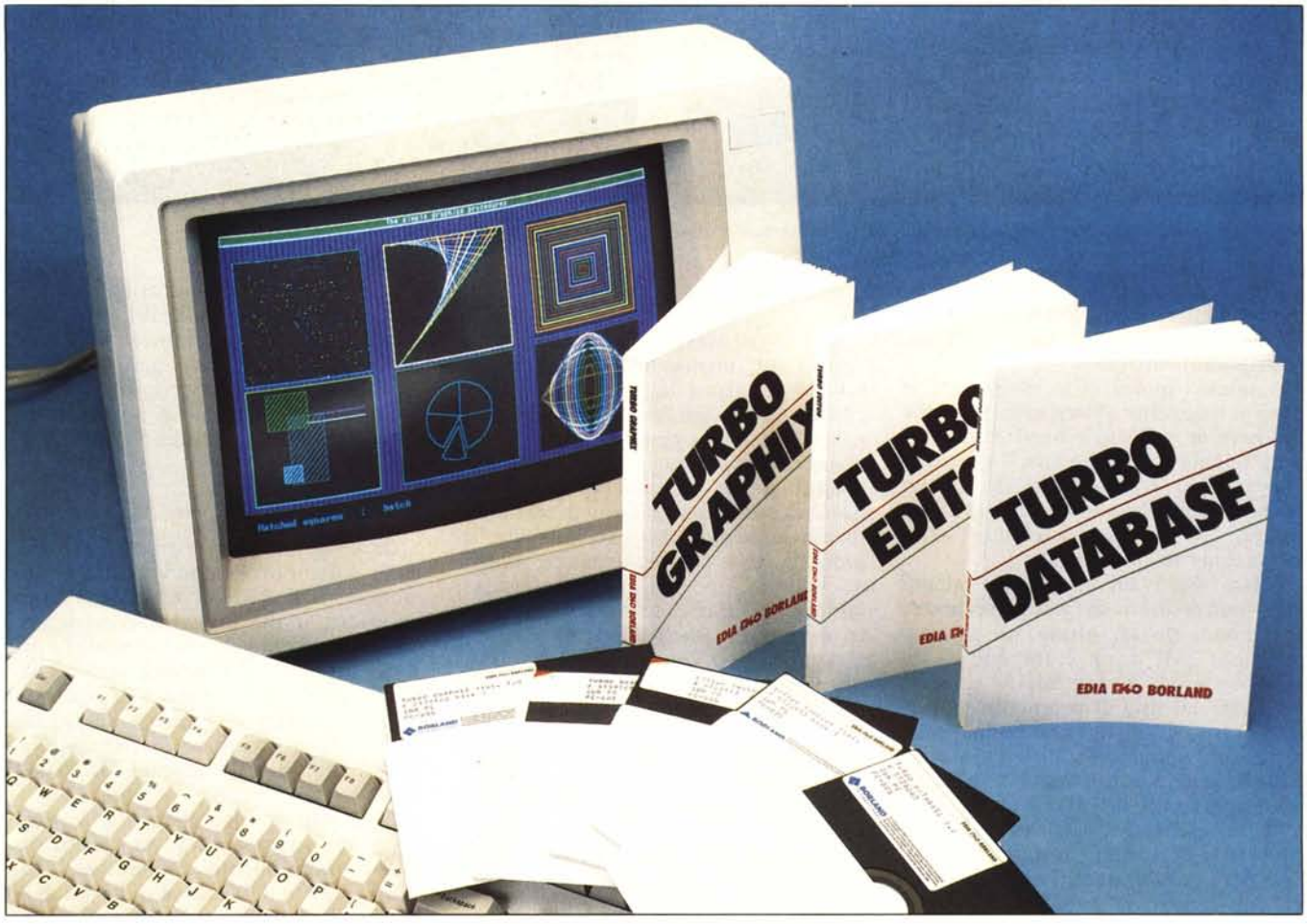
# Borland Turbo Pascal Toolbox




di Sergio Polini

*Quando abbiamo esaminato il Turbo Pascal (MC n. 62), abbiamo sottolineato che non è possibile compilare separatamente moduli di funzioni e procedure, in quanto la compilazione di ogni programma avviene «in una volta sola»: tutto deve essere allo stato di codice sorgente. Questo può essere un limite (temperato dalla notevole velocità del compilatore), ma ha sicuramente avuto benefiche conseguenze per gli utenti.*

*Ogni programma comporta infatti qualcosa di specifico, ma anche molte routine di carattere generale (controlli sull'input dei dati, ordinamento, un sistema di menu, ecc.), per le quali capita non di rado di poter utilizzare tecniche o algoritmi noti. Ecco quindi che si trovano sul mercato librerie di funzioni già pronte, che ognuno può incorporare nei propri programmi.*

*Così facendo si ottengono diversi vantaggi. In primo luogo un cospicuo risparmio di tempo, come è evidente, ma anche maggiore efficienza e minore possibilità di errori: non si tratta infatti solo di soluzioni già pronte, ma anche del risultato del lavoro di programmatori esperti, assistito dalla più aggiornata teoria e sottoposto al vaglio di numerose installazioni.*

*Prima del vistoso successo del Turbo Pascal, tuttavia, l'utente poteva spesso disporre solo di «scatole chiuse»: librerie in codice oggetto, di uso agevole e flessibile, ma solo linkabili al proprio programma così come erano, senza possibilità né di esaminarne l'intimo funzionamento né di adattarle ad eventuali particolari esigenze. Il sorgente veniva offerto solo a prezzi sensibilmente più elevati.*

