Programmare in C su Amiga (41)

di Dario de Judicibus

Con questa puntata ci spostiamo dal mondo delle funzioni a quello dei dati. Analizzeremo i vari formati standard dell'Amiga, dalle mappe di tastiera fino all'IFF. Vedremo come si interpretano, come si possono modificare ed addirittura estendere. Il tutto sia dal punto di vista della versione 1.3, sia da quello della nuova 2.04. Cominciamo questo mese con le mappe di tastiera

Introduzione

Una delle caratteristiche più interessanti del sistema Amiga rispetto al più famoso e conosciuto DOS, è quello di incorporare un certo numero di standard relativi al formato dei dati, sia del sistema che delle applicazioni. Il più conosciuto è il formato per lo scambio dei dati fra programmi, chiamato *Interchange File Format (IFF)*; un altro è il formato dei caratteri [*font*]; un altro ancora è quello delle mappe di tastiera.

Il vantaggio di avere tutta una serie di formati standard per i vari dati, utilizzati non solo dal sistema operativo, ma da *tutte* le applicazioni che girano sotto Amiga, è uno dei punti di forza di questo sistema.

Una prima conseguenza di ciò è che tutte le applicazioni che lavorano con dati per i quali è stato definito un formato standard possono scambiarsi tranquillamente informazioni senza aver bisogno di funzioni di importazione ed esportazione dei file [*Import/Export*]. Questo permette all'utente di utilizzare più applicazioni per elaborare un certo file, sfruttando le specifiche capacità di ogni singolo programma al meglio. Ad esempio, è possibile digitalizzare un'immagine con un certo prodotto, alterarne le caratteristiche con un altro, e sovrapporvi un disegno fatto a mano con il buon vecchio *DeLuxe Paint*. Il tutto senza dover passare da un formato ad un altro. Ed ecco *Roger Rabbit* che vi stringe la mano sul terrazzo di casa vostra. A questo si aggiunge un altro vantaggio, e cioè che le singole applicazioni sono più piccole, occupano cioè meno spazio su disco ed in memoria. Infatti, se da un lato non hanno bisogno di introdurre un sacco di codici in più per gestire la conversione da e verso altri formati, dall'altro possono utilizzare molti servizi di sistema già disponibili per gestire i formati standard.

Da qui ne viene un secondo vantaggio, e cioè che sull'Amiga non abbiamo bisogno delle centinaia di prodotti per la conversione da un formato all'altro che sono invece necessari sul DOS e su altri sistemi. Certo, anche per l'Amiga esistono prodotti per la conversione dei formati, ma il loro scopo è appunto quello di importare ed esportare file dai formati Amiga a quelli più utilizzati di altri ambienti operativi. Fintanto che ci si muove solo in ambiente Amiga, d'altra parte, tali prodotti non sono necessari.

A questo proposito, vorrei far notare che le funzioni di conversione da HAM a LORES e viceversa, non sono conversioni di formato, quanto elaborazioni di

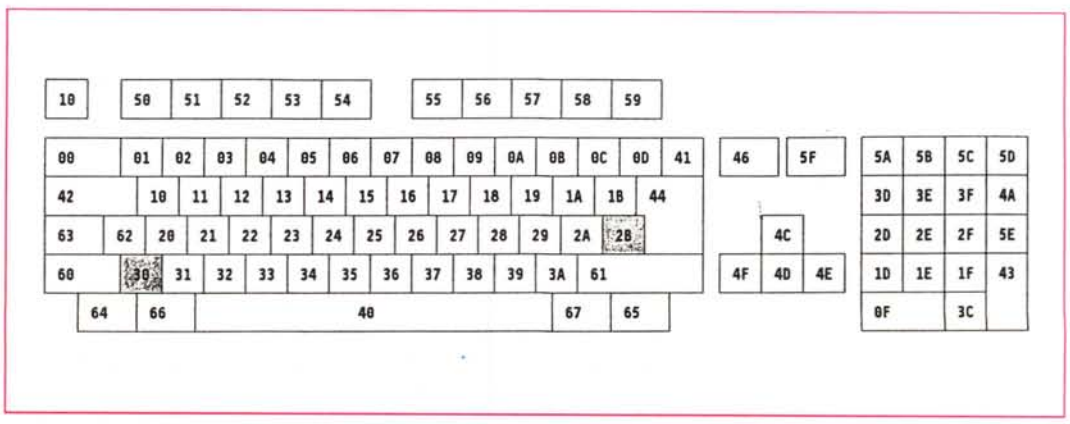


Figura 1 - Codici di scansione esadecimale (A500/A2000).

immagine, dato che si passa da una coppia risoluzione/tavolozza ad un'altra.

Un terzo vantaggio è che alcuni formati possono essere estesi dagli sviluppatori secondo regole definite dal sistema, senza per questo generare effetti collaterali sui programmi già esistenti. È il caso dell'IFF.

Uno dei formati standard Amiga è quello che permette di ridefinire la mappa della tastiera, quello cioè che associa ad ogni tasto, o combinazione di tasti, un determinato effetto, in genere relativo alla visualizzazione di uno o più caratteri.

Le mappe di tastiera

Una tastiera non è altro che un insieme di tasti disposti più o meno come su una macchina da scrivere. A differenza di quest'ultima, tuttavia, non esiste nei computer una associazione fissa tra un tasto ed il carattere che viene visualizzato sullo schermo, o comunque l'effetto che produce sugli altri tasti (maiuscolo, controllo, blocco numerico). Questa associazione è gestita via software dal sistema operativo, utilizzando una tabella che associa ad ogni codice di scansione una sequenza di byte.

Sotto la tastiera infatti, corre un segnale che *tocca* uno dopo l'altro tutti i tasti per poi ricominciare da capo ciclicamente. Questo segnale è detto segnale di scansione della tastiera. Quando un tasto viene premuto, il segnale memorizza una serie di informazioni. La cosa funziona un po' come se il segnale fosse un trenino merci che passa sotto una serie di contenitori che ogni anno si aprono per far cadere delle merci nei vagoni. Non necessariamente tutte le tastiere sono fatte allo stesso modo, ma il risultato finale è più o meno lo stesso.

Il sistema riceve quindi una serie di informazioni del tipo se un tasto è stato premuto o meno, per quanto tempo, se è stato rilasciato, se altri tasti sono stati premuti contemporaneamente, e così via. Ma come viene identificato ciascun tasto? Certo non con una lettera, od un nome, dato che le tastiere possono essere differenti nei vari paesi, pur mantenendo più o meno la stessa struttura. Viene invece utilizzato un numero, detto codice di scansione [*scan code*], che indica generalmente l'ordine con cui il segnale di scansione *tocca* i vari tasti. In figura 1 sono riportati i codici di scansione di una tastiera di un Amiga 500 o 2000.

Il codice di scansione identifica univocamente il tasto e quindi permette l'utilizzo di tastiera con una diversa disposizione dei tasti senza per questo confondere il sistema. Ad esempio, i tasti cursore sono disposti diversamente sulla tastiera dell'Amiga 1000 e su quella dell'A2000. Il loro codice di scansione è tuttavia lo stesso, per cui il sistema li utilizza allo stesso modo. In quale modo, poi, è un problema di associazione tra codice di scansione e sequenza di emissione. Quest'ultima è una sequenza di byte che il sistema associa ad un certo codice di scansione, o ad una combinazione di codici (come nel caso dei qualificatori), utilizzando appunto la mappa di tastiera.

