PROVA

Borland Turbo Pascal 5.0

di Sergio Polini

«L'unico vero limite del Turbo Pascal 4.0 è la mancanza di un debugger simbolico».

Così scrivevamo nella prova apparsa nel numero 76 di MC (luglio/agosto 1988); dopo solo pochi mesi ecco la replica della Borland: una nuova versione del compilatore, un debugger simbolico integrato nel familiare ambiente interattivo, la possibilità di scegliere tra una confezione normale ad una «professionale», comprendente un eccezionale debugger simbolico separato (che verrà esaminato il mese prossimo) ed un ottimo assembler (di cui vi riferiamo su questo stesso numero).

Se non fosse per la benevola politica di upgrade da sempre adottata dalla Borland, ci sarebbe quasi da arrabbiarsi: come si fa a star dietro a prodotti che si aggiornano ogni 6 mesi? In realtà l'aggiornamento di questa volta è stato molto più rapido che in passato per un ben preciso motivo. Da quando la Microsoft introdusse il CodeView nel luglio del 1986, è cambiato il concetto stesso di «compilatore professionale». Autorevoli testate americane (si pensi al Dr. Dobb's Journal*) avevano sempre contestato la validità delle prime versioni del Turbo Pascal per un uso professionale; la Borland aveva efficacemente replicato con il 4.0, ma non si poteva trascurare la mancanza di adeguati strumenti di debugging. In un mercato in continuo movimento, caratterizzato dalla sempre maggiore diffusione di altri linguaggi (in primo luogo il C, grazie anche al Turbo C, ma pure il Modula-2), non era prudente accontentarsi dei risultati raggiunti: era urgente superare un limite che, alla lunga, avrebbe potuto allontanare gli utenti e dal Pascal e dalla Borland.*

Questo dal «loro» punto di vista. Vedremo tuttavia che «loro», coerentemente con consolidate abitudini, non hanno trascurato il «nostro» punto di vista: non solo i due debugger, pur non facendo altro che colmare un ritardo, premiano adeguatamente la lunga attesa, ma nella versione 5.0 troviamo anche molte altre interessanti novità.

L'introduzione delle unit, un MAKE incorporato, uno *smart linker*, la compatibilità con la versione 3.0 assicurata da apposite unit e guidata da un programma UPGRADE, un potente insieme di routine grafiche, il superamento del limite dei 64K per il codice, nuovi tipi di dati, una più efficiente valutazione delle espressioni booleane, nuove direttive per la compilazione condizionale e per la scrittura di procedure associate ad un interrupt, la gestione degli «errori critici» del DOS, l'ottimizzazione del codice oggetto, la «pick list», le potenti funzioni di help. Quando si cerca di elencare le migliorie del Turbo Pascal 4.0 rispetto alla versione precedente si corre un serio pericolo di dimenticare qualcosa.

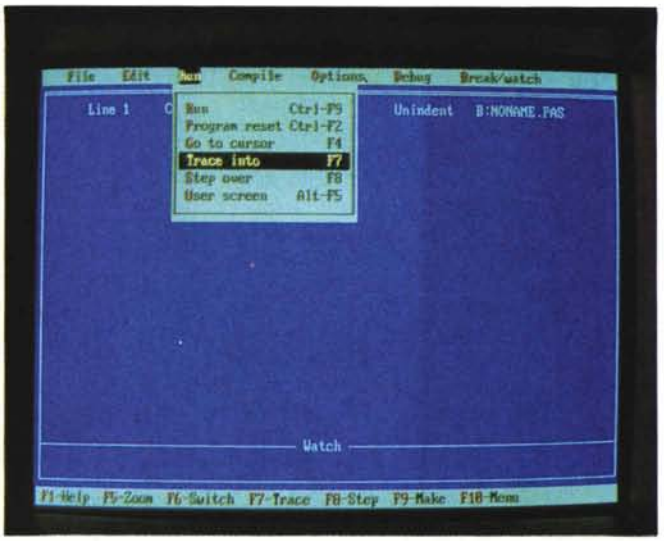
Tutte queste caratteristiche si ritrovano anche nella versione 5.0, ma ovviamente non ne riparliamo, essendo state già illustrate in occasione della prova della versione 4.0 (nel numero di luglio dello scorso anno).

Possiamo così concentrarci sulle novità.

