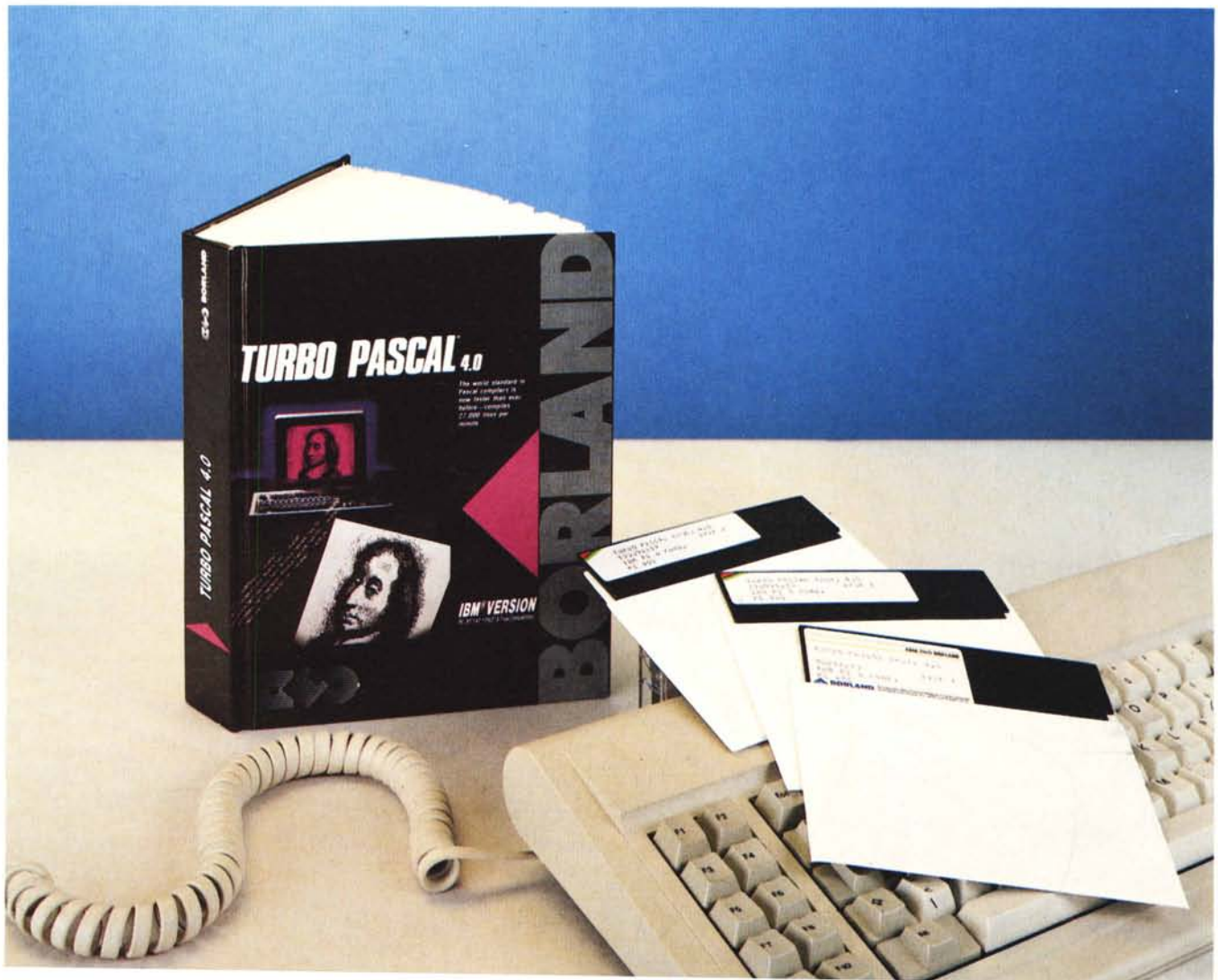
Turbo Pascal 4.0

di Sergio Polini

1985: l'incredibile successo del Turbo Pascal, del SideKick e di SuperKey ha reso ormai famosa la Borland; oltre alle riviste specializzate, anche Time e The Wall Street Journal intervistano Philippe Kahn, tutti desiderano conoscere il presidente e fondatore della Borland, tutti si interrogano sui segreti del suo successo, tutti gli chiedono quali sono i suoi programmi, in particolare vogliono sapere se verranno prodotti compilatori per altri linguaggi. Kahn confessa di avere una particolare predilezione per il Modula-2, e annuncia che al Turbo Pascal seguiranno per primo un Modula-2, poi C, Basic e Ada. Pronuncia anche una frase rimasta famosa: il C è un linguaggio «write-only» (a sola scrittura), in quanto può essere estremamente arduo capire cosa fa un programma scritto in C leggendone il sorgente; il Pascal è invece un linguaggio «read and write», in quanto consente di scrivere codice molto più chiaro e leggibile. Precisa che la Borland farà un compilatore C se il mercato proprio lo vorrà, ma solo dopo il Modula-2, in quanto questo linguaggio, secondo Kahn, offre insieme la funzionalità del C e la leggibilità del Pascal.

La scelta di campo era chiara e almeno in parte scontata: rispetto al Pascal, linguaggio nato per l'insegnamento della «buona programmazione» e fino a poco tempo fa dominatore incontrastato nelle università di mezzo mondo, il C, disegnato da Dennis Ritchie a proprio uso e consumo, tollerante verso il programmatore ardito e a tratti un po' indisciplinato, sembrava più un espediente che un Linguaggio degno del nome. E il Modula-2, dallo stesso Wirth ideato, si presentava come il naturale successore del Pascal: gli stessi principi di buona programmazione portati dal mondo della didattica a quello dell'effettivo sviluppo di progetti complessi, sistemi operativi compresi.



