Microsoft QuickBASIC
Borland Turbo Basic

di Raffaello De Masi

Basic, bel suol d'amore» cantavano i legionari alle prese con le catene del Modula 2 e con le liste simboliche del Lisp. E, per usare le parole di Buzzati in un suo famoso racconto (Il cane che ha visto Dio) «Mai che non ci fosse uno che non dicesse: 'Ah, i bei tempi', sospirando, e non c'era nessuno che non capisse a volo i bei tempi in cui ognuno poteva fare il suo comodo» quanta gente vorrebbe continuare, in tutte le manifestazioni della vita, a fare a modo suo, secondo il suo personale piacere! Oggi parlare di Basic è divenuto poco di moda, ognuno si professa cultore di questo o quel raffinato idioma informatico (e magari nasconde il suo bel listatone basicale sotto il pietoso velo di un compilatore) e, nei più raffinati salotti non si fa altro che parlare dell'ultimo nato di turno degli idiomi informatici che «bla bla bla...». Bene, oggi consentiteci di non essere ipocriti, e se è vera (come è vera) la considerazione che nessun costruttore si sognerebbe di costruire una sua macchina priva di questo

bistrattato linguaggio (HP, con il top delle sue macchine, tra cui il prestigioso 9000, ha introdotto una nuova release del suo Basic dalle prestazioni impressionanti) e se consideriamo che, nel mondo, sono stati pubblicati più di 15.000 titoli sull'argomento (come riferisce autorevolmente il manuale del Turbo Basic), dobbiamo per forza di cose concludere che senza Basic molti santoni (o autopresunti tali) del modo informatico starebbero in idilliaca quiete a foraggiare ruminanti sulle alte montagne dell'Irpinia.

La reticenza nei confronti del Basic non è comunque del tutto priva di fondamento ma, secondo chi scrive, gli errori che hanno portato a guardare con diffidenza questo idioma stanno altrove; il Basic è, senza ombra di dubbio, di gran lunga il linguaggio più semplice da imparare e da usare; ciò ha consentito ad una messe innumerevole di utenti, senza alcun background culturale nel campo dell'analisi e della tecnica di programmazione, di redigere programmi senza alcun costrutto logico, frutto di una enorme fatica da parte di chi li aveva creati, e privi di qualsiasi leggibilità per un qualsivoglia utente. Ciò ha portato a confondere il linguaggio col prodotto, in ciò confortato dal proliferare di numerose riviste che hanno fornito a piene mani ciarpame insulso e senza alcun nesso logico.

Da qui a bollare il Basic di ignominia il passo è stato breve. Perché ciò non è successo con altri linguaggi (ipocritamente autodefinentisi strutturati, autoesplicanti, autodocumentanti, ecc. ecc.)? Solo perché al «C» od al Cobol od allo stesso Fortran hanno lavorato generalmente persone dalla solida cultura informatica, che hanno prodotto materiale di pregio, e, soprattutto, non hanno la concorrenza (si fa per dire) del solito programmatore della domenica. Bisogna ammettere con molta onestà che è altrettanto facile scrivere un programma guazzabuglio, incomprensibile e pieno di turpitudini, in Pascal che in Basic; questione di «manico», come si dice dalle mie parti. Perciò, non facciamo di ogni erba un fascio, e se è vero, come fermamente credo, che almeno l'80-90% del software verticale prodotto da programmatori autonomi (parlo di programmatori seri!) è redatto in Basic, anche se poi compilato, siamo sinceri, ed accettiamo di professare le nostre tendenze (lo fanno i «diversi», a ragione; perché dovremmo vergognarci nel nostro habitat programmatorio, con alle spalle un linguaggio che, oggi, non ha nulla da invidiare ad altri, ma non sempre è il contrario!).

Dai tempi del 1964, quando ad Hanover, nel New Hampshire, Kenemy e Kurtz davano vita, assieme ad un gruppo di studenti, al più famoso ed utilizzato linguaggio di programmazione, molta acqua è passata sotto i ponti. Oggi il Basic è divenuto un linguaggio raffinato, potente, elegante, dotato di tutte le più sofisticate feature proprie degli idiomi più avanzati; è strutturato, può essere compilato, può gestire dinamicamente la memoria, può essere ricorsivo, può accedere direttamente ai livelli più bassi di programmazione. È divenuto portatile da macchina a macchina (si veda l'articolo su True Basic in Mac Corner di dicembre), è strutturato in maniera molto avanzata, ha abbandonato i numeri di linea, possiede caratteristiche di utilizzo nel campo matematico impensabili qualche anno fa (ricordo che nel 1982 comprai, per circa 3/4 di milione una ROM matrix per il mio HP 87, mentre il True Basic per il Mac, un linguaggio completo, con un'altrettanto efficiente gestione delle matrici costa oggi circa dieci volte di meno). Quale migliore occasione che provare, oggi, un po' del meglio che c'è in circolazione? E come non iniziare proprio con le due case che oggi sono divenute, de facto, le produttrici per antonomasia di linguaggi, vale a dire la Borland e la potentissima Microsoft?