*Quando però i clienti della Borland sono diventati centinaia di migliaia, costituendo un mercato di dimensioni decisamente accattivanti, i produttori hanno dovuto cambiare registro, immettendo sul mercato librerie di funzioni nell'unico modo possibile per il Turbo Pascal, il codice sorgente. Quello che una*

*volta era solo un costoso optional è quindi diventato la norma, e il prezzo contenuto di tutti i prodotti della Borland (quasi sempre sotto i cento dollari) ha imposto una analoga economicità anche per i loro accessori. Con poca spesa si può quindi venire in possesso di software di ottima qualità, utile non solo per svolgere compiti specifici, ma anche per vedere da vicino come è fatto il software «professionale», per procedere velocemente dal primo apprendimento del linguaggio alla capacità di realizzare programmi sempre più sofisticati. Editor, DataBase e Graphix Toolbox ne sono un esempio principe.*

*A proposito. Chi non conosca il Pascal potrebbe forse ritenere troppo difficile l'apprendimento di un nuovo linguaggio e delle avanzate tecniche di programmazione che questo consente, o magari potrebbe sentirsi scoraggiato chi abbia sperimentato solo altri compilatori più tradizionali. Niente paura: MC non lascerà cadere l'argomento, e cercherà nei prossimi numeri di proporvi esempi quanto più possibile agili di cosa si può fare con il Turbo Pascal, cose semplici e ardite, tecniche eleganti e trucchi sfacciati. Siete naturalmente invitati a dire la vostra; sapete bene che le indicazioni e i suggerimenti che ci perverranno saranno tenuti nella massima considerazione.*

**Produttore:**
*Borland International 4585 Scotts Valley Drive Scotts Valley, CA 95066, USA.*
**Distributore per l'Italia:**
*EDIA Borland - V.le Cirene 11, 20135 Milano*
**Prezzi** *(IVA esclusa):*

| | |
|---|---|
| *Turbo Database Toolbox* | *L. 149.000* |
| *Turbo Graphix Toolbox* | *L. 149.000* |
| *Turbo Editor Toolbox* | *L. 149.000* |

## Esplorando

Avete una buona familiarità con il Basic, magari ve la cavate anche con l'Assembler, ma considerate che se esistono tanti altri linguaggi una ragione ci deve pure essere. Provate quindi a cimentarvi con il Turbo Pascal, tanto per vedere.

Sapete che il Pascal si è rivelato utilissimo per l'insegnamento degli algoritmi e delle strutture di dati fondamentali. Vi basta quindi spulciare in qualche libreria ben fornita per trovare testi (spesso in inglese, ma anche tradotti o «made in Italy») ricchi di spunti interessanti.

Sfogliate, leggete, meditate, e vi accorgete di quanto siano potenti strumenti quali le variabili di tipo puntatore e l'allocazione dinamica della memoria.

Un array consente l'immediato accesso ad ogni suo elemento, ma ha una sua propria dimensione predefinita e risulta piuttosto scomodo quando dovete inserire, eliminare o spostare qualcosa. L'ordine degli elementi di un array è dato dalla loro stessa rappresentazione fisica: consecutive locazioni di memoria. Per alterare un dato ordine bisogna quindi intervenire su questa rappresentazione, riscrivendo su se stesso (in tutto o in parte) l'array.

Una lista è invece una successione logica (non fisica) di elementi, collegati tra loro mediante puntatori: A B e C sono l'uno dopo l'altro solo nel senso che il puntatore di A «punta» a B (è cioè una variabile che ha come valore l'indirizzo di B), e quello di B punta a C. Per scambiare di posto B e C, o inserire un nuovo elemento tra questi due, basta modificare il valore dei puntatori senza spostare fisicamente alcun elemento (potreste riguardarvi l'articolo di Andrea de Prisco su «File, liste e chiavi primarie» nel numero 50 di MC). Una lista non ha una dimensione prestabilita: per aggiungere un nuovo elemento D basta «allocare» la memoria che ne conterrà la rappresentazione fisica ed assegnare il corrispondente indirizzo al puntatore dell'elemento precedente (poniamo che sia C); analogamente, per poi eliminarlo, si assegna a questo puntatore un altro valore (ad es. l'indirizzo di A - e si ottiene così una lista circolare - o magari «nil», cioè nulla) e si libera la memoria precedentemente allocata, lasciandola pronta per eventuali future necessità.