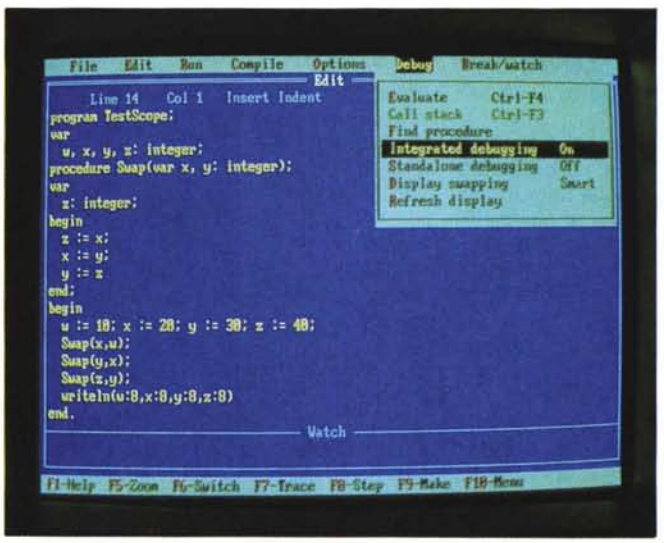
Debug, Break, Watch, Evaluate

Basta far partire il compilatore per accorgersi che qualcosa è cambiato: vi sono due nuove opzioni nel menu (*Debug* e *Break/Watch*), la finestra inferiore ha cambiato nome, da *Output* a *Watch*. Premendo Alt-R si scopre qualcosa di più: *Run*, prima semplice comando, è diventato un menu dal quale si può scegliere non solo di eseguire il programma in memoria, ma anche di seguirne passo passo l'esecuzione.

Perché i nuovi comandi abbiano effetto si devono impostare alcune direttive (anche mediante il menu *Options*), in modo da rendere disponibili al debugger integrato le informazioni relative ai simboli sia globali ($D+) che locali ($L+). Dopo di ciò si può cominciare l'esecuzione passo passo del programma con *Trace* (F7) o *Step* (F8); la differenza tra i due è che *Trace* ci porta fin dentro le funzioni o procedure che vengono man mano chiamate, mentre *Step* salta subito all'istruzione immediatamente successiva alla chiamata. Se invece si vuole passare subito ad una particolare sezione del codice, basta raggiungerla



Appena partito, il Turbo Pascal 5.0 mostra subito alcune differenze con il 4.0: due opzioni in più nel menu (Debug e Break/Watch), una finestra Watch invece di quella Output, diversi valori assegnati ai tasti funzione F6, F7 e F8. Premendo Alt-R si scopre poi che Run, prima semplice comando, è ora diventato un menu.



Le opzioni del menu Debug.

Turbo Pascal 5.0

Produttore:
Borland International
1800 Green Hills Road
P.O. Box 660001
Scotts Valley, CA 95066-0001
Distributore:
Edia Borland Srl
Via Cavalcanti, 5 - 20127 Milano
Telefono: 02/2610102
Prezzi *(IVA 9% esclusa):*

Turbo Pascal 5.0	*L.*	*299.000*
Turbo Pascal 5.0 Professional	*L.*	*547.000*
Upgrade dalla versione precedente		
— al Turbo Pascal 5.0	*L.*	*149.000*
— al Turbo Pascal 5.0 Professional	*L.*	*398.000*

con il cursore e dare il comando *Go to cursor* (F4). Se si vuole ricominciare da capo, si può dare in qualsiasi momento il comando *Program reset* (Ctrl-F2).

In ogni caso si ha la possibilità di esaminare il sorgente del programma man mano che questo viene eseguito, anche quando vengono chiamate procedure o funzioni contenute in altre unit (purché compilate con le direttive $D+ e $L+).

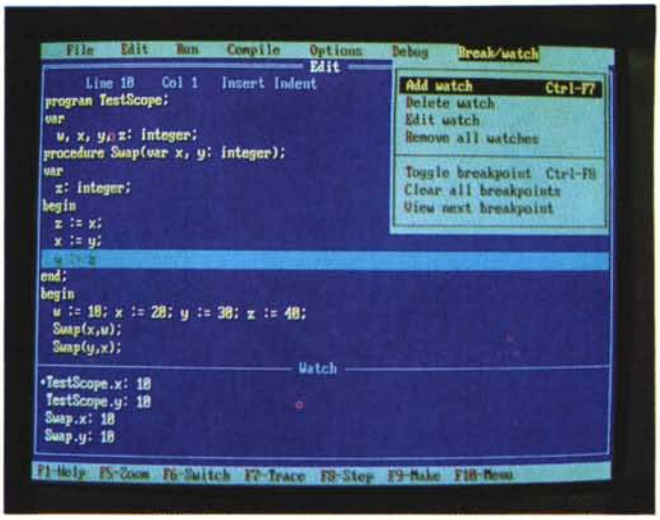
I programmi producono quasi sempre un output su video.