Turbo Basic

Borland International Inc.
4585 Scotts Valley Drive
Scotts Valley
LA 95066

Edia Borland s.r.l.
Viale Cirene, 11
20135 Milano
Prezzo L. 199.000
memoria minima necessaria 384K

QuickBASIC

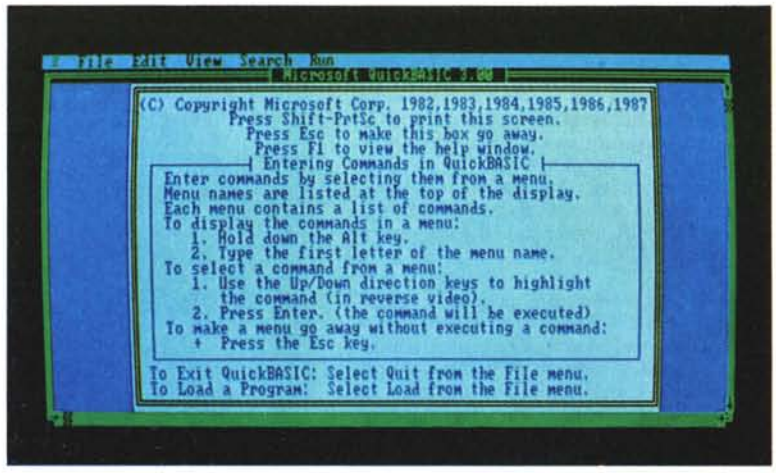
Microsoft Corp.
16011 NE 36th Way
Box 97017 - Redmond
WA 98073

Microsoft Italia
Via Michelangelo, 1
20093 Cologno Monzese (MI)
Prezzo L. 175.000
memoria minima necessaria 256K
Upgrade da 3.0 a 4.0: L. 70.000

Il Microsoft QuicKBASIC

Quick Basic, giunto alla edizione 3.0, è racchiuso in una doppia coppia di dischetti, differenti solo per le macchine di destinazione (se cioè prive o no di coprocessore matematico) e si presenta come un ambiente di sviluppo completo più che come un semplice linguaggio. Potente, facile da imparare ed usare, in possesso di una interfaccia utente semplice da utilizzare e dotato di inumerevoli facility, QB include un editor di schermo efficiente, un compilatore in linea immediato, un debugger di prim'ordine, un linker completamente (volendo) invisibile all'utente; in altre parole (così come, d'altro canto, avviene anche in TB) la classica trafila edit-compile-debug avviene senza mai uscire dell'ambiente, magari, volendo, affidandosi ad un esteso uso del mouse. La compilazione può avvenire senza interruzione al primo errore; in questo caso, sebbene, ovviamente, non sia generata alcuna applicazione .EXE, QB «ricorda» e mette a disposizione dell'utente tutte le locazioni degli errori, e, alla fine della compilazione, un comando speciale per la rapida correzione degli stessi.

Le caratteristiche più avanzate e preziose di QB possono essere così riassunte:

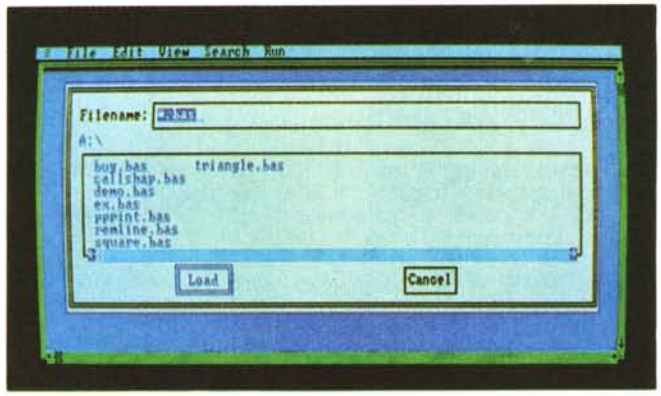


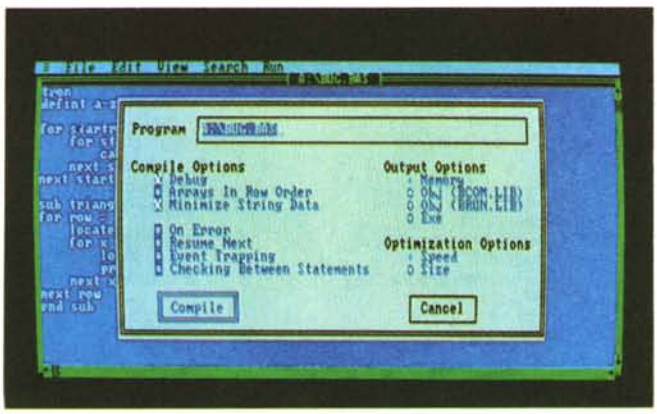
QuickBASIC: *lo screen d'apertura*

— costruzione di sottoprogrammi: esiste, così, la possibilità di suddividere i classici monolitici (l'unico vero grande problema) programmi Basic in moduli distinti. Tramite l'opzione $INCLUDE è possibile accedere a librerie, definite dall'utente, di comune utilizzo, senza che queste stesse librerie siano fisicamente presenti nel linguaggio stesso.

— Strutturalità avanzata, con caratteristiche che non hanno nulla da invidiare ai più sofisticati linguaggi: IF...THEN...ELSE multilinea, SELECT...CASE (presente solo dalla versione 3.0), DO LOOP, EXIT...FOR-DO e chiamate a sottoprogrammi con passaggio di variabili locali o globali, oltre ai più classici costrutti FOR...NEXT (manca, ancora, stranamente, il costrutto BEGIN...UNTIL) DEF FN

Operazioni di caricamento di un file, la selezione viene effettuata battendo il nome del file, mentre non è possibile la selezione con i tasti cursore.





Opzioni di compilazione-debug; si notino, tra le altre, le ottimizzazioni relative alla velocità di esecuzione (settate) ed alla dimensione.

(anche di tipo Long, e supportante anche un non comune EXIT DEF) permettono di disegnare programmi di elevata leggibilità, compatti, di agevole manutenzione, e, il che non guasta, di flessibilità e potenza elevata.

— Abolizione dei numeri di linea, ormai definitivamente abbandonati grazie al notevole miglioramento degli editor. Gli indirizzamenti dei GOSUB e GOTO vengono effettuati ad etichette alfanumeriche, anche questo con un notevole miglioramento delle caratteristiche di leggibilità.

