a cura di Tommaso Pantuso

Qualche numero fa abbiamo parlato di labirinti e, stimolato, un lettore ci ha inviato un programma in Basic per la loro generazione che pubblichiamo ritenendo interessante il metodo utilizzato. Il dischetto inviato contiene un'elaborazione più complessa in linguaggio macchina del programma sotto forma di gioco. In un primo momento ho generato un caricatore Basic, ma, vista la grande quantità di linee DATA (circa 160 blocchi di programma su disco!), non era proponibile la pubblicazione. Conclusione: tra queste pagine troverete un programma in Basic per la generazione ed eventuale stampa del labirinto; per... avere un po' di più potrete sempre richiedere il dischetto completo in redazione.

A quanto pare i nostri lettori non si stancano mai di costruire tool per migliorare il Basic del C64. Quello che pubblichiamo è molto sintetico e vi assicuro che vi sarà di grande utilità nella programmazione. Le linee DATA del caricatore Basic sono in numero inaccettabile e perciò non pubblichiamo il relativo listato. Prima di concludere aggiungo alcune brevi note riguardanti il materiale che inviate per la pubblicazione.

Gli articoli dovrebbero essere per quanto possibile scritti a macchina (o stampati con una normale stampante ad aghi). Puntualizzo ciò perché molti inviano articoli scritti a mano che prendere in considerazione conduce ad una inevitabile perdita di tempo.

Inviate sempre una copia del programma (anche se breve) su supporto magnetico perché non è pensabile di dover mettere in memoria i programmi da esaminare mediante la... tastiera.

Se inviate dei programmi in linguaggio macchina, accludete SEMPRE un caricatore Basic perché, oltre a risparmiarci il tempo della traduzione, avrete più probabilità che siano pubblicati

Mot Mot Labyrinth

di Roberto Larcher e Alessandro Corazzin
Montebelluna (TV)

Primo tempo

Il punto di partenza del nostro gioco è la generazione casuale di un labirinto, ovvero il poter creare di volta in volta uno schema diverso.

Abbiamo ritenuto necessario mantenere fisse alcune regole:

1) La forma è rigorosamente rettangolare e le dimensioni sono costanti.
2) Il tracciato risolutivo è unico.
3) Il punto di partenza ed il punto di arrivo sono fissi.

Su queste basi realizziamo un algoritmo, che segue un'idea, almeno sulla carta, semplice. Supponiamo di avere uno spazio, in due dimensioni, con un punto A (di entrata) ed un punto B (di uscita). Da A realizziamo un percorso; questo sarà rappresentato da una linea spezzata casuale che non si interseche-rà, né avrà punti morti, punti cioè dai quali non potrà uscire; l'altro estremo della spezzata, che più che una linea è un corridoio, lo chiameremo C (fig. 1).

Dal punto di uscita B realizziamo una linea simile alla precedente, non completamente casuale, e rivolta alla RICERCA di C. Trovato questo punto, facciamo avvenire il meccanismo di UNCINAZIONE, ovvero il raccordo tra i due rami del percorso di soluzione (fig. 2).

Questo secondo passaggio può sollevare qualche obiezione. Perché infatti non passare da A a B direttamente? La risposta risiede nel vantaggio ottenuto dalla maggiore imprevedibilità della spezzata, senza che sia particolarmente oneroso il calcolo.

Ripercorriamo ora il tracciato aprendo ogni tanto delle FINESTRE, dalle quali nascono dei rami che sono delle ulteriori spezzate; il Processo viene ripetuto riempiendo prima la zona di destra e poi quella di sinistra (fig. 3). Infine FILLER!! è il semplice riempimento dello spazio rimanente: si completa così il labirinto (fig. 4).

Questo in sintesi è il meccanismo logico con cui viene generato il labirinto. Ma tradurlo in byte risulta cosa molto più complessa. Procediamo dunque con ordine e stendiamo quanto detto finora in Basic. Data la relativa complessità del programma di cui diamo copia per noi è piuttosto arduo dare una dettagliata spiegazione delle quasi 300 linee...

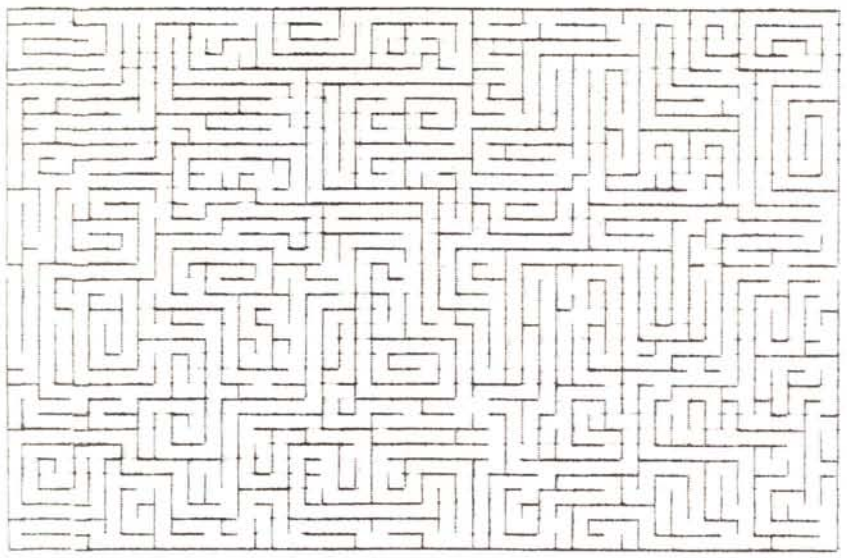
Comunque volendo dare un'idea, il nucleo nel quale opera l'algoritmo è una griglia ideale di 52×40 posizioni, anche se le reali dimensioni sono di 50×38 poiché due righe e due colonne sono utilizzate per i bordi; il labirinto d'esempio ha grandezza 9×10.

Questo nucleo viene collocato in memoria a partire da $4000 (#16384) ed occupa 52×40=2080 byte. Ogni singolo byte svolge più funzioni; nei suoi bit vengono indicati il tipo di carattere di stampa per la mappa, la direzione logica (negativa o positiva rispetto ad un ideale sistema di riferimento), oppure vengono usati come flag per segnalare se la locazione è un angolo, un bordo o la parte del tracciato di soluzione. Grazie alla somma di queste informazioni è possibile costruire il labirinto, rispettando in toto le condizioni iniziali.

