendiamo dare una definizione inconfutabile e operativa di «intelligenza», ma vogliamo solo mostrare una particolare «strategia di ricerca» che potrebbe avere a che fare (almeno lontanamente) con questo concento così sfuggevole.

Supponiamo di liberare tre cavie in un labirinto (in una zona del quale sia stato collocato del cibo), e supponiamo che ogni cavia si comporti diversamente dalle altre.

I tre animali si dedicheranno alla ricerca del cibo con strategie differenti e una di loro, quella che avrà adottato la strategia con il grado più alto di efficienza, si nutrirà per prima.

Possiamo supporre che una delle tre cavie proceda casualmente per il labirinto e casualmente prenda le sue decisioni; la seconda cavia potrebbe procedere in modo sistematico, cioè, ad esempio, ad ogni bivio deviare a destra; la terza, invece, terrà conto della variazione dell'intensità di un particolare fattore, come l'odore del cibo, che la guiderà nel percorso e le permetterà di nutrirsi. Credo che saremo tutti d'accordo nel reputare il comportamento della terza cavia come quello più «intelligente», e il programma presentato cerca di simulare un comportamento assimilabile a quello considerato ma, eliminato il fattore «odore» ne considera altri.

Il programma

Le righe 3 e 4 inizializzano alcune variabili.

Dalla riga 5 alla riga 42 viene eseguita la routine di «formazione del labirin-

Labyrinth

```
0 REM *** LABYRINTH ***
1 REM ***     DI      ***
2 REM *** C. D'URSO ***
3 REM ******************
4 N=5:M=8:NF=N:MF=M:PRINT"(CLR)":DIMC(40,25),BX(300),BY(300)
5 X=N:Y=M:P=1:FL=0:POKE53280,0:POKE53281,0
6 INPUT"X - MAX";NN
7 INPUT"Y - MAX";MM
8 IFNN>38ORMM>23THEN2
9 REM *** FORMAZIONE BORDI ***
10 PRINT"(CLR)":FORR=1TONN
11 POKE1024+R+40,224
12 POKE1024+R+40*MM,224
13 C(R,1)=0:C(R,MM)=0
14 NEXTR
15 FORRR=1TOMM
16 POKE1025+40*RR,224
18 POKE1024+NN+40*RR,224
20 C(1,RR)=0:C(NN,RR)=0
22 NEXTRR
24 FORZ=2TONN-1:FORZZ=2TOMM-1
25 C(Z,ZZ)=1
26 NEXTZZ:NEXTZ
27 REM *** FORMAZIONE GRIGLIA ***
28 REM ***************************
29 X=2:Y=2:CA=42
30 GETA$:FX=X:FY=Y
32 POKE1024+X+40*Y,CA
33 C(X,Y)=CA:IFCA=224THENC(X,Y)=0
34 IFCA=42THENC(X,Y)=1
35 IFA$="(UP)"THENGOSUB2000:Y=Y-1:GOSUB2050
36 IFA$="(DOWN)"THENGOSUB2000:Y=Y+1:GOSUB2050
37 IFA$="(RGHT)"THENGOSUB2000:X=X+1:GOSUB2050
38 IFA$="(LEFT)"THENGOSUB2000:X=X-1:GOSUB2050
39 IFA$=" "THENCA=224
40 IFA$="B"THENCA=2
41 IFA$="(F1)"THEN47
42 GOTO30
44 REM ****************************
45 REM *** PROGRAMMA DI RICERCA ***
46 REM ****************************
47 TI$="000000"
50 N=X:M=Y:IFC(X,Y)=2THENPRINT"HO TROVATO":END
55 GOSUB500
56 PRINT"(HOME)"TAB(15)"(RVS)TIME(OFF) "LEFT$(TI$,2)" "MID$(TI$,3,2)" "RIGHT$(TI
$,2)
60 IFC(X,Y-1)=1ORC(X,Y-1)=2THENNO=1:GOTO62
61 NO=0
62 IFC(X,Y+1)=1ORC(X,Y+1)=2THENS=1:GOTO65
63 S=0
65 IFC(X-1,Y)=1ORC(X-1,Y)=2THENW=1:GOTO67
66 W=0
67 IFC(X+1,Y)=1ORC(X+1,Y)=2THENE=1:GOTO69
68 E=0
69 BI=0
70 IFNO=1AND(E=1ORS=1ORW=1)THENBI=1
72 IFS=1AND(NO=1ORE=1ORW=1)THENBI=1
74 IFW=1AND(NO=1ORE=1ORS=1)THENBI=1
76 IFE=1AND(NO=1ORW=1ORS=1)THENBI=1
78 IFBI=1THENBX(P)=X:BY(P)=Y:P=P+1
80 C(X,Y)=4
82 IFC(X,Y-1)=1ORC(X,Y-1)=2THENY=Y-1:GOSUB1000:GOTO50
84 IFC(X,Y+1)=1ORC(X,Y+1)=2THENY=Y+1:GOSUB1000:GOTO50
86 IFC(X-1,Y)=1ORC(X-1,Y)=2THENX=X-1:GOSUB1000:GOTO50
88 IFC(X+1,Y)=1ORC(X+1,Y)=2THENX=X+1:GOSUB1000:GOTO50
90 IFP-1>0THENP=P-1:X=BX(P):Y=BY(P)
92 GOSUB1000:GOSUB500:N=X:M=Y:IFC(X,Y)=2THENPRINT"HO TROVATO":END
105 GOTO50
498 REM *** VISUALIZZAZIONE UNITA' ***
499 REM *******************************
500 POKE1024+X+40*Y,62:FORU=0TO100:POKE1024+X+40*Y,96:C(X,Y)=4:RETURN
998 REM *** CONTROLLO COORDINATE ***
999 REM *******************************
1000 IFX<1ORX>NNTHENX=N
1010 IFY<1ORY>MMTHENY=M
1020 RETURN
1998 REM *** MOVIMENTO CURSORE ***
1999 REM ************************
2000 IFCA<>42THENRETURN
2010 POKE1024+X+40*Y,32
2020 RETURN
2050 L=0:IFX>NN-1ORX<2THENX=FX:L=1
2060 IFY>MM-1ORY<2THENY=FY:L=1
2065 IFL=0THENCA=42
2070 RETURN

READY.
```

