Anche questa volta ci sono programmi per tutti i gusti: Righe in C per Hercules, Sprotezione in Basic e Anagrammi in Pascal. Del programma Righe non pubblichiamo né il listato né la descrizione dettagliata della routine in quanto eccessivamente lunghi

Righe

di Mauro Silverini - Torre del Lago (LU)

Righe è nato per colmare una mancanza delle recenti versioni del Microsoft C: il supporto della diffusa grafica Hercules. Infatti, le librerie fornite con il compilatore supportano solamente la famiglia di schede IBM (cga, ega, vga, mcga) ed ignorano completamente la nostra Hercules.

Per porre un parziale rimedio al problema, ho scritto l'insieme di routine a cui ho dato il nome di Righe. Righe è scritto in Assembler per la parte in cui la velocità è essenziale ed in C per le parti meno critiche oppure troppo complesse da scrivere in Assembler.

Per lo sviluppo delle routine C, ho utilizzato il QuickC 1.0 con l'ausilio del debugger Codeview (indispensabile!) mentre, per la parte Assembler, ho usato il MASM 4.0 (anch'esso supportato da Codeview).

Per una maggiore chiarezza, è bene iniziare subito con le principali limitazioni (della versione attuale) del pacchetto:
a) supporta solamente lo SMALL memory model (64k dati, 64k programma).

Righe usa circa 10kb dello spazio codice, 2kb di stack e meno di un kb di dati globali;
b) manca di una funzione di FILL (riempimento di figure);
c) non disegna ellissi ed archi;
d) non supporta sistemi di coordinate definiti dall'utente (spesso usati nel disegno di grafici).

Dopo le limitazioni, passiamo a descrivere quelli che io considero i punti a favore:
a) gestione delle due pagine grafiche della Hgc più un numero a piacere di pagine situate nella memoria centrale;
b) funzioni di disegno ed inversione di blocchi e linee;
c) disegno veloce di cerchi vuoti e pieni;
d) funzioni di animazione (get e put);
e) disegno di caratteri in varie dimensioni posizionati punto per punto;
f) hardcopy di sezioni rettangolari della pagina grafica (come nel MAC);
g) linguaggio grafico parzialmente compatibile con quello del GWBASIC (questa caratteristica è stranamente tralasciata da molte librerie grafiche).

Le caratteristiche del pacchetto sono:
a) gestione delle due pagine hardware più un numero a piacere di pagine virtuali posizionate in memoria centrale tramite un insieme di funzioni primitive;
b) primitive grafiche:
— cerchi (corretta l'asimmetria dovuta al rapporto x, y dei punti sullo schermo) vuoti e pieni
— linee
— linee orizzontali (algoritmo ottimizzato)
— punti
— inversione di punti (modo Xor)
— riempimento di blocchi rettangolari
— inversione di blocchi rettangolari
— animazione di blocchi (get & put)
— copia di blocchi nella stessa pagina
— salvataggio della pagina su file
— caricamento della pagina da file
— disegno di poligoni
— disegno di caratteri 8*8 in matrice 90*44 inversi e normali
— disegno di caratteri 8*8 con posizionamento punto per punto e dimensioni a piacere da 8 punti in su
— hardcopy della pagina o di una porzione su stampante che supporta il protocollo Epson ESC/P
— macrolinguaggio grafico compatibile GWBASIC
— copia di una pagina in un'altra
— funzione di test del disegno di un punto;
c) — test di presenza della scheda video
— funzionamento a singola pagina per evitare interferenze in sistemi con più schede grafiche.

Ritengo che la vera mancanza del pacchetto (hobbisticamente parlando, si capisce) sia quella di un'efficiente funzione di FILL per figure non regolari ed una limitazione (ma non troppo) è data dal fatto che il pacchetto è previsto per funzionare con il modello di memoria SMALL (cioè 64k per il programma e 64k per i dati). Non dovrebbe risultare impossibile aggiornare il pacchetto per il funzionamento in modo LARGE (1MB di codice e 1MB di dati), ma non credo che persone che possono scrivere programmi di queste dimensioni (specie

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. 243

per il codice si tratterebbe di decine di migliaia di linee sorgente) pensino di utilizzare le semplici routine da me proposte al posto di ben più professionali (e costose) librerie come Halo e Metawindow. La modifica per altri compilatori C non dovrebbe essere problematica, bisogna prestare un po' di attenzione all'interfacciamento con le routine Assembler e ai vari modelli di memoria.

