a cura di Valter Di Dio

Velocissima introduzione perché i programmi di questo mese sono molto lunghi (come testo). Per gli aficionados del Basic un gestore di menu Pull-Down scritto il Turbo Basic che darà un aspetto professionale a tutti i vostri programmi. Per i seguaci di Wirth tre routine di gestione della CGA che eliminano il fastidioso effetto neve nelle operazioni di refresh del video

Pulldown Menu

di Daniele Bufarini - Rieti

Era da molto tempo che avevo l'intenzione di realizzare una routine che mi permettesse di sviluppare programmi con un'interfaccia utente simile a quella del MacIntosh. Il risultato di un mio lungo lavoro è l'insieme di routine che vi propongo sotto il nome di Pulldown Menu, che se non implementano completamente un'interfaccia utente uguale a quella del MacIntosh, offrono la possibilità di realizzare dei menu pulldown gestibili sia da mouse, che deve essere compatibile Microsoft sia, in mancanza di quest'ultimo, da tastiera. È chiaro comunque che i più volenterosi potranno, fornita loro questa base, ampliare ulteriormente le routine aggiungendo tutto ciò che a loro parere sembrerà mancare (come, ad esempio, il colore che non è stato implementato in quanto il programma è stato inizialmente sviluppato con una scheda Hercules). Detto questo, faccio notare che Pulldown Menu è stato scritto in Turbo Basic, e che comunque con poco sforzo potrà essere modificato per essere compilato anche con il QuickBasic della Microsoft.

Una volta incluso con una «metaistruzione» $INCLUDE "PULLDOWN.INC" nel proprio programma e richiamato con l'istruzione CALL Menu (...,...,...) Pulldown Menu controllerà l'esistenza del mouse Microsoft (ma si può benissimo usare un mouse compatibile od un programma per emularlo con il proprio mouse; io, ad esempio, ho usato un emulatore del mouse Microsoft su di un Mouse System Mouse); nel caso che il mouse sia rintracciato, Pulldown Menu si predisporrà ad operare con questo, altrimenti il programma userà la tastiera: ci si potrà muovere sulle varie opzioni con i tasti freccia destra e sinistra, e selezionarle con il Return; con i tasti freccia su e giù si potrà scegliere una delle opzioni del menu pulldown aperto, mentre con il tasto Esc o con una delle due frecce si potrà uscire da un menu pulldown.

Come usare Pulldown Menu in un programma

Per poter usare Pulldown Menu in un proprio programma, occorre fare tre cose:

1 - includere, come detto sopra, Pulldown Menu nel programma che state sviluppando con la metaistruzione $INCLUDE.

2 - Definire dei data che conterranno i nomi dei menu; e le relative opzioni. Per fare ciò si deve mettere una label «Menu:» sopra la riga contenente i data relativi ai nomi dei menu, mentre per le opzioni si deve mettere una label «Opzioni:».

Passiamo ora all'esempio di figura A in cui il numero che segue l'istruzione DATA relativa alle voci del menu indica il numero dei menu che si intende creare (in questo caso 2), mentre i numeri che si trovano nei data relativi alle opzioni indicano il numero delle voci che ogni menu deve contenere (nell'esempio il numero 5 indica le opzioni del menu File, il numero 3 quelle relative al menu Edit, ecc.).

3 - Effettuare una chiamata alla SUB Menu con l'istruzione:

```
CALL Menu (Xmx%, Xmy%, Xms%, Menu, Opzione)
```

dove le variabili Xmx% e Xmy% rappresentano le coordinate X ed Y a partire dalle quali verrà stampata la riga con il menu, mentre la variabile Xms% determina lo spazio tra le varie voci del menu. Nell'ultime due variabili, Menu ed Opzione, viene ritornato, come è facile intuire, il menu e l'opzione selezionata (a questo proposito vorrei far notare che mentre alle variabili Xmx%, Xmy%, Xms% può essere sostituita una costante numerica, non si possono cambiare, se non intervenendo sul programma, il nome delle variabili Menu ed Opzione).

Esempio:

```
CALL Menu (6, 2, 4, Menu, Opzione)
```

Con questa istruzione Pulldown Menu stamperà la riga del menu a partire dalla riga 2, colonna 6, con un numero di 4 spazi tra una voce e l'altra.

A questo punto si è pronti per poter utilizzare Pulldown Menu nel proprio programma; un esempio di applicazione è riportato in figura B.

Descrizione delle routine

FN MaxLen (Num%, Menu)

Questa funzione ritorna il numero massimo di caratteri che ha l'opzione più lunga nel menu specificato dalla variabile Menu; la variabile Num% contiene il numero delle opzioni del menu specificato nella variabile Menu.

SUB Center (Var$, YCor, Colore)

Questa subroutine stampa la stringa contenuta nella variabile Var$ centrata sulla riga specificata nella variabile

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. 281

Figura A

```
Menu:
  DATA  2, File, Edit
Opzioni:
  DATA  5, Load, Save, Save as..., Os Shell, Quit  ' menu File
  DATA  3, Cut, Copy, Paste   ' menu Edit
```

YCor, con il colore indicato nella variabile Colore.