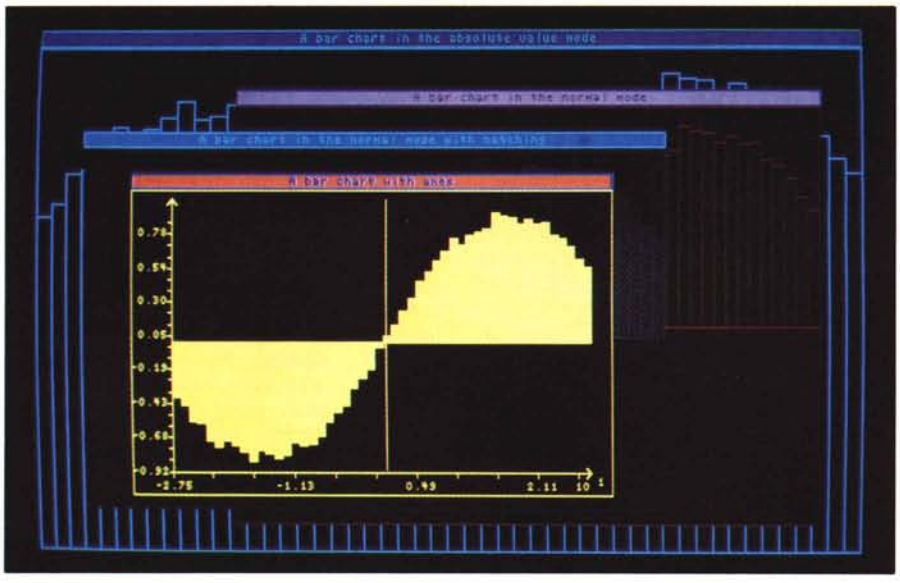
Immaginiamo ora che dobbiate realizzare un programma che consenta di scrivere dei testi, non necessariamente come suo compito primario (quale un editor o un word processor), ma anche solo come funzione accessoria. Pensate ai campi «memo» del DBIII. Non ci vuole molto per vedere che un array di n stringhe sarebbe ben poco conveniente. Inserire, copiare, muovere e cancellare delle righe richiederebbe marchingegni algoritmici complicati e pesanti, sprechereste memoria per ogni testo che contenesse meno di n righe, entrereste in crisi se vi servisse la riga (n+1)-esima.

Molto meglio lavorare con liste e puntatori.

Continuando a sfogliare leggere e meditare, vi accorgete che le stesse tecniche usate per la gestione delle liste possono essere impiegate anche con strutture di dati più complesse: alberi binari, alberi AVL, B-alberi, ecc. I B-alberi, in particolare, vi appaiono preziosi per una gestione efficiente di file di dati: ad ogni campo chiave si associa un file indice, si realizza questo come un albero in cui ogni «nodo» è una «pagina» che contiene delle terne <valore del campo chiave> - <numero del corrispondente record nel file di dati> - <numero della pagina con le chiavi successive> e vi trovate a poter operare sui vostri file come quando in DBIII date il comando INDEX ON <campo chiave> TO <nome del file indice>.

Se poi volete realizzare dei grafici, magari a partire dai dati dei vostri archivi, sapete che potete trovare algoritmi in gran numero in molti linguaggi; basta pensare alla rubrica di grafica di Francesco Petroni.

Ma come va se volete delle «finestre» grafiche? Aprire una finestra sul video è concettualmente semplice: si memorizza da qualche parte quello che appare sul video, si «apre» la finestra, ci si scrive o ci si disegna dentro, e poi la si «richiude» rimettendo al suo posto quello che appariva sul video prima dell'«apertura». Tutto sta a creare le opportune strutture di dati.



Se le vostre finestre non si sovrappongono, per memorizzare il video può bastare anche un array «allocato» una volta per tutte; se però volete gestire finestre di dimensioni variabili, che si aprano l'una sull'altra, dovete poter allocare ogni volta strutture di diversa ampiezza e manipolare un vero e proprio «stack» di queste strutture. Anche qui variabili dinamiche e puntatori risultano di grande aiuto.

Il problema è che sapere quali tecniche di programmazione possono essere utilizzate è solo il primo passo; per arrivare al programma completo c'è ancora molta strada da fare.

I prodotti che ora esamineremo vi propongono una comoda guida.

## Scrivendo

Un manuale, due dischetti e due programmi completi con tutto il relativo codice sorgente: un editor (FIRST-ED) e un word processor (MicroStar). Così si presenta l'Editor Toolbox: strumenti per realizzare programmi per scrivere con un IBM o compatibile. Programmi «completi» vuol dire che, appena aperta la confezione, potete subito usarli; non vi sono particolari difficoltà, in quanto i vari comandi sono per lo più gli stessi dell'editor del Turbo Pascal (e del WordStar), e due tabelle sul manuale vi consentono di individuare rapidamente le differenze.

Alcune caratteristiche dei due programmi appaiono decisamente interessanti. Ambedue effettuano lo scrolling orizzontale (con una lunghezza massima della riga fino a 32767 caratteri) e il ritorno a capo automatico (word wrap), potete definire margine destro e sinistro, ed anche «riformattare» un paragrafo con un comando Ctrl-B simile a quello del WordStar. Il FIRST-ED consente inoltre di aprire più finestre sul video, e quindi di tenere sott'occhio diverse parti di uno stesso testo o di lavorare contemporaneamente su testi diversi, anche spostando dei brani dall'uno all'altro. Il MicroStar differisce dal FIRST-ED in quanto presenta una interfaccia utente più moderna, con i suoi menu «pull down», ma consente di aprire solo due finestre e solo su due porzioni dello stesso testo. Offre in compenso la stampa di un file in background, la visualizzazione del contenuto della directory, la memorizzazione su disco dei parametri di configurazione (auto indent e word wrap sì o no, margini, intervallo di tabulazione, ecc.).

La tentazione di considerarli prodotti a se stanti è forte, ma bisogna resistere! Se provate infatti a usarli così come sono potreste anche restare delusi. Potete ad esempio operare su blocchi di testo (per cancellarli, copiarli, muoverli), ma solo su blocchi di righe intere: non si può definire un blocco che inizi o finisca nel mezzo di una riga. Vi sono i consueti comandi per la ricerca e sostituzione, ma ogni volta che si trova qualcosa la riga corrispondente diventa la prima dello schermo, dando così luogo ad uno scrolling verticale a volte inutile e fastidioso. Non vengono visualizzati né stampati il grassetto o il sottolineato. È certo proponibile una incorporazione dell'uno o dell'altro in altri programmi che richiedano funzioni accessorie di editing, ma non siamo ancora al WordStar 5.0!