Turbo Pascal 4.0

Produttore:
Borland International
4585 Scotts Valley Drive
Scotts Valley, CA 95066 USA

Distributore per l'Italia:
Edia Borland Srl.
Viale Cirene, 11
20135 Milano

Prezzi

Turbo Pascal 4.0	*L. 249.000 + IVA*
Upgrade dal Turbo Pascal 3.0	*L. 149.000 + IVA*

La Borland si mise dunque di buona lena a lavorare ad un Turbo Modula-2, sia per CP/M che per MS-DOS. Le due versioni vennero effettivamente portate a termine: quella per CP/M più di un anno fa, quella per MS-DOS quest'anno. Ma non sono targate Borland: vengono infatti commercializzate la prima dalle società americane Eschalon e Micromint, la seconda dalla inglese Jensen and Partners International. Sono intanto arrivati un inaspettato Turbo Prolog e poi il Turbo Basic e il Turbo C. Cosa è successo?

Restiamo al 1985. Vengono intervistati da Computer Language Donald Knuth e Nicklaus Wirth: era ben noto che i due grandi, ambedue vincitori del prestigioso Turing Award (rispettivamente nel 1974 e nel 1984), avevano idee diverse sui linguaggi di programmazione, e si chiede a Knuth cosa ne pensa del Modula-2. Apriamo una parentesi. Quando abbiamo provato il Turbo Pascal 3.0 (MC n. 62), abbiamo sottolineato come la Borland fosse riuscita a proporre una implementazione del linguaggio che ne superava molti limiti. Alcune delle caratteristiche della versione Turbo (pensiamo in particolare ai parametri variabile senza tipo, alle dichiarazioni «absolute», alle operazioni sui puntatori, ecc.) erano proprio tese a rendere il linguaggio più flessibile, e quindi più usabile, di quanto non volessero «buoni principi» rigidamente intesi. Altre estensioni, come le costanti tipizzate o gli operatori di shift o l'allocazione di memoria mediante GetMem, sembravano ispirate più dal C che dal Pascal di Wirth. Il Modula-2 invece, pur superando alcuni dei limiti del Pascal, introduce nuove rigidità; basti pensare che non solo si scoraggia il Goto (come in ogni linguaggio che si rispetti), ma addirittura il Goto non esiste per niente! Bene: Knuth dice senza riserve che questo gli sembra sciocco (silly), ed esprime un giudizio severo anche sulla impossibilità di effettuare comparazioni tra puntatori diversi da un test di eguaglianza.

Il mercato ha dato ragione a Knuth: il C domina la scena della programmazione professionale su mini e micro, dai sistemi operativi all'intelligenza artificiale, il Turbo Pascal 3.0, nonostante il limite dei 64K sia per il codice che per i dati, è tuttora il linguaggio più diffuso sui PC (secondo un sondaggio condotto dalla americana 8C World); i sostenitori del Modula-2 sono una sparuta minoranza.

Kahn ne ha preso atto e così, dalle ceneri del Turbo Modula-2, è nato il Turbo Pascal 4.0.

Il Pascal originario e la sua versione Turbo (fino alla 3.0) non consentono la compilazione separata: il programma va compilato tutto insieme. La Borland aveva un po' rimediato consentendo di includere nel testo del programma principale altri file (con la direttiva $I), altri produttori avevano invece realizzato versioni del linguaggio che consentivano la compilazione di diverse «unit» che venivano poi «linkate» per produrre il file eseguibile finale.

Più radicale la proposta di Wirth. Il suo Modula-2 consente di scomporre un programma in «moduli», ognuno compilabile separatamente. Ogni modulo consta di due file: una *definition part* e una *implementation part*; la prima contiene le dichiarazioni di costanti, tipi, variabili e procedure del modulo acces-

Figura <43 line EGA display>. L'editor del Turbo Pascal può sfruttare un monitor EGA o VGA in modo da consentire di lavorare su uno schermo di 43 o 50 righe. Vediamo nella foto il menu Options con i suoi sottomenu fino a quello che propone la scelta tra 25, 43 e 50 righe.

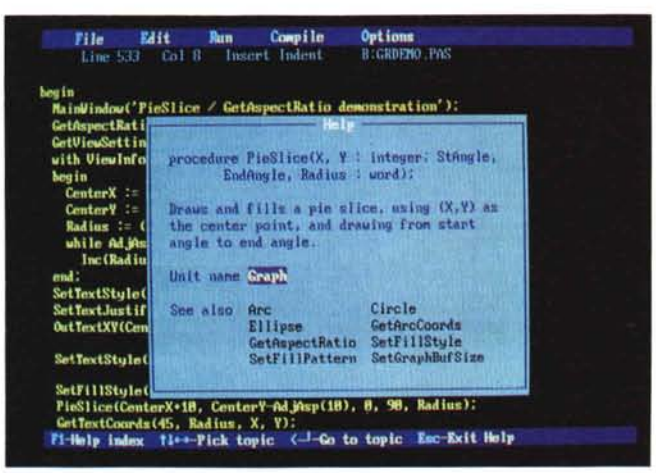


Figura <Help>. Posizionando il cursore sul nome di una unit, funzione o procedura, e premendo Ctrl-F1 si apre una finestra contenente spiegazioni. Di qui si può poi accedere ad altre finestre di Help relative alle unit usate e alle funzioni o precedure correlate.

sibili da altri moduli, la seconda contiene il codice delle procedure, una eventuale routine di inizializzazione delle variabili della *definition part*, eventuali dati e procedure «privati», in quanto usati da quelli «pubblici» ma non accessibili dall'esterno. Questa scomposizione comporta diversi vantaggi, soprattutto nel caso di programmi realizzati da un team di programmatori: ogni modulo (ogni programmatore incaricato dello sviluppo di un particolare modulo) ha solo bisogno di conoscere la *definition part* degli altri moduli che deve usare, ma può disinteressarsi completamente di come gli oggetti in essi definiti vengono implementati. Al tempo stesso, un primo test del programma completo può essere condotto dopo una versione provvisoria (e magari poco efficiente) delle *implementation parts*; eventuali successive modifiche di queste, fino a che non incidono sulle corrispondenti *definition parts*, possono essere approvate senza alcuna dipendenza da quanto avviene in altri moduli e senza imporne adattamenti e nuove compilazioni. Sarà la fase finale di «link» ad assicurare che il programma completo contenga le versioni aggiornate di tutti i moduli.