SUB Box (X1%, Y1%, X2%, Alt%, Mode)

Questa subroutine «disegna» un box dalle coordinate X1%, Y1% alla coordinata X2%, con un'altezza Alt%. Se la variabile Mode sarà True, allora il box avrà i quattro spigoli «smussati», altrimenti i due superiori saranno di questa forma «T». Questo perché la barra dei menu può essere circondata o meno da un box (nel primo caso allora la variabile Mode dovrà essere False).

FN Mouse

Questa funzione ritorna un valore True o False a seconda che il mouse sia installato o meno.

SUB GetS (X1%, Y1%, X2%, Alt%)

Questa subroutine prende una porzione dello schermo dalle coordinate X1%, Y1% alla coordinata X2% con un'altezza Alt%, e lo salva negli array Char% () e Style% (), per essere successivamente ripresa dalla subroutine simmetrica PutS.

SUB PutS (X1%, Y1%, X2%, Alt%)

Questa subroutine riscrive sullo schermo la zona precedentemente «catturata» dalla subroutine GetS.

FN Where (Y1, X1, X2)

Questa funzione scandisce la riga Y1 dalla posizione X1 a quella X2, scrivendo nell'array LocX () le coordinate X in cui iniziano e finiscono le singole voci della barra menu. Ritorna il numero delle voci trovate sulla riga Y1.

FN YMenu (X1%, Y1%, X2%, Alt%, Max%, Menu)

Questa funzione si occupa di gestire il menu pulldown aperto. Disegna un box dalle coordinate X1%-1, Y1% alla coordinata X2%+1 con un'altezza Alt%. La variabile Max% contiene il numero delle opzioni, mentre la variabile Menu il menu aperto; usa la tastiera e ritorna l'opzione scelta.

FN XMenu$ (X%, Y%, Spcl%, Max%)

Questa funzione stampa la barra dei menu alle coordinate X%, Y% con un numero Spcl% di spazi tra le singole voci della barra dei menu, il numero delle quali è contenuto nella variabile Max%; usa la tastiera e ritorna la voce scelta e la sua coordinata X, separata da CHR$(0).

FN YMouse (X1%, Y1%, X2%, Alt%, Max%, Menu)

Questa funzione è simmetrica alla FN YMenu, con la differenza che gestisce il mouse al posto della tastiera.

FN XMouse$ (Xmn, Ymn, Xmn1, Xmn2, Spcl, Max)

Questa funzione è simmetrica alla Fn XMenu$, con la differenza che gestisce il mouse al posto della tastiera.

Le variabili Xmn1 e Xmn2 (con valore fisso nel programma) sono i parametri (X1 e X2) da passare alla funzione FN Where.

FN Menu (Xmx%, Xmy%, Xms%, Menu, Opzione)

Questa funzione si occupa della gestione complessiva dei menu pulldown; le variabili Xmx%, Xmy%, Xms% hanno la stessa funzione delle prime tre nella funzione FN XMenu$. Ritorna il menu e l'opzione selezionata.

Figura B

```
' Esempio di applicazione di Pulldown Menu

$INCLUDE "PULLDOWN.INC"

CLS
CALL Box(1,1,80,2,%True): CALL Center("Prova",1,4): COLOR 15,0
FOR I=1 TO 20
  LOCATE I+3,1: PRINT STRING$(80,"!")  ' crea uno sfondo di "!"
NEXT
Xmx% = 4: Xmy% = 2: Xms% = 10: CALL Menu(Xmx%,Xmy%,Xms%,Menu,Opzione)
LOCATE 18,1
PRINT "Menu= "Menu" ("Title$(Menu)")";TAB(5);
PRINT "Opzione= "Opzione" ("Options$(Opzione,Menu)")"

Menu:
  DATA 4, File,Edit,Run,Compile
Opzioni:
  DATA 9, Load,New,Save,Write to,Main File,Directory,Change Dir,_
          Shell,Quit
  DATA 5, 12345,DDDD,RRRR,EEEE,0123456789012345678901234567890
  DATA 6, XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX,YY,Write to,_
          Main File,Directory,Change Dir
  DATA 4, Memory, Exe File, Com File, Chain
```

Ricordarsi che le voci della barra menu non possono avere degli spazi intermedi (ad esempio non è valida una voce del tipo «File System», mentre al contrario lo è una del tipo «File_System»).

Retrace

di Luca Padovano - Imola

Come è noto, una buona gestione dello schermo, cioè la possibilità di scrivere caratteri sottolineati, in reverse o a diversi colori, il tutto ad una certa velocità, è condizione irrinunciabile, se si vogliono scrivere programmi «seri».

Sfortunatamente la Write del Turbo Pascal non è esattamente lo strumento dei nostri sogni. Oltre a disinteressarsi completamente di attributi e colori, è terribilmente lenta nella maggior parte dei casi.