In realtà, come vedremo quando analizzeremo in dettaglio il formato della mappa di tastiera, a ciascun codice di scansione possono essere associate diverse informazioni, alcune relative alle caratteristiche del tasto, altre alle sequenze che si possono ottenere dalla pressione del tasto da solo, od in combinazione con alcuni tasti speciali, chiamati qualificatori [*qualifier*]. Questi ulti-

	b8	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
	b7	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
	b6	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
	b5	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
b4 b3 b2 b1		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0 0 0 0	0	NUL	DLE	SP	0	@	P	`	p	▪▪	DCS	NBSP	°	À	Ð	à	ð
0 0 0 1	1	SOH	DC1	!	1	A	Q	a	q	▪▪	PU1	¡	±	Á	Ñ	á	ñ
0 0 1 0	2	STX	DC2	"	2	B	R	b	r	▪▪	PU2	¢	²	Â	Ò	â	ò
0 0 1 1	3	ETI	DC3	#	3	C	S	c	s	▪▪	STS	£	³	Ã	Ó	ã	ó
0 1 0 0	4	EOT	DC4	$	4	D	T	d	t	IND	CCH	¤	´	Ä	Ô	ä	ô
0 1 0 1	5	ENQ	NAK	%	5	E	U	e	u	NEL	MW	¥	µ	Å	Õ	å	õ
0 1 1 0	6	ACK	SYN	&	6	F	V	f	v	SSA	SPA	¦	¶	Æ	Ö	æ	ö
0 1 1 1	7	BEL	ETB	'	7	G	W	g	w	ESA	EPA	§	·	Ç	×	ç	÷
1 0 0 0	8	BS	CAN	(	8	H	X	h	X	HTS	▪▪	¨	¸	È	Ø	è	ø
1 0 0 1	9	HT	EM	)	9	I	Y	i	y	HTJ	▪▪	©	¹	É	Ù	é	ù
1 0 1 0	10	LF	SUB	*	:	J	Z	j	z	VTS	▪▪	ª	º	Ê	Ú	ê	ú
1 0 1 1	11	VT	ESC	+	;	K	[	k	{	PLD	CSI	«	»	Ë	Û	ë	û
1 1 0 0	12	FF	FS	,	<	L	\	l	|	PLU	ST	¬	¼	Ì	Ü	ì	ü
1 1 0 1	13	CR	GS	-	=	M	]	m	}	RI	OSC	SHY	½	Í	Ý	í	ý
1 1 1 0	14	LS1	RS	.	>	N	^	n	~	SS2	PM	®	¾	Î	Þ	î	þ
1 1 1 1	15	LSO	US	/	?	O	_	o	DEL	SS3	APC	¯	¿	Ï	ß	ï	ÿ

Figura 2 - ECMA-94 Latin 1 International 8-bit.

```
struct  KeyMap
{
  UBYTE  *km_LoKeyMapTypes ; // Base: struttura delle definizioni
  ULONG  *km_LoKeyMap      ; // Base: sequenze di emissione
  UBYTE  *km_LoCapsable    ; // Base: effetto del blocco del maiuscolo
  UBYTE  *km_LoRepeatable  ; // Base: emissione continua
  UBYTE  *km_HiKeyMapTypes ; // Speciali: struttura delle definizioni
  ULONG  *km_HiKeyMap      ; // Speciali: sequenze di emissione
  UBYTE  *km_HiCapsable    ; // Speciali: effetto del blocco del maiuscolo
  UBYTE  *km_HiRepeatable  ; // Speciali: emissione continua
};
```

Figura 3 - La struttura KeyMap.

mi hanno la caratteristica di modificare la sequenza emessa da un certo tasto, se vengono premuti in contemporanea con questo. Ad esempio, il tasto di *Maiuscolo* [*shift*] ha la caratteristica di far emettere ad un tasto alfabetico la versione maiuscola del carattere associatovi. Lo stesso qualificatore può avere effetti diversi a seconda del tasto con cui è premuto. Infatti, se nel caso precedente fosse stato un tasto numerico nella riga superiore della sezione principale della tastiera ad essere premuto invece di un tasto alfabetico, l'effetto risultante sarebbe stato l'emissione di uno dei caratteri speciali presenti in tale riga, come il simbolo della *lira sterlina* od il *percento*. Per distinguere i tasti qualificatori dagli altri tasti, li chiameremo anche *modificatori*, dato che *modificano* la sequenza di emissione di un altro tasto.

I caratteri

Ma come viene interpretata la sequenza di byte emessa quando un tasto viene premuto? Dipende dalla tabella dei caratteri utilizzata. L'Amiga utilizza come insieme di caratteri standard quello internazionale ECMA-94 Latin-1, riportato in figura 2. Si tratta di una tabella molto simile allo standard ANSI X3.134.1-198x, ma differente dalle tabelle usate dal DOS. Quest'ultimo utilizza infatti una tabella con una sola sezione di caratteri di controllo, un blocco di caratteri grafici, ed un certo numero di variazioni nelle associazioni tra caratteri e valori esadecimali, detti codici di pagina [*codepage*]. Nell'Amiga non esistono codici di pagina, ma una sola tabella con due sezioni di caratteri di controllo, riportati nelle cellette grigie in figura.

Un carattere di controllo, a differenza di un carattere cosiddetto *stampabile* [*printable character*], rappresenta una determinata azione che deve avvenire sullo schermo. Spesso il carattere di controllo è seguito da un certo numero di caratteri normali che rappresentano i parametri che definiscono in dettaglio l'azione da effettuare. Ad esempio, certe sequenze spostano il cursore del testo nelle quattro direzioni possibili, o cambiano il colore del testo, o mandano a capo il cursore stesso.

Caratteri di controllo sono lo spazio indietro [*backspace*], il ritorno del carrello [*carriage return*] ed il prefisso delle sequenze di controllo ANSI [*control sequence introducer (CSI)*].

Ricapitolando, la mappa di tastiera determina l'associazione tra un codice di scansione, e quindi un tasto, e la sequenza di emissione da mandare al sistema. La tabella dei caratteri, invece, associa la sequenza di emissione, che altro non è che una sequenza di numeri compresi fra 0 e 255 (un byte), ed i caratteri stampabili o di controllo che devono essere utilizzati.

Vedremo che il modo in cui avviene la sezione del carattere, od in generale della sequenza di emissione a partire dalla mappa di tastiera, è un po' più complicato di quello che permette di associare un carattere ad un byte nella tabella dei caratteri. Quest'ultima infatti, è una semplice griglia di tipo *battaglia navale*, dove le righe rappresentano i valori corrispondenti ai quattro bit bassi del byte, mentre le colonne rappresentano quelli corrispondenti ai quattro bit alti. Ad esempio, il punto esclamativo è rappresentato da **0x21**, dato che si trova a riga **0001** (o **_1**) e colonna **0010** (o **2_**).

La mappa della tastiera, invece, è una struttura complessa formata da varie tabelle a dimensione fissa ed un numero variabile di blocchi a dimensione variabile, differenti per ogni tastiera.

La tastiera

La tastiera dell'Amiga può essere divisa in due sezioni. La prima, detta sezione base o bassa, dato che corrisponde ai valori più bassi del codice di scansione (da **0x00** a **0x3F**), viene utilizzata per i caratteri stampabili, come quelli dell'alfabeto, i numeri, i segni di interpunzione, i simboli speciali e via dicendo. La seconda, detta speciale od alta, dato che corrisponde ai valori più alti del codice di scansione (da **0x40** a **0x67**), viene utilizzata per i tasti di edizione, i tasti funzionali, gli operatori del tastierino numerico, ed i qualificatori.

La struttura KeyMap

Vediamo ora come viene memorizzata nel sistema una mappa di tastiera.

```
#define  KC_NOQUAL   0        // Nessun qualificatore
#define  KC_VANILLA  7        // shift + alt + control

// Nota che KCF_xxxx = 1 << KCB_xxxx

#define  KCB_SHIFT   0        // shift - numero di bit
#define  KCF_SHIFT   0x01     // shift - codice di qualificazione
#define  KCB_ALT     1        // alt - numero di bit
#define  KCF_ALT     0x02     // alt - codice di qualificazione
#define  KCB_CONTROL 2        // control - numero di bit
#define  KCF_CONTROL 0x04     // control - codice di qualificazione
#define  KCB_DOWNUP  3        // doppia emissione - numero di bit
#define  KCF_DOWNUP  0x08     // doppia emissione - codice di qualificazione

#define  KCB_DEAD    5        // "morto" - numero di bit
#define  KCF_DEAD    0x20     // "morto" - codice di qualificazione

#define  KCB_STRING  6        // stringa - numero di bit
#define  KCF_STRING  0x40     // stringa - codice di qualificazione

#define  KCB_NOP     7        // indefinito o riservato - numero di bit
#define  KCF_NOP     0x80     // indefinito o riservato - codice di qualificazione
```

Figura 4 - Codici di qualificazione.