In realtà FIRST-ED e MicroStar non sono altro che esempi molto ben curati di un possibile modo di utilizzare le tecniche di programmazione illustrate nella prima parte del manuale. Questo muove dalla definizione di alcuni termini di «text editing», dalla rapida descrizione dei diversi possibili tipi di editor e di diverse possibili implementazioni, per giungere poi ad illustrare le strutture di dati utilizzate nei due programmi: una lista doppiamente concatenata (in cui cioè ogni elemento «punta» sia al precedente che al successivo) di «descrittori di finestre», contenenti puntatori al testo, a sua volta gestito mediante un'altra lista doppiamente concatenata.

Studiare queste strutture di dati e le tecniche per la loro manipolazione vuol dire da un lato capire sul serio come si può lavorare con le liste (nessun esempio può valere quanto un programma completo), dall'altro mettersi in grado di modificare il FIRST-ED o il MicroStar secondo le proprie esigenze. Buona parte del loro funzionamento non è altro che il risultato di tecniche di gestione di liste di record e, una volta che si sia acquisita familiarità con questi strumenti, non è difficile togliere, aggiungere o cambiare quello che si desidera.

Un editor «full screen» comporta anche la necessità di sincronizzare l'immissione dei caratteri con il continuo aggiornamento dello schermo, e questo deve essere operato con grande rapidità. Il manuale illustra quindi una tecnica per il coordinamento dell'input da tastiera con l'output su video, il cui aspetto più interessante è forse dato dalla presenza di un «aggancio» per procedure da eseguire nei tempi morti, tra la pressione di un tasto e la successiva. Esempio tipico (realizzato nel MicroStar) è la stampa in background di un file mentre se ne sta editando un altro.

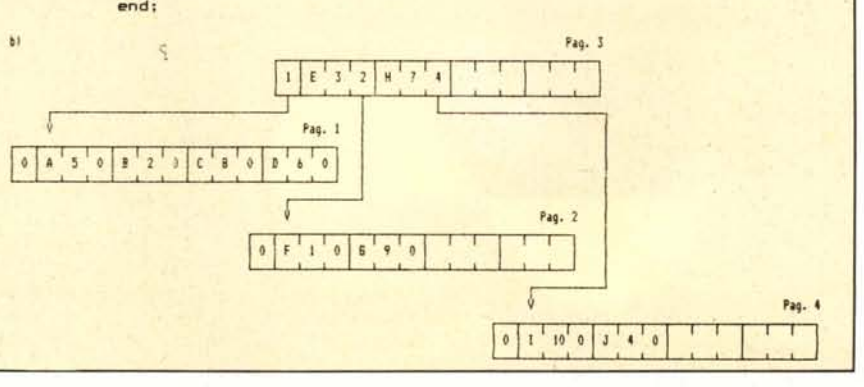
Va sottolineato che si tratta in questo caso di tecniche di programmazione di uso generale. Quello che distingue infatti questo Toolbox dagli altri due è proprio il fatto che viene proposto il sorgente di due programmi completi; non vi si trova quindi solo ben illustrato quanto serve per il text editing, ma anche un gran numero di spunti interessanti per qualsiasi programma. Ne sono esempio la sincronizzazione tra input da tastiera e aggiornamento del video e la stampa in background, ma anche la gestione di un buffer per i caratteri digitati, l'accesso alla directory, la possibilità di memorizzare permanentemente nello stesso codice oggetto su disco del programma i parametri di configurazione, l'organizzazione del video in finestre, la realizzazione di un sistema di menu «pull down».

Si tratta quindi di un vero e proprio corso di programmazione avanzata in Pascal in generale e in Turbo Pascal in

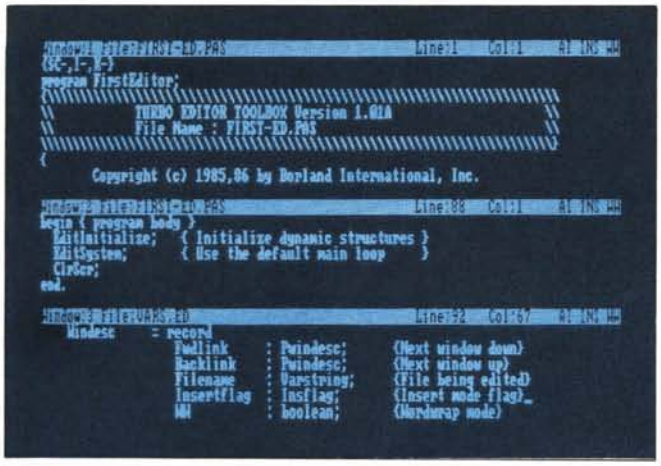
a)