Come si può fare quindi a gestire lo schermo in Turbo Pascal senza vergognarsi di se stessi? Scartata come vile la filosofia del «chi si accontenta gode», rimangono due soluzioni al problema.

La prima è stata proposta negli scorsi numeri di MC da Pierluigi Panunzi; usando le routine del BIOS è possibile compiere tutte quelle operazioni proibite da Pascal. Il risultato è certamente di alto livello e con poche istruzioni è possibile ottenere effetti «professionali». Ma rimane la lentezza. Una routine che stampi alle coordinate X, Y, con un determinato attributo è veloce (o lenta) più o meno come la Write.

Rimane la «soluzione finale»: scrivere direttamente in memoria video.

Il video del PC funziona perché dentro al computer un certo numero di circuiti integrati dalle sigle strane operano in maniera tale da visualizzare lettere e punti. Questi integrati, vanno a formare una Scheda Video, o più precisamente un Display Adapter (adattatore di visualizzazione).

Le più comuni schede video in circolazione sono la scheda MDA (Monochrome Display Adapter) usata dai vecchi PC IBM, che può stampare solo testo, e la scheda CGA (Color Graphic Adapter) che può stampare sia in modo testo che in modo grafico. Almeno per quanto riguarda il modo testo, le altre schede funzionano come la MDA e/o come la CGA (vedi Hercules).

Queste due schede sono memory mapped; questo vuol dire che ogni carattere visualizzato, è anche conservato in una ben precisa area di memoria accessibile dall'utente. In particolare la MDA «legge» 4K a partire dalla locazione $B000:$0000, mentre la CGA dispone di ben 16K (troppa grazia!) a partire dalla locazione $B800:0000.

Ogni word a partire dall'offset 0 con-

Listato 1 - **MOVTOSCR. ASM**

```
0000                    CSEG    SEGMENT
                                ASSUME CS:CSEG
                                PUBLIC movtoscr
0000                    movtoscr PROC NEAR
0000  55                        push bp         ;salva i registri
0001  8B EC                     mov bp,sp
0003  1E                        push ds
0004  33 C0                     xor ax,ax
0006  8E C0                     mov es,ax
0008  26 8A 0E 0449             mov cl,es:[449h] ;controlla scheda
000D  B8 B000                   mov ax,45056    ;scheda IBM
0010  80 F9 07                  cmp cl,7        ;7 per IBM MDA
0013  74 03                     je  ibm_mda
0015  05 0800                   add ax,2048     ;scheda CGA
0018  8E C0             ibm_mda: mov es,ax      ;seg target
001A  33 FF                     xor di,di       ;ofs target
001C  C5 76 04                  lds si,[bp+4]   ;source
001F  B9 07D0                   mov cx,2000     ;lunghezza schermo
0022  BA 03DA                   mov dx,03dah    ;apro la porta....
0025  FA                        cli
0026  EC               check1 : in  al,dx
0027  24 01                     and al,1
0029  75 FB                     jnz check1
002B  EC                check2 : in  al,dx
002C  24 01                     and al,1
002E  74 FB                     jz  check2
0030  A5                do_it  : movsw          ;muove
0031  E2 F3                     loop check1     ;ripete
0033  FB                        sti             ;riabilita
0034  1F                        pop ds          ;resetta
0035  8B E5                     mov sp,bp
0037  5D                        pop bp
0038  C2 0004                   ret 4           ;bye bye!!
003B                    movtoscr ENDP
003B                    CSEG    ENDS
                                END
```

Listato 2 - **MOVTOMEM. ASM**

```
0000                    CSEG    SEGMENT BYTE PUBLIC
                                ASSUME CS:CSEG
                                PUBLIC movtomem
0000                    movtomem PROC NEAR
0000  55                        push bp         ;salva i registri
0001  8B EC                     mov bp,sp
0003  1E                        push ds
0004  33 C0                     xor ax,ax
0006  8E C0                     mov es,ax
0008  26 8A 0E 0449             mov cl,es:[449h] ;controlla scheda
000D  B8 B000                   mov ax,45056    ;scheda IBM
0010  80 F9 07                  cmp cl,7        ;eh si, è IBM MDA
0013  74 03                     je  ibm_mda
0015  05 0800                   add ax,2048     ;scheda CGA
0018  8E D8             ibm_mda: mov ds,ax      ;seg source
001A  33 F6                     xor si,si       ;ofs source
001C  C4 7E 04                  les di,[bp+4]   ;target
001F  B9 07D0                   mov cx,2000     ;lunghezza schermo
0022  BA 03DA                   mov dx,03dah    ;apro la porta
0025  FA                        cli             ;via gli interrupt
0026  EC               check1 : in  al,dx
0027  24 01                     and al,1
0029  75 FB                     jnz check1
002B  EC                check2 : in  al,dx
002C  24 01                     and al,1
002E  74 FB                     jz  check2
0030  A5                 do_it : movsw          ;muove
0031  E2 F3                     loop check1     ;ripete
0033  FB                        sti             ;riabilita
0034  1F                        pop ds          ;risetta
0035  8B E5                     mov sp,bp
0037  5D                        pop bp
0038  C2 0004                   ret 4           ;ciao, ciao...
003B                    movtomem ENDP
003B                    CSEG    ENDS
                                END
```

tiene il codice del carattere da stampare e il rispettivo attributo. Quindi abbiamo 25*80 caratteri + 25*80 attributi = 4000 byte; il conto torna.