Utilizzazione

L'uso del pacchetto è estremamente semplice: basta includere il file RIGHE.C nel vostro file sorgente e specificare (al momento del lancio del compilatore) di linkare il file RIGHEASM.OBJ al file oggetto generato dal compilatore. Nel caso che il vostro programma sia composto da più file sorgente, includete RIGHE.C nel primo file da compilare e dichiarate EXTERNAL negli altri file le variabili globali a cui volete accedere (ad esempio XT e YT che indicano a RIGHE la posizione del cursore in cui scrivere il prossimo carattere). Se l'ingombro del pacchetto risulta eccessivo, potete eliminare le routine che non vi interessano, facendo però attenzione ad eventuali legami. Per esempio Poly (int, int, int, int) richiama la funzione line(), se cancellate line() il compilatore al momento di compilare Poly() vi darà errore perché non trova la funzione richiamata (più precisamente l'errore avverrà al momento del linkaggio). Un altro sistema per diminuire l'occupazione del pacchetto è quello di eliminare l'inclusione di eventuali librerie (dopo aver però eliminato le funzioni di Righe che le utilizzano). Riprendiamo la funzione Poly(), essa è l'unica funzione di Righe che usa l'aritmetica Floating Point e le funzioni trascendentali SIN() e COS() perciò, cancellandola, è possibile eliminare l'inclusione di MATH.H e di FLOAT.H, risparmiando così diversi kbyte.

Per quanto riguarda la parte in Assembler, la sua occupazione di memoria è irrisoria (1.6 kb) e quindi non credo che qualcuno abbia la necessità di limarla. L'interfacciamento con il C non pone problemi, l'unica necessità è quella di scrivere in maiuscolo i nomi delle routine in Assembler quando vengono richiamate. (Non fate confusione con i parametri perché le routine non fanno controlli!!).

Bibliografia:
Algoritmo della funzione line
The Sot Warehouse, file graphics.lib del pacchetto muLisp '83. (Modificato e tradotto dal Lisp)
Algoritmo della funzione circle
R.J. Miner, Byte dicembre '87. (Corretto, modificato e tradotto da pseudo codice)
Idea ed esempi
D. Pountain, Byte settembre '87.

Sprotezione di programmi Basic

Salvati con SAVE "FILENAME",P

di Andrea Patelli - Livorno

Vi è mai capitato di trovare un programma Basic protetto col famigerato ,P e di non riuscire a listarlo per individuarne la tecnica di programmazione oppure con lo scopo di modificarlo per soddisfare le vostre esigenze?

Vi è mai capitato di proteggere un vostro lavoro e, al momento di fare pulizia dei file, di cancellare il sorgente?

Io mi sono trovato in tutte e due le situazioni e pertanto mi sono documentato ed ho messo a punto una tecnica di sprotezione che se non si può proprio definire automatica è senza dubbio istruttiva.

D'altra parte le situazioni sopra descritte non capitano tutti i giorni e poi un po' di esercizio nell'uso del DEBUG non credo che sia dannoso... anzi!

Vediamo anzitutto in cosa consiste la protezione fornita dal Parametro ,P. Non si tratta soltanto di una semplice disabilitazione del LIST, ma di una cosa più complessa; infatti il salvataggio viene effettuato in «codice» al pari di un messaggio cifrato in modo da non renderlo comprensibile anche da parte di un esperto.

Mi sono dimenticato di premettere che un file Basic, salvato in modo normale, è perfettamente decifrabile facendone il DUMP sia col DEBUG che con HEXDUMP; infatti si possono riconoscere i numeri di linea (in esadecimale), le istruzioni (ad ogni istruzione corrisponde un TOKEN cioè un codice abbreviato), le stringhe che sono inalterate, i fine linea (zero) ecc.

Però, una volta caricato con LOAD, anche il file protetto assume nella RAM la configurazione normale, pertanto basta riuscire a risalvarlo con le opportune modifiche per raggiungere lo scopo, cosa che ovviamente non si può fare col SAVE.

```
10  REM --------- programma MON.BAS controllo inizio file basic ----------
20  DEF SEG=0:X=PEEK(&H510)+PEEK(&H511)*256:SEGM$=HEX$(X)+":"
30  PRINT "SEGMENTO BASIC    CS = ";SEGM$;"0000";PRINT
40  DEF SEG=X
50  INIPRO=PEEK(&H30)+256*PEEK(&H31)
60  FINPRO=PEEK(&H358)+256*PEEK(&H359)
70  FINBAS=PEEK(&H127)+256*PEEK(&H128)
80  PRINT "INIZIO    PROGRAMMA   a ";SEGM$+HEX$(INIPRO)
90  PRINT "FINE      PROGRAMMA   a ";SEGM$+HEX$(FINPRO)
100 PRINT "FINE del BASIC       a ";SEGM$+HEX$(FINBAS)
110 PRINT "N^ BYTE  PROGRAMMA    ";FINPRO-INIPRO
120 REM --------------- fine programma  ---------------
```

Figura 1

Naturalmente una breve routine in Basic non poteva mancare e questa riportata ha lo scopo di determinare il segmento del Basic e l'offset di inizio del programma Basic (valido per tutti i prog. caricati con LOAD).