È disponibile, presso la redazione, il disco con i programmi pubblicati in questa rubrica. Le istruzioni per l'acquisto e l'elenco degli altri programmi disponibili sono a pag. 263.

Caratteristiche programmi contenuti nel dischetto in vendita

BOOT:	Caricatore Basic
MMUSICA:	Dati e gestione musica in loop Inizio: $9000 36864 Fine: $9854 38996
SPRITES:	Dati sprite Inizio: $8000 32768 Fine: $8500 34048
CARATTERI:	Dati caratteri Inizio: $B100 45312 Fine: $B200 45568
MOT.LM:	Programma: generatore labirinto L.M. ecc. Inizio: $0800 2048 Fine: $2050 8872 Start: $1F60 8032
LAB.BASIC:	Generatore labirinto Basic Programma a parte



Esempio di labirinto generato dal programma.

Intervallo

Abbiamo il tempo di prendere il caffè? Sì, il Basic infatti è molto lento e aspettare che il programma finisca la generazione richiede almeno 10-15 minuti, che non è proprio poco. È possibile certo visionare il labirinto (tasti F1-F4 per far scorrere la mappa giù e su) e anche stamparlo (tasto S) ma per chi ha poca pazienza come noi o una passione folle per il tempo reale e l'interattività, la cosa risulta ben poco consolante. Il compito del listato pubblicato finisce qui.

Secondo tempo

Arrivati a questo punto della realizzazione era d'obbligo una serie di modifiche per rendere almeno un po' accattivante il programma. Abbiamo allora reso in tre dimensioni il labirinto, con parvenza di assonometria cavaliera, animato il tutto con lo scrolling nelle quattro direzioni e donato il movimento al protagonista di questa pazzia (una trottola e poi altri oggetti rotanti). Una musichetta in loop (è un ragtime originale, scritto e composto con tecnica fingerpiching per chitarra), un conto alla rovescia per creare (ma neanche tanto) un po' di angoscia, una mappa per aiutare il solutore ed infine una varietà di quattro schemi ci sono sembrati sufficienti come cornice al tutto.

«In Basic?!?!?» dirà qualcuno. No, purtroppo non siamo maghi, il tutto è stato realizzato in linguaggio macchina, necessario per movimentare l'azione e dare un po' di brio al tutto.

Ed i risultati si sono visti!!!

In primo luogo la generazione del labirinto, che anche se leggermente più piccolo non è meno complesso, richiede qualche secondo (ricordate in 15 minuti di prima?).

Ora addentriamoci nell'intimo del programma. La parte grafica utilizza una matrice inclinata di dimensione 12*6 (fig. 5) che permette con l'ausilio di alcuni parametri la ricostruzione assonometrica; quando dopo la prima volta, viene ricostruito il quadro video, si fa un semplice spostamento di memoria schermo e di memoria colore preoccupandosi di ricostruire solo la riga e/o la colonna necessaria. Per dare un po' di fluidità all'azione senza usare lo Smooth Scrolling che l'avrebbe rallentata troppo siamo ricorsi al controllo del RASTER (vedi MC n. 41) per l'uso del doppio schermo alternato, trucchi sicuramente noti ai programmatori in L.M.

In poche parole il raster è il pennello di elettroni che «disegna» la schemata su un televisore. Nel C64 esistono due byte che controllano tale pennello. Immaginiamo ora due mappe di memoria che contengono i valori da inviare al televisore; MOT MOT costruisce un nuovo schermo mentre visualizza il precedente. Quando ha finito tale operazione, aspetta che il raster «disegni» il bordo. In quel momento scambia le due mappe e riprende il ciclo.

Per quanto riguarda la gestione delle interruzioni se ne è parlato a lungo in questa rivista. Comunque vediamone brevemente le caratteristiche. All'interno del C64 esiste un integrato (6567) che ogni sessantesimo di secondo forza il microprocessore (6510) ad eseguire una routine che gestisce la tastiera, il clock interno ecc. Per permettere tale operazione esiste un vettore ($0314-$0315) che contiene il valore della locazione in cui prende avvio la procedura. Modificando questo vettore è possibile ottenere una routine che viene richiamata 60 volte al secondo e che è indipendente da qualsiasi programma. nel MOT MOT essa è gestita in modo che controlli il joystick, il movimento del protagonista, l'aggiornamento dell'orologio e la musica.

Su queste routine ci soffermiamo con due parole. Jostick è una routine molto semplice, infatti è un filtro: i dati forniti dalla porta joystick numero 1 ($DC01) vengono modificati in modo che possa-

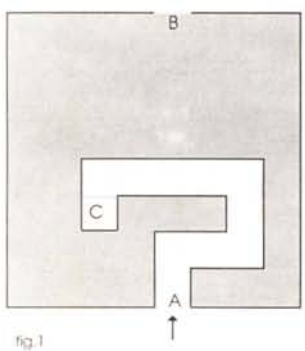


fig. 1

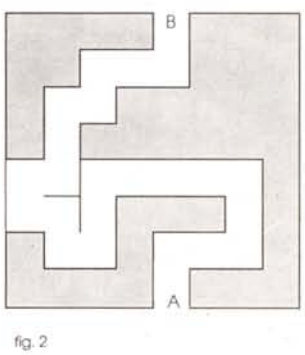


fig. 2

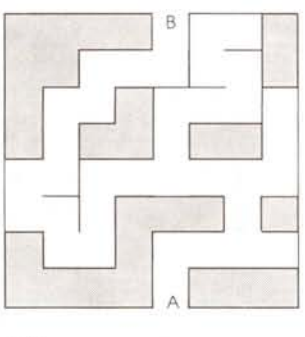


fig. 3

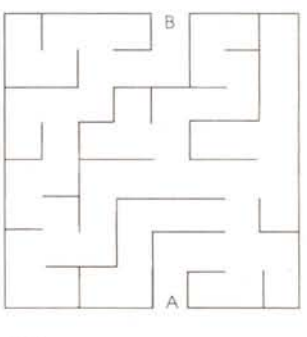
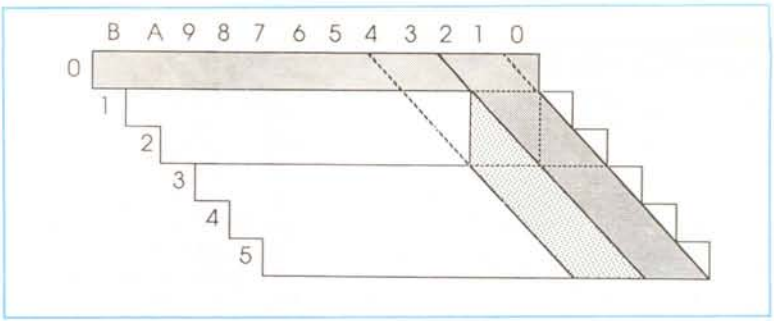


fig. 4



no essere accettati dalla routine di scrolling video.
Schema:

Direzione	Input ($DC01)	Output ($C5)
su	FE	41
giù	FD	40
sinistra	FB	81
destra	F7	80

Movimento del protagonista

Questa routine è responsabile dell'effetto di rotazione del protagonista. Una breve premessa permette di spiegare esaurientemente la questione. Parte un po' da lontano comunque... L'apparato visivo umano percepisce un movimento come una successione di singole immagini. Quindi per ottenere una sensazione di movimento è sufficiente creare una successione di immagini ad una velocità maggiore di quella di percezione dell'occhio. Questo è il motivo per cui i cartoon della Disney (anche Roger Rabbit) sono più fluidi e più belli di quelli giapponesi. Bando a tutto ciò questa tecnica è usata in MOT MOT. La routine, dipendendo dall'IRQ, è attiva sessanta volte al secondo, frequenza maggiore rispetto a quella dell'occhio.

Ogni sprite poi ha un byte che punta una zona di memoria in cui sono contenuti i valori del disegno ($XXF8-$XXFF, vedi manuale per programmatori del

Mot Mot Labyrinth.

```
3 FORI=49152TO49175:READQ:POKEI,Q:NEXT:SYS49152:GOTO499
5 POKE53280,0:POKE53281,0:POKE650,128:POKE56,160:PRINT"(CLR)";
10 PRINT"(CYN)
11 PRINT"(CYN)
(RVS)(OFF)
(YEL)
(PUR)
(WHT)LABYRINTH (PUR)
(RVS)(OFF)
12 FORI=1TO40:PRINT"(RVS)(GRY2) ";:NEXT
13 FORI=1TO17:PRINT" (OFF)(GRN)
   (RVS)(GRY2) ";:NEXT
15 FORI=1TO40:PRINT"(GRY2) ";:NEXT
20 PRINT"(HOME)(DOWN)(DOWN)(GRY2)(RVS)"SPC(38)"":FORI=1TO8:PRIN
T"(DOWN)":NEXT:PRINT"(RVS)"SPC(38)""
25 FORI=1904TO2023:POKEI,160:POKE54272+I,2:NEXT:PRINT"(DOWN)"
30 FORI=0TO52
35 POKE16384+I*40,160:POKE16423+I*40,160:NEXT
37 FORI=49152TO49186:READQ:POKEI,Q:NEXT
38 FORL=0TO96:READLL :POKE49200+L,LL:NEXT
40 X=0:PRINTTAB(12)"(UP)(RVS)(RED)F1 SU'  F7 GIU'  S STAMPA":GOSU
B100
50 GETA$:IFA$=""(F7)"ANDX<36THENX=X+1:GOSUB100
60 IFA$="(F1)"ANDX>0THENX=X-1:GOSUB100
65 IFA$="S"THENSYS49200:PRINT#4:CLOSE4
70 GOTO50
100 Y=X*40+16384:Z=INT(Y/256):W=Y-Z*256
110 POKE251,W:POKE252,Z:SYS49152
120 PRINT"(UP)(RVS)(RED) POS.:";STR$(X)" ":RETURN
130 DATA160,64,169,0,162,0,157,0,64,232,208,250,200,192,100,240,6
,238,8,192,76,4
140 DATA192,96,37,103,161,227,35,97,167,229,35,97,167,229,37,103,
161,227
142 DATA-1,1,40,-40
145 DATA160,64,132,255,160,0,132,254,177,254,208,4,169,4,208,3,74
,41,3,170
146 DATA189,102,156,145,254,200,208,236,166,255,232,134,255,224,7
3,208,227,96
147 DATA32,119,106,80,160
150 DATA169,4,141,11,192,160,0,177,251,153,120,4,200,240,12,192,1
68,208,244,173
160 DATA11,192,201,6,208,237,96,230,252,238,11,192,76,7,192
169 :
170 REM    ***  PRINTER  ***
171 :
180 DATA169,4,170,160,255,32,186,255,32,192,255,162,4,32,201,255,
169,51,133,252
181 DATA160,64,132,254,160,40,132,253,160,1,177,253,201,80,208,4,
169,112,208,18
182 DATA201,106,208,4,169,167,208,10,201,119,208,4,169,163,208,2
183 DATA169,32,32,210,255,200,192,38,208,220,169,8,32,210,255,169
,13,32,210,255
184 DATA169,15,32,210,255,24,169,40,101,253,133,253,144,2,230,254
,198,252,208
185 DATA188,96
298 :
299 REM    ***   DIREZIONE Y   ***
300 :
301 IFCY=0THENB=1:GOTO304
302 IFFY=0THENB=0:GOTO304
303 B=100
304 L=INT(RND(0)*5+1)*((RND(0)>1-Y1/100*B)*2+1):M=SGN(L):DR=0
305 FY=ABS(M*RS)>0:PS=P1:Y2=Y1:CX=0:Y=Y1:AC=0:H=0
306 GOSUB309:IFCX=1THENRETURN
307 IFABS(Y2-Y1)>ABS(L)ANDIN<>5THENGOTO326
308 GOSUB316:GOTO306
309 IFAC=0THENIFY-Y2=0THENAN=1:GOSUB700
310 Y=Y+M:PS=PS-M*40:IN=0:IFPEEK(PS)=16THENGOSUB316:GOTO800
311 FORJ=1TO1:IN=IN+ABS((PEEK(PS+J)AND239)<>0)*2^(J+1):NEXT
312 IFIN=0THENRETURN
313 IF(INAND2)=2THENGOTO319
314 IFIN=5THENGOTO324
315 G=(IN-2.5)/1.5:H=1:IF(PEEK(PS+G)AND64)=64*FYTHENPS=PS+G:GOTO3
19
316 IFAC=1THENCX=1:FX=ABS(G*RS>0)
317 RETURN
318 Y1=Y:P1=PS:POKEP1,FY*640R52:RETURN
319 F1=(PEEK(PS)AND128)/128:IFH<>0THENIFABS(G<0)=F1THENGOTO323
320 IFABS(Y2-Y)<=2THENGOTO323
321 IFDR=1THENGOTO323
322 FORY=Y1TOY1-RND(0)*(ABS(Y2-Y1)-2)*MSTEP-M:POKEP1,0:P1=P1+M*40
:NEXT:Y1=Y
323 FX=F1:CX=1:RETURN
324 IF(PEEK(PS-1)AND64)<>(PEEK(PS+1)AND64)THENPS=PS+G*RS:GOTO319
325 DR=1:RETURN
326 AC=1:GOSUB316
327 IFIN=10RIN=4THENGOTO330
328 GOSUB309:IFIN=5ANDCY=0THENGOTO306
329 RETURN
330 GOSUB309:IFIN=0THENGOSUB316
331 RETURN
398 :
399 REM    ***   DIREZIONE X   ***
400 :
401 IFCX=0THENB=1:GOTO404
402 IFFX=0THENB=0:GOTO404
403 B=37
404 L=INT(RND(0)*5+1)*((RND(0)>1-X1/37*B)*2+1):M=SGN(L):DR=0
405 FX=ABS(M*RS)>0:PS=P1:X2=X1:CY=0:X=X1:AC=0:H=0
406 GOSUB409:IFCY=1THENRETURN
407 IFABS(X2-X1)>ABS(L)ANDIN<>5THENGOTO426
408 GOSUB418:GOTO406
409 IFAC=0THEN1FX-X2=0THENAN=0:GOSUB700
410 X=X+M:PS=PS+M:IN=0:IFPEEK(PS)=16THENGOSUB418:GOTO800
411 FORJ=-1TO1:IN=IN+ABS((PEEK(PS+J*40)AND239)<>0)*2^(J+1):NEXT
412 IFIN=0THENRETURN
413 IF(INAND2)=2THENGOTO419
414 IFIN=5THENGOTO424
415 G=(IN-2.5)/1.5:H=1:IF(PEEK(PS+G*40)AND128)=FX*128THENPS=PS+40
*G:GOTO419
416 IFAC=1THENCY=1:FY=ABS(G*RS<0)
417 RETURN
418 X1=X:P1=PS:POKEP1,FX*1280R50:RETURN
419 F1=(PEEK(PS)AND64)/64:IFH<>0THENIFABS(G<0)=F1THENGOTO423
420 IFABS(X2-X)<=2THENGOTO423
421 IFDR=1THENGOTO423
422 FORX=X1TOX1-RND(0)*(ABS(X2-X1)-2)*MSTEP-M:POKEP1,0:P1=P1-M:NE
XT:X1=X
423 FY=F1:CY=1:RETURN
424 IF(PEEK(PS-40)AND128)<>(PEEK(PS+40)AND128)THENPS=PS-40*G*RS:G
OTO419
425 DR=1:RETURN
426 AC=1:GOSUB418
427 IFIN=10RIN=4THENGOTO430
428 GOSUB409:IFIN=5ANDCY=0THENGOTO406
429 RETURN
430 GOSUB409:IFIN=0THENGOSUB416
431 RETURN
497 :
498 REM  ***  MASTER  ***
499 PRINT"(CLR)ATTENDI PREGO...(10-15 MIN.)"
500 MS=255:RS=1:P%=RND(0)*10+30:F%=0:FY=0:CY=1:FX=0:P1=10444:DR=0
502 FORI=0TO50:POKEI*40+16385,12:POKEI*40+16422,12:NEXT
503 FORI=2TO19:POKE16384+I,10:POKE18424+I,138:NEXT
504 FORI=20TO37:POKE16384+I,138:POKE18424+I,10:NEXT:POKE16444,16
505 DIMA%(7,1):PORTV=0TO1:FORI=0TO7:READA%(I,TV):NEXT:NEXT
506 G=1:X1=20:Y1=0:TV=0
507 DIMB%(3):FORI=0TO3:READB%(I):NEXT
510 FORN=F%TOP%:ON(NAND1)+1GOSUB300,400
520 NEXTN
597 :
598 REM  ***  AGGANCIO  ***
600 :
601 POKE16444,0:FORI=16403TO16405:POKEI,0:NEXT:I=16:GOSUB1600
```

C64). Detto questo è semplice spiegare il funzionamento della routine: essa cambia i valori di questo byte ciclicamente. I disegni infatti sono stati pensati in modo che un eventuale quinto disegno corrisponda al primo e così via.

Due parole sull'uso del programma fornito dalla redazione su dischetto: per caricare il gioco battere LOAD"BOOT", 8 attendendere il caricamento e premere ancora RETURN. Apparirà la schermata iniziale per partire FIRE (Joystick in Port#1). È possibile vedere parte della mappa del labirinto premendo FIRE in modo continuo, ma solo per tre volte. Per caricare il listato BASIC con la sola generazione del labirinto LOAD"LAB.BASIC", 8.

BIBLIOGRAFIA

Guida di riferimento per il programmatore C64
Il S.O. del C64 editrice E.V.M.
MCmicrocomputer nn. 40-41-48-49.

TOOL-64

di Gaetano Minardi - Niscemi (CL)

Ormai è cosa risaputa, che il Basic V2.0 del C64 presenta molte lacune e una carenza di comandi che talvolta risultano indispensabili per facilitare la programmazione. Lunghe linee di PEEK e POKE accompagnano ormai quotidianamente i nostri programmi con grande disappunto dei principianti, i quali non conoscendo il significato di alcune locazioni di memoria abbandonano i programmi e si dedicano pazientemente alla risoluzione di qualche game.

Espansioni del Basic V2.0 ne esistono ormai a centinaia (sia su cartuccia che su cassetta), e danno la possibilità a chiunque di sbizzarrirsi a programmare con estrema semplicità effetti speciali grafici o sonori, o talvolta solo di creare programmi didattici ed educativi (i cosiddetti di utilità varia). Ormai molti programmatori di linguaggio macchina si divertono a creare espansioni per il loro Commodore, cercando di superare le barriere del Basic standard. La mia espansione serve proprio a questo, risolve però i problemi che si incontrano durante la programmazione, proponendo dei potenti comandi che facilitano la stesura dei programmi, quindi accrescere la versatilità del C64.

L'esigenza di avere comandi che servissero nei programmi Basic durante la fase di sviluppo, di messa a punto o di debug, si sentiva ormai da tempo, e Tool-64 fa proprio questo. Tool-64 quindi non è una vera e propria espansione

```
602 P=X1:Q=Y1:D=AN:F%=RND(0)*2+1:RS=-1:CX=0:CY=0
603 F2=FY:F3=FX:TV=1:FY=1:FX=1:X1=20:Y1=52:P1=16364:DR=1:M=-1
604 IFF%=2THENL=0:GOSUB305:F%=1:GOTO606
605 L=-RND(0)*5+2:GOSUB305:F%=1
606 ONF%GOTO610,620
610 F%=2:IFCX=0THENFX=ABS(X1<P)
611 IFFX=0THENB=37:GOTO613
612 B=0
613 GOSUB404:GOTO606
620 F%=1:IFCY=0THENFY=ABS(Y1<Q)
621 IFFY=0THENB=100:GOTO623
622 B=0
623 GOSUB304:GOTO606
697 :
698 REM  ***  SVOLTE  ***
699 :
700 I=AN*4+FX*2+FY:POKEP1,A%(I,TV)ANDMS:RETURN
797 :
798 REM  ***  UNCINAZIONE  ***
799 :
800 IFTV=1THENOND=1GOTO820,810
801 IF(NAND1)=1THENFY=1:AN=1:GOSUB700
802 GOTO900
810 IFP=X1THENGOTO815
811 IFD<>(F%AND1)THENGOTO813
812 FX=ABS(X1>P):AN=0:GOSUB700
813 P1=P1+P-X1
814 AN=1:FY=F2:GOSUB700:GOTO900
815 IFD=(F%AND1)THENGOTO900
816 GOTO814
820 IFQ=Y1THENGOTO825
821 IFD<>(F%AND1)THENGOTO823
822 FY=ABS(Y1>Q):AN=1:GOSUB700
823 P1=P1+(Y1-Q)*40
824 AN=0:FX=F3:GOSUB700:GOTO900
825 IFD=(F%AND1)THENGOTO900
826 GOTO824
897 :
898 REM  ***  ESTERNO  ***
899 :
900 P1=P+(51-Q)*40+16384:D=0:GOSUB1600
901 TV=0:MS=223
902 FORI=16403TO16405:POKEI,10:POKE16404,16:NEXT
917 :
918 REM  ***  PERCORRITORE  ***
919 :
920 FORVL=-1TO1STEP2:K=2+RND(1)*4:PS=18364:DR=1:F2=1
921 K=K-1:IFK<0THENGOSUB1000:K=(1-RND(1)↑2)*7+1:GOTO921
922 IFDR=0THENPS=PS+F3*2-1:GOTO924
923 PS=PS-(F2*2-1)*40
924 A=PEEK(PS):IF(AAND1)=1THENF3=(AAND128)/128:F2=(AAND64)/64:DR=
(NOTDR)AND1
925 IFA=16THENNEXTVL:GOTO1300
926 GOTO921
997 :
998 REM  ***  PRE-FINESTRA  ***
999 :
1000 IFDR=1THENGOTO1021
1011 IFF3=1THENGOTO1014
1012 P1=PS-40*VL:IFPEEK(P1)=0THENFY=ABS(VL>0):GOTO1016
1013 RETURN
1014 P1=PS+40*VL:IFPEEK(P1)=0THENFY=ABS(VL<0):GOTO1016
1015 RETURN
1016 GOSUB1100:GOTO1200
1021 IFF2=0THENGOTO1024
1022 P1=PS+VL:IFPEEK(P1)=0THENFX=ABS(VL>0):GOTO1026
1023 RETURN
1024 P1=PS-VL:IFPEEK(P1)=0THENFX=ABS(VL<0):GOTO1026
1025 RETURN
1026 GOSUB1102:GOTO1200
1097 :
1098 REM  ***  FINESTRA  ***
1099 :
1100 IFFY=1THENGOTO1103
1101 RETURN
1102 IFFX=0THENRETURN
1103 POKEPS,PEEK(PS)AND(255-2↑(DR+1)):RETURN
1197 :
1198 REM  ***  RAMIFICAZIONE  ***
1199 :
1200 P%=RND(0)*7+8:H=1:D=0
1201 P%=P%-1:GOSUB1700:L=RND(0)*8+1:I=4
1202 I=I-1:C=B%(I)
1203 IFABS(D)=ABS(C)THENGOTO1213
1204 IFPEEK(P1+C)<>0THENGOTO1213
1205 AN=ABS(ABS(C)<>1):D=C:IFAN=1THENFY=ABS(C<0):GOTO1207
1206 FX=ABS(C>0)
1207 IFH=0THENGOTO1209
1208 H=0:IFAN<>DRTHENGOTO1211
1209 GOSUB700
1210 IFL>0ANDPEEK(P1+D)=0THENP1=P1+D:POKEP1,2↑(1+AN)+16:L=L-1:GOT
01210
1211 IFP%<>0THENGOTO1201
1212 GOTO1220
1213 IFI<>0THENGOTO1202
1217 :
1218 REM  ***  CHIUDE FINESTRA  ***
1219 :
1220 IFH=1THENAN=(NOTDR)AND1:FX=ABS(P1>PS):FY=(NOTFX)AND1:L=1:P%=
0:H=0:GOTO1210
1221 IFAN=0THENGOTO1224
1222 IFFY=1THENGOTO1225
1223 RETURN
1224 IFFX=0THENRETURN
1225 POKEP1,7:RETURN
1297 :
1298 REM  ***  FILLER  ***
1299 :
1300 FORR=18384TO16384STEP-40:H%=(RND(0)>.5)*2+1:VL=1
1301 R1=2:R2=37:IFH%<0THENR2=2:R1=37
1302 E=1
1310 IFPEEK(R+R1)=0THENE=0:GOTO1320
1311 IFR1=R2THENGOTO1313
1312 R1=R1+H%:GOTO1310
1313 IFE=0THENGOTO1301
1314 NEXTR:GOTO1500
1320 GOSUB1700:I=4
1321 I=I-1:C=B%(I):A=PEEK(R+R1+C)
1322 IFA<>0THENGOTO1330
1323 IFI<>0THENGOTO1321
1324 GOTO1311
1330 IF(AANDB)=BTHENGOTO1323
1331 DR=ABS(ABS(C)=1):PS=R+R1+C
1332 F2=ABS(C<0):F3=F2:GOSUB1000:GOTO1311
1497 :
1498 REM  ***  USCITA  ***
1499 :
1500 FORI=16384TO16423:POKEI,8:NEXT
1501 FORI=0TO51:POKEI*40+16382,8:NEXT
1505 FORI=40000TO40042:READQ:POKEI,Q:NEXT:SYS40000:GOTO5
1597 :
1598 REM  ***  RAGGI  ***
1599 :
1600 FORJ=1TO40STEP39:FORI=-1TO1STEP2:PS=P1+J*I
1601 IF(PEEK(PS)AND239)=0THENPOKEPS,D
1602 NEXT:NEXT:RETURN
1697 :
1698 REM  ***  MIX  ***
1699 :
1700 FORI=0TO6
1701 N=RND(0)*4:M=RND(0)*4
1702 C=B%(N):B%(N)=B%(M):B%(M)=C:NEXT
1703 RETURN

READY.
```

in senso classico, ma ambiente di sviluppo con efficientissimi comandi da usare solo in modo diretto (e non poteva essere diversamente), in quanto servirsi di questi comandi per inserirli nei programmi è pressoché inutile.

Il programma si colloca in memoria nei soliti 4K di RAM situati dopo l'interprete Basic, e quindi non intacca la RAM del Basic. Più precisamente il programma parte dall'indirizzo $C000 (49152 decimale) e termina in $CC8D (52365 decimale); viene attivato da SYS 49152. Una volta dato il RUN, dopo alcuni minuti Tool-64 segnala che tutto è pronto con il messaggio di Copyright, e a questo punto tutti i comandi nuovi sono disponibili nel modo diretto.

Descrizione dei comandi

Ecco un elenco alfabetico completo dei 14 comandi di Tool-64, con le rispettive istruzioni e degli esempi dimostrativi.

AUTO: questo comando attiva la funzione di numerazione automatica di riga, facilitando l'operazione di input dei programmi, in quanto il computer provvede a stampare automaticamente i numeri di riga ogni volta che viene premuto il tasto RETURN per fare accettare la riga di programma corrente. La sintassi è la seguente: AUTO riga. Esempi: AUTO 20 numera automaticamente le righe di programma con incrementi di 20; AUTO disattiva la numerazione automatica di riga.

CATALOG: visualizza sullo schermo il contenuto dell'elenco del disco, senza cancellare il programma in memoria. Per rallentare la visualizzazione premere il tasto CTRL.

CODE: serve per la generazione di linee DATA. Indispensabile per trasformare in dati decimali i programmi in L.M. e qualsiasi locazione o zona di memoria del C64. La sintassi è la seguente: CODE inizio, fine, prima linea DATA; dove con inizio si intende la prima locazione di memoria che si desidera trasformare in DATA; con fine si intende l'ultima locazione di memoria desiderata e con «prima linea DATA» si intende il numero di linea da cui si vogliono fare partire le linee DATA. Esempio: CODE 0,255,250 trasforma in DATA tutta la pagina zero. La prima linea DATA sarà la linea 255.

DELETE: Consente di cancellare un gruppo di linee Basic nella sequenza specificata. La sintassi è identica a quella del comando LIST. Esempi: DELETE 20 cancella la linea 20; DELETE 20-80 cancella tutte le linee comprese tra 20 e 80, estremi compresi; DELETE-60 cancella tutte le linee dall'inizio del programma fino alla linea 60 compresa; DELETE 60- cancella tutte le linee dalla 60 all'ultima linea del programma, 60 compresa.

DUMP: questo comando consente di visualizzare il contenuto delle variabili memorizzate (array esclusi) nell'ordine in cui sono state dichiarate. Nel caso in cui le variabili occupano più di una schermata, è possibile rallentare lo scrolling tenendo premuto il tasto CTRL. Esempi: DUMP lista tutte le variabili; DUMP$ visualizza solo le variabili stringa; DUMP% lista solo le variabili intere; DUMP& lista solo le variabili reali.

FIND: elenca tutte le linee dove è contenuta la sequenza specificata. Se il parametro è indicato tra virgolette la ricerca avverrà solo su stringhe ASCII, mentre in caso contrario verrà effettuata una tokenizzazione delle parole chiave del Basic. Esempi: FIND A elenca le linee dove è contenuta la variabile A; FIND"TOOL-64" elenca le linee dove è contenuta la stringa ASCII specificata tra le virgolette (nell'esempio la stringa TOOL-64); FIND PRINT ricerca ed elenca tutte le linee dove è contenuta l'istruzione PRINT.

KILL: consente di compattare un programma Basic eliminando gli spazi e le REM. Utilissimo nella messa a punto dei programmi per risparmiare preziosa memoria.

MERGE: consente di leggere un programma da disco o nastro e di aggiungerlo al programma in memoria. Usando questo comando occorre molta attenzione per evitare di avere i due programmi con numeri di linea eguali. Per questo si consiglia di rinumerare le linee del programma corrente con il comando RENUMBER (presente in questa espansione), oppure a unione avvenuta è d'obbligo rinumerare il programma. Prima di usare il comando MERGE si consiglia di eseguire il comando CLR, per cancellare il contenuto delle variabili in memoria. La sintassi è identica al comando LOAD:MERGE"nome file",n dove «n» indica il numero di periferica desiderata (1 per il registratore e 8 per il drive).

QUIT: disattiva i comandi di Tool-64. Non necessita di parametri.

OLD: Comando utilissimo che consente di recuperare un programma Basic cancellato dalla memoria dal comando NEW. Se però dopo la cancellazione si fossero inserite delle nuove linee, oppure dichiarata una variabile, il programma precedente non potrà più essere recuperato.

RENUMBER: consente di rinumerare le linee del programma in memoria compresi i GOSUB, GOTO, ON GOSUB, ON GOTO, IF/THEN. La sintassi possibile è la seguente: RENUMBER nnri,inc,vnri. La sigla «nnri» indica la nuova riga iniziale, cioè il numero della prima riga del programma dopo la rinumerazione. La sigla «inc» sta per incremento, cioè l'intervallo tra i numeri di riga. Infine «nnri» sta per vecchio numero di riga iniziale cioè il numero della prima riga prima della rinumerazione del programma. Esempi: RENUMBER rinumera le righe del programma a partire dalla 10 con incrementi di 10; RENUMBER 10,20,50 rinumera il programma partendo dalla linea 50. La linea 50 diventa la linea 10, e tutte le linee successive verranno incrementate di 20 in 20; RENUMBER,,50 rinumera con incrementi di 10 partendo dalla linea 50. La linea 50 diventa la linea 20. Se si desidera omettere un parametro, al suo posto si dovrà inserire una virgola, come nell'esempio.

REPLACE: sostituisce una sequenza con un'altra nel programma Basic. La sintassi è la seguente: REPLACE [sdr][sds]. Il primo parametro indica la sequenza da ricercare, mentre il secondo la sequenza da sostituire. Se i parametri sono messi tra doppi apici, verrà ricercata la stringa ASCII specificata e sostituita con quella indicata nel secondo parametro. Se invece i parametri non sono messi tra doppi apici, vengono tokenizzati, cioè viene effettuato un riconoscimento delle istruzioni fondamentali del Basic V2.0. Esempi: REPLACE [A][B], cambia la variabile A con B, cioè la variabile A viene annullata e il suo posto viene preso dalla nuova variabile B; REPLACE [GOTO][GOSUB], vengono ricercati tutti i GOTO del programma e sostituiti da GOSUB; REPLACE ["TOOL-64"]["BASIC EXTENSION"], nel programma Basic, ogni qual volta viene incontrata la stringa ASCII "TOOL-64", verrà sostituita dalla nuova stringa specificata (nell'esempio "BASIC EXTENSION").

TRON: utilissimo comando che permette di cercare gli errori in un programma Basic. Consente infatti di vedere il flusso esecutivo del programma mentre è in esecuzione. La linea in esecuzione verrà visualizzata mentre è in esecuzione. La linea in esecuzione verrà visualizzata nella parte superiore dello schermo in modo reverse. Sintassi: TRON [numero linea][,rit,rit]. Il primo parametro indica il numero di linea dal quale si vuole vedere il flusso esecutivo; gli altri due parametri riguardano i ritardi di esecuzione, permettendo di rallentare il programma, per controllare meglio l'esecuzione globale e le linee in esecuzione in particolare. Esempi: TRON comincia ad elencare le linee in esecuzione a partire dalla prima linea; TRON[1000][,10,10] permette di elencare le linee in esecuzione a partire dalla linea 1000, con un rallentamento dell'esecuzione pari a dieci unità.

TROFF: disabilita in modo TRON, ristabilendo le condizioni normali di esecuzione. Non necessita di alcun parametro.

Elenco del software disponibile su cassetta o minifloppy

Per ovviare alle difficoltà incontrate da molti lettori nella digitazione dei listati pubblicati nelle varie rubriche di software sulla rivista, MCmicrocomputer mette a disposizione i programmi più significativi direttamente su supporto magnetico. Riepiloghiamo qui sotto i programmi disponibili per le varie macchine, ricordando che i titoli non sono previsti per computer diversi da quelli indicati. Il numero della rivista su cui viene descritto ciascun programma è riportato nell'apposita colonna; consigliamo gli interessati di procurarsi i relativi numeri arretrati, eventualmente rivolgendosi al nostro Servizio Arretrati utilizzando il tagliando pubblicato in fondo alla rivista.

Per l'ordinazione inviare l'importo (a mezzo assegno, c/c o vaglia postale) alla Technimedia srl, Via Carlo Perrier 9, 00157 Roma.

Codice	Titolo Programma	MC n.	Prezzo
APPLE II			
DA2/06	Miniset + LevaDOS	37	15000
DA2/07	27 programmi grafici	38	30000
DA2/08	Adventure Editor	38	15000
DA2/09	Animazione Funzioni	42	15000
DA2/12	Routine Grafiche Estese	44	15000
DA2/13	Scroll 300 linee	46	15000
DA2/14	Assembler in Basic	50	15000
DA2/15	G-Basic II	53	15000
DA2/16	Disk Editor	54	15000
DA2/17	Latino	57	15000
DA2/18	Battaglia	61	15000
DA2/19	Catalogo	64	15000
DA2/20	Apple Puzzle II	65	15000
DA2/21	Precisione Multipla	66	15000
DA2/22	Sistema 2 + Toto 5.3 IIGS	68	15000
DA2/23	Operazione Apokalypsis	71	30000
DA2/24	Classifiche di Formula 1	72	15000
DA2/25	Programmabile RPN	73	15000
DA2/26	Supercircle + Poligonale	74	15000
DA2/27	Hard Copy OKI 83/A	76	15000
DA2/28	ProDOS Utility	77	15000
DA2/29	Modulo Base	78	15000
DA2/30	List db	79	15000
DA2/31	Bioritmi	80	15000
COMMODORE AMIGA			
DAM/01	F-15	63	15000
DAM/02	Gest. liste programmi	64	15000
DAM/03	Studio di Funzioni	66	15000
DAM/04	Math Pack	68	15000
DAM/05	Redcode & Mars (Core Wars)	68	15000
DAM/06	Life	69	15000
DAM/07	Rubrica Telefonica	70	15000
DAM/08	Piramidi	70	15000
DAM/09	Regolazione dei colori	71	15000
DAM/10	Analitica	71	15000
DAM/11	Grafici	72	15000
DAM/12	Traduttore	73	15000
DAM/13	La Borsa	74	15000
DAM/14	DMA Music Compiler	74	15000
DAM/15	Poker	78	15000
DAM/16	Programmi per il Copper	79	15000
DAM/17	Mandelbrot mania	81	15000
DAM/18	SF-Search File	86	15000
MS-DOS			
DMS/01	Plotter + Morse	67	15000
DMS/02	Melole + Spawn	68	15000
DMS/03	Pretty + Scritte scorrevoli + Compute	69	15000
DMS/04	Emulatore CGA per Hercules	70	15000
DMS/05	Turbo Directory	71	15000
DMS/06	Math Tool S	72	15000
DMS/07	Bioritmi + Routine	72	15000
DMS/08	Salvavideo + Scritte scorrev. + PG151	73	15000
DMS/09	Optimizer + Indenter dBase III	74	15000
DMS/10	Joystick Controller	75	15000
DMS/11	BootSlow & SlowDown + Turbo Utility	76	15000
DMS/12	Redcode & Mars (Core Wars)	76	15000
DMS/13	Gestione Errori Critici Disco + PosCur	77	15000
DMS/14	Finestre & Desk	78	15000
DMS/15	General Manager	78	15000
DMS/16	Tool 05	79	15000
DMS/17	Pulldown Menu + Retrace	80	15000
DMS/18	Righe	81	15000
DMS/19	La spada di Krall	82	15000
DMS/20	Regressione	82	15000
DMS/21	Tesseract + Charset Editor	83	15000
DMS/22	Sega-File + Installatore	84	15000
ATARI ST			
DST/01	Virus Killer	74	15000
DST/02	Mandelbrot + Proiez. Ort. + Bilancio	78	15000
DST/03	Diagrammi di Heron	81	15000
DST/04	Paroliamo	84	15000
DST/05	Enalotto	85	15000
COMMODORE 128			
D28/01	MMCalc	53	15000
D28/02	Hardcopy 128	55	15000
D28/03	Sheet II	57	15000
D28/04	Star Quest	58	15000
D28/05	Family Budget	60	15000
D28/06	La Casa Stregata	61	15000
D28/07	Strutture 80/33	63	15000
D28/08	Bas 80 V. 2.0a	64	15000
D28/09	Paint 80 1.0	65	15000
D28/10	Bas 80 V. 2.11	66	15000
D28/11	Calendario Perpetuo + Montecarlo	67	15000
D28/12	Disegna Circuiti	68	15000
D28/13	Mark's Data Base	70	15000
D28/14	Label Disk + Disk Editor + Dem DOS	71	15000
D28/15	Pulldown 128HR + Menu + Drawer	72	15000
D28/16	Prospettive	73	15000
D28/17	Char 80 V. 1.0	74	15000
D28/18	Italia 128	75	15000
D28/19	Super Sprite	77	15000
D28/20	Othello	80	15000
D28/21	Expert System Shell 128	81	15000
D28/22	Kit di programmazione S.O.G.A.R. 128	82	15000
D28/23	Caratteri Programmabili	83	15000
D28/24	Brush & Mouse	84	15000
COMMODORE 64			
D64/11	Anno Domini	57	15000
D64/12	The Disk Editor	54/6/7	15000
D64/13	Boz's Adventure	57	15000
D64/14	Link-64	57	15000
D64/15	New Char 2.2	58	15000
D64/16	Music 64	59	15000
D64/17	TRX-MEM	59	15000
D64/18	WOS + WBasic	60	15000
D64/19	Strange Basic + Dracula	63	15000
D64/20	File Rescue	64	15000
D64/21	La Casa	64	15000
D64/22	Digital Voice	65	15000
D64/23	Vita 3D	65	15000
D64/24	Corso di Linguistica	66	15000
D64/25	Archiplus	66	15000
D64/26	Math Pack Plus	66	15000
D64/27	Scroll + Multitask + Classifica	67	15000
D64/28	Calend. Perpetuo + Effetto Telecamera	68	15000
D64/29	Listing Plus + Utility Data	69	15000
D64/31	Trucchi e Routine per programmatori	71	15000
D64/32	Flow-Chart + Flower's Love	73	15000
D64/33	Sprite Editor	76	15000
D64/34	Portfolio 64 + Elimin. bordi schermo	77	15000
D64/35	Alfabeto Morse + Locate + Menu/Driver	78	15000
D64/36	Schedario Gare	80	15000
D64/37	Intonatore	81	15000
D64/38	Gendata 64	82	15000
D64/39	La mappa del Domino	83	15000
D64/40	Block Utility	84	15000
D64/41	Sprite	85	15000
D64/42	Mot Mot Labyrinth + Tool 64	86	15000
MSX			
DMX/01	Toto 13	60	15000
DMX/02	Painter	62	15000
DMX/03	MSX Bank	63	15000
DMX/04	Grafica 3D + Hard Copy	65	15000
DMX/05	Easy Disk	66	15000
DMX/06	Classifiche	67	15000
DMX/07	Magic Paint	67	15000
DMX/08	Autogest	68	15000
DMX/09	Compilatore v. 1.01	69	15000
DMX/10	Diskmap	70	15000
DMX/11	Mini dBase MSX	71	15000
DMX/12	Grafica in Turbo Pascal	72	15000
DMX/13	Math Pack Plus 3.20	73	15000
DMX/14	RGB CAD	75	15000
DMX/15	Simple Desk	76	15000
DMX/16	The MSX2 Super Print	77	15000
DMX/17	Grafica in Turbo Pascal (Graph 1&2)	77	15000
DMX/18	Hard Copy	78	15000
DMX/19	HEXDUMP	79	15000
DMX/20	Utilities in Turbo Pascal	80	15000
DMX/21	dBase MSX Plus	81	15000
DMX/22	Turbo Pascal Turtle Graphics	82	15000
DMX/23	PutChar + 4010 S	84	15000
DMX/24	Analysis + Lister Basic	85	15000
DMX/25	Aliment MSX	86	15000

Nota:
l'iniziale del codice è C per le cassette, D per i floppy.

Le radici dell'azienda sono nei nostri programmi.

Abbiamo fatto grandi passi avanti in questi anni. Lo dobbiamo alla grande esperienza Buffetti che c'è alla base della nostra azienda. Un'esperienza nella gestione aziendale che a noi è servita e che voi adesso trovate nel software e nell'hardware Buffetti Data.

Un'esperienza che ha permesso di creare dei programmi dove innanzitutto emerge una grande competenza in tutti i settori trattati, unita ad una grande chiarezza e facilità d'uso.
Sono le caratteristiche che privilegia chi conosce a fondo il settore ed è interessato prima di tutto alla soluzione dei problemi di gestione. Non ultimo quello di avere garantita un'assistenza capillare, la stessa offerta dai 900 punti vendita Buffetti e rivenditori autorizzati in tutta Italia.

Buffetti data

Soluzioni hardware e software per aziende e professionisti