to»: inserite le dimensioni desiderate (per un massimo di 38x23), l'utente provvede a formare i corridoi, nei quali si dovrà muovere l'unità di ricerca, collocando i «mattoni» delle pareti (gli spazi pieni), e una «B» che rappresenta la meta da raggiungere.

I tasti da usare sono quelli del cursore per spostarsi all'interno del labirinto; lo spazio forma le pareti; «F1» viene utilizzato al termine della formazione per dare inizio alla ricerca.

(Attenzione: il tasto «F1» deve essere premuto quando la «*» è visualizzata).

Se si compie un errore basta spostare il cursore sulla cella da modificare e digitale il carattere desiderato.

Importante: non dimenticare di collocare la «B» in una posizione raggiungibile, altrimenti il programma, dopo aver esplorato il labirinto interamente, si bloccherà.

Per un perfetto funzionamento del programma consiglio di formare corridoi di una sola cella.

Dalla riga 50 in poi si sviluppa il programma principale con le relative subroutine.

Come prima cosa vengono esaminate le posizioni adiacenti a quella attuale e, qualora si presenti un incrocio, le coordinate della posizione vengono memorizzate nelle matrici BX() e BY(). In questo modo se un determinato percorso risulta cieco il computer può ritornare all'incrocio passato per ultimo e scegliere una nuova strada.

L'utilizzo dei dati contenuti nelle 2 matrici è assimilabile a quello dello STACK: il valore memorizzato per ultimo è il primo ad essere prelevato.

Le righe dalla 60 alla 78 prendono nota degli incroci; la 80 provvede ad assegnare il valore «4» alle celle già esaminate; quelle dalla 82 alla 92, con le relative subroutine, controllano la visualizzazione dell'unità di ricerca.

Probabilmente noterete dei salti di locazione improvvisi: per evidente risparmio di tempo (quando il labirinto inizia ad essere superiore a certe dimensioni, il tempo di ricerca diviene inevitabilmente lungo) l'unità di ricerca giunge all'ultimo bivio incontrato spesso senza ritornare sui suoi passi.