I rimanenti 12K disponibili per la CGA servono per i modi grafici (ma è un'area di memoria che possiamo anche utilizzare per altri scopi!).

È facile capire che usando le istruzioni Pascal MEMW e MEM è possibile scrivere o leggere nel video in maniera del tutto ovvia. Per stampare la stringa «dato» alla posizione x, y (1<=x<=80, 1<=y<=25) basteranno poche righe di Turbo (vedi figura C) dove «W» è 8 per la CGA e 0 per la MDA.

Questo programmino è veloce e oltretutto occupa pochissimo spazio.

Ma c'è (ah destino!) il rovescio della medaglia; le schede CGA in modo testo soffrono di un grave male, il cosiddetto flickering (sfarfallìo) o «effetto neve».

Vediamo perché. Come ogni televisione che si rispetti, anche il monitor dispone di un «pennello elettronico» che va avanti e indietro per lo schermo e aggiorna il video. Un opportuno «controllore» del video si occupa di accendere e spegnere il pennello a seconda che un determinato bit sia 1 o 0 (e che quindi vada o no «acceso»). È così che si formano le immagini sul video.

Se la CPU tenta di accedere alla memoria video mentre il pennello sta lavorando, quest'ultimo si offende e crea l'effetto neve. Per evitare problemi con il pennello, basta dire alla CPU di scrivere in memoria solo durante il vertical retrace, cioè quel brevissimo intervallo di tempo (1.25 millisecondi) in cui il raggio, arrivato in fondo allo schermo, si deve riposizionare in cima al video per eseguire una nuova scansione.

Il santo protettore dei programmatori ci fornisce un particolare byte (il «segnale di sincronizzazione verticale») nella porta $3DA dell'adattore; il bit 3 è settato quando siamo in vertical retrace e il bit 0 è 1 quando possiamo aggiornare il video. Perché non approfittarne?

In Pascal, tutto questo si può tradurre nella seguente riga:

```
REPEAT UNTIL ((port[$3DA] and 1)=1);
```

che aspetta il momento buono per l'aggiornamento del video.

Ma siamo alle solite; la lentezza ci opprime nuovamente. A questo punto ci vediamo costretti ad usare il linguaggio macchina (desolato signor Wirth!). E qui entrano in scena MOVTOSCR, MOVTOMEM e PRINT. Le prime due routine spostano uno schermo in memoria e viceversa. PRINT invece stampa una stringa alle coordinate specificate. Tutto questo ad una velocità finalmente sufficiente per ogni applicazione, senza neve e occupando pochissima memoria.

Figura C

```
for k:=1 to length(dato) do
  begin
    memw[$bw00:2*(x-1)+(y-1)*160+2*(k-1)]:=ord(dato[k]);
    memw[2*(x-1)+(y-1)*160+2*(k-1)+1]:=attributo
  end;
```

