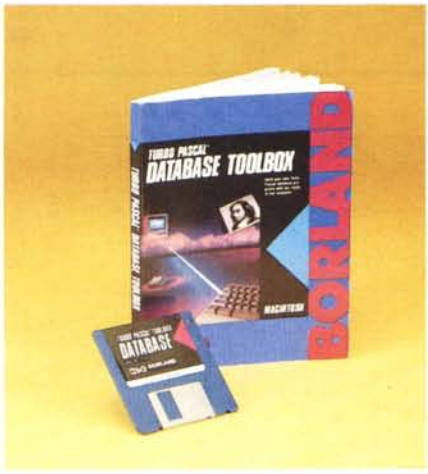
di Raffaello De Masi

Il Turbo Pascal Database Toolbox

Continuando con la politica di aggiornamento ed ampliamento continuo dei suoi prodotti, la Borland sta dedicando ampio spazio al settore Macintosh, producendo tutto quel set di utility e programmi accessori già presenti nell'area MS-DOS finalizzati al potenziamento del suo prodotto principe, il Turbo Pascal. Ecco quindi, come logica ed intelligente



evoluzione di questo linguaggio, comparire sul mercato il Turbo Pascal Database Toolbox, che gli utenti MS-DOS già conoscono, e che rappresenta, comunque, solo il primo di una serie di tool dedicati che in tempi brevi appariranno sul mercato. Vediamone in anteprima le caratteristiche, visto che si tratta di un prodotto nuovo, appena uscito sul mercato

Devo confessare una verità; la prima sorpresa avuta aprendo questo pacchetto non ha niente a che vedere con il Toolbox o con il Turbo Pascal in genere; fatto sta che a pagina 0 (vale a dire la prima pagina di copertina) leggo testualmente: «Questo manuale è stato prodotto, nella sua interezza, con Sprint™, il word processor professionale prodotto dalla Borland».

Accidenti, che perfezione! Avevo sentito parlare a più riprese di questo WP, senza però mai sapere nulla di più dei soliti sentito dire; invece, da quello che vedo, i risultati sono di eccellente qualità, e questo Sprint può dare punti ai migliori WP sul mercato, ed addirittura a qualche programma di Desktop Publishing.

Ma torniamo al Toolbox. Per chi già conosce quello dedicato al sistema operativo MS-DOS le novità non sono molte, visto che il pacchetto ricopia in maniera abbastanza fedele lo schema e le direttive del suo precedente fratello. Si tratta di un package contenente, in definitiva, due tool principali destinati a facilitare lo sviluppo dei programmi in Turbo Pascal:
il Turbo Pascal Sort System
il Turbo Pascal Access System
ambedue sviluppati in forma di moduli, in modo da poter essere utilizzati oltre che autonomamente, come parti precostituite di programmi in altri più ampi, di più grande respiro.

Il primo utilizza il ben noto algoritmo del QuickSort per eseguire un facile e veloce ordinamento dei dati. Il programma è articolato in modo da creare poche difficoltà all'utente, consentendo, tra l'altro, tecniche di sort con singola o multipla chiave, o analisi di blocchi di dati diversi anche congiuntamente, nell'ambito dello stesso programma; inoltre si tratta di un programma discretamente evoluto, visto che utilizza, alla bisogna, lo spazio sulla memoria di massa quando il file da manipolare è troppo grande per essere contenuto interamente nella RAM della macchina che si sta usando.

Il secondo programma rappresenta invece un mezzo facile ed efficiente per conservare e richiamare informazioni contenute in file di dati di grande ampiezza. Turbo Pascal Access System accede alle informazioni sia in maniera casuale che in maniera sequenziale, servendosi di una efficiente tecnica che descriveremo.

Ambedue le applicazioni, presentate sia in sorgente che compilate, possono essere utilizzate tali e quali o come parte di programmi costruiti dall'utente, o, ancora (e crediamo sia questo il miglior spirito con cui questo package va preso), come miniera e spunto di notizie e di tecniche di programmazione cui attingere per scopi anche diversi da quelli in intestazione del pacchetto stesso.

Ciò premesso, diamo un'occhiata più da vicino al contenuto del package stesso, dividendo equamente lo spazio disponibile in questo articolo tra le due applicazioni principali.

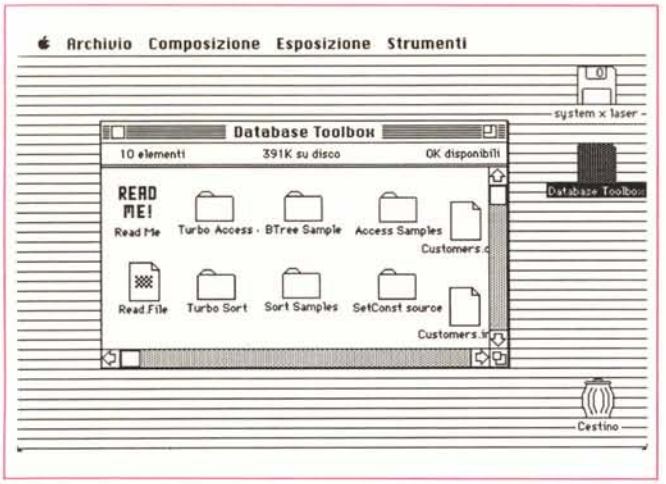
Il Turbo Pascal Sort

Turbo Pascal Sort analizza e riordina ogni tipo di dato facilmente ed efficientemente, senza richiedere all'utente più di una piccola collaborazione; utilizza l'algoritmo noto sotto il nome di *Quicksort* ed una sorta abbastanza sofisticata di utilizzo di memoria virtuale che, in teoria, assicura un riordino di un numero di dati pressoché illimitato (in pratica, se i dati richiedono più memoria di quella disponibile, lo spazio presente sul disco viene automaticamente utilizzato come una estensione della memoria stessa); ovvia, quindi la pressoché mancanza di limiti di quantità di dati manipolabili, in presenza di un hard disk di generose dimensioni.

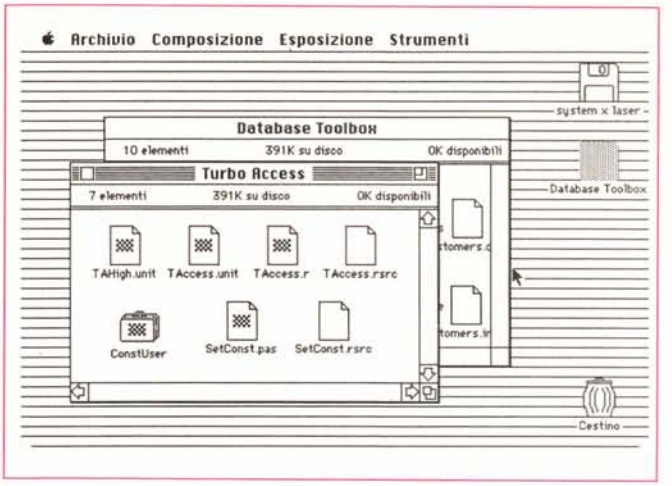
Le routine presenti, dedicate al «sorting» sono in effetti due, precompilate: *"Sort"* ed *"LSort"*, del tutto analoghe e funzionanti sullo stesso principio; la differenza sta nel fatto che la prima manipola

Turbo Pascal
DataBase Toolbox
Borland International
4585 Scotts Valley Dr.
Scotts Valley
CA 95066 U.S.A.

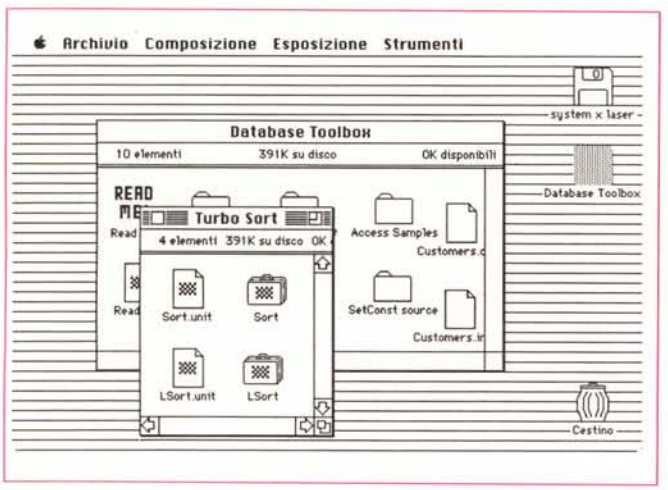
Edia Borland Italia
Via Cirene
Milano
Prezzo: *L. 179.000*



◀ *Figura a.*
Il contenuto del disco fornito col package.



▲
Figura c.
Idem con le unità di Access; si notino le due risorse a destra.



◀ *Figura b.*
Le due unità di sort con cui si parla nel testo, in forma sorgente (a sinistra) e compilate.

un massimo di 32767 valori, dati, mentre la seconda è destinata a blocchi di dimensioni più elevate (oltre due milioni), ma con l'inevitabile scotto di una lentezza maggiore. Si tratta di due funzioni invocabili ambedue nel modo:

function TurboSort (Numero_dei_dati: Integer; InpPtr, LessPtr, OutPtr: ProcPtr):Integer;

La routine di Sort divide il suo lavoro in tre fasi:
fase di input
fase di sort
fase di output,

ognuna delle quali è guidata da una routine specifica, codificata da un indirizzo (che tra l'altro, viene restituito da un operatore del tipo @). La procedura di input è guidata da una chiamata ad un indirizzo specificato dall'InpPtr presente nella call alla routine principale di TurboSort e che è rappresentata generalmente da un bruto loop del tipo repeat-until o while-wend che passa gli oggetti da manipolare alla routine di sort. Tramite questa tecnica di input vengono ricevuti i dati (gli oggetti) direttamente dall'operatore o da una periferica, e, una volta completata, risulta terminata anche la fase di sorting. La funzione successiva (Less), del tipo
function LessRec (var x,y,: CustRec): Boolean;
affida ad una funzione booleana il compito di stabilire quale dei dati è il più piccolo. Infine l'ultima routine (Output), una procedura, è la più banale, e, anche essa chiamata una volta sola; senza parametri, invia allo schermo (od alla periferica di output prescelta) la lista degli oggetti riordinati.

Secondo la solita prassi di inclusione del Turbo Pascal, l'unità di Sort deve essere inclusa nel programma principale secondo la ben nota e collaudata sintassi Borland:

program nome_del_programma;
{$U Sort}
uses..., Sort; {ev. LSort}

o, utilizzando l'utility *UnitMover*, presente nel pacchetto Turbo Pascal, forzare direttamente nel programma l'unità desiderata, evitando così la routine di inclusione guidata da [$].

Circa l'utilizzo della memoria, Turbo Pascal alloca automaticamente, in presenza della routine di Sort, lo spazio in memoria man mano che i dati vengono manipolati. Lo spazio minimo di utilizzo dinamico richiesto è pari a tre volte la lunghezza dell'elemento da utilizzare stesso; ancora Turbo Pascal esegue, ove possibile, il sorting interamente in memoria centrale; se lo spazio non è sufficiente abbiamo già detto che tratta il disco come una estensione di memoria.

La cartella di Sort, sul dischetto, contiene i seguenti file-applicazioni

Sort.unit il codice sorgente per l'unità Sort
Sort il file compilato dell'unità precedente
LSort.Unit il codice sorgente dell'altra unità (L-Sort)
LSort idem come Sort

Il manuale contiene inoltre una descrizione accurata ed esauriente, oltre che estremamente dettagliata e commentata, delle sottounità (funzioni, procedure) presenti nel programma Sort-LSort, che rappresentano una utility nella utility, visto che è possibile prelevarle di sana pianta e demandarle ad altri scopi in altri programmi.

Il Turbo Pascal Access

Un dentista di paese, mr. U. Flossmore, ha un piccolo ma frequentato studio dentistico, un Macintosh ed una copia del suo Turbo Pascal (vogliamo sperare non pirata!), e desidera usare il suo computer per tener traccia della sua clientela. Ha preso in considerazione la possibilità di spendere 500$ per l'acquisto di un ben noto database ma ha scoperto, appena in tempo, che questo programma dovrà poi essere accuratamente «customizzato» per funzionare alla bisogna. Decide, allora, di armarsi di buona volontà e di prepararsi da solo il programma che gli serve.

La prima fase, di analisi del problema, lo porta a considerare i fattori che il pro-

Struttura di una B+ Tree

Sotto il nome di struttura ad albero (Tree Structure) va una ben nota tecnica di ricerca, già diverse volte trattata sulle pagine della nostra rivista. Sebbene non sia questo il momento di affrontare, ancora una volta, l'argomento, e mancandoci, oltre tutto lo spazio necessario, ci è parso opportuno spendere solo qualche parola per semplificare una tecnica consorella della precedente, piuttosto specializzata, la B+, utilizzata dall'unità Access del Toolbox TurboPascal per Mac.

La ricerca ad albero è una delle strade più intuitive e semplici per eseguire lo scanning, l'analisi di un file dati alla ricerca di un parametro; ma non sempre la via più facile è la più breve e la più efficace per giungere ad un risultato; vediamo perché.

Facciamo un esempio; immaginiamo di aver inserito in un database dei record contenenti delle registrazioni relative ai clienti del buon dr. Flossmore; valori inseriti: nome, numero di telefono, indirizzo, età, ecc. Se i record non sono ordinati secondo uno schema logico e finalizzati, ogni ricerca di dato (ad esempio un numero di telefono di un cliente) verrà eseguito pedissequamente, con l'analisi, uno per uno, dei record, sequenzialmente, alla ricerca del valore cercato; se, per caso, il cliente cercato si trova in fondo all'elenco, l'attesa può essere lunga o comunque snervante.

La ricerca ad albero consiste nel principio di leggere attraverso il data file senza guardare fisicamente ogni record; in altre parole un albero è un metodo di organizzazione di dati; esso si basa su un elemento essenziale, il nodo, che rappresenta, in pratica una biforcazione della struttura dell'albero stesso. I nodi possono essere di tre tipi: un nodo radice, il capostipite, per così dire, un nodo interno (intermedio) ed un nodo finale (o terminale) (in gergo tecnico e, è il caso di dire, fiorito, un nodo «leaf», foglia). In un albero a struttura binaria, ogni nodo può avere 0, 1 o 2 figli (biforcazioni); e viceversa, un nodo figlio è un nodo puntato da un «padre» che a sua volta può essere «figlio» di un altro. In altre parole una struttura ad albero è fatta di nodi, e di puntatori (indirizzi) che connettono tra loro i nodi.

Per trovare un dato in una struttura ad albero, questo deve essere scorso un nodo alla volta, partendo dalla radice. Ad ogni nodo corrispondono quattro possibilità:

— Il corrente nodo contiene il dato ricercato; la ricerca è finita.

— Il dato ricercato è al di sotto del nodo, per cui la ricerca continua.

— Il dato è più grande dell'attuale nodo, per cui occorre tornare indietro al nodo precedente.

— Il corrente nodo è un terminale e non contiene il dato cercato; la ricerca non ha avuto esito, o la tecnica di ricerca binaria non può essere applicata convenientemente.

Il vantaggio di una ricerca di questo tipo è ovvio; ogni volta che si giunge ad un nodo e si esegue una decisione, metà dei successivi elementi viene esclusa dall'algoritmo di analisi; da ciò l'efficienza e l'utilità della tecnica stessa.

La ricerca di tipo binario si dimostra efficace in molti scopi; ma esistono altre tecniche, ancor più avanzate, che possono realizzare gli stessi risultati di efficienza in casi in cui la ricerca binaria mostra un po' la

gramma sarà chiamato a manipolare; tanto per esemplificare, per ogni paziente sarà necessario tenere traccia del nome, indirizzo, telefono, pagamenti parziali, numero delle sedute, stato dei denti, ecc. (in totale, circa 300 caratteri di informazioni diverse). Tenendo conto di dover manipolare una clientela di circa 2500 pazienti, occorrerà un file di circa 750k.

Che metodo usare per immagazzinare i dati su disco? Una scorsa completa del file, con un Mac Plus, in modo sequenziale, richiederebbe circa un minuto (che talvolta può sembrare un'eternità); occorre una soluzione più rapida: una sistemazione random, magari in ordine alfabetico; il tempo medio d'accesso, in queste condizioni, sarebbe intorno ai 5 secondi; buono, ma ancora migliorabile. Non sarebbe meglio dividere il disco in blocchi virtuali, destinati ognuno a una differente lettera dell'alfabeto? Ma il problema si complica; come dimensionare queste partizioni visto che è prevedibile che i pazienti, con iniziali in «A» saranno probabilmente in numero maggiore di quelli in «Z»?.

Il buon dottor Flossmore comincia a sentirsi un po' scoraggiato (d'altro canto lo studio si sta riempiendo di gente che attende, e lui non ha ancora preso una decisione neppure su come affrontare il problema); che fare? Ecco arrivare un rappresentante Borland e risolvergli in un attimo tutto con Turbo Pascal Access.

Questa è una favoletta scritta proprio così (anche se abbondantemente riassunta) nella pagine 27-29 del manuale e crediamo non possa dare una idea migliore delle funzioni di Turbo Access. Questa unità è davvero un data base in sedicesimo (solo per forma, ma non per potenza, visto che riesce a manipolare fino ad oltre due miliardi, spazio permettendo, di record, utilizzando la (non molto) ben nota tecnica B+Tree, di cui parliamo a fianco); i 2500 pazienti del dr. Flossmore saranno manipolati in una media di due secondi, secondo un complicato gioco di flussi che consente di accedere pressoché istantaneamente ad ogni dato del file. Turbo Pascal Access è un programma ovviamente molto più complesso del precedente (tra l'altro nel manuale occupa uno spazio ben maggiore), e, ovviamente necessita di uno studio ben più accurato del fratello; per questo motivo i tecnici Borland hanno sapientemente resa questa routine molto più end-user, vale a dire che anche l'utente «scarso» in programmazione potrà giungere ad utilizzare l'unità rapidamente, senza dover per questo comprendere effettivamente ed intervenire in maniera massiccia nell'unità fornita; tanto per intenderci, mancano solo poche personalizzazioni, peraltro facilmente realizzabili, per poter disporre di una data base efficiente e discretamente potente, soprattutto (il che non guasta) in termini di velocità.

La complessità della realizzazione è tale che lo spazio di riferimento tecnico dell'unità, sul manuale d'istruzioni, è circa quattro volte più ampio del corrispondente dedicato a Sort. D'altro canto lo stesso argomento è diversificato in una serie di routine dedicate, definite di alto livello, che sono rappresentate tra l'altro da un codice più compatto. Anche qui, come in precedenza, esiste una disanima accurata di tutte le routine presenti nel programma che, come innanzi, viene fornito sia in forma compilata che come sorgente.

Conclusioni

Il principio, anzi sarebbe il caso di dire il voto, abbracciato dalla Borland pare essere il seguente: «Spendere poco per il meglio; se credete a questo principio i nostri prodotti sono per voi». Rischiamo di ripeterci per la terza volta, dopo la prova del Turbo Basic e del Turbo Pascal, ma quello che abbiamo esaminato stavolta sarei più propenso a considerarlo come applicazioni più che come parti di linguaggio, in particolare l'Access, che somiglia proprio ad una di quelle scatole di montaggio così care agli americani, con la parte più difficile già predisposta che chiede all'utente solo qualche ora domenicale per la «customizzazione». Tanto per metterci qualche altra cosa, per buona misura non manca un gioco-esercizio-utility riservato ai programmatori, «SetConst Worksheet», che consente di eseguire verifiche e calcoli sugli ingombri dei file destinati ad essere manipolati dalle due unità principali.