Naturalmente i più pignoli di voi potranno provare a stilare quelle poche e semplici righe che provvederanno al controllo del movimento di ritorno dell'unità. È facile notare che la strategia adottata è ragionevolmente efficace ma non efficiente in quanto l'unità

esplora (quindi percorre) inevitabilmente dei vicoli ciechi e in questo modo si ha un notevole dispendio di tempo, in un problema come questo (cioè l'esplorazione di un labirinto) l'efficienza risulta determinata da un fattore casuale in quanto dovremo fornire prima la mappa del labirinto al computer per pretendere che scelga il percorso più economico e riesca quindi a raggiungere la meta nel minor tempo possibile.

Qualora, quindi, il calcolatore non sia in possesso di una mappa da esaminare dovrà esplorare il labirinto completamente così da crearsela e scegliere poi (alla luce di determinati parametri come, ad esempio, lunghezza del percorso e tempo impiegato) la strada più economica da percorrere in un secondo momento.

Il programma presentato risulta, in questo senso, essere un soddisfacente inizio per raggiungere l'efficienza.

E ora bando alle chiacchiere e provate a sfidare il computer.

New Flashtape

di Giuliano Freiles - Gaeta (LT)

Questo programma non è altri che una modifica del Flashtape di Alessandro Guida e Gianni Iotta pubblicato su MC n. 50/51 nella rubrica VIC da zero + 64 e risolve un piccolo bug del programma. Ho notato che ogni volta che si batte la sequenza RUN STOP + RESTORE il 64, resettando i valori della Jump Table, resetta ovviamente anche il vettore della routine di SAVE, con la conseguenza che diventa impossibile salvare programmi in formato Flash; per ovviare a questo inconveniente ho inserito nel Flashtape una piccola routine in LM che riaggancia la routine di Save in formato Flash modificando il valore del vettore di SAVE. Per fare ciò ho collocato da $C000 a $C01F la mia routine, ed ho spostato la routine di interfaccia dal buffer del registratore alle locazioni $C020 - $C057. La routine, oltre a riagganciare il vettore, visualizza l'avvenuto riaggancio tramite un opportuno messaggio; qualora tale messaggio non apparisse, vorrà dire che si è pasticciato nell'aria di memoria $C000 - $C057 e quindi si dovrà ricaricare il Flashtape di nuovo, sempre che nel frattempo il 64 non si sia inchiodato. Il programma così com'è allocato non interferisce col Turbo Tape; è perciò possibile salvare in Flash programmi precedentemente salvati col Turbo; a tal proposito risulta utile caricare in formato Flash il Turbo stesso, che verrà caricato in soli 3 giri contro i normali 17.

Dati tecnici

Siccome il programma è stato allungato di 32 byte rispetto all'originale, per caricare in memoria il caricatore Basic occorre prima digitare POKE 44,17: POKE 4352,0: NEW e dare il RETURN per spostare l'area programmi. Una volta salvato il file dati in LM in formato Flash e con l'opzione R, basterà caricarlo ed eseguirlo: il 64 visualizzerà un messaggio di copyright (un po' differente dall'originale) e si aggancerà alla routine di SAVE. Per riagganciare il Flash dopo un RUN STOP + RESTORE basterà digitare SYS 49152; se tutto è a posto il 64 visualizzerà il messaggio di aggancio effettuato. Il programma viene memorizzato nell'area di memoria tra $C000 e $C057; tale area è perciò tabù per le POKE. Il programma vero e proprio è stato modificato ai salti posti in $E049, $E05F ed $E150 per seguire lo spostamento della routine di interfaccia che contiene la routine ERROR, VALGET e SAVEND. La routine vera e propria è invece sempre posta nella RAM presente a partire da $E000.

Dissasemblato routine

```
C000    LDA #$20
C002    STA $0332
C005    LDA #$C0
C007    STA $0333
C00A    LDY #$00
C00C    LDA $C018,Y
C00F    BNE $C012
C011    RTS
C012    JSR$FFD2
C015    INY
C016    BNE $C00C
C018    BYT
$46.$4C.$41.$53.$48.$23.$0D.$00.
```

Bibliografia

D. Lawrence - M. England / LM del Commodore 64 / Ed. Jackson; MC nn. 50/51 / Rubrica: VIC da zero +64; Commodore User's Guide

La logica de «Le Torri di Hanoi»

di Ciro D'Urso - Roma

Non è ormai una novità la vicenda che si racconta intorno ad un monastero nei pressi di Hanoi nel quale i monaci sono intenti a svolgere un compito affidato loro da una ipotetica creatura soprannaturale.

