*L'autore dei primi due programmi di questo mese si lamenta continuamente del fatto che abbiamo pubblicato una utility assolutamente inutile (e avreste dovuto leggere con che veemenza); tuttavia buona parte dei suoi lavori derivano dal fatto che noi abbiamo pubblicato la famigerata KB_CTRL! Ebbene è proprio questa la ragione che mi ha indotto a selezionare una routine la cui utilità poteva anche essere limitata: l'ho scelta proprio perché era DIDATTICA! E questo significa che anche se pochi lettori la usano molti, studiandola, scoprono qualcosa di più sul loro computer. Magari tutte le routine che ci arrivano fossero così "inutili".*

# Num-UnLock & ESC

*di Alessandro Oddera - Genova*

Vorrei rifarmi all'articolo apparso nell'ottobre dell'89 dal titolo Svuota Keyboard Buffer, dove viene descritto un programma dallo scomodo nome di KB_CTRL.EXE, che azzera il buffer di tastiera alla pressione del tasto CTRL.

Sebbene il programma mi sembri assolutamente inutile, l'articolo è estremamente limpido e degno di interesse perché fornisce un semplice esempio di uso dei T.S.R., argomento fra i meglio non documentati: infatti la Borland non ne fa cenno nei manuali dei suoi compilatori, se non per affermare che si tratta di cosa difficile, mentre la Microsoft nel suo QuickC acclude solo un esempio, peraltro estremamente poco didattico.

Inoltre l'articolo spiega il contenuto di alcune delle locazioni di memoria che riguardano la tastiera, cosa che nella scarsezza di manuali in mio possesso ho trovato utilissima.

Fatto sta che grazie all'inutile KB_CTRL io ho potuto realizzare un paio di programmi per me utilissimi.

Il primo, scritto il Turbo Pascal (listato 1), è dedicato agli sfortunati possessori di un AT: pare infatti che gli AT debbano essere necessariamente usati da contabili, i quali accesa la macchina non fanno altro che dedicarsi alla digitazione di lunghe serie di cifre sul tastierino numerico.

Si è dunque deciso che, all'accensione, il tasto NumLock venga automaticamente attivato.

**ESC**

```
/*
Program ESC
Trasforma la pressione del tasto ESC in ^C (richiesto da AutoCad)
file batch consigliato per il lancio di AutoCad:
ACAD.BAT
       ECHO OFF
       ESC                T.S.R.
       acad %1 %2

Norme per la compilazione: meglio disattivare tutto,
                           compiler debug off,
                           linker debug off,
                           incremental link off,
                           stack & pointer check off,
                           model=small

architetto Alessandro Oddera - GENOVA
*/

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

/* Stack e pointer checking disabilitati */
#pragma check_stack( off )
#pragma check_pointer( off )
/* Per l'allineamento byte anziche' word */
#pragma pack (1)


void far *function;
void far *OldInt16;
void far *answer;
void far *enquiry;
char BOOL= 1;
/*char REENTER= 0;*/

struct KEY /*Buffer di tastiera */
 {
 unsigned char ctrl      ; /*0x417*/
 unsigned char nulla[2];  /*0x418*/ /*Due byte che non interessano */
 unsigned int dove ;      /*0x41a*/
 unsigned int vuoto;      /*0x41c*/
 unsigned char buf[30];   /*0x41e*/
 ) far *skey= (void far *) 0x00000417L;



void interrupt far EscControl()
{
/*if (skey->ctrl & 0x04) skey->vuoto= skey->dove;
Riga che svuoterebbe il buffer in caso di pressione di ctrl*/

if (skey->dove != skey->vuoto)
{
if ( skey->buf [skey->dove-30] == 240 ) BOOL= (char) !BOOL;/*ALT+ESC disable*/

if ( ( skey->buf [skey->dove-30] == 27 ) &&    /*ESC*/
     ( skey->buf [skey->dove-29] != 0 )  &&    /*Non composto con ALT+2+7*/
     (BOOL) )
```

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

```
program NumLock;
{Inverte lo stato del tasto NumLock}
var i: byte absolute $0:$0417;

begin
i:= (i xor 32) ;
write ('Inverte NumLock :');
writeln (' arch. A. Oddera - GE');
end.
```

*Listato 1*

E così per migliaia di volte io mi sono trovato all'interno di un qualche editor a premere <DEL><DEL><DEL> per ritrovarmi con <><><> ma ecco che ora il problema non esiste più!