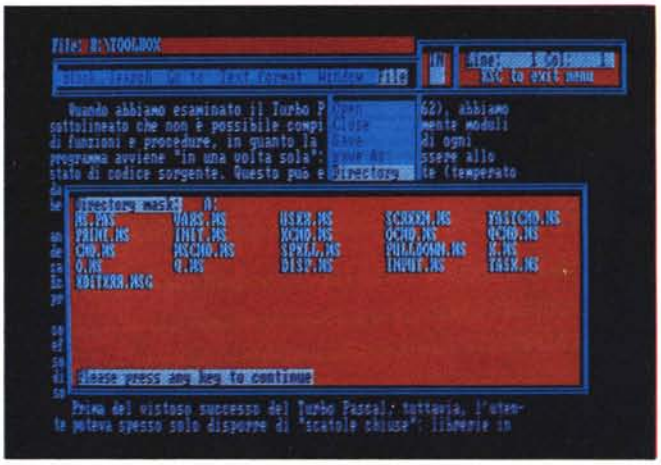
```
type
  terna = record
    chiave : char;
    numrec : integer;    { numero di un record nel file dati  }
    numpag : integer;    { numero di una "pagina" dell'indice }
  end;
  pagina = record
    n : integer;         { numero della "pagina precedente" }
    t : array[1..4] of terna;
  end;
```

b)




*Esempio di dichiarazione di tipi (a minimi termini...) per un B-albero (a), struttura di un B-albero per l'indice di un file di dati con un campo chiave di tipo carattere (b).*




*L'editor FIRST-ED consente di aprire diverse finestre sul video. Qui vediamo nelle finestre 1 e 2 l'inizio e la fine del file FIRST-ED.PAS, e nella terza l'inizio della dichiarazione del tipo «windesc» nel file VARS.ED, che contiene le dichiarazioni delle costanti, dei tipi e delle variabili globali del programma.*

*Il MicroStar offre una interfaccia utente caratterizzata da menu «pull down». Premendo i tasti F10 - F - D, si apre una finestra che mostra il contenuto della directory; in questo caso potete vedere i file del secondo dischetto, con il sorgente del programma.*

particolare: vengono infatti usate frequentemente anche alcune caratteristiche tipiche del compilatore, quali le procedure «GetMem» e «FreeMem» (più elastiche delle tradizionali «New» e «Dispose»), «Move» e «FillChar» (analoghe alle funzioni «memcpy» e «memset» del C), l'array predefinito «MemW» (che consente di accedere alla memoria come con i PEEK e POKE del Basic).

## Archiviando

Il DataBase è stato il primo Toolbox realizzato dalla Borland, e si presenta quindi con una veste più tradizionale: sul dischetto trovate una libreria di funzioni, un programma di utility (GINST.COM), qualche «demo». È anche l'unico prodotto dei tre che può girare anche in ambiente CP/M.

GINST crea programmi di installazione che consentano agli utenti di installare un programma compilato adattandolo alle caratteristiche del proprio terminale, così come si può fare con il TINST.COM per lo stesso Turbo Pascal. Si tratta di una utility destinata ad agevolare la commercializzazione di programmi realizzati in Turbo Pascal, in quanto evita la necessità di compilare versioni multiple di uno stesso programma secondo le diverse possibili configurazioni hardware. Non è possibile tuttavia (ma verrebbe da dire: ovviamente) servirsene per trasportare un programma da un sistema operativo ad un altro, e non appare indispensabile se si rimane nello standard IBM e suoi cloni; in ogni caso, se dovesse servire, GINST c'è.

Di interesse sicuramente maggiore appaiono le librerie di funzioni.

Capita frequentemente di dover scrivere programmi che gestiscano archivi di dati, e non è certo agevole scrivere routine che siano al tempo stesso efficienti e... a prova di bug! Il DataBase Toolbox vi offre proprio questo.

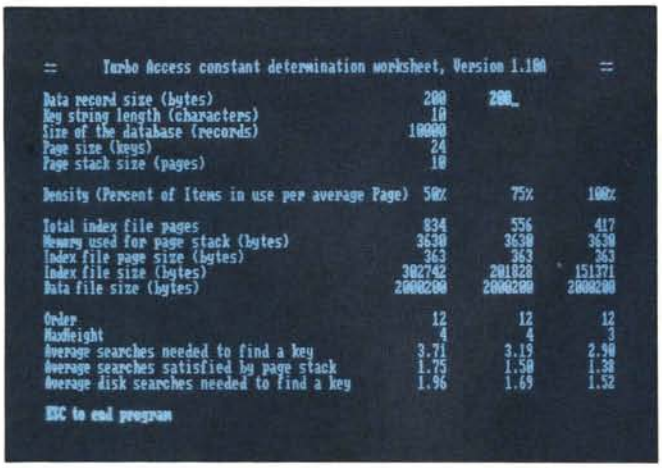
Tutti sappiamo che capita molto spesso di dover cercare un particolare record in un file di dati, o accedere sequenzialmente ad alcuni record secondo il contenuto di uno o più campi chiave, e quindi secondo un criterio logico che può essere ben diverso dall'ordine fisico dei record nel file. Sappiamo anche che a volte conviene riscrivere il file portando l'ordine fisico a coincidere con un certo ordine logico, a volte conviene aprire dei file indice che consentano, ad esempio, di scorrere il file di dati secondo criteri di volta in volta diversi.