Essi hanno nel loro giardino tre pioli di uguale altezza, di cui uno fa da sostegno a 64 dischi tutti di diverso diametro formanti una pila e ordinati in modo decrescente (dal basso verso l'alto).

Ora il compito dei monaci è quello di spostare la pila dei dischi in un altro piolo rispettando le seguenti regole:

— è lecito spostare un disco alla volta;
— un disco di grandezza maggiore non può essere mai collocato su uno di grandezza minore;
— è possibile usare il 3° piolo (quello interessato non definitivamente dal passaggio dei dischi) come temporaneo deposito degli stessi.

Quando i nostri monaci avranno terminato lo spostamento dei 64 dischi potremmo dire addio a questo caro e amato mondo in quanto ne è prevista la fine.

Il programma presentato provvede a spostare un numero N di dischi dalla base (o piolo) numero 1 a quella numero 2.

Credo che sia opportuno esporvi quale strada è stata seguita nella risoluzione del problema, e come si sia potuto giungere ad un livello di semplificazione della procedura ragionevolmente soddisfacente per la sua traduzione in Basic.

Infine osservo che volutamente non ho usato avanzate procedure ricorsive.

Le basi sono anche identificate con le lettere A, B, C.

Dopo un esame non necessariamente accurato del problema si perviene a questa conclusione:
per formare una pila di N elementi nella base numero 2 bisogna formare una pila di (N-1) elementi nella base numero 3, ed ancora (N-2) elementi nella base numero 2, e così via.

Si riscontra facilmente un andamento ciclico del numero delle basi in cui si devono formare le pile: quindi procedendo dal risultato finale alla mossa iniziale così si presenta la successione delle basi: 2-3-2-3... Ora il primo problema è questo: come determinare la mossa inziale? Ovvero, in quale pila (2 o 3) andrà posizionato il primo elemento? Se il numero degli

elementi è pari si comincerà posizionando il primo elemento nella base 3 (base diversa da quella in cui si vuole terminare la costruzione della pila); se dispari nella base 2.

Il secondo problema che risulta da questo modo di procedere è più complesso: come costruire le pile successivamente nelle diverse basi?

Intuitivamente la soluzione è da ricercare in una sequenza che si ripeta periodicamente e che si riscontri «indipendentemente» dal numero degli elementi.

Un primo e significativo risultato lo si raggiunse quando, esaminando la soluzione trovata per 3, 4 elementi, si riscontrò un periodico passaggio degli elementi da una base ad un'altra con ulteriori periodiche alternanze delle basi. In sostanza la periodicità riscontrata era questa: si presentava un nucleo fondamentale di 3 coppie di basi interessate a diversi passaggi (A-B; A-C; B-C) all'interno del quale le basi erano suscettibili di cambiamenti di stato (da sorgente a ricevente) B→A; C→A; B→C * A→B; C→A; C→B.

Ma le alternanze delle basi erano troppo irregolari per potere arrischiare una semplice legge generale.

Il risultato definitivo fu ottenuto riuscendo a capire che le espressioni di scambio che dovevano formare l'algoritmo risolutivo avrebbero dovuto distinguere le basi non definitivamente come 1°, 2°, 3°, ma in modo variabile come base originaria, ausiliaria, fondamentale.

E tutto questo doveva avvenire in una procedura ricorsiva che prevedesse l'assegnazione dello specifico carattere della base ad ognuna delle 3, periodicamente. Considerando come base Originaria quella che contiene gli elementi da prelevare, come base Fondamentale quella che avrebbe dovuto contenere la pila degli elementi prelevati, e come base Ausiliaria quella in cui gli elementi avrebbero dovuto soltanto transitare, il risultato fu questo:

	B. 1	B. 2	B. 3	
1° passaggio	PO	PA	PF	**P**ila **O**riginaria
2° passaggio	PA	PF	PO	**P**ila **A**usiliaria
3° passaggio	PF	PO	PA	**P**ila **F**ondamentale

Da cui al termine di ogni passaggio (che a sua volta contiene 4 istruzioni di scambio): PF=PA; PA=PO; PO=PF.