Affinché una mappa venga associata alla tastiera, è necessario caricarla in memoria. La mappa base, presente nel Kickstart, è quella corrispondente alla tastiera americana semplificata (tastierino numerico ridotto). Se si vuole la tastiera americana completa, od un'altra delle tante tastiere disponibili nell'indirizzario **devs/keymaps**, bisogna caricarla in memoria con il comando **SetMap**. Abbiamo quindi una serie di mappe memorizzate su disco in un certo numero di file, ed una o più mappe caricate in memoria, di cui una sola attiva alla volta per ogni *console*, più una di *default* per le *console* per le quali non è stata specificata alcuna particolare mappa.

Questo vuol dire che in Amiga, al contrario di quanto succede in altri sistemi, ogni applicazione o finestra può avere una sua mappa di tastiera associata. **SetMap** fa sì che una certa mappa sia usata sia come *default*, sia per la finestra dalla quale è stato lanciato.

Ogni mappa è mantenuta in memoria in una struttura chiamata **KeyMap**, la cui definizione è riportata in figura 3.

Questa struttura contiene otto puntatori ad otto aree di memoria di dimensione prefissata. Le prime quattro riguardano la sezione base della tastiera, le altre quattro quella speciale. Ogni area contiene una tabella utilizzata dal sistema per l'interpretazione della tastiera.

La prima tabella è quella puntata da **km_LoKeyMapTypes**.

Essa utilizza *un byte* per ogni codice di scansione. In tutto sono 64 byte. Questa tabella definisce in pratica la struttura della mappa vera e propria dei caratteri nella sezione bassa della tastiera. Infatti, la tabella che contiene la mappa vera e propria dei tasti ha a disposizione solo quattro byte per ogni codice di scansione, come vedremo nella prossima puntata. La struttura di questi quattro byte, e quindi il loro contenuto, è determinata appunto dalla prima tabella, che chiameremo quindi *tabella dei formati*.

Ogni byte nella tabella dei formati determina il *formato* dell'elemento corrispondente nella tabella con le sequenze di emissione, che è quindi la tabella che indica quale o quali caratteri devono essere emessi a fronte di quel codice di scansione. Chiameremo questa tabella, per semplicità, *mappa di emissione*.

La tabella puntata da **km_HiKeyMapTypes** è equivalente a quella puntata dalla **km_LoKeyMapTypes**, ma contiene solo 38 byte e si riferisce alla sezione alta della tastiera.

Abbiamo detto che esiste una serie di tasti chiamati qualificatori che, se premuti contemporaneamente agli altri tasti, ne modificano il comportamento. I tre qualificatori principali sono il modificatore per il maiuscolo [*shift*], che indicheremo nella forma abbreviata con «s», il modificatore per la tastiera alternata [*alt*], che abbrevieremo come «a», ed il modificatore di controllo [*control*], o «c». A questi si aggiungono una serie di *qualificatori logici*, non necessariamente legati ad un tasto reale. La lista completa dei *codici di qualificazione* è riportata in figura 4.

Codici dei qualificatori	1° byte	2° byte	3° byte	4° byte
KC_NOQUAL	-	-	-	k
KCF_SHIFT	-	-	s+k	k
KCF_ALT	-	-	a+k	k
KCF_CONTROL	-	-	c+k	k
KCF_SHIFT + KCF_ALT	s+a+k	a+k	s+k	k
KCF_ALT + KCF_CONTROL	a+c+k	c+k	a+k	k
KCF_SHIFT + KCF_CONTROL	c+s+k	c+k	s+k	k
KC_VANILLA	s+a+k	a+k	s+k	k

s = shift : modificatore per il maiuscolo
a = alt : modificatore per la tastiera alternata
c = control : modificatore per le sequenze di controllo
k = key : tasto

Figura 5 - Formato a quattro caratteri.

Vediamo per ora come la mappa di emissione è influenzata dalla tabella dei formati.

Modello a quattro

Il formato più utilizzato per la mappa di emissione bassa, è quello che contiene un carattere per byte, e che associa questo carattere ad una ben precisa combinazione dei qualificatori principali, come riportato in figura 5. Fate attenzione: il formato non rappresenta i qualificatori effettivamente premuti con il tasto, bensì quelli che è permesso premere, o che comunque hanno un qualche effetto su quel tasto. Sono le colonne nella tabellina in figura, e quindi la posizione del byte nel modello «a quattro» a determinare quale carattere va emesso a fronte della combinazione effettivamente utilizzata di modificatori. Infatti, come si può vedere in figura, il quarto byte rappresenta sempre il carattere emesso quando è premuto il tasto da solo (indicato dalla lettera «k»). Il terzo, quello emesso quando oltre al tasto viene premuto il modificatore *maggiore*. Il secondo, quello emesso quando oltre al tasto viene premuto il modificatore *minore*, se c'è. In caso contrario questo byte sarà nullo (e comunque ignorato). Il quarto, rappresenta una combinazione dei due qualificatori (maggiore e minore), ed il tasto stesso. Da tener presente che l'ordine con cui si valuta se un modificatore è maggiore di un altro è il seguente (a partire dal modificatore più «alto»):

1. shift
2. alt
3. control

Un caso a sé è rappresentato dall'ultima combinazione in tabella. Questa rappresenta il potenziale utilizzo di tutti e tre i qualificatori in contemporanea con uno dei cosiddetti *tasti vaniglia*, così chiamati a causa del color crema che hanno nella maggior parte delle tastiere (per distinguerli da quelli, come i tasti funzionali, colorati in grigio scuro). Ovviamente, quattro byte non bastano a descrivere tutte le combinazioni possibili. In questo caso, il sistema interpreta i quattro byte come se fossimo nel caso *shift + alt*, mentre il carattere emesso a fronte della pressione del solo *control* più il tasto vaniglia, viene ricavato azzerando il quinto e sesto bit del carattere emesso quando non è utilizzato alcun modificatore.

Da notare che questa struttura limita comunque il numero massimo di combinazioni di modificatori che possono essere effettivamente usate con un certo tasto a sei (6). In pratica questo succede solo con **KC_VANILLA**. Negli altri casi abbiamo al massimo quattro (4) combinazioni.

Facciamo un paio di esempi.

Supponiamo che a fronte del codice di scansione **0x28** la tabella dei formati contenga il valore **KC_VANILLA**, e che la mappa di emissione contenga

0xA3 0xA3 0x4C 0x6C

Questo vuol dire che la pressione del

La scheda tecnica: Inside 2.0

Nella scorsa puntata abbiamo visto sette schede relative alle funzioni della nuova **graphics.library**, a partire dalla **ReadPixelArray8()** alla **TextExtent()**.

In questa vedremo le ultime cinque schede, e cioè quelle che vanno dalla **TextFit()** alla **WritePixelLine8()**.

Nella prossima puntata incominceremo a vedere le funzioni della **gadtools.library**.

TextFit

Conta il numero di caratteri che stanno in una area testo descritta da una struttura **TextExtent** (vedi figura 7).

```
prototipo

ULONG TextFit  // il numero di caratteri che stanno nei limiti posti
(
  struct RastPort   *rastport             , // dove siamo
  STRPTR             string               , // puntatore alla stringa
  UWORD              strLen               , // numero dei caratteri
  struct TextExtent *textExtent           , // estensione possibile
  struct TextExtent *constrainingExtent   , // estensione richiesta
  WORD               strDirection         , // direzione del testo
  UWORD              constrainingBitWidth , // Larghezza richiesta
  UWORD              constrainingBitHeight  // Altezza richiesta
) ;

// "strDirection" può essere 1 o -1. In quest'ultimo caso "string"
// punta all'ultimo carattere della stringa, non al primo.
```

Ci sono due modi di specificare il rettangolo in cui il testo dovrebbe entrare. O con una struttura di estensione, la **constraining Extent**, oppure dando direttamente l'altezza e la larghezza del rettangolo in pixel, utilizzando gli ultimi due parametri della funzione.