Si può scegliere (dal menu *Debug*) di ottenere questo output sulla stessa schermata del compilatore, ma è certamente preferibile optare per un'altra possibilità: ogni volta che c'è un output su video quella schermata scompare, viene riproposto lo schermo come era prima che si facesse partire il compilatore, come se al suo posto fosse stato invece eseguito il programma in memoria. Se questo chiede un input dall'utente si ha tutto il tempo che si vuole per provvedere, altrimenti si torna immediatamente al debugger (premendo Alt-F5 si può poi riesaminare in ogni momento l'output del programma eseguito).

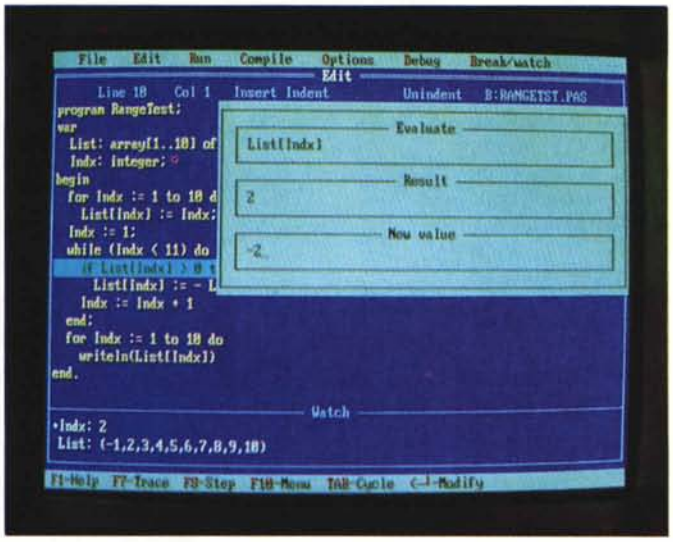
A volte può essere utile eseguire un programma «a scatti»: piuttosto che procedere passo passo anche in quelle parti di cui ci sentiamo sicuri, possiamo impostare dei *breakpoint* all'inizio delle sezioni sospette. Basta portarsi su queste con il cursore e premere Ctrl-F8; dando poi *Run* il programma verrà eseguito normalmente fino al primo *breakpoint* e si arresterà, consentendoci di esaminare la situazione ed eventualmente ripartire con *Trace* o *Step*, oppure con un nuovo *Run* fino al *breakpoint* successivo o fino all'**end** finale.

Sia durante un *Trace* che quando ci si ferma per un *breakpoint*, è possibile ricostruire il cammino percorso mediante un'opzione del menu *Debug*: il *call stack* (richiamato con Ctrl-F3) mostra le successive chiamate di funzione o procedura che vi hanno portato dove vi trovate.

La finestra *Watch* consente di tenere sotto controllo le variabili: premendo Ctrl-F7 si può aggiungere alla finestra la variabile su cui è posizionato il cursore, o un'altra qualsiasi scrivendone il nome. Ogni volta che il valore di una variabile «sotto osservazione» viene modificato, la finestra viene aggiornata. Vengono trattati efficacemente anche i tipi di dati strutturati: si possono vedere gli elementi di un array (anche multidimensionale), i campi di un record, gli elementi di un insieme. Se un array è troppo grande può essere esaminato «a pezzi» («List[6],3» mostra tre elementi dell'array List a partire dal sesto), ma è anche possibile scorrerlo tutto con il cursore, grazie allo *scrolling* orizzontale dalla finestra.



Le opzioni del menu Break/Watch e la finestra Watch. Si noti come è possibile evitare confusioni tra variabili con lo stesso nome, qualificandole con l'indicazione del modulo o della procedura in cui sono definite.



L'opzione Evaluate del menu Debug consente di cambiare durante l'esecuzione di un programma il valore delle sue variabili. Vediamo nella figura che sia Indx che List [Indx] valgono 2, mentre il programma è giunto ad un test in cui si controlla se List [Indx] è maggiore di zero. Prima di eseguire il test potremmo cambiare il valore di List [Indx] da 2 a −2.

