1995年4月1日（1日1善）監修・ボーランド
C/C++言語技術処理系
BCマガジン Borland

BC MAGAZINE

1995
APR. VOL.1 No.1

4.0J

特集|DOS,Windows,Windows NTの

アプリケーションを効率よく開発する

スクープ近日発売!!

Turbo C++4.0J for DOS
Borland PowerPack for DOS 16
✓DOS開発の救世主
✓発売間近の評価レポート

Borland C++4.0J総力レポート

●ObjectWindows 2.0、他社製品では実現できない裏ワザ・表ワザ全機能紹介 ●階層プロジェクトマネージャのパワー
●16ビットから32ビットアプリケーション開発まで、シームレスな開発を実現した統合開発環境 ●新技術への対応
●例外処理とRTTIによる信頼性の向上の仕組み ●究極のカスタマイズと拡張性が実現された内蔵エディタの真価
●VBXに対応したResource Workshopの威力 ●98&DOS/V両対応は、ボーランド開発陣のクラフトマンシップの証

ボーランド株式会社
〒151 東京都渋谷区笹塚1-64-8 笹塚サウスビル TEL.03-5350-9380 FAX.03-5350-9369

＊All Borland product names are trademarks of Borland International, Inc. ©Borland International, Inc.
＊その他、商品名は一般に各社の商標または登録商標です。

Borland U-FAX SERVICE
ボーランド製品に関する情報は、Borland U-FAX Serviceからどうぞ!
TEL:03-5350-9340 Borland C++4.0JのBOX番号は"2301"です。無料セミナーのBOX番号は"3101"です。
※プッシュ回線でご利用ください。

創刊の挨拶 | 開発者の生産性を追及する BC MAGAZINE

C言語市場もひところの多品種時代を過ぎ、アプリケーション開発の主流言語としてかなり落ち着いているように見える。しかし、生産性を向上させるための各社の競争は、以前にも増して激しくなっているのが現実である。

はたして、開発に必要なものはコンパイラの速度か統合開発環境の機能か、メーカーのマーケティング力か、氾濫する情報の中で処理系の選択に悩まされる開発者は少なくないだろう。

わがBC MAGAZINE誌は、主にC/C++言語を中心とし、アプリケーションを開発する際の生産性を真に高めてくれる処理系を紹介する目的で創刊された。本誌は、独自の調査により他誌には決して真似のできないレベルの記事を提供できていると自負している。

なお、本誌の性格上、製品の評価が独断的あるいは一方的であることは、あらかじめご了承いただきたい。

特集

DOS,Windows,Windows NT のアプリケーションを効率よく開発する

アプリケーション開発を効率よく開発するためには何が必要でしょうか。あなたがプログラミング初心者だと仮定して、これからDOSのアプリケーションを作成することになったとします。単にDOSがあるだけでは、何をどうやって開発してよいのか途方にくれてしまうに違いありません。そこで、アプリケーションを開発するには、まずコンパイラというものが必要です。どんなコンパイラを選べばよいでしょうか。初心者にとって重要なことは、最適化が充実していたりライブラリ関数がどれだけそろっているかということではないでしょう。むしろ、わかりやすいマニュアルと使いやすい開発ツールが含まれているかどうかが大切です。それに、アプリケーションを開発するための参考書も揃っていた方がよいでしょう。

あなたがプログラミングのプロであれば、コンパイラへの要求も変わってきます。たとえ、わかりやすいマニュアルがついていても最適化の機能が貧弱であったり、利用できるライブラリ関数が揃っていなかったり、開発に必要な機能が揃っていないようなコンパイラを選ぶことはないでしょう。

また、そもそも効率とは何でしょうか。とにかく安価なシステムが欲しいという人がいるかもしれません。お金に糸目はつけないから開発時間が節約できればよいという人がいるかもしれません。たとえ、100万円のシステムでも半年かかる仕事を2カ月で実現できれば十分もとがとれるでしょう。あるいは、とにかく信頼性の高いアプリケーションを開発したいという人もいるでしょう。アプリケーションの信頼性が低いと、せっかく仕事をしても、その後の保守作業に時間がかかって余計な経費がかかるかもしれません。

そういった意味では、アプリケーションを効率よく開発するというテーマはあまりに漠然としすぎているかもしれません。しかし、条件を限定していくつかの処理系を比べてみれば、どちらが優位であるかという参考にはなるはずです。

効率とは再利用性である - オブジェクト指向プログラミングの活用

たいていの場合、プログラムはただ作成して終わるというものではありません。プログラムを作成するためにプログラムした、というものでなければ何かの問題を解決するという目的を持っているわけです。時間がたてば、その目的も変わるでしょう。一度作成したプログラムは、新たな要求や環境のために修正したり、改良されることがしばしばあります。もしかすると、プログラムを改良したくないために、プログラムの目的も変更したがらないというケースもあるかもしれません。

C言語での古典的なプログラミングスタイルでは、プログラムを改良する場合はプログラム全体についての見直しやデバッグが必要なことが少なくありませんでした。ファイルに記録するためのデータ構造を変えると、そのデータ構造がプログラムのどこで使われているかを調べ直さなければなりませんでした。たとえ、誰も使わないように祈っていてもプログラム上は「そのデータを使うな」と指示できないので、可能性を否定することはできないのです。

オブジェクト指向プログラミングにおけるカプセル化は、こうしたプログラムを拡張するときの問題を解決するために役立ちます。何かの処理をあらわすのデータ構造をひとつの“クラス”にカプセル化しておけば、クラスの利用者が内部構造に立ち入ることを禁止できます。データ構造の実装方法を変更しても、外部とのやりとりが変わらなければ変更部分だけをデバッグするだけで済みます。クラスはオブジェクト(対象)を表現するための方法であり、オブジェクト指向プログラミングではクラスをどのように組み立てていくかが重要になります。

さらに、オブジェクト指向プログラミングには継承という大きな特長があります。継承とは、既存のクラスに手を加えず、新たな機能を追加したりいくつかのクラスを組み合わせて新たなクラスを作ることです。あるクラスが要求されていた機能をきちんと実現していれば、要求が追加された場合には、その新たな要求のためだけに元のクラスに機能を追加すればよいのです。既存のクラスに手を加えないということは非常に重要です。

C言語などで構造体を利用していると、新たな情報を追加するためには構造体そのものを変更しなければなりません。そうすると構造体を使っている部分すべてについて、デバッグや見直しが必要になります。別の構造体を作って、その中にもとの構造体をメンバーとして埋め込むこともできますが、そうすると新たな機能と元の機能は、同列には扱えなくなります。これは構造体を利用する側にとっては不便でしょう。

継承は、まさに再利用性を高める重要な機能です。オブジェクト指向プログラミングには多態性という機能もあり、クラスを拡張する際にもとのクラスの特定の機能を置き換えられます。

このように、オブジェクト指向プログラミングはプログラムの再利用性を高め、機能を拡張するときの手間を軽減する画期的な手段となるものです。また、オブジェクト指向プログラミング言語にはさまざまな種類があります。C++やSmalltalkをはじめ、ボーランドから発売されているTurbo Pascalもオブジェクト指向プログラミングに対応しています。しかし、本誌は、あくまでもC++言語専門誌なのでC++以外の処理系は考えないことにします。

効率アップのための処理系選択 - Editor's Choice "Borland C++ 4.0J"

本特集のテーマであるDOS、Windows、Windows NTに限定した効率のよい開発言語として本誌ではC++を勧めます。これは、本誌がC++専門誌であるという以外にも、WindowsやWindows NT自身がCで開発されていてC++との親和性が高いことが挙げられます。

C++言語に対応するコンパイラーにはさまざまな種類がありますが、単にC++に対応していれば何でもかまわないというわけにはいきません。C++のパワーを活かすには、コンパイラーがそれにみあう環境を提供できているかが重要な要素となります。そして、DOS、Windows、Windows NTの開発に最適と言えるのはBorland C++ 4.0Jでしょう。Borland C++ 4.0Jは、ひとつのパッケージでこれらのすべてに対応するアプリケーションを開発できます。たとえば、開発する対象をWindowsからWindows NTに移行しなければならないとしても、環境そのものを変更する必要がなく無駄な手間がかかりません。しかも、どの環境に対しても最新のコンパイラの機能を使えます。

DOSアプリケーションの開発に関しても、98とDOS/Vの両対応をはじめ、すぐれた機能を提供している他、オプションのBorland PowerPack for DOS16を使えば、プロテクトモードアプリケーションを開発したり、メニューやダイアログボックスを備えたユーザーインターフェースを構築するためのTurbo Visionというクラスライブラリが使えます。

ひとつのパッケージで、これらの環境に対応していると宣伝している処理系は他にもあります。しかし、16ビット用と32ビット用に2種類の開発システムをインストールしなければならなかったり16ビット対応のコンパイラでは最新の機能が使えないというものや、日本語化が不十分だったり32ビットリソースに対応していないなど実用性に乏しいものもあります。DOS、Windows、Windos NTのすべてに対応し、きちんと対応しているものはBorland C++ 4.0Jだけだと言ってよいでしょう。しかも、Borland C++ 4.0Jは豊富な機能をそろえながらも68.000円(税別、CD-ROM版)/74,000円(税別、FD版)というリーズナブルな価格で販売されています。

なお、開発の対象をWindowsに限定するならボーランドのTurbo C++ for Windows 3.1という選択もあります。これは、29,800円(税別)という入手しやすい価格設定でありながら、Windows上の統合開発環境やクラスライブラリ、リソースエディタ、スタンドアロンデバッガなどWindowsアプリケーションの開発に必要なすべてが揃っています。Borland C++ 4.0Jに比べても、必要なハードディスク容量が少なく、多少処理能力が劣るパソコンでも軽快な開発が実現できます。あるいは、Turbo Pascal for Windowsという選択肢もあります。Pascalは、より洗練された構文を持つ処理系で、Turbo Pascalはオブジェクト指向プログラミングに対応しています。

効率アップに不可欠なクラスライブラリの活用

オブジェクト指向プログラミングで重要になるのが、どんなクラスライブラリが提供されているかということでしょう。

オブジェクト指向プログラミングでは、アプリケーションの基礎から自分でプログラミングするのではなく、継承を使ってすでにあるアプリケーションのフレームワーク(骨格)に肉付けをしていく方法をとります。このため、フレームワークがしっかりしたものでなければ、肉付けする作業も面倒なものになってしまいます。また、クラスライブラリが十分な機能を提供していないのであれば、その部分をプログラマーが肉付けしなければならず、作業が増えることになります。

この意味でも、Borland C++ 4.0J のObjectWindows Library(OWL) 2.0が勧められます。OWL 2.0は、本誌でも特集していますが、他のクラスライブラリにはない数多くの特長を備えており、しかもオブジェクト指向プログラミングのパワーを存分に活かしています。

OWL 2.0では、スピードバーやステータスライン、ツールパレットなどWindowsアプリケーションの一般的なインターフェースをオブジェクト指向のスタイルで提供しているので、容易にアプリケーションを肉付けすることができ、自分自身の独創的な部分だけにプログラミングの力を集中できます。

効率とは機能である - 統合開発環境の活用

よいクラスライブラリを選択することは重要ですが、それを開発する環境が貧弱であれば、やはり生産性の低下をまねくことになります。特に、Windowsアプリケーションを開発する場合は、DOSのコマンドラインでコンパイル・リンクしてWindowsを起動してプログラムを実行する、というスタイルは効率のよいものではないでしょう。Windowsのアプリケーションは、やはりWindows上の開発環境で作成するものです。

この意味でもすぐれた操作性を持つBorland C++ 4.0Jの統合開発環境(IDE)が勧められます。Borland C++ 4.0JのIDEには、単にエディタ、コンパイラ、リンカ、デバッガの機能が組み込まれているだけではありません。機能が豊富なのはもちろんですが、全般にわたってカスタマイズ性が非常に高いのです。

エディタはキーをカスタマイズしたりキーボードマクロを定義できます。スピードバーもよく使うものだけを配置できる酔うカスタマイズできます。構文強調表示は配色を変更できるばかりでなく、どんな単語を強調表示するかも指定できるようになっています。

このカスタマイズ性の高さは「利用者がソフトウェアに合わせる」のではなく「ソフトウェアが利用者の使い方に合わせる」ことをあらわしており、非常にユーザーフレンドリーなソフトウェアだと言えるでしょう。

もちろん、エディタの機能をはじめとするIDEの機能も豊富です。特にコンパイル速度が速いことは、プログラマーの生産性を向上させる意味でも大きな役割を果たします。

まとめ

よりよいアプリケーション開発の近道は、やはりよりよい開発環境を選択することです。BC MAGAZINEは、もっともすぐれたDOS、Windows、Windows NT用の開発システムとしてBorland C++ 4.0Jを推薦します。

Borland C++ 4.0J 総力レポート

ボーランドが昨年9月に発売を開始したBorland C++ 4.0Jは、その豊富な機能とすぐれた操作性が各方面で高い評価を受けています。その一方で、あまりに機能が豊富なために、どんな特長があるのか把握しにくいとか、応用例が知りたいと言った声があるのも事実です。限られた紙面ですが、ここでBorland C++ 4.0Jの特長を紹介してみることにします。

ObjectWindows 2.0、他社製品では実現できない表ワザ・裏ワザ全機能紹介

ObjectWindows Library(OWL) 2.0は、Borland C++ 4.0Jに添付されている新しいWindows用のクラスライブラリです。Borland C++ 3.1以前では、OWL 1.0というバージョンがサポートされていましたが、OWL 2.0では1.0に比べて全体的な構成の見直しと機能強化がはかられています。もっとも大きな変更点は、C++の標準仕様を使って書き直されたことでしょう。OWL 2.0はマルチプラットフォームに配慮し、ボーランドの独自拡張に頼らないクラスライブラリとなっています。

OWL 2.0は他社のクラスライブラリに比べて多くの優れた点を持っています。紙面の関係上、簡単に列挙します(95.1現在)。

- ANSI C++対応の例外処理機能
- ANSI C++対応の文字列クラス
- レイアウトウインドウ
- ガジェットウインドウ
- オブジェクト指向で実現したツールバー
- オブジェクト指向で実現したステータスバー
- VBXのドラッグアンドドロップサポート
- 使いやすいツールパレットクラス
- 簡単なダイアログボックスでのデータ転送
- 使いやすい印刷クラス
- メタファイルのサポート
- その他

ここでは、ツールバーについて取り上げます。

これに対して、OWL 2.0ではツールバーのボタンはオブジェクトとして定義されています。ツールバーは、TControlBarという名前のクラスで表現します。ここに、ビットマップボタンのオブジェクトを追加することでツールバーを実現します。ボタンのイメージは独立したビットマップとして扱われています。次のツールバーを考えてみます。

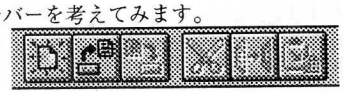

図. ObjectWindowsで作成したツールバーの例

OWL 2.0では、このツールバーのためのプログラムは次のようになります。

```
TControlBar *cb = new TControlBar(parent, direction);
cb->Insert(*new TButtonGadget(CM_MDIFILENEW,
                              CM_FILENEW));
cb->Insert(*new TButtonGadget(CM_MDIFILEOPEN,
                              CM_FILEOPEN));
cb->Insert(*new TButtonGadget(CM_FILESAVE,
                              CM_FILESAVE));
cb->Insert(*new TSeparateGadget(6));
cb->Insert(*new TButtonGadget(CM_EDITCUT,
                              CM_EDITCUT));
cb->Insert(*new TButtonGadget(CM_EDITCOPY,
                              CM_EDITCOPY));
cb->Insert(*new TButtonGadget(CM_EDITPASTE,
                              CM_EDITPASTE));
```

ビットマップは、それぞれのボタンごとに定義できます。このため、他のボタンに影響させる心配なくボタンの順序を変更したり、追加や削除することができます。

このように、OWL 2.0ではそれぞれのボタンをオブジェクトとして設計しているため、取り扱いが簡単です。また、それぞれのボタンはコントロールのように独立していますが、限られたウインドウハンドルを消費しないようガジェットウインドウという仕組みを用意しています。

他社製品では、ボタンのイメージをただひとつのビットマップにまとめて定義するようなオブジェクト指向プログラミングとはかけはなれた設計をしているものもあります。しかし、OWL 2.0では、ツールバーをはじめC++のパワーを十二分に活用しているため、利用しやすく安全なものになっています。

階層プロジェクトマネージャのパワー

Borland C++ 4.0Jのプロジェクトマネージャでは複数のターゲットを管理できます。実行ファイル(.EXE)やDLL、あるいはヘルプファイル(.HLP)など複数のターゲットが必要なアプリケーションでも、まとめて管理できるようになっています。

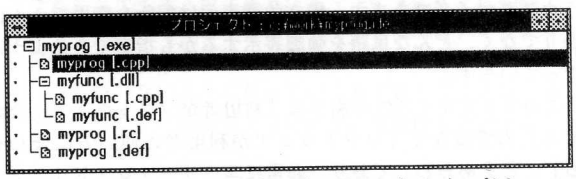

図. Borland C++ 4.0Jのプロジェクトウインドウ

さらに、階層構造を採用したことで、実行ファイルに必要なソースファイルやライブラリを登録した上で、ライブラリを構築するのに必要なソースファイルをそのライブラリのためのノードとして登録できます。そのソースコードが何かのソースコードジェネレータで生成されるものであれば、さらに元になるファイルをそのソースのノードとして登録できます。階層構造は、特定の部分をまとめて隠して表示したり、広げて表示することができるので、プロジェクトの管理も簡単です。

また、IDEにはDOSやWindowsのツールを組み込む機能があります。プロジェクト項目はそのツールをIDE自身の機能のように利用できます。たとえば、オプションのTurbo Assemblerを組み込んでおけば、プロジェクトに登録したアセンブリソースを自動的にアセンブルできるようになります。あるいは、.RTFの編集ツールとしてMS-Wordを組み込んでおけば、プロジェクトの.RTFファイル名をダブルクリックしただけで、MS-Wordを起動できるようにもなります。アセンブラなどのフィルタとして機能するものは、フィルタが出力する内容をIDEに取り込む仕組みは公開されているので、フィルタを自作してIDEの機能を拡張することもできます。

プロジェクトのオプションの設定は、使いやすいノートブック形式のダイアログボックスが使えます。さらに、プロジェクト全体とは別に個々のノードに対するローカルなオプションも設定できます。このときもノートブック形式のダイアログボックスを使って、全体のプロジェクトの設定を引き継ぐか、ローカルなオプションで設定を強制するかを選べます。

複雑で大規模なプロジェクトを効率よく管理するためには、Borland C++ 4.0Jの階層プロジェクトが最適でしょう。

16ビットから32ビットまで、シームレスな開発を実現した統合開発環境

プロジェクトマネージャには、TargetExpertという機能が組み込まれており、ターゲットを16ビットアプリケーションとしてコンパイルするか、32ビットアプリケーションとしてコンパイルするかを簡単に切り替えられます。それぞれの条件に必要なオプションは自動的に設定しなおされます。

また、Windows用のクラスライブラリOWL 2.0は、16ビットと32ビットの間でのソースコードの互換性を確保しているため、OWLの仕様で書かれたアプリケーションは16ビットと32ビットを自由に切り替えられます。このとき開発環境そのものを変更する必要はありません。

究極のカスタマイズと拡張性が実現された内蔵エディタの真価

IDEのエディタは、自分の好みにあわせてキー操作をカスタマイズできます。デフォルトでは、IDEはWindowsの標準的なキー操作CUA(Common User Access)をサポートしていますが、この他のキー操作としてDOS IDE、Brief、Epsilonなどのキー操作に合わせるボタンが用意されています。

さらに、KEYMAPRというユーティリティを使えば、好みの機能を適当なキー操作に割り当てられます。たとえば、"main(int argc, char *argv)"と入力し、改行し、"{"を入力し、改行し、タブを挿入する、という操作もひとつのキー(またはプレフィックスつきのキー操作)に割り当てられます。

VBXに対応したResource Workshopの威力

もともと、Resource Workshopは、すぐれた操作性を持つリソースエディタとして高い評価を受けていたものです。実行ファイルやDLL内にバインドされたリソースの編集、細かい調整機能がついたダイアログエディタ、カスタムコントロールのサポート、メニューやビットマップフォント、アクセラレータなどのすべてのリソースのビジュアルな編集や管理など、リソースの編集にかかわるすべての作業はResource Workshopだけで実現できます。

Borland C++ 4.0Jに付属しているResource Workshop 4.0は、新たにVBXのサポート、IDEとの連携、32ビットリソースの編集がサポートされています。VBXはMicrosoftのVisual Basic用のカスタムコントロールです。市販のVBXを利用できることで、アプリケーション開発の効率化がはかれます。なお、VBXはOWL 2.0のクラスでもサポートされています。

IDEとの連携とは、IDEのプロジェクトから特定のリソースを直接編集させたり、ClassExpertで作成したダイアログやメニューを編集するために簡単にResource Workshopを呼び出したり、逆にダイアログ編集中に特定のコントロールのための動作を定義するためにResource WorkshopからClassExpertを呼び出すことができるというものです。

32ビットリソースの編集とは、文字通り32ビットアプリケーションが使うリソースを編集できる機能です。32ビットアプリケーションを構築できる環境ならば当然のような機能ですが、32ビットリソースはUniCodeの取り扱いなど特殊な処理が必要になります。他の製品では、編集するときは16ビットリソースとしてしか編集できず、32ビットのリソースでは制約を受けるものもあります。また、DOSのコマンドラインで32ビットリソースがコンパイルできるのは、Borland C++ 4.0Jだけです。

98&DOS/V両対応は、ボーランド開発陣のクラフトマンシップの証

Borland C++ 4.0Jは、CD-ROM版でもFD版でも98とDOS/Vの両方の開発環境を提供しています。2種類の製品であれば、ユーザーが開発環境を変更したときにもう一本売れるはずですが、なぜ売り上げが下がるようなことをしているのでしょう。

この疑問に対して、ボーランドはこう回答しています。

「特にWindowsの場合、必ずしも開発環境が98かDOS/Vのどちらかに限定される必要はありません。開発環境を移行するときに余計なコストがかからない方が喜ばれるでしょう。

また、Borland C++ 4.0Jの98&DOS/V対応は、単に2種類の開発システムが入っているというだけではありません。特に機種依存性が問題にされるコンソール関数、グラフィックス関数、浮動小数演算例外については機能の共通化がはかられています。つまり、同じ実行ファイルをどちらの機種でも使えるように、自動判別機能が含まれているのです。

これは、オプション製品のPowerPack for DOS16のDOSエクステンダー機能やTurbo Visionでも同様です。Turbo Visionで作ったアプリケーションは、98のノーマルモード・ハイレゾモード、DOS/Vで動作するようになっています。H98の拡張アトリビュートにも対応していますし、非公式ですがDOS/Vの英語モードでも罫線などが化けないように工夫されています。」

特に、DOSアプリケーションの移植を求められているプログラマーには非常に嬉しい話ではないでしょうか。さらに、この機能の発想についてボーランドはこう回答しています。

「98とDOS/V共通化は、もともとBorland C++ 3.1対応のTurbo Visionから始まっています。オブジェクト指向で機能をカプセル化しているため、機種依存部分だけを変更すれば実現できるというのが発想のもとになりました。実際、両対応のためのコストはそれほどは大きくありません。

それに、この機能はオブジェクト指向を活かすという意味でより"ボーランドらしい"という技術陣の声があったのです。むしろ、マーケティング面は後からついていった感じですね。実際に、サイズ増加もTurbo Vision全体の数%に収まることがわかったので、両対応に踏み切りました。

あまり目立った宣伝をしていないせいか、この機能のことが知られていないケースがあるようですが、活用されている方にはかなり好評です。もっと、活用していただきたいですね。」

まさに、ユーザー本位の開発環境を提供してくれるボーランドここにあり、という印象を受けるような機能ではないでしょうか。今後も、このような製品を開発し続けてくれることに期待します。

*おわび 紙面の都合上、“新技術への対応”と“例外処理とRTTIによる信頼性向上の仕組み”は割愛させていただきました。

スクープ！！ Turbo C++ 4.0J for DOS / BorlandPowerPack for DOS16

本誌では、今春ボーランドから発売されるというTurbo C++ 4.0J for DOSとBorland PowerPack for DOS16について、他誌に先駆けて情報を入手することができました。とかく、Windows中心で開発環境が整備され、DOS開発への要求が見送られることが多い中、DOSアプリケーション開発の救世主ともなりそうなこれらの製品について主な機能をさぐってみます。

Turbo C++ 4.0J for DOS

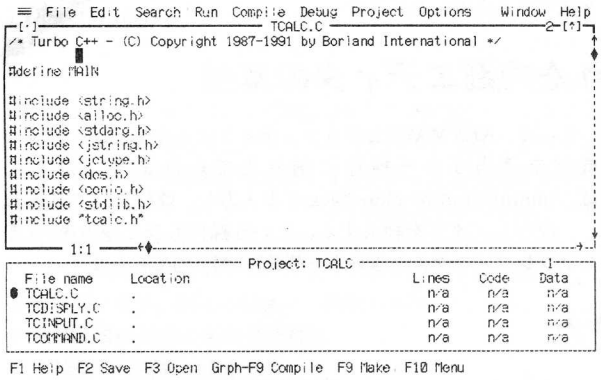

図. Turbo C++ 4.0Jの統合開発環境

Turbo C++ 4.0J for DOS(以下、Turbo C++ 4.0J)は、米ボーランドの製品系列にない日本独自の製品です。日本では、Turbo C++ 1.0というコンパイラが4年以上の長期に渡るベストセラー商品として発売されてきました。米ボーランドでは、すでに新バージョンのTurbo C++ 3.0が発売されていたのですが、結局この製品は日本語化されませんでした。ボーランド社によれば、この理由は「Turbo C++ 3.0は、プロテクトモードで動作します。しかし、DOS市場でもっとも影響力のあるNEC製のDOSで、EMM386がVCPIもDPMIもサポートしていなかったためエントリーユーザーの方を対象としているTurbo C++のユーザーの方の障害となりかねないためです。」(ボーランド言語プロダクトマネージャ談)とのことです。

しかし、NECはDOS 5.0AでDPMI.EXEというDPMIをサポートする常駐プログラムを提供するようになりました。DOS/VではもともとEMM386がVCPIをサポートしていますし、サードパーティ製のメモリマネージャでもVCPIやDPMIのサポートは当然の機能になっています。アプリケーション側でも機能拡張のためにプロテクトモードを使うようになっており、ボーランドから発売されているDOS版のParadoxやdBASE IVはプロテクトモードで動作するものになっています。

いま販売されているパソコンでは、2～3MBのメモリは当然のように実装されています。こうした状況で、Turbo C++もリアルモードに固執する必要がなくなったということでしょう。Turbo C++ 4.0Jは、プロテクトメモリを利用し、数多くの新しい機能を取り込んでいます。「Turbo C++ 3.0ではなく、独自に4.0Jというバージョンを実現したのは、DOSでも最新のC++の機能を利用していただくためです。C++の新しい機能は、アプリケーション開発に非常に役立つものです。」(同氏)

Turbo C++ 4.0Jの新たな特長を順に紹介していきましょう。

1. プロテクトモードでの動作

Turbo C++ 1.0ではDOSのリアルメモリだけを使っていたので、大きなプログラムをコンパイルするためにオーバーレイが多用されていました。Turbo C++のオーバーレイは、VROOMM(Virtual Runtime Object Oriented Memory Manager)という独自技術のもので拡張メモリにも対応していましたが、オーバーヘッドは避けられません。

これに対してTurbo C++ 4.0Jは、プロテクトモードで動作します。このため、リアルモードよりも飛躍的に大きなメモリを使え、大規模なプログラムも軽快にコンパイルできます。統合環境も、オーバーレイによる速度低下の心配なく使えます。

扱えるプログラムの規模やシンボル数が向上したことで、別売のBorland PowerPack for DOS16やTurbo Visionなどのクラスライブラリを使って、より快適な開発環境が利用できます。

2. 最新のC++に対応、コンパイラの強化

Turbo C++ 4.0Jは、既に発売されているBorland C++ 4.0Jと同様にテンプレート、例外処理、RTTIを含む最新のC++の仕様を取り入れています。これらの機能は、他社のプロ向けの製品ですらサポートしていないものもあります。

ボーランドは、Turbo CがANSI Cに積極的に対応してきたように、標準仕様への適合に力を入れてきた会社です。C++についてもANSI C++への適合を強化することで、将来性を確保した上で、より安全なプログラムの開発が実現されています。

コンパイラの強化はこれだけに留まりません。C/C++のプログラム中にかかれているアセンブリプログラムをコンパイラ自身が処理するための組み込みアセンブラ、386の32ビットレジスタを利用してlong整数演算を高速化する-3オプションなど各種の機能が強化されています。

3. 強化された統合開発環境、エディタ

Turbo C++ 4.0Jの統合開発環境(IDE)は、Borland C++ 3.1をベースにし、Borland C++ 4.0Jのコンパイルオプションなどを取り込んでいます。つまり、IDEそのものはすでに多くのBorland C++ 3.1ユーザーに使われている実績を持つものです。

これは、Turbo C++ 1.0に比べてもかなり機能が強化されています。たとえば、エディタは予約語や定数、文字列などを色分けして表示する構文強調表示をサポートしています。また、操作ミスを取り消すためのエディタのアンドゥ・リドゥがメモリの許す限り無制限にサポートされます。これにより、いったん全置換してしまった文字列を元に戻すこともできます。

もちろん、従来の使いやすいプロジェクトマネージャ、マルチファイルエディタ、IDEによるソースレベルデバッグ、充実したヘルプなどの機能はそのまま引き継がれています。

4. 98とDOS/Vへの両対応

Turbo C++ 4.0Jは、DOS/Vユーザーの方には特に歓迎されるかもしれません。これまでは、Turbo C++ for Windows 3.1というWindows向けの製品はあったのですが、DOS用のTurbo C++はなかったためです。Turbo C++ 4.0Jの登場で、DOS/VでもDOS用の安価なC/C++処理系が手に入るようになりました。

しかも、98版とDOS/V版はひとつのパッケージで提供されます。ボーランドのライセンスでは、同時に別のマシンにインストールするのでなければ、あるマシンから別のマシンに開発環境を移行することは問題ありません。つまり、今98を使っている場合、Turbo C++ 4.0Jを購入すれば、DOS/Vに移行した場合でも新たにDOS/V用のTurbo C++ 4.0Jを購入する必要はないわけです。

ライブラリ関数も、コンソール関数、グラフィックス関数、浮動小数例外について98とDOS/Vで共用できるようになっています。これは、単に2種類のライブラリが入っているというわけではありません。これらの機能は、単独の実行ファイルで98とDOS/Vの両方で実行できるというものなのです。

これは、Borland C++ 4.0Jでも実現されていたことですが(総力レポート参照)、このようなユーザーフレンドリーな発想が、またボーランドの魅力でもあります。

5. Turbo Debugger 4.0 の標準添付

Turbo C++ 4.0Jには、DOS用のスタンドアロンデバッガとしてTurbo Debugger 4.0が標準添付されています。Turbo Debuggerは、オブジェクト指向的なデバッグ手法を取り入れた使いやすいデバッガとして従来から定評のあるものです。

Turbo Debuggerを使えば、ソースレベルデバッグ、アセンブリレベルデバッグ、イベントログや式の実行も指定できるブレークポイント、C/C++式を評価できるウオッチなどさまざまな方法を使って、快適なデバッグが実現できます。しかも、Turbo Debugger 4.0は、DPMIに対応しているため、デバッガ自身が消費するリアルメモリはわずかです。

6. 充実したオプション

Turbo C++ 4.0Jでは、Borland PowerPack for DOS16やTurbo Assembler 4.0Jなどの充実したオプションがサポートされています。これらは、より機能が高く、優れたアプリケーションを開発するのに役立ちます。

このようにTurbo C++ 4.0Jは、Borland C++にも匹敵するパワーを備えるようになりました。価格は29,800円とリーズナブルですが、必ずしもエントリーユーザーのものだけでなく、DOSでの開発を続けているプロフェッショナルプログラマの方にとっても大いに役立つ製品と言えるでしょう。

Borland PowerPack for DOS16

Borland PowerPack for DOS16(以下PowerPack for DOS16)は、Borland C++ 4.0JやTurbo C++ 4.0Jを使って、DOSの16ビットに対応したプロテクトモードアプリケーションを開発するためのライブラリやツールを提供します。

もともと、米ボーランドでは英語版のBorland C++ 4.0用にBorland PowerPack for DOSという製品を発売していました。これは、16ビットと32ビットの両方のプロテクトモードアプリケーションを作成できるものです。日本で16ビットに限定した製品を発売することになったのは「16ビットに限定し価格を抑えることで、個人ユーザーの方でも利用しやすいものにしたかったのです。理論上は16MBまでのメモリを利用できますし、Windows形式のDLLも16ビットの方が使われるでしょう。」(ボーランド言語プロダクトマネージャ談)とのことです。32ビット対応版についての予定は検討中とのことです。

では、PowerPack for DOS16の特長を紹介しましょう。

1. **安価な 16 ビット DOS エクステンダー**

DOSエクステンダーとは、DOS上のプログラムをプロテクトモードで動作させるための技術で、これによって640KBのメモリの制約から解放され、メモリが実装されていれば理論的には16MBまでのプロテクトメモリをプログラムで利用できます。

しかも、プロテクトメモリをオーバーレイのバッファとしてではなく、プログラムのメインメモリとして使えますから、大規模なデータを扱う場合にも利用できます。特に、最近の機種では2～3MBのメモリが実装されているのは当たり前ですから、これがDOSで活用できるというのは嬉しいことです。

PowerPack for DOS16は、Borland C++ 4.0JやTurbo C++ 4.0J用のDOSエクステンダー製品で、各製品のユーザーに18,000円で販売されます。この18,000円という価格がいかに脅威的であるかは、従来のDOSエクステンダー製品の価格は10万～数10万になることを考えてもわかります。これによって、個人ユーザーでも十分にDOSエクステンダーのパワーを活用できるようになったといえるでしょう。

2. **作成したプログラムの配布が自由**

PowerPack for DOS16は、製品そのものが安価であることに加え、作成したアプリケーションを配布したり販売する場合でも、特別なライセンス料などはとられません。プログラムを配布するたびに、ライセンス料を徴収される製品がある中で、PowerPack for DOS16の姿勢は評価されるべきでしょう。

3. **ボーランド製品での実績**

PowerPack for DOS16は、これまでのボーランド製品になかった新製品です。とかく新たな製品というものは、いろいろな問題をかかえがちですが、PowerPack for DOS16はボーランドが自社製品の開発のために利用してきたDOSエクステンダー技術そのものが製品化されたものです。

このため、Borland C++ 2.0からのボーランド製品での実績をそのままユーザープログラムで利用できるわけです。機能より実績を重視する場合でも、安心して利用できます。

作成したアプリケーションは、VCPI/DPMIに自動的に対応するため、VCPI/DPMI対応の仮想86EMSなどとも共存できるようになります。

4. DOS 用のクラスライブラリ Turbo Vision 2.0

Turbo Visionは、ボーランドが早くから提供していたDOS上でキャラクタベースのユーザーインターフェースを提供するC++のクラスライブラリです。Turbo Visionの登場によって、メニュー、ステータスバー、マルチウインドウ、ダイアログボックスなどを備えたアプリケーションの開発が飛躍的に効率化されました。

PowerPack for DOS16には、診断機能などが強化されたTurbo Vision 2.0というバージョンが入っています。もちろん、このTurbo Vision 2.0はPowerPack for DOS16のDOSエクステンダー機能とともに使うことができるため、従来メモリが不足しがちで使いたくてもTurbo Visionを使いにくかった場合でも、活用できるようになるでしょう。

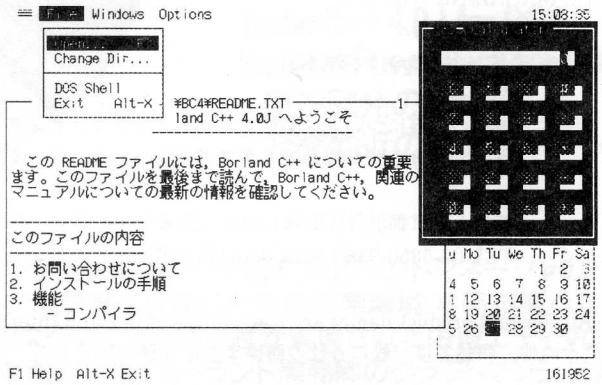

図. Turbo Visionを使ったプログラム例

5.　98とDOS/Vに両対応

Borland C++ 4.0JやTurbo C++ 4.0Jのライブラリ関数ができるだけ98とDOS/Vの共通化をはかっていることと同じように、PowerPack for DOS16も同一の実行ファイルで98とDOS/Vの両方に対応するよう設計されています。

これは、前述のTurbo Visionでも同様です。Turbo Visionの標準的な機能を使って開発したアプリケーションは、98とDOS/Vのどちらの機種でも実行できます。このことは、DOSエクステンダーやTurbo Visionを使った高度で使いやすいアプリケーションを開発する場合に、98とDOS/Vの両方に対応するためのコストがほとんど不要になるということです。つまり、ひとつのアプリケーションを開発する労力で、より多くの利用者が獲得できるということになります。

6.　プロテクトモード対応のBGI

BGIは、ボーランドの言語製品で使われている汎用性の高いグラフィックスインターフェースです。PowerPack for DOS16では、16ビットのプロテクトモードに対応したBGI(.BGX)がサポートされるため、プロテクトモードアプリケーションからグラフィックス関数が利用できます。

もちろん、コンソール関数も利用できます(BIOS関数はご利用いただけません)。

7.　Windows形式のDLLをサポート

PowerPack for DOS16の大きな特長のひとつに、Windows形式のDLLをサポートしていることが挙げられます。PowerPack for DOS16を使うと、複数の実行ファイル(.EXE)から共通に呼び出す機能を、実行時にリンクできるDLLで提供することができます。そして、PowerPack for DOS16を使って作成したDLLは、そのままWindows 3.x用のアプリケーションからも参照できます。

つまり、PowerPack for DOS16を使ってDLLを開発しておけば、開発対象がWindowsに移行した場合でも、そのままそのDLLを使えるわけです。これは、将来Windowsへの移行を検討している開発者の人には朗報でしょう。

8.　Turbo Debuggerを使ったデバッグ

機能はともかく、DOSエクステンダー機能を提供する製品は決して少なくありません。しかし、その多くが満足なデバッグ環境を提供していません。

PowerPack for DOS16では、Borland C++ 4.0JやTurbo C++ 4.0Jで使われているDOS用のTurbo Debugger 4.0Jを使って統合型のユーザーインターフェースを使った、ソースレベル/アセンブリレベルのデバッグを実現しています。

Turbo Debuggerは、豊富な機能とすぐれた操作性を兼ね備えたスタンドアロンの統合環境型デバッガで、アプリケーション開発においてもっとも時間のかかるデバッグ作業を効率化してくれるものとして高い評価を受けています。Turbo Debuggerのサポートによって、プロテクトモードアプリケーションの開発効率は飛躍的に改善されるでしょう。

Borland PowerPack for DOS16は、さまざまな目的で使われ続けているDOS環境でのプログラミングに、新しいパワーをそそぐことになるでしょう。

まとめ

WindowsやWindows NTなどの新しいOSが普及していくのとは対象的に、既存のDOSアプリケーションのメンテナンスや拡張、あるいは新規開発の要求は少なくありません。Turbo C++ 4.0JとPowerPack for DOS16は、こうしたDOS開発の要求を満たすものとして大いに活躍することでしょう。

また、どちらの製品も既存技術を組み合わせたり、社内でつちかわれてきた技術を利用して開発されたものです。このため、これらの製品は、"実績のある新製品"としてあたかも何年も前にリリースされた製品のように安心して製品を使うことができるでしょう。

休刊の挨拶 | 未来に向けて

残念ながら、本誌は今月号で休刊することとなった。短期間であったが、本誌を読み通していただいた読者の方には、ただ感謝あるのみである。あまりに短期間であったため、本誌が創刊の目的をどこまで達成できたかどうかは今後の評価を待たねばならない。しかし、本誌の意志が今後も読者の記憶の片隅にでも残ることとを願う。

また、どこかでお会いする機会があるかもしれないので、その時を楽しみにしてほしい。

BC MAGAZINE 4月1日号
平成7年4月1日発行
制作・著作　ボーランド株式会社

*注意　本誌の内容について、弊社インフォメーションセンターまたはテクニカルサポートでのお問い合わせはご遠慮ください。
本誌で紹介された製品のカタログについては、以下までご連絡ください。

〒151 東京都渋谷区笹塚1-64-8　笹塚サウスビル　ボーランド株式会社
TEL 03-5350-9380 FAX 03-5350-9369

All Borland product names are trademarks of Borland International, Inc.
*その他、商品名は一般に各社の商標または登録商標です。