Con un semplice programma, basta richiamarlo da AUTOEXEC.BAT, il NumLock viene automaticamente invertito.

Il secondo programma è dedicato a quei poverini che, come me, usano quel polpettone composto da un quarto di stile Basic, un quarto di stile Window, un quarto di intelligenza artificiale, un quarto di qualcos'altro: sto parlando di quel programma di disegno che è AutoCAD.

Nelle prime versioni di AutoCAD per interrompere l'esecuzione di un comando era necessario premere la scomoda combinazione CTRL+C, credo per una qualche forma di filosofico rispetto verso il DOS (che a sua volta possiede qualcosa del Basic Like).

Nelle ultime versioni alla AutoDesk qualcuno si deve essere accorto che CTRL+C non è una combinazione così comoda da premere ogni momento, ed ecco che ora da alcune delle nuove funzioni si può uscire mediante la pressione del tasto ESC (stile Windows); a quanto pare però i programmatori AutoDesk non hanno avuto voglia di modificare tutte le funzioni esistenti, e quindi in certi casi va bene l'ESC, in altri è indispensabile il CTRL+C; il che è un altro punto a favore per la mia teoria del polpettone.

Ma ecco che, sempre grazie all'aiuto di KB_CTRL, ho potuto costruire una sorta di speciale setaccio, che filtra i granellini dell'ESC dall'impasto e li trasforma in ^C.

Di svuotare il buffer di tastiera in caso di pressione di tasto errato non se ne parla neanche: la AutoDesk per puro dispetto verso gli utenti ha dotato AutoCAD di un proprio buffer interno, una volta premuto un tasto non c'è più nulla da fare, o sarà che l'AutoCAD Language non possiede la funzione ungetch()?

Il programma (vedi listato 2) è in QuickC.

Concluderei con un paio di domande: non potreste indicarmi l'eventuale esistenza di un bit che possa disattivare il monitor? Così potrei fare un T.S.R. simile a quello che esiste sui Mac per la protezione del monitor, che disattivi il monitor se per un certo tempo non viene premuto un tasto o mosso il mouse, per riattivarlo all'accadere di uno di questi due eventi.

E già che ci siamo: non si potrebbe, sempre come sul Mac, mettere il calcolatore in stato di stand-by, in condizioni di basso consumo, disattivando la ventola, magari con l'aggiunta di un relé?

*Rispondo brevemente qui (vdd): il bit esiste ma non ricordo a memoria la locazione, ma comunque il programma SALVAVIDEO è già stato pubblicato su MCmicrocomputer numero 73 e può essere ordinato su disco con il codice DMS 08. Per quanto riguarda uno stato di stand-by la cosa non è così semplice come potrebbe sembrare, e spegnere la ventola significa mandare in protezione l'alimentatore dopo pochi minuti.*

```
 {
 skey->buf [skey->dove-30]=  3;  /* Codice carattere ^C*/
 skey->buf [skey->dove-29]=  46; /* Codice di posizione ^C */
 /*REENTER=1;*/
 }

}
/*
else if (REENTER)
 {
 REENTER= 0;
 skey->buf [skey->vuoto-30] = 27;
 skey->buf [skey->vuoto-29] = 1;
 if (skey->vuoto < 60) skey->vuoto= skey->vuoto+2; else skey->vuoto= 30;
 }
*/
_chain_intr (OldInt16);
}

void assumecontrol () {
  OldInt16=_dos_getvect (0x16);
  function=EscControl;
  _dos_setvect (0x16, function);
  }

/* Puntatori Huge. */
char huge *tsrstack;
char huge *appstack;
char huge *tsrbottom;

void main () {
    unsigned tsrsize;
    /* Inizializzazione dello stack e coda del programma. */
    _asm mov  WORD PTR tsrstack[0], sp
    _asm mov  WORD PTR tsrstack[2], ss
    FP_SEG( tsrbottom ) = _psp; /* _psp in stdlib.h */
    FP_OFF( tsrbottom ) = 0;

    /* Uso di 16 paragrafi di heap (global in malloc.h). */
    _amblksiz = 256;

    /* Le dimensioni del programma sono:
     *     testa dello stack
     *   - coda del programma (convertito in paragrafi)
     *   + paragrafi nello heap
     *   + un paragrafo giusto per sicurezza
     */
  tsrsize= ((tsrstack - tsrbottom) >> 4) + (_amblksiz >> 4) + 1;
  answer= EscControl;
  enquiry= _dos_getvect (0x16);
  if ( FP_OFF (answer) == FP_OFF (enquiry) )
    printf ("\nProgramma ESC-> ^C gia' installato!\n\0");
    else {
     assumecontrol();
     function=EscControl;
     if ( function == _dos_getvect (0x16) )
        printf ("Programma ESC-> ^C. Progetto: arch. Alessandro Oddera - GENOVA.\n");
     _dos_keep (1,tsrsize);
    }
  }
```