Le operazioni da effettuare sono quindi le seguenti:

1) **DEBUG GWBASIC.EXE**
2) **G**

G sta per GO; in tal modo si entra in GWBASIC pur rimanendo in ambiente DEBUG raggiungibile con SYSTEM.

3) **RUN "MON**

Il programma MON (figura 1) fornisce il segmento e l'offset dell'inizio di un

programma Basic nella configurazione di memoria corrente.

4) **LOAD "prog. protetto"**

5) **SYSTEM**

Si rientra in ambiente DEBUG, ma il programma caricato rimane intatto in memoria.

6) **RCS**
:CS

Si richiama Code Segment e dopo il prompt: si inserisce il CS dato da MON.

7) **D CS:INIZIO**

Dump del programma protetto a partire dall'inizio (dato da MON).

Questa operazione non è necessaria, ma è utile per vedere come si presenta in RAM il programma Basic ormai decodificato.

Continuando a battere D si può proseguire nel Dump.

8) **E CS:INIZIO−1 FF**

Trovato l'inizio esatto del programma Basic occorre inserire un FF prima dell'inizio. Infatti i programmi protetti hanno come primo byte FE mentre quelli normali FF.

9) **S CS:INIZIO FF00 00 00 00**

Si cerca la fine del programma che è data dalla sequenza di tre byte consecutivi a zero.

10) **E CS:FINE +1 1A**

Trovato la fine (terzo zero) del programma Basic occorre inserire 1A subito dopo la fine. Il byte 1A corrisponde al EOF.

11) **H FINE+1 INIZIO−1**

Si calcola la lunghezza del programma come differenza tra fine e inizio (secondo valore fornito dalla funz. H del DEBUG).

12) **RCX**
:numbyte+2

Si richiama il registro CX che dà il numero di byte del programma caricato e dopo il prompt: si inserisce il numero di byte da salvare, calcolato in precedenza, +2.

13) **RBX**
:0

Si richiama il registro BX e ci si accerta che sia a zero. In caso contrario si pone a zero perché un programma Basic non può superare 64 k.

14) **N nomeprog.bas**

Si inserisce il nome del programma sprotetto.

15) **W CS:INIZIO−1**

Salvataggio su disco a partire da INIZIO−1.

16) **Q**

Uscita dal DEBUG.

A questo punto potrete caricare il vostro programma col GWBASIC e listarlo e modificarlo a vostro piacimento.

Le prove potete farle direttamente con il programma MON e potete verificare l'inizio e la fine facilmente dato che il token di REM è 8F e ciò che segue REM rimane in codice ASCII.

La prima linea visualizzata con DEBUG apparirà così: ▼

```
77E0:0B60                                              B7 0B 0A              7..
77E0:0B70  00 20 8F 20 2D 2D 2D 2D-2D 2D 2D 2D 2D 20 70 72   . . ---------- pr
77E0:0B80  6F 67 72 61 6D 6D 61 20-4D 4F 4E 2E 42 41 53 20   ogramma MON.BAS
77E0:0B90  63 6F 6E 74 72 6F 6C 6C-6F 20 69 6E 69 7A 69 6F   controllo inizio
77E0:0BA0  20 66 69 6C 65 20 62 61-73 69 63 20 2D 2D 2D 2D    file basic ----
77E0:0BB0  2D 2D 2D 2D 2D 2D 00 EA-0B 14 00 20 97 20 53 45   ------.j... . SE
```

dove B7 0B o meglio 0BB7 è l'indirizzo dell'inizio della seconda linea (dopo 00), 0A 00 o meglio 000A è il numero della prima linea (10), 20 è lo spazio, 8F è il REM, poi un altro spazio e quindi il commento.

Non fate caso se il vostro CS non corrisponderà a quello qui riportato; è normale perché dipende dalla configurazione iniziale e dai programmi residenti.

Tutto sommato la procedura è un po' macchinosa, ma spero che sia risultata più interessante e istruttiva di una routine che fa tutto da sola.

Anagrammi

Risoluzione del crivello di A. Pugliese

di Salvatore Ruggieri - Matino (LE)

Tutto cominciò nel mese di novembre dell'anno 1986. Mi barcamenavo nelle spossanti lezioni di informatica quando da un dibattito con il mio professore (appunto: Aspromonte Pugliese) ne uscì fuori un algoritmo strizzacervelli. Lo scontro era nato dall'incompatibilità di concezioni informatiche tra me ed il suddetto professore.

Egli è un duro della programmazione strutturata, anti-Basic al massimo.

Io ero un tenero simpatizzante del GOTO universale e dei programmi automodificanti. Fu così che tra una parola e l'altra ci diede da risolvere questo semplice algoritmo:

Data una parola di N lettere trovarne tutti gli anagrammi, ossia tutte le combinazioni delle lettere che la compongono; (ad es. ROMA-AMOR-MARO ecc.).