Note tecniche

Per usare PRINT, MOVTOSCR e MOVTOMEM, bisogna disporre di un

Listato 3 - **PRINT. ASM vers. 3.0**

```
0000                    CSEG    SEGMENT BYTE PUBLIC
                                ASSUME CS:CSEG
                                PUBLIC print
0000                    print   PROC NEAR
0000  55                        push bp             ;salva i registri
0001  8B EC                     mov bp,sp
0003  1E                        push ds
0004  33 C0                     xor ax,ax           ;segmento 0000
0006  8E C0                     mov es,ax
0008  26 8A 0E 0449             mov cl,es:[449h]    ;legge modo video
000D  B8 B000                   mov ax,45056        ;Scheda IBM
0010  80 F9 07                  cmp cl,7
0013  74 03                     je ibm_mda          ;se è 7 ok
0015  05 0800                   add ax,2048         ;...se no è CGA
0018  8E C0             ibm_mda : mov es,ax
001A  8B F5                     mov si,bp
001C  83 C6 07                  add si,7            ;inizio stringa
001F  8C D0                     mov ax,ss
0021  8E D8                     mov ds,ax
0023  8B 4E 06                  mov cx,[bp+6]       ;lunghezza
0026  32 ED                     xor ch,ch
0028  8B 56 57                  mov dx,[bp+107]     ;questa è la Y
002B  8B 7E 59                  mov di,[bp+109]     ; questa è la X
002E  4F                        dec di
002F  4A                        dec dx
0030  B8 00A0                   mov ax,160
0033  F7 E2                     mul dx
0035  D1 E7                     shl di,1
0037  03 F8                     add di,ax           ;160*(Y-1)+2*(X-1)
0039  8A 66 04                  mov ah,[bp+4]       ;attributo
003C  BA 03DA                   mov dx,03dah        ;apro la porta
003F  FA                        cli
0040  FC                       cld                 ;DF = 0
0041  AC                main  : lodsb             ;leggo
0042  3C 24                    cmp al,'$'          ;è un dollaro?
0044  74 17                     je c_char          ;se si salta
0046  50                  alfa  : push ax           ;salva attributo
0047  EC                  ret1  : in  al,dx         ;retrace
0048  24 01                     and al,1
004A  75 FB                     jnz ret1
004C  EC                  ret2  : in  al,dx
004D  24 01                     and al,1
004F  74 FB                     jz  ret2
0051  58                        pop ax
0052  AB                       stosw              ;stampa..
0053  E2 EC                     loop main          ;ripete...
0055  FB                exit   : sti              ;abilita
0056  1F                        pop ds
0057  8B E5                     mov sp,bp
0059  5D                        pop bp
005A  C2 0057                   ret 107             ;esce e rilascia
005D  49                c_char : dec cx
005E  AC                       lodsb              ;car. successivo
005F  3C 24                    cmp al,'$'         ;altro '$' ?
0061  74 E3                    je alfa            ;se si stampalo.
0063  3C 44                    cmp al,'D'         ;è una 'D' ?
0065  75 06                    jne eval
0067  8A 66 04                 mov ah,[bp+4]      ;vecchio colore
006A  49                       dec cx
006B  EB D4                    jmp main           ;torna al main...
006D  2C 3F             eval   : sub al,63        ;controlla se il
006F  73 02                    jnc let1           ;carattere è una
0071  04 07                    add al,7           ;lettera o una
0073  04 08             let1   : add al,8         ;cifra
0075  D0 E0                    shl al,1
0077  D0 E0                    shl al,1
0079  D0 E0                    shl al,1
007B  D0 E0                    shl al,1           ;al * 16
007D  49                       dec cx
007E  8A E0                    mov ah,al
0080  AC                      lodsb              ;seconda cifra
0081  2C 3F                    sub al,63
0083  73 02                    jnc let2
0085  04 07                    add al,7
0087  04 08             let2   : add al,8
0089  02 E0                    add ah,al          ;nuovo attributo
008B  49                       dec cx
008C  EB B3                    jmp main           ;torna al main
008E                    print   ENDP
008E                    CSEG    ENDS
                                END                ;that's all folks
```

compilatore MASM e di una mezz'oretta di tempo; dopo aver digitato con un editor di proprio gusto le routine bisogna eseguire i seguenti comandi:

```
MASM print (o movtoscr, o movtomem)
LINK print     ''  ''    ''  ''
EXE2BIN print.exe,print.bin
```

Fatto questo, o acquistato il disco presso la redazione, le intestazioni delle procedure per il Pascal saranno quelle riportate in figura D dove:
line: definisce la lunghezza di «stringa».
colonna, riga: sono le coordinate in cui si stamperà «stringa»
stringa: stringa da stampare,
stile: attributo per «stringa»,
seg, ofs: segmento e offset della zona di memoria in cui salvare lo schermo, o da cui prelevare i dati da stampare.

PRINT

PRINT è di facile utilizzo. È possibile fissare un attributo «generale» e tanti attributi «particolari» all'interno della stringa. Per esempio l'istruzione print (5,8, «$70inverso$01 sottolineato $Dcolore di default», def) stampa alla posizione 5,8 la stringa
inverso sottolineato colore di default
usando i seguenti attributi:
inverso ($87) per «inverso»
underline ($01) per «sottolineato»
def (a scelta) per «colore di default».

La routine non presenta particolari problemi di comprensione. Dopo aver letto dal byte 449h con che scheda abbiamo a che fare (se il byte è 7, abbiamo una MDA), disponiamo di tutte le informazioni necessarie. Facendo attenzione a salvare i registri BP, SP e DS come da manuale del gentil programmatore e ricordandoci che il primo dato passato dalla procedura Pascal nello stack è alla locazione BP+4, carichiamo i registri nel modo seguente:
ES: segmento della memoria video (b800h per CGA; b800h per MDA)
DI: posizione in cui stampare la stringa (si ottiene con la formula 160* (y - 1) + (x − 1) *2, dopo aver caricato la x in di e la y in dx (x

Figura D

```
type line:string[100];
procedure print(colonna,riga : byte;
                stringa      : line;
                stile        : byte);external 'print.bin'
procedure movtoscr(seg,ofs : integer);external 'movtoscr.bin'
procedure movtomem(seg,ofs : integer);external 'movtomem.bin'
```