*Listato 2*

## Math Parser

*di Tullio Menga - Milano*

Quasi ogni programmatore, almeno una volta, è tentato di scrivere un programma di matematica che, ad esempio, disegni grafici 3D di funzioni inserite da input in prospettiva reale con cancellazione delle linee nascoste e output in 2000x1500 punti su una stampante a 24 aghi... Ci vuole un po' di tempo, ma in fondo non è difficile. C'è

un solo scoglio di fronte al quale la maggior parte dei computeromani si ferma: il suo oscuro e terribile nome è <PARSER>. Un parser è un "oggetto" che valuta un'espressione algebrica contenente numeri, operatori, parentesi, e anche variabili e funzioni complesse; deve essere perciò in grado di valutare espressioni tipo:

2*x-y^2+1.5*cosh(1.57+atan(sqrt(abs(y-2)*3))/2)^(2/3)

per determinati valori di x e y tenendo conto della priorità delle operazioni. Detto così può sembrare semplice, ma dopo 5 minuti sulla carta si tentati di rinunciare e non pensarci più, aspettando che qualcun altro risolva il problema. Dopo vari mesi di attesa improduttiva ho ripreso in mano il problema e, insperatamente, sono riuscito a risolverlo (e a fare il famoso programma di matematica). Ho scelto il C per sviluppare il parser perché è sintetico, veloce, portabile su qualsiasi macchina e interfacciabile con altri linguaggi.

Per non analizzare la stringa dell'espressione ogni volta che deve essere valutata (il che porterebbe ad una perdita di tempo quando ciò deve avvenire molte volte, per esempio nel calcolo di un grafico), la stringa viene preventivamente "parserizzata" (funzione "parse") una volta per tutte. Il processo si compone di due fasi: una di priorizzazione, nella quale vengono aggiunte nella stringa le parentesi necessarie a rispettare la priorità delle operazioni, e una di compilazione in cui viene generato il vero codice per il calcolo. A questo punto, assegnando alle variabili (da "a" a "z") dei valori specifici, si può valutare velocemente l'espressione (funzione "compute") quante volte si vuole.

Del software fa parte un file header "parser.h" che contiene la definizione di due vettori: uno di stringhe contenenti i nomi delle funzioni che si vogliono implementare (per far riconoscere al programma la fine della lista bisogna inserire un NULL), e uno di puntatori alle funzioni implementate (di tipo double che accettano un parametro double), che in pratica consiste nella lista dei loro nomi veri (quelli della libreria matematica del C o definite precedentemente dall'utente). Di seguito vi è la definizione di una struttura e di un nuovo tipo, "COMP", che serve per la definizione del vettore che conterrà il codice dell'espressione compilata: definito per esempio un "COMP v_formula[nnn]", dove v_formula[i].o è l' istruzione (es. moltiplica, eleva a potenza), e v_formula[i].v è l'operando (un numero double, una variabile, una funzione o