Questa, oltre a ritornare il numero di caratteri che entrano nel rettangolo, riempie anche un'altra struttura **TextExtent** che contiene le caratteristiche del rettangolo minimo che circoscrive esattamente quella parte della stringa che entra nel rettangolo richiesto. I due rettangoli non sono necessariamente uguali dato che il rettangolo richiesto può contenere anche una frazione di carattere in più.

Novità rispetto alla versione precedente.

Si tratta di una nuova funzione.

VideoControl

Esegue i comandi di controllo del video contenuti nell'area fornita dal programmatore, operando sulla mappa dei colori associata ad una certa *viewport*.

```
prototipo

ULONG VideoControl  // Se l'operazione ha avuto successo o meno
(
  struct ColorMap  *cm    , // mappa dei colori
  struct TagItem   *tags    // lista dei comandi da effettuare
) ;
```

I comandi che possono essere utilizzati nella lista sono riportati in figura 8.

Novità rispetto alla versione precedente.

Si tratta di una nuova funzione.

WeighTAMatch

Vàluta se un dato *font* abbia o meno una certa somiglianza con un altro, e se sì ritorna un valore positivo che indica il livello di corrispondenza.

```
prototipo

WORD WeighTAMatch  // Ritorna il livello di corrispondenza tra i due font
(
  struct TTextAttr *reqTextAttr   , // Attributi del testo richiesti
  struct TextAttr  *targetTextAttr, // Attributi del testo proposti
  struct TagItem   *targetTags      // Attributi estesi proposti, o NULL
) ;

// Corrispondenza massima : MAXFONTMATCHWEIGHT
// Corrispondenza minima  : 1
// Nessuna corrispondenza : 0
```

Il campo **ta_Name** nella struttura **TextAttr**, ed il campo **tta-_Name** nella struttura TTexAttr non sono utilizzati.

I ***tag*** influenzano il risultato solo se **FSF_TAGGED** è stato impostato nel campo **ta_Style** di **reqTextAttr** e contemporaneamente **targetTags** non è nullo.

```
struct TextExtent
{
  UWORD           te_Width  ; // Lunghezza della stringa, in pixel
  UWORD           te_Height ; // Altezza del "font", in pixel
  struct Rectangle te_Extent ; // Rettangolo del testo. Vedi sotto...
};

// La lunghezza in pixel è la stessa che riporterebbe TextLength()
// L'altezza è quella riportata in tf_YSize

/*
** Il rettangolo contenente il testo è misurato relativamente alla
** linea "guida" del testo, o base:
**
**  te_Extent.MinX    Solitamente   0
**  te_Extent.MinY    Sempre come   - tf_Baseline
**  te_Extent.MaxX    Solitamente   te_Width - 1
**  te_Extent.MaxY    Sempre come   te_Height - tf_Baseline - 1
**
*/
```

Figura 7 - Struttura TextExtent.

```
VTAG_ATTACH_CM_GET
VTAG_ATTACH_CM_SET
VTAG_BATCH_CM_CLR
VTAG_BATCH_CM_GET
VTAG_BATCH_CM_SET
VTAG_BATCH_ITEMS_ADD
VTAG_BATCH_ITEMS_GET
VTAG_BATCH_ITEMS_SET
VTAG_BITPLANEKEY_CLR
VTAG_BITPLANEKEY_GET
VTAG_BITPLANEKEY_SET
VTAG_BORDERBLANK_CLR
VTAG_BORDERBLANK_GET
VTAG_BORDERBLANK_SET
VTAG_BORDERNOTRANS_CLR
VTAG_BORDERNOTRANS_GET
VTAG_BORDERNOTRANS_SET
VTAG_CHROMAKEY_CLR
VTAG_CHROMAKEY_GET
VTAG_CHROMAKEY_SET
VTAG_CHROMA_PEN_CLR
VTAG_CHROMA_PEN_GET
VTAG_CHROMA_PEN_SET
VTAG_CHROMA_PLANE_GET
VTAG_CHROMA_PLANE_SET
VTAG_COERCE_DISP_GET
VTAG_COERCE_DISP_SET
VTAG_END_CM
VTAG_NEXTBUF_CM
VTAG_NORMAL_DISP_GET
VTAG_NORMAL_DISP_SET
VTAG_VIEWPORTEXTRA_GET
VTAG_VIEWPORTEXTRA_SET
VTAG_VPMODEID_CLR
VTAG_VPMODEID_GET
VTAG_VPMODEID_SET
```

Figura 8 - Video controlli.

Novità rispetto alla versione precedente.
Si tratta di una nuova funzione.

WritePixelArray8

Imposta il numero di penna di ogni singolo pixel di un'area rettangolare di un certo *RastPort*, di cui sono date le coordinate dell'origine e dell'angolo diagonalmente opposto.

Vi ricordo che il numero di penna permette di ricavare dalla tavolozza dei colori associati a quello specifico *RastPort* il colore del pixel stesso.

prototipo

```
LONG WritePixelArray8  // Ritorna il numero di pixel scritti
(
  struct RastPort *rp      , // Puntatore alla struttura RastPort
  UWORD            xstart  , // Ascissa del punto di partenza
  UWORD            ystart  , // Ordinata del punto di partenza
  UWORD            xstop   , // Ascissa del punto di arrivo
  UWORD            ystop   , // Ordinata del punto di arrivo
  UBYTE           *array   , // Vettore delle "penne" da utilizzare
  struct RastPort *temprp    // Puntatore ad una RastPort temporanea
) ;

// Allocazione di memoria:
// ------------------------------------------------------------------
//   array     almeno        ((((xdelta+15)>>4)<<4)*(ydelta+1) bytes
//             dove          xdelta = xstop - xstart
//                           ydelta = ystop - ystart
// ------------------------------------------------------------------
// Assicuratevi che xstop >= xstart ed ystop >= ystart
```

array contiene tanti byte quanti sono i pixel da colorare, e per quali la penna può prendere un valore da **0** a **255**.

Novità rispetto alla versione precedente.
Si tratta di una nuova funzione.

WritePixelLine8

Imposta il numero di penna di ogni singolo pixel di un segmento orizzontale in un certo *RastPort*, di cui sono date le coordinate dell'origine ed il numero di pixel da impostare verso destra.

prototipo

```
LONG WritePixelLine8   // Ritorna il numero di pixel scritti
(
  struct RastPort *rp      , // Puntatore alla struttura RastPort
  UWORD            xstart  , // Ascissa del punto di partenza
  UWORD            ystart  , // Ordinata del punto di partenza
  UWORD            width   , // Numero di pixel da leggere
  UBYTE           *array   , // Vettore delle "penne" da utilizzare
  struct RastPort *temprp    // Puntatore ad una RastPort temporanea
) ;

// Allocazione di memoria:
// ------------------------------------------------------------------
//   array     almeno        ((((width+15)>>4)<<4) bytes
// ------------------------------------------------------------------
// Assicuratevi che width > 0
```

array contiene tanti byte quanti sono i pixel da colorare, e per i quali la penna può prendere un valore da **0** a **255**.

Novità rispetto alla versione precedente.
Si tratta di una nuova funzione.

tasto da solo emette una «l», in congiunzione con *shift* una «L», mentre sia con *alt* che con *alt* + *shift* verrà emesso un «£». Nel caso invece che venga premuto il modificatore *control*, verrà emesso il carattere di controllo per il salto pagina [*Form Feed*] indicato anche come *FF* (**0x0C**). Questo indipendentemente dall'eventuale pressione di altri modificatori.

