ISSN 1346-650X

九州大学 情報基盤センター

広報

学内共同利用版
2002年 Vol. 2 No. 3

目次

Computing and Communications Center
Kyushu University

www.cc.kyushu-u.ac.jp

情報基盤センターサービス機器一覧

■ 教育用システム

ホスト計算機	ah.cse.ec.kyushu-u.ac.jp
WWW サーバ	www.cse.ec.kyushu-u.ac.jp
ウェブメーラー (GraceMail)	https://mailserv.cse.ec.kyushu-u.ac.jp/

■ KITE 関連機器

教職員向けメールサーバ	mbox.nc.kyushu-u.ac.jp

■ リモートアクセスサービス

電話番号	制御手順	種別	通信速度	回線数
092-642-7341	PPP	モデム	最高 56Kbps	64 回線
		ISDN	同期 64Kbps	92 回線
		PSH(PIAFS)	32/64Kbps	32 回線

特集「Windowsの統合開発環境」

井上 仁*

情報基盤センターの前身施設の一つである情報処理教育センターが設立されたのが1977年であるから，九州大学で本格的な情報処理教育の場が提供されて，ちょうど25年が経過したことになる。もちろん，本学での情報処理教育は，それ以前から開始されており，1972年当時の調査では，なんらかの形で情報処理教育を始めていると回答のあった学部は，教育・経済・理・薬・工・農の六学部である[1]。とはいえ，当時は受講学生数も少なく，実習の機会にもほとんど恵まれていない状況であった。現在では，全学教育の情報処理科目として「情報処理基礎演習」を1年生全員が履修し，全学生が情報リテラシやプログラミングといった情報教育の機会を得ている。

ソフトウェアの開発にはさまざまな工程があるが，その中心となるのは，プログラム作成や編集，コンパイル，実行，デバッグといった作業であろう。これらは，相互に深く関係し繰り返し行なわれる作業であるため，それぞれ独立したツールを利用する場合には，効率よくソフトウェアを開発することができない。

統合開発環境は，プログラムの作成や編集，コンパイル，実行，デバッグといった一連の作業を行なうツール類を一つのアプリケーションで行なうための統合されたシステムである。統合開発環境では，単に各ツール類を同一アプリケーションで利用できるだけでなく，それらが相互に連係することにより，ソフトウェアを効率的に開発・管理することが可能となる。

統合開発環境は，特にWindows環境で充実しており，各ツールに加えて，以下の機能を備えているものが多い。

ビジュアル開発機能

Windowsのソフトウェアの多くは，ウィンドウ内にボタンやリストボックス等の部品を配置し，それらのプロパティ(属性)の設定やイベント(動作)の記述により，プログラムを作成していくのが一般的である。これらのソフトウェア部品の配置は，プログラムコードとして直接記述することも可能であるが，視覚的な操作により，効率的な作業が可能となる。

*情報基盤センター研究部　E-mail : jin@cc.kyushu-u.ac.jp

[1]大槻説乎: "情報処理教育センターの発足まで", 情報処理教育 広報, Vol.1, No.1, pp.2-5, 1978, 九州大学情報処理教育センター

入力支援機能

ソフトウェア部品のプロパティや，イベントの記述に用いる関数・手続き・メソッド等は他数あるため，すべてを把握するのは困難である。それらの一覧表示機能や入力補完機能，また関数等の引数の型宣言表示機能を備えたものもある。

プロジェクト管理機能

ソフトウェアは一般に複数のプログラムから構成され，また関連するファイルも数多いため，それらを一つのプロジェクトとして統一的に管理する機能がある。

運用環境への配布機能

ソフトウェアは，開発に利用した以外のコンピュータ上で運用することが多い。そのため，実行プログラムファイルや動作に必要となるデータファイル等を運用環境へ配布するための機能がある。

本特集では，Windows 上の代表的な統合開発環境について，簡単なプログラム例を用いながら紹介していく。これらの統合環境の多くは，情報基盤センターの教育用システムのパーソナルコンピュータに設定されているので，興味ある方は記事を読むだけでなく，実際に統合環境を使用されることをお勧めする。

- Delphi 6 でプログラミング入門
- Visual C++ 利用法入門
- Visual Basic 入門
- JBuilder 入門

Delphi6でプログラミング入門

峯　恒憲[1]，正代　隆義[2]，菅沼　明[3]

1. はじめに

「問題を分析し，その解法を導きだす能力」は，どの分野にとっても必要な能力である．プログラミングを学ぶ目的は，まさに，そのような問題解決能力を身につけることにある．そのため，本学の全学教育科目の中の情報処理基礎演習には，コンピュータの操作に関するリテラシー項目だけでなく，プログラミングに関する項目が含まれている．本稿では，そのようなプログラミングを学ぶためのツールとして，今年から学習目的のために無償で利用できるようになったDelphi6を利用してのプログラムの作成方法を紹介する[1]．

Delphi6は，プログラミング教育用言語として開発されたPascal言語にオブジェクト指向[4]の概念を取り入れたObject Pascal言語の開発環境である．ここでいう開発環境とは，プログラムの編集を行うためのエディタや，そのプログラムを機械語に翻訳する高速なコンパイラ，プログラムエラーの発見と修正を支援するデバッガーなどを提供するツールの集合のことをいう．Delphi6では，Windows上でのプログラム開発を容易にするために，視覚的プログラミング環境を提供している．

Delphi6のダウンロードやライセンス情報の取得は，
http://reservoir.cc.kyushu-u.ac.jp/delphi/
から行うことができる[2]．

2. プログラムの作成手順

プログラムを作成する手順を記すと以下のようになる．

(1)　問題を分析し，解法を見つける．
(2)　データの入出力関係を明確にする．
(3)　大まかなプログラムの形を作成する．
(4)　実際にプログラムを何らかの言語で記述する．
(5)　記述したプログラムの修正，打ち直し，保存，検査を行う．
(6)　実行する．

実際にプログラムを作成する時には，(5)の部分を何度も繰り返すことになる．また検査に通り，(6)の実行ができたとしても，正しい結果が得られるとは限らない．(5)番目に戻る小さな修正ですむ事もあれば，エラーの種類によっては，再

[1]大学院システム情報科学研究院　E-mail: mine@is.kyushu-u.ac.jp
[2]大学院システム情報科学研究院　E-mail: shoudai@i.kyushu-u.ac.jp
[3]大学院システム情報科学研究院　E-mail: suga@is.kyushu-u.ac.jp

[4]オブジェクト指向とは，扱うべき情報と，それに関連する処理とを一塊のオブジェクトとして，そのオブジェクトをプログラムで扱う単位とすることで，モジュール化を容易にし，プログラムの開発や保守管理を容易にする概念である．しかし，ここでは触れない．

度，最初(１番目)から見直さなければならないこともあり得る．ただ，(1)からの手順通りに分析した結果を記録しておれば，見直す際の労力は少なくてすむため，この手順に従いながら，途中結果を必ず記録として残していくことが重要である．

3.　Delphi 統合環境の起動

Delphi の起動には，下の２つの方法がある．

(1)　スタートメニューからの起動：[プログラム]→[Borland Delphi 6]→[Delphi 6]をクリックすると起動する．

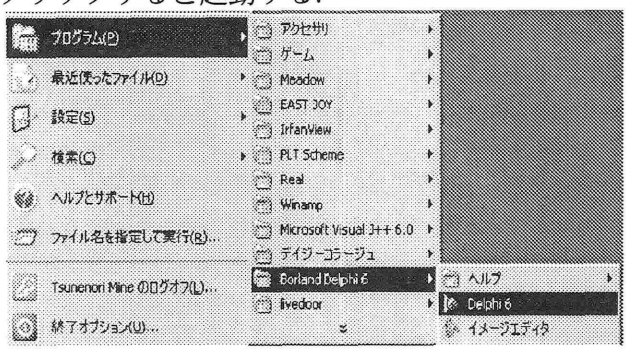


図 1　Delphi6 の起動

(2)　Delphi ファイルからの起動：Delphi で作成したファイル（拡張子は dpr）がある場合は，そのファイルのアイコン（例えば，右の newton.dpr）をダブルクリックすると起動する．この場合，この後すぐに「5. エディタを使用してプログラムを入力」に進む．

Delphi をスタートメニューから起動すると，図２のように多くのウィンドウが開く．

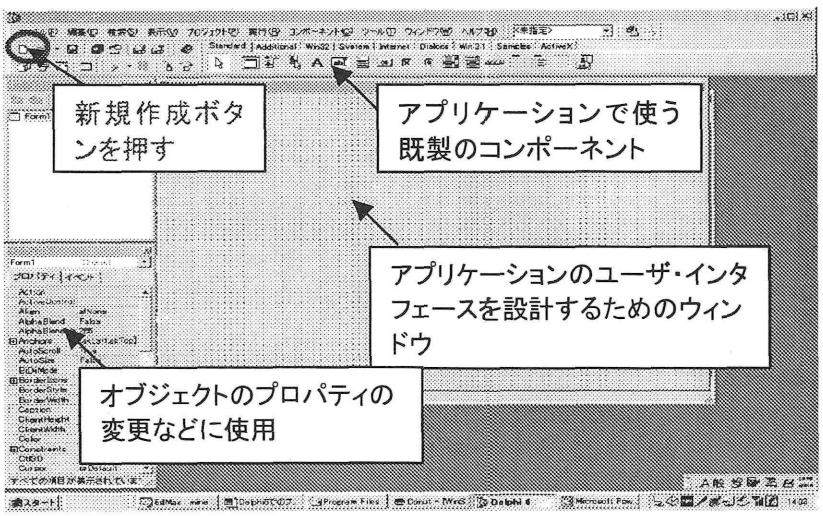


図 2　Delphi が起動した初期画面

これらは Windows 上でボタンやフィールドなどを持つウィンドウを作成するために利用されるものである．しかし，本稿では簡単のためこれらは利用せず，コンソールウィンドウ上で実行を行うプログラミング作成を例とする．そのためのウィンドウを開くには，「ファイル」→「新規作成」→「その他」→「コンソールアプリケーション」（図 3）を選ぶか，図 2 の左上の「新規作成ボタン」を押し，「コンソールアプリケーション」（図 3）を選ぶ．

