1996年4月1日(毎年1回1日1膳ダイエット発行)

提供・ボーランド オブジェクト指向開発環境
Dマガジン Borland

D MAGAZINE

Delphi & Borland C++

1996
APR. VOL.2 No.1

April Fool

特長
オブジェクト指向と生産性の向上

探求
アルファベット進化論

- Assembler
- BASIC
- C/C++
- Delphi

英語版総力レポート

Delphi 2.0

●最適化が強化された32ビットネイティブコードコンパイラの威力 ●大幅に拡張されたObjectPascal
●Windows95コントロールと新データベースコンポーネント活用法 ●OCXとOLEオートメーションの効率的利用

Borland C++5.0

●ObjectScriptingによる究極のカスタマイズ ●ObjectWindows5.0:飛躍的な拡張と互換性
●ANSIC++最新機能 ●Visual Database Toolsによるデータベースアプリケーションのビジュアル開発

【D MAGAZINE SPECIAL】 Javaの話

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

誌名変更・復刊の挨拶｜開発者の生産性を追及するD MAGAZINE

定期購読の申し込み依頼すらあったにもかかわらず(実話)、創刊号で休刊となったBC MAGAZINEであるが、関係者のご厚意により誌名を変更して復刊することになった。復刊のためにご支援いただいた関係各位には、心より感謝する次第である。

さて、昨年度のコンピューター界の最大の話題といえばやはりWindows 95の出荷ということになるだろう。さらに、インターネットの急速な普及や、IBMによるロータスの買収、ごく最近ではノベルからコーレルにアプリケーション事業部が売却されるなど、かつてない激動の年だったと言える。

本誌の主題である開発ツールにおいても新星が登場した。いうまでもなくDelphiである。16ビットのビジュアルツールとしてはかなり後発だったが、その高い生産性が認められ今や開発ツールの主流のひとつになっている。

Delphiは、「ビジュアルツールで作成したものは遅くても仕方がない」という既成概念をくつがえす重要な製品となった。新誌名D MAGAZINEもこのエポックメイキングな製品の頭文字を拝借した。D Magazineには、Developers Magazineという意味もある。創刊以来貫かれている開発者の生産性を追及するという意味が込められているのである。

前号のEditor's Choiceに輝いたBorland C++も、バージョンアップが待ち構えている。こちらも単なる32ビット化にとどまらない大幅な機能アップが施されている。

なお、紙面が限られているのでそれぞれの特集がかなり限定的なものになっていることをあらかじめお断りしておく。また前号同様、製品の評価が独断的あるいは一方的であることは、あらかじめご了承願いたい。

[特集]オブジェクト指向と生産性の向上

OOP普及委員会

「生産性が上がるそうだが、何のことだかよくわからない」かつて、オブジェクト指向プログラミング(OOP: Object Oriented Programming)はこう思われていました。いや、今でもそう考えている方がいるかもしれません。特に手続き型プログラミング言語に慣れた方には、既存の知識がかえってOOPへの移行の邪魔になることがあります。また、OOPに過大な期待を抱いてしまい、いざ実践するときにとまどってしまう方もいらっしゃるでしょう。しかし、OOPはいまや開発の主流としてさまざまな場面で使われています。OOPの基礎と生産性について見直してみることにします。

OOPの基礎

OOPの要素にはカプセル化、継承、多態性(ポリモーフィズム)という特長があります。

カプセル化は、C++やObject Pascalでクラスとして実現されています。クラスは、OOPにおけるオブジェクトの表現です。プログラムの要素の材料(データ)と手順(コード)をひとまとめにし、あたかも小さいアプリケーションのように機能します。たいていの場合、外部とのやりとり(インターフェース)はメンバ関数とかメソッドといったクラスに依存する手続きによって実現されます。Borland C++のObjectWindowsやVisual C++のMFCをはじめ、さまざまなクラスライブラリがクラスを提供しているため、開発者はこれらのクラスを利用して自分がすべき負担を軽減できます。

また、VBXと呼ばれるVisual Basicコントロールもオブジェクトのひとつとみなしてよいでしょう。VBXには、プロパティとメソッドというインターフェースがあり、特定の機能がVBXの中に実装されています。Delphiのコンポーネントも同様です。実際、Delphiのコンポーネントは Object Pascalのクラスとして実装されているものです。

継承は、クラスに新しい機能を追加するためのものです。アプリケーションの機能を拡張するときに、既存のプログラムコードを複製して新たなデータ項目や機能を追加することはできます。しかし、直接既存のプログラムコードを変更すると、従来のプログラムと互換性がなくなってしまうばかりか、デバッグ作業もすべてやり直しになってしまいます。継承を使えば、必要な部分だけを新たに追加して、既存の部分については従来のプログラムコードをそのまま利用できます。このため、デバッグも追加した部分ですみます。

誤解のないように付け加えておくと、継承を使うからといって既存のプログラムコードを変更してはいけないわけではありません。既存のプログラムコードに拡張性が欠けていたり、修正した方がよいという十分な理由があるなら、修正をためらうことはありません。そうして試行錯誤しながら、よりよいクラスライブラリができあがるのです。

継承は、OOPの最大の特長と言ってもよいでしょう。継承こそが、既存のプログラムコードの再利用性を高め、生産性を上げる基礎になります。ビジュアルツールの中には、VBXやOCXを使ったオブジェクトベースの開発はできても、きちんとしたOOP言語を採用していないオブジェクト指向の開発ができないものがあります。たとえば、Delphi は Object PascalというOOP言語を採用しているため、こうした能力が十分に活かされています。

多態性(ポリモーフィズム)は、動的バインディングとしてとらえられることもあります。動的とは実行時という意味で、つまり実行時に呼び出すメンバ関数が決められるというものです。これは継承に密接に関係し、継承したクラスで上位のクラスの機能を置き換えるものです。

なぜOOPで生産性があがるのか

ちょっと乱暴な言い方をすると「OOP言語を使えば、今日から生産性があがるのだ」という発想は間違いです。「いや、Delphiを買ってきたら、その日からいろんなことに利用できたよ」という話はあるでしょう。でも、それはDelphiがたくさんのコンポーネントをあらかじめ持っているからです。

コンピュータから離れて家計簿をつけることを考えます。とりあえず、今日の分の入金・出金をメモしておくために、チラシの裏を使うことにしました。しかし、毎日チラシの裏にメモしておくだけでは、1週間、1カ月といった単位で集計するのに不便です。そこで、罫線が引かれたノートを使うことにしまし

た。1日1ページに項目ごとに記入しておけば、集計も楽になります。しかし、項目を並べて書いているだけでは後から分類するのが大変です。そこで、項目を記入するたびに分類名も記入できる欄が入ったノートを使うことにしました・・・

ちょっと強引ですが、あなたはノートを買いに行くのが面倒でチラシの裏を使うようなプログラムを書いていないでしょうか。プログラムがその場限りのものなら、チラシの裏でもかまいません。しかし、そのプログラムを後の資産として活かしたいのであれば、よりまともな方法をとるべきです。

OOPの話に戻ります。家計簿を付けるためにノートを買うのは、プログラムを開発するためにコンポーネントを使うようなものです。チラシの裏に定規で罫線を引くように、自分で最初からプログラムを書いてもよいのですが、あるものを使った方が便利でしょう。コンポーネントは便利なのですが、新たに分類を付け足したくなったらどうすればよいでしょう。

単なるコンポーネントベースの開発ツールの限界はここにあります。もちろん、分類付きの別のコンポーネントを探すこともできるでしょう。でも、それが今まで使っていたものと互換性がなければ、すべてやりなおしです。

OOPをきちんと実現した環境であれば、たとえ自分の望むコンポーネントがなくても、既存のコンポーネントから新しいコンポーネントを作成できます。当然、既存のコンポーネントが持っているすべての要素が継承されますから、追加した部分についてだけ互換性について配慮しておけばよいのです。

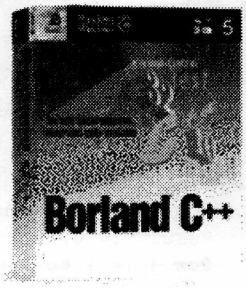
もし、何ひとつコンポーネントがなくても、コンポーネントを作る環境さえそろっていれば、時間はかかっても試行錯誤しながらでも必要なコンポーネント組み立て、階層化していくことができるでしょう。アプリケーションがOOPで開発されていれば、継承が使えますから機能も容易に拡張できます。これは、単なるコンポーネントベースの組み合わせでは、絶対にまねのできないことです。最初のうちはコンポーネントを組み合わせて簡単にプログラムができるかもしれません。しかし、顧客が要求する改良をコンポーネントが満たさなければ、要求を満たすコンポーネントを探して、最初から設計をやりなおすか、プログラミングでむりやりなんとかするか、それとも「これは、処理系の限界ですからねぇ」と顧客を納得させるしかないのです。長期的にプログラムを改良し続けるつもりなら、今すぐOOP環境を手にいれるべきです。

また、あなたが仕事を依頼するのであればOOP環境を使っている人に依頼すべきです。OOP環境を使いこなしている開発者なら、アプリケーションは長期的な改良にも耐えうるものになるからです。

クラスライブラリの重要性

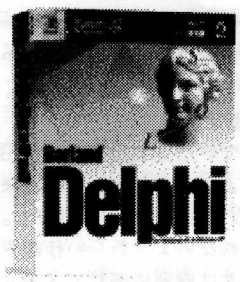
さて、前言を翻すようですが、OOPのコンポーネント、つまりクラスの品質が高ければ、OOPの価値も高まります。なぜなら、OOPはクラスを拡張する形でアプリケーションを構築していくため、OOPの品質がそのままアプリケーションの品質に反映されるからです。

クラスは、開発者のすべき仕事を肩代わりしてくれるものだと言えます。古典的なライブラリ関数群と違い、開発者はクラスの能力を活かしたまま継承によって機能を拡張できます。それだけにクラスの設計がまずければ、設計を見直さない限り継承したアプリケーションの品質が向上することはありません。



もちろん、クラスライブラリが豊富な機能を持っているかどうかということも重要です。クラスが開発者の仕事を肩代わりするものならば、いろんな仕事を肩代わりしてくれる方が生産性を高めるということに議論の余地はありません

Editor's Choice - Delphi 2.0 & Borland C++ 5.0



以上のことを考察すれば、本誌が勧める開発ツールに選択の余地はありません。ボーランド製のDelphi 2.0とBorland C++ 5.0は、いずれも自分自身を作るために開発した高い品質のクラスを豊富に提供しています。

Delphi 2.0は、ユーザーインターフェースやデータベースアクセスのための豊富なコンポーネントを提供しています。また、ビジュアルなフォームの継承やオブジェクトリポジトリなど既存の資産を活かす仕組みが整っています。

Borland C++ 5.0は、C++の多重継承を活かしたクラス階層を持ち、Doc/Viewなどのプログラミング指向のクラスやWindows APIをカプセル化しただけでない独特のクラス、さらにANSI C++の標準C++ライブラリを提供しています。

いずれの製品も、完全なOOP言語を採用しており、前述したOOPの特長を活かして生産性を高めることができます。それぞれの製品の特長については、次ページ以降をご覧ください。

[D MAGAZINE SPECIAL] Javaの話

Java Coffee 普及委員会

インドネシアの首都ジャカルタ。バタビアと呼ばれた1945年までのオランダ統治から独立し、人口は800万強に急増。この大都市は、大スンダ列島に属するジャワ島北西部にある。ジャワはコーヒーの名として知られているが、石油・錫などの鉱物資源にも富む12.6万平方キロメートルの島である。仏教、ヒンズー教、イスラム教の他、ヨーロッパ文化が伝来し人口密度も高い。

ジャワは英語でJava。日本では大塚ビバレッジの清涼飲料JAVA TEAにもあるとおりジャワ（爪哇）と読まれるが、英語読みではジャバとなる。ちなみにジャワ島の人はJavanese。しゃべる言葉もJavanese。一字違いで大違いといった趣もあり、なにやら親近感を覚えてしまうのはきっと気のせいだろう。

Javaneseは、古代にはサンスクリット語、中世にはアラビア語・オランダ語の影響を受け、相手の階級に応じて言葉を使い分けるのだそうだ。まるで日常語と敬語を使い分ける某国のようではないか。ついでに、ジャワでとれる米はJavanicaといい、粘り気のあるJaponica米と似ているそうだ。(8ページ先頭へ)

英語版総力レポートPart1 Delphi 2.0

Visual Programming普及委員会

昨年3月に英語版が、9月には日本語版が出荷され、ユニークな開発ツールとして注目を集めたDelphiの32ビット版が発売されました。ページ数は限られていますが、開発のすばやさと作成したアプリケーションの高速性を両立したDelphiの最新版では、何が改良されているのか、どんな性能の向上が見られるのか紹介します。

最適化が強化された32ビットネイティブコードコンパイラの威力

80386が登場して8年以上になりますが、Pentium全盛の今、Windows 95の登場でようやく32ビットの真価を発揮する場ができました。たしかに、Windows 95の中には大量の16ビットコードが残っており、OSとしても16ビットからくる制約が残っています。しかし、32ビットネイティブなアプリケーションがまともに実行できるという意味は非常に大きいでしょう。そこで問題になるのが、32ビット開発ツールと言われるものがきちんとCPUやOSの能力を活かしているかということです。残念ながら、開発ツールの中には16ビットアプリケーションを単に32ビット化したものや、OSの機能をすべて利用していないものもあります。こうした開発ツールでは高速で利用価値の高いアプリケーションは作成できません。

Delphi 2.0には、32ビットCPUを活かした最適化を取り入れ、OSの機能をすべて活かせるコンパイラが搭載されています。まず、簡単なパフォーマンスチェックを見てみましょう。

次のプログラムは、加算を200000000回繰り返すだけの単純なプログラムです。

```
function Sample: Longint;
var
  i, j: Longint;
begin
  for i := 1 to 20000 do begin
    Result := 0;
    for j := 1 to 10000 do
      Result := Result + j;
  end;
end;
```

Delphi 2.0は、わずか3秒でこれを実行します(Pentium/133MHz)。ちなみに、16ビット版Delphi 1.0での実行速度は27秒でした。同じ環境で中間コードを使う別の32ビット開発ツールで同様のプログラムを実行させたところ587秒かかりました。実に200倍近い実行速度の差があることになります。

Delphiはオーバーフローのチェックをしていないから速いんじゃないか、と思われる方がいらっしゃるかもしれません。しかし、Delphiでオーバーフローチェックのオプションを付けても、実行速度にほとんど違いはありません。

これはDelphiのビジュアルプログラミングを活かしたものではありませんから極端な例かもしれませんが、逆にネイティブコンパイラと中間コードコンパイラの違いが端的にあらわれていると言ってよいでしょう。Delphiに移行して中間コードを使うツールには戻れないという人が多いのも当然なのです。

もちろん、Delphi 2.0のコンパイラは単に高速というだけではありません。Windows 95やWindows NTで提供されるマルチスレッドなどすべてのWindows APIを利用できます。APIを利用するためにいちいちツールを呼び出す必要もありません。OLEサーバーに限らず他の開発ツールと共有できるDLLも作成できますし、必要なら独自のコンポーネントも作成できます。

もし、あなたが仕事を依頼する側の立場で、納入されたプログラムの速度が遅いと感じるようなら、開発に使ったツールが中間コードを使っているものかどうか聞いてみるとよいでしょう。もしかすると、開発者は貴重な時間とコストを遅いアプリケーションを作成するために費やしているのかもしれません。

大幅に拡張されたObject Pascal

Delphiのコンパイラは、単に作成したアプリケーションを高速に実行できるだけではありません。プログラミングの利用範囲をさらに広げるために、Object Pascalにも拡張が施されています。紙面の都合で構文を基礎から説明する余裕はありませんが、新たに拡張された構文について紹介します。

まず、バリアント型がサポートされました。バリアント型は、整数型、浮動小数型、文字列型、論理型、日付時間型に加え、OLEオートメーションオブジェクトも保持できます。さらに、任意の型に対する任意の大きさや次元の配列さえもバリアント型で保持できるようになっています。通常、配列の要素はあらかじめ範囲を決めておかなければなりませんが、バリアント型を使えば、実行時に配列の大きさを決めることができます。次のプログラム例は、バリアント型の任意サイズの配列をひとつのバリアント型変数に代入するプログラム例です。

```
var
  A: Variant;
  I: Integer;
begin
  A := VarArrayCreate([1,3,0,9], varVariant);
  for I := 0 to 9 do A[1,I] := I;
  for I := 0 to 9 do A[2,I] := Sqrt(I);
  for I := 0 to 9 do A[3,I] := IntToStr(I);
end;
```

他の処理系と異なり、Object Pascalは常に変数を宣言してから使うため、意図せずに変数がバリアント型になることはなく型保障もされるため安全にバリアント型を利用できます。

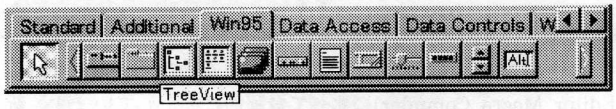
また、従来255文字までに制約されていた文字列型は、2GB（理論値）までの長い文字列を扱えるようになりました。従来との互換性のためにShortStringという型もサポートされます。長い文字列では、メモリが無駄遣いされないよう新しい仕組みがサポートされています。文字列データは内容に応じてメモリが確保され、文字列はその領域へのポインタになります。文字列はPCharに変換することなく、そのままWindows APIで利用できます。また、単なる代入だけでは内容を転送せず、参照カウントを使って同じ領域を共有します（文字列が変更されると新しい領域が確保されます）。文字列処理は、場合によって非常に高速かつ効率的に処理できるようになります。さらに、Unicodeを扱うためのWideChar型やWideString型もサポートされています（※注 UnicodeはシフトJISではありません）。

また、財務計算などで誤差が生じないようにするための通貨型(Currency)がサポートされました。

その他、Integerの16ビットから32ビットへの拡張にともなうSmallintの採用や、DLLなどで他の処理系とのインターフェースをとるためのstdcall修飾子などさまざまな拡張が、広範囲のプログラミングに対応します。

Windows 95 コントロールと新データベースコンポーネント活用法

Delphi 2.0 は、当然ながら Windows 95 の新しいコントロールにも対応しています。Delphi 2.0 のコンポーネントパレットの Win95 ページには、Windows95 の新しいコントロールに対応する 12 種類のコンポーネントが登録されています。



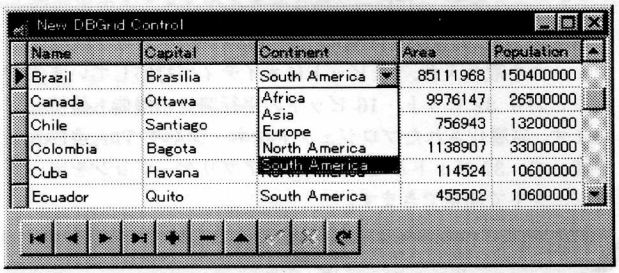
コンポーネントの名前と目的は次の通りです。

- TabControl - TabSet と同様のタブセット
- PageControl - マルチページダイアログのためのページ
- TreeView - オブジェクトリストをアウトラインで表示
- ListView - リストをカラム表示
- ImageList - 一連のグラフィックイメージを管理する
- HeaderControl - Header と同様のヘッダー
- RichEdit - 書式付きで編集できるエディットコントロール
- StatusBar - 動作状態を示すコントロール
- TrackBar - 段階付きでスライドできるコントロール
- ProgressBar - Gauge と同様に進行状況を表示する
- UpDown - 値の増減を制御する矢印付きボタン
- HotKey - コンポーネントへホットキーを割り当てる

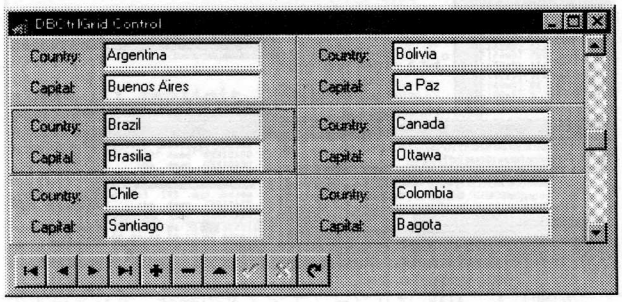
これらを使って、Windows 95 のエクスプローラのようなスタイルのユーザーインターフェースを容易に作成できます。

従来の TabSet や Outline などのコンポーネントもサポートされているので、これらを使えば Delphi 1.0 の 16 ビットアプリケーションと Delphi 2.0 の 32 ビットアプリケーションとの間でソースコードの互換性を保てます。

Delphi 2.0 では、データベース対応のコンポーネントも改良されています。データを表形式で表示する DBGrid コンポーネントはフィールドごとに項目リストを登録しておき、そのフィールドを編集するときに一覧を表示できます。また、[...]ボタンを付けて、独自の方法で編集させることもできます。



さらに、Delphi Developer 2.0 以上では、連続した領域に複数のレコードを表示できる DBCtrlGrid という新しいコンポーネントがサポートされます。DBCtrlGrid を使えば、好きなスタイルで複数のレコードを繰り返し表示できます。



OCX と OLE オートメーションの効率的利用

従来、Delphi 1.0 が VBX をコンポーネントとして登録できたように、Delphi 2.0 では OCX をコンポーネントとして登録できます。市販の OCX を利用して、Delphi 2.0 でのアプリケーション開発の生産性を一層高めることができます。

OCX を登録すると、自動的に Object Pascal のコードが生成され、Delphi 自身のコンポーネントと同様に OCX を使うことができます。つまり Object Pascal の継承を使って既存の OCX に新しいプロパティやメソッドを追加できるのです。

Delphi 2.0 では OLE オートメーションのサーバーとコントローラーの両方を作成できます。Delphi 2.0 で OLE オートメーションサーバーを作成するのは、コンポーネントを作成することに似ています。コンポーネントエキスパートに似たオートメーションオブジェクトエキスパートというツールを使うと、自動的に GUID などのレジストレーション情報を含み、TAutoObject を基本クラスとする新しいオートメーションオブジェクトクラスを定義したソースコードが生成されます。そして、次のように新しく追加された予約語 automated の後ろにプロパティやメソッドを定義します。

```
TCountApp = class(TAutoObject)
private
  FCount: Integer;
  function GetCount: Integer;
automated
  procedure Increment;
  property Count: Integer read FCount write GetCount;
end;
```

OLE オートメーションの制御は、さらに容易です。Delphi 2.0 では、OLE オートメーションオブジェクトを CreateOLEObject という関数で生成し、バリアント型の変数に代入します。次のプログラムは、MS-Word のオブジェクトを制御する例です。

```
var
  MSWord: Variant;
begin
  MSWord := CreateOleObject('Word.Basic');
  MSWord.FileNew('Sample');
  MSWord.Insert('Hello, OLE Object!');
  MSWord.FileSaveAs('C:¥TEST.DOC',,,'Borland');
```

まとめ

Delphi 2.0 は、この他にもフォームの継承やデータベースエクスプローラなど、ユニークな特長を数多く実現しています。

しかし、最大の特長といえるものは、やはり高速な実行を実現するネイティブコードコンパイラです。作成したものを C コードに展開してコンパイルできるようなツールもあるようですが、Delphi 2.0 では開発時点から最終版を作成するのと同じコンパイラを使えるのです。いわゆるトランスレータ形式が不便で好まれないことは、C++処理系の歴史でもあります。

Delphi 2.0 の生産性の高さをいますぐ活用することが、激しい競争に勝ち抜く最良の手段だと言えるかもしれません。

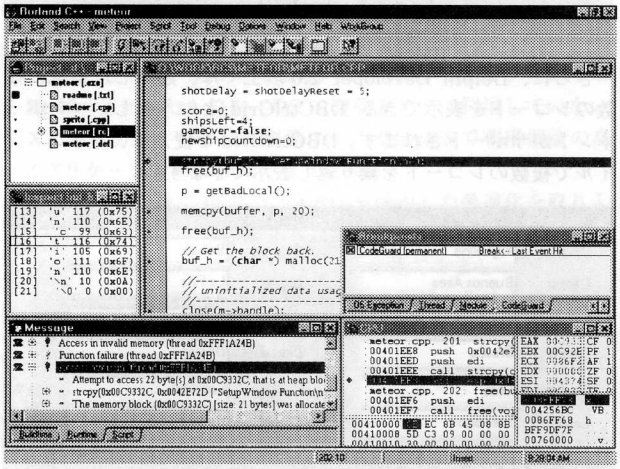
英語版総力レポートPart2 Borland C++ 5.0

C++ 普及委員会

他の製品に先駆けて 16 ビット・32 ビットの平行開発を実現していた Borland C++ですが、これまで統合開発環境は 16 ビット環境のままでした。新たに発売される Borland C++ 5.0 では、たんなる開発環境の 32 ビット化や Windows 95 対応にとどまらず、さまざまな機能の拡張が盛り込まれています。早速、この新製品の概要について紹介します。

ObjectScripting による究極のカスタマイズ

新しい IDE(統合開発環境)は、Windows 95 ユーザーインターフェースを取り入れ、32 ビットにネイティブ対応していますが、従来どおり 32 ビット・16 ビットの平行開発の機能も維持しています。階層化されたプロジェクトマネージャや TargetExpert を使って、32 ビットと 16 ビットのアプリケーションを同一プロジェクトで開発できます。



この他にも、IDE はリソースエディタの統合、モジュールのアタッチやマルチスレッド対応を含むデバッガの強化、バックグラウンドコンパイルなど、さまざまな改良が施されています。

中でも IDE のカスタマイズ機能は、従来の TEMC(Turbo Editor Macro Compiler)に比べて飛躍的に向上しています。新しい IDE では、カスタマイズ用に ObjectScripting という機能を搭載しています。ObjectScripting では、組み込み済みのクラスや C++に似た cScript 言語を使い IDE を自由に制御できます。次は、ObjectScripting を使った簡単なプログラム例です。

```
import IDE;
hello()
{
  IDE.Message("Hello, World!");
}
```

IDE モジュールには、IDEApplication クラス型の IDE というオブジェクトが含まれており、ここでは Message メソッドを使っています。IDE の[Script|Run]メニューで hello()を実行すると、メッセージが表示されます。IDE を終了させたいのなら IDE.Quit();とするだけです。IDE のキーボードマネージャ(KeyboardManager)を使えば、こうした処理を適当なキーに割り当てることができます。

ObjectScipting では IDE や Editor、Debugger などの IDE 要素が定義されています。これらのオブジェクトを使って Borland C++自身を直接操作し、自分の好きなようにカスタマイズできるのです。また、cScript で外部の DLL を呼び出したり、OLE オートメーションを使って IDE を制御することもできます。この自由度は、他の開発環境では得られないでしょう。

ObjectWindows: 飛躍的な拡張と互換性

Borland C++に添付される GUI クラスライブラリ ObjectWindows(OWL)が単に Windows API をグループ化しただけのものではなく、OOP や C++言語の機能を活かしているかについて、前号で説明しました。今回、この OWL がバージョン 5.0 となり、大幅に機能強化されました。

OWL 5.0 では、Windows 95 コントロール対応のクラス、スプラッシュ画面、ドッキングツールバー、アニメーション、分割ウィンドウ、WinSock や WinG のカプセル化、VBX/OCX のサポートなど数多くの新しいクラスが追加されています。もちろん、レイアウトウインドウや DIB クラスなど、以前からあるユニークなクラスも引き継がれています。

特に重要なことは、32 ビット専用と思われている Windows 95 スタイルのコントロールや 16 ビット専用と思われている VBX などが 32 ビットと 16 ビットの両方で利用できるように設計されているということです。Borland C++は、バージョン 4.0 で 32 ビットに対応するときもプログラマに急激な 32 ビット化を要求するのではなく、16 ビットから 32 ビットへのスムーズな移行環境を提供してきました。この思想は、そのまま OWL 5.0 にも引き継がれています。対応する VBX のレベルも 1~3 まで拡大されました。

リッチエディットやアニメーションなど一部のコントロールを除けば、ページタブやツールチップ付きのコントロールバーを使ったアプリケーションでも 32 ビットと 16 ビットでソースコードを共有できるのです。また、VBX を使っているアプリケーションを 32 ビットに移行する際に、対応する OCX が存在しなくても、当面は VBX を 32 ビットで利用し、OCX がサポートされた時点で置き換えるということができます。

IDE に組み込まれた AppExpert を使えば、容易に OWL アプリケーションのスケルトンを生成でき、ClassExpert でのカスタマイズができます。TargetExpert を使えば、簡単な操作で 32 ビットと 16 ビットアプリケーションの切り替えることができます。32 ビットのアプリケーション開発に移行したからといって、16 ビットを完全に見捨ててしまう必要はないのです。

OWL 5.0 は、OWL 1.0 から 2.0 のような大規模な構造の変更はありません。また、OLE2 プログラミングを効率化する ObjectComponents Framework(OCF)やコンテナクラスライブラリ(BIDS)なども従来どおりサポートされています。このため、既存の OWL 2.0 プログラムは、ほとんど変更することなく OWL 5.0 に移行できます。なお、Borland C++ 5.0 には、MFC を使うためのパッチファイルも含まれているため、MFC を Borland C++ 5.0 で再構築することもできます。

OWL 5.0 のサンプルプログラムは、EXAMPLES¥OWL ディレクトリにも数多く含まれています。しかし、OWL 5.0 の実用性を示す最も有意義な例は Borland C++ 5.0 自身でしょう。Borland C++ 5.0 の IDE も OWL 5.0 を使って開発されたものなのです。

ANSI C++最新機能

標準化に積極的に対応してきたBorland C++らしく、Borland C++ 5.0は、最新のANSI C++に対応しています。従来の例外処理や実行時型情報はもちろん、新たにSTLを含む標準C++ライブラリ、名前空間やbool、mutable、explicitなどの予約語やマクロなどが提供されています。

Borland C++ 5.0で提供される標準C++ライブラリは、C++ライブラリメーカーとして実績のあるRogueWaveからライセンス供給されるものです。これには、標準テンプレートライブラリ(STL)をはじめ、文字列や複素数など豊富なクラスが含まれます。プログラマは、データ構造やアルゴリズムを自作する代わりに、標準C++ライブラリを使うことで開発時間を節約でき、プログラムを標準化に対応させることができます。

名前空間は、自作またはサードパーティ製のライブラリなどで識別子が衝突しないようにするための仕組みを提供します。名前空間は、namespaceという予約語を使って定義します。

```
namespace Sample {
void hello() { cout << "Hello, World!" << endl; }
};
```

名前空間を定義しておけば、別のモジュールに同じhelloという名前の関数があってもSample::hello();と明示的に名前空間を指定して正しい関数を呼び出せます。しばしば呼び出す関数であれば、

```
using Sample::hello;
```

としておけば、hello();だけで呼び出せるようにもなります。

boolは、まさに論理型をあらわします。従来、C++では論理値をあらわすためにint型を使ってきましたが、新しいC++では論理型のためにboolという型を用意しました。これにともない、trueとfalseも予約語になっています。

mutableは、クラスのメンバー関数のうちconst宣言されているものからも変更できるデータメンバーを宣言するためのものです。通常、const宣言したメンバー関数はどのデータメンバーを変更することもできませんが、mutable宣言されたものだけは例外となります。

```
class A {
  mutable int Count;
  void Inc(void) const
  { Count++; }
};
```

explicitは、コンストラクタの初期化を明示的にしか実行できないようにするものです。通常、単独引数を持つコンストラクタは暗黙的に値の代入にも使えます。

```
class X { X(int); };
X a(0);
a = 1;  // OK!
```

しかし、explicitを指定されたコンストラクタは、必ず明示的にコンストラクタを記述する必要があります。

```
class X { explicit X(int); };
X a(0);
a = 1;     // BAD!
a = X(1); // OK!
```

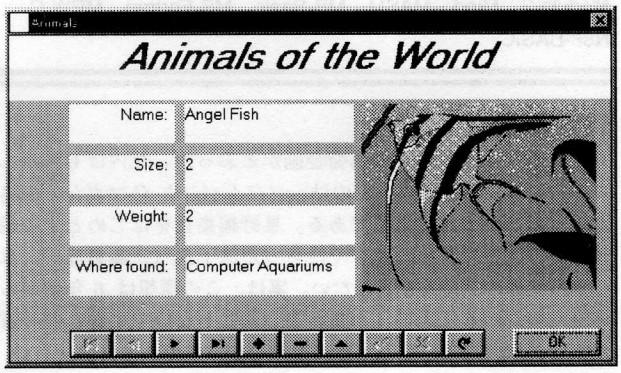
こうした標準C++への積極的な対応が、プログラムの標準化をすすめ、プログラムの将来性を高めることになります。

Visual Database Toolsによるデータベースアプリケーションのビジュアル開発

Visual Database Tools(VDBT)は、ごく簡単に言えばデータベース対応のコントロール集です。

従来、Borland C++でのデータベースプログラミングは、Borland Database Engine(BDE)を使ってAPIレベルで記述する必要がありました。しかし、VDBTを使えばリソースエディタで作成するダイアログボックスにデータベース対応コントロールを配置するだけで、ParadoxやdBASEなどのローカルデータからODBC経由でのデータ接続、SQLサーバーデータのアクセスできるアプリケーションをビジュアルに開発できます。そう、あたかもDelphiのようなデータベース開発が実現されるわけです。

VDBTは、BDEをカプセル化してテーブル、問い合わせ、データソース、ストアドプロシージャなどを実行するコントロールを提供します。また、リストボックスやエディット、メモなどの表示用コントロールを提供します。Delphiと同様、これらは設計画面で実際のデータを表示できるため、いちいちアプリケーションを構築せずにBorland C++のIDEの中でデータがどのように表示されるかを確認できます。



まとめ

Borland C++ 5.0の機能はここに書き上げたものだけではありません。Pentium/Pentium Proに対応しさらに改善された最適化32ビットコンパイラ、IDEのデバッグ機能の強化、豊富なコマンドラインツールなどすぐれた機能が盛り込まれています。

また、インターネット上のプログラミング言語として注目されているJavaの開発ツールもバンドルされています。これは、JDK(Java Development Kit)が使えるだけでなく、Java自身で開発された独自のGUIデバッグやAppExpert for Javaなどが含まれます。

さらに、上位バージョンにあたるBorland C++ Development Suite 5.0には、Javaのバイトコードを機械語に変換し、アプリケーションの実行速度を数倍にも向上させるAppAccelerator、見つけにくいメモリのバグを発見するためのデバッグツールCodeGuard、アンインストールにも対応したインストーラ作成ツールInstallShield Express、バージョン管理ツールPVCSが含まれており、迅速なアプリケーション開発を推進します。

Borland C++ 5.0は、新しい時代の高性能なプログラミング環境として幅広く利用されるようになるでしょう。

(3 ページより)ところで、この Javanese とは関係のない新しい言語が誕生した。Java という名の新しい言語が生まれたのは、ジャワ島ではなく、日出ずるところの天子といえば聖徳太子だが何の関係もなく、Sun Microsystems という米国の企業である。

Java は、コンピュータ言語のひとつであり C++に似た構文を持つ。しかし、簡易性を目指すためポインタや多重継承をなくし、メモリの解放も不要にした。このため、ちょっとしたプログラミングミスによってシステムがクラッシュすることはない(誤解はないと思うが、エラーは発生する)。重要なことは、きちんとしたオブジェクト指向プログラミング(OOP)をサポートしていることである。OOP の重要性については、2 ページ以降の特集でも紹介されている。また、マルチスレッドやネットワーク対応についても、すでにクラスが用意されていてこれらのための複雑なプログラミングにも対応しやすい。

基本的に Java コンパイラは、Java 仮想マシン(JVM)のためのバイトコードを生成する。このバイトコードは、いわば中間言語のようなものでプラットフォームには依存しない。したがって、Java 環境があれば Windows でも Mac でも UNIX でも Java アプリケーション/アプレットを実行できることになる。

現在、Java が注目されているのは、インターネット上のプログラミング言語としてである。急速に成長している World-Wide Web で利用する HTML には、Java のアプレットを埋め込むことができる(もちろんブラウザは Java 対応でないといけない)。Java アプレットがプラットフォームに依存しないことで、どのプラットフォーム上でも同じようにアプレットが実行される、さらにバイトコードは小さいのでネットワークのトラフィックも増えない。しかも、ネットワークにおいて重要なシステムクラッシュを招く可能性が低いのである。Java を使えば、インターネットの可能性はますます広がるだろう。

Java が急激に取り上げられるようになり、Borland C++ 5.0 では GUI デバッガがバンドルされるなど環境の整備も進んでいる。だが、単に流行ものだから取り上げるというのでは、Java の真価は見えてこないだろう。むしろ、Java の欠点が洗い出され、それらを充足する環境が整っていくことが、Java 普及の鍵になるのではないか。かつて C の欠点が残っているとか、洗練されていないと言われた C++も、Turbo C++/Borland C++が発売され、さまざまな機能が取り入れられ、さらに標準クラスが整備され、生産性の高い言語として成長していったのである。

[探究] アルファベット進化論

実在プログラマ

Assembler、Algol、Ada、APL、AWK、Aztec C++、Actor
BASIC、Borland C++、Borland Pascal、BCPL
C、C++、COBOL、CPL、Common Lisp、CASL、CodeWarrior
Delphi、dBASE、dBXL、Datalight C、DeSmet C
Eiffel
FORTRAN、FORTH、Family BASIC、Fred、FTL-Modula2
Gnu C、GW-BASIC
HTML、High C、HITECH C、Hu-BASIC
ISO PASCAL、iAPX86 Assembler、Icon
Java、JavaScript、Janus Ada
K-Prolog、Kaba(Prolog)
Lisp、Lex、LOGO、Let's C、Lattice C、LSI C、Lotus Script
Modula-2、Mind、MASM、MS-Basic、MS-Fortran、MPW C
N88-BASIC
Object Pascal、Objective C、Object PAL、Occam、Oberon
PL/I、Perl、Plolog、Penguin C、Perl、Power C、Power Basic
Quick Pascal、Quick Basic、Quick C
REXX
Smalltalk、Scheme、Simula、Symantec C++、SQL
Tck/Tk、Turbo Prolog、Turbo Modula-2、TopSpeed C、Think C
Ultra C
Visual Basic、Visual C++、VBA、VBS、VRML
Whitesmith C、Web、Word Basic、Watcom C/C++
xBase
Yacc
Zortech C++、Z80 Assembler

この言語／処理系は実在する／した。

編集後期

幸か不幸か昨年に続き単発企画がとおった。社内はもちろんだが、この企画が実現したのは、ソフトバンク C マガジン編集部の寛容さがあってこそである。星野編集長をはじめとする編集部の方、特にきっかけをくださった流王氏(元編集部)には心から感謝の気持ちを表したい。実は、この構想は 6 年前からあったのだが流王氏の「かまわないんじゃないですか」の一言がなければ、永久に実現することはなかっただろう。

断るまでもないと思うが、ボーランドはこういうことだけに力を注ぐようないい加減な会社というわけではない。事実、ほとんど深夜の作業で、レイアウトも単調であり、内容にも若干深みが足らなかったのではないかとも思う。とりあえず、融通のきかない製品しか出していない自由度に欠けた会社とは違った雰囲気を楽しんでいいただければ幸いである。

ボーランドにとってもうひとつ重要な D、つまり Database アプリケーションである dBASE と Paradox について触れていないのは、ページ数と本誌の経緯によるものでそれ以上の意味はない。根拠のない噂(または嘘)によって、不当かつ腹立たしい評価をまきちらさぬよう願いたい。

今回で本当の休刊にしたいところだが、2 度あることが 3 度あるかどうかは神のみぞ知るという心境である。 (O)

D MAGAZINE 96 年 4 月 1 日号
平成 8 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