— Gestione dinamica della memoria, con allocazione elastica della stessa in runtime, e risoluzione dei notevoli problemi spesso collegati alla allocazione statica, come accade, ad esempio, nel dimensionamento delle array. Le stesse array, inoltre, possono occupare fino a 64K di memoria. Ancora, sempre a proposito di array, è possibile controllare l'allocazione di array già dimensionate e l'analisi di file addizionali aggiuntivi (tramite, ad esempio, l'$INCLUDE) direttamente da Basic.

Tutto questo ed altro è illustrato e gestito da un pesante manuale di circa 650 pagine, che, comunque, non può essere utilizzato come tutorial del linguaggio e che presume, ovviamente, una buona conoscenza del DOS. Nello stesso manuale vengono consigliati alcuni libri utili da consultare, tra cui ci ha fatto piacere vedere citato il volume di Van Wolverton sull'MS-DOS, un eccellente manuale su questo standard.

Dopo i primi due capitoli di presentazione il manuale affronta un «guided tour», così caro agli implementatori d'oltre oceano, in cui vengono, soprattutto, illustrate le qualità dell'editor finalizzate al debug di uno dei programmi presenti nel dischetto. Viene presentato l'uso di una libreria esterna e la creazione di un file oggetto.

Dal capitolo 4 in poi si entra direttamente nella descrizione dell'ambiente di programmazione; secondo la più classica notazione MS-DOS, il linguaggio (con i suoi orpelli ed accessori) viene chiamato tramite il comando

qb

che può essere seguito da una innumerevole messe di opzioni, regolanti, tra l'altro, il caricamento automatico di un file (listato del programma), il settaggio di parametri relativi all'uso di schede grafiche o monitor a colori (la presenza di schede colore viene riconosciuta direttamente dal linguaggio); inoltre è possibile già caricare file di libreria, allocare l'ampiezza di buffer (fino a 32767 byte), presettare alcune direttive del compilatore. Si è in ambiente, con la riga di menu di testa selezionabile tramite la pressione di un singolo tasto (o tramite mouse, in questo pacchetto ampiamente supportato, e del tutto consigliabile anche per un uso non avanzato del linguaggio); una particolarità interessante è l'uso della opzione [shell], che consente, momentaneamente, di ritornare in ambiente DOS, del tutto funzionale, per eseguire altri programmi o comandi specifici di sistema operativo. Quick Basic resta in linea, disponibile. Battendo [exit] lo schermo QB appare nello stesso stato in cui era stato lasciato.

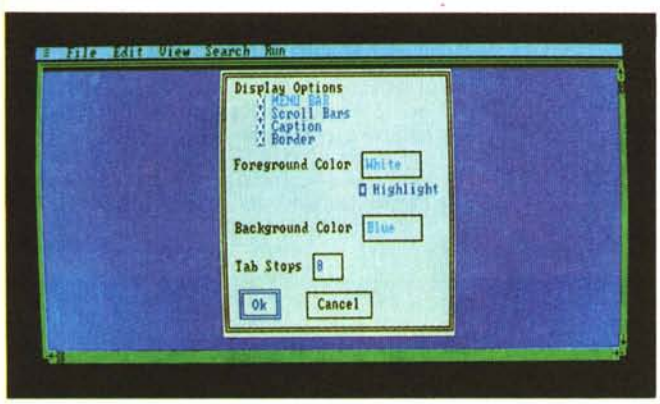
Oltre le opzioni [File] ed [Edit], piuttosto standard, il menu [View] consente una più agevole manipolazione dello schermo tramite il settaggio dei tabulatori, la scelta dei colori di default, l'uso delle barre di scroll. È ancora possibile installare-deinstallare la finestra d'errore. La compilazione avviene tramite l'opzione [Run], che permette, altresì, di settare diverse opzioni di compilazione, ivi compresa la destinazione dell'oggetto (.EXE, .LIB, ecc.).

Particolare cura è stata offerta alle operazioni di debug, sempre un po' critiche nei linguaggi compilati; oltre i classici TRON e TROFF, il linguaggio offre opzioni come il TRACE, STEP, e, infine, una originale e pratica feature, ANIMATE; tramite essa il programma scorre continuamente, ma, a seconda di una particolare opzione scelta (Slow o Fast), le linee vengono eseguite con intervalli di qualche secondo. Oltre a ciò, dopo il settaggio dell'opzione «Debug» (che è indipendente, eseguita com'è da menu, dalla opzione TRON), viene generato un codice oggetto un po' più ampio e lento, ma che

QuickBASIC b)

Opzioni del compilatore

Opzione	risultato
/d	setta il debug
/e	setta l'ON ERROR
/o•	impone l'opzione Obj (BCOM.LIB)
/q	opzione velocità
/r	array in ordine inverso
/s	evita l'opzione di ottimizzazione delle stringhe
/v	controllo del confronto delle istruzioni
/w	event trapping
/x	resume next



Finestra VIEW: selezione delle opzioni di editing su schermo.

testa, in fase di compilazione, una serie di eventi come:

— overflow dinamico
— limiti e dimensionamento delle array
— indicazione, sotto forma di locazione di linea, di errori in runtime
— verifica di corrispondenza di tutti i RETURN
— verifica di CTRL-BREAK.

Si tratta di una operazione, questa, consigliabile almeno nella fase di testaggio iniziale dei programmi, visto che, in caso contrario, non solo il programma non si ferma (generando il relativo messaggio d'errore), ma prosegue per vie imprevedibili, falsando qualsiasi logica di flusso e, d'altro canto, rendendo vana qualunque forma di debug.