Abbiamo quindi sia una routine di ordinamento, sia un insieme di routine per la gestione di file indice realizzati con i B-alberi. Il codice sorgente non è molto commentato (a differenza di quello dell'Editor), ma non si poteva pretendere di più: sia un quicksort non ricorsivo con uso, se necessario, di file temporanei sul disco, sia gli algoritmi per l'uso dei B-alberi sono piuttosto sofisticati e sarebbe ben arduo seguirne la logica solo scorrendo il codice sorgente, commentato o no. Chi voglia un approfondimento teorico ha comunque una facile soluzione: il libro di Niklaus Wirth «Algoritmi + Strutture di dati = Programmi» (tradotto in italiano dalla Tecniche Nuove) è la dichiarata fonte di ispirazione della routine di ordinamento, ed anche il codice per la gestione dei B-alberi è molto simile a quello proposto da Wirth (con la differenza che i suoi alberi crescono solo in RAM, quelli del Toolbox vanno e vengono dal disco. Non è un dettaglio da poco!). Si tratta poi di materiale che in linea di principio va usato così com'è, senza per questo dover rinunciare a nulla. Sia il sort che i file indice consentono infatti una notevole flessibilità.

Per ordinare i record di un file si deve includere nel proprio programma il file SORT.BOX e scrivere due procedure e una funzione: la procedura «Inp» dovrà passare i dati alla routine di sort, la funzione «Less» verrà chiamata da questa per confrontare i dati, la procedura «OutP» invierà i dati ordinati sullo schermo, su un file, sulla stampante o dove si vorrà. Dato che Inp, Less e OutP vanno scritte dal programmatore, questi dispone della massima libertà per quanto concerne l'origine e la destinazione dei dati e le caratteristiche del confronto; è possibile quindi, ad esempio, ordinare un file con chiavi multiple (il manuale contiene chiare indicazioni al riguardo).

Per gestire un file di dati mediante file indice si devono includere nel proprio programma i file ACCESS.BOX, ADDKEY.BOX, GETKEY.BOX e DELKEY.BOX e definire alcune costanti: in particolare l'ampiezza massima di un record, la lunghezza massima dei campi chiave, l'ampiezza di una «pagina» del B-albero (il numero delle «terne» per ogni pagina), il numero di pagine da tenere in RAM. Si può avere così accesso alle routine di apertura e chiusura dei file (dati e indici), di inserimento e cancellazione di record, di ricerca del record corrispondente ad un dato valore di un determinato campo chiave. Si può anche scorrere il file (con le procedure «NextKey» e «PrevKey») come se lo si stesse leggendo sequenzialmente dopo un sort.

Al momento dell'apertura di un file indice si può scegliere se consentire o no chiavi «doppie». Facciamo un esempio. Supponiamo che vi sia un archivio Clienti con due campi chiave: il codice del cliente e la sua città di residenza. È chiaro che ad ogni cliente




*Insieme al DataBase Toolbox viene offerto un programma SETCONST che aiuta a scegliere i valori migliori per le costanti richieste da ACCESS. BOX, in particolare l'ampiezza di ogni «pagina» di un file indice e quella dello stack di pagine da tenere in RAM.*

spetterà un suo codice, che non vi potranno essere due clienti con lo stesso codice, e che invece è ben probabile che vi siano molti clienti in ogni città. Vogliamo quindi che il programma rifiuti (inviandoci un messaggio d'errore) l'immissione per un nuovo cliente di un codice che è già stato assegnato ad un altro, ma vogliamo anche poter produrre un elenco di tutti i clienti che risiedono in una data città. Basta in questo caso aprire l'indice dei codici escludendo le chiavi doppie, quello delle città abilitandole (il tutto è ben esemplificato nel programma BTREE.PAS sul dischetto).

Possiamo anche scegliere se aggiornare i file indice ogni volta che si aggiunge/cancella/modifica un record nel file di dati, oppure solo subito prima di servirsi degli indici (il manuale illustra una procedura «RebuildIndex» dagli effetti analoghi a quelli del comando REINDEX del DBIII).

L'unico vero limite del prodotto è rappresentato dal numero massimo di record che si possono gestire: 65535 con i B-alberi, 32767 con il sort; non sono poche comunque le applicazioni per le quali 30000 record risultano più che sufficienti. Per il resto, vi è un solo punto che potrebbe forse giustificare una modifica del sorgente: i record cancellati vengono prima «marcati», e quindi riutilizzati scrivendoci sopra altri record aggiunti in seguito; non è pertanto possibile recuperare un record cancellato solo logicamente, come consente il DBIII, né è prevista una routine di «impaccamento» analoga al comando PACK di questo. Se lo si desidera, è tuttavia molto facile modificare le procedure «NewRec» e «DeleteRec» in modo da implemmentare la sola cancellazione logica, e scriversi poi una procedura che copi il file di dati escludendo i record cancellati (operazione cui seguirà ovviamente un «RebuildIndex»).

In generale l'uso delle diverse routine è piuttosto semplice e ottimamente illustrato dai due programmi «demo» presenti sul dischetto, TBDEMO.PAS e BTREE.PAS; il primo propone tra l'altro un esempio di ricostruzione dei file indice mediante la procedura «RebuildIndex», e il secondo comprende anche interessanti procedure di creazione e gestione di maschere per l'immissione dei dati.

Il solo punto che richiede una certa attenzione è la scelta del valore da assegnare a quelle costanti cui si accennava prima, soprattutto per il dimensionamento delle «pagine» dei B-alberi e dello stack di pagine da tenere in RAM: una scelta sbagliata può condizionare sensibilmente le prestazioni di un programma. Qui abbiamo una prova della grande cura con cui la Borland realizza i propri prodotti: un'appendice del manuale spiega sinteticamente cosa sono e come funzionano i B-alberi, proponendo anche una breve bibliografia, e un programma sul dischetto (SETCONST.PAS, anch'esso in sorgente) vi consente di valutare immediatamente le conseguenze di diverse scelte in termini di numero medio degli accessi al disco necessari per cercare una chiave.