MS-DOS

Listato 4 - **PRINT. ASM vers. 4.0**

```
0000                    CSEG    SEGMENT BYTE PUBLIC
                                ASSUME CS:CSEG
                                PUBLIC print
0000                    print   PROC NEAR
0000  55                        push bp             ;salva i registri
0001  8B EC                     mov bp,sp
0003  1E                        push ds
0004  33 C0                     xor ax,ax           ;segmento 0000
0006  8E C0                     mov es,ax
0008  26 8A 0E 0449             mov cl,es:[449h]    ;legge modo video
000D  B8 B000                   mov ax,45056        ;Scheda IBM
0010  80 F9 07                  cmp cl,7
0013  74 03                     je ibm_mda          ;se è 7 ok...
0015  05 0800                   add ax,2048         ;...se no è CGA
0018  8E C0             ibm_mda: mov es,ax
001A  C5 76 06                  lds si,[bp+6]       ;inizio stringa
001D  AC                        lodsb
001E  8A C8                     mov cl,al           ;lunghezza
0020  32 ED                     xor ch,ch
0022  8B 56 0A                  mov dx,[bp+10]      ;questa è la Y
0025  8B 7E 0C                  mov di,[bp+12]      ; questa è la X
0028  4F                        dec di
0029  4A                        dec dx
002A  B8 00A0                   mov ax,160
002D  F7 E2                     mul dx
002F  D1 E7                     shl di,1
0031  03 F8                     add di,ax           ;160*(Y-1)+2*(X-1)
0033  8A 66 04                  mov ah,[bp+4]       ;attributo
0036  BA 03DA                   mov dx,03dah        ;apro la porta
0039  FA                        cli
003A  FC                        cld                 ;DF = 0
003B  AC                main    : lodsb             ;leggo
003C  3C 24                     cmp al,'$'          ;è un dollaro?
003E  74 17                     je c_char           ;se si salta
0040  50                alfa    : push ax           ;salva attributo
0041  EC                ret1    : in  al,dx         ;retrace
0042  24 01                       and al,1
0044  75 FB                       jnz ret1
0046  EC                ret2    : in  al,dx
0047  24 01                       and al,1
0049  74 FB                       jz  ret2
004B  58                        pop ax
004C  AB                        stosw               ;stampa..
004D  E2 EC                     loop main           ;ripete...
004F  FB                exit    : sti
0050  1F                        pop ds
0051  8B E5                     mov sp,bp
0053  5D                        pop bp
0054  C2 000A                   ret 10              ;esce e rilascia
0057  49                c_char  : dec cx
0058  AC                        lodsb               ;car. successivo
0059  3C 24                     cmp al,'$'          ;altro '$' ?
005B  74 E                      je alfa             ;se si stampalo.
005D  3C 44                     cmp al,'D'          ;è una 'D' ?
005F  75 06                     jne eval
0061  8A 66 04                  mov ah,[bp+4]       ;vecchio colore
0064  49                        dec cx
0065  EB D4                     jmp main            ;torna al main...
0067  2C 3F             eval    : sub al,63         ;controlla se il
0069  73 02                     jnc let1            ;carattere è una
006B  04 07                     add al,7            ;lettera o una
006D  04 08              let1    : add al,8         ;cifra
006F  D0 E0                     shl al,1
0071  D0 E0                     shl al,1
0073  D0 E0                     shl al,1
0075  D0 E0                     shl al,1            ;al * 16
0077  49                        dec cx
0078  8A E0                     mov ah,al
007A  AC                        lodsb               ;seconda cifra
007B  2C 3F                     sub al,63
007D  73 02                     jnc let2
007F  04 07                     add al,7
0081  04 08              let2    : add al,8
0083  02 E0                     add ah,al           ;nuovo attributo
0085  49                        dec cx
0086  EB B3                     jmp main            ;torna al main
0088                    print   ENDP
0088                    CSEG    ENDS
                                END                 ;that's all folks
```

in BP+ 109, y in BP+107)

DS: segmento dello stack (per accedere ai dati).

SI: posizione del primo carattere della stringa nello stack

CX: lunghezza stringa (in BP+6)

AH: attributo (in BP+4; come al solito l'ultimo dato passato allo stack, è il primo ad essere accessibile!)

AL: carattere corrente.

Ricordo che per usare le funzioni di stringa LODSB e STOSW, bisogna caricare in DS:SI il segmento è l'offset della posizione in memoria della stringa, in ES:DI il segmento e l'offset dove trasferire la stringa stessa. LODSB carica il byte nel registro AL.

Detto questo, la routine è già fatta. Si esegue CX Volte il loop main stampando il carattere dopo aver controllato che non sia un «$». Nel qual caso, la routine c_char legge il carattere successivo e se è un altro «$» lo stampa normalmente; se è una cifra acquisisce il nuovo attributo, se è una «D» setta l'attributo iniziale. Da notare che il codice dell'attributo deve essere in esadecimale e le lettere A, B, C, D, E, F devono essere maiuscole. In ogni caso, il demo presente nel disco contiene numerosi esempi di utilizzo di PRINT.

Ma veniamo alla routine che controlla il retrace (da ret1 a ret2). Si legge il contenuto della porta 03DAh e si controlla se il bit 0 è settato. Il primo loop si «posiziona» all'inizio di un nuovo retrace; il secondo ne aspetta la fine per poi permettere l'accesso in memoria video.