Gli oggetti «esportati» da un modulo possono essere «importati» in due modi: o si importano solo determinati oggetti da un modulo (FROM nome-modulo IMPORT elenco-oggetti), o si importa tutto il modulo (IMPORT nome-modulo). In quest'ultimo caso però i nomi dei suoi oggetti vanno poi preceduti dal nome del modulo, come i campi di un record dal nome di questo (ad esempio, la procedura Push del modulo Stack va chiamata con Stack.Push).

Si crea così quella che Wirth definisce «gerarchia di astrazioni». Si parla di astrazioni perché si può e si deve prescindere dai dettagli della implementazione degli oggetti che vengono importati da un modulo, di gerarchia perché chiaramente ogni modulo che ne usi altri dipende da questi, ad esempio nel senso che bisogna fare attenzione all'ordine con cui i diversi file vengono compilati: ogni DEFINITION MODULE deve essere compilato prima del corrispondente IMPLEMENTATION MODULE e prima di ogni modulo che lo importi, ogni IMPLEMENTATION MODULE che sia stato modificato dopo l'ultimo «link» deve essere ricompilato prima di procedere ad un nuovo «link».

3 + Unit = 4

Wirth indicava nei moduli il principale progresso del Modula-2 rispetto al Pascal. Analogamente, possiamo dire che le unit (simili, dal punto di vista sintattico, a quelle del Pascal UCSD) rappresentano la principale innovazione della versione 4.0 del Turbo Pascal rispetto alle precedenti. Si può anche dire che, salvo la maggior potenza che i moduli offrono in situazioni di eccezionale complessità, le unit presentano qualche vantaggio rispetto ai loro cugini del Modula-2, in particolare una maggiore snellezza d'uso.

Non abbiamo quattro tipi diversi di file (DEF, MOD, SYM e LNK nel Modula-2/86 2.0 della Logitech; il programma finale può poi avere estensione LOD o EXE), ma normali file PAS, dei quali vengono compilati con estensione EXE quelli che cominciano con **program**, con estensione TPU (Turbo Pascal Unit) quelli che cominciano con **unit**. In que-

```
(* file STACK.DEF, compilato in STACK.SYM *)

DEFINITION MODULE Stack;
  EXPORT QUALIFIED Push,Pop;          (* procedure usabili da altri moduli *)
  PROCEDURE Push(x: REAL);
  PROCEDURE Pop(): REAL;
END Stack.

(* file STACK.MOD, compilato in STACK.LNK *)

IMPLEMENTATION MODULE Stack;
  CONST Max = 10;                     (* numero max di valori nello stack *)
  VAR   Stk: ARRAY[0..Max-1] OF REAL; (* stack di reali e stack pointer,  *)
        Sp:  INTEGER;                 (*  non accessibili da altri moduli *)
  PROCEDURE Push(x: REAL);
  BEGIN
    IF Sp < Max THEN
      Stk[Sp] := x;
      INC(Sp)                         (* equivale a: Sp := Sp + 1 *)
    END
  END Push;
  PROCEDURE Pop(): REAL;              (* in Modula-2 si chiamano PROCEDURE *)
  BEGIN                               (*  anche le funzioni *)
    IF Sp > 0 THEN
      DEC(Sp);                        (* equivale a: Sp := Sp - 1 *)
      RETURN(Stk[Sp])
    END
  END Pop;
BEGIN (* Inizializzazione *)
  Sp := 0
END Stack.

(* file SOMMA.MOD, compilato in SOMMA.LNK *)

MODULE Somma;
  FROM InOut     IMPORT WriteCard, WriteString, WriteLn;
  FROM RealInOut IMPORT ReadReal, WriteReal;
  IMPORT Stack;
  VAR x: REAL;
      i: CARDINAL;
BEGIN
  FOR i := 1 TO 10 DO
    WriteCard(i,2); WriteString(" numero: "); ReadReal(x); WriteLn;
    Stack.Push(x)
  END;
  x := 0.0;
  FOR i := 1 TO 10 DO
    x := x + Stack.Pop()
  END;
  WriteString("Il totale e': "); WriteReal(x,15); WriteLn
END Somma.
```

```
(* file STACK.PAS, compilato in STACK.TPU *)

unit Stack;

interface
procedure Push(x: real);
function  Pop: real;

implementation
const Max = 10;
var   Stk: array[0..9] of real;
      Sp : integer;
procedure Push;
  begin
    if Sp < Max then begin
      Stk[Sp] := x;
      inc(Sp)
    end
  end;
function Pop;
  begin
    if Sp > 0 then begin
      dec(Sp);
      Pop := Stk[Sp]
    end
  end;
begin
  Sp := 0
end.
(* file SOMMA.PAS, compilato in SOMMA.EXE *)
program Somma;
uses Stack;
var  x: real;
     i: word;
begin
  for i := 1 to 10 do begin
    Write(i:2,' numero: '); Readln(x);
    Push(x)
  end;
  x := 0.0;
  for i := 1 to 10 do begin
    x := x + Pop
  end;
  Writeln('Il totale e': ',x:15)
end.
```

Figura 2 - Lo stesso programma della figura 1 tradotto in Turbo Pascal 4.0 SOMMA.EXE è lungo 4576 byte.

◀ *Figura 1 - Un esempio di programma scritto in Modula-2 (abbiamo usato il Modula-2/86 2.0 della Logitech) con un file principale e un modulo Stack. Il programma accumula in uno stack 10 interi, poi ne mostra la somma.*

```
procedure IntHandler(Flags,CS,IP,AX,BX,CX,DX,SI,DI,DS,ES,BP:word)
interrupt
begin
...
end;
```

*Figura 3 - Ora è più facile scrivere una procedura da associare ad un interrupt. La direttiva **interrupt** fa si che il compilatore generi automaticamente il codice che salva e poi ripristina i registri dal microprocessore e inizializza il registro DS. «Passando» i registri come fossero parametri, vi si può poi accedere senza bisogno di un «inline statement».*

sti abbiamo una parte di definizione dei dati e delle routine esportati, introdotta dalla **keyword interface**, una parte con cui vengono messi a punto i dettagli degli oggetti «pubblici» (eventualmente con ricorso a dati e/o routine «privati»), introdotta dalla keyword **implementation**, infine una parte contenente il codice incaricato della eventuale inizializzazione delle variabili della unit, introdotta da un normale **begin**.