Maledetto quel giorno, solo ora, 22 mesi dopo, riacquisto il possesso di me stesso.

Mi misi subito al lavoro. Il numero di combinazioni è pari a N fattoriale. Gli insuccessi sono stati infiniti. In Basic riuscii solo a fare un programma «contatore» che attraverso indici combinava le parole (es. ROMA=1234 AMOR=4321 quindi le combinazioni si trovano da 1234 a 4321). Lo provai sul mio C64 e credo che stia ancora elaborando.

Quando la mia mente era già malata approdai nel mondo nemico del Turbo Pascal e ne fui felicemente impressionato.

La soluzione

Il programma (solo una manciata di righe) sfrutta le più avanzate tecniche di

```
MICRO   MICOR   MIROC   MIRCO   MIOCR   MIORC   MCROI   MCRIO
MCOIR   MCORI   MCIRO   MCIOR   MROIC   MROCI   MRICO   MRIOC
MRCOI   MRCIO   MOICR   MOIRC   MOCRI   MOCIR   MORIC   MORCI
ICROM   ICRMO   ICOMR   ICORM   ICMRO   ICMOR   IROMC   IROCM
IRMCO   IRMOC   IRCOM   IRCMO   IOMCR   IOMRC   IOCRM   IOCMR
IORMC   IORCM   IMCRO   IMCOR   IMROC   IMRCO   IMOCR   IMORC
CROMI   CROIM   CRMIO   CRMOI   CRIOM   CRIMO   COMIR   COMRI
COIRM   COIMR   CORMI   CORIM   CMIRO   CMIOR   CMROI   CMRIO
CMOIR   CMORI   CIROM   CIRMO   CIOMR   CIORM   CIMRO   CIMOR
ROMIC   ROMCI   ROICM   ROIMC   ROCMI   ROCIM   RMICO   RMIOC
RMCOI   RMCIO   RMOIC   RMOCI   RICOM   RICMO   RIOMC   RIOCM
RIMCO   RIMOC   RCOMI   RCOIM   RCMIO   RCMOI   RCIOM   RCIMO
OMICR   OMIRC   OMCRI   OMCIR   OMRIC   OMRCI   OICRM   OICMR
OIRMC   OIRCM   OIMCR   OIMRC   OCRMI   OCRIM   OCMIR   OCMRI
OCIRM   OCIMR   ORMIC   ORMCI   ORICM   ORIMC   ORCMI   ORCIM
```

Esempio di anagrammi della parola «MICRO».

```
{*****************************************************************************}
{*****  Risoluzione del Crivello di A.Pugliese by Ruggieri Salvatore     *****}
{*****************************************************************************}

PROGRAM Anagrammi ;

TYPE
Longstr = string [255] ;
VAR
Nome    : Longstr     ;
Stack   : Integer     ;

PROCEDURE Anas (mioset : longstr) ;
VAR
a,r             : Byte   ;
Primocar        : Char   ;

BEGIN
     a:=length(mioset) ;
     if a=1 then  writeln  (Nome) { writeln (lst,Nome)  }
            else  begin
                  Stack:=succ (Stack);
                  for r:=1 to a do
                              begin
                                  Primocar:=mioset[1] ;
                                  Nome:=copy (Nome,1,Stack) + mioset;
                                  mioset:=copy(mioset,2,a-1);
                                  Anas (mioset) ;
                                  mioset:=mioset+Primocar
                              end;
                  Stack:=pred (Stack)
                  end
END ;

BEGIN
 Clrscr  ;
 Writeln ('Introduci il nome da anagrammare :');
 Readln (Nome);
 Stack:=-1  ;
 Anas   (Nome);
END.
```

programmazione: la Ricorsione e la lista lineare Stack.

L'anagramma è permesso solo per parole di lunghezza inferiore a 255 (non è colpa mia). In «Nome» è via via contenuta la parola corrente, mentre «Stack» indica il livello di Ricorsione. Naturalmente Stack viene incrementato all'inizio della procedura e decrementato alla fine (lista LIFO: Last In First Out); Stack in effetti indica il livello della parola da aggiornarsi; in «Mioset» è contenuta la parte ennesima della parola che viene passata alla procedura inferiore e scrollata.

(Roma Arom Mora Amor).

Sviluppato integralmente il sistema consiste di una serie ennesima di cicli «for» annidati. Ciò vuol dire che può essere fatto anche in Basic, ma solo se la lunghezza della parola è fissa.

Così le combinazioni di una parola non sono altro che le sue lettere combinate con le combinazioni delle lettere restanti passate a livello inferiore.

$$N * (N-1)!$$

Pugliese è vinto...

MC