Note:

Per chi usa il Turbo Pascal 4.0 dobbiamo dire che le procedure esterne vanno «linkate» (con l'opzione $L) al codice Pascal, un po' come succede in C. Il segmento del codice deve per forza chiamarsi CODE o CSEG e si deve specificare il nome del modulo con l'opzione PUBLIC; inoltre, la procedura Pascal e la routine external devono avere stesso nome. Quindi nessun problema per MOVTOMEM e MOVTOSCR che sono state scritte (volutamente, of course) in maniera del tutto generale.

PRINT invece non funziona. Questo perché il turbo 4.0 trasferisce sullo stack non l'intera stringa, come abbiamo visto fare al turbo 3.0, ma l'offset e il segmento in cui trovare la stringa. Nessun problema; con poche modifiche PRINT è pronta per la sfida (vedi listato 4). L'intestazione Pascal adesso diventa:

```
procedure print (x, y: integer, stringa: line,
stile: integer); external;
procedure movtoscr (seg, ofs: word); external;
procedure movtomem (seg, ofs: word); external;
  {$L print. obj }
  {$L movtomem. obj }
  {$L movtoscr.obj }
```

Il Turbo 4.0 usa le cosiddette UNIT. Queste sono praticamente delle «librerie» di procedure e funzioni che, una volta definite, vengono, se richiesto, inserite nel programma Pascal.

Volendone creare una con PRINT, MOVTOMEM e MOVTOSCR, si deve far attenzione a definire le procedure FAR e non NEAR. Inoltre, va ricordato che il primo carattere sullo stack è alla posizione BP+6 e non più BP+4. MC

ZORRO BIG BLUE

IL POTENTE SISTEMA MULTIMODULARE PER AMIGA 500 E 1000 CHE VI OFFRE:

- 3 SLOT A 100 PIN ZORRO 2 AMIGA 2000 COMPATIBILE
- 3 SLOT IBM XT COMPATIBILE
- 3 SLOT IBM AT COMPATIBILE
- 2 POSTI PER 2 DRIVE DA 3,5''
- 1 POSTO PER 1 DRIVE DA 5 1/4''
- 1 POSTO PER HARD DISK
- ALIMENTATORE SWITCHING

HARDITAL

VIA TORTONA, 12
20144 MILANO

...SE HAI L'AMIGA NON LASCIARLA DA SOLA...

GLI HARD DISK

AMEGADRIVE SCSI
Hard Disk e controller per A500/1000 in standard SCSI con orologio e batteria tampone.Formattato con Fast File System.
AMEGADRIVE 20Mb 3,5" SCSI 40ms L. 990000
" 40Mb " " 28ms L.1390000
" 80Mb " " 12ms L.1990000

AMEGADRIVE ST506
Hard Disk e controller per A500/1000 in standard ST506.Autoboot ,formattato con FFS.Espansione opzionale RAM da 2 a 8 MB.
AMEGADRIVE 20 Mb ST506 L. 890000 (con RAM CHIEDERE)
AMEGADRIVE 40 Mb ST506 L. 1090000 (con RAM CHIEDERE)

FLASHBANK A2000 HARDITAL
HD DMA controller SCSI per A2000.Autoboot.FFS.Basato sul processore DMA Motorola MC68440 .L.340000

IMPACT A2000 GVP
HD controller SCSI piu' Espansione RAM da 1 o 2Mb per A2000 con autoboot. 0 Kb 1Mb L.490000-Per altre configurazioni CHIEDERE

A2090 Commodore
Hd controller piu' Harddisk da 20 Mb per A2000.L.1090000

HD2000card 32 Mb
Controller ed HardDisk da 32 Mb su scheda per AMSTRAD, IBM/XT o A2000 con Janus.L.790000

JANUS XT
Emulatore IBM/XT per A2000 + drive da 5.1/4.L.349000
JANUS XT +HardDisk da 32Mb HD2000card L.1490000

LE ESPANSIONI DI MEMORIA

AMEGABOARD
Espansione di memoria per A500/1000 da 2a 8Mb.Esterna, Autoconfigurante. Si installa sul connettore laterale. Munita di connettore passante per altre periferiche, completa di LED e di interruttore per il disinserimento senza disconnetterla dal computer. Dimensioni 21 x 10 x 4,7 cm. 0 Kb L. 290000 - 2 Mb CHIEDERE

RAMINT 1-4
Espansione di memoria da 1 a 4Mb per A1000.Interna.Si inserisce all'interno del computer senza effettuare nessuna saldatura o dissaldatura di componenti.Con orologio tampone. 1 Mb CHIEDERE 0 Kb 190000

AMINTERAM
Espansione di memoria per A500 da 512 Kb.Con orologio tampone.0 Kb L. 59000 - 512 Kb L. 290.000

SUPER0TTO HARDITAL
Espansione da 0-2-4-6-8Mb sulla stessa scheda per A2000. 0 Kb L.290000

A2058 Commodore
Espansione da 2 a 8 Mb per A2000. 2 Mb L.1040000

KICKROM 1.3 A1000
Kickstart 1.3 su eprom senza saldature per A1000 con orologio tampone.L.149000

