1998年4月1日(毎年1回1日いちいち発行)
提供・ボーランド オブジェクト指向開発環境
Aマガジン Borland

A MAGAZINE

APR. 1998
April Fool

究極の Assembler 探究

[A MAGAZINE SPECIAL]"A"からはじまるC++Builder3の魅力

- ●特別報告 ActiveXから多層開発まで、Delphi3の威力
- ●実践報告 Application Server構築におけるIntraBuilderの実力
- ●最新情報 Java開発のA(エース)JBuilderの底力 ●恒例 復刊の挨拶

ボーランド株式会社
〒151-0073 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

ボーランドの最新情報がインターネットでごらんになれます。
http://www.borland.co.jp/

誌名変更・復刊の挨拶｜開発者の生産性を追及する A MAGAZINE

今年も再び企画が通り、復刊することになった。といっても、突然『A MAGAZINE』を読まれる方には、何のことだかわからないかもしれない。これは、『C MAGAZINE』誌（ソフトバンク刊）の裏表紙で、毎年エイプリルフールにちなんだ『なんとか MAGAZINE』という広告を出しており、その誌面に相当するものである。

ボーランドが「発行」しているものなので、「プログラミング系雑誌」というスタイルは持っているものの内容にはいささか偏りがあることは否めない。また、「カタログ」ではないので、多少文章に違和感をお持ちの方もいらっしゃるだろう。しかし、エイプリルフールだからといって「嘘」が書いてあるわけではない（最終ページを除く）。この点だけは自信を持って言える。

今回は紙メディアではなく「Borland CD」の一部として、他の情報とともに提供されることになった。本誌で語られていることは、CD-ROM に収録されているトライアル版やオートデモによって、より現実的なものとして認識していただけるだろう。

さて、毎年個人的な感想から初めて恐縮だが、昨年の最大のニュースは日経 MIX の廃止であった。もちろん、廃止の憂き目にあったいわゆるパソコン通信業者は日経 MIX だけではない。インターネットが普及し、定着するにつれ、古典的なパソコン通信が厳しい状況に置かれていることは事実だろう。事実、MIX 自身の廃止が決定される数カ月前、ボーランドは利用度の低下を理由に MIX での会議室の閉鎖を決めた。

だが、ボーランド（正確には前身となった MSA）への入社後、オンラインコミュニケーションの場として活用してきた利用者のひとりとして、MIX そのものが廃止されることには特別な感情を否めない。そして、それだけに時代の大きな変化を身近に感じるのである。

一方、ボーランドとしてのニュースは、C++Builder や JBuilder など、C++や Java の世界にも真のビジュアル開発力を盛り込んだ点がある。これらは好調な滑り出しを見せており、新バージョンの登場でますます勢いづいている Delphi とともに、現場での開発にその威力を発揮している。経済の低迷が続き、不況が深刻化する中、宣伝文句ばかりで効率の上がらないツールを使い続けていては、競争力を高めることはできない。

最近、官僚の不正や銀行の横並び意識が批判の対象となっている。しかし、形が違うとは言え、安易な横並び意識がプログラマーにはないのか、改めて問いかけてもらいたい。本誌には、横並びから抜け出して勝者となるためのボーランドのツールが紹介されている。是非、活用してほしい。

探究 究極のアセンブラ

コンパイル技術の急激な進歩やコンピュータ自身の高速化によって、アセンブリ言語が使われるケースは以前に比べて少なくなってきた。しかし、CPU 命令に 1 対 1 対応しているアセンブリ言語は、コンパイラの生成コードより常にすぐれたプログラミングが可能なはずである。また、MMX や 3D 命令など、CPU の進化が注目されている今、改めてアセンブリプログラミングを見直してみることにしよう。

其の一　はじめに

かつて MS-DOS という OS が主流だった時代、アセンブリ言語だけで実行ファイルを作成することはさほど難しいものではなかった。プログラムは、実行開始位置と終了ファンクションの呼び出しさえ行っていればよかったのである。MS-DOS 用の便利なツールもアセンブリ言語で開発されることが多かった。しかし、Windows 以降、アセンブリ言語だけのプログラムはほとんどなくなった。ごく単純なプログラムでも面倒な初期化が必要であり、その方法はほとんど解説されていない。

そして、実際、そこまでアセンブリ言語で書かなければならないことはほとんどない。初期化処理など全体から見ればわずかな時間しか消費しないし、Windows によるオーバーヘッドを考えればメインウインドウの作成やメッセージループといった部分をアセンブリ言語で書き直すことはあまり有意義な努力とは言えないだろう。以前は、実行ファイルを小さくするか速度を高速化するかが分かれていたが、現在ではメモリやハードディスクの拡張も容易であり、わずかなメモリを節約するよりも速度を向上させる方が好まれている。

また、一般にアプリケーションの 10%の部分が実行時間の 90%を消費すると言われている。もちろん、実際の動作はアプリケーションによって違うだろうが、たいして時間を消費しない部分に力を注いでも全体的な速度向上には結びつかない。あるいは、ユーザーインターフェースの入力待ち状態など、速度に影響しない部分もあるだろう。当たり前のことだが、数値計算やグラフィック処理など、時間がかかっている部分を集中的に効率をあげることが望ましい。

※注意　正しくは「アセンブラ」とは Turbo Assembler のようなアセンブリ言語で書かれたプログラムを処理するもののことである。しかし、ここではアセンブリ言語によるプログラミングについて述べる。また、16 ビットについては考えない。

其の二　整数計算の効率化

一般に、コンピュータは整数計算が得意である。CPU には、32 ビットの整数つまり C/C++の int や Delphi の Integer に対応するレジスタがあり、整数計算はお手の物である。しかし、加減算よりも乗算はやや遅く、除算はさらに遅い。

たとえば、Pentium において 32 ビットの加減算はほぼ 1 クロックで実現できるが、乗算は数クロックかかり、除算は

40 クロックもかかる。条件は限られるが、定数倍の乗算を高速に実行するために LEA 命令を使える場合がある。

この例を次の表に示す。

2 倍	LEA	EAX, [EAX*2]
3 倍	LEA	EAX, [EAX*2 + EAX]
4 倍	LEA	EAX, [EAX*4]
5 倍	LEA	EAX, [EAX*4 + EAX]
6 倍	LEA	EAX, [EAX*2 + EAX]
	SHL	EAX, 1
7 倍	LEA	EAX, [EAX*2 + EAX]
	LEA	EAX, [EAX*2 + EAX]
8 倍	LEA	EAX, [EAX*8]
9 倍	LEA	EAX, [EAX*8 + EAX]
10 倍	LEA	EAX, [EAX*4 + EAX]
	SHL	EAX, 1

このくらいの命令は C++Builder や Delphi でも使ってくれるので、このためだけにアセンブリ言語を使う必要はない。

また、32 ビットを超える整数演算が必要ならば、アセンブリ言語は役立つだろう。C++Builder には__int64、Delphi には Comp という 64 ビット整数を扱う型があるが、前者は乗除算で内部関数を呼び出し、後者は浮動小数演算に置き換えられる。これらをアセンブリ言語で直接記述すれば、かなり高速になるはずだ。ただし、多倍長の除算は面倒である（除数が 32 ビットなら、DIV 命令の繰り返しでよい）。

其の三　遺物

8086 の時代には乗算は 100 クロックもかかる遅い命令だった。そのため、定数倍をシフト命令や加算命令に置き換えることもあった。しかし、今まではこんな置き換えは無駄であり、かえって遅くなることがある。古代の遺物に惑わされていると、かえって無駄なプログラムを書いてしまいかねない。

たとえば、[BX+AL]の内容を AL レジスタに取り出す XLAT という命令は、以前は便利だったかもしれないが、単純なデータ転送命令が 1 クロックで済む今でも 4 クロックもかかる上に Pentium のペアリングにも対応していない。また、繰り返し処理によく使われた JCXZ や LOOP は、デクリメントと条件ジャンプに置き換えた方が速くなる。リピート命令も繰り返し回数が少ない場合は効率が悪い。また、LODS は、バイト数が節約できるだけで、MOV AL,[ESI]の方が速い。

逆に、以前は非常に遅かったものでも速くなっているものがある。8086 では 10 クロック近くかかっていた PUSH/POP 命令が、486 や Pentium では 1 クロックで済む。たとえば、DX:AX の値を ECX に設定するために、PUSH AX、PUSH DX、POP ECX とできる（MOV CX, AX、ROL ECX, 16、MOV CX, DX でもよい）。

また、Pentium Pro・Pentium II では、ジャンプ命令は、分岐予測によってずいぶん高速化されている（near/short ジャンプは 1 クロック）。分岐予測がはずれた場合のペナルティは大きいが、履歴や静的予測（前方条件分岐は分岐しない、ループのような後方条件分岐は分岐すると予測）などによって、ペナルティが発生することは少ない。数回の命令を展開してジャンプを減らすことはあるが、ジャンプ命令を恐れて面倒なプログラムを書いてしまうことは避けるべきである。また、Pentium Pro や Pentium II には、CMOVcc という条件付き転送命令が 2 クロックで実行でき、ペナルティの心配もない。

MOVSX は、符号拡張のための命令で、MOVSX ECX, CX とすれば、EAX 以外の符号拡張にも使える。しかし、この命令は Pentium では 3 クロックかかるので、ROL ECX, 16、SAR ECX, 16 の方が速い。と思っていたら、Pentium II での実測値は MOVSX の方が速かった。置き換えの際に命令数が増えて、好ましくない結果になってしまった例である。

其の四　ビット演算

C++や Pascal が苦手とし、CPU に直接実行させて効果があがるのはビット演算だろう。単純なビット演算は、C++や Object Pascal にもあるが、多倍長のビット演算は CPU 命令を直接使う方がよい。たとえば、16 バイトのメモリをレジスタに入れて 4 ビットずらしたい場合は、次のようにする。

```
SHLD    EDX, ECX, 4
SHLD    ECX, EBX, 4
SHLD    EBX, EAX, 4
SHL     EAX, 4
```

また、EAX レジスタに含まれる「1」のビットの数を調べるために、次のようなプログラムが書ける。

```
SHLD    EBX, EAX, 31
AND     EBX, 55555555H
SUB     EAX, EBX
SHLD    EBX, EAX, 30
AND     EAX, 33333333H
AND     EBX, 33333333H
ADD     EAX, EBX
SHLD    EBX, EAX, 28
ADD     EAX, EBX
AND     EAX, 0F0F0F0FH
SHLD    EBX, EAX, 16
ADD     AX, BX
ADD     AL, AH
```

次のプログラムは、EAX レジスタのビットを左右反転させるものである。

```
ROR     EAX, 7
MOV     EBX, EAX
AND     EAX, 55555555H
XOR     EBX, EAX
ROL     EBX, 2
OR      EAX, EBX
MOV     EBX, EAX
AND     EAX, 33333333H
XOR     EBX, EAX
ROL     EBX, 4
OR      EAX, EBX
MOV     EBX, EAX
AND     EAX, 0F0F0F0FH
XOR     EBX, EAX
ROL     EBX, 8
OR      EAX, EBX
BSWAP   EAX
```

最後の 2 つは、日経 MIX のアセンブラ会議で moritan（森田和郎氏）が提示されたものである。理屈はともあれ、間違いなく動作する。こんなものを見てしまうと、とても「究極のアセンブラ」なんて記事は、恥ずかしくて書き続けられないので、今日はここまで。

MAGAZINE SPECIAL

“A”からはじまる C++Builder 3 の魅力

昨年リリースされた C++Builder は、真の意味での「ビジュアル開発」と「C++」を両立させた画期的なツールでした。今回、新たにリリースされた C++Builder 3 では、従来にもまして魅力的な機能がサポートされています。ここでは、代表的な機能を 26 個にまとめてレポートします。

ANSI C++

C++Builder の魅力は C++が使えるという点です。ビジュアル開発の生産性が高いといっても、既存の C++のノウハウが活かせないようでは「ビジュアルな C++開発ツール」としての魅力は半減してしまいます。C++Builder は、名前空間、テンプレート、演算子オーバーロードなど最新の ANSI C++の機能に加えて、実績のある RogueWave 社の標準 C++ライブラリ 2.0 が添付されています。

また、C++としてのしっかりした基盤を持っているため、開発したアプリケーションがビジュアルでない他の C++処理系と同等のパフォーマンスを実現しています。

BusinessInsight

高度な意思決定に求められる多次元解析をビジュアルに設計できる Decision Cube コンポーネントが組み込まれています（C/S 版のみ）。Decision Cube は、複数の条件に対応するクロス集計を、コンポーネントベースで実現するものです。コンポーネントとして提供されていることで、解析したい情報はダイアログボックスを使って簡単に設定できる上、アプリケーションに解析機能を埋め込むことができます。また、解析結果は表（グリッド）形式だけでなくグラフ形式でも表示できます。

Compiler

高速なコンパイラ・リンカが、ビジュアル開発に欠かせないすばやいレスポンスを実現します。C++Builder 3 では、さらにコンパイラ技術を進化させ、プロジェクトが利用するプリコンパイルヘッダをダイナミックに認識して、自動的に適切なプリコンパイルヘッダ情報を取り込むように改良されています。このため、2 回目以降のプロジェクトの構築時間がいっそう短くなっています（すばやいレスポンスのために、48～64MB 以上の RAM を搭載することをお勧めします）。

C++Builder 3 では、コンパイルオプションが詳細に設定できるようになり、他の処理系で開発されたプロジェクトやソースプログラムの利用にも配慮されています。また、OWL 5.0 や MFC 4.2 もサポートされています。

Database Scalability

dBASE、Paradox、Access .MDB（DAO 経由）、テキスト形式のデータから ODBC 接続、SQL Link による SQL データベースへの直接接続までを共通のデータベースエンジン（Borland Database Engine）でサポートします。テーブル、問い合わせ、ストアドプロシージャなどのデータセットやバックグラウンドでの問い合わせなどを処理するためのセッション、多彩なデータ表示用コントロールが、すべてのデータに対して共通のスタイルで利用できます。

Environment

使いやすいビジュアル開発環境も C++Builder の特長です。C++Builder では、ビジュアルな開発や高速性だけが重視されているのではありません。従来の C++のノウハウを活用したいソースコード開発においても、すぐれた使い勝手が実現されています。ビジュアル開発とソースコード開発を密接に連携させる 2Way-Tool によって、ビジュアル開発で設計したフォームに対応して、即座にソースコードの内容が更新される機能や、ビジュアル開発とソースコード開発の間でカット＆ペーストを行なったり、ソースコードでの編集がビジュアル開発に反映される機能の総称です。

また、スピードバーやコンポーネントパレットのカスタマイズ、スピードメニューを使ったコマンドの呼び出し、位置やサイズの微妙な調整を実現する機能など、使いやすいユーザーインターフェースを効率的に設計するための手段が豊富に提供されています。また、スピードメニューなどを使って目的の機能を簡単に呼び出せます。

さらに、開発で時間をとられがちな、デバッグ作業についても通過回数や条件式を指定できるブレークポイント、関数呼び出しもできる式の評価、即座に変数の値を調べられるツールチップ、実行ファイルや DLL の状態を調査できるモジュールビュー、アセンブリレベルでのデバッグをサポートする CPU ビューなど、信頼性を高めるためのデバッグ機能も充実しています。

Form Inheritance

C++Builder では、プロジェクト中のフォームから新しいフォームを継承して作成できます。基本フォームに変更を加えると、それは継承したフォームにも反映されます。フォームの継承は、作成したプロジェクトや設計したフォームの再利用性を高めるものです。しかも、継承を何段階繰り返しても、実行速度の低下を招きません。

Global Optimization

Pentium スケジューリング、メモリ・文字列関数のインライン展開、誘導変数の最適化、共通式の削除など、高度な最適化が実装されています。また、テンプレート構文の評価を改善し、コード生成の効率を向上させました。

この他にも、動的メソッドのサポートや try～finally スタイルの例外処理による効率化、命令セットや呼び出し規約など多くのコンパイルオプション、MMX 対応アセンブラを使ったインラインアセンブリ処理など、究極のパフォーマンスに対する妥協のない機能を実装しています。

Help

改良されたオンラインヘルプや充実したサンプルプログラムが、C++Builder 3 の理解を助け、いっそう迅速なプログラミングを支援します。

Internet
17 種類ものプロトコルコンポーネントを使って、インターネット対応のアプリケーションもビジュアルに開発できます。

Japanese
C++Builder 3 では、日本語対応をプロジェクト自身と分離でき、各国語対応が容易になりました。リソース DLL ウィザードを使うと、フォームのリソースだけを分離した専用の DLL プロジェクトを作成できます。オリジナルを英文にしておけば、英語環境と日本語環境で自動的にユーザーインターフェースが使う文字列を変更できます。

Key customize
Windows 標準、ボーランドの古典的なキー設定、Brief 相当、Epsilon 相当の 4 種類のキー設定から好みのキー操作を選択できます。また、デバッグ機能を使う際には、Turbo Debugger 互換のキー設定も選べます。

Local InterBase
Local InterBase スタンドアロン環境で Client/Server スタイルのアプリケーションを開発するために便利なツールです。本格的な RDBMS である InterBase との互換性を持ちながら、スタンドアロンでの接続をサポートしており、ストアドプロシージャやトリガーなど SQL データベース特有の手法を使ったプログラムを開発できます。

Multi Thread
Windows 95/NT の機能であるマルチスレッドを使えば、時間のかかる処理をバックグラウンドで処理し、ユーザーインターフェース部分に負担をかけないというプログラミングができます。また、BDE（Borland Database Engine）もマルチスレッドに対応しているため、時間のかかる問い合わせをバックグラウンドで処理することもできます。

Native Code
かつてビジュアル開発ツールの多くは、中間コードという手法を使っていました。C++Builder では、開発時点から他の C++処理系と同じくネイティブコードを生成しますから、実行速度が低下する心配はありません。

Object Oriented
C++Builder の基盤はオブジェクト指向が支えています。カプセル化、継承、ポリモーフィズムといったオブジェクト指向プログラミングの特長が、C++Builder のビジュアルな魅力としてあらわれていると言ってもよいでしょう。オブジェクト指向のメリットは、ある時点でのアプリケーションを開発が楽になるだけでなく、開発したアプリケーションの保守や拡張が容易であるという点にもつながっています。きちんとしたオブジェクト指向を組み込むことで、目先の便利さだけでなく長期的な効率化が実現されています。

Project Manager
C++Builder 3 のプロジェクトマネージャは、複数のターゲットを管理できます。実行ファイルと DLL、オートメーションサーバーとコントローラ、リモートデータサーバーと軽量クライアントなど、複数の実行ファイルからなる大規模なプロジェクトを開発する際に便利です。

QuickReport
大幅に改良された QuickReport 2.0J では、QuickRep というグリッドが表示されたビジュアルコンポーネントを使って、レポートの設計がさらに容易になりました。

Remote Dataset
多層環境を実現するためのリモートデータ制御もコンポーネントベースで開発できます。リモートデータセットは、BDE(Borland Database Engine)に依存しないデータセットであり、この仕組みを応用して他のデータベースエンジンを利用するコンポーネントも作成できます。

SQL Link
SQL Link を使うと、InterBase や Oracle といった SQL データベースに対して、直接 SQL を使って接続でき、高速なデータベース処理を実現できます。

TeeChart
TeeChart は、円グラフや棒グラフをはじめとする 11 種類の表示形式や立体化、複数ページの作成、形状の詳細な設定など、収集した情報やデータを多彩なスタイルで表示できるグラフコンポーネントです。

Upsizing
Data Migration Wizard を使うと、ローカルテーブルを SQL データベースにアップサイズすることも容易です。

Visual Component Libary
C++Builder 3 の魅力には、通常のコントロールから編集機能付きの文字列グリッドや書式付きで入力できるマスクエディットといった拡張コンポーネントなど 100 種類以上のコンポーネントが利用できるところにもあります。また、C++Builder 自身で新しいコンポーネントを開発することもできます。C++Builder のコンポーネントは技術的には C++のクラスですが、設計時にもコンポーネントとして動作させることができ、設計時点での微妙な調整ができます。

Web
NSAPI/ISAPI や CGI を使った Web サーバーアプリケーションを開発するための専用モジュールやコンポーネントが提供されています。テーブルや問い合わせといったデータセットから自動的に HTML の表を作成することもでき、他の HTML デザイナなどと組み合わせて、高速なイントラネットシステムを構築できます。

ActiveX
C++Builder 3 では、既存のコンポーネントを ActiveX コントロールに変換したり、Active フォームによるフォーム形式での ActiveX コントロールの開発がサポートされています。タイプライブラリエディタによるプロパティやメソッドの拡張、プロパティページの作成、インターネット配信用の設定など、機能も充実しています。

repositorY
設計したフォームやプロジェクトはリポジトリ（倉庫）に登録できます。登録した要素は、別のプロジェクトを作成する際のテンプレートとして再利用できます。また、メニュー項目もテンプレートとして保存したり、再利用することができます。

wiZard
プロジェクトやフォーム、コンポーネント、スレッドなど、C++Builder で使うさまざまな要素を作成するために各種のウィザードが提供されています。TeeChart によるグラフを簡単に作成するための TeeChart ウィザードやレポートを簡単に設計するための QuickReport ウィザードなどもあります。

※注意　製品形態によって利用できる機能は異なります。

ActiveXから多層開発まで、Delphi 3の威力

製品出荷前でありながらそのパワフルな開発力が評価されCOMDEXで「BEST OF BYTE」賞を受賞したのが1994年の秋である。あれから3年半、Delphiは着実に実力を発揮する場を獲得してきた。最新版のDelphi 3では、ワンステップでのActiveXコントロール開発やコンポーネントベースでの多層開発など、ビジュアルな開発力が存分に活かされる内容となっている。この開発力は一朝一夕にできあがったものではない。Delphiの真価は、長年にわたる開発ツールの実績の積み重ねにあるのだ。今回は、ボーランドのPascalコンパイラの歴史を振り返ることで、Delphiの背景を再確認してみよう。

8ビット時代からTurbo Pascal 3.0まで

Delphiの最新版はバージョン3です。しかし、搭載されているコンパイラは15年前に発売されたTurbo Pascalからバージョンアップを繰り返してきたものです。コマンドラインコンパイラであるDCC32.EXEを実行させると「Delphi for Win32 Version 10.0」と表示されます。つまり、最初のバージョンを除き、コンパイラの改良が9回繰り返されてきたわけです。まず、その歴史からひも解いてみましょう。

1983年にリリースされた最初のバージョンは8ビットのOSであるCP/M用に開発されたもので、このときすでにエディタとコンパイラが統合された環境を持っていました。しかしながらサイズは20KB程度とコンパクトであり、開発環境の一部を実行ファイルのためのライブラリに利用するといったテクニックも使われていました。

Turbo Pascalは、コンパクトなサイズ、軽快な動作、そして生成されるプログラムの品質の高さという、開発ツールとしての魅力を持ったツールで、しかも安価でした。「安価だからといって心配は要りません。我々はボーランドという確かな技術力を持った会社なのです」という当時のメッセージは、今でも色あせていません。こうして、Turbo Pascalはホビーからビジネスまで幅広い支持を得ました。

DelphiのObject Pascalが、この頃のTurbo Pascalの正統な継承者である証拠が今も残されています。フォームのOnKeyPressイベントを次のように定義してください。

```
procedure TForm1.FormKeyPress(Sender: TObject; var Key: Char);
begin
  if Key = ^A then Caption := 'Ctrl-A';
end;
```

プログラムを実行して[Ctrl]+[A]を押すとこのイベントハンドラが呼び出されます。「^A」という表記は、[Ctrl]+[A]に対応する文字コードをあらわすものです。^は、コントロール＋英字をあらわすためによく使われていたものですが、このような実用的な拡張も取り込んでいく姿勢は、当時からありました。もちろん、Turbo PascalはNiklaus Wirth（ニクラウス・ヴィルト）氏が開発したPascal言語を処理するもので、Pascal言語が教育用の覚えやすい言語だったこともTurbo Pascal普及の一因であることに違いはありません。

1984年にバージョン2.0がリリースされ、CP/M86やMS-DOSにも移植され、Turbo Pascalはますますその勢いを伸ばしていきました。日本語マニュアルが添付されたバージョンが最初に発売されたのも、このバージョン2.0です。

そして、1985年にリリースされたバージョン3.0では、オーバーレイのサポートやBCD・8087に対応したバージョン、グラフィックや数値計算に関するツールボックスの提供など、初期のTurbo Pascalの完結編と言ってよい製品でした。

DOS版Turbo Pascal

1987年ユーザーインターフェースが一新されたTurbo Pascal 4.0がリリースされました。Turbo Pascal 4.0は、マシンの機能を利用して開発環境が全面的に書き直され、拡張されたエディタやダイアログによるオプションの設定、その場で呼び出せるオンラインヘルプなど、当時としては画期的な機能が組み込まれた製品でした。

また、Delphiでは当たり前のように使われる「ユニット」が採用されたバージョンでもあります。ただし、ボーランドのPascalで最初にユニットを採用したのは、Macintosh用のTurbo Pascalです。ユニットは、おおざっぱにたとえるとC++のヘッダとプリコンパイル情報をまとめて扱うもので、大規模なプロジェクトも短時間でコンパイルできました。

実のところ、Turbo Pascal 4.0の日本での評価は賛否両論でした。もちろん、新しい機能は歓迎されたのですが、Turbo Pascal 3.0以前のユーザーの方には特定の機種に依存することへの抵抗感があり、またオーバーレイがサポートされなくなったことへの不満もありました。

1988年、ボーランドは開発ツールを一新しました。そのひとつがTurbo Pascal 5.0です。開発環境でのソースレベルデバッグや言語・ライブラリのオンラインヘルプをサポートし、オーバーレイが高度な仕組みによって復活しました。Turbo Assembler&Debuggerという味方とともに、その威力をいかんなく発揮できるようになったのです。この時期、すでに日米双方でマシンの寡占化が進んでおり、機種依存性に対する抵抗は以前よりも少なくなっていました。

1989年、Turbo Pascal 5.5がリリースされました。バージョン番号からはマイナーなバージョンアップに思われるかもしれませんが、これはボーランドがはじめてオブジェクト指向プログラミングを取り入れた製品です。

Turbo Pascal 5.5は静かな船出ではありましたが、翌年にリリースされたTurbo Pascal 6.0ではTurbo Visionというユーザーインターフェースを設計するためのクラスライブラリも添付され、オブジェクト指向プログラミングへの取り組みが本格化していきます。Turbo Visionは、Turbo Pascal 6.0の統合開発環境にも使われていました。自社製品の開発に自社製品の開発ツールを使うという当たり前のことが、当たり前に行われているのです。当時のCEOであるPhilip Kahn氏（現StarFish社CEO）はこう語っています。「ボーランドの強みは、我々自身が最先端の開発言語を持っていることです。私たちは、常に改良を心がけています。もし、ボーランドが『もう改良は必要ない』と言い出すようなら、ボーランドのことは忘れなさい。」

この後、1992年にはTurbo Pascal 7.0（日本語版未発売）がリリースされています。

Turbo Pascal for Windows

ボーランドがはじめて Windows 用の開発ツールをリリースしたのは、1991 年の Turbo Pascal for Windows です。Windows 1.0、2.0 は一般に定着するには至っていませんでしたが、Windows 3.0 は違っていました。CPU のプロテクトモードを使ったメモリ空間の広がりによって実用性が増し、急速にユーザーを獲得していきました。Turbo Pascal for Windows は、そこに登場しました。

もともと C 言語中心で設計された Windows に Pascal を持ち込むため、Pascal に大幅な改良が加えられることになりました。C 言語のスタイルでもあるヌルコードで終了する文字列を扱うために PChar 型が取り入れられたのも、Turbo Pascal for Windows からです。

また、Turbo Pascal for Windows では、オブジェクト指向プログラミングの能力を Windows で活かすためのクラスライブラリ ObjectWindows を提供していました。ここで、単純なクラスライブラリでは Windows メッセージを処理効率が悪くなります。通常のクラスライブラリの設計手法では、扱うメッセージが増えるほどクラスを継承するたびに消費するメモリが多くなってしまうのです。これを避けるために、virtual の直後にメッセージ定数を記述して、動的メソッドを定義できるようにしました。これは、現在の message 文に似た文法です。動的メソッドによって、効率的なクラスライブラリが実装できるようになりました。

1992 年には、Windows 3.1 対応の Turbo Pascal for Windows 1.5（日本語版未発売）がリリースされました。また、Turbo Pascal の上位製品としてリリースされた Borland Pascal 7.0（日本語版未発売）は、DOS/Windows 対応をひとつのパッケージにまとめた製品です。また、DOS 用のプロテクトモードアプリケーションや Windows DLL の共有などを実現していました。

ビジュアル開発の先駆け ObjectVision

Pascal や Delphi とは直接関係ありませんが、ボーランドの「ビジュアル開発ツール」として忘れられないのが ObjectVision です。Turbo Pascal for Windows とほぼ同時期にリリースされた ObjectVision（日本語版未発売）は、データベースアプリケーションを設計する専用のビジュアル開発ツールでした。ObjectVision は、まさにデータベースコンポーネントをフォームに配置する感覚でアプリケーションを作成できました。

ObjectVision は、現在の Borland Database Engine の前身である Paradox Engine を使ってデータベース処理を実現していました（制限付きで dBASE や Btrieve も利用できました）。Paradox Engine は、当時ボーランドが発売していたデータベースアプリケーション Paradox で使われているテーブルを開発ツールで利用するためのライブラリです。今でこそ、データベースアプリケーションのビジュアル開発は珍しくありませんが、当時はデータベースライブラリを使ったソースコードプログラミングが主体であり、ObjectVision は画期的なツールでした。

しかし、ObjectVision はプログラミング言語ではなく、あくまでユーザーインターフェースの設計ツールであったため、市場での認知度はさほど高まりませんでした。しかし、ユーザーの方には高い評価を受けていたのも事実です。

その後、ObjectVision 2.0、2.1、OS/2 版などがリリースされています。

Delphi の登場

1994 年の COMDEX で、Delphi は「BEST OF BYTE」という賞を受賞しました。1992 年に Borland Pascal をリリースして以来、2 年以上にわたって開発が続けられてきた新バージョンであり、ベータ版の評価が非常に高かった結果でした。

製品がリリースされた 1995 年、Delphi は開発ツールの既成概念を打ち破るツールとしてまたたく間に広がりを見せはじめます。いままで、どちらかといえば地道な活動が多かった Pascal ユーザーの中には、Delphi によって活躍の場が広がったという方も少なくありません。

1996 年には 32 ビット版 Delphi 2.0 がリリースされ、1997 年には、Delphi 3 がリリースされました。いまや趣味のプログラミングから大規模な業務システムまで、Delphi は幅広い利用者を獲得しています。今日の Delphi の躍進ぶりは、よく知られるところでしょう。

Delphi は、当初からネイティブコードを生成するコンパイラ、豊富なコンポーネントを備えたビジュアル開発環境、ローカルデータから SQL データベースにまで対応できるデータベーススケーラビリティを統合した開発ツールとして宣伝されてきました。

しかし、これらが「単純に統合されたツール」ということが本質ではありません。たとえば、Delphi のコンポーネントは Object Pascal におけるクラスです。ビジュアル開発ツールの多くは、コンポーネントはツール自身に組み込まれた機能であったり、完全に分離した OCX や DLL です。こうした関係では、オブジェクト指向プログラミングの特長である継承などの特長が活かせませんし、何より既存のソースコード中心のコンパイラに匹敵するパフォーマンスは得られません。

Delphi は、コンポーネントを設計時に利用するため、そのクラスをコンポーネントライブラリに登録した時点で、専用の DLL にも埋め込んでしまいます。つまり、DLL という実行可能な形式が用意されているために、Delphi は設計時にクラスを利用できますし、コンポーネントは設計フォームの上で実際に機能するのです。ビジュアル開発に欠かせないすばやいレスポンスは、毎分数 10 万行というコンパイル速度とユニットという仕組みによって支えられています。

Delphi 自身でコンポーネントを設計できるのも、技術的にはコンポーネントがクラスだからです。フォームの継承も Object Pascal のフォームクラスを継承しているに過ぎません。もちろん、その背後には、継承によって生じるさまざまな問題を吸収するための Delphi なりの工夫があります。

Delphi 3 で、ActiveX コントロールを作成したり、多くのコンポーネントが提供されたり、リモートデータ制御のビジュアル開発ができるのも、確かな技術に基づいているのです。オブジェクト指向プログラミングの特長は、既存の資産を活かす保守性や拡張性にありますが、Delphi では多くの資産があらかじめコンポーネントとして提供されていると言ってよいでしょう。

まとめ

ここに述べた歴史や内部の仕組みは、普段 Delphi を活用するために必要な知識ではありません。しかし、今は目先の機能や特長を追いかける時代ではありません。安易な発想がもたらしたバブル経済が、今日の深刻な不況を招いています。今こそ、確かな目で開発ツールを選んでほしいと願います。

実践報告 Application Server 構築における IntraBuilder の実力

イントラネットの導入が進む

イントラネット(注1)は本格的に普及し始めました。イントラネットの導入調査(注2)では、従業員 5,000 人以上の大企業の場合、すでに半数近くが導入を済ませ、中小企業を含めた全体の場合でも、1年後には過半数の企業が導入を予定している模様です。

企業はイントラネットに何を求めているのでしょうか？答えは、社内での情報共有の効率化と、社内情報システム構築の効率化です。

社内の情報共有の大部分はデータベースの運用で行われますが、部署によりデータのフォーマットは異なります。また、クライアント PC にデータベースと接続するためのソフトのインストールや、複雑な設定が必要な場合があります。さらに、典型的な C/S モデル(注3)やファイルサーバー形式(注4)では、TCO(注5)の削減は望めません。

イントラネットで情報を共有する場合、システムを提供する Web サーバーと情報を蓄積するデータベースとが連携した 3 階層アプリケーション(注6)が利用されます。3 階層アプリケーションは、ファンクショナリティ層部分の開発に専念でき、かつ、クライアント PC とデータベースサーバー間の通信負荷が軽減できます。また、クライアントは、Web ブラウザが利用でき、簡単な操作性とプラットフォームに対する非依存性は、TCO 削減に大きな役割を果たします。

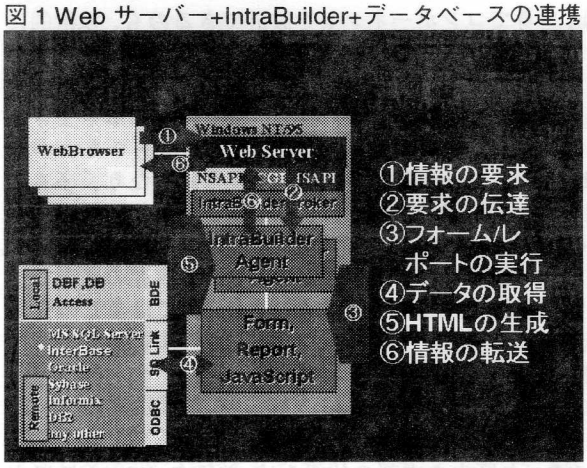
Web サーバーとデータベースの連携

Web サーバーとデータベースの連携ソフトは、1996 年頃から、データベースの検索結果を HTML に出力する、静的な HTML ジェネレータがいくつか提供されました。しかし、1 回(1 画面)のデータ入力で、複数のデータベースを更新するトランザクション機能を搭載したものでなくては、社内での情報共有を図る業務システムを構築することはできません。

ボーランドの IntraBuilder は、主要な Web サーバー(注7)とデータベース(注8)をサポートする Web アプリケーションサーバー構築ツールです。IntraBuilder の特長は、ダイナミックな HTML 生成機能を持ち、常に最新のデータを反映した HTML ベースのフォームとレポートを作成することです。フォームとレポートは、コンポーネントベースのビジュアルなプログラミング機能を備えた開発環境でデザインでき、カスタマイズ言語には JavaScript(注9)を利用します。また、柔軟な開発に欠かせない 2Way-Tool 機能(注10)を搭載しています。

IntraBuilder が最新データを反映した動的な Web パブリッシングを実現する仕組みは、次のようになります(図 1 参照)。

図 1 Web サーバー+IntraBuilder+データベースの連携



(1) Web ブラウザが Web(HTTP) サーバーに情報を要求する。

(2) Web ブラウザが受け取った要求を IntraBuilder ブローカが IntraBuilder エージェントに伝達する。

(3) IntraBuilder エージェント(HTML 生成エンジン)で、IntraBuilder デザイナで作成したフォームかレポートを実行する。

(4) Borland Database Engine を通じて、最新のデータを取得する。

(5) IntraBuilder エージェントが、最新データを含んだフォームスタイルかレポートスタイルの HTML を生成する。

(6) 生成された HTML は、IntraBuilder ブローカと Web サーバーを経由して、Web ブラウザに返信される。

(注 1)Web などインターネット技術をベースに構築する社内情報システム

(注 2)日経コンピュータ誌(日経 BP 社)の「情報システム利用実態調査」

(注 3)アプリケーションを「プレゼンテーション層」「ファンクショナリティ(アプリケーション)層」「データ層(データベースサーバー)」に分類すると、プレゼンテーション層とファンクショナリティ層がクライアント PC で処理され、データベースサーバーがサーバーPC で運用される 2 階層アプリケーション

(注 4)アプリケーションを「プレゼンテーション層」「ファンクショナリティ(アプリケーション)層」「データ層(データベースサーバー)」に分類すると、データベースは PC-LAN で共有されるが、プレゼンテーション層、ファンクショナリティ層、実質的なデータ層のすべてがクライアント PC で処理される単層アプリケーション

(注 5)トータルコストオブオーナーシップ(総所有費用)

(注 6)アプリケーションを「プレゼンテーション層」「ファンクショナリティ(アプリケーション)層」「データ層(データベースサーバー)」に分類すると、プレゼンテーション層はクライアント PC で処理され、データベースサーバーがサーバーPC で運用され、ファンクショナリティは中間層のアプリケーションサーバーPC で処理されるアプリケーション

(注 7)Netscape FastTrack Server、Netscape Enterprise Server、Microsoft Internet Information Server、O'Reilly WebSite、Borland Web Server をサポート

(注 8)dBASE 形式、Paradox 形式、InterBase*、Oracle*、MS SQL Server*、Sybase*、Informix*、DB2*、ODBC に対応した Access や SQL データベースをサポート(*付きの RDB は SQL Link を使用することでネイティブ接続可)

(注 9)ネットスケープ社のサーバーやクライアントツールで広く使用されているスクリプト言語

(注 10)エディタによるハードコーディングと、デザインツールによるビジュアルプログラミングのどちらか一方の開発環境に縛られることなく、2 つの環境を切り替えながら開発を進めることができるのシームレスな開発環境

(注 11)一部の業務を自動化したり、進行の状況をチェックするなど、業務の流れを管理すること

イントラネットとグループウェア

イントラネットでの情報共有の手段として、ワークフロー管理(注 11)を含んだグループウェアの利用が目立っています。グループウェアが利用される範囲は広がりつつあります。本来、電子メールやスケジュール機能などでグループ内の情報共有を目的としていましたが、ワークフロー管理などを含むことで、全社的な企業内の情報共有手段になりつつあります。

IntraBuilder は、企業内の情報共有手段のサンプルプログラムとして、掲示板、電子メール、スケジュール、会議室予約、勤怠管理で構成されている小規模事業所内でのグループウェア構築を支援する IntraBuilder グループウェアを用意しています。IntraBuilder グループウェアは、http://www.borland.co.jp/intrabuilder/intragw.htm からダウンロードできます。含まれていない機能などは、自由にカスタマイズして追加して利用することができます。

図 2 IntraBuilder グループウェア

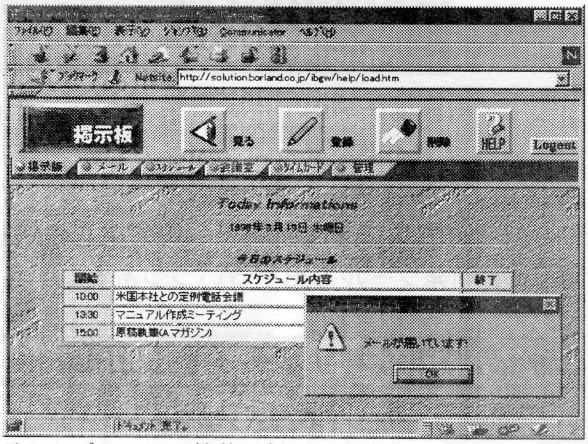
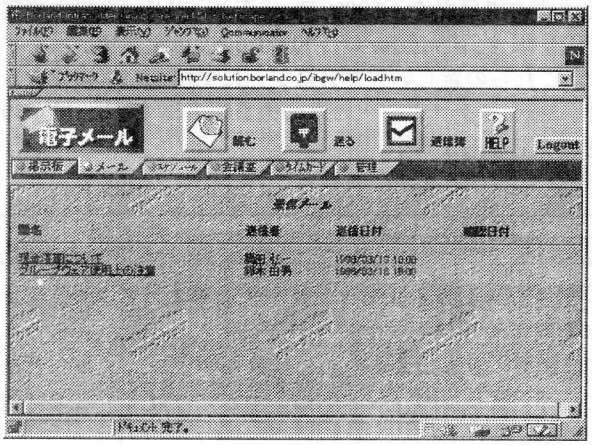


図 3 IntraBuilder グループウェアのメール機能



IntraBuilder グループウェアの機能は以下のとおりです。

- 掲示板　IntraBuilder グループウェアを使用する全ユーザーに対して、情報提供を行なう機能。情報はジャンル別に登録できる。
- 電子メール　IntraBuilder グループウェアを使用する特定ユーザーに対して、メールの送受信を行なう機能。送信簿により、メールの済読/未読が確認できる。
- スケジュール　IntraBuilder グループウェアを使用するユーザー個人のスケジュール管理機能。また、複数人の空き時間を調べるグループスケジュール確認も含まれる。
- 会議室予約　会議室や他の施設のスケジュール管理機能。自分の PC から、複数の会議室の空き時間を確認/予約が行なえる。
- 勤怠管理　タイムカードをイメージした勤怠管理機能。出社、外出、帰社、退社の時間の登録が、自分の PC から行なえる。

IntraBuilder グループウェアは、ボーランドのソリューションサーバー（http://solution.borland.co.jp/ibgw/index.htm）で体験できます。

アプリケーションサーバーの規模

IntraBuilder 1.5 には、Standard 版、Professional 版、Client/Server 版の 3 パッケージがあり、構築する Web アプリケーションの規模により、利用するパッケージが異なります。それぞれの想定規模は、表 1 の通りです。

表 1 パッケージ毎の想定規模

製品	想定サーバー機種と台数	想定同時接続数
IntraBuilder 1.5 Standard	Windows 95/NT×１台	～５ユーザー程度
IntraBuilder 1.5 Professional	Windows NT×１台	～２０ユーザー程度
IntraBuilder 1.5 Client/Server	Windows NT×２台	～５０ユーザー程度

Standard 版は、IntraBuilder エージェントがシングルインスタンスのため、大量のデータ処理には不向きです。しかし、Professional 版と Client/Server 版の IntraBuilder エージェントは、マルチインスタンスで WindowsNT 上に複数起動できるので、スループットを維持したまま大量のデータ処理を行うことができます。また、Client/Server 版の IntraBuilder ブローカーは、複数台の Windows NT マシンに対するディスパッチ機能を持ち、サーバーPC 1 台あたりの負荷を分散するとともに、同時接続ユーザー数を拡大することができます。以下に想定されるシステムの規模とおよそその PC サーバーのスペックを紹介します。

表 2 小規模の場合(同時接続数を 10～20 台に想定)

サーバーＰＣ	Pentium Pro/200 クラス×１台
ＯＳ	Windows NT Server 4.0
データベースサーバー	MS SQL Server 6.5
Web サーバーソフト	MS Internet Information Server
IntraBuilder	IntraBuilder 1.5 Professional（IntraBuilder エージェント×３）
ＲＡＭ	80MB
ＲＡＭの内訳	NT 4.0 が 24MB、SQL Server が 12MB、IIS が 8MB、IntraBuilder が 36MB (12MB/IntraBuilder エージェント×3)

※これらの RAM 数値は最低限の動作を保証するもので、不足分は仮想メモリ(スワップファイル)に展開することになります。もう 1 台の Windows NT マシンを用意し、アプリケーションサーバー機とデータベースサーバー機を分ける事を推奨します。

表 3 大規模 Web システムの場合(同時接続数 50 台を想定)

サーバーＰＣ	Pentium Pro/200 クラス×2 台
ＯＳ	Windows NT Server 4.0 と Windows NT Workstation 4.0
データベースサーバー	Oracle Workgroup Server 7.3
Web サーバーソフト	Netscape FastTrack Server (IntraBuilder 1.5 C/S にバンドル)
IntraBuilder	IntraBuilder 1.5 Client/Server（IntraBuilder エージェント×３×２台）
ＲＡＭ	Server マシン: 92MB Workstation マシン: 108MB
ＲＡＭの内訳	Server マシン: NT 4.0 が 24MB、FastTrack が 32MB、IntraBuilder が 36MB（IntraBuilder エージェント×３） Workstation マシン: NT 4.0 が 24MB、Oracle7.3 が 48MB、IntraBuilder が 36MB

最新情報

Java 開発の A(エース) JBuilder の底力

JDK 1.1 の登場により、Java においてもコンポーネントベースでの開発スタイルが普及した。当初から、積極的な JavaBeans への対応を実現していた JBuilder に続き、JavaBeans への対応を表明している Java 開発ツールも出そろいつつある。

だが、一歩先を歩んできた JBuilder は、今や Java 環境において唯一実践に耐えうる能力を身につけているツールだと言える。Java 開発ツールとして他をリードする JBuilder の底力はいったいどこにあるのか、詳しくレポートする。

Java の将来性をサポートする JBuilder

JBuilder は、100% Pure Java 開発をサポートするビジュアル Java 開発ツールといわれています。ボーランドの 100% Pure Java サポートの姿勢は、Java の将来性を見据えています。

Java と Java に関連するテクノロジーは、驚異的なスピードで進歩しています。これらのテクノロジーは、独自に Java を拡張するのではなく、100% Pure Java サポートの中で実現されています。JBuilder は 100% Pure Java をサポートするので、これらの新しいテクノロジーにすばやく対応できるのです。Java テクノロジーの真の力を引き出す JBuilder の底力を紹介しましょう。

強力な Pure Java 2Way-Tools

JBuilder のビジュアル開発は、Pure Java 2Way-Tools によって実現されます。Pure Java 2Way-Tools は、JavaBeans によるビジュアル開発と、コードの双方向の連動を実現した高度な RAD 環境です。

たとえば、ユーザーインターフェースの設計は、Beans コンポーネントのドラッグ&ドロップによって行なえます。配置したコンポーネントは、コンポーネントインスペクタと呼ばれるウィンドウに表示されたプロパティを設定することで、属性を変更できます。このようなビジュアル操作は、直ちに対応する Java のソースコードに反映されます。配置したコンポーネントのインスタンスを生成するコードや、プロパティを設定するコードはすべて Pure Java のコードであり、ビジュアルを実現するための特殊な中間ファイルなどは用いていません。このコードを修正することはもちろん可能です。

また、コンポーネントを配置するコードやプロパティを設定するコードを記述すれば、それは直ちにビジュアルに反映されます。このような双方向の連動を実現させた Java 開発ツールは、JBuilder だけです。

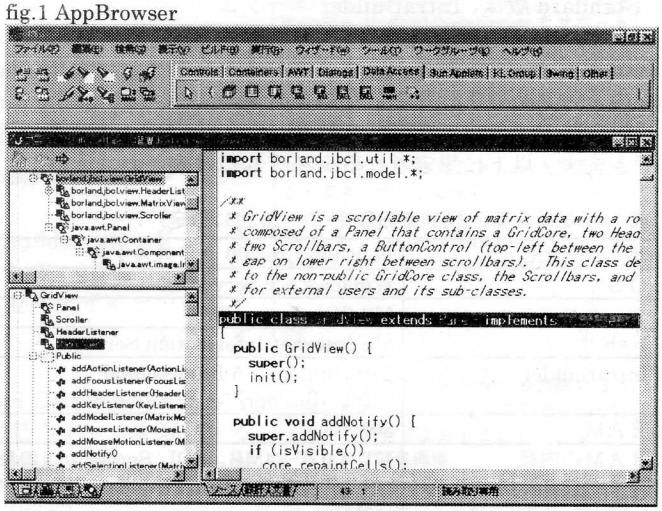
AppBrowser の底力

Java の開発は、すべてビジュアルによって実現できるものではありません。クラスの設計、ビジネスロジックの実装、コンポーネントのカスタマイズなど、多くの場面でコードによる作業が必要になります。JBuilder は一見華やかな開発支援機能を持っていないように見えますが、実は統合開発環境の中心である AppBrowser が強力にコード開発をサポートします。

AppBrowser の構造ペインには、編集中のクラスの構造がリアルタイムに表示されます。ツリー状に表示されたクラス情報には、クラスに含まれるメソッド、変数などがあります。これらの項目をクリックするとソースコードの対応する行にジャンプするので、ソースの内容を把握するのに便利です。また、オブジェクトなどの項目をダブルクリックすると、そのクラスのソースコードを表示します。さらにそのソースに含まれるクラスの情報を見たい場合も、同様の操作で次々にドリルダウンできます。

AppBrowser は、クラスの階層構造を把握するのにも役立ちます。クラス階層はナビゲートペインに表示され、選択したクラスのソースと構造を同時に見ることもできます。

fig.1 AppBrowser



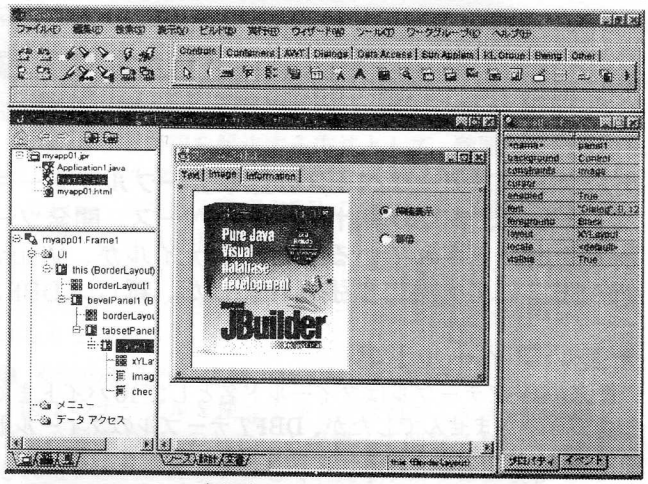
JBCL の実力

JavaBeans は Java の標準コンポーネントアーキテクチャです。コンポーネントは、自律的なプログラム部品としてプログラム開発に使用されます。コンポーネントは、部品内部の動作を隠蔽し、コンポーネントを利用するプログラマに安全で強力な機能を提供します。

Java はマルチプラットフォームでの動作をサポートしていますが、コンポーネントにおいても Java の標準に準拠することで、極めて高い再利用性を実現することができます。JavaBeans 準拠のコンポーネントは、JavaBeans に準拠したビジュアル開発環境で使用でき、さまざまなプラットフォームで動作するアプリケーションを迅速に開発できます。JavaBeans を用いることで、再利用性が高く、オープンな、そして堅牢なアプリケーションを作成できるのです。

JBCL は、ボーランドが提供する高機能な JavaBeans ライブラリです。JBCL は、モデルビューアーキテクチャに基づいた極めて拡張性の高いコンポーネントを提供します。高度なコンポーネントの機能を利用することはもちろん、独自のカスタマイズも容易です。データ構造とビジュアル表示を分離した設計が、柔軟な拡張をサポートします。

fig.2 JBCL によるビジュアル開発



JBuilder では、高機能な JavaBeans を多数搭載するばかりでなく、市販の JavaBeans を登録して使用する機能もサポートしています。登録した JavaBeans は、既存の Beans と同じようにビジュアル開発で使用できます。

BeansExpress の生産力

BeansExpress は、JavaBeans を最も簡単に作成できる機能です。BeansExpress を用いれば、アプリケーションのフレームやアプレットと同様に、ビジュアルな操作で JavaBeans を開発できます。既存のコンポーネントを組み合わせて複合コンポーネントを作成したり、コンポーネントのをカスタマイズして独自のコンポーネントを作成できます。

アプレットとして作成したプログラムを JavaBeans にすることもできます。アプレットとして提供された機能を JavaBeans 化することで、アプリケーションや他のアプレットで再利用することが可能です。

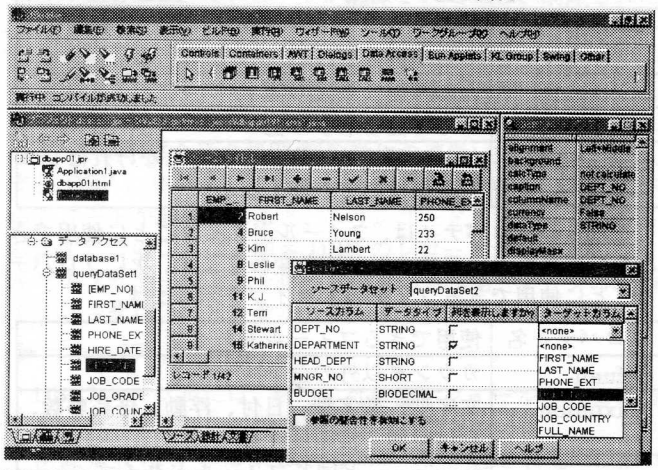
作成した Beans コンポーネントを配布するには、配布ウィザードを使います。配布ウィザードは、クラスの依存関係をチェックして、最適なサイズの JAR ファイルや ZIP ファイルを作成することができるツールです。

DataExpress の開発力

DataExpress は、Java の標準データベース接続アーキテクチャ JDBC に完全に準拠したコンポーネントによって、ビジュアルなデータベースシステム開発をサポートします。DataExpress は、JDBC API をカプセル化したデータアクセスコンポーネントと、データベースに対応した JBCL コントロールによる連係によって、極めてわずかなコーディングによるデータベースシステム開発を実現します。

データアクセスコンポーネントは、データベースへの接続や問い合わせの実行などを制御します。これらのコンポーネントは、JDBC に完全準拠しているため特定のデータベース接続に依存しません。DataExpress は、すべての JDBC に準拠したドライバを使用できる極めてオープンなアーキテクチャです。

fig.3 DataExpress によるデータベースシステム開発



エンタープライズシステム開発への対応力

Java は、CORBA（Common Object Request Broker Architecture）によって、大規模分散オブジェクトシステム開発の扉を開こうとしています。CORBA と Java の出会いによって、Web における分散オブジェクトの可能性が広がりました。

CORBA と Java をいち早く結び付けたのは、VisiBroker for Java です。CORBA 仕様に完全準拠した業界標準の ORB VisiBroker for Java を搭載した JBuilder Client/Server Suite は、エンタープライズシステムを強力にサポートする開発環境です。ボーランドは、CORBA を核とした分散オブジェクトシステム開発をサポートしていきます。言語やプラットフォームに依存しない CORBA と、マルチプラットフォームで動作する Java を用いることで、真の異種接続（Heterogeneous）をサポートします。

What's DBF7 Table?

Visual dBASE 7 タイプテーブルの新機能

デスクトップデータベースの標準テーブルフォーマットとして、多くの表計算/データベース、開発ツールでサポートされている DBF ファイルが、Visual dBASE 7 にて大幅に拡張されました。新しい DBF7 テーブル(以下、DBF7 テーブル)は、Visual dBASE 7 と共通の Borland Database Engine(バージョン 4.51 以降)を搭載している開発ツールでも利用できます。ここでは、DBF7 ファイルの新機能を紹介します。

▶長いフィールド名

従来の DBF テーブルはフィールド名として 10 バイトまでしか使用できませんでしたが、DBF7 テーブルのフィールド名には 31 バイト以下の文字が使用可能になりました。

▶新しいフィールド形式

DBF7 テーブルには、次のフィールド形式が追加されました。

フィールド名	機能
倍長整数型	-2,147,483,647～2,147,483,647
倍精度浮動型	2.2×10^-308～3.4×10^308
日付時間型	YY/MM/DD HH:MM:SS 形式
カウンタ型	1 ～ 2,147,483,647 の整数 ※レコードが作成されると自動的に連番が作成される。ある行を削除しても，ほかの行のカウンター型フィールド値は変わらない。1 テーブルにカウンタ型のフィールドは複数定義できない

▶フィールドプロパティ

DBF7 テーブルには、フィールドごとに次のプロパティが使用でき、入力を制御できるようになりました。

プロパティ名	機能
defualt	新たにレコードが作成されると自動的にこの値が挿入される
maximum	入力可能な最大の値
minimum	入力可能な最小の値
required	true に設定すると値を入力しなければレコードを保存できなくなる

これらのプロパティは、フィールドタイプごとに使用できるものが異なり、デフォルト値も変わります。次にプロパティごとに使用できるフィールドタイプを示します。

プロパティ名	使用できるフィールドタイプ
defualt	カウンタ以外の全フィールドタイプ
maximum	文字型、数字型、日付、浮動、倍長整数、日付時間、倍精度浮動
minimum	maximum と同じフィールドタイプ
required	全フィールドタイプ

なお、数値、論理、浮動、倍長整数、日付時間、倍精度浮動フィールドの「required」以外のプロパティのデフォルト値は「Null」になります。また、論理型フィールドの「defualt」プロパティのデフォルト値は「Null」になります。

▶カスタムフィールドプロパティ

DBF7 テーブルにはフィールド毎にカスタムフィールドプロパティを作成することができます。カスタムフィールドプロパティは、設定されているフィールドを使用する各コンポーネントにカスタムフィールドプロパティの値をコピーし、各コンポーネントのプロパティのように振る舞います。

たとえば、金額を入力するフィールドに「picture」と言う名でカスタムフィールドプロパティを作成し、値を「999,999,999」に設定します。このフィールドを entryfield や grid コンポーネントで使用すればコンポーネントの「picture」プロパティに何も設定しなくても値は位置取りカンマが表示されます。

カスタムフィールドプロパティは、自由にどんなものでも作成できますが、使用するコンポーネントに反映するのは、コンポーネントが持っているプロパティにかぎられます。例えば「enabled」と言うカスタムフィールドプロパティを値「false」で作成し、このフィールドを「entryfield」コンポーネントで使用する場合は機能しますが、「grid」コンポーネントで使用する場合は機能しません。

▶固有インデックス

新しく DBF7 テーブルでは、「重複を許さない」インデックスオプションが追加されました。このオプションを指定してインデックスタグを作成すると、既に同じ値が別のレコードに存在している場合、同じ値を入力し保存しようとすると「キー違反です」とエラーが表示されます。

既に重複した値があるフィールドでこのオプションを指定してインデックスタグを作成すると、重複した値を持つレコードの中の先頭のレコード以外すべて完全に削除されます。インデックスタグを削除し、RECALL コマンドを使用しても削除されたレコードを復活させることはできません。

▶1次キー

DBF7 テーブルには、新しく 1 次キーが作成できる機能が追加されました。1 次キーは、重複した値の保存を制限し、基本的なレコード順を規定します。

Visual dBASE 7 タイプの 1 次キーは、他のテーブル形式の 1 次キーと違い、テーブルをオープンしてもレコード順を決定しません。しかし、1 次キーを定義したフィールドに重複した値を保存することを制限する機能は働きます。1 次キーを使ってレコード順を制限するには、通常のインデックスタグを使う方法と同様に「SET ORDER TO」コマンドや「query」コンポーネントの「rowset」オブジェクトの「indexName」プロパティに指定する方法で行います。

他のテーブル形式では 1 次キーは最初のフィールド以外に設定できないものがほとんどですが、DBF7 テーブルでは

1次キーを作成するフィールドは何番目のフィールドでもかまいません。複数のフィールドで1次キーを作成する場合でも、1次キーを作成するフィールドが連続している必要がありません。

Visual dBASE 7タイプの1次キーでは、降順と昇順を選択することができます。また、通常のインデックスタグと同様に式を含めることもできます。

既に重複した値があるフィールドで1次キーを作成すると、重複した値を持つレコードの中の先頭のレコード以外すべて完全に削除されます。1次キーを削除し、RECALL コマンドを使用しても削除されたレコードを復活させることはできません。

▶参照の整合性

DBF7テーブルには、参照の整合性の規則を定義することができます。参照の整合性の規則は、複数のテーブルをキーフィールドを基にリンクさせて使用する場合のデータ変更の整合性を保ち、テーブルのリンク間の矛盾を解消するために使用します。また、実際にテーブルが使用されるアプリケーションの不具合によるデータの矛盾の発生をテーブルレベルで抑制することができます。

参照の整合性の規則の情報は、DBF7テーブルに保存され、同一のデータベースまたは、同一のディレクトリ内のテーブル間で定義することができます。参照の整合性の規則を定義したテーブルは、別のデータベースやディレクトリに移動した場合でも有効になります。また、DBF7テーブルと他の種類のテーブルとでは参照の整合性の規則は定義できません。

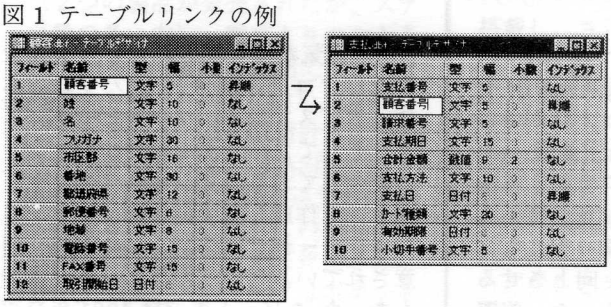
参照の整合性の規則は、インデックスによって整合性が確認されますので、親テーブルのキーとなるフィールドには、必ず1次キーを設定する必要があります。子テーブルのキーになるフィールドには、インデックスタグを作成する必要があります。また、リンクされる子テーブルには、該当するレコードが必ず1レコード以下にする場合(子テーブルに該当レコードがあるか、ないか場合)、子テーブルのキーとなるフィールドには1次キーを設定する必要があります(インデックスタグでは、複数のレコードが作成されるのを監視できないため)。

いかなる参照の整合性の規則を定義した場合でも、該当するレコードが親テーブルに無い子テーブルのレコードの作製や、キーフィールドの変更は禁止され、子テーブルに矛盾したレコードの作製を完全に制御できます。

● 参照の整合性の規則 － 更新動作の制限

たとえば、図1のように2つのテーブルをリンクさせて使用する場合、参照の整合性の規則を定義します。親テーブルとなる顧客.DBF の顧客番号フィールドには、1次キーを設定します。子テーブルとなる支払.DBF の顧客番号フィールドには、インデックスタグを作成しておきます。

図1 テーブルリンクの例



この例では、顧客番号のフィールドをキーとしてリンクを行いますが、何かの理由である顧客.DBF の顧客番号を変更した場合、リンクしている支払.DBF の該当するレコードの顧客番号と一致しなくなり、データは間違ったものになってしまいます。また、誤って支払.DBF の顧客番号を変更してしまった場合、正しい顧客番号に訂正するには大変な労力が必要になります。

このような問題を解決する方法として、参照の整合性の規則(更新動作の制限)を定義します。支払.DBF にリンクしているレコードがある顧客.DBF のレコードの顧客番号フィールド値の変更は制限されます。また、このような状態の顧客.DBF の顧客番号フィールドの値を変更するには、支払.DBF のリンクされているレコードをすべて削除しなくてはなりません。

● 参照の整合性の規則 － 削除動作の制限

図1のテーブルを使ってリンクを行っている場合、親テーブルの顧客.DBF のレコードが誤って削除された場合、削除されたレコードにリンクしていた子テーブルの支払.DBF のレコードは、まったく無意味なレコードになってしまいます。

この問題を解決する方法として、参照の整合性の規則(削除動作の制限)を定義します。支払.DBF にリンクしているレコードがある場合、顧客.DBF のレコードの削除は行えなくなります。また、このような状態の顧客.DBF のレコードを削除するには、先に支払.DBF のリンクされているレコードをすべて削除しなくてはなりません。

● 参照の整合性の規則 － 更新動作の連動

図1のテーブルを使ってリンクを行っている場合、参照の整合性の規則(更新動作の連動)を定義した場合、支払.DBF にリンクしているレコードがある顧客.DBF のレコードの顧客番号フィールド値を変更した場合、自動的に支払.DBF のリンクしているレコードのキーフィールドの値が自動的に更新されます。

● 参照の整合性の規則 － 削除動作の連動

図1のテーブルを使ってリンクを行っている場合、親テーブルの顧客.DBF のレコードが削除された場合、削除されたレコードにリンクしていた子テーブルの支払.DBF のレコードも削除しなければなりません。通常この場合リンクしている子テーブルのレコードを削除し、削除が正常に行われた場合、親テーブルのレコードを削除すると言った方法が使われます。参照の整合性の規則(削除動作の連動)を定義した場合、支払.DBF にリンクしているレコードがある場合、顧客.DBF のレコードが削除されると、このレコードも自動的に削除されます。このことは、顧客.DBF のレコードがいかなるツールやアプリケーションを使って削除されても、データの矛盾を生じさせないことを意味しています。

▶テーブル制約

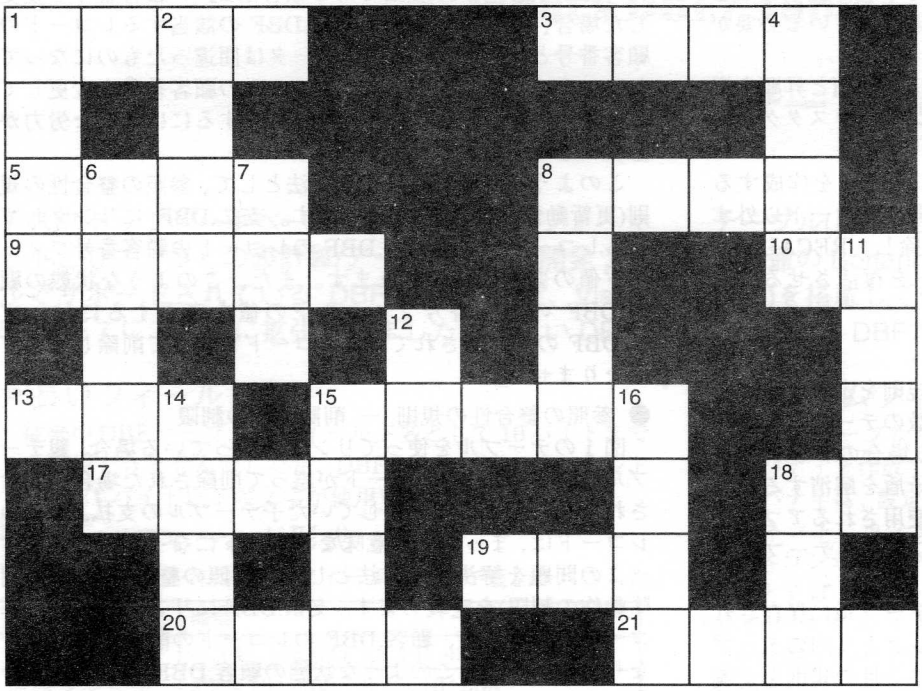
DBF7テーブルには、新たに行単位でフィールドの値を検証するテーブル制約の機能が追加されました。

検証に使用できる式は、フィールド名を含む Visual dBASE の式が使用できます。例えば、注文数のフィールドの値より在庫数のフィールドの値の方が大きくなければならないと言った式を設定する場合、次のように指定します。

注文数 < 在庫数

これにより、条件に合わないレコードの保存が禁止されます。

クロスナンバーパズル



コンピュータ関係の数字だけを使ったクロスワードならぬクロスナンバーパズルです。それぞれのマスには、0～9までの数字がひとつだけ入ります。やや古い題材が多いので、パソコン歴の浅い方には不利かも知れません。

※注意　→[3]という表記は、ヨコ[3]に該当するCPUや機種を指します。

ヨコのヒント

[1]1981年、IBMが発売したIBM-PCに搭載されていたインテル製のCPU、○○○○。i8086と互換性のある命令セットを持っていたが、外部バスが8ビットしかなかったため、8ビットCPUとして分類されることもあった。IBMはパーソナルコンピュータ市場への参入にあたって、仕様を広く公開することで、急速な普及を狙った。これが、今日のIBM-PC互換機というビジネスの始まりにもなった。

[3]→[8]の下位機種として、日本電気が1981年にパピコンという愛称で発売したパソコン、PC-○○○○。下位機種といってもROMカートリッジによるソフトの供給、サウンド機能（PSG、3重和音）など、→[8]を上回る機能も備えており、ホビーユースを中心にヒットした。

[5]→[8]の上位機種として1981年に発売された日本電気の8ビットパソコン、PC-○○○○。→[8]との上位互換性を維持するため、2つのBASIC ROMを搭載していた。メインのBASICでは、プログラム用に利用できる32KBのRAM、640×200のカラーグラフィックを実現する48KB RAM、専用ディスプレイを使った640×400のサポートなど、ビジネスユースを考えたパソコンだったと言えるが、時代とともにゲームマシンとしての性格が強まっていった。

[8]1979年に発売された日本電気としての最初の本格的なパソコン、PC-○○○○（COMPO-BS80を除く）。キャラクタ単位ではあるものの160×100ドットのカラーグラフィック機能を持ち、BASICを標準装備しており、国産パソコンとしてベストセラーとなった。NX以前のPC-9800シリーズのキーボードにある[GRPH]キーは、この頃の名残りとも言えるものである。

[9]→[20]のVMシリーズに搭載された日本電気製CPU、μPD○○○○○。V30とも呼ばれ、ごく一部の命令を除き、→[15]との互換性を持っていた。その後、上位のCPUも開発されたがパソコンとしてはLTシリーズにV50が採用された以外には使われず、制御機械などで使われることが多かった。

[10]富士通がFM-Rシリーズの前に発売していた16ビットパソコン、FM-○○β。携帯型のFM-○○πもあった。

[13]インテル初の32ビットCPU、iAPX○○○（または、80○○○）。もっとも、OS自身の32ビット化がさほど進まず、仮想86モードを使ったEMSエミュレータやプロテクトモードとリアルモードを容易に切り替えられる仕組みを使った環境切り替えなど、16ビット環境の利便性を向上させるために使われることが多かった。実際に32ビットOSが普及する頃には、ほとんど姿を消してしまった。

[15]i8086との上位互換性を持った16ビットCPU、i○○○○○。即値によるシフト・ローテイト命令、レジスタ全体のPUSH/POPなど、拡張された命令セットを持っていたがプロテクトモードはなく、さほど普及しなかった。

[17]モトローラ社初のCPU、MC○○○○。単純な命令セットを持っていたが、CP/Mというosを持ち、ソフトが充実していたi8080に敗れ去った。

[18]富士通がFM-8の上位機種として1982年に発売した、FM-○○。ST、AD、EXの3機種があり、FM-8同様、メインCPUとしては↓[14]を使っていたが、EXは→[1]とCP/M-86を搭載して16ビット対応を果たしていた。しかし、モトローラ製ではなくインテルCPUを使ったことで↓[14]ファンには不評だった。しかし、歴史を振り返れば、この選択が間違いだったとは言えないだろう。

[19]沖電気から発売されたオールインワンパソコン、if○○○。パソコン本体、ディスプレイ、フロッピーディスク、そしてプリンタをひとつのマシンに統合しており、ビジネスアプリケーションを作成しやすいよう、ファンクションキーがディスプレイの下部に用意されているといったユニークな機能もあった。8ビット機では1981年発売のモデル30、16ビット機では1983年発売のモデル50が有名。

[20]日本電気が 1982 年に発売し、パソコン界に築いた金字塔、PC-○○○○。→[5]との BASIC レベルでの互換性を持った 16 ビットパソコンで、i8086（5MHz）、128KB RAM、640×400 のカラーグラフィック（8 色）など先進的な機能を搭載し、当時一般的だったカセットインターフェースをオプションとするなど、ビジネス向けであることをアピールした。テキスト VRAM や漢字 ROM など、この頃の設計は NX 以前の PC-9800 シリーズまで受け継がれている。

[21]日本電気が 1982 年に発売したハンドヘルドコンピュータ、PC-○○○○。テンキーやファンクションキー、N20-BASIC など、パソコンに近い機能を搭載していたものの、当時はコンパクトなポケットコンピュータが流行しており、あまりヒットしなかった。

タテのヒント

[1]インテルの CPU、i8086 の浮動小数演算を高速化するためのコプロセッサ、i○○○○。当時、CPU 自身はレジスタや演算命令が整数に限られており、高速かつ精度の高い浮動小数演算を実行するために必要だった。また、コプロセッサのないマシン上で浮動小数演算を実行させるためにエミュレータという仕組みを提供している開発ツールも多かった。486DX 以上では、この機能は CPU 自身に組み込まれている。

[2]日本電気が 1983 年に発売したノート型コンピュータ、PC-○○○○。京セラの OEM であり、40 桁×8 行（240×64）の LCD パネルを持ち、フルキーに近いキーボードを備えていた。しかし、現在のように漢字を表示することは難しく日本のビジネスでの利用に耐えるものではなかった。

[4]シャープが 1980 年に発売した最初のポケットコンピュータ、PC-○○○○（２つの製品が発売されたが、ステップ数の多い方）。電卓よりやや大きめのサイズで BASIC が使えるという画期的なものだった。しかし、CPU が 4 ビットであったため数値計算の処理速度は電卓よりも遅く、BASIC プログラムも for 文を 100 回繰り返すのに 23 秒もかかるほどだった。この製品の発売以後、一時期はカシオに急追されたが（↓[7]参照）、シャープはポケットコンピュータ市場での最大のヒットメーカーとなり、この実績が現在の電子手帳の繁栄にも結びついている。

[6]→[15]の上位 CPU、i○○○○○。16 ビットのプロテクトモードによって 16MB までのメモリ管理やマルチタスク OS に必要な機能を備えていたが、当初は高速な i8086 として使われることがほとんどだった。この CPU に対応した OS/2 は売れ行きが伸びず、本格的に使われ始めたるのは、Windows 3.0 の登場を待たねばならなかった。ただし、Windows 3.0 が登場したころには→[13]があり、これを使ったエンハンスモードの方が好まれた。

[7]カシオのポケットコンピュータ、PB-100 の後継機種、PB-○○○。1981 年に発売された PB-100 ではオプションだった 1KB のメモリが標準実装されていた。PB-100 は、カシオ初のポケットコンピュータであり、↓[4]に比べて桁違いの高速性と関数電卓に匹敵する数値演算機能が売りであり、ヒットした。しかし、BASIC がやや特異であるなど、速度が向上したシャープ製のポケットコンピュータに比べて、後続の機種は競争力を失っていった（↓[4]参照）。カシオは後年 FP-1100 という数値計算にすぐれた低価格パソコンを発売したが、スクロール速度が遅いといったマイナス面もあり、ヒットには至らなかった。他社に遅れて MSX パソコンもリリースしたが、MSX 自身の衰退とともに自然消滅。現在は Windows CE 対応の CASSIOPEIA で巻き返しを図っている。

[8]現在、CPU の圧倒的なシェアを誇るインテルの初代 8 ビット CPU、i○○○○。これ以前に、インテルは 4 ビット CPU である i4004 を送りだし、これは世界初のマイクロプロセッサと言われている。また、急速に普及したのは 1974 年に発表された改良型の i8080（i8080A）である。

[11]→[3]の上位機種として、1984 年に日本電気から発売されたパソコン、PC-○○○○。フロッピーディスクを搭載し、さらに音声合成を組み込んでいた。また、1000 文字程度の教育漢字を ROM で提供していた。

[12]↓[15]の設計者がザイログ社に移籍して開発した CPU、Z-○○。i8080 との上位互換性を持ち、実行速度が高速だった。裏レジスタや拡張レジスタの存在も忘れ難い。μPD780 といった互換 CPU も含め 8 ビットパソコンの主流として、圧倒的なシェアを獲得し、→[3]、→[5]、→[8]でも採用された。

[14]モトローラから発売された CPU、○○○○。→[17]の後継であり、マルチユーザー、マルチタスクを実現した OS-9 というすぐれた OS も提供された。賢くなった電卓と言われたインテル系 CPU と違い、小さくなった汎用機とも言われ、命令体系は洗練されていたが、80 系 CPU を凌駕することはできなかった。モトローラは、後に内部 32 ビット、外部 16 ビットの 68000 をリリースした。68000 から 68040 までが Macintosh に採用され、一時代を築いたものの、現在では IBM との共同開発による PowerPC にその座を譲っている。

[15]↓[12]で一世を風靡したザイログ社が 1979 年にリリースした 16 ビット CPU、Z-○○○○。しかし、↓[12]と違い、ほとんど普及しなかった。16 ビット時代には、ふたたびインテル対モトローラの時代を迎えるのである。

[16]Motorola をスピンアウトした技術者が設立した MOS Technologies の代表的な CPU（MPU）、○○○○。パイプライン処理や 10 進演算用のフラグなど、画期的な機能を持ち、この CPU を搭載したあるパソコンは一世を風靡した。ファミリーコンピュータで使われている CPU は、この CPU との下位互換性を持っている（10 進演算などはない）。

[18]→[20]に続いて日本電気が発売した、PC-○○○。京セラの OEM であり、MS-DOS やマウスの標準装備、512×720 のグラフィック、縦横切り替え可能なディスプレイなど、進んだ機能を実現していたが、高価であることやソフトウェア不足により普及は進まなかった。

プレゼント

答えの中で、０～９のうちまったく使われない数字がひとつだけあります。答えを以下の URL にて応募してください。正解者の中から抽選で 10 名様に、「C++Builder 3 テレホンカード」を差し上げます。正解者が 10 名に満たない場合は、全正解者で山分けにします。

（締め切り：９８年４月３０日）

http://www.borland.co.jp/amagazine/ans.htm

XMAGAZINE（仮称）

１９９９年４月号予告▶４月１日発売　定価０円

探究　ボーランドのエンタープライズ戦略

その実績と生産性を再評価

特集　ボーランドの開発ツール、新バージョンの魅力

緊急レポート　８ヶ月で対応できる2000年問題

歴史探訪　Turbo C

好評連載　復刊の挨拶・休刊の挨拶

※この部分の記載は冗談です。購読のお申し込みなどは受け付けておりませんので、あしからずご了承ください。

編集後記・休刊の挨拶

性懲りもなく、今年も発行しました。同じことを毎年書いているような気もしますが、ほんとうに今回が一番大変でした。なぜ今、最初のトピックが「アセンブラ」なのか疑問に思われた方もいらっしゃると思います。今回で４号めとなる「C MAGAZINE」誌の裏表紙広告企画ですが、第１号が「BC MAGAZINE」、第２号が「D MAGAZINE」、第３号が「CB MAGAZINE」でした。そこで、「BC から D になり CB と戻ってきたから来年は A でアセンブラの特集でもしよう」と昨年の編集後記に書いたのです。それ以上の理由はありません。C++Builder 3 に MMX 対応アセンブラが添付されるということは、昨年の時点ではまったくの未定事項でした。ですから、最初のトピックなのにたいして中身がありません。期待された方、（もしいらっしゃったら）ごめんなさい。今年のタイトルの第二候補は「J MAGAZINE」でした。来年は、発行するかどうかも含めて未定です。いつものことながら、「C MAGAZINE」編集部の方々には感謝しております。

実のところ個人的にはアセンブラは好きな方です。しかし、アセンブラといえば記事で紹介した moritan（あの森田将棋の森田和郎氏）の強烈な印象があり、私自身が「究極」なんてほんとうに恥ずかしくて書けません。企画段階では「アセンブラ再考」というおとなしいタイトルだったのに、途中でうっかり変更されてしまいました。今では、moritan の世界に触れることは難しいのですが、MIX の assembler 会議における moritan のプログラムにはいつも驚嘆させられていました。Delphi の記事といい、私も古き良き時代を懐かしむオジサンになってきたのかもしれません。温故知新、新しいものを追いかけるだけではだめなので、と言い訳しておきます。

なお、アセンブラプログラミングに興味のある方は、インテルのホームページ（http://www.intel.co.jp/）で日本語のドキュメントが入手できます。プログラミングテクニックについては http://www.sonycsl.co.jp/person/fnami/hobby.htm が詳しいようです。

また、クロスナンバーパズルは、以前から考えていたアイデアなのですがもともとパズルを作るのは苦手なので、対称性がないばかりか、黒マスがカタマリになって残っています。愛読するニコリとは雲泥の差。お恥ずかしいばかりで申し訳ありません。どこかに 8086 を入れたかったのですが時間切れにより、あえなく断念しました。　(O)

A MAGAZINE　９８年４月１日号
平成１０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は、弊社インフォメーションサービスセンター、テクニカルサービスセンターでのお問い合わせはお受けしておりません。内容についてのご質問はご遠慮ください。
本誌で紹介している製品のカタログについては、以下までお問い合わせください。

〒151-0073　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

価格 ¥0[税込み]

9M0000T010