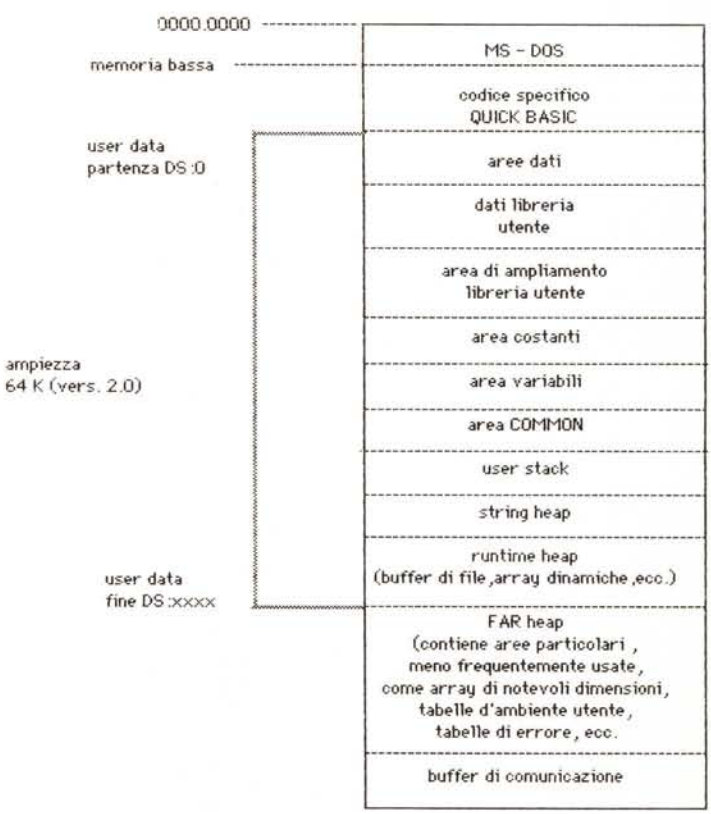
Lo stesso menu, comunque, possiede numerose altre opzioni che consentono di ottenere, tra l'altro, programmi il più veloci e meno ingombranti possibile, tramite una più accurata gestione della memoria.

Molta enfasi (giustamente) è data, nella illustrazione del linguaggio, ai sottoprogrammi, vere e proprie procedure pascaliane capaci di manipolare, come abbiamo già detto, variabili locali e globali, e, addirittura array, anche multidimensionali. L'utilizzo è del tutto analogo alle procedure, con la possibilità di sistemare sottoprogrammi anche in librerie esterne (un test ha dimostrato che è possibile nidificare sottoprogrammi appartenenti a librerie diverse) e con l'unica restrizione che salti indirizzati con GOTO e GOSUB possono solo avvenire nella stessa libreria.

Questa delle librerie è un'altra delle opzioni più potenti del linguaggio: pilotata dalla istruzione [buildlib] la creazione di una libreria utente è rappresentativa di un tool di programmazione estremamente efficace. Per chi vuol penetrare, ancora, nel più profondo del cuore del sistema, tramite la solita [CALL], Quick Basic può chiamare subroutine in linguaggio macchina (ovviamente sempre ammettendo un eventuale passaggio di parametri).

Come è nello stile di tutti i manuali di questo linguaggio la maggior parte del volume è dedicata (circa 400 pagine) a riferimento delle istruzioni e dei comandi. Il tutto è piuttosto standard (ovviamente visto nell'ottica dei canoni più avanzati di questo linguaggio), ma un esame approfondito del manuale di base e dell'update 3.0 (una settantina di pagine, di colore grigio, in premessa di manuale) ci permette di scoprire cose e feature di grande interesse. Vediamo così la possibilità di inserire breakpoint (al massimo 8; vera manna per i linguaggi compilati, dove l'analisi di flusso dei valori delle variabili è sempre disastrosamente fastidiosa), definire variabili Watch (Vigilanza), che saranno monitorate automaticamente ad ogni loro reperimento nel listato (chi conosce il Basic HP ritroverà un vecchio amico, il TRACE VAR) utilizzare in maniera l'efficiente coprocessori matematici come l'8087 o l'80287, adottare formati numerici in standard IEEE, definire costanti (finalmente) simboliche, da utilizzare al posto di valori numerici o di stringa definiti col classico [=]. Il manuale di riferimento si sviluppa in maniera ordinata, con una analisi ragionata, esauriente e circostanziata di tutte le istruzioni-funzioni-comandi, e con riferimenti incrociati ad altri comandi abbastanza numerosi. Per numerose istruzioni viene riferito il diverso comporamento compilatore-interprete, anche se non viene dedicato molto spazio ai programmi di tipo CHaiNed. Alla troppo spesso maltrattata e mal considerata DEF FN viene dedicato ampio respiro; essa, in ossequio ai canoni più avanzati, viene qui ampliata alla più potente FN multilinea.

La grafica appare abbastanza curata e, anche se non raggiunge, neppure alla lontana, implementazioni di altre macchine e linguaggi (v. HP), possiede alcuni statement particolarmente raffinati. Pur mancando un avanzato sistema di gestione grafica, alcuni comandi si comportano come vere e proprie macro. [DRAW], ad esempio, accetta, come argomento, una stringa che, con un formato



Mappa della memoria in runtime durante l'esecuzione di un programma (QuicKBASIC).

Fase di runtime di un programma, con chiamata ad una funzione esterna non disponibile (si noti il messaggio d'errore)

abbastanza simile all'HPGL (ma con uno standard diverso; peccato! si era fatto 30, si poteva fare 31) definisce il disegno da eseguire (le modalità di movimento della penna sono, tra l'altro, abbastanza simili a quelle adottate dal LOGO); CIRCLE consente di tracciare cerchi ed archi con pattern, colori e proporzioni diverse. SCREEN, specificatore del «mode» dello schermo supporta le schede MDPA, CGA, EGA, peraltro differenziate in sette tipologie diverse (Screen 0,1,2,7,8,9, e 10). Comandi specifici sono disponibili per il controllo della penna luminosa; il suono è regolato da un potente metacomando, PLAY, che, in maniera abbastanza simile a DRAW, accetta una stringa-macro in una sola istruzione.