**Math Parser.H**

```
#include <math.h>

char far *Fns[]= { "abs","int","sqrt","exp","ln","log","sin","cos","tan",
                   "sinh","cosh","tanh","asin","acos","atan",NULL };

double (*Fnp[])(double x)= { fabs,floor,sqrt,exp,log,log10,sin,cos,
                             tan,sinh,cosh,tanh,asin,acos,atan };


struct CMP
 {
 char o;
 double v;
 };

typedef struct CMP COMP;

extern struct
 {
 char prior[512];
 double stack[128];
 int cp;
 int pos;
 int fsn;
 int err;
 } PrIn;

int matherr (struct exception *e)
 {
 PrIn.err=1;
 e->retval=1;

 /* eventuali altre operazioni */

 return (1);
 }


int far parse (char *ex,COMP c[]);
int far compute (COMP c[],double vvalue[],double *value);
```

niente). Vi è poi un'altra struttura contenente varie informazioni usate nelle operazioni sull'espressione, della quale è utile ricordare "PrIn.prior", che contiene la stringa della formula già priorizzata. Di seguito vi è una ridefinizione della funzione di libreria C "matherr" (vedere il manuale C per ulteriori informazioni) che cattura gli errori di calcolo e setta il flag "PrIn.err" usato dalla "compute" senza fermare il programma. La "matherr", per ragioni che vedremo poi, vede una divisione per 0 come una chiamata alla funzione "pow(0,–1)", ovvero 0 alla –1, che di fatto è proprio uno 0/0. Infine vi è il prototipo delle due funzioni che verranno usate dall'utente:

– PARSE, che priorizza e compila, accetta come parametri la stringa contenente l' espressione (*ex) e un array di COMP in cui verranno messe le istruzioni per il calcolo. Ritorna il numero di istruzioni necessario per valutare l'espressione (la posizione nel vettore di COMP dell'ultima istruzione), o NULL se c'è un qualche errore di sintassi.

Nella formula non sono ammessi due operatori consecutivi, quindi per fare, ad esempio, x alla -3, bisognerà scrivere "x^)(–3)". Non sono ammessi numeri in forma esponenziale (es. 2.5E–3) che, se proprio necessario, si possono emulare come, ad esempio, "2.5*10^)(–3)". È interessante notare come, grazie alla particolare struttura ricorsiva del programma, eventuali parentesi inutili come per esempio 2*(((3+1))) non generino codice di troppo.

– COMPUTE, che valuta l'espressione compilata, accetta come parametri l'array di COMP contenente le istruzioni per il calcolo, l'array di double contenente i valori delle 26 variabili disponibili (da "a" a "z" ) e il puntatore alla variabile double in cui mettere il risultato. Ritorna il numero di istruzioni di calcolo eseguite (cioè la posizione nel vettore di COMP dell' ultima istruzione) se tutto è andato bene, altrimenti, nel caso di un errore matematico (overflow, underflow, divisione per 0, parametro fuori range di una funzione, etc.), ritorna NULL. La funzione quindi non riconosce gli errori matematici tranne la divisione per 0, ma controlla solo che, durante la sua esecuzione, non ne occorra alcuno. Nel caso di divisione per 0, invoca una "pow(0,–1)", che genera un errore riconosciuto dalla "matherr".

Come abbiamo visto, nel file "parser.h" la funzione "matherr" già ridefinita per essere usata con il parser, ma l'utente può ridefinirla di nuovo aggiungendo prima del "return (1)" eventuali altre istruzioni, per esempio che facciano il display, magari in una apposita finestra, del tipo di errore occorso.

Il programma di esempio illustra in modo semplice l'uso del parser: chiede un'espressione, permette di assegnare dei valori alle variabili x e y (le altre valgono 0) e quindi mostra la formula priorizzata ed il risultato.

Ora abbiamo a disposizione un potente strumento in grado di facilitare la scrittura di programmi matematici rendendoli più eleganti: non resta che mettersi al lavoro e sfruttarlo.

**Math Parser.C**

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <string.h>
#include <math.h>
#include <ctype.h>

extern char *Fns[];

extern double (*Fnp[])(double x);

extern struct CMP
 {
 char o;
 double v;
 };

typedef struct CMP COMP;

struct
 {
 char prior[512];
 double stack[128];
 int cp;
 int pos;
 int fsn;
 int err;
 } PrIn;

int compile (char *s,COMP c[]);