Invece le istruzioni fondamentali di scambio dei dischi risultarono essere PA←PO; PF←PO; PF←PA; PA←→PO; il cui significato è questo: il primo elemento di PO deve passare in PA; il «nuovo primo» di PO deve passare in PF, e così fino all'ultima istruzione il cui enunciato è: se l'elemento di PO è maggiore di quello di PA allora quello di PA passa in PO, altrimenti il contrario (si ricordi che per definizione due elementi non possono risultare uguali). Concludo facendo solo una modesta osservazione: se fosse stata la macchina a procedere in questo modo cosa avreste pensato?

Voglio dire: se, dopo averle comunicato tutti i dati necessari affinché potesse formulare una precisa configurazione del problema, e poi avesse proceduto nel modo ora esposto per la sua soluzione (che, del resto, non è soddisfacentemente efficiente), ebbene cosa sareste stati portati a concederle: solo «meccanica» capacità risolutiva; elevata abilità nella soluzione di problemi di questo tipo; sorprendente familiarità con processi affini all'induzione completa; capacità astrattiva e padronanza dell'analogia come facoltà intelligente? MC

Descrizione del listato

Non sono necessari particolari commenti. Facciamo comunque notare le righe:
25 - valutazione della prima mossa.
115-125 visualizzazione contenuto delle pile.
1000-1040 1200-2050 disposizione ed etichettatura degli elementi delle pile.
1910-2050 1200-2050 istruzioni di scambio con relativo ordinamento delle pile interessate.

Elenco variabili

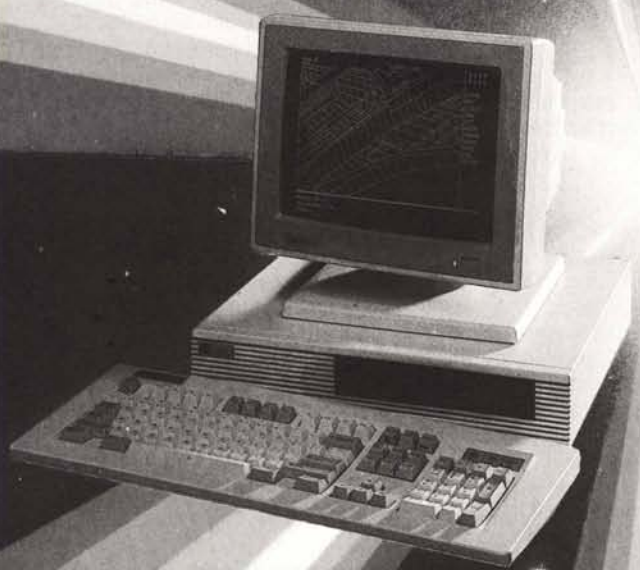
PP(P,N) - matrice degli elementi delle pile
C() - numero degli elementi di una pila
PO - variabili di servizio indicanti il numero della pila
PA - che assume il par
PF - titolare carattere
FF - variabile della funzione di ordinamento degli elementi