Microsoft probabilmente pensa che, almeno in Basic, la sua metodologia di manipolazione dei file sia divenuta uno standard de facto; probabile, ma il manuale, sebbene del tutto esauriente, è forse, per questo argomento, fatto per chi già conosce in maniera non superficiale l'insieme di comandi che regolano le operazioni di I/O da file nel vecchio Basic. Pur essendo vero che il manuale non è un tutorial, avremmo preferito che all'argomento fosse dedicato maggior spazio per evitare che il neofita (che notoriamente, affronta la manipolazione dei file solo un certo tempo dopo aver acquisito una buona padronanza del linguaggio) subisca una battuta d'arresto nella fase di apprendimento.

Per concludere l'esame del manuale, un'appendice di una cinquantina di pagine, posta in posizione strategica alla fine del volume, è dedicata ai messaggi d'errore, che risultano corredati da efficaci spiegazioni e, ove possibile, da consigli sulle tecniche di recupero. Vengono, ancora, descritte tecniche diverse, come, ad esempio, la possibilità di usare una compilazione separata secondo i canoni classici, per chi proprio vuole fare come ai vecchi tempi.

Il Turbo Basic della Borland

Turbo Basic si presenta sotto forma di due dischetti pieni zeppi, e di un grosso manuale d'istruzione di circa 500 pagine (che, non vorremmo sbagliarci, pare redatto con un word processor piuttosto avanzato, un Mac ed una stampante laser), istruzioni cui vanno ad aggiungersi quelle comprese nei file «read_me» presenti sul dischetto stesso. Il dischetto comprende i seguenti file/directory:

TB.EXE: è il programma principale e rappresenta l'ambiente di sviluppo, editor, compilatore, biblioteca di runtime del linguaggio stesso. Battendo semplicemente TB, seguito dal [RETURN] si è direttamente in ambiente.

Upgrade 4.0 al QuickBASIC Microsoft

Proprio qualche giorno prima dell'uscita di questo articolo, Microsoft si comunicava tramite una serie di fax e riservandoşi di perfezionare successivamente il look del pacchetto, che era disponibile la versione 4.0 del suo QuickBASIC. Con la fretta necessaria a fornire l'informazione il più rapidamente possibile, riferiamo, in questo riquadro, sulle principali differenze tra la nuova release ed il pacchetto che è stato oggetto di prova.

QB, in questa ultima versione, supporta una serie di feature, alcune del tutto nuove, altre migliorate rispetto alla versione precedente, di cui, di seguito, forniamo qualche ragguaglio:

(dove N sta per Nuova feature ed M per Migliorata)

— presenza di tipi definiti dall'utente N
— supporto avanzato del coprocessore matematico M
— help in linea intelligente N
— integer di tipo long (32 bit) N
— stringhe di lunghezza fissa N
— controllo della sintassi al return (che meraviglia! n.d.r.) N
— I/O su file binari N
— istruzione FUNCTION N
— moduli multipli in memoria N
— compatibilità con Sidekick e Prokey M
— modo insert-sovrascrittura M
— tastiera con configurazione tipo WStar N
— ricursione (chi ha detto che la concorrenza non dà frutti?) N
— listing degli errori durante la compilazione separata M

Sono state migliorate, anche notevolmente, alcune caratteristiche del linguaggio, tra cui citiamo l'ampliamento di range dei numeri in formato IEEE, la riduzione di certe restrizioni in file di tipo INCLUDE, l'inserimento di «watchpoint», dalla funzione analoga alle watch variable descritte nell'articolo; inoltre è stato aumentato il numero dei breakpoint possibili. Alcuni statement, come SEEK, OPEN, SELECT CASE, DECLARE SETMEM, STATIC, AS CVL e numerosi altri sono stati aggiunti od aggiornati per una maggiore flessibilità dell'idioma. Alcuni comandi sono stati automatizzati tramite le solite combinazioni CTRL-SHIFT-ESC-DEL-ALT-INS ecc. e alcuni, del tutto nuovi, sono stati resi maggiormente intuibili. L'opzione DEBUG di menu, che vedete nella foto relativa al pacchetto oggetto di prova, è stata rimossa dal menu RUN, visto che in questa versione l'operazione di debug può essere eseguita in qualunque momento.

Programmi sorgente redatti con vecchie versioni (fin dalla 2.0) possono essere letti dal nuovo linguaggio, e ricompilati in questo nuovo ambiente. File di vecchio tipo, attraverso l'uso di un programma compreso nella documentazione, possono essere riconvertiti e resi trasparenti alla nuova versione.

Niente male, come update, visto il già eccellente livello della versione precedente! E interessante, fra l'altro, la possibilità di ricevere l'uprgrade dalle versioni, precedenti al costo di 70.000 lire.

aaa.BAS: (dove aaa è una stringa di caratteri) sono programmi sorgenti redatti in Basic, pronti per essere utilizzati.

TB.HPL: file di HELP in linea.

README.COM: programma per la lettura e la stampa del successivo.

README-UPDATE: si tratta di file testo (leggibili, comunque, con un qualunque editor), che possono essere anche non presenti su versioni nuove o più vecchie di quella da noi utilizzata, e contengono informazioni supplementari od errata corrige al manuale.