## Disegnando

Commentare singolarmente o anche solo elencare le 114 funzioni e procedure descritte nel manuale del Graphix Toolbox e i 55 demo contenuti nei due dischetti richiederebbe troppo spazio, ma è possibile seguire un preciso filo conduttore grazie al fatto che non si tratta di un «aggregato» di routine grafiche, ma di un vero e proprio «sistema», implicitamente articolato su primitive, attributi, sistemi di coordinate e gestione di finestre.

Per primitive qui intendiamo le cose più semplici che si possono disegnare e con cui si possono costruire immagini più complesse: punti, linee rette e spezzate, poligoni, cerchi ed ellissi, settori circolari, testo (si dispone, oltre a quello standard, anche di un set di caratteri disegnati su una matrice 4x6, posizionabili in qualsiasi punto sullo schermo e con le dimensioni che si desidera). Ad ogni primitiva sono associate strutture di dati funzioni e procedure che ne determinano alcuni dei modi di rappresentazione. Gli attributi sono altri «modi» di validità più generale: rimangono cioè in effetto per le primitive che verranno tracciate fino a che non venga cambiato l'attributo; «SetLineStyle» consente ad esempio di scegliere tra linee continue o tratteggiate in varia maniera, «SetAspect» di fissare l'eccentricità delle ellissi (che appaiono come cerchi con un Aspect pari a 1).

Vengono previsti diversi ambienti hardware (IBM PC con CGA, EGA o scheda Hercules, Olivetti/AT&T, Zenith Z100, IBM 3270-PC), ognuno dei quali ha un suo proprio sistema di coordinate «assolute» (così il manuale traduce l'espressione «screen coordinate»), in cui ci si muove contando i pixel a partire dall'angolo in alto a sinistra dello schermo: tracciare grafici disponendo solo di questo sistema di riferimento può risultare piuttosto noioso, e anche d'ostacolo quando si vuole scrivere un programma compilabile su macchine diverse. Viene quindi offerta la possibilità di lavorare con coordinate «arbitrarie» («world coordinates»: quelle proprie del «mondo» popolato dagli oggetti che avete in animo di disegnare, ad es. un piano cartesiano con x che va da $-10$ a 20 e y da $-100$ a 1000), rendendo automatica e del tutto trasparente la conversione dal sistema logico a quello fisico.

Le finestre sono per così dire la «cornice» del mondo che volete rappresentare, e vengono definite in termini di coordinate assolute (una appendice del manuale spiega come usare la procedura «DefineWindow» in modo praticamente indipendente da un particolare ambiente hardware, ed anche come posizionare il testo con analoga libertà); al loro interno è possibile fare però riferimento a coordinate puramente logiche, impostate con la procedura «DefineWorld». Un'altra procedura («FindWorld») permette anche di attivare un sistema di coordinate commisurato al grafico che volete disegnare, in funzione dei suoi valori massimi e minimi in orizzontale e in verticale. Con «DrawAxis», infine, si tracciano gli assi ortogonali e le scale dei valori in ascissa e in ordinata, eventualmente ritagliando un'area di disegno più piccola della finestra e delimitata da un proprio bordo.

Ogni finestra appare così come una entità a se stante, al punto che può essere salvata in RAM (in uno schermo virtuale o in uno stack di finestre) o su

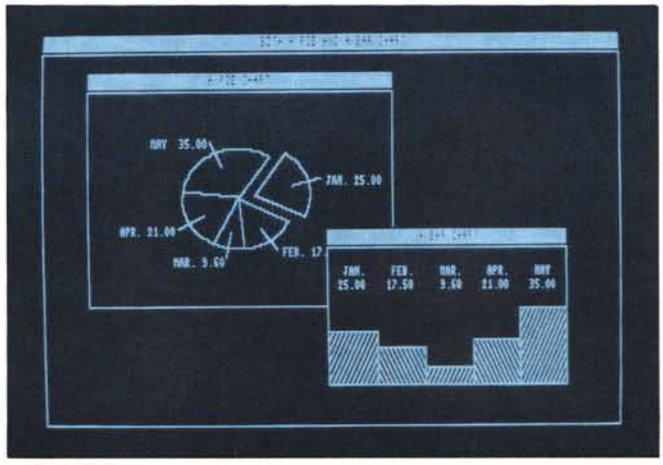
disco, e spostata orizzontalmente o verticalmente sullo schermo. Spostare una finestra pone il problema di non distruggere quello che appare sul video, soprattutto se vengono rappresentate contemporaneamente diverse finestre, ma diventa possibile proprio grazie alle diverse tecniche di memo rizzazione in RAM e su disco, alle quali si aggiungono tecniche analoghe per l'intero display.

Il tutto è costruito in modo da consentire una notevole flessibilità. Proviamo a fare un esempio con una linea. Se è attivato il «clipping» non vengono tracciate le parti che cadrebbero oltre i margini della finestra attiva; se è abilitato il «modo finestra» la linea viene tracciata con riferimento al sistema di coordinate arbitrarie (cioè logiche). Sono queste le condizioni normali di operatività, ma possono essere ribaltate quando si vuole, e anche ignorate: «DrawLine» traccia la linea in coordinate arbitrarie o assolute, secondo che sia abilitato o no il modo finestra, non oltrepassando i margini della finestra se è attivo il clipping, andando pure oltre in caso contrario; «DrawLineClipped» usa in ogni caso le coordinate assolute e il clipping, «DrawStraight» disegna in ogni caso una linea orizzontale in coordinate assolute e senza clipping. E i poligoni possono essere spostati, ruotati, ingranditi o impiccoliti, si può variare la velocità di movimento verticale delle finestre, visualizzarle con o senza un titolo, ecc.

Non è difficile fare grafica tecnica o commerciale su queste basi, ma la Borland ha comunque voluto facilitare l'uso del sistema offrendo alcune procedure già pronte per grafici a torta e istogrammi e per curve di interpolazione. Per ogni «pie chart» si può scegliere se evidenziare alcuni settori circolari staccandoli dal resto della figura, per gli istogrammi se disegnare un tratteggio alternato nelle varie barre; in ambedue i casi è possibile includere le descrizioni letterali dei valori rappresentati dai singoli settori o barre. Per l'interpolazione vengono proposte le «cubic splines» e i polinomi di Bezier; con le prime si ottengono grafici che passano per tutti i punti dati in modo più intuitivo di quanto sarebbe possibile ottenere mediante l'interpolazione lineare di Lagrange (che produce a volte curve ben diverse da quelle che si traccerebbero cercando di congiungere «a mano» i diversi punti), e con calcoli meno complessi; i secondi generano curve «più morbide», in quanto passano con certezza solo per il primo e per l'ultimo dei punti e sono solo «influenzate» dagli altri.

Si tratta sì di procedure utilizzabili così come sono, ma soprattutto di esempi di uso del sistema; non è difficile infatti sostituirle con altre eventualmente più consone alle proprie esigenze. Un esempio: la procedura «Spline» non traccia il primo e l'ultimo dei punti da interpolare, e richiede che le ascisse dei singoli punti siano ordinate, non consente cioè di tracciare grafici che vadano «avanti e indietro» non vi sono problemi tuttavia ad impiegare un algoritmo più elastico (quale quello usato da SPLINES.PAS, un programma che potete trovare nell'area Pascal di MC-Link). Più in generale, chi abbia acquisito la necessaria familiarità con il Graphix può intervenire su diverse sue caratteristiche; il manuale spiega tra l'altro quali costanti del sistema è possibile o utile modificare, quali funzioni o procedure di basso livello sono convenientemente utilizzabili per scopi diversi da quelli previsti.

*Esempio di grafico a torta e di istogramma (PIEHISTO.PAS sul disco).*




L'unico vero limite ... non è un limite del prodotto, ma dell'hardware. Si usa infatti sempre la grafica ad alta risoluzione e questo vuol dire, di norma, solo due colori (uno per lo sfondo e uno per testi e disegni). Fino alla penultima versione l'unico modo di «vivacizzare» i disegni era dato dalla possibilità di definire una «trama» per lo sfondo delle finestre (un po' come fa il FrameWork). È appena uscita tuttavia la versione 1.07 che consente di utilizzare le maggiori capacità della scheda EGA anche per quanto riguarda i colori: con una scheda da 256 K si dispone di 640x350 pixel e si possono usare contemporaneamente fino a 16 colori scelti da una palette di 64. Il manuale non ne parla (si fa prima ad aggiungere un secondo dischetto - tutto EGA - alla confezione che a stampare un nuovo manuale), ma un file EGA.DOC contiene tutte le necessarie descrizioni e istruzioni.

## Concludendo

L'Editor Toolbox vi consente di incorporare funzioni di editing nei vostri programmi o di farvi un editor su misura; non potendosi definire un modo «unico» di scrivere con il computer, vi si offrono numerosi strumenti e vi si propongono due programmi completi, diversi tra loro per struttura e funzionalità, perché possiate prenderli ad esempio. Probabilmente ogni impiego del Toolbox nei vostri programmi richiederà qualche modifica delle routine, ma non si tratta di un compito difficile. Direi anzi che il codice presenta a tratti qualche ridondanza, quasi a rendere ancora più agevole la comprensione dei listati. Il DataBase e il Graphix Toolbox possono essere invece utilizzati così come sono ed infatti, per quanto siano in teoria possibili numerose varianti (soprattutto nelle routine di «basso livello»), si possono dire ben definiti i compiti e il funzionamento di un sort o di un sistema di file indice, e anche di un sistema grafico organizzato intorno a primitive, attributi, «screen» e «world coordinate», finestre. Tuttavia anche qui l'adattamento alle proprie esigenze non è né escluso né difficile.

In tutti e tre i casi i manuali offrono poi una documentazione chiara e completa e sono stati ben tradotti in italiano dalla EDIA. Si tratta quindi di prodotti di ottima fattura, con prezzi senza dubbio convenienti, vivamente consigliabili a tutti quelli che vorranno avvicinarsi al mondo della programmazione «seria».

Correte qualche rischio? Una sola cosa può dirsi con quasi assoluta certezza: potrete rimanere fedeli al Pascal, oppure continuare ad esplorare per poi sposare quel formidabile macro-macro-assembler che è il C, o magari il LISP, il linguaggio che ha fatto del «LISt Processing» e della gestione dinamica della memoria la sua stessa ragione d'essere. Ma è ben difficile che vi torni la voglia di sgambettare col Basic.

MC