È poi lo stesso compilatore che si incarica di gestire le dipendenze. Possiamo, se vogliamo, compilare un solo file indipendentemente dagli altri (ad esempio se è unico, o se vogliamo verificare che non vi siano errori di sintassi), possiamo compilare indistintamente tutti i file che compongono il programma in una volta sola con l'opzione BUILD (utile se vogliamo cambiare il valore di qualche direttiva di compilazione), possiamo compilare «il meno possibile» con l'opzione MAKE: vengono ricompilati solo i file sorgente che sono stati modificati dopo l'ultima compilazione, ma vengono ricompilati anche i file che «includono» altri file o chiamano procedure **external** (che vanno assemblate in file OBJ) che risultano modificati, oppure «usano» una unit di cui si è cambiata la «interface», viene ricompilata unicamente la unit se era stata cambiata la sola «implementation». Dopo la compilazione segue automaticamente il «link». Ancora. Nel Modula-2 bisogna importare praticamente tutto, in particolare le procedure di I/O, che non fanno parte della definizione del linguaggio ma sono fornite da *Standard Utility Modules* (illustrati nel *Programming in Modula-2* di Wirth). Il programmatore deve incaricarsi di indicare quali procedure di questi moduli intende importare. Nel Turbo Pascal 4.0 viene invece sempre usata una unit *System* (che non è quindi necessario menzionare espressamente), e in ogni caso per nessuna unit bisogna indicare quali oggetti se ne vogliano «usare»: all'inizio di ogni programma o unit basta scrivere la clausola **uses** seguita dai nomi delle unit che interessano, e sarà poi lo *smart linker* del compilatore ad aggiungere al file EXE del programma finito solo le routine effettivamente usate, scartando le altre. Il risultato è un file EXE di dimensioni ridotte all'essenziale. Per quanto poco possano valere confronti tra diversi compilatori di diversi linguaggi, in figura 2 è proposto un programma in Turbo Pascal 4.0 in tutto e per tutto equivalente a quello scritto con il Modula-2/86 2.0 della Logitech, illustrato nella figura 1; si può notare, oltre alla maggiore concisione del sorgente, anche la vistosa differenza tra le dimensioni dei file eseguibili: 4576 byte contro 34496 (una traduzione dello stesso programma in Microsoft C 4.0, ha portato ad un file EXE lungo 22284 byte). Se ne può in ogni caso concludere che lo *smart linker* fa un buon lavoro.

La unit *System* non è l'unica già pronta all'uso. In essa sono implementate le procedure per la gestione dei file e delle directory e dei device (CON, LPT1, LPT2, LPT3, COM1 e COM2), nonché un folto gruppo di procedure standard per la manipolazione di numeri e stringhe, per l'allocazione dinamica della memoria, ecc. Si tratta per lo più di procedure già ben familiari a chi abbia fin qui usato le versioni precedenti del Turbo Pascal, ma che presentano a volte qualche differenza, dovuta ora ad una esigenza di maggiore conformità allo standard ANSI (come per *Read* e *Readln*), ora al superamento di artifici non più necessari; si dispone infatti di un tipo *longint* per interi a 32 bit che permette, ad esempio, di sostituire le precedenti funzioni *FilePos, FileSize* (che ritornavano un intero, quindi con un valore massimo di 32767), *LongFilePos* e *LongFileSize* (che ritornavano un reale) con le nuove *FilePos* e *FileSize* con risultato di tipo *longint* (fino a 2.147.483.647). Sono comunque anche presenti le unit *Turbo3* e *Graph3* che consentono di utilizzare, in questo caso ed in altri analoghi, le stesse procedure del Turbo Pascal 3.0, in modo da rendere molto ampia la compatibilità della nuova versione con la precedente (un programma UPGRADE.EXE agevola la conversione dei programmi, apportando automaticamente le modifiche necessarie o segnalando quei punti in cui si richiede un intervento manuale).

Oltre a queste vi sono poi le unit *Printer, Dos, Crt* e *Graph*. *Printer* si limita a dichiarare l'abituale file *Lst*, usato per l'invio di dati alla stampante. La unit *Dos* offre tra l'altro procedure per la scrittura di un interrupt handler, per leggere o modificare la data e l'ora di sistema o di un file, per controllare lo spazio residuo su disco o per cercare un file nella directory, per eseguire comandi di DOS o altri programmi da un programma Turbo Pascal. La unit *Crt* consente di sostituire alle normali operazioni di output su video (quelle presenti in qualsiasi sistema MS-DOS) in altre in grado di sfruttare a fondo il BIOS e l'hardware di un PC IBM o compatibile: e quindi finestre e colori come già con il Turbo Pascal 3.0, ma anche scrittura diretta sulla memoria video con controllo automatico della presenza di una scheda CGA (che imporrebbe altrimenti un'accurata codifica in Assembler per evitare un fastidioso sfarfallio dell'immagine).

La unit *Graph* è incredibile. La procedura *DetectGraph* riconosce il tipo di

ciclo	frequenza	TP 3.0	TP 4.0	+%
repeat	20000	390	501	28,5
while	20000	324	492	51,8
for	20000	498	581	16,7
aritmetica su interi (4 operazioni)	10000	196	216	10,2
aritmetica su reali (senza 80x87)	5000	10	13	30,0
funzioni trascendenti (sin,ln,exp,sqrt)	500	6	8	33,3
assegnazione di elementi di array	20000	122	156	27,9
assegnazione di elementi di matrici	10000	225	234	4,0
chiamata di procedura senza parametri	20000	80	111	38,7
chiamata di procedura con parametri	20000	66	90	36,4
assegnazione di interi array	500	71	71	-
attraversamento lista di 100 elementi	500	47	102	117,0
lettura da file	5000	15	15	-

Figura 4 - I risultati di un benchmark analogo a quello messo a punto da Wirth per i compilatori Modula-2. Vengono definiti diversi cicli, con contatori variabili da 500 a 20000. I numeri riportati per le due versioni del turbo Pascal rappresentano il numero di esecuzioni di ogni ciclo in 30 secondi; un numero più alto indica quindi maggiore velocità. L'ultima colonna mostra l'incremento percentuale della velocità del codice prodotto dal Turbo Pascal 4.0 rispetto a quello prodotto dal 3.0.