KICKROM 1.3 A500/A2000
Kickstart 1.3 su Eprom interno per A500/2000.L.99000

I DRIVE

ADRIVE
Drive da 3,5" esterno per A500/1000/2000 con passante.L.219000

ADRIVE TOWER
Come sopra ma triplo nello stesso contenitore.L.590000

ADRIVE2000
Drive interno da 3,5" per A2000.L.189000

ACCELERATOR-PROCESSORI-COPROCESSORI

ATTENZIONE !!! I PREZZI SOTTOINDICATI COMPRENDONO LE SCHEDE ACCELERATRICI SENZA PROCESSORI E COPROCESSORI CHE SONO INDICATI A PARTE.LA SCHEDA BANG PUO' MONTARE LE LE COPPIE 68010-68881/68882 O 68020/68030-68881/68882 MENTRE LA HURRICANE 68020/68030- 68881/68882.QUESTO PER LASCIARE IL MASSIMO GRADO DI LIBERTA' ALL'UTENTE

HURRICANE
Scheda acceleratrice per A1000/A2000.Hurricane A1000 L.599000-Hurricane A2000 L.999000.

BANG
Scheda acceleratrice per A1000. L.340000

HURRICANE MEMORY 1-4
Espansione di memoria a 32 bit per Hurricane.Hurricane Mem 1Mb L.990000

BANG MEM
Espansione di memoria con Ram statiche a 32 bit per BANG da 128 a 512 Kb.BANGMEM 128Kb L.240000

ADAPTER 030
Adattatore per 68030 per Hurricane e Bang.L.390000

PROCESSORI:68010-L.48000 68020-L.340000 68030- L.890000
COPROCESSORI:68881 12MHzL.290000 16MHzL.390000 25MHzL.840000 - 68882 16MHzL.570000 25MHzL.1100000

A2652
Scheda acceleratrice contenente 68020 a 14,3MHz+68881 a 16MHz+MMU68851+RAM a 32 bit da 1 a 4Mb.Completa di epiom per sistema operativo UNIX.CHIEDERE

I DIGITALIZZATORI AUDIO VIDEO

DIGIBOARD
Digitalizzatore audio stereo piu'interfaccia MIDI per A500/1000/2000.L.99000

LIVE! ASQUARED
Digitalizzatore a colori video in tempo reale con effetti video per A500/1000 o A2000(su scheda).Live500 L.549000-LIVE1000 L.440000-LIVE2000 L.630000

PROGEN PROGRESSIVE PER.SYS.
Genlock in standard RS 170A per A500/1000/2000.L.629000

FLICKER FIXER MICROWAY
Scheda da inserire nello slot video dell'A2000 ed elimina il flicker.L.890000

ZORRO BIG BLUE

UNITA' CENTRALE
Chassis metallico per A500/1000 per contenere tutte le periferiche dello Zorro Big Blue.L.140000

ZORRO BIG BLUE BUS
Scheda contenente 3 slot 100 pin A2000 piu' 3 slot XT e 3 slot AT compatibili piu'1 slot a 86 pin per schede con il 68020/68881.L.170000

ALIMENTATORE SWITCHING
Alimentatore da 40W e' necessario solo in caso di montaggio di hard disk o piu' schede.L.89000

MODULO DRIVE
1 o 2 drive da 3,5"-880Kb.L.199000

MODULO MIDI + DIGI STEREO
scheda contenente digitalizzatore stereo piu'interfaccia MIDI.L.79000
ATTENZIONE!!! NELLO ZORRO BIG BLUE SI POSSONO MONTARE TUTTE LE SCHEDE PER L'AMIGA 2000 (JANUS XT/AT,FLASHBANK,IMPACT,A2652,A2090,ecc.)

I MONITORS

COMMODORE 1084 S
Monitor HiRes stereo per A500,A1000,A2000.L.540000

COMMODORE 2080
Monitor HiRes ad alta persistenza per A500,1000,2000.L.820000

PHILIPS 8833 L.480000

STAMPANTI

STAR LC 10 L.420000
STAR LC 10 color L.490000
STAR LC 24-10
Stampante a 24 aghi 150cps NLQ.L.680000

MANNESMANN GP-905 LASER
Stampante laser.L.2590000

I COMPUTER

AMIGA 500 CHIEDERE
AMIGA 2000
CON MONITOR E SECONDO DRIVE DA 3,5" L. 2350000
SONO INOLTRE DISPONIBILI TUTTI I COMPUTER E LE PERIFERICHE AMSTRAD.CHIEDERE.
DISPONIBILI ANCHE TUTTI I COMPATIBILI XT,AT E PS2.CHIEDERE

PER INFORMAZIONI E/O ORDINAZIONI

COMPUTER CENTER

VIA FORZE ARMATE 260
20152 MILANO
TELEFONO 02-4890213

VENDITA SOLO PER CORRISPONDENZA
TUTTI I PREZZI SONO IVA COMPRESA