int argument(char *s,COMP c[],int a1,int a2,int a3,int a4)
 {
 int q;
 char fnct[10];
 PrIn.pos++;
 if (s[PrIn.pos]=='(')
  {
  c[PrIn.cp++].o=6;
  PrIn.pos++;
  PrIn.pos=compile(s,c);
  if (PrIn.pos==NULL) return(NULL);
  c[PrIn.cp++].o=a1;
  }
 else if (s[PrIn.pos]>='a' && s[PrIn.pos]<='z')
       {
       q=0;
       while (s[PrIn.pos+q]>='a' && s[PrIn.pos+q]<='z')
         fnct[q]=s[PrIn.pos+q++];
       fnct[q++]=NULL;
       if (q==2)
        {
        c[PrIn.cp].o=a2;
        c[PrIn.cp++].v=s[PrIn.pos++]-'a';
        }
       else
        {
        c[PrIn.cp++].o=6;
        PrIn.pos+=q;
        PrIn.pos=compile(s,c);
        if (PrIn.pos==NULL) return(NULL);
        q=0;
        while (strcmp(fnct,Fns[q++]));
        c[PrIn.cp].o=a3;
        c[PrIn.cp++].v=q-1;
        }
       }
      else
       {
       c[PrIn.cp].o=a4;
       c[PrIn.cp++].v=atof(s+PrIn.pos++);
       while (s[PrIn.pos]>='0' && s[PrIn.pos]<='9' || s[PrIn.pos]=='.')
         PrIn.pos++;
       }
 return (PrIn.pos);
 }


int compile (char *s,COMP c[])
 {
 char fnct[10];
 int q;
 while (!NULL)
  {
  switch (s[PrIn.pos])
   {
   case NULL: return(NULL);
   case '(':  PrIn.pos++;
              PrIn.pos=compile(s,c);
              if (PrIn.pos==NULL) return(NULL);
              break;
   case ')':  return(++PrIn.pos);
   case '*':  PrIn.pos=argument(s,c,7,4,2,3);
              if (PrIn.pos==NULL) return(NULL);
              break;
   case '/':  PrIn.pos=argument(s,c,11,9,5,8);
              if (PrIn.pos==NULL) return(NULL);
              break;
   case '+':  PrIn.pos=argument(s,c,15,13,10,12);
              if (PrIn.pos==NULL) return(NULL);
              break;
   case '-':  PrIn.pos=argument(s,c,19,17,14,16);
              if (PrIn.pos==NULL) return(NULL);
              break;
   case '^':  PrIn.pos=argument(s,c,23,21,18,20);
              if (PrIn.pos==NULL) return (NULL);
              break;
   default :  if (s[PrIn.pos]>='a' && s[PrIn.pos]<='z')
               {
               q=0;
               while (s[PrIn.pos+q]>='a' && s[PrIn.pos+q]<='z')
                 fnct[q]=s[PrIn.pos+q++];
               fnct[q++]=NULL;
               if (q==2)
                {
                c[PrIn.cp].o=1;
                c[PrIn.cp++].v=s[PrIn.pos++]-'a';
                }
               else
                {
                PrIn.pos+=q;
                PrIn.pos=compile(s,c);
                if (PrIn.pos==NULL) return(NULL);
                q=0;
                while (strcmp(fnct,Fns[q++]));
                c[PrIn.cp].o=22;
                c[PrIn.cp++].v=q-1;
                }
               break;
               }
              c[PrIn.cp].o=0;
              c[PrIn.cp++].v=atof(s+PrIn.pos++);
              while (s[PrIn.pos]>='0' && s[PrIn.pos]<='9' ||
                     s[PrIn.pos]=='.') PrIn.pos++;
              break;
```

*(continua a pag. 254)*

*(segue da pag. 253)*

```
    }
   }
  }


void mkprior (char *s)
 {
 int br,ps,c;
 char o[3]={'^','*','/'};
 for (c=0;c<3;c++)
  {
  for (PrIn.pos=0;PrIn.pos<strlen(s);PrIn.pos++) if (s[PrIn.pos]==o[c])
   {
   ps=PrIn.pos;
   br=0;
   do
    {
    if (ps<=0) break; else ps--;
    if (s[ps]==')') br++;
    if (s[ps]=='(' && br>0)
     {
     br--;
     if (ps>0 && isalpha(s[ps-1])) ps--;
     }
    if (br==0 && !(s[ps]!='*' && s[ps]!='/' &&
        s[ps]!='+' && s[ps]!='-' && s[ps]!='^' && s[ps]!='('))
     {
     ps++;
     break;
     }
    } while (!NULL);
   for (PrIn.cp=strlen(s);PrIn.cp>=ps;PrIn.cp--) s[PrIn.cp+1]=s[PrIn.cp];
   s[ps]='(';
   PrIn.pos++;
   ps=PrIn.pos;
   br=0;
   do
    {
    if (ps>=(strlen(s)-1)) break; else ps++;
    if (s[ps]=='(') br++;
    if (s[ps]==')' && br>0) br--;
    if (br==0 && !(s[ps]!='*' && s[ps]!='/' &&
        s[ps]!='+' && s[ps]!='-' && s[ps]!='^' && s[ps]!=')'))
     {
     ps--;
     break;
     }
    } while (!NULL);
   for (PrIn.cp=strlen(s);PrIn.cp>ps;PrIn.cp--) s[PrIn.cp+1]=s[PrIn.cp];
   s[ps+1]=')';
   }
  }
 }