Prima di entrare in ambiente il manuale (dal settembre '87 è disponibile in italiano, in una traduzione fedele ed accurata) consiglia, se necessaria, una più accurata definizione dell'ambiente operativo, in modo da ridirigere il compilatore a specifiche opzioni, come l'uso del coprocessore 8087 o della scheda grafica CGA. Fatto questo si batte TB e si entra in ambiente principale che appare abbastanza simile a quello visto il mese scorso per il Prolog; siamo in Editor, un editor molto interattivo, già descritto esaurientemente su queste pagine anche da Giustozzi nella prova del Turbo C; si tratta, comunque, di un tool comune (nella maggior parte delle sue caratteristiche) a tutti i pacchetti della Borland, e che assomiglia, volutamente, nel suo utilizzo a WordStar della MicroPro Int. od al Multimate della Multimate Inc (niente impedisce, comunque, di utilizzare un qualsiasi wp per redigere il textfile, ma mi sembra una complicazione inutile); l'uso, anche per chi non conosce il principe dei wp è del tutto intuitivo, specie nella utilizzazione più immediata; ciononostante la pressione del tasto F1 consente di disporre di un Help in linea intelligente, che fornisce immediatamente un messaggio esauriente circa la situazione in cui ci si trova.

Chiamata al file di Help (intelligente) del linguaggio.

Turbo Basic: *la classica schermata d'apertura di tutti i linguaggi Borland.*

Saltiamo a piè pari la fase di editing (l'editor è talmente friendly da rendere la sessione di prova di questo un semplice tour, tranne poi a doverci ritornare per le feature più sofisticate) per avanzare imperterriti nel cuore del linguaggio. Ma prima di passare ad analizzare le caratteristiche stesse dell'idioma, vediamo alcune caratteristiche circa la compilazione; in ciò abbastanza simile alla maggior parte dei Turbo, Basic, attraverso le [OPTIONS] può accedere a diversificate possibilità di compilazione. Il comando [COMPILE TO] permette di scegliere di dirigere la compilazione stessa del programma per ottenere un programma .EXE, un file concatenato [ChaiNed, estensione .TBC] o di eseguire una immediata compilazione in memoria, il metodo più pratico durante la fase di test del programma stesso. Ancora è lecito selezionare alcuni «commutatori di compilazione», così riassumibili:

— opzione 8087
— break da tastiera
— bounds
— overflow
— stack test.

Con la prima opzione si «indica» al compilatore che è disponibile il coprocessore aritmetico, e TB si regola di conseguenza per generare l'oggetto con la migliore codifica possibile per le operazioni in virgola mobile. Ovviamente, settando questa opzione, il programma così generato non girerà su macchine prive dell'hardware necessario. In mancanza (default), il programma userà l'8087 solo se necessario e disponibile, altrimenti utilizzerà apposite routine soft per ottenere lo scopo (il tutto, comunque, comporta un rallentamento generale). A protezione di tutto esiste comunque un settaggio [SET87 = NO] che sbloccherà automaticamente il programma in caso di incompetenza hardsoft.

La seconda opzione è intuitiva. Con la terza il compilatore attiva un controllo sui controller delle array, soprattutto per la fuoriuscita di indici fuori intervallo. Allo stesso modo viene controllato l'overflow, per la verifica di supero di capacità, ad esempio, di variabili. Il controllo dello stack permette di settare, infine, spazio sufficiente per accogliere elementi che sarà necessario dislocare, ad esempio, nelle subroutine, nelle funzioni, o nei sottoprogrammi.

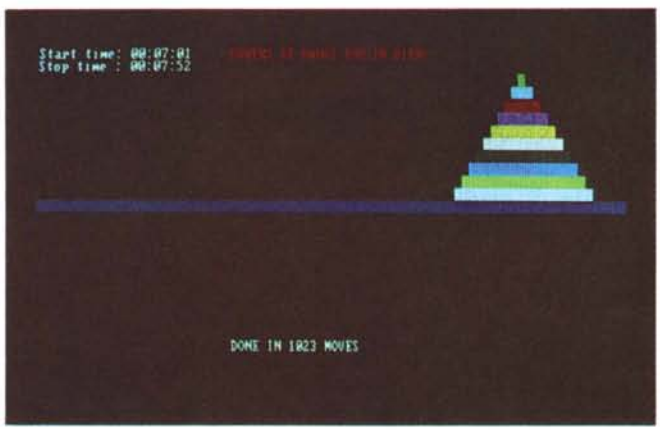
Di un'ultima opzione non abbiamo ancora parlato, quella riguardante i metastatement. Questa ha un proprio menu, e serve ad allocare memoria, rispettivamente, per i buffer di stack, musica, e comunicazioni.

Ancora più interessante appare il comando Setup, che permette di inizializzare alcune opzioni di linguaggio-sistema; tramite questa opzione è lecito definire colori di sfondo e di uso, i percorsi per le directory (anche in funzione delle operazioni di Include di file accessori), le attivazioni di finestra, il dimensionamento delle stesse, lo zoom su una finestra prescelta (utile in fase di editing, soprattutto). Il comando [DEBUG], ancora, offre due opzioni, Trace e Runtime error; con la prima si attiva il tracciamento globale del programma (l'opzione dipende, comunque, dai comandi TRON e TROFF presenti eventualmente nel listato; label, numeri di riga e nomi di procedure sono visualizzati in questa finestra ogni qualvolta vengono considerati dall'esecuzione); la funzione Runtime error viene invece utilizzata per trovare errori in file .EXE.

Dalla pagina 59 in poi (cap. 4) il manuale diviene un efficiente tutorial del linguaggio. Niente pare lasciato al caso, e l'utente appena esperto sarà forse seccato da certe precisazioni forse un po' pedanti. Ma non si può condannare il perfezionismo, se questo è precisione ed illustrazione esaustiva. Perciò il manuale parte dalle più semplici regole di redazione di un programma per passare all'e-

same dei numeri, delle variabili e delle costanti (definibili, allo stesso modo delle variabili, mediante una operazione di assegnazione), delle array, delle espressioni, degli operatori aritmetici, relazionali e logici, delle sobroutine, funzioni e procedure (queste ultime chiamate appunto con questo nome, e del tutto simili all'equivalente struttura del Pascal od alla function del «C»), con tanto di variabili locali globali, statiche, shared e non. Udite, udite, viene supportata la ricursione nella sua forma più potente (ne parliamo nel riquadro a fianco) ed una intera sezione viene dedicata alla manipolazione dei file, dove ne viene trattato in maniera estesa anche uno forse un po' esotico per i conoscitori del Basic: il fiel binario (dove tutto è sequenza numerica di byte, senza alcun riferimento a caratteri ASCII, stringhe, delimitatori, ecc). La sintassi di manipolazione dei file è abbastanza simile allo standard Microsoft (se ne poteva fare a meno?) e chi proviene dal miserello Basic non trovevà difficoltà ad adattarsi. La grafica si affida ad alcuni comandi specializati, abbastanza potenti anche se, alla fin fine non numerosi (a quando un Turbo Graphics dedicato al Basic), con la comodità di disporre (come d'altro canto accade anche in QB) di coordinate relative settabili anche in analogia ad un normale piano cartesiano (con l'origine in basso a sinistra). I comandi grafici sono in tutto una quindicina, ed alcuni si servono di un vero e proprio metalinguaggio (DRAW e CIRCLE sono praticamente identici a quelli che abbiamo visto in Quick Basic). Anche qui la parte più ampia del manuale è rappresentata dalla guida di riferimento dei 200 e passa comandi disponibili, e anche qui andiamo un po' a braccio: vediamo tra quelli meno comuni [$COM] che alloca spazio di memoria per il buffer della porta di comunicazione seriale [$DYNAMIC], una metaistruzione che dichiara che l'allocazione dei vettori dovrà essere dinamica, [$EVENT] che consente il controllo del verificarsi degli eventi definiti dagli statement ON..., [$INCLUDE], principe delle metaistruzioni, che consente di compilare un file di testo esterno, assieme al file corrente, [$SEGMENT], che permette di spezzettare in memoria il programma sorgente in segmenti scaricabili (si noti la presenza comune dell'indicatore [$]). Manca purtroppo l'opzione DEG per l'uso di gradi trecentosessantesimali, ed è possibile caricare e scaricare file ad immagine di memoria. La funzione CALL può chiamare una procedura od una routine in linguaggio Assembler, oltre ad interrupt di sistema. Alcuni comandi curiosi permettono di eseguire alcune operazioni per lo meno atipiche, come DECR, che permette di diminuire una variabile di un valore stabilito ed

Opzioni del compilatore; si notino tra l'altro il settaggio del coprocessore e la grandezza dei buffer di stack, musica, comunicazioni.



Ricursione in Basic: la classica torre di Hanoi, con 10 dischi, risolta, anche con animazione grafica, in 51 secondi.

Le funzioni principali dell'editor Turbo Basic

tasto/i	funzione
Esc	esce dall'editor
tasti cursore e PgUp, PgDn, Home, End	spostamento cursore per l'editing di linea e pagina
Del	cancellazione carattere
Ins/ctrlV	insert-o.write
Ctrl-Kb	marca l'inizio del blocco
Ctrl-Kk	marca la fine del blocco
Ctrl-Kh	cancella la marcatura del blocco
Ctrl-Kc	operazione di copy di un blocco
Ctrl-Ky	cancellazione di un blocco marcato
Ctrl-Kv	spostamento di un blocco
f1	chiamata all'Help
f2	salvataggio di un file
f3	inizio di un nuovo file
f4	n.u.
f5	zoom in linea
f6	attiva le finestre
f7	inizio del blocco
f8	fine del blocco

Quick Sort in Basic; vengono riordinati 5000 numeri in 10 secondi circa.

INCR, che esegue il contrario (ricordate qualcosa del genere nel «C»); ENVIRON modifica le informazioni nella tabella di ambiente, IF si presenta in maniera pluritest. LOG ammette diverse basi, MEMSET dichiara il limite massimo di memoria, MTIMER legge e reinizializza il microtimer interno, PMAP traduce coordinate fisiche in virtuali e viceversa, SCREEN ammette 7 tipologie di video, con riconoscimento di modo EGA e CGA. La difesa della strutturalità del linguaggio è affidata agli ormai soliti, anche nel Basic, SELECT CASE, DO-LOOP, WHILE, ecc.

Come al solito, la rimanente problematica è affidata ad una serie di appendici che prendono in considerazione la rappresentazione dei numeri, il controllo degli eventi, l'interfaccia col linguaggio Assembler (con chiamate di funzioni DOS e BIOS, e creazioni di file INLINE.COM), il confronto tra il TB ed il Basic interpretato, i messaggi di errore (finalmente con una diagnostica sintetica ed esauriente; i number error arrivano fino a 600, anche se, ovviamente, ne vengono utilizzati effettivamente molti di meno). Un'appendice richiama una serie di comandi DOS particolarmente utili o dedicati al linguaggio, e non mancano una serie di codici estesi dei tasti e dei codici di scansione della tastiera. Pare proprio che la Borland abbia redatto questo manuale con il principio di certi grandi magazzini che ragionano secondo il principio: «Una volta entrato il cliente non deve uscire per andare a cercare qualcosa da qualche altra parte!».

Infine un programma contenuto nel dischetto di accompagno al linguaggio permette di personalizzare il Turbo Basic, il tutto in maniera definitiva o provvisoria, tramite le seguenti operazioni:
— impostazione di un percorso per i file di aiuto ed impostazione
— personalizzazione della tastiera
— modifica dei modi di edit implicito
— impostazione di modo di schermo implicito.

Altro che semplice linguaggio, e quanto siamo lontani dal Basic o dall'Applesoft del buon vecchio Apple II; abbiamo parlato in questo articolo spesso di «ambiente di programmazione», e così può essere considerato, a tutti gli effetti TB; l'unico commento, per utilizzare un gergo «turbo» è dire «Che bomba!».

Volete un difetto? Uno l'abbiamo trovato e ve lo diciamo senza timore di essere smentiti. Edia Borland, nel tradurre il manuale, ha conservato la vecchia copertina ma ha usato carta semipatinata di buona qualità per comporre il volume. La durezza di questa carta, in una con la robusta rilegatura, ha portato a presentare un manuale destinato a sopportare impunemente maltrattamenti oltraggiosi; ma questa robustezza si è tradotta in una difficoltosa manipolabilità del volume, che, almeno quando è nuovo, non ne vuol sapere di stare aperto sul tavolo; ho provato a forzare, forse un po' troppo, ed il dorso ha ceduto. Troppa grazia, come si dice dalle mie parti!

Conclusioni (Quick e Turbo)

Quanto cammino è stato fatto dai Basic in pochi anni! Altro che linguaggio debole e rugginoso; i linguaggi di oggi sono attrezzi di precisione, ben adeguati alla raffinate macchine su cui girano! E di fronte a tool come Quick o Turbo impallidiscono fior di Pascal e C. Perciò, è proprio il momento di smetterla con la vecchia storia del linguaggio per mongoloidi del software.

Ma dal confronto chi esce, alla fin fine, vincitore? Turbo, con la velocità di complicazione entusiasmante, la sua ricorsività, il suo editor raffinatissimo, la maggiore vicinanza alla macchina, o Quick, con le sue indubbie facility, le innumerevoli opzioni, la sua, se vogliamo, consanguineità col DOS? Consentiteci di essere buoni (questo articolo è stato scritto qualche giorno prima di Natale) e di dichiarare un incontro pari: questo ha qualcosa, quello ne ha altre!

Ma cosa importa? al di fuori di tutto fa piacere, ad uno che ha sempre ammirato il Basic e non si è mai vergognato di professarsene cultore (tenga conto, il lettore, che chi scrive è uno che ha lavorato e lavora, per certe sue applicazioni, in Forth, Lisp, Pascal, e C.), vedere l'evoluzione che questo linguaggio ha subito, divenendo raffinato, completo, elegante, strutturato, pulito, e restando comunque facile da usare.

Sarà difficile, per chi abbia provato QB o TB, atteggiarsi a giudice sprezzante di un linguaggio... troppo poco «in» nei sacri ambienti... MC

Che cosa è la ricursione

TB ammette la ricursione, finora relegata nelle nascoste oscurità di certi linguaggi dedicati, come C e Forth. Quanto si è detto a proposito di essa! La definizione più spiritosa ricordo di averla letta qualche anno fa su una rivista inglese, dove, come esempio di discorso ricursivo, veniva proposta la frase «La pioggia è quella cosa che cade quando piove». Scherzi a parte, se è merito di Borland di aver cavato dalla nebbia questo potente processo, diamo a Cesare quello che è di Cesare e vediamo in che cosa questa consista, utilizzando un esempio contenuto nel manuale stesso.

Si abbia il seguente programma, anzi per essere precisi procedura (o funzione), che adotta una tecnica ricursiva:

```
DEF FNFattoriale#(n%)
 if n% > 1 AND n% <= 170 THEN
  FNFattoriale# = n% * FNFattoriale#(n%-1)
 ELSEIF
 n% = 0 OR n% = 1 THEN
  FNFattoriale# = 1
 ELSE
  FNFattoriale# = -1
 END IF
END DEF
```

A parte la brevità del codice, la struttura si presenta complessa, anche più di quanto sembra. Per renderci conto di come funziona una procedura ricorsiva, vediamo come essa fa a calcolare il fattoriale di 3.

Battiamo in modo immediato l'ordine:

```
PRINT FNFattoriale#(3)
```

Il valore 3 viene passato per valore alla funzione. Questa (si segua il listato) controlla se l'argomento è maggiore di 1 e minore od eguale a 170. Superato il controllo si tenta di eseguire l'operazione:

```
FNFattoriale# = 3 * FNFattoriale#(2)
```

A questo punto il programma si trova in difficoltà in quanto gli viene chiesto di considerare come valore definito [FNFattoriale#(2)] qualcosa di cui non conosce invece il valore. Allora si sospende momentaneamente l'elaborazione ed il programma si applica alla soluzione di FNFattoriale#(2).

Si ricomincia da capo, e la funzione FNFattoriale# tenta di passare il valore 2. Superato il controllo del > o <=170 ci si ritrova di nuovo di fronte alla chiamata ricorsiva della funzione, che ancora una volta non può essere risolta immediatamente. Anche questa DF viene messa in lista d'attesa e FNFattoriale# viene chiamato per la terza volta, stavolta con valore 1.

Il loop a questo punto si blocca, in quanto c'è la trappola del secondo THEN (n% = 1). FNFattoriale# può terminare la sua corsa all'indietro e richiamare l'ultima messa da parte [FNFattoriale#(2)] per poterla eseguire. Viene così ripercorsa, il senso contrario, la scala delle istruzioni accantonate, fino all'esaurimento dell'argomento della funzione principale. La struttura viene completata ed il risultato (6) stampato sullo schermo.

Le tecniche di ricursione, se correttamente usate, snelliscono molto il lavoro programmatorio, anche se talvolta non è facile decidere se è il caso di fare o no a loro ricorso.

C'è da dire, come appare evidente nell'analisi appena eseguita, che occorre dello spazio, da qualche parte (per essere precisi, nello stack) in cui accantonare le operazioni sospese. È opportuno, quindi, se si prevede di far uso di tecniche ricursive, di usare la metaistruzione $STACK per creare abbastanza spazio disponibile per lo stack stesso.