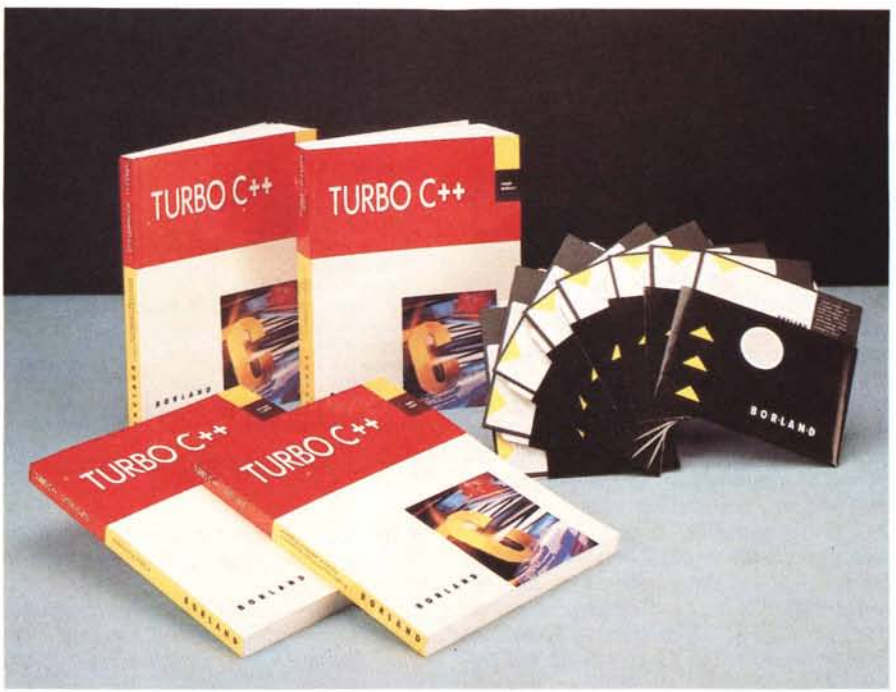
È insomma un notevole passo avanti rispetto al «vecchio» ambiente integrato del Turbo Pascal originario. Un ambiente realmente produttivo per un programmatore, il quale oltretutto può estenderlo a piacimento integrandovi, se vuole, perfino dei propri tool.

Evoluzione aziendale

Parallelamente all'evoluzione dei propri prodotti, non si può non sottolineare come Borland abbia operato una evoluzione della propria immagine aziendale. Il primo Turbo Pascal che girava in 30K era un meraviglioso compilatore «per hacker» ma non certo uno strumento per professionisti. Le versioni successive, fatte uscire parallelamente ad una pletora di nuovi linguaggi Turbo e di prodotti assai disparati, erano molto migliorate ma ancora tipicamente prodotti rivolti al mondo degli studenti e degli hobbysti.

Ora però l'immagine un po' freak della Borland collegata a questi prodotti giovanili è cambiata. Una drastica riorganizzazione interna ha portato la ditta di Philippe Kahn a concentrarsi su pochi aspetti strategici del mercato ricercando in ciascuno l'eccellenza. I prodotti collaterali sono stati cancellati, e dei molti linguaggi Turbo solo due, il C ed il Pascal, sono stati salvati dall'accetta. Questi però sono stati portati a livelli professionali mediante il miglioramento tecnico dei compilatori e l'introduzione delle rispettive estensioni per OOP, tecnica nella quale in Borland si crede molto. I tool di supporto sono stati migliorati e rinforzati. La missione aziendale di Borland è ora quella di creare strumenti di programmazione per professionisti, per la manipolazione di dati in ambienti «corporate», per la produttività individuale nel mondo del lavoro. Meno romantico, forse, ma più concreto di prima. Lo sforzo di promozione del Turbo C++ è dunque legato anche al consolidamento di questa immagine «seria» dell'azienda; la quale per dimostrare tutta la sua autorevolezza ha organizzato per l'anno in corso una serie di seminari sulla programmazione OOP, che si terranno un po' in tutto il mondo, ed è riservata agli sviluppatori professionisti.

Qualche anno fa, introducendo la prova su MC del compilatore Turbo Pascal, dissi che la Borland aveva «Turbato» il mondo. Credo che lo stia facendo ancora. In un modo diverso e maggiormente adatto ai tempi, ma indubbiamente lo sta facendo ancora. Se oggi siamo tutti abituati ad un certo modo di intendere il ciclo di sviluppo del software è anche merito loro.



Turbo C++ 1.0

di Sergio Polini

Noi ce la prendiamo con i carabinieri, i danesi con gli abitanti di Aarhus. Le barzellette nostre e loro, fatta eccezione per le «vittime», sono praticamente identiche. Sappiamo bene che i veri militi sono ben diversi da quelli delle storielle (delle quali magari ridono anche loro...), e lo stesso può sicuramente dirsi di quei connazionali di Bo. Basti pensare che proprio ad Aarhus, nel 1975, Bjarne Stroustrup si è laureato in matematica e informatica. Quattro anni dopo, ottenuto il Ph.D. in Inghilterra dalla università di Cambridge, Stroustrup varcò l'oceano per approdare ai mitici AT&T Bell Laboratories. Lì si è interessato di sistemi operativi e di sistemi distribuiti, ma anche di simulazione. In quegli anni si disponeva per queste cose del Simula 67, che aveva arricchito la struttura a blocchi tipica dell'Algol 60 con le «classi», ovvero con costrutti atti ad emulare gli oggetti del mondo reale in quanto capaci di comportamenti autonomi piuttosto che «nidificati». Ma era poco efficiente. Stroustrup mise così a punto un «C con Classi»: un linguaggio che, pur mantenendo una notevolissima compatibilità con il C e l'efficienza di questo, vi innestava fin dal 1980 il concetto di «classe». Seguirono poi «operator overloading», «reference» e funzioni virtuali. Poco alla volta, dalla mera astrazione sui dati consentita dalle classi si passava all'ereditarietà e al polimorfismo, cioè alla programmazione orientata all'oggetto. Permaneva tuttavia la ferma intenzione di non dar vita ad un nuovo linguaggio, ma piuttosto di implementare un «migliore C», che comprendesse come subset il «vecchio C»; per dirla con Stroustrup, nessun linguaggio è perfetto, ognuno ha i suoi problemi, ma almeno quelli del C erano ben noti. A sottolineare che si trattava appunto di evoluzione invece che di rivoluzione, Rick Mascitti coniò il nome C++. Che rende molto bene l'idea. Nel luglio del 1983 avvenne la prima installazione fuori del gruppo di ricerca guidato da Stroustrup. Nell'ottobre del 1984, il C++ veniva illustrato dal suo ideatore in un articolo dell'AT&T Bell Laborato-

ries Technical Journal, significativamente intitolato «Data Abstraction in C». La AT&T immise nel mercato le tre versioni dalla 1.0 alla 1.2 nel biennio 1985-86; in tutti i casi si trattava di un traduttore: il codice veniva convertito in «vecchio C» e quindi compilato. Nel 1988 vennero realizzati i primi compilatori; nel 1989 è arrivata la versione 2.0, caratterizzata da importanti innovazioni quali l'ereditarietà multipla (la più appariscente), ma soprattutto molte «piccole» ma importanti modifiche tese a facilitare la costruzione e l'uso di librerie di funzioni.

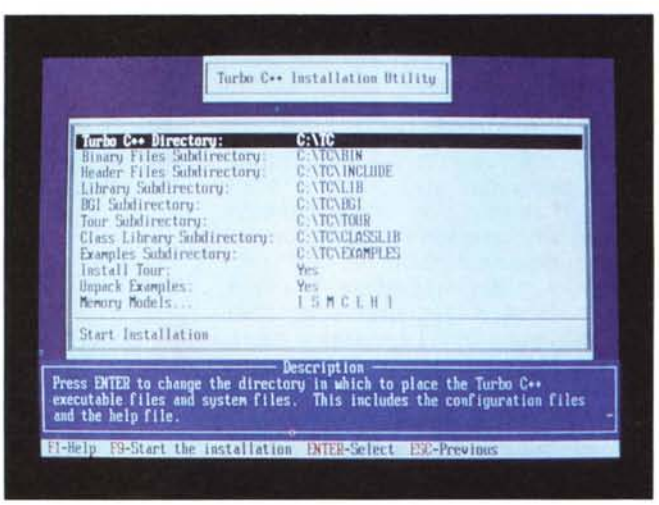
Un linguaggio giovane, quindi, ma estremamente promettente. Un linguaggio sicuramente molto «di moda» ma anche, soprattutto nella versione 2.0, uno strumento in grado di aumentare significativamente la produttività di chi fa software. Ne volete un'autorevole testimonianza? Dr. Dobbs's Journal, maggio 1989: il direttore Jonathan Erickson sottolinea l'importanza del trasferimento di Greg Whitten, impegnato per dieci anni nella realizzazione di tutti i linguaggi Microsoft, al settore dedicato allo sviluppo di programmi applicativi. Lo stesso Whitten aveva spiegato che il codice degli applicativi Microsoft ammonta a svariati milioni di linee di codice; le immaginabili difficoltà insite nella gestione di tali volumi rischiavano di compromettere i tempi di rilascio e quindi gli utili della società. Di qui i nuovi incarichi assegnatigli dalla Microsoft: dirigere lo sviluppo di nuove applicazioni object-oriented e applicare la tecnologia object-oriented allo sviluppo di applicazioni. Quanto ai linguaggi da usare, gli orientamenti Microsoft non sono ancora noti. È sicuro che un compilatore C++ avrà da dire la sua, ma si è parlato anche di un «Visual» Basic dedicato alla realizzazione di interfacce utente grafiche. Soprattutto ancora non sembra giunto il momento di un C++ per il mercato. C'è chi ritiene che la Microsoft intenda proporre graduali estensioni del suo compilatore C, ma può anche ritenersi che la recente proposta di un compilatore «ancora C» abbia le sue buone motivazioni: la necessità di fare i conti con i diversi ambienti dalla stessa Microsoft proposti (MS-DOS, Windows, OS/2), l'attenzione estrema alla ottimizzazione del codice con conseguente urgenza di contrastare anche concorrenti «minori» (la Datalight al tempo del C 5.0, la Watcom ora), la relativa giovane età di un linguaggio che da un lato solo lo scorso anno, con la versione 2.0, ha messo a punto meccanismi adeguati per la costruzione e l'uso di librerie di funzioni, dall'altro, come vedremo, richiede l'esistenza di apposite librerie di classi per poter essere usato al meglio.

Turbo C++ 1.0

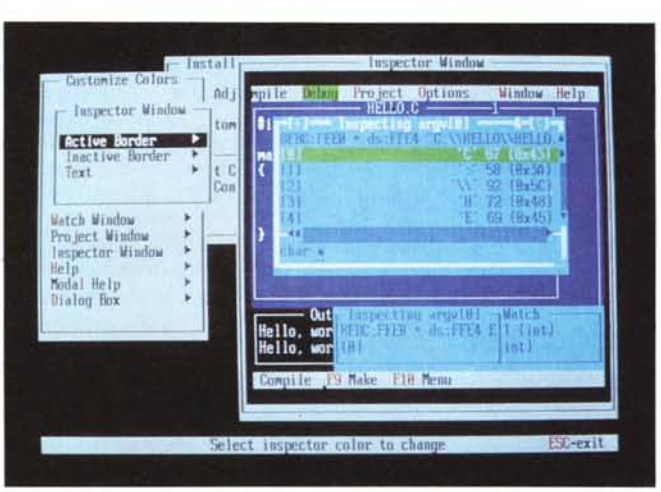
Produttore:
Borland International, Inc.
1800, Green Hills Road
P.O. Box 660001
Scotts Valley, CA 95066-0001
Distributore:
Borland Italia Srl
Via Cavalcanti, 5 - 20127 Milano
Telefono: 02/2610102
Prezzi *(IVA esclusa):*

Turbo C++ 1.0	*L. 399.000*
Turbo C++ 1.0 Professional	*L. 599.000*
Upgrade (da qualsiasi linguaggio Borland)	*L. 299.000*

La Borland ha fatto il suo prepotente ingresso nel mercato con prodotti innovativi come il Turbo Pascal e il SideKick; ha proseguito con un incredibile Turbo Prolog; ha abbandonato il progetto di un Turbo Modula-2 in favore di un Turbo C non appena ha capito che l'erede del Pascal non avrebbe scalfito il dominio del C; ha arricchito il Turbo Pascal prima di caratteristiche «modulari» poi di strumenti per la programmazione object-oriented tratti sia dall'Object Pascal che — guarda caso — dal C++. Società piccola (almeno rispetto alla Microsoft) e quindi forse proprio per questo più orientata all'innovazione, poteva lasciarsi sfuggire il «nuovo C»? Sicuramente no, tanto più che da tempo erano presenti sul mercato traduttori e compilatori C++ per MS-DOS (Glockenspiel, Guidelines, Zortech, ecc.), ed era importante offrire non appena possibile non solo un compilatore, ma forse soprattutto una libreria di classi che si proponesse come standard. Le caratteristiche evolutive del C++ consentivano del resto un approccio relativamente indolore: sia il



Le opzioni proposte all'utente dalla procedura di installazione.



Mediante TCINST è possibile, tra l'altro, cambiare i comandi dell'editor e le varie combinazioni di colori.

C di Kernighan e Ritchie che l'ANSI C possono infatti essere considerati, salvo incompatibilità di poco momento, come sottoinsiemi del C++. Ecco quindi l'idea Borland: un compilatore multi-linguaggio in un nuovo ambiente integrato notevolmente potenziato e assistito da superbi tool di sviluppo. Non vi piace (ancora...) il C++? Vi si offre una sorta di super Turbo C 3.0. Vi interessa un compilatore C++ conforme alla versione 2.0 della AT&T? Eccovi il Turbo C++ 1.0.

L'uso di _new_handler e di set_new_handler(), non documentato né nei manuali né nell'help in linea, ma correttamente implementato.

```
// newhand.cpp
// esempio di uso di _new_handler o di set_new_handler() adattato da:
// - Bjarne Stroustrup, The C++ Programming Language, Addison-Wesley,
//   Reading, Massachussets, 1987, p. 93
// - Stanley B. Lippman, C++ Primer, Addison-Wesley, Reading,
//   Massachussets, 1989, pp. 139, 171

#include <iostream.h>
#include <process.h>

typedef void (*PF)();           // oppure:
extern PF set_new_handler(PF);  // extern void (*_new_handler)();

void memoria_insufficiente()
{
        cerr << "       memoria insufficiente\0;
        exit(1);
}

void esaurisciMemoria(unsigned blocco)
{
        static int livelli = 1;
        static int riferito = 0;
        ++livelli;
        double *ptr = new double[blocco];
        if (ptr)
                esaurisciMemoria(blocco);
        delete ptr;
        if (!riferito++)
                cout << "       Memoria esaurita:"
                     << "   blocco: " << blocco
                     << "   livelli: " << livelli << "\0;
}

main()
{
        unsigned blc;
        cout << "Blocco: ";
        cin >> blc;

        cout << "Senza _new_handler:\0;
        esaurisciMemoria(blc);

        cout << "Con _new_handler:\0;
        set_new_handler(memoria_insufficiente);
        // oppure: _new_handler = memoria_insufficiente;
        esaurisciMemoria(blc);
        return 0;
}
```

Installazione

Il Turbo C++ può essere installato su computer IBM o compatibili con almeno 640K di RAM, un disco rigido e un floppy. Si richiede un DOS 2.0 o successivo, anche se, per sfruttare al meglio le possibilità offerte dall'ambiente integrato, è preferibile una versione 3.x o 4 (solo in questo caso, infatti, i programmi conoscono la directory in cui risiede il loro eseguibile; detta in C, argv[0] contiene il pathname completo del programma). È possibile, per la prima volta in un compilatore Borland, l'uso del mouse, che può essere Microsoft, Logitech, Mouse Systems o IMSI, con versioni minime, rispettivamente, 6.1, 3.4, 6.22 e 6.11.

Vengono forniti otto dischetti, che possono essere installati sul disco rigido unicamente mediante un programma INSTALL; non è possibile semplicemente copiare e scompattare i vari file, in quanto alcuni di questi sono divisi in più tronconi che solo INSTALL può riunificare. Ciò si rende necessario in quanto TC.EXE, il compilatore integrato, occupa circa 870K, e il relativo file di help altri 670K. La procedura di installazione è comunque estremamente semplice e rapida, secondo lo standard a cui la Borland ci ha ormai abituato: si tratta solo di indicare le directory in cui installare i vari tipi di file e di scegliere se si vuole installare TCTOUR (un tutor interattivo su alcune caratteristiche dell'ambiente integrato), se si vogliono scompattare gli esempi (proposti in file ZIP), se si vogliono installare le librerie per tutti i modelli di memoria.

Al termine di una installazione completa, il prodotto occupa circa 5 mega e mezzo; volendo eliminare il TCTOUR, gli esempi, il sorgente del codice di start-up e i file DOC, si risparmierebbe circa un mega. È preferibile comunque assicurarsi di avere sufficiente spazio libero su disco, non solo, come è ovvio, per le proprie applicazioni, ma anche perché sia il compilatore integrato che quello separato (TCC.EXE) si avvalgono del VROOMM, il sistema di *swapping* già usato nel Quattro Pro che, con tecnologia analoga a quella adottata in sistemi operativi come Unix, consente di usare il disco per far girare applicazioni che richiederebbero più RAM di quella fisicamente disponibile. Grazie al VROOMM (Virtual Run-Time Object Oriented Memory Manager) è possibile compilare nell'ambiente integrato anche dopo aver visto giungere fino a zero il conto della RAM residua, o vedere un messaggio «Available memory 0» dopo una compilazione con TCC.EXE.

A completamento dell'installazione si può editare il file TURBOC.CFG, nel quale si possono riportare, con un normale editor, le opzioni che si preferiscono per il compilatore separato. Si può anche usare TCINST.EXE per modificare molte delle caratteristiche del compilatore integrato, in particolare i comandi dell'editor e le combinazioni di colori, non modificabili in altro modo (molte altre caratteristiche possono essere impostate dallo stesso ambiente integrato). In un complesso caratterizzato da notevole potenza e flessibilità abbiamo riscontrato un solo neo. Correggendo una scelta tradizionale della Borland, nei menu dell'ambiente integrato è ora possibile posizionarsi con il cursore anche sulle opzioni disabilitate (utile per chiedere aiuto su di esse premendo F1), ma la barra di selezione assume in questo caso una colorazione grigio scuro su nero, scarsamente leggibile su molti monitor; non siamo riusciti ad assegnare nuovi colori con TCINST.

Documentazione

Vengono forniti quattro manuali. *Getting Started* fornisce istruzioni sulla installazione e sull'uso dell'ambiente integrato, e contiene due tutorial sul C e sul C++. Ambedue dedicati a chi si accosti per la prima volta a uno dei due linguaggi, svolgono egregiamente il loro compito e sono accompagnati da numerosi file di esempio su disco. La *User's Guide* contiene l'illustrazione completa dell'ambiente integrato e illustra con

molta chiarezza le numerose nuove caratteristiche di questo. Vi vengono anche documentate le opzioni del compilatore separato e il funzionamento di MAKE e TOUCH e delle nuove versioni di TLIB e TLINK. Un'appendice è dedicata ad una delle nuove possibilità dell'editor: viene fornito un programma TEMC.EXE (Turbo Editor Macro Compiler) che, avuto in input un file ASCII redatto secondo semplici regole, consente di aggiungere al file di configurazione TCCONFIG.TC nuovi comandi per l'editor, definiti come combinazioni di quelli predefiniti. La *Programmer's Guide* è invece espressamente dedicata al programmatore esperto e, potremmo dire, esigente. Il lungo primo capitolo (oltre 150 pagine) contiene una descrizione sia formale che discorsiva del C e del C++ come implementati dalla Borland. Un'appendice illustra anche tutte le scelte operate in relazione a quegli aspetti per i quali lo standard ANSI lascia qualche libertà agli implementatori. Dopo un breve capitolo dedicato alla illustrazione per categorie delle funzioni di libreria (documentate in dettaglio in un quarto manuale), seguono una esposizione della libreria di I/O del C++ 2.0, *iostream* (i brevi cenni dedicati agli *stream* delle precedenti versioni del C++ AT&T sono integrati da un esauriente file OLDSTR.DOC su disco) e dettagliate informazioni sui diversi modelli di memoria, sulle opzioni per i calcoli in virgola mobile, sulle possibilità di overlay mediante il VROOMM, sulle funzioni di gestione del video nei modi testo e grafico, sulla interfaccia con moduli in Assembler, sui messaggi d'errore. Il quarto e ultimo manuale, *Library Reference*, contiene la documentazione delle oltre 470 funzioni di libreria (esclusa la TCLASSx.LIB, di cui diremo poi) e delle variabili globali predefinite. Per ogni funzione abbiamo una breve descrizione dell'azione svolta, la sintassi, il file «h» che ne contiene il prototipo, note sull'uso e sul funzionamento, i possibili valori del risultato, precisazioni sulla portabilità (con riguardo all'ANSI C e a Unix) e sulla eventuale presenza di funzioni simili in Turbo Pascal, l'indicazione di eventuali altre funzioni correlate, quasi sempre un esempio d'uso.

Si tratta in complesso di oltre 1400 pagine ben organizzate, di sicuro aiuto sia per il principiante che per l'esperto. Si potrebbe magari desiderare che, nella *Programmer's Guide*, venissero illustrare schematicamente, oltre che discorsivamente, le differenze tra Turbo C++, Turbo C 2.0 e ANSI C, ora sparpagliate qua e là; ma non è questione di grande importanza. Lascia invece un po' perplessi qualche curiosa omissione. Un C++ 2.0 dovrebbe essere accompagnato da un file *new.h*, contenente le dichiarazioni di una variabile predefinita *new handler* (di tipo puntatore a una funzione senza argomenti che ritorni **void**), di una funzione *set new handler()* (con identico tipo per l'argomento e per il valore di ritorno) e di una variante dell'operatore **new** con un ulteriore operando, indicante l'indirizzo nel quale va collocato l'oggetto allocato. Per chi non lo sapesse, l'operatore **new** viene usato in C++ al posto della funzione *malloc()* per assegnare ad un puntatore l'indirizzo di un'area allocata dinamicamente; nel caso di esaurimento della memoria **new** ritorna zero, ma prima verifica se *new handler* ha valore nullo o contiene l'indirizzo di una funzione che, in tal caso, viene eseguita; ciò consente di definire una funzione che gestisca le situazioni di memoria insufficiente (ad esempio: chiusura ordinata dei file, emissione di un messaggio di errore, terminazione del programma) e di installarla o mediante una assegnazione diretta di *new handler* o con *set new handler*. Tutto ciò non viene documentato (si riesce solo a trovare *new handler* tra le variabili predefinite di un programma mediante il Turbo Debugger), ma, per fortuna, funziona perfettamente. La variante "estesa" dell'operatore **new** funziona invece solo se si provvede all'*overloading* dell'operatore predefinito. Sarebbe stato certo preferibile che tutto ciò fosse documentato.

Accanto ai manuali abbiamo alcuni file di documentazione su disco, alcuni specifici (OLDSTR.DOC per gli *stream*, THELP.DOC per l'help residente da usare fuori dell'ambiente integrato, UTIL .DOC per i vari programmi di utilità, TCALC.DOC per il «solito» spreadsheet che accompagna i compilatori Borland, CLASSLIB.DOC, per la libreria di classi), altri genericamente relativi al prodotto: README, che per fortuna non contie-

La variante «estesa» dell'operatore new consente di collocare in un'area di memoria scelta dal programmatore una struttura di dati. Sotto Unix, ad esempio, può essere usata per allocare spazio nella «shared memory»; sotto DOS potrebbe essere usata per definire strutture coincidenti con aree della memoria di sistema (nella figura, con la memoria video). Si richiede l'«overloading» dell'operatore new, a cui si può provvedere con un file new.h come quello qui proposto.

```
// new.h

extern void (* _new_handler)();
extern void *set_new_handler(void (*)());

inline void* operator new(unsigned, void *p) { return p; }


// newdemo.cpp
// NB: da compilare nei modelli "large data" (es.: compact)

#include <conio.h>
#include <new.h>

const int Rows = 25;
const int Cols = 80;
const int VideoSize = Rows * Cols;

struct VideoCell {
    unsigned char ch;
    unsigned char attr;
};

class VideoBuff {
    VideoCell array[VideoSize];
public:
    VideoBuff() {
        for (int i = 0; i < VideoSize; ++i) {
            array[i].ch = '?';
            array[i].attr = 0x07;
        }
    }
    ~VideoBuff() {
        for (int i = 0; i < VideoSize; ++i) {
            array[i].ch = ' ';
            array[i].attr = 0x07;
        }
    }
    void SetVideoCell(int i, unsigned char c, unsigned char a) {
        array[i].ch = c;
        array[i].attr = a;
    }
};

main(void)
{
    text_info ti;
    gettextinfo(&ti);
    void *VRAMptr = ti.currmode == MONO ? (void *)0xB0000000L
                                        : (void *)0xB8000000L;
    VideoBuff *VBPtr = new (VRAMptr) VideoBuff;
    for (int i = 9 * Cols; i < 10 * Cols; ++i)
        VBPtr->SetVideoCell(i, '!', 0x70);
    getch();
    VBPtr->VideoBuff::~VideoBuff();
}
```

```
Object                  classe "base"
 Error                  usata per errori di allocazione di memoria
 Sortable               classi per oggetti ordinabili
 Association            classi per oggetti da includere in un Dictionary
 Container              classi che contengono oggetti di altre classi
  Stack                 classi di tipo "stack" (con push(), pop(), ecc.)
  Queue                 classi di tipo FIFO (con put(), get(), ecc.)
  Deque                 Queue a due estremi (putLeft(), putRight(), ecc.)
  Collection            classi per gestire collettivamente n altri oggetti
   HashTable            classi per Collections non ordinate
    Bag                 sostanzialmente equivalente a HashTable
     Set                come Bag, ma senza oggetti uguali tra loro
      Dictionary        Set di oggetti derivati dalla classe Association
   List                 Collection di oggetti concatenati tra loro
   DoubleList           Collection di oggetti concatenati nei due sensi
   AbstractArray        classi di oggetti accessibili mediante un indice
    Array               AbstractArray con addAt() e operatore []
    SortedArray         AbstractArray con elementi ordinati e operatore []
ContainerIterator       classe "base" della gerarchia degli iteratori
 ArrayIterator          per iterazioni sugli array
 ListIterator           per iterazioni sulle List
 DoubleListIterator     per iterazioni sulle DoubleList
```

La gerarchia di classi implementata nella libreria TCLASS.

ne, come in altre occasioni, sostanziose rettifiche alla documentazione cartacea, e HELPME!.DOC, che curiosamente è l'unico posto in cui abbiamo trovato una importante indicazione circa l'uso delle keyword **far** e **huge**, su cui torneremo tra breve.

Abbiamo infine un potente help in linea, accessibile sia dall'ambiente integrato che da dovunque se reso residente con THELP.COM, e contenente in pratica buona parte dei manuali *User's Guide* e *Library Reference*. Funzionalità già viste, come l'uso della tecnologia dell'ipertesto e la possibilità di copiarne brani e esempi in una finestra di editing, vi compaiono significativamente potenziate.

Implementazione del linguaggio

Sarebbe magari meglio dire «dei linguaggi». Il Turbo C++ è infatti insieme un ANSI C rispettoso delle prescrizioni IEEE e un C++ conforme alla versione 2.0 della AT&T. Un'apposita opzione consente perfino di attivare quattro diversi insiemi di parole riservate: quelle del C Kernighan e Ritchie, del C Unix System V, dell'ANSI C e del Turbo C++, il quale ultimo comprende alcune estensioni quasi tutte già presenti nel Turbo C 2.0. Si può scegliere anche se assumere che tutti i file con estensione «c» sono da intendere come sorgenti C++ o se limitare il nuovo linguaggio ai file con estensione «cpp». Quasi non si riesce a credere che il compilatore effettivamente riesce a rispettare le scelte dell'utente, consentendo e vietando costrutti ora validi ora illegali secondo il tipo di linguaggio e la portabilità che si preferiscono.

Analogo sforzo si osserva nella organizzazione degli header file, che la *Programmer's Guide* non a caso classifica, tra l'altro, anche in relazione allo standard di riferimento: troviamo i file voluti dallo standard ANSI (da *stdio.h* a *locale .h*), qualcuno ispirato da Unix (si tratta di *fcntl.h* e *values.h*), quelli del C++ (dai tipici *stream.h, iostream.h o complex.h* a un interessante *bcd.h*, che definisce una classe per l'uso di numeri BCD), e quelli relativi alle estensioni Borland (da *alloc.h* a *graphics.h*).

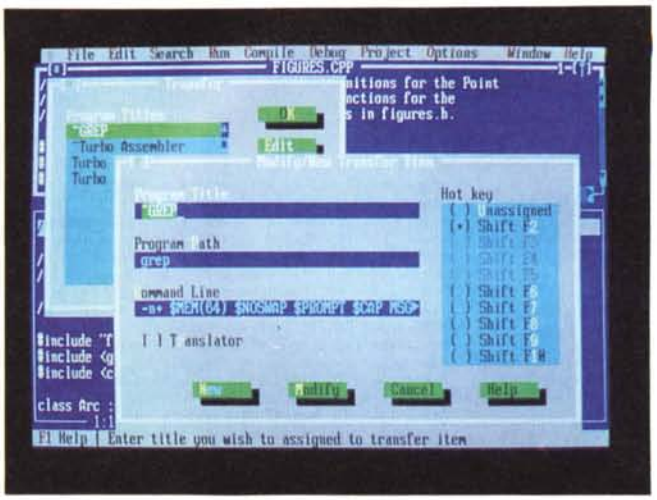
Le estensioni consentono tra l'altro di trarre il massimo beneficio dalla articolata offerta di sei modelli di memoria: tiny, small, medium, compact, large e huge. Qui il lettore utente dei compilatori Microsoft deve fare attenzione: compact, large e huge hanno infatti un significato diverso da quello a lui noto. Non c'è problema per tiny, small e medium: il primo consente di realizzare file COM invece che EXE, il secondo lascia un massimo di 64K sia per il codice che per i dati, il terzo consente fino a un mega per il codice accanto a 64K per i dati. I modelli compact e large, pur differendo tra loro per quanto riguarda il codice (limitato a 64K nel compact), offrono ambedue fino a un mega per i dati, ma solo per quelli allocati dinamicamente; i dati statici sono posti nei segmenti _DATA e _BSS (rispettivamente se inizializzati o no) i quali, raggruppati in DGROUP, sono limitati ad una dimensione complessiva di 64K (potremmo dire che i due modelli riproducono l'organizzazione della memoria tipica, rispettivamente, della versione 3 e delle versioni successive del Turbo Pascal). Il modello huge corrisponde invece al modello large della Microsoft, nel senso che vengono definiti tanti segmenti di dati (ognuno limitato alla dimensione massima di 64K) quanti sono i moduli del programma. Non consente però la definizione di singoli oggetti di dimensione superiore ai 64K. Per giungere a tanto è necessaria una innovazione documentata, come notavamo, solo nel file HELPME!.DOC: con il Turbo C++, a differenza di quanto si poteva con il Turbo C 2.0, è possibile usare le keyword **far** e **huge** anche per i dati oltre che per funzioni e puntatori; con ciò si ottiene sia di definire dati per oltre 64K complessivi in uno stesso modulo (ad esempio con: char far array1[60000L]; char far array2[60000L];), sia definire dati occupanti individualmente più di 64K (ad esempio: char huge array3[100000L];).

Tranne che per questo aspetto, può dirsi che il C del nuovo compilatore è sostanzialmente lo stesso del Turbo C 2.0 (provato nel numero di gennaio 1989, comprese le funzioni ANSI *signal()* e *raise()* o il tipo **long double**); il C++ è per parte sua un vero C++ 2.0, con poche inevitabili incertezze: la AT&T non ha precisato tutti i dettagli della implementazione degli *stream*, con la conseguenza che la Borland ha evidenziato nei file *fstream.h* e *iostream.h* quei punti per i quali potrebbero sorgere incompatibilità con altre implementazioni. Precisiamo infine che anche il C++ si può avvalere delle estensioni che la Borland ha da tempo proposto nella sua implementazione dell'ANSI C.

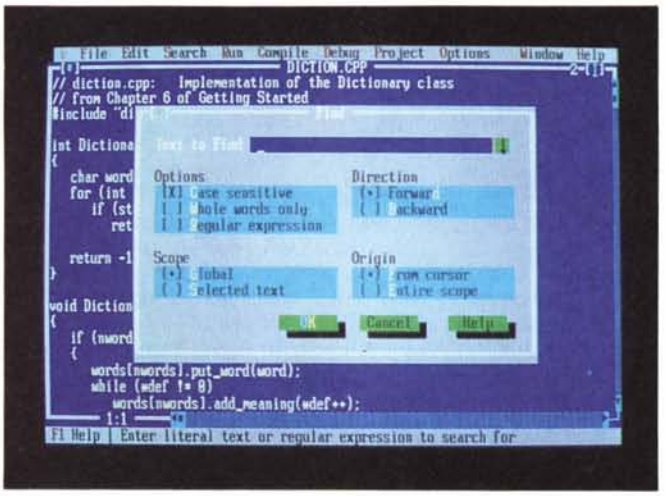
Librerie

Accanto alla tradizionale ricca libreria del C e all'ulteriore corredo del C++ (quale una classe *complex*, che consente di trattare i numeri complessi come gli altri tipi numerici predefiniti), la Borland offre sia i suoi tradizionali strumenti per la gestione del video nei modi testo e grafico, sia la classe *bcd* cui abbiamo già fatto cenno. Ma non è tutto.

Il C++ si propone come un «migliore C» grazie a numerose interessanti caratteristiche, usabili anche indipendentemente dalle «classi». La keyword **const** può rappresentare una migliore alternativa alla definizione di costanti manifeste mediante **#define**, in quanto è soggetta ai controlli sintattici. La possibilità di definire funzioni **inline** costituisce, per analoghi motivi, un progresso rispetto alle macro tradizionali. Il più rigido controllo dei tipi aiuta a scrivere codice più affidabile, mentre dalle possibilità di indicare valori di default per gli argomenti di una funzione o di definire funzioni con lo stesso nome ma con diversi argomenti discendono maggiore flessibilità e un codice spesso più chiaro. Analoghi i benefici delle *reference* che permettono di passare ad una funzione, per dirla in pascalese, parametri variabile oltre che parametri valore. Quando poi si passa alle classi, si sco-



Con l'opzione Transfer è possibile aggiungere al System menu un programma da eseguire senza uscire dall'ambiente integrato, reindirizzandone l'output in una Edit o Message window.



Le opzioni del menu Search, arricchite dalla possibilità di ricercare regular expressions, in una dialog box nuovo stile.

pre ben presto che l'incapsulamento di funzioni nella definizione dei dati consente non solo di realizzare una effettiva astrazione sui dati, ma anche di scrivere codice meno soggetto ad errori.

Qui però si scopre che la ricchezza e la potenza del C++ hanno un prezzo: la programmazione object-oriented stimola ma anche rende praticamente obbligatoria una più approfondita analisi. Esagerando un po' si può dire che, nella programmazione tradizionale, se serve un nuovo dato lo si aggiunge e basta; nella OOP, invece, si può e si dovrebbe sempre considerare la possibilità di fare di un oggetto l'istanza di una classe, e di individuare la collocazione di questa in una organica gerarchia di classi. Cosa è una «buona classe»? Stroustrup risponde: qualcosa cui attiene un insieme contenuto e ben definito di operazioni, qualcosa che può essere visto come una sorta di scatola nera manipolata esclusivamente mediante quelle operazioni, qualcosa di cui sarebbe desiderabile avere più esemplari.

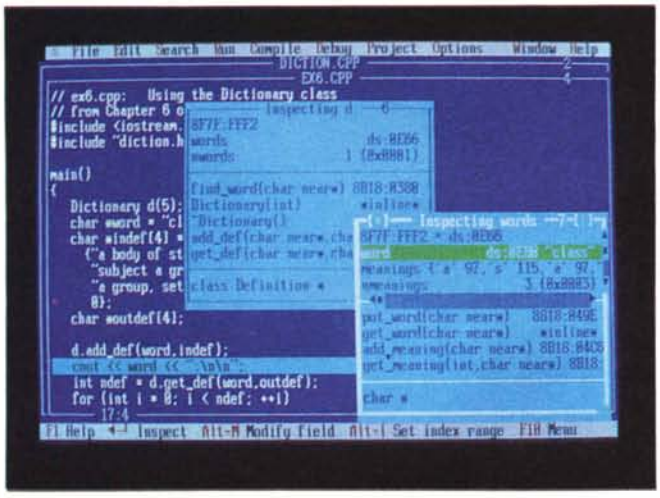
Proviamo a immaginare di realizzare una interfaccia utente. Potremmo pensare a una classe Window, da cui derivare classi per menu e dialog box. Dato che una window può avere i suoi menu, ogni menu i suoi sottomenu, attribuiamo ad ogni window un puntatore ad una subwindow, intendendo con ciò consentire di appendere ad ogni window una lista di subwindow. Quanto ci vuole perché ci accorgiamo che «lista» è un ottimo candidato per una «buona classe»? Ben poco. Ma così facendo si giunge ben presto a sentire il bisogno di una ricca gerarchia di classi, magari modellata su quella classica (e sperimentata) dello Smalltalk. Non stupisce, quindi, di trovare una libreria OOPS (Object-Oriented Program Support) scritta in C++ da Keith E. Gorlen e modellata proprio sulla gerarchia dello Smalltalk (di pubblico dominio ma, per quanto risulta a chi scrive, non ancora portata su MS-DOS), né di ritrovare una gerarchia più semplice ma di uguale ispirazione nell'Objective-C di Brad J. Cox, interessante mistura di C e Smalltalk. Non volendo o non potendo usare la libreria di Gorlen, non avendo voglia o tempo di tradurre in C++ la libreria di Cox, che fare?

La Borland ci propone una libreria TCLASS che, riprendendo alcune soluzioni di Cox, ci mette a disposizione una gerarchia di classi semplice e potente al tempo stesso; non ha l'estensione, forse eccessiva, di quella di Gorlen, ma appare ben costruita e facilmente estensibile. E naturalmente ci sono anche i sorgenti.

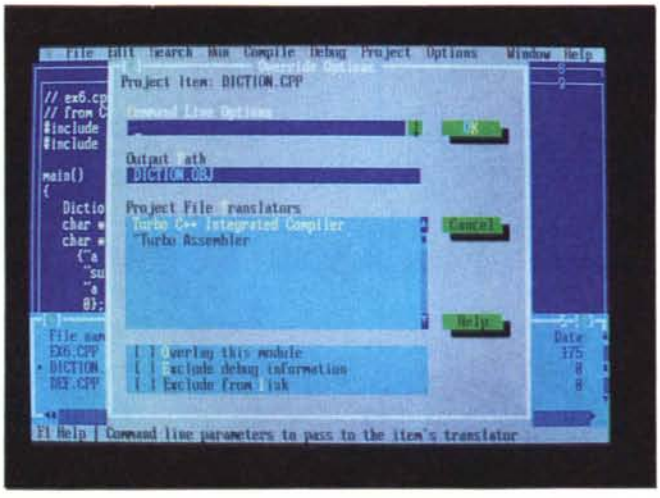
Ambiente di sviluppo

L'ambiente di sviluppo è stato notevolmente migliorato. Per prima cosa, abbiamo (finalmente) un multi-file editor: è possibile aprire diverse finestre di editing, sia per due o più viste su uno stesso file, sia per operare contemporaneamente su più file. È stato inoltre elevato da 64K a 8 mega (!) il limite per il buffer dell'editor, che può usare anche la memoria estesa o espansa eventualmente presente. Ma tutta l'interfaccia utente è cambiata, non solo per la possibilità di usare il mouse.

Nel menu principale compaiono diverse nuove opzioni. Con Alt-spazio si accede ad un *System menu* che consente tra l'altro di chiamare altri programmi. Non si tratta semplicemente di una scorciatoia rispetto al temporaneo ritorno al DOS, ma di una valida integrazione dei tool che l'utente predilige nell'ambiente integrato, in quanto è possibile far sì che l'output di qualsiasi programma richiamato dal menu compaia poi in una Edit window o Message window. *Edit*, da semplice comando, è diventato un menu con le opzioni di *cut and paste* tipiche delle interfacce che si lasciano usare con il mouse, mentre è del tutto nuovo il menu *Search* che, oltre alle usuali opzioni di ricerca e sostituzione (arricchite peraltro dalla possibilità di usare *regular expressions*), ne comprende anche altre per saltare ad un dato numero di riga del file nella finestra di editing attiva, o per ricercare una funzione anche in file diversi da questo ma a questo collegati (tale ultima opzione è abilitata solo durante il debugging). Nuovo anche il menu *Help*, comprendente le opzioni *Contents*, che propone una schermata analoga a quella iniziale del vecchio help, e *Index*, che



Il menu Debug comprende ora anche una opzione Inspect, fin qui riservata al Turbo Debugger.



Ora è possibile associare ad ogni file di un progetto opzioni specifiche, compresi i programmi necessari per la compilazione (tipicamente un Assembler oltre al compilatore integrato) e l'overlay del modulo.

propone tutte le parole per le quali è attivato l'help mediante ipertesto; quando si apre la relativa finestra, è possibile muoversi in essa sia con il cursore o con il mouse, sia digitando progressivamente le lettere della parola cercata, fino ad identificarla univocamente.

Sono cambiati sostanzialmente anche i menu *Project*, dedicato alla gestione dei progetti, e *Debug*, arricchito da una opzione di *Inspect* molto simile a quella fin qui riservata al Turbo Debugger (provato a febbraio dello scorso anno) e da *breakpoint* sia condizionati che incondizionati. Il menu *Window* dispone anche di una opzione *Register* che apre una finestra utile per osservare il contenuto dei registri del microprocessore quando, durante il debugging, si passa attraverso istruzioni Assembler.

Tutto il sistema di menu è molto comodo da usare, sia con la tastiera che con il mouse. In esso si integrano a meraviglia le *dialog box, dotate di input box* (alle quali è associata una *history list* immediatamente accessibile per ripescare scelte effettuate in precedenza: basta premere il tasto di freccia in basso o clickare con il mouse sulla freccia che appare accanto alla *box*), *list box* (nelle quali la selezione può pure essere operata con «digitazione incrementale» come nell'index dell'help), *check box* e *radio button*. Il risultato è una interfaccia utente che... si vorrebbe poter incorporare nei propri programmi.

Dietro un tale scenario si cela però anche una diversa filosofia di sviluppo di progetti complessi. Nel Turbo C 2.0 i file PRJ erano normali file ASCII, ora sono binari a causa della loro complessità. Non solo è possibile salvare in essi, e in associati file DSK, buona parte della configurazione dell'ambiente integrato (dalla disposizione delle finestre alle selezioni dei messaggi di avvertimento attivati), ma anche opzioni specifiche per ogni file; si va dalle opzioni di compilazione al path per il risultante file OBJ o EXE, da programmi scelti tra quelli aggiunti al *System menu* che possono essere necessari per portare a termine la compilazione (tipicamente un Assembler) alla opzione se destinare o meno il modulo all'overlay. Come se non bastasse, il file di progetto contiene informazioni relative a tutti i file inclusi da altri; se ci si posiziona su un nome di file nelle Project window, premendo la barra spaziatrice si apre una finestra con l'elenco di tutti i file inclusi; scegliendo poi uno di essi e quindi l'opzione *View* il file selezionato appare in una Edit window.

Un'ultima nota. Abbiamo menzionato la possibilità di destinare alcuni moduli di un programma all'overlay. Va sottolineato al riguardo non solo che sono ovviamente presenti le relative opzioni sia nell'ambiente integrato che nel compilatore separato, non solo che il TLINK è stato coerentemente aggiornato, ma soprattutto che la Borland ha messo a disposizione dei suoi utenti la stessa tecnologia VROOMM usata nel Quattro Pro e nel Turbo C++: ne segue che cadono alcune tradizionali restrizioni (quali l'impossibilità per una funzione presente in una unità di overlay di chiamare un'altra funzione in un'altra unità) e che ne guadagna l'efficienza del programma.

Conclusioni

Due compilatori in uno, per il linguaggio più diffuso in ambito professionale e per quello più promettente, una interfaccia utente praticamente perfetta, una intelligente gestione della RAM sia per il compilatore che per i programmi con esso compilati, una documentazione eccellente nonostante qualche omissione, un ricco corredo di strumenti di sviluppo e di librerie. Si può ben sopportare l'aumento di circa centomila lire rispetto al Turbo C 2.0. L'aumento è comunque temperato da una favorevole offerta di upgrade, valida per chi provenga da un *qualsiasi* linguaggio Borland.

Valutazioni monetarie a parte, il prodotto può essere preso in seria considerazione sia da chi voglia dotarsi di un ambiente di sviluppo più evoluto per la programmazione in «vecchio» o «nuovo» C, sia da chi voglia accostarsi al C++ traendo beneficio dalla compilazione diretta, senza traduzioni intermedie, nonché dal supporto offerto dalla manualistica e dai file su disco, dai semplici esempi al sorgente di una efficace gerarchia di classi.

MC