int parse (char *ex,COMP c[])
 {
 char *s;
 for (PrIn.fsn=0;Fns[PrIn.fsn]!=NULL;PrIn.fsn++);
 s=malloc(strlen(ex)*2+4);
 strcpy (s,"(");
 strcat (s,ex);
 strcat (s,")");
 strlwr(s);
 mkprior (s);
 s++;
 strcpy (PrIn.prior,s);
 PrIn.prior[strlen(PrIn.prior)-1]=NULL;
 PrIn.cp=0;
 PrIn.pos=0;
 if (compile(s,c)==NULL) PrIn.cp=0;
 c[PrIn.cp].o=24;
 free(s);
 return(PrIn.cp);
 }


int compute (COMP c[],double vvalue[],double *value)
 {
 double dv;
 *value=0;
 PrIn.cp=0;
 PrIn.err=0;
 while ((c[PrIn.cp].o)!=24 && !PrIn.err)
  {
  switch (c[PrIn.cp].o)
   {
   case  0: *value=c[PrIn.cp].v;
            break;
   case  1: *value=vvalue[c[PrIn.cp].v];
            break;
   case  2: *value=PrIn.stack[--PrIn.pos]*(*Fnp[c[PrIn.cp].v])(*value);
            break;
   case  3: *value*=c[PrIn.cp].v;
            break;
   case  4: *value*=vvalue[c[PrIn.cp].v];
            break;
   case  5: dv=(*Fnp[c[PrIn.cp].v])(*value);
            if (dv!=0) *value=PrIn.stack[--PrIn.pos]/dv;
            else *value=pow(0,-1);
            break;
   case  6: PrIn.stack[PrIn.pos++]=*value;
            *value=0;
            break;
   case  7: *value=PrIn.stack[--PrIn.pos]**value;
            break;
   case  8: dv=c[PrIn.cp].v;
            if (dv!=0) *value/=dv;
            else *value=pow(0,-1);
            break;
   case  9: dv=vvalue[c[PrIn.cp].v];
            if (dv!=0) *value/=dv;
            else *value=pow(0,-1);
            break;
   case 10: *value=PrIn.stack[--PrIn.pos]+(*Fnp[c[PrIn.cp].v])(*value);
            break;
   case 11: if (*value!=0) *value=PrIn.stack[--PrIn.pos]/(*value);
            else *value=pow(0,-1);
            break;
   case 12: *value+=c[PrIn.cp].v;
            break;
   case 13: *value+=vvalue[c[PrIn.cp].v];
            break;
   case 14: *value=PrIn.stack[--PrIn.pos]-(*Fnp[c[PrIn.cp].v])(*value);
            break;
   case 15: *value=PrIn.stack[--PrIn.pos]+*value;
            break;
   case 16: *value-=c[PrIn.cp].v;
            break;
   case 17: *value-=vvalue[c[PrIn.cp].v];
            break;
   case 20: *value=pow(PrIn.stack[--PrIn.pos],(*Fnp[c[PrIn.cp].v])(*value));
            break;
   case 21: *value=pow(*value,vvalue[c[PrIn.cp].v]);