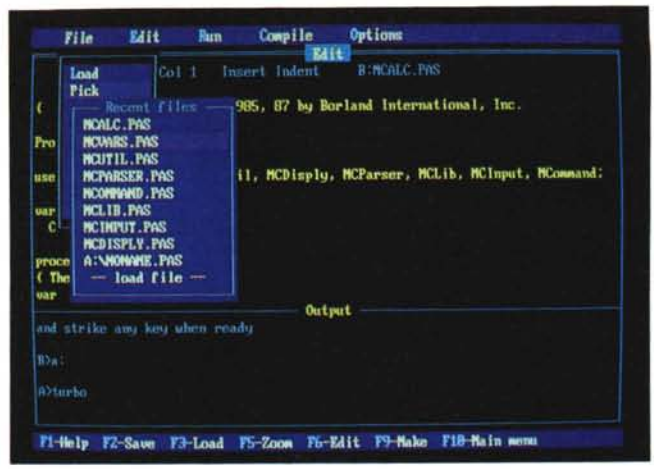


Figura <Recent files>. L'editor mantiene una lista dei file su cui si sta lavorando. Per tornare ad un file già salvato su disco si può premere Alt-F3: compare su video quella lista, con il cursore sull'ultimo file salvato; se si vuole caricarlo in memoria basta premere Enter, altrimenti si può scegliere un altro file spostando il cursore, o anche caricare un nuovo file posizionandosi su «– –load file – –». La lista può essere salvata su disco in un file con estensione PCK.

video (CGA, MCGA, EGA, VGA, Hercules, Olivetti, PC 3270) e *InitGraph* setta una serie di variabili interne, accessibili mediante apposite funzioni, con valori quali la risoluzione orizzontale e verticale e il numero di colori e di pagine grafiche disponibili. Usando funzioni come *GetMaxColor* o *GetMaxX* è possibile scrivere programmi usabili su macchine diverse: si tratta di quella che viene chiamata BGI (Borland Graphics Interface), che sarà presto disponibile anche con gli altri compilatori Turbo (già ora con il C 1.5). Sono presenti le «solite» procedure per tracciare linee, poligoni, archi, cerchi, ellissi, ma anche barre tridimensionali e diagrammi a torta; si possono definire come nel Graphix Toolbox i modi di tracciamento delle linee e di riempimento di aree, nonché finestre grafiche con «clipping» o senza; si dispone di cinque diversi font di caratteri. In pratica, se non metà almeno un buon terzo del turbo Graphix Toolbox è diventato parte integrante del compilatore, con in più la possibilità di scrivere programmi indipendenti dalla configurazione hardware.

Ma non è tutto

Le unit permettono una programmazione modulare, il codice di ogni unit può arrivare fino a 64K, quello del programma completo fino a tutta la RAM disponibile (per questo non vi sono più meccanismi di chaining e di overlay). L'introduzione delle unit ha fatto sì che, per evitare ingiustificati e noiosi impacci, ora i file «include» possono essere nidificati (come le stesse unit) fino a otto livelli.

Ma non è tutto. L'esperienza maturata con i compilatori Turbo Modula-2 e Turbo C ha portato a molteplici utilissime estensioni del linguaggio, tante che non possiamo che limitarci ad una esposizione sintetica.

Abbiamo ora nuovi tipi di dati: oltre ai *longint* appena visti, e oltre ai tradizionali *byte* e *integer*, ci sono ora anche *shortint* (da −128 a 127) e *word* (da 0 a 65535, analogo al CARDINAL del Modula-2), quest'ultimo utilissimo anche per operazioni sui valori ritornati da funzioni come *Seg* e *Ofs*. Oltre al tipo *real*, se si compila con la direttiva $N+ si produce codice per il coprocessore numerico e si dispone dei tipi *single* (a 1.5×10E-45 a 3.4×10E−38 +38 con 7-8 cifre significative), *double* (da 5.0×10E-324 a 1.7×10E+308 con 15-16 cifre significative), *extended* (da 1.9×10E−4951 a 1.1×10E+4932 con 19-20 cifre significative) e *comp* (interi a 64 bit).

Sono stati introdotti un tipo *pointer*, compatibile con i puntatori di qualsiasi altro tipo, e un operatore «@» che ritorna l'indirizzo, segmento e offset, del suo operando.

Alle tradizionali procedure standard se ne sono aggiunte altre, tra cui ci piace ricordare *Inc* e *Dec*: non fanno altro che incrementare o decrementare di una unità il loro argomento intero, ma possono contribuire a migliorare le prestazioni di un programma, in quanto le corrispondenti istruzioni del microprocessore sono più efficienti delle normali operazioni di addizione e sottrazione.

È ora possibili attivare la cosiddetta «short circuit bollean evaluation». Si tratta di questo: con le versioni precedenti non era possibile scrivere

```
if (B=0) or (A/B=C) ecc.
```

Perché in ogni caso il programma avrebbe eseguito ambedue i test, ovviamente provocando un errore nel caso B=0. È ora invece possibile (con $B-) produrre codice che valuti una espressione booleana sempre da sinistra verso destra, smettendo appena il suo risultato è chiaro (il primo termine non vero rende comunque falsa una espressione **and**, il primo termine vero rende comunque vera una espressione **or**).

Alla classe della direttive di compilazione è stato aggiunto il potente strumento delle direttive di compilazione condizionale: $DEFINE, $UNDEF, $IFDEF, $ IFNDEF, $IFOPT, $ELSE e $ENDIF, molto simili alle corrispondenti direttive del C, consentono di isolare parti di codice sorgente che verranno compilate o meno secondo che sia definito (con $DEFINE) un certo simbolo o attivata una certa opzione di compilazione.

Accanto agli «inline statments» abbiamo ora anche delle «inline directives»: l'intero corpo di una funzione o procedura, dichiarazioni e istruzioni, viene sostituito da un **inline** seguito da istruzioni in linguaggio macchina; quando il compilatore incontra la chiamata della funzione o procedura non la traduce nella normale CALL, ma inserisce direttamente nel codice oggetto quelle istruzioni. Si tratta di uno strumento estremamente comodo nel caso di routine molto brevi, per le quali la CALL tradizionale produrrebbe una inutile perdita di efficienza. Un esempio tratto dal manuale:

```
procedure DisableInt; inline($FA);
procedure EnableInt; inline($FB);
```

Quando il compilatore trova un DisableInt nel sorgente, inserisce nel codice oggetto un solo byte ($FA, cioè CLI). È anche possibile costruire in questo modo procedure o funzioni con parametri. Oltre al guadagno in termini di spazio e velocità di esecuzione, questa innovazione consente di implementare facilmente persino una caratteristica del Pascal standard fino qui ignorata dalla Borland: la possibilità di passare come parametro anche il nome di una funzione o procedura (la tecnica è illustrata nel file PROCPTR.PAS, uno dei numerosissimi «demo» offerti insieme al compilatore). È stata aggiunta una direttiva **interrupt** che agevola grandemente la scrittura di procedure associate ad un interrupt: si possono passare ad una tale procedura i registri del microprocessore come pseudo-parametri, ed avere così la possibilità di accedervi senza bisogno di far ricorso all'Assembler; il compilatore provvede inoltre automaticamente a salvare i registri all'inizio e a ripristinarli alla fine, nonché ad assegnare al registro DS l'indirizzo del «data segment» del programma.

Vengono ora gestiti anche gli «errori critici» del DOS: i codici d'errore ritornati dalla funzione IOResult compresi tra 150 e 162 corrispondono a quelli che abbiamo visto il mese scorso nella rubrica Turbo Pascal.

Più turbo che mai

Unit, nuovi tipi, compilazione condizionale, soprattutto lo *smart linker*: il lavoro che il Turbo Pascal 4.0 deve compiere quando premiamo il tasto C per compilare è notevolmente più complesso rispetto al passato. Dopo aver visto il Turbo C («turbo» anche lui, ma meno rapido del Turbo Pascal 3.0), ci aspetteremmo, e accetteremmo volentieri, tempi di com-

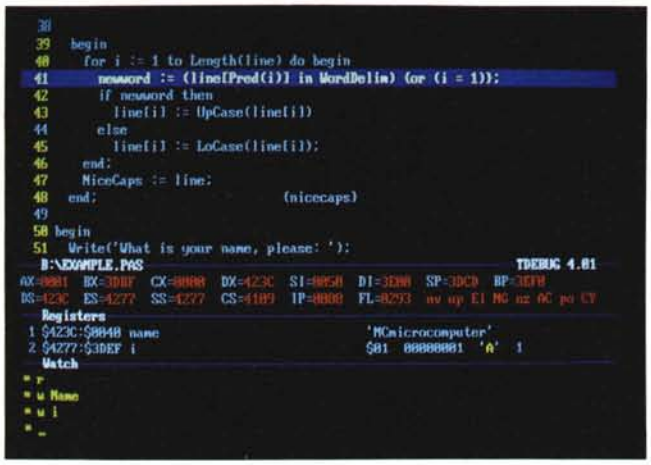


Figura <TDEBUG 4.01>. L'unico vero limite del Turbo Pascal 4.0, è la mancanza di un debugger simbolico. Con 45 dollari si può tuttavia acquistare dalla TurboPower Software il T-DebugPlus 4.0, un debugger funzionalmente analogo al CodeView Microsoft. Nella foto vediamo il video diviso in quattro finestre: il sorgente del programma (o l'Assembler prodotto dal compilatore), i registri del microprocessore, il valore corrente di alcune variabili (sia globali che locali), un'area per i comandi.

pilazione più lunghi di quelli cui ci aveva abituato la versione precedente, ma invece accade proprio il contrario! La Borland dichiara un incremento di velocità di oltre il 60% rispetto alla versione 3.0, e quasi due mesi di sperimentazione consentono di affermare che non c'è nessuna esagerazione: è proprio vero.

Per ottenere un così straordinario risultato si è anche approfittato di quelle che potremo definire le nuove caratteristiche standard di un PC, in particolare per quanto riguarda la RAM: una volta sembravano tanti 64K, poi ci si accontentava di 256, ora è proprio difficile trovare un sistema con meno di 512K. Ecco quindi che il compilatore si carica in memoria un file TURBO.TPL che contiene le unit più usate, in modo da poter evitare «lenti» accessi al disco. Un apposito programma (TPUMOVER.EXE) permette di aggiungere proprie unit a TURBO.TPL o di toglierne quelle meno usate (ad esempio *Turbo3 Graph3*, utili solo se si devono compilare con il 4.0 programmi preparati in origine per il Turbo Pascal 3.0).

Non solo. Potremmo pensare che una compilazione così veloce non può produrre codice di buona qualità; chi ha curiosato nel codice prodotto dal Turbo Pascal 3.0 ha potuto constatare che non venivano condotte ottimizzazioni; ora invece abbiamo «short circuit boolean evaluation» (di cui abbiamo appena parlato), «constant folding» (x:= 3+4*2 viene compilato come se fosse x:= 11, sqr(3) viene subito tradotto in 9), valutazione di una espressione secondo l'ordine che consente di produrre il codice più efficiente, sostituzione delle moltiplicazioni di interi per costanti che siano potenze di due con equivalenti operazioni di shift, in particolare nel caso di quelle moltiplicazioni «invisibili» che vengono eseguite per accedere agli elementi di un array.

I risultati si vedono. Per quanto MC non creda molto alla utilità dei benchmark (avete mai scritto un programma «vero» che passi il suo tempo a calcolarsi un migliaio di numeri primi?), abbiamo voluto confrontare il Turbo Pascal 4.0 e il suo predecessore sulla base di un benchmark a suo tempo messo a punto da Wirth per compilatori Modula-2: tranne due casi strettamente dipendenti il primo dal set di istruzioni dell'8086, il secondo dalle funzioni del DOS, si nota un generale significativo miglioramento di prestazioni, con un vistoso picco nel caso dell'attraversamento di una lista.

Strumenti di sviluppo

Le precedenti versioni del Turbo Pascal ci avevano abituati ad un ambiente di sviluppo integrato semplice e facile da usare: nel corso della prova della versione 3.0 avevamo potuto affermare che non si rimpiangeva affatto l'immediatezza d'uso tipica degli interpreti. Ora abbiamo le unit, la possibilità quindi di compilare separatamente diverse parti di uno stesso programma, in definitiva la possibilità di portare a termine progetti di maggiore complessità.

Tutto ciò non si è però tradotto in una minore immediatezza d'uso, ma al contrario in un ambiente ancora più comodo e soprattutto più versatile. Al posto del tradizionale scarno menu iniziale, abbiamo ora un menu principale (prima riga dello schermo) attraverso il quale si accede alle opzioni di editing (Edit) e di esecuzione (Run) di un programma, ma anche ai sottomenu File, Compile Options. Il primo consente di caricare e salvare file, di esaminare il contenuto di una directory e di cambiare la directory corrente, di tornare momentaneamente al DOS senza bisogno di uscire dal Turbo Pascal. Il secondo offre le diverse possibilità di compilazione del solo file in memoria, di tutti i file che compongono un programma (opzione Build), dei soli file «non aggiornati» (opzione Make), di compilazione in memoria e su disco; c'è anche un'opzione Find error che consente di rintracciare nel sorgente il punto che ha causato un errore durante l'esecuzione di un programma. Il menu Options permette di attribuire un valore di default alle direttive di compilazione (in modo da evitarci di dover riscrivere le scelte che riteniamo più appropriate nel sorgente di tutti i nostri programmi), di determinare le directory in cui il compilatore deve salvare e poi cercare i vari tipi di file, di definire alcuni aspetti del comportamento dell'editor (ad esempio di attivare o disattivare la creazione di file BAK); tutte le scelte possono essere poi salvate permanentemente in un file TURBO.TP. Mentre si percorre l'albero dei menu si può premere F1 per avere in ogni momento una schermata di help relativa all'opzione su cui si è posizionati.

L'editor ha mantenuto le caratteristiche delle versioni precedenti, con qualcosa in più; in particolare è risultata anche più comoda di quanto si potesse pensare la cosiddetta «pick list». Un programma può essere scomposto in diverse unit, sia il programma principale che le sue unit possono includere dei file; ogni volta che si passa da un file all'altro, l'editor memorizza in una sua tabella (la «pick list», appunto) i nomi dei vari file; quando si vuole caricare in memoria un file diverso da quello già in memoria, premendo Alt-F3 compare su video questa tabella con il cursore posizionato sull'ultimo file su cui si è lavorato: se si vuole scegliere un altro file si può spostare il cursore, altrimenti basta premere *Enter*; verrà caricato in memoria il file, con il cursore posizionato lì dove lo si era lasciato l'ultima volta. Anche in fase di editing si dispone inoltre di una potente funzione di help: se si posiziona il cursore sul nome di una unit, funzione o procedura, premendo Ctrl-F1 compare una schermata di spiegazioni (caratteristiche generali, unit usate, altre funzioni o procedure affini o complementari).

C'è però chi è irrimediabilmente affezionato al proprio editor; nessun problema. Oltre al compilatore interattivo (TURBO.EXE, che richiede 384K di RAM e due foppy o un hard disk), viene fornito anche un compilatore più tradizionale (TPC.EXE), che ha le stesse potenzialità del primo ma si accontenta di 256K e un floppy e può essere usato per compilare file prodotti con un altro editor.

Ancora. Può capitare che alcune parti di un programma vadano scritte in Assembler; in questi casi l'opzione Make del compilatore (interattivo o tradizionale che sia) non può gestire automaticamente quelle situazioni in cui il file sorgente ASM è stato modificato ed è quindi affiancato da un file OBJ ormai «vecchio»: è necessario servirsi di un Assembler (possono essere usati il MASM della Microsoft o un prodotto compatibile). Viene quindi offerto MAKE.EXE, un programma di utilità che, analogamente all'omonimo di UNIX, definisce le dipendenze tra i diversi file di un programma

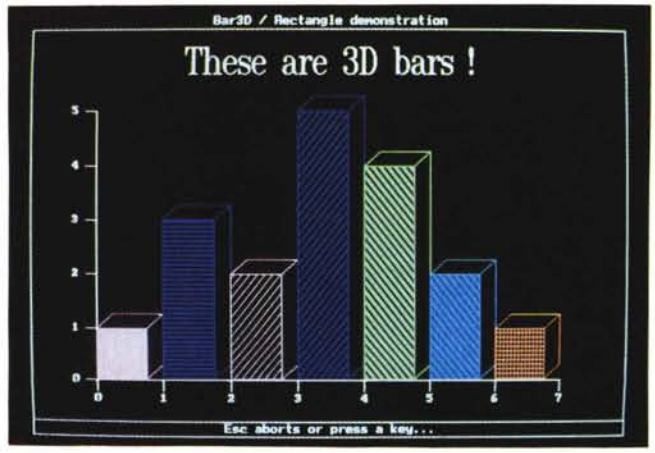


Figura <3 D Bars>. Con la procedura Bar3D *della unit* Graph *si possono tracciare facilmente grafici a barre tridimensionali; la procedura* SetTextStyle *permette di scrivere con diverse font e di variare a volontà le dimensioni dei caratteri.*

ed esegue le opportune azioni ogni volta che un file viene trovato più «vecchio» di uno di quelli da cui dipende. Queste azioni possono essere qualsiasi comando DOS, compresa l'esecuzione di un Assembler o di un file batch. Un altro programma (TOUCH.COM, anch'esso derivato da UNIX) permette di influenzare il comportamento di MAKE cambiando la data di uno o più file.

Ispirato da UNIX è anche GREP.COM: si tratta di un programma in grado di cercare una stringa in più file contemporaneamente, in grado, ad esempio, di mostrarci in quale riga di quale file è dichiarato, ed in quali usato, un certo identificatore (in realtà GREP è molto più potente di quanto può sembrare da questi brevi cenni, in quanto non cerca semplici stringhe ma «regular expressions»; una discussione esauriente di MAKE, TOUCH e GREP richiederebbe tuttavia anche più di un intero articolo).

Infine il manuale. Non siamo ancora ai livelli di eccellenza che si riscontrano in prodotti che costano quattro o cinque volte tanto, ma è molto migliorato. Bisogna qui fare un inciso: il manuale di un compilatore viene usato inizialmente per acquisire familiarità con il linguaggio e la sua implementazione, poi come testo di consultazione. Si tratta di usi molto diversi, dei quali è sicuramente più importante il secondo; è per questo che possiamo esprimere un giudizio largamente positivo: all'inizio si fatica un po' a capire cosa è cambiato rispetto alle versioni precedenti, in quanto l'esposizione molto graduale sembra sempre rimandare a qualcosa di non ancora detto; una volta rotto il ghiaccio, tuttavia, ci si trova a poter disporre di informazioni estremamente precise ordinate e dettagliate. Vengono illustrati con dovizia di particolari i meccanismi di gestione della memoria in generale e della memoria dinamica in particolare, la gestione dei file (c'è anche un esempio di scrittura di propri «text file device drivers» associati ad una porta seriale), le convenzioni da seguire per la scrittura di routine in Assembler. Un lungo capitolo è dedicato alla esposizione di tutte le funzioni e procedure, presentate in ordine alfabetico; per ognuna si propongono una breve descrizione, la sintassi, il tipo dell'eventuale risultato, osservazioni di carattere generale, limitazioni, differenze dalla versione 3.0, i nomi di funzioni o procedure correlate, un breve esempio. Rimangono inoltre le tradizionali appendici dedicate alle direttive di compilazione e ai codici d'errore, integrate da esaurienti esposizioni di MAKE, TOUCH e GREP.

Limiti

Le variabili globali non possono occupare complessivamente più di 64k, ma si può aggirare l'ostacolo creando variabili dinamiche.

Si può compilare un programma in modo che tragga vantaggio dalla presenza di un coprocessore numerico ma, come nelle versioni precedenti, il programma non girerebbe su una macchina che ne fosse priva.

Attivando le direttive $D e $T si può produrre un file con estensione TPM (Turbo Pascal Map) che può essere convertito (da un programma TPMAP.EXE) in un file con estensione MAP, utile per un debugger simbolico che però non viene fornito. Il manuale fa riferimento al SYMDEB della Microsoft (antenato del ben più potente CodeView) e al Periscope della omonima casa americana. La TurboPower Software ha inoltre proposto il T-DebugPlus 4.0, un debugger simbolico di tutto rispetto, funzionalmente equivalente al CodeView e forse con qualcosa in più (ad esempio la possibilità di richiamare ed editare comandi già dati), a prezzi molto convenienti: 45 dollari, o 90 se si desidera anche il sorgente (è infatti scritto in Turbo Pascal). Può essere richiesto alla TurboPower Software, P.O. Box 66747, Scotts Valley, CA 95066-0747 USA. Dovrebbe comunque essere presto disponibile anche un debugger targato Borland.

Non è possibile creare file OBJ, ma si possono usare quelli prodotti dal Turbo C.

Non è ancora disponibile (inizio di giugno) il manuale in italiano, ma la Edia sta provvedendo: è probabile che sia già pronto quando la rivista sarà in edicola.

Conclusioni

Non siamo riusciti ad esporre in poche pagine tutte le caratteristiche del Turbo Pascal 4.0, né tutte le notevoli utili innovazioni rispetto alle versioni precedenti. Non siamo riusciti ad andare oltre poche righe per i suoi limiti. Con questo potremmo avere detto tutto.

Il Turbo Pascal 4.0 è un vero e proprio super-Pascal, ma anche un Modula-2 a misura d'uomo; è uno strumento sicuramente adatto alla programmazione professionale, ma mantiene intatte le caratteristiche di agilità d'uso che avevano decretato il successo delle versioni precedenti; ancor più di queste si presta anzi anche ad un uso didattico, vista la maggiore enfasi sulla programmazione modulare. Il prezzo poi è in sé ampiamente giustificato, vista la qualità del prodotto, ed è molto conveniente la possibilità di «upgrade» per chi già avesse la versione precedente.

Vorremmo tuttavia aggiungere qualcosa riguardo al prezzo, un po' più alto di quello del 3.0: sarebbe un grave errore pensare alla nuova versione come destinata a «pochi eletti», alla precedente come a quella per «tutti».

Negli Stati Uniti il Pascal non domina più incontrastato la scena universitaria: al MIT, ad esempio si insegna informatica con un testo — bellissimo — di Abelson e Sussman (*Structure and Interpretation of Computer Programs*, MIT Press, 1985) basato sullo Scheme, una sorta di Lisp «pascalizzato». I primi due capitoli sono dedicati ai fondamentali concetti di procedure *abstraction* e *data abstraction*. Il Pascal standard poco si presta ad approfondire argomenti del genere, ma le unit del Turbo Pascal 4.0, come già si è notato oltre Atlantico, possono costituire uno strumento molto agile per un approccio al tempo stesso facile e non banale a quei concetti. Il Turbo Pascal si sta diffondendo sempre più nel mondo della scuola e sarebbe auspicabile che, anche al fine di non allargare ancor più quell'oceano, si approfittasse tutti della possibilità che la Borland ci offre di adottare un linguaggio che, oltre ad essere comodo e divertente da usare, consente l'adozione indolore delle migliori tecniche di programmazione.

MC

power & compatibility

ERSONAL WORK STATION 16 e 32 BIT

PX-30
Cpu 8088 10MHz, 256-640K ram, floppydisk 3,5 pollici, hard disk 20-40MB

PX-50
Cpu 80286 8MHz, 512K-1MB ram, floppy disk 3,5 pollici, hard disk 20-40MB

PX-80
Cpu 32 bit 80386 16MHz, 2MB ram, floppy disk 3,5 pollici, hard disk 20-40MB

SUPERMICRO 16 e 32 BIT

AX-60
Cpu 16 bit 80286 12MHz, 512K-2MB ram, floppy disk 5,25 e 3,5 pollici, hard disk 40-230MB

AX-80
Cpu 32 bit 80386 16MHz, 2MB ram, floppy disk 5,25 e 3,5 pollici, hard disk 40-230MB

UNIDATA

UNIDATA - Via S. Damaso 20 - 00165 Roma

Tel. 06/63.66.58 - Telex 623122 UNITAT I