E ancora. Consideriamo il codice di scansione **0x2D**. Esso corrisponde ad un tasto del tastierino numerico. Nella mappa americana, il formato è **KC_NOQUAL**, che vuol dire che i modificatori non hanno alcun effetto su di esso. Ed infatti, i quattro byte corrispondenti, nella mappa di emissione, sono

0x00 0x00 0x00 0x34

Se noi volessimo modificare tale tasto aggiungendo una parentesi graffa aperta nel caso venga premuto anche lo *shift*, come succede nella tastiera italiana, non basterebbe modificare i quattro byte così

0x00 0x00 0x7B 0x34

ma bisognerebbe anche cambiare **KC_NOQUAL** in **KCF_SHIFT**.

Emissione di stringhe

Ci sono situazioni, tuttavia, in cui un byte non è sufficiente comunque a rappresentare la sequenza di emissione a fronte della pressione di uno o più tasti contemporaneamente. Ad esempio, la sequenza di emissione di un tasto funzionale data dal carattere di controllo *CSI* più uno o due caratteri rappresentanti un numero compreso fra zero e nove, più la tilde («~»). Cioè una stringa di tre o quattro byte. E questo solo per la pressione del singolo tasto. Una sequenza simile va emessa anche quando viene premuto il modificatore per il maiuscolo insieme al tasto funzione.

L'indicazione che va emessa una stringa piuttosto che un singolo carattere *per almeno una* delle varie possibili combinazioni di modificatori, viene fornita usando il qualificatore logico **KCF_STRING**. Questo qualificatore può essere utilizzato con una qualunque combinazione di modificatori già vista in precedenza in figura 5.

Nel caso che venga usato **KCF_STRING**, la mappa di emissione conterrà l'indirizzo di un blocco extra che contiene le varie stringhe secondo un formato che verrà spiegato nella prossima puntata. Da notare che in questo caso, essendo le sequenze di emissioni memorizzate in un altro blocco, non è più necessario limitare il numero di combinazioni effettive a sei, ma è possibile definire fino ad otto stringhe per ciascun tasto. Vedremo in seguito comunque, che alcune limitazioni esistono anche qui.

È evidente inoltre che la presenza di blocchi extra per le stringhe fa sì che non esiste una dimensione fissa per una mappa di tastiera, ma questa sarà tanto più grande quante più stringhe sono state definite al suo interno.

Da notare che le stringhe in questione *non* terminano con il carattere *null* come succede di solito in C, ma sono semplicemente vettori di caratteri.

I tasti morti

Il terzo ed ultimo formato è legato al qualificatore logico **KCF_DEAD**. Un tasto identificato da questo codice è detto *tasto morto*. Attenzione però! Questo non vuol dire che il tasto è disabilitato, cioè che non funziona, come succede al tasto con la «ù» nella tastiera italiana se si carica la mappa di tastiera americana. In quel caso, il tasto è veramente morto, in quanto nelle tastiere americane esso fa parte del grosso ta-

sto di invio a forma di *L rovesciata* tipico delle tastiere *Made in USA*.

I tasti morti sono tasti che non emettono da soli, ma solo se premuti in sequenza con altri tasti morti. Il risultato è in genere un carattere formato dalla sovrapposizione grafica di due caratteri. Questa tecnica è stata sviluppata per permettere alle tastiere americane di emettere caratteri accentati o compositi (come «æ» o «ø») premendo alcuni tasti in congiunzione con *alt*, specificando così l'accento da usare (grave, acuto, circonflesso, ecc.), e successivamente il carattere base (in genere una vocale).

Al contrario dei modificatori, che vanno premuti contemporaneamente al tasto base, i tasti morti vanno premuti uno dopo l'altro, in un ordine ben preciso. Da qui il nome. Infatti, quando il primo tasto morto viene premuto, non succede assolutamente niente, in apparenza. Non compare cioè nessun carattere, né il cursore si sposta a destra. Solo alla pressione del secondo tasto il carattere viene generato, ed il cursore si muove nella nuova posizione.

Ad esempio, per ottenere una «à», va *prima* premuta la combinazione *alt + g*, che identifica l'accento grave, e *poi* il tasto con la vocale «a», altrimenti non funziona. Esistono quindi tasti morti modificatori e tasti morti modificabili. Le vocali sono in genere tasti morti modificabili, dato che formano la base per molti caratteri speciali, usati soprattutto in Europa. Altri tasti morti modificabili sono la «y», utilizzata ad esempio per il carattere «ÿ», e la «n», che serve a formare la «ñ» spagnola.

Anche in questo caso, la mappa di emissione contiene l'indirizzo di un blocco extra, la cui struttura sarà descritta nella prossima puntata.

Chi di voi ha comprato uno dei primi Amiga 1000 con tastiera italiana, si ricorderà certamente i tre grossi errori nella definizione della tastiera italiana:
1. il fatto che fosse QWERTZ invece di QWERTY,
2. i tasti con la «ù» e i simboli di maggiore e minore disabilitati,
3. l'accento circonflesso che non funzionava.

Il primo problema non era immediato da fissare, proprio perché non basta trovare nella mappa di emissione la «y» e la «z» e scambiarle. La prima, infatti, corrisponde ad un tasto morto, e quindi la mappa di emissione per questo tasto contiene un puntatore ad un blocco extra. La seconda, invece, corrisponde ad un tasto normale, e quindi ad un modello a quattro. Lo scambio doveva avvenire tra questo ed il puntatore al blocco extra. I vari caratteri «y» non andavano toccati. Tra l'altro non si capisce perché ancora nei *RKM 1.3* si insiste a presentare la tastiera italiana come QWERTZ.

Byte	Bits	Codici di scansione
#1	76543210	07 <--- 00
#2	76543210	0F <--- 08
#3	76543210	17 <--- 10
#4	76543210	1F <--- 18
#5	76543210	27 <--- 20
#6	76543210	2F <--- 28
#7	76543210	37 <--- 30
#8	76543210	3F <--- 38

Figura 6 - Ordine di scansione delle tabelle Capsable e Repeatable.

Il secondo problema era collegato alla diversa conformazione fisica delle tastiere americane ed europee. Anche qui non bastava aggiungere i caratteri giusti, ma bisognava modificare il valore **KCF_NOP** associato ai due tasti.

Il terzo problema era legato appunto al fatto che l'accento circonflesso, nelle tastiere americane, è spesso un tasto morto. In queste tastiere, infatti, per ottenere l'accento da solo bisogna prima premere il tasto e poi la barra spaziatrice. Se questi è premuto prima di una vocale invece, dà luogo a vocali accentate. Per risolvere quest'ultimo problema, era necessario ridefinire come **KC_VANILLA** il tasto, e riconvertire il blocco extra in un modello a quattro.

Le tabelle minori

Le tabelle puntate da **km_LoCapsable** e **km_LoRepeatable** indicano rispettivamente se un tasto è soggetto o meno all'influenza del *blocco del maiuscolo* [*Caps Lock*], e se esso è in grado di emettere continuamente la stessa sequenza nel caso venga mantenuto premuto.

Entrambe le tabelle contengono un bit per ogni codice di scansione, e sono formate da 8 byte (per totali 64 codici di scansione). Inoltre, in entrambi i casi l'ordine con cui vanno analizzati i bit a fronte di codici di scansione crescenti è dal bit **0** al bit **7**, e dal primo all'ottavo byte (vedi figura 6).

La prima tabella determina l'effetto del *Caps Lock* sul tasto. Se il bit corrispondente è impostato ad uno ed il tasto del blocco del maiuscolo è premuto (led rosso acceso), l'effetto risultante sarà quello della pressione contemporanea del tasto con lo *shift*. Un vantaggio di questa tabella è che è possibile far sì che i tasti alfabetici siano influenzati dal blocco del maiuscolo, ma non i segni di interpunzione, per cui non è necessario disattivare il blocco quando si vogliono utilizzare tali segni mentre si sta scrivendo il testo tutto in maiuscolo.