La finestra *Watch* consente di tenere sotto controllo le variabili: premendo Ctrl-F7 si può aggiungere alla finestra la variabile su cui è posizionato il cursore, o un'altra qualsiasi scrivendone il nome. Ogni volta che il valore di una variabile «sotto osservazione» viene modificato, la finestra viene aggiornata. Vengono trattati efficacemente anche i tipi di dati strutturati: si possono vedere gli elementi di un array (anche multidimensionale), i campi di un record, gli elementi di un insieme. Se un array è troppo grande può essere esaminato «a pezzi» («List[6],3» mostra tre elementi dell'array List a partire dal sesto), ma è anche possibile scorrerlo tutto con il cursore, grazie allo *scrolling* orizzontale dalla finestra.

Apposite convenzioni di formato e una sorta di *typecasting* consentono infine di esaminare nel modo che si ritiene più opportuno numeri interi e reali, stringhe e caratteri, puntatori e aree di memoria.

Può capitare di riconoscere un bug proprio grazie al valore assunto da una variabile nella finestra *Watch*, ma si può desiderare anche di vedere subito cosa succederebbe se quella variabile assumesse un particolare valore. Si dispone quindi di una opzione *Evaluate* nel menu *Debug* (attivabile anche con Ctrl-F4); si apre una finestra divisa in tre zone: nella prima si scrive il nome della variabile desiderata, nella seconda ne compare il valore corrente, nella terza si può modificare questo valore.

Cosa manca? Non si possono esaminare né il codice prodotto dal compilatore né il valore corrente dei registri nel microprocessore. Si tratta di una limitazione in qualche modo necessaria, in quanto un debugger più potente non potrebbe risiedere nell'ambiente integrato se non consumando memoria preziosa. Non è comunque una limitazione drastica, in quanto da una parte gli strumenti ora descritti risultano già di validissimo aiuto, dall'altra è anche disponibile un super-debugger separato (il Turbo Debugger, che verrà esaminato, ricordiamo, il mese prossimo).

Unit, 80x87, Overlay, un nuovo tipo

Dicevamo all'inizio che la Borland non si è limitata ai debugger integrato e separato, ma ha apportato altri interessanti ritocchi al suo Pascal. Prendiamo ad esempio le unit. Pur essendo

per tanti aspetti tra loro indipendenti, le unit del 4.0 dovevano rispettare una certa gerarchia: se la unit A usava la unit B e questa la unit C, la unit C non poteva usare la A. Questa limitazione viene superata dal Turbo Pascal 5.0, grazie ad una clausola **uses** posta nella *implementation section*. Per le clausole **uses** nella *interface section* valgono ancora le vecchie regole, ma quelle nella *implementation section* consentono di scrivere unit che si «usano» l'una l'altra.

Ricordiamo che le unit avevano permesso di superare il limite dei 64K per il codice, tanto che la Borland aveva creduto di poter eliminare dal 4.0 il meccanismo di overlay. Gli utenti (come si è visto anche su MC-Link) non ne sono stati entusiasti. Ecco quindi che ora si dispone di una unit *Overlay*, ed è possibile compilare con una opzione $0+ le proprie unit che si voglia caricare in memoria solo quando necessarie (mentre nel Turbo Pascal 3.0 il meccanismo di overlay operava su singole procedure, ora l'unità minima è rappresentata dalla unit). Il compilatore produce, oltre al file EXE, un file con estensione OVR che comprende tutte le unit da gestire in overlay; una procedura *OvrInitEMS* cerca di caricare il file OVR nella memoria EMS se presente e disponibile, altrimenti le unit verranno lette da disco.

Se questo costituisce in fondo un (apprezzato) ritorno all'antico, il trattamento dei numeri in virgola mobile presenta invece felici caratteri di novità. Il Turbo Pascal ha sempre usato due formati non compatibili tra loro: o numeri di 6 byte gestiti da apposite librerie, o numeri di 8 byte gestiti da un coprocessore numerico. Era sempre mancata una libreria di emulazione software del coprocessore. Ora abbiamo il meglio dei due mondi: con la direttiva $N- si rimane alla tradizione il tipo *real* di 6 byte), con $N+, affiancata da $E+, si dispone dell'intera gamma dei nuovi tipi introdotti dal 4.0 (*single, double, extended* e *comp*). Soprattutto si possono realizzare programmi in grado di utilizzare il coprocessore se presente, di emularlo se assente. È vero che in quest'ultimo caso si perde un po' di velocità rispetto al 4.0, ma è anche vero che a volte risulta molto utile poter portare i programmi, e i relativi file di dati, da una macchina con 80x87 a una senza e viceversa. A proposito di tipi, un'altra cosa che era sempre mancata sono i parametri procedurali. Il Pascal standard consente di passare come parametri anche i nomi di funzioni o procedure, ma nel Turbo Pascal ciò non era possibile (nel 4.0 si poteva solo aggirare l'ostacolo mediante una *inline directive*). Ora abbiamo finalmente non solo questa possibilità, ma anche quella di definire variabili di tipo *procedure*, sia come variabili semplici che come elementi di variabili strutturate (array di funzioni, procedure come campi di un record, ecc.).

Spiccioli

Accanto alle novità maggiori, troviamo anche un folto gruppetto di innovazioni meno vistose, ma non per questo trascurabili.

Viene ad esempio estesa la sintassi delle dichiarazioni di costanti, nel senso che diventa possibile dichiarare una costante assegnandole come valore il risultato di una *constant expression* (cioè, in pratica, di una espressione che possa essere calcolata durante la compilazione, anche usando costanti dichiarate in precedenza o funzioni quali *Abs, Length, Ord, Succ*, e simili).

Le routine grafiche riconoscono ora anche la scheda grafica IBM-8514, e comprendono procedure per installare un driver eventualmente fornito dal produttore di altre schede, oppure file con fonti di caratteri non comprese nel sistema BGI (Borland Graphics Interface). Alcune nuove funzioni (quali *GetMaxMode* e *GetPaletteSize*) agevolano poi la scrittura di programmi grafici in grado di girare su hardware diversi.

Viene infine proposto un insieme di funzioni e procedure per l'accesso all'environment. Tra queste risultano particolarmente utili *GetEnv*, che ritorna il valore di una variabile dell'environment, e *FSearch*, che cerca un file in una lista di directory e, se lo trova, ritorna una stringa in cui sono concatenate la directory in cui il file è stato trovato e il nome di questo. Poiché quella lista deve avere lo stesso formato della stringa assegnata alla variabile PATH dell'environment, *FSearch* può essere usata per cercare un file mediante una lista fornita da *GetEnv*.

Conclusioni

Che il Turbo Pascal 4.0 ci era piaciuto molto lo avevamo detto a luglio. Che dire ora della nuova versione? Avevamo individuato dei limiti nella mancanza di un debugger simbolico e di una libreria di emulazione del coprocessore numerico, e la Borland ha subito provveduto. Avevamo apprezzato le *inline directive* soprattutto perché consentivano di emulare i parametri procedurali del Pascal standard, e ora abbiamo non solo parametri, ma anche variabili procedurali.

Dopo aver usato per qualche mese il compilatore, ci sentivamo un po' legati dalla impossibilità di riferimenti circolari tra le unit, e ora abbiamo anche questi. Altri utenti hanno lamentato la scomparsa dell'overlay, e sono stati accontentati.

Il tutto in una nuova versione che non ha mancato di arricchire sia la sintassi del linguaggio che una già ricchissima libreria grafica.

Ad un prezzo che, se si esclude il debugger separato, è solo marginalmente aumentato rispetto a quello precedente. Non è cambiato, peraltro, il costo dell'upgrade, e sono già disponibili i manuali in italiano.

Un giudizio sintetico sul Turbo Pascal 5.0? Semplice: è Borland. MC

```
{$F+}
Program ProcVar;
uses Crt;
type
  GotoProc = procedure(x, y: integer);
var
  Go: array[1..2] of GotoProc;
procedure GotoRC(Riga, Col: integer);
begin
  gotoxy(Col, Riga)
end;
procedure GotoCR(Col, Riga: integer);
begin
  gotoxy(Col, Riga)
end;
begin
  clrscr;
  Go[1] := GotoRC;
  Go[2] := GotoCR;
  Go[1]( 1,40); write('<-- Riga 1 / Col 40');
  Go[2](40,10); write('<-- Col 40 / Riga 10');
  Go[1](24,1);
  write('Premi <Enter> ...');
  readln
end.
```

Oltre ai parametri procedurali (presenti nel Pascal standard ma implementati dalla Borland solo ora), il Turbo Pascal 5.0 propone anche variabili di tipo procedura, che rendono possibile, tra l'altro, costruire array di funzioni o procedure.