            break;
   case 22: *value=(*Fnp[c[PrIn.cp].v])(*value);
            break;
   case 23: *value=pow(PrIn.stack[--PrIn.pos],*value);
            break;
   }
  PrIn.cp++;
  }
 return (PrIn.err?NULL:PrIn.cp);
 }
```

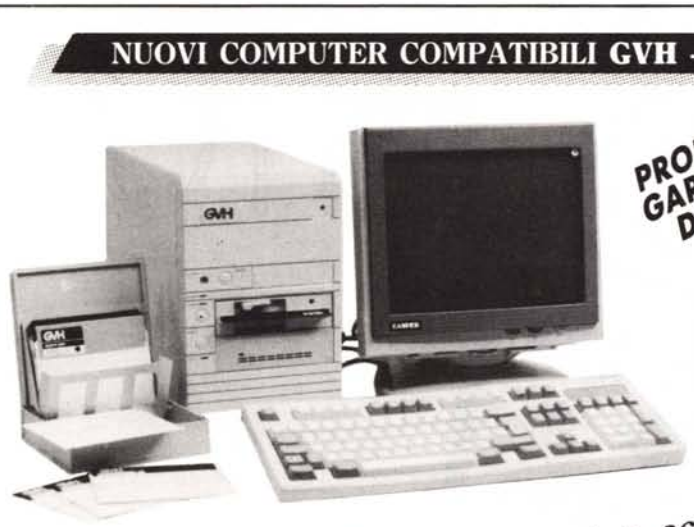
**IMPORTAZIONE DIRETTA**

# GVH linea computer

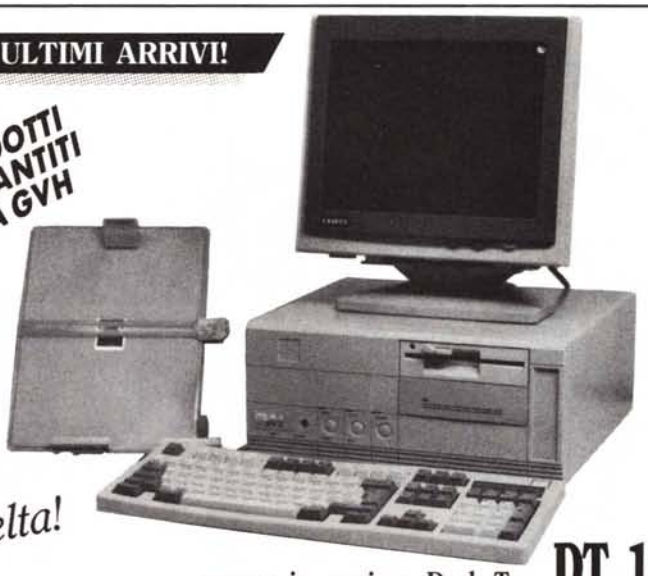
**PREZZI INGROSSO**

## SERVIZIO CASH CARRY EXPRESS

## NUOVI COMPUTER COMPATIBILI GVH - ULTIMI ARRIVI!



PRODOTTI GARANTITI DA GVH



*A vostra scelta!*

**In versione mini Tower: MT 12** — **oppure in versione Desk Top: DT 12**

Main board Mini Size CUP 80286/12 MHz ∅ WS Bios AMI Zoccolo per 80287 6 slot a 16 bit+1 slot a 8 bit Clock a 12 MHz ∅ WS

- RAM installate=1M byte
- Scheda video doppia frequenza CGA+HGA
- Uscita parallela+seriale RS 232
- Scheda controller IDE per HD+FD
- Floppy drive da 1,44M 3,5" Japan
- Hard disk 40 Mb 28 mS
- Involucro metallico tipo mini Tower da tavolo oppure Desk top
- Alimentatore switch 200 W
- Tastiera estesa 101 tasti italiana
- Monitor 14" monocromatico base swivel fosfori bianchi. Bifreq.

Il tutto montato e collaudato da GVH comprese spese di spedizione: **L. 1.950.000**

**OPZIONI: A)** Stesse caratteristiche come sopra descritte ma con scheda video VGA 640×480 16 bit 256 K e Monitor VGA monocromatico Philips . . . . . aggiungere **L. 250.000**

**B)** Scheda video VGA 640×480 16 bit 256 K e Monitor VGA a colori Philips o Hyunday . . . . . aggiungere **L. 650.000**

## PERIFERICHE

### STAMPANTI

| | |
|---|---|
| NEC P2 Plus | L. 630.000 |
| NEC P6 Plus | L. 1.150.000 |
| STAR LC 10 | L. 360.000 |
| STAR LC 24/10 | L. 550.000 |
| STAR LC 15 | L. 680.000 |
| Panasonic KX 1180 | L. 390.000 |
| Panasonic KX 1124 | L. 640.000 |

### Altri modelli pronti a stock

Sempre disponibili:
Hard disk 20-200 Mb
Floppy drive
Hard disk removibili Syquest per MAC

### MONITOR

| | |
|---|---|
| CASPER TM 5157 Multisync colore 0,31 | L. 980.000 |
| CASPER GM 1489D Bifrequenza | L. 190.000 |

### FLOPPY DISK

NO NAME scatole da 10 pezzi con accessori - 100 pezzi minimo -

| | |
|---|---|
| 1,44 Mb HHD 3,5" | L. 2.400 |
| 1,2 Mb HD 5 1/4" | L. 1.000 |