Le torri di Hanoi

```
 0 REM*** LE TORRI DI HANOI ***
 1 REM***         DI          ***
 2 REM***      C.   D'URSO    ***
 3 REM*************************
 4 REM
 5 POKE53280,0:POKE53281,0
 6 CLR:PRINT"(CLR)":DIMPP(20,20)
 10 INPUT"NUMERO ELEMENTI";N
 15 J=1:C(1)=N:C(2)=0:C(3)=0
 20 FORR=1TON:PP(1,R)=R:NEXTR
 25 IF(N/2)<>INT(N/2)THENPF=3:PA=2:GOTO35
 30 PF=2:PA=3
 35 PO=1
 37 GOSUB1100:GOSUB115
 40 FORT=1TO(M+1)/4
 50 GOSUB1900
 90 KF=PF
 100 PF=PA:PA=PO:PO=KF
 110 NEXTT
 112 END
 115 FORJ=1TO3
 120 FORRP=1TON:PRINTPP(J,RP);:NEXTRP
 122 PRINT"PILA"J:NEXTJ:PRINT
 123 POKE198,0:WAIT 198,1
 125 RETURN
 500 END
 1000 IFC(FF)<1THEN:RETURN
 1010 FORTT=C(FF)TO1STEP-1
 1020 PP(FF,TT+1)=PP(FF,TT)
 1030 NEXTTT
 1040 RETURN
 1100 M=1
 1110 FORRR=1TON-1
 1120 M=(M*2)+1
 1130 NEXTRR
 1140 RETURN
 1200 IFC(FF)<1THENRETURN
 1210 FORR=1TOC(FF)
 1220 PP(FF,R)=PP(FF,R+1)
 1230 NEXTR
 1240 RETURN
 1900 FF=PA:GOSUB1000
 1910 PP(PA,1)=PP(PO,1):C(PA)=C(PA)+1
 1915 FF=PO:GOSUB1200:C(PO)=C(PO)-1:GOSUB115
 1920 FF=PF:GOSUB1000
 1925 PP(PF,1)=PP(PO,1):C(PF)=C(PF)+1
 1930 FF=PO:GOSUB1200:C(PO)=C(PO)-1:GOSUB115
 1940 FF=PF:GOSUB1000
 1945 PP(PF,1)=PP(PA,1):C(PF)=C(PF)+1
 1950 FF=PA:GOSUB1200:C(PA)=C(PA)-1:GOSUB115
 1955 IFPP(PA,1)=0THEN2010
 1960 IFPP(PO,1)=0THEN2020
 2000 IFPP(PO,1)>PP(PA,1)THEN2020
 2010 FF=PA:GOSUB1000:PP(PA,1)=PP(PO,1):C(PA)=C(PA)+1
 2015 FF=PO:GOSUB1200:C(PO)=C(PO)-1:GOSUB115:RETURN
 2020 FF=PO:GOSUB1000:PP(PO,1)=PP(PA,1):C(PO)=C(PO)+1
 2025 FF=PA:GOSUB1200:C(PA)=C(PA)-1:GOSUB115
 2050 RETURN
```

power & compatibility

PERSONAL WORK STATION 16 e 32 BIT

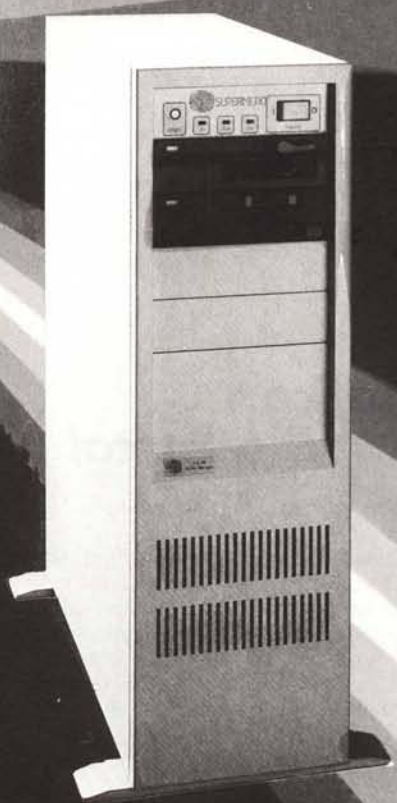


PX-30
Cpu 8088 10MHz, 256-640K ram, floppydisk 3,5 pollici, hard disk 20-40MB

PX-50
Cpu 80286 8MHz, 512K-1MB ram, floppy disk 3,5 pollici, hard disk 20-40MB

PX-80
Cpu 32 bit 80386 16MHz, 2MB ram, floppy disk 3,5 pollici, hard disk 20-40MB

SUPERMICRO 16 e 32 BIT



AX-60
Cpu 16 bit 80286 12MHz, 512K-2MB ram, floppy disk 5,25 e 3,5 pollici, hard disk 40-230MB

AX-80
Cpu 32 bit 80386 16MHz, 2MB ram, floppy disk 5,25 e 3,5 pollici, hard disk 40-230MB

IMMAGINERIA

UNIDATA

UNIDATA - Via S. Damaso 20 - 00165 Roma

Tel. 06/63.66.58 - Telex 623122 UNITAT I