La seconda tabella determina il comportamento del tasto quando è mantenuto in pressione. In alcuni casi può essere comodo fare emettere in continuazione lo stesso carattere. Ad esempio, tenendo premuto il meno si può ottenere facilmente una linea di separazione nel testo. Per far la stessa cosa senza l'emissione continua, bisognerebbe premere molte volte in continuazione lo stesso tasto, il che è abbastanza scocciante. In altri casi è importante evitare assolutamente che nel premere un tasto venga emessa più di una sequenza di caratteri, come nel caso dei tasti funzione.

Nel primo caso basta impostare il bit corrispondente al codice di scansione del tasto ad uno, nel secondo caso va impostato a zero.

Le tabelle puntate da **km_HiCapsable** e **km_HiRepeatable** sono equivalenti a quelle puntate da **km_LoCapsable** e **km_LoRepeatable**, ma si riferiscono alla sezione alta della tastiera. Benché siano relative tuttavia a solo 38 codici di scansione, anch'esse sono per comodità formate da otto byte, piuttosto che da cinque.

La doppia emissione

Esiste infine un qualificatore il cui utilizzo *non è documentato*, ma il cui nome appare comunque una volta nei *RKM*.

Si tratta di **KCF_DOWNUP**. Se si utilizza da solo, viene emesso il carattere nel 4° byte del modello a quattro quando il tasto è premuto, eventualmente ripetuto più volte se l'emissione continua è attiva per quel tasto, e quello nel 3° byte quando è rilasciato. Si tratta quindi di una doppia emissione.

Dato che non ho altri elementi né conosco alcuna tastiera in cui viene utilizzato, farò alcuni esperimenti e poi vi farò sapere.

Conclusione

Abbiamo visto in questa puntata sei delle otto tabelle che formano una mappa di tastiera. Nella prossima puntata vedremo in dettaglio le mappe di emissione, con le strutture dei vari blocchi extra, ed alcuni esempi pratici su come si possono definire questi blocchi. MC

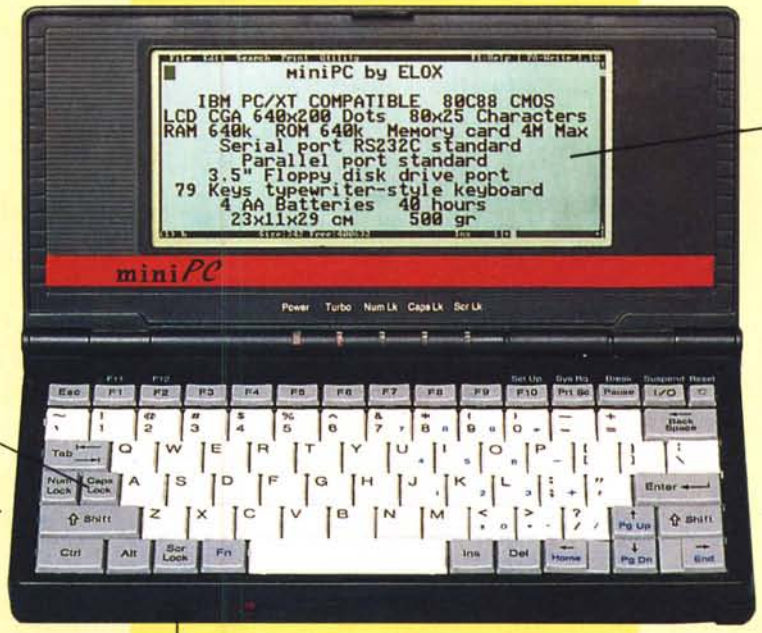
Dario de Judicibus è raggiungibile tramite MC-link alla casella 2120.

miniPC® ELOX®

IL PERSONAL COMPUTER IN 580 GRAMMI

PREZZO PUBBLICO
Lit. 980.000
IVA ESCLUSA

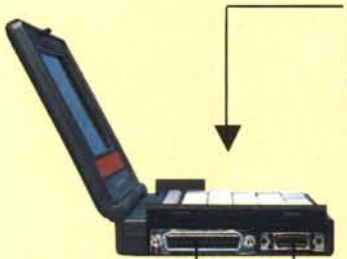
DIMENSIONI 230x110x29 mm / IBM PC/XT COMPATIBILE
RAM 640 K – ROM 640 K / MEMORIA DI MASSA = MEMORY CARD FINO A 2 MB (8 MB NEL 1992)
È POSSIBILE UTILIZZARE SOFTWARE STANDARD MS-DOS



Schermo LCD bianco e nero, 80 colonne per 25 righe, 640x200 punti, CGA

Tastiera 79 tasti tipo dattilo

Regolatore intensità schermo



Porta per Disk Drive

Porta parallela standard per stampante

Memory Card
L'HARD DISK
DEL FUTURO

Slot per Memory Card

Porta seriale RS232C standard

CARATTERISTICHE DI SISTEMA
CPU: 80C88 (CMOS) IBM PC/XT COMPATIBILE
LOW-POWER: CHIP AVANZATO SUPER INTEGRATO

MEMORIA
ROM: 640 K
RAM: 640 K
MEMORY CARD SLOT PER UTENTE (CARD, RAM, ROM, OTP)

SOFTWARE • ROM-DOS: DR-DOS V.5.0 • SOFTWARE APPLICATIVO INCLUSO:
– FILE LINK (per trasferire dati e programmi da/per altri Computer) – WORD PROCESSOR – DOS FILE - VIEWMAX – SCHEDULE REMINDER (con allarme) – CALENDARIO – DIARIO DIGITALE - NOTE PAD – DIARIO CARD / INDEX CARD – NAME CARD / INDEX CARD – CALCOLATRICE 12 DIGIT

ALIMENTAZIONE
BATTERIA "AA" x 4
DURATA DELLA BATTERIA CIRCA 20 ORE

DOTAZIONE
4 PILE ALCALINE AA
BORSA
ALIMENTATORE RETE 220 VOLT 50 Hz

▼ IL COMPLETO

▼ L'UFFICIO NELLA 24 ORE

▼ IN VIAGGIO

EVART S.r.l. - Via Rossetti, 17 - 20145 Milano - Tel. 02/4814619 - Fax 02/48006714

RICHIEDERE IL RIVENDITORE PIÙ VICINO

Il software MS-DOS, Amiga e Macintosh di Pubblico Dominio e Shareware distribuito da

in collaborazione con Microforum

Questo software non può essere venduto a scopo di lucro ma solo distribuito dietro pagamento delle spese vive di supporto, confezionamento, spedizione e gestione del servizio. I programmi classificati Shareware comportano da parte dell'utente l'obbligo morale di corrispondere all'autore un contributo indicato al lancio del programma.