## TOWER 386

Computer 386/32 bit con CPU a 25 MHz e Main Board MYCOMP certificata. RAM installate 2 Mbyte SIMM 70 mS Scheda video doppia freq. CGA+HGA

- Uscite 2 parallele + 2 seriali
- Controller AT bus per FD+HD Western Digital
- Floppy Drive 1,2 Mbyte 5" 1/4
- Floppy Drive 1,44 Mb 3,5"
- HD 40 M byte 28 mS Western Digital
- Involucro Tower da pavimento
- Alimentatore switch 200 W
- Tastiera estesa 101 tasti italiana
- Monitor 14" monocromatico fosfori bianchi base swivel

Il tutto montato collaudato garantito GVH compreso spese di trasporto **L. 3.950.000**
(fotografie del sistema a richiesta)

**SPEDIZIONI IN CONTRASSEGNO (contanti alla consegna)** **TUTTI I PREZZI+IVA 19%**

Per l'Italia: **Gianni Vecchietti GVH**
Bologna - Via Selva Pescarola, 12/8 - Tel. 051/6346181

Per Forlì: Player - Via F.lli Valpiani 6/A - Tel. 31323

Per Bologna: La Bottega Elettronica - Via S. Pio V° 5 - Tel. 550761
Per Mantova: RED Telematica - Via Pilla, 29/A - Tel. 381159
Per Modena: Elec. Center - Via Malagoli, 36 - Tel. 210512
Per Trieste: DUAL SOFT - Via Valdirivo, 40/E - Tel. 631226

Inviateci questo COUPON, vi inseriremo GRATUITAMENTE nella ns. MAILING LIST. Verrete automaticamente informati su tutte le ns. ultime novità.

cognome nome n°
via città (prov.)
CAP

MC

# Microforum

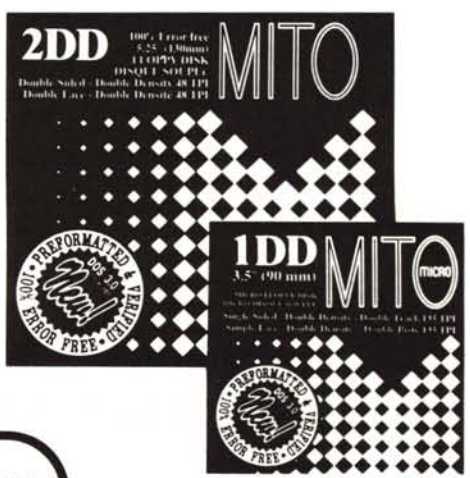
# MITO

PREFORMATTED & VERIFIED • 100% ERROR FREE •

DOS 3.0 OR HIGHER

Nuovo!

Nuovo da Microforum!
I dischi Mito oggi li trovi anche preformattati e verificati:
**al costo di un normale dischetto**
ti assicuri un risparmio di tempo, la certezza della qualità e una velocità impagabile nelle situazioni in cui devi salvare i dati senza l'obbligo di uscire dal programma.
**Mito Microforum**, *ancora una volta più avanti.*




**Media Disk Antonelli**
12, Via Ciociaria - 00162 Roma
Telefono 06/4240379

**Floppy's Market**
5, P.za del Popolo
56029 S. Croce sull'Arno (PI)
Tel. 0571/35124 Fax 32768

**Non Stop spa**
11, Via B. Buozzi
40057 Cadriano di Granarolo (BO)
Tel. 051/765299 Fax 765252

**Spy Cash & Carry**
Piazza Arenella, 6 - 80128 Napoli
Tel. 081/5785623 Fax 5785167

Microforum
MANUFACTURING INC.
TORONTO - CANADA