Fare il redattore per MC microcomputer comincia a diventare un lavoro faticoso, specialmente da quando abbiamo avviato questa rubrica. Infatti il software che arriva in redazione è aumentato in modo tanto considerevole da traboccare fuori dalla mia casella postale. Se continua così sarò costretto a comprare una carriola per portare alla mia scrivania la pila di pacchetti postali che arriva ogni giorno. A proposito di questi è interessante notare la «cura» con cui sono confezionati: si passa dal semplice dischetto inviato senza alcuna protezione (affidandolo solo al buon cuore degli addetti postali) al pacco a prova di film catastrofico tipo «Terremoto sotto l'uragano nella città in fiamme». Per fare un pacchetto di questo tipo si comincia con uno schermo anti radiazioni cosmiche costituito da un sottile foglio di stagnola (ci vorrebbe l'oro, ma possiamo accontentarci della stagnola) poi un primo strato per assorbire le vibrazioni ad alta frequenza generalmente composto da due fogli di polistirolo di differente spessore (per evitare le risonanze). A questo punto due fogli di compensato garantiscono la necessaria resistenza alla flessione, poi si mette lo schermo contro i campi elettromagnetici: una scatola di lamierino di ferro tipo quelle dei biscotti andrà benissimo e infine, contro le vibrazioni di bassa frequenza, alcuni strati di quella plastica a bolle per imballaggi. Il tutto viene tenuto insieme da vari passaggi con lo scotch da pacchi e infilato in una busta imbottita su cui va scritto, più che altro per precauzione, «ATTENZIONE CONTIENE DISCHETTO MAGNETICO».

Scherzo? Niente affatto almeno il dieci percento dei dischi in arrivo viaggia in imballaggi del genere, e non vi dico la fatica per aprirli (sempre a patto di disporre di una officina abbastanza attrezzata)! Quanto al contenuto, come già accennato tempo fa, il dischetto dovrebbe contenere:

— un file con i dati dell'autore
— un file ASCII con il testo dell'articolo
— i file dei programmi (possibilmente ASCII)
— se ci sono i file delle tabelle o delle didascalie.

Se ci sono delle fotografie che accompagnano il software sarebbe preferibile usare delle diapositive (vengono meglio in stampa).

Alcuni lettori hanno inviato il disco con il solo programma, senza una sola riga di descrizione. È un po' poco...

Scusate, ma devo correre a puntellare la mia scrivania che minaccia di crollare sotto l'ultimo quintale di software appena scaricato dal postino, se non leggerete più il mio nome su questa rubrica vorrà dire che il puntello ha ceduto.

Bootslow & Slowdown

di Nicolò Trio - Milazzo (ME)

Posseggo da un anno un compatibile PC, dotato di un misero 8088 a 4.77 MHz. Ma conosco tanta gente che possiede compatibili più veloci, tipo M24 e simili. Questi sono felicissimi della maggiore velocità, quando vogliono usare il computer per impieghi seri. Nel momento in cui vogliono usare il computer per giocare, però, la maggiore velocità diventa un problema: infatti molti giochi risultano troppo veloci o addirittura praticamente ingiocabili.

Ora si potrebbe dire che il PC non è fatto per giocare, e che se si vuole giocare ci si compra un Commodore 64 o simili, ma se non si vuole spendere ma ci piacerebbe utilizzare il computer anche per scopi ludici?

Una soluzione c'è: è possibile realizzare un programma rallentatore.

Sfruttando opportunamente le risorse hardware del PC, sono riuscito a trovare un sistema per rallentare di un fattore arbitrario l'esecuzione di quasi tutti i programmi, inclusi i giochi (anche quelli protetti da boot-strappare).

Utilizzo di Bootslow e Slowdown

Queste utility rallentano l'esecuzione dei programmi di un fattore che può essere selezionato a piacere da un minimo di 1 a un massimo di 65535.

Purtroppo, non tutti i programmi possono essere rallentati, e qualcuno potrebbe persino dar adito a malfunzionamenti (in qualche raro caso).

Comunque, per quelli che si possono rallentare, il fattore di rallentamento può essere scelto a piacere fino a velocità anche di 1/10 rispetto all'originale.

Differenze tra i due programmi

Bootslow: si carica da DOS e, se viene chiamato con un parametro (il fattore di rallentamento), questo effettua automaticamente il bootstrap caricando dal drive A. Quindi, questa versione serve per rallentare programmi protetti e illegibili da DOS. Attenzione, il programma non riserva memoria per la routine rallentatrice, ma si limita a rendere improbabile l'interferenza con altri programmi caricando la routine rallentatrice in locazioni molto alte di memoria.

Comunque se qualche programma non dovesse funzionare (nel senso che mostra dei malfunzionamenti, non nel senso che non si rallenta affatto), è possibile specificare come secondo parametro un indirizzo di segmento dove caricare la routine rallentatrice, oppure si può provare a diminuire il fattore di rallentamento, e se nessuno di questi sistemi funziona, beh, non c'è niente da fare.

Il programma bootslow può essere chiamato da qualsiasi drive o directory. Inoltre è possibile effettuare il bootstrap anche dal drive b: specificandolo nella linea di comando (chiamando bootslow senza parametri, esso dà la lista delle opzioni e come usarle).

Attenzione: fare sempre il bootstrap prima di riusare bootslow.

Sintassi del comando: bootslow numero [segmento][drive]; «numero» è un parametro necessario in quanto spe-

cifica il fattore di rallentamento (1= minimo rallentamento, 65535 = massimo rallentamento). Gli altri due parametri sono opzionali: il primo serve a specificare un indirizzo (di segmento) al quale caricare la routine rallentatrice diverso da quello di default (hex 9000:0000 cioè segmento hex 9000 cioè in decimale 36864, e offset 0); nota che è possibile specificare solo l'indirizzo di segmento: suggerisco di provare sempre e soltanto valori alti.

Il secondo parametro permette, se specificato, di caricare il programma che intendete rallentare dal drive <B:>. Questa è una comodità che evita, ogni volta che si cambia programma, di dover togliere anche il disco con il bootslow. Se il drive <B:> non esiste, bootslow considera come un errore la selezione. Si può specificare anche il drive <A:>, anche se scarsamente utile. Attenzione: bisogna specificare solo la lettera corrispondente al drive (cioè A oppure B): la lettera C non è accettata.

Esempi:
Bootslow 5000
(rallenta con valore 5000 e carica dal drive <A:>)
bootslow 4000 e 30000
(rallenta con valore 4000, segmento=30000 e carica dal drive <A:>)
bootslow 4500 a
(rallenta con valore 4500, e carica dal drive <A:>)
bootslow 3500 20000 b
(rallenta con valore 3500, segmento=20000 e carica dal drive <B:>)
ecc.

Slowdown: anche questo si carica da DOS, e anche questo richiede un parametro, ma non effettua il bootstrap. Nota che questo programma riserva memoria per la routine rallentatrice, impedendo qualsiasi interferenza con i programmi in esecuzione (che non facciano un uso poco ortodosso della memoria, naturalmente).

Il parametro è sempre un numero da 1 a 65535, dove 1=minimo rallentamento, 65535=massimo rallentamento.

È disponibile, presso la redazione, il disco con il programma pubblicato in questa rubrica. Le istruzioni per l'acquisto e l'elenco degli altri programmi disponibili sono a pag. 219.

Bootslow

```
/*   bootslow    */

#include<stdio.h>
#include<stdlib.h>
#include<dos.h>

func(long k,long k1,int drv)

{
        unsigned char i;

        unsigned char code[53]={ 0xfb,0x06,0x1e,0x50,0x52,0x51,0x53,0x56,
                                 0x57,0x55,0x2b,0xc0,0xb8,0x01,0x00,0xbb,
                                 0x01,0x00,0xba,0x01,0x00,0xb9,0x00,0x00,
                                 0x40,0x03,0xd8,0xd1,0xe2,0xe2,0xf9,0x5d,
                                 0x5f,0x5e,0x5b,0x59,0x5a,0x58,0x1f,0x07,
                                 0xea,0x00,0x00,0x00,0x00,0xfb,0xb2,0x01,
                                 0xea,0x00,0x00,0x00,0x00 );

        for(i=0;i<53;++i)
                pokeb(k1,i,code[i]);            /* copia la routine in memoria */

        pokeb(k1,22,k%256);                     /* modifica il fattore di      */
        pokeb(k1,23,k/256);                     /* rallentamento nella routine */

        pokeb(k1,41,peekb(0x0,0x20));           /*                             */
        pokeb(k1,42,peekb(0x0,0x21));           /* aggancia la fine della      */
        pokeb(k1,43,peekb(0x0,0x22));           /* routine con l'inizio del    */
        pokeb(k1,44,peekb(0x0,0x23));           /* codice dell'interrupt 8     */

        if(drv)
        {
                pokeb(k1,49,peekb(0x0,0x4c)); /*                             */
                pokeb(k1,50,peekb(0x0,0x4d)); /* usa un altro "pezzo" di co- */
                pokeb(k1,51,peekb(0x0,0x4e)); /* dice per modificare         */
                pokeb(k1,52,peekb(0x0,0x4f)); /* l'interrupt 13              */

                pokeb(0x0,0x4c,45);           /*                             */
                pokeb(0x0,0x4d,0);            /* e sposta il puntatore       */
                pokeb(0x0,0x4e,k1%256);       /* all'interrupt 13 in modo    */
                pokeb(0x0,0x4f,k1/256);       /* che punti al nostro codice  */
        }

        disable();
        pokeb(0x0,0x20,0);                      /*                             */
        pokeb(0x0,0x21,0);                      /* sposta il puntatore allo    */
        pokeb(0x0,0x22,k1%256);                 /* interrupt 8 in modo che     */
        pokeb(0x0,0x23,k1/256);                 /* punti alla nostra routine   */
        enable();

        pokeb(0,0x74,0xa4);                     /* modifica il puntatore       */
        pokeb(0,0x75,0xf0);                     /* ai parametri video in modo  */
        pokeb(0,0x76,0);                        /* che punti ai parametri vi-  */
        pokeb(0,0x77,0xf0);                     /* deo del BIOS ( questa ope-  */
                                                /* razione è necessaria sul    */
                                                /* mio e su altri computers:   */
                                                /* su qualche computer non è   */
                                                /* necessaria )                */
                                                /* se il programma non funzio- */
                                                /* na provate a togliere que-  */
                                                /* ste quattro righe           */
}


main(argc,argv)
int argc;
char *argv[];

{
        int i,flag,drv;
        long k,k1,flag1,flag2,flag3;
        union REGS inregs,outregs;
        drv=0xff;
        if(argc>1)
        {
                flag=1;
                for(i=0;i<160 && argv[1][i]>=48 && argv[1][i]<=57;++i);
                if(argv[1][i]=='\0')
                        k=strtol(argv[1],NULL,10);
                else
                        flag=0;
                if(argc>2 && flag)
                {
                        for(i=0;i<160 && argv[2][i]>=48 && argv[2][i]<=57;++i);
                        if(argv[2][i]=='\0')
                        {
                                k1=strtol(argv[2],NULL,10);
                                if(argc<=3)
                                        drv=0;
```

(continua a pag. 200)

(segue da pag. 199)

```
                }
                else
                if((argv[2][0]=='a' || argv[2][0]=='b' ||
                    argv[2][0]=='A' || argv[2][0]=='B')  &&
                    argv[2][1]=='\0')
                {
                        drv=tolower(argv[2][0])-'a';
                        if(argc<=3)
                                k1=0x9000;
                }
                else
                        flag=0;
                if(drv==0xff && flag && argc>3)
                {
                        if((argv[3][0]=='a' || argv[3][0]=='b' ||
                            argv[3][0]=='A' || argv[3][0]=='B')  &&
                            argv[3][1]=='\0')
                                drv=tolower(argv[3][0])-'a';
                        else
                                flag=0;
                }
                else
                if(flag && argc>3)
                {
                        for(i=0;i<160 && argv[3][i]>=48 &&
                                         argv[3][i]<=57;++i);
                        if(argv[3][i]=='\0')
                                k1=strtol(argv[3],NULL,10);
                        else
                                flag=0;
                }
        }
        else
        if(flag)
        {
                k1=0x9000;
                drv=0;
        }
}
if(drv==1)                      /* controlla l'esistenza del drive b: */
{
        flag1=inportb(0x61);
        flag2=flag1 | 8;
        outportb(0x61,flag2);
        flag3=inportb(0x62);
        if(!(flag3 & 12))
                flag=0;
        outportb(0x61,flag1);
}
if(argc==1 || !flag || k<=0 || k > 65535L || k1<0 || k1> 65535L)
{
        puts("\n\nPROGRAMMA RALLENTATORE:");
        puts("\nQuesta utility rallenta l'esecuzione dei programmi;");
        puts("USO: bootslow numero [segmento][drive];");
        puts("Devi specificare un numero da 1 a 65535 ");
        puts("(1=minimo rallentamento, 65535=massimo rallentamento)");
        puts("La routine rallentatrice è caricata all'indirizzo");
        puts("(hex) 9000:0000; Puoi opzionalmente specificare un ");
        puts("nuovo indirizzo di segmento come secondo argomento ");
        puts("(un numero da 1 to 65535); Default=36864 ( hex 9000 );");
        puts("Ma non devi curarti di specificare alcun segmento se ");
        puts("il programma funziona correttamente ");
        puts("(anche se non si rallenta).");
        puts("Per default, il bootstrap è fatto dal drive a:");
        puts("Se lo vuoi dal drive b:, scrivi 'b' come ultimo ");
        puts("argomento o anche 'a', se vuoi specificare il drive ");
        puts("esplicitamente.");
        puts("Nota: se non possiedi il drive b:, e scrivi 'b', sarà ");
        puts("considerato un errore.");
        puts("\nCopyright Nick Trio software 1988, Pisa, Italy.\n");
}
else
{
        func(k,k1,drv);
        puts("\n\nprogramma rallentatore in memoria.");
        puts("\nCopyright Nick Trio software 1988, Pisa, Italy.");
        while(kbhit())
                getch();
        if(drv)
                puts("\nInserisci il disco di sistema in drive B:");
        else
                puts("\nInserisci i disco di sistema in drive A:");
        puts("e premi un tasto quando sei pronto...");
        while(kbhit())
                getch();
        getch();
        puts("\a");
        int86(0x19,&inregs,&outregs);  /* esegui il caricamento */
}
}
```

Listato della routine che esegue il boot dopo aver rallentato il computer.

I malfunzionamenti che si possono verificare sono fondamentalmente di due tipi:
1) non si ha alcun rallentamento, ma il programma che si intendeva rallentare funziona correttamente: in genere i programmi professionali difficilmente causano questi tipo di malfunzionamento.
2) Veri e propri crash del sistema: non resta che resettare il computer.

Notare che solo nel secondo caso si può pensare a un malfunzionamento del programma rallentatore; nel primo caso è il programma da rallentare che spontaneamente disabilita la routine rallentatrice.

```
GIOCHI CHE RALLENTA:

Non "DOS":

 1) MOON PATROL
 2) COSMIC CRUSAIDER
 3) DIGGER
 4) DONKEY KONG
 5) PAC MAN  "ATARI"
 6) BIG TOP
 7) DEFENDER
 8) PITSTOP II
 9) ZAXXON
10) ROBOTRON


"DOS":

1) BUCK ROGERS
2) SPACE INVADERS
3) ASTRO
4) PACKMAN
5) CHOMP
6) CENTIPEDE
7) TENNIS IMAGIC
8) THE DAM BUSTER
9) FLIGHTMARE
10)SEA DRAGON
11)MEAN 18 (GOLF)
12)ALLEY CAT
13)STRIKER
14)MACK
15)ROUND 42
16)MASTER BLASTER
17)PARATROOPER


GIOCHI CHE NON RALLENTA:

Non "DOS":

1) DIG DUG
2) DECATHLON
3) BURGER TIME
4) NIGHT STALKER
5) J-BIRD
6) FLIGHT SIMULATOR
7) THE JET
8) CONGO BONGO
9) SHAMUS
10)FROGGER
11)CONQUEST
12)BOLDER DASH


"DOS":

1) MONTEZUMA'S REVENGE
2) SPACE WAR
```

Presumo che alcuni dei programmi «DOS» fossero inizialmente «non DOS», cioè protetti, quindi è possibile che circoli per questi anche la versione protetta (o viceversa); comunque non dovrebbe cambiare niente.

I programmi professionali in genere funzionano, anche se ha poco senso rallentarli.

Esempio: il Turbo C è uno di quei programmi che fanno un uso poco ortodosso della memoria, e infatti se si carica un qualsiasi programma residente dopo aver caricato il Turbo C, il compilatore stesso si blocca al momento della compilazione! (soluzione del problema: è sufficiente caricare i programmi residenti prima del Turbo C).

Principio di funzionamento

Forse non tutti sanno che il PC possiede 8 livelli di priorità per gli interrupt di sistema, e che il livello 0 gestisce l'interrupt 8, chiamato nel Technical Reference Manual «timer interrupt». Questo viene chiamato 18,2 volte al secondo e effettua varie operazioni, ad esempio incrementa un contatore usato per mantenere l'ora corrente a computer acceso. Ma quello che ci interessa è che questo interrupt viene eseguito qualunque sia il programma in esecuzione. Io ho semplicemente complicato il codice dell'interrupt aggiungendo un ciclo che non fa niente, oltre a rallentare l'esecuzione dell'interrupt, e quindi anche del programma in esecuzione.

I programmi sono stati realizzati in turbo C, e forse dovranno essere leggermente modificati per essere compilati con altri compilatori; notare ancora che entrambi i programmi contengono una matrice con il codice in linguaggio macchina (un semplice loop) che verrà lasciato in memoria e che effettua il rallentamento. È stato usato il modello di memoria «TINY». In particolare, «SLOWDOWN» DEVE essere compilato in modo «TINY» (altrimenti non funziona).

Inoltre «SLOWDOWN» DEVE essere trasformato in un comando (estensione «.COM») usando l'utility DOS «EXE2BIN».

Un'ultima cosa: il programma non è stato fatto direttamente in linguaggio macchina perché... non so usare il Microsoft Assembler! Anzi, il ciclo in linguaggio macchina è stato letteralmente copiato da un libro che non ho avuto ancora il coraggio di studiare; ho soltanto aggiunto qualche particolare, come salvataggio dei registri e istruzioni di salto; quindi mi scuso per l'incomprensibilità e la prolissità dei programmi e vi auguro buon divertimento!

Riferimenti

IBM technical reference manual
Il libro del programmatore
di Peter Norton
Assembler for the IBM PC and PC-XT
di Peter Abel

Turbo Utility

di Roberto Bettati - Marcallo (MI)

In un periodo caratterizzato dal frenetico mutamento del mercato e da prodotti totalmente innovativi mi è sembrato quantomeno doveroso cercare di risollevare le sorti del buon PC IBM, scrivendo un paio di procedure in un linguaggio ormai «standard»: il Turbo Pascal. Ho privilegiato questo linguaggio rispetto ad altri di grande successo (C e Basic) prevalentemente per la chiara leggibilità dei listati; risulterà infatti chiaro dalla lettura dei sorgenti che tramite qualche piccola modifica i due programmini potranno essere facilmente adattati anche per altre macchine. Ma veniamo ora ad una sommaria descrizione delle procedure CINPUT e MENU: la prima, il cui listato non viene pubblicato data l'eccessiva lunghezza, è destinata ad evitare quei tremendi errori di inserimento dati legati alla lunghezza strettamente definita delle stringhe in Pascal; chi come me viene da un'esperienza Basic (ove questo problema non si pone) sa cosa significa digitare delle importanti informazioni ed accorgersi nel proseguo della elaborazione che parte di queste non sono state prese in considerazione proprio perché di lunghezza superiore a quella definita per la variabile stringa assegnata. Dopo un paio di collassi mi sono deciso a creare questa procedura che inserita in un qualsiasi programma si sostituisce alla predefinita READ e permette un input localizzato in un qualsiasi punto del video vincolando il numero di caratteri costituenti la stringa ad un parametro passato. Vi risulterà molto utile anche nel caso in cui si vogliano inserire numeri in variabili numeriche in quanto tramite il parametro «tipo» si può decidere l'insieme di caratteri validi tra l'alfanumerico e lo strettamente numerico. Per una spiegazione tecnica di questa prima procedura e per le relative modalità d'impostazione dei parametri per un corretto funzionamento vi rimando alle numerose remark delle quali è costellata e spendo ancora due parole sulla trasportabilità della stessa su altri compilatori standard: purtroppo essa si avvale dell'uso di variabili assolute in quanto solo così è stato possibile renderla compatibile con tutti i formati delle stringhe del programma (o procedura) chiamante e, a quanto mi risulta, tali variabili sono una prerogativa del solo Turbo Pascal cosicché tradurla per un altro compilatore risulterà piuttosto difficile se non impossibile.

Passiamo ora a visitare MENU che sarà forse meno utile della precedente ma vi aiuterà certamente a dare un certo tono di professionalità ai vostri programmi: la sua mansione è quella di creare una barra orizzontale di lunghezza variabile in una qualsiasi posizione video, che si possa muovere verticalmente evidenziando con colori a vostra scelta tutto ciò che si trova al di sotto della stessa e ripristinando allo stato precedente la pozione evidenziata ogni qual volta che la barrà verrà spostata; in definitiva risulterà chiaro che il suo utilizzo primario è da porre in relazione ai menu anche perché è presente un parametro (opz) che ritorna in pratica il numero dell'opzione scelta (cioè il numero della barra attiva quando si preme Return in relazione alla sua posizione rispetto a quella di partenza a cui è assegnato il valore 1; quella di partenza è posta come limite alto). In questa procedura è interessante notare l'utilizzo dell'istruzione MEM (equivalente al PEEK e al POKE del Basic) che permette di leggere e scrivere direttamente nella memoria video del PC (Hex B800: offset) ad una notevole velocità; proprio per questa ragione raccomando attenzione ai possessori di altre macchine per quanto riguarda gli indirizzi di memoria che quasi sempre differiscono tra i diversi computer.

Turbo Utility

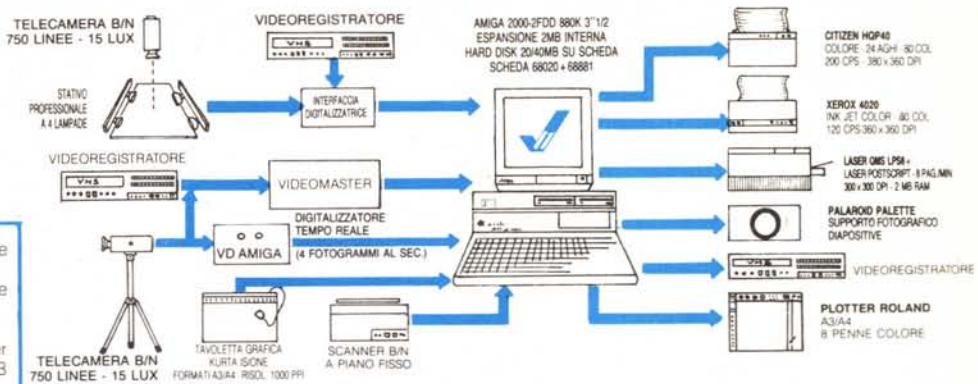
```
PROGRAM example(INPUT,OUTPUT);
VAR
  opzione : INTEGER;


{ ------------------------------------------------------------------------
  NOME DELLA PROCEDURA: menu'.
  FUNZIONE: visualizza delle barre orizzontali che evidenziano il testo sottostante e permette la scelta tra piu' opzioni
            usando le freccie dei tasti cursore.
  PARAMETRI FORMALI: il parametro variabile 'opz' contiene il numero della scelta che varia tra 1 e numvoci; il secondo ed
                     il terzo parametro, rig e col, indicano le coordinate del primo carattere della barra di scelta della
                     prima voce mentre numvoci e lungvoce indicano rispettivamente il raggio di movimento verticale della
                     barra e la sua lunghezza in caratteri; infine colore e coloretesto specificano il colore della barra ed
                     il colore del testo al suo interno.
  LUNGHEZZA DELLA PROCEDURA IN LINEE: 89 rem comprese.                    }

PROCEDURE menu(VAR opz:INTEGER;rig,col,numvoci,lungvoce,colore,coloretesto:INTEGER);
TYPE
  salvariga=^carattere;                 { definizione di un tipo puntatore ad un record che contiene codici ascii }
  carattere=RECORD                      { e attributo colore dei caratteri letti dal video.                       }
      car_let,car_attr : BYTE;
      next             : salvariga
      END;
VAR
  salva,salvavec,salvain : salvariga;
  tasto                  : CHAR;
  colorattr,i            : INTEGER;
  giusto,err             : BOOLEAN;

PROCEDURE riscriviriga;          { procedura che ripristina una riga nel momento in cui la barra di scelta viene spostata. }
BEGIN
FOR i:=0 TO lungvoce-1 DO
  BEGIN
  MEM[$B800:((160*(rig-1))+(col+i-1)*2)]:=salvain^.car_let;
  MEM[$B800:((160*(rig-1))+((col+i-1)*2)+1)]:=salvain^.car_attr;
  salvain:=salvain^.next
  END
END;

BEGIN
err:=FALSE;
IF (col+lungvoce>80) OR (rig+numvoci>24) THEN        { vengono rilevati possibili errori ed in caso ne esistano la procedura }
  err:=TRUE;                                         { non viene eseguita ed il controllo passa al programma chiamante.      }
IF NOT(err) THEN
  BEGIN
  colorattr:=colore*16+coloretesto;       { viene approntato il codice-attributo per i caratteri della barra di scelta. }
  opz:=1;
  REPEAT
    FOR i:=0 TO lungvoce-1 DO
      BEGIN
      NEW(salva);
      IF i=0 THEN
        salvain:=salva
      ELSE
        salvavec^.next:=salva;
      salva^.car_let:=MEM[$B800:((160*(rig-1))+(col+i-1)*2)];        { legge il codice ascii del carattere letto sul video. }
      salva^.car_attr:=MEM[$B800:((160*(rig-1))+((col+i-1)*2)+1)];   { legge l'attributo-colore del carattere letto.       }
      salva^.next:=NIL;
      MEM[$B800:((160*(rig-1))+(col+i-1)*2)]:=salva^.car_let;
      MEM[$B800:((160*(rig-1))+((col+i-1)*2)+1)]:=colorattr;
      salvavec:=salva
      END;
    giusto:=FALSE;
    REPEAT              { ripete finche' non viene premuto il RETURN oppure i tasti 2 o 8 del paddle numerico. }
    READ(KBD,tasto);              { viene letto un carattere da console: se a questo carattere e' associato un codice }
    IF ORD(tasto)=27 THEN         { esteso in 'tasto' viene messo il numero 27 quindi lo statement IF fa si che venga }
      BEGIN                       { letto in 'tasto' il secondo codice in modo tale che possa poi essere interpretato }
      READ(KBD,tasto);            { dalla successiva CASE.    }
      CASE ORD(tasto) OF
        72 : IF opz>1 THEN        { premuto il tasto 'freccia in su' }
               BEGIN
               riscriviriga;
               opz:=opz-1;
               rig:=rig-1;
               giusto:=TRUE
               END;
        80 : IF opz<numvoci THEN  { premuto il tasto 'freccia in giu' }
               BEGIN
               riscriviriga;
               opz:=opz+1;
               rig:=rig+1;
               giusto:=TRUE
               END
      END
    END
    ELSE
      IF ORD(tasto)=13 THEN giusto:=TRUE;
    UNTIL giusto;
  UNTIL ORD(tasto)=13
  END
END;

{ ---------------- FINE DELLA PROCEDURA MENU' ------------------------------ }

BEGIN       (* corpo del programma esempio *)
CLRSCR;
GOTOXY(10,10);
WRITE('LETTURA');
GOTOXY(10,11);
WRITE('STAMPA');
GOTOXY(10,12);
WRITE('AGGIORNA');
menu(opzione,10,9,3,10,1,15);
GOTOXY(10,17);
WRITE('La voce scelta e'' la numero: ',opzione)
END.
```

Sorgente Turbo Pascal del programma MENU.

AMIGA WORKSTATIONS GRAFICHE AMIGA

ENLOCK PROFESSIONALE PER AMIGA

TUTTI I PREZZI SONO IVA INCLUSA



VIDEOMASTER

CARATTERISTICHE
- Ingressi Video Composito
- Ingresso RGB Computer
- Uscita Video Composito
- Uscita RF
- Uscita RGB + Sinc.
- Alimentazione interna

* Controllo e processo segnale video
* Contrasto - Colore - Saturazione
* Tastiera multifunzione
* Mix Ingressi 1 & 2
* Esclusione video in o computer
* Fornitura sui colori RGB (Croma-Key)

HARDWARE

MIGA 500	930.000
MIGA 500 + Monitor 1084	1.550.000
MIGA 2000 senza monitor	1.950.000
MIGA 2000 2 drive 3"1/2	2.190.000
SPANSIONE 512K interna A500	260.000
SPANSIONE 1MB esterna A1000	Telef.
SPANSIONE 2MB esterna A500/A1000	Telef.
SPANSIONE 2MB interna A2000	700.000
SK DRIVE 3"1/2 esterno A500/A1000	290.000
SK DRIVE 3"1/2 interno A2000	250.000
ARD DISK 20MB EST. A500/A1000	1.250.000
ARD CARD 20MB SCSI A2000	1.250.000
ARD CARD 20MB 20MB SCSI A2000	750.000
ARD CARD 40MB MS-DOS A2000	950.000
stema a Cartridge da 12MB removibili della Kodak + Cartridge (60 MB)	2.650.000
CHEDA JANUS XT A2000	850.000
CHEDA JANUS AT A2000	1.550.000
T SOSTITUZIONE MOTOROLA 68010	90.000
CHEDA 68020 + 68881 16MHZ	1.850.000
IGI-VIEW ORIGINALE	290.000
MIGA-EYE A500/A1000/A2000	130.000
D AMIGA FRAMEGRABBER	750.000
D 2000 DIGITALIZZATORE COLORE IN CVBS 500/A1000/A2000	1.150.000
ELECAMERA SECURIT T-979	550.000
STATIVO PROFESSIONALE 4 LAMPADE	350.000
AMIGA SOUND A500/A1000/A2000	150.000
INTERFACCIA MIDI A500/A1000/A2000	90.000
GENLOCK PROFESSIONALE VIDEOMASTER	1.390.000

TAVOLETTE GRAFICHE KURTA:

PENMOUSE (6" x 9" 200 PPI	250.000
SERIE IS 8,5" x 11" 1000 PPI	840.000
SERIE IS 12" x 12" 1000 PPI	1.040.000
SERIE IS 12" x 17" 1000 PPI	1.740.000
PENNA A DUE BOTTONI	290.000
CURSORE A 4 BOTTONI	290.000
CAVO E SOFTWARE PER AMIGA	110.000

STAMPANTI:

PANASONIC KX-P1081 80 COL 120 CPS	550.000
NEC P2200 80 COL 216 CPS 24 AGHI	950.000
NEC P6 80COL 216CPS 24 AGHI	Telef.
NEC P6 KIT COLORE	Telef.
NEC P7 136 COL 216 CPS 24 AGHI	1.650.000
NEC P7 136 COL 216 CPS 24 AGHI	1.790.000
CITIZEN HQP40-24 AGHI	1.350.000
CITIZEN HQP40-KIT COLORE	1.550.000
XEROX 4020 INK JET COLORE	3.450.000
OKI LASER LL6 PPM	3.850.000
LASER QMS LPS8+POSTSCRIPT	Telef.
HARD COPIER SHINKO	Telef.
POLAROID PALETTE PER AMIGA	3.450.000

SOFTWARE ORIGINALE:

FINITY SOFTWARE:

OT LICKS	55.000
HAKESPEARE	289.000
ALILEO 2.0	89.000

M INC:

HE SURGEON	65.000

ICROPROSE:

LENT SERVICE	55.000
OEBIUS	49.000
LTIMA III	49.000

ICROMAGIC:

ORMS IN FLIGHT	110.000

ICROILLUSIONS:

RE POWER	35.000
YNAMIC CAD	690.000
HOTON PAINT	120.000

INDSCAPE:

EFENDER OF THE CROWN	59.000
HE THREE STOOGES	65.000
ALLEY PROJECT	69.000
EJA VU	69.000
NINVITED	69.000

EWTEK:

IGI-PAINT	79.000

XXY INC

AXIPLAN 500	190.000
AXIPLAN PLUS	250.000

SYGNOSIS:

ARBARIAN	55.000
BLITERATOR	55.000

UBLOGIC:

LIGHT SIMULATOR	75.000
ET	75.000
CENERY DISK 7	39.000

UMA:

V SHOW	129.000

GOLD DISK:

PROFESSIONAL PAGE	445.000
PAGESETTER ITAL.	210.000

ACTIVISION:

HACKER II	29.500
THE ART OF CHESS	29.500
SHANGHAI	29.500
BORROWED TIME	65.000
LITTLE COMPUTER PEOPLE	35.000
MINDSHADOW	35.000
TASS TIMES	35.000
PORTAL	55.000
GEE BEE AIR RALLY	55.000

AEGIS:

ANIMATOR	175.000
ARAZOK'S TOMB	49.000
AUDIOMASTER	75.000
DIGA	99.000
DRAW PLUS	320.000
IMPACT	110.000
SONIX	99.000
VIDEOTITLER	125.000
PORT OF CALL	129.000
VIDEOSCAPE 3D	299.000

BYTE BY BYTE:

SCULPT 3D	129.000
ANIMATE 3D	199.000

COMMODORE:

MIND WALKER	69.000
TEXTCRAFT PLUS	145.000
SUPERBASE PERSONAL	190.000
SUPERBASE PROFESSIONAL	390.000
LOGISTIX	120.000

DISCOVERY:

ARKANOID	75.000

EPYX:

DESTROYER	29.000
WINTER GAMES	29.000
WORLD GAMES	29.000

NEW HORIZONS:

PROWRITE	175.000

RIGHT ANSWER GROUP:

THE DIRECTOR	89.000

METACOMCO:

MCC PASCAL	139.000
ASSEMBLER LANGUAGE	139.000

EAGLE SOFTWARE:

BUTCHER 2.0	49.000

ELECTRONIC ARTS:

ADVENTURE C. SET	38.000
ARTIC FOX	29.500
BARD'S TALE I	29.500
CHESSMASTER 2000	29.500
INSTANT MUSIC	33.000
MARBLE MADNESS	29.500
SKYFOX	29.500
TEST DRIVE	33.000
DE LUXE MUSIC C.S.	94.000
DE LUXE PAINT II	99.000
DE LUXE PRINT	90.000
DE LUXE VIDEO 1.2	109.000
FERRARI FORMULA 1	38.000
RETURN TO ATLANTIS	38.000

PROGRESSIVE P. &S:

PIXMATE	94.000
MATH ANIMATION	89.000

MASTERTRONIC:

FEUD	19.900
KIKSTART II	19.900
NINJA MISSION	19.900
SPACE RANGER	19.900

FIREBIRD:

BUBBLE BOBBLE	29.000

MIRRORSOFT:

DARK CASTLE	49.000
KING OF CHICAGO	59.000
TETRIS	39.000

ANCO:

FLIGHT PATH 737	19.900
JUMP JET	19.900
KARTING GRAND PRIX	19.900
LAS VEGAS	19.900
STRIP POKER	19.900
THAI BOXING	19.900
XR 35	19.900

RAINBIRD:

GOLDEN PATH	79.000
JINXTER	49.000

CDS:

FOOTBALL FORTUNE	49.000

MELBOURNE HOUSE:

ROADWARS	39.000
XENON	39.000

MIMETICS:

PRO MIDI STUDIO	230.000
3 DEMON	125.000

IMPULSE:

SILVER	155.000

SYNDESIS:

INTERCHANGE	99.000

TAURUS:

ACQUISITION	450.000
X-CAD	950.000

PERSONAL COMPUTER

LINEA HITECH PERSONAL COMPUTER

LINEA XT 4.7/10 MHZ

XT-HT 256K 1FDD 360K TAST. AVANZ.	850.000
XT-HT 256K 2FDD 360K TAST. AVANZ.	1.050.000
XT-HT 256K 1FDD 360K HD 20MB TAST. AVANZ.	1.550.000

LINEA AT 10MHZ 0 WAIT STATE

AT-HT 512K 1FDD 1.2MB TAST. AVANZ.	1.950.000
AT-HT 512K 1FDD 1.2MB 1 HD 20MB TAST. AVANZ.	2.550.000
AT-HT 512K 1FDD 1.2MB 1 HD 85MB TAST. AVANZ.	3.150.000
AT-HT 512K 1FDD 1.2MB 1 HD 140MB TAST. AVANZ.	4.750.000

LINEA 386 16-20 MHZ

TOWER 2MB 1FDD 1.2 MB 1 HD 40MB TAST. AVANZ.	6.280.000
TOWER 2MB 1FDD 1.2MB 1 HD 85MB TAST. AVANZ.	7.750.000
TOWER 2MB 1FDD 1.2MB 1 HD 140MB TAST. AVANZ.	9.850.000

SCHEDE PC

SCHEDA SERIALE	58.000
SCHEDA PARALLELA CENTRONICS	36.000
SCHEDA EGA AUTOSWITCH	490.000
SCHEDA FAX	1.450.000
SCHEDA COPY CARD II	160.000

HARD DISK

HARD DISK 20MB + CONTROLLER	590.000
HARD DISK 40MB + CONTROLLER	950.000
HARD CARD 20MB	690.000
HARD CARD 40MB	1.050.000

COPROCESSORI MATEMATICI

INTEL 8087 6MHZ	250.000
INTEL 8087 8MHZ	380.000
INTEL 80287 6MHZ	390.000
INTEL 80287 8MHZ	580.000
INTEL 80287 10MHZ	690.000
INTEL 80387 16MHZ	1.250.000

MONITOR

PHILIPS 7502/7513 MONOCROMATICO 12"	180.000
PHILIPS 9073 EGA COLORE 14"	850.000
PHILIPS 8833 COLORE 14"	550.000
MULTISYNC MONOCROMATICO	550.000
MULTISYNC COLORE	1.250.000

MODEM

ESSEGI 1200M 300/1200 BAUD V21/V22 FULL DUPLEX	360.000
ESSEGI 1203M 300/1200/75 V21/V23 VIDEOTEL	420.000
ESSEGI 2400M 1200/2400 BAUD V22/V22 BIS	750.000
ESSEGI 1200C CARD	360.000

TELEFAX

TELEFAX BACON-TELEFONO G2/G3 FORMATO A4	2.250.000

Pix computer

PIX COMPUTER S.R.L.
VIA F. D'OVIDIO, 6C
TEL. 06/8293507-825731
00137 ROMA
COMPUTER & Co.
P. IVA 08309630583

DISPONIBILE LATTICE C COMPILER VERS. 4.0 LIT. 250.000

VENDITA PER CONTRASSEGNO SU TUTTO IL TERRITORIO NAZIONALE. OFFERTE E PREVENTIVI SU WORKSTATIONS GRAFICHE COMPLETE. SETTORI CAD 2D/CAD 3D/ANIMAZIONI 3D/DIGITALIZZAZIONI/VIDEO BROADCAST/DESKTOP PUBLISHING. SI INVIANO A RICHIESTA SCHEDE TECNICHE PRODOTTI. SCONTI PER RIVENDITORI QUALIFICATI.