MSDOS

COMUNICAZIONE

CODICE	TITOLO&DESCRIZIONE	REC. HARDWARE
COM/01	ONE TO ONE	mc104
COM/02	PROCOMM Noto programma di comunicazione	Hard disk
COM/03	OMEGA LINK	mc106
COM/04	BACKCOMM	mc103
COM/05	ZIP	mc110
COM/06	FOSSIL DRIVER & .TPU	mc110
COM/07	MAXIHOST	mc110

DATABASE

CODICE	TITOLO&DESCRIZIONE	REC. HARDWARE
DBS/02	VIDEO DATABASE	mc105 Hard disk
DBS/03	HOME MANAGER DataBase, calcolatrice e calendario	Hard disk
DBS/04	MAIL-MONSTER	mc103
DBS/06	PC-FILE+	mc106
DBS/07	TASK MASTER Projet Plaining	
DBS/09	DMS	mc107
DBS/10	ARCHIVIO PARROCCHIALE	mc109
DBS/11	ABSTRACT	mc115

EDUCATIVO

CODICE	TITOLO&DESCRIZIONE	REC. HARDWARE
EDU/01	ABC FUN KEYS	mc103
EDU/02	COMPUTER TUTOR Auto-apprendimento del computer	
EDU/04	GEOBASE ARCH. GEOGRAFICO	mc109

GIOCO

CODICE	TITOLO&DESCRIZIONE	REC. HARDWARE
GIO/04	ALDO'S ADVENTURE	mc103 EGA/VGA
GIO/05	CAESAR Strategia	BASIC+EGA/VGA
GIO/08	EGAINT	mc104 EGA/VGA
GIO/09	PC-JIGSAW Puzzle	
GIO/11	SUPER PINBALL Super Flipper	
GIO/12	ARK Clone di Arkanoid	EGA/VGA
GIO/13	BANYON WARS Strategia	EGA/VGA
GIO/14	CAPTAIN COSMIC Gioco grafico	EGA/VGA
GIO/16	EGA GOLF Gioco del Golf	EGA/VGA
GIO/17	EGA TREK Star Trek	EGA/VGA
GIO/18	JOUST VGA Gioco da bar	VGA
GIO/19	MINER VGA	mc104 VGA
GIO/21	MOSAIX Puzzle	VGA
GIO/22	OTHELLO EGA	mc103 EGA/VGA
GIO/23	POKER SOLITAIRE Poker da soli	EGA/VGA
GIO/24	QUATRIS Tetris con Bombe ecc.	EGA/VGA
GIO/25	SHARKS Giocate ai sommozzatori	EGA/VGA
GIO/26	SLOT EGA Slot Machine	EGA/VGA
GIO/27	BASSTOUR Pesca d'altura	EGA/VGA
GIO/29	GALACTIC BATTLE Clone di Invaders con sonoro	EGA/VGA
GIO/30	HOUSE OF HORRORS Casa degli orrori	EGA/VGA
GIO/31	NOID Consegnate la pizza all'ultimo piano	EGA/VGA
GIO/32	PINBALL EGA Super Flipper	EGA/VGA
GIO/34	MAHJONG EGA Gioco di società orientale	EGA/VGA
GIO/35	MR.SPOCK	mc105 EGA/VGA
GIO/36	MONUMENTS OF MARS	mc106
GIO/37	PHARAOH'S TOMB	mc106
GIO/38	POKER	mc107 EGA/VGA
GIO/39	NIM	mc108 CGA
GIO/40	TESORI	mc108 CGA
GIO/41	TOMBOLA	mc108 CGA
GIO/42	SMILE	mc109 VGA
GIO/43	CHINESE SOLITARIE	mc111 VGA
GIO/44	TRETRIX	mc111 VGA
GIO/45	SICHUAN	mc112 VGA
GIO/46	EGAWALLS	mc113 EGA
GIO/47	GRID HER	mc113 VGA
GIO/48	BANDIERE!	mc114
GIO/49	PETWORLD	mc114
GIO/50	FORZA4	mc114
GIO/51	CROBOTS	mc115
GIO/52	YAHTZEE!	mc115 VGA
GIO/53	PAROLOSO	mc115

GRAFICA

CODICE	TITOLO&DESCRIZIONE	REC. HARDWARE
GRF/01	FINGER PAINT Programma di disegno	
GRF/02	PC-KEY-DRAW	mc107 CGA
GRF/03	H&P CALENDAR	mc103
GRF/04	PC-DEMO SYSTEM	mc105
GRF/05	GRAPHICWORKSHOP	mc106
GRF/06	SOLAI & TRAVI	mc112
GRF/07	GOSTPAINT	mc112

SPREADSHEET

CODICE	TITOLO&DESCRIZIONE	REC. HARDWARE
SPD/01	AS-EASY-AS	mc103
SPD/02	EXPRESS-CALC	mc104
SPD/03	EZ-SPREADSHEET Calcoli di budget	
SPD/04	INSTACALC	mc107
SPD/05	QUEBECALC Spreadsheet 3D	

UTILITY

CODICE	TITOLO&DESCRIZIONE	REC. HARDWARE
UTI/01	PC-DESK-TEAM	mc107
UTI/02	HARD DISK UTILITIES Per gestire l'Hard Disk	Hard Disk
UTI/03	DOS HELP	mc104
UTI/04	DISK SPOOL II	mc103
UTI/05	LOCKTITE Protegge i file con password	
UTI/07	LHARC	mc105
UTI/08	ARJ	mc105
UTI/09	LZEXE	mc105
UTI/10	DIET	mc105
UTI/11	PKLITE	mc105
UTI/12	NEWSPACE	mc105
UTI/13	CATDISK	mc105
UTI/14	POINT&SHOOT	mc105
UTI/15	SHEZ	mc106
UTI/16	ZZAP	mc106
UTI/17	GUARDIAN ANGEL	mc107
UTI/18	STORE	mc107
UTI/19	TXT	mc107
UTI/20	xSET	mc108
UTI/21	ZAPDIR	mc108
UTI/22	UTILITY COLLECTION	mc109
UTI/23	DiR	mc109
UTI/24	CLEANUP	mc111
UTI/25	SAB DISKETTE UTILITY	mc111
UTI/26	TIF2GRAY	mc111
UTI/27	FILLDISK	mc111
UTI/28	ORASCO	mc111
UTI/29	XDIR	mc111
UTI/30	WINCOMMANDER	mc112
UTI/31	MOUSE FORMATTER	mc112
UTI/32	WINZIP	mc112
UTI/33	MOUSE EDITOR	mc113
UTI/34	DEPURA	mc113
UTI/35	DISK FATTER	mc113
UTI/36	POWER DOS	mc114
UTI/37	SIM_LIB	mc114
UTI/38	UTILITY PC	mc114
UTI/39	DBOOK 1.0	mc115

VARIE

CODICE	TITOLO&DESCRIZIONE	REC. HARDWARE
VAR/01	COMPOSER Per suonare al computer e stampare lo spartito	
VAR/02	CHECK-MATE Controllo delle finanze personali	
VAR/03	PIANO-MAN	mc104
VAR/04	BARTENDER Tutti i cocktail	mc103
VAR/05	DIET DISK La dieta al computer	
VAR/06	ELEMENTARY C Per programmatori in C	
VAR/07	RECIPES	mc104
VAR/08	PERSONAL C COMPILER	mc105
VAR/09	MOUSE.TPU & NEWEXEC	mc106
VAR/10	TSR, PRINT & GESTECC	mc106
VAR/11	ARIANNA	mc106
VAR/12	TOTOPROJET	mc108 CGA
VAR/13	COVER	mc108
VAR/14	CODICE FISCALE	mc109 Hard disk
VAR/15	FLIGHT	mc109
VAR/16	DIZIONARIO INFORMATICO	mc109
VAR/17	ITALIA90	mc110
VAR/18	TATA-BIGNOMIX UTILITY	mc110
VAR/19	QUICK BASIC ROUTINES	mc110
VAR/20	MICROGESTS	mc113
VAR/21	CALCOLO INDICE ELO	mc113

CODICE	TITOLO&DESCRIZIONE	REC. HARDWARE
VAR/22	MENU	mc113
VAR/23	PROMETEO	mc114
VAR/24	IRIS	mc115
VAR/25	MODELLI DI TERRENO	mc115

WORDPROCESSOR

WPR/02	FREEWORD	mc103
WPR/03	PC-WRITE	mc106
WPR/05	GALAXY	mc104
WPR/06	EDITOR	mc110
WPR/07	NOTEBOOK	mc112
WPR/08	WORDY	mc113
WPR/09	VIX	mc114

AMIGA

COMUNICAZIONE

AMCO/01	AMIPAC	mc110
AMCO/02	FC FREE COMMUNICATION	mc113

DATABASE

AMDB/01	BADGER	mc113

GIOCO

AMGI/02	WELLTRIX	mc105
AMGI/03	SYS	mc105
AMGI/04	SCOPONE SCIENTIFICO	mc108
AMGI/05	LA FINE DI UN TIRANNO	mc109
AMGI/06	LA PANTERA SIAMO NOI	mc109
AMGI/07	MEGABALL	mc110
AMGI/08	REVERSI	mc114
AMGI/09	FRIENDLY CARD	mc115

GRAFICA

AMGR/01	PRINTSTUDIO	mc104
AMGR/02	TEXTPAINT	mc105
AMGR/03	SCREENX	mc105
AMGR/04	SETPAL	mc105
AMGR/05	FREEPAINT	mc113
AMGR/06	LABEL MAKER	mc114
AMGR/07	PICTSAVER	mc114

SPREADSHEET

AMSP/01	SPREAD	mc104
AMSP/02	EQUATIONWRITER	mc110

UTILITY

AMUT/01	MACH III	mc104
AMUT/02	RULER	mc104
AMUT/03	HEX	mc104
AMUT/04	MOM	mc104
AMUT/05	CB	mc104
AMUT/06	ZETAVIRUS	mc104
AMUT/07	DIRMASTER	mc105
AMUT/08	KDC	mc105
AMUT/09	XCOPYIII	mc105
AMUT/10	CD2TAPE	mc105
AMUT/11	BBS & LOG	mc106
AMUT/12	UTILITIES	mc106
AMUT/13	VIEW80 II	mc106
AMUT/14	MATCALC	mc106
AMUT/15	ICONMASTER	mc106
AMUT/16	HERMIT	mc106
AMUT/17	TURBO IMPLODER	mc106
AMUT/18	FONTS2PRINTER	mc107
AMUT/19	SVD	mc107
AMUT/20	MC-PROGRAMS	mc107
AMUT/21	CHP&SAVE-PREFS	mc107
AMUT/22	CIDITEIP	mc108
AMUT/23	DISKEDITOR	mc108
AMUT/24	5 UTILITY	mc108
AMUT/25	OROLOGIO PARLANTE	mc108
AMUT/26	LSLAB	mc110
AMUT/27	DIRWORK	mc111
AMUT/28	SCREENMOD	mc111
AMUT/29	SYSINFO	mc111
AMUT/30	SUPERDUPER	mc111
AMUT/31	PRFONT	mc113
AMUT/32	TG	mc113
AMUT/33	ICONS	mc113
AMUT/34	TURBOGIF	mc113

VARIE

AMVR/01	FRACTUS	mc108
AMVR/02	RUBRICA, DACIA & GESTFATT	mc109
AMVR/03	FUNZ3D	mc109
AMVR/04	PLAYSMUS	mc110
AMVR/05	MULTI PLAYER	mc111
AMVR/06	DRAWMAP	mc111
AMVR/07	TOTAMIGA	mc112
AMVR/08	AUTO	mc112
AMVR/09	SOUNDMASTER	mc112
AMVR/10	AMIGA L8	mc112
AMVR/11	FRACTAL	mc112
AMVR/12	SPECTROGRAM	mc114
AMVR/13	CHEMESTHETICS	mc114
AMVR/14	DAY2DAY	mc114
AMVR/15	CEMENTO ARMATO	mc115
AMVR/16	CORTES	mc115
AMVR/17	TUCANENTA	mc115
AMVR/18	CALORIEBASE	mc115

MACINTOSH

COMUNICAZIONE

MICO/01	RED RYDER	mc110
MICO/02	ZTERM	mc115

EDUCATIVO

MIED/01	KID PIX	mc107
MIED/02	NUMBER TALK	mc107
MIED/03	ALPHA TALK	mc107

GIOCO

MIGI/01	STELLA OBSCURA	mc106
MIGI/02	PARARENA	mc106
MIGI/03	VIDEO POKER FOR FUN	mc106
MIGI/04	SPACE STATION PHETA	mc106
MIGI/05	STRATEGO	mc106
MIGI/06	THE LAWNZAPPER	mc107
MIGI/07	MACTRIS	mc107
MIGI/08	CANFIELD	mc107
MIGI/09	YAHTZEE	mc108
MIGI/10	GLIDER	mc108
MIGI/11	MACNINJA	mc108
MIGI/12	GLIPHA	mc108
MIGI/13	MONOPOLY	mc109
MIGI/14	GOLF	mc109
MIGI/15	WHEEL	mc109
MIGI/16	GUNSHY	mc109
MIGI/17	MEGAROIDS	mc110
MIGI/18	SHUFFLEPUCK	mc110
MIGI/19	CRIMINALS	mc111
MIGI/20	SQUIX	mc112
MIGI/21	HOTEL CAPER	mc112
MGI/22	RISIKO	mc115
MGI/23	SPACE INVADERS	mc115
MGI/24	CONTINUUM	mc115
MGI/25	QUESTER	mc115
MGI/26	SCEPTERS	mc115

GRAFICA

MIGR/01	CALENDAR MAKER	mc106

SPREADSHEET

MISP/01	BIPLANE	mc112

STACK

MISK/01	FOOD 1	mc111
MISK/02	BUSINESS 1	mc111
MISK/03	SOUND 1	mc111

UTILITY

MIUT/01	OLIVER'S BUTTONS	mc107
MIUT/02	POPCHAR	mc107
MIUT/03	RAMDISK	mc108
MIUT/04	SCROLL2	mc109
MIUT/05	DECK EDITOR	mc109
MIUT/06	BANNER MAKER	mc110
MIUT/07	SPEEDOMETER	mc110
MIUT/08	LOODLE	mc112
MIUT/09	FAST FORMAT	mc112
MIUT/10	SOUND MASTER	mc112
MIUT/11	STUFFIT CLASSIC	mc112
MIUT/12	DISKDUP+	mc114
MIUT/13	DTPPRINTER	mc114
MIUT/14	FOLDER FROM HELL	mc114
MIUT/15	NUMBERCRUNCH	mc114
MIUT/16	PASTE-IT	mc114
MIUT/17	SAVE A TREE	mc114
MIUT/18	MACBINARY	mc114
MIUT/19	DOCMAKER	mc115

VARIE

MIVR/01	RIDICOLO	mc108
MIVR/02	ELIZA	mc109
MIVR/03	HYPERSTAR	mc113

Compilare e spedire a: ***MCmicrocomputer***

Desidero acquistare il software di seguito elencato al prezzo di **L. 8.000 a titolo (ordine minimo: tre titoli).** Per l'ordinazione inviare l'importo (a mezzo assegno, c/c o vaglia postale) alla: Technimedia srl, Via Carlo Perrier 9, 00157 Roma.

dischetti da ☐ 3.5" ☐ 5.25"

Codici:

Totale dischi ☐ x 8.000=Lire

Manuali in italiano:

☐ TSPD/01 AS EASY AS ☐ TUTI/01 HARD DISK UTILITIES

☐ TVAR/02 CHEKMATE ☐ TWPR/05 GALAXY

Totale manuali ☐ x 8.000=Lire

Nome e Cognome

Indirizzo

CAP/Città

Telefono

MCmicrocomputer non offre alcuna garanzia e non si assume alcuna responsabilità sugli eventuali danni diretti o indiretti derivanti dall'utilizzo del software distribuito

TEUCO DB Maker taglia tempi e costi fino all'

80%

Un bel risparmio davvero, se sviluppate in Clipper. E una bella riduzione delle possibilità di errore, senza cambiare il modo di lavorare.

Tra l'altro, visto che parliamo di risparmi, Teuco DB Maker costa molto meno di quanto immaginiate.

E molto meno di programmi analoghi.

Se volete dare un taglio deciso a tempi, costi ed errori, chiedete Teuco DB Maker al vostro rivenditore, o direttamente a noi. Naturalmente siamo a disposizione per ogni tipo di informazione, anche per telefono.

Se chiamate subito il nostro numero verde, la riduzione di tempi e costi sarà del 100%.

NUMEROVERDE 1678 - 44071

TEUCO

TEUCO S.r.l. Via Filanda, 15
29100 Piacenza - Tel. 0523/36738 - 31700

® Clipper è un marchio registrato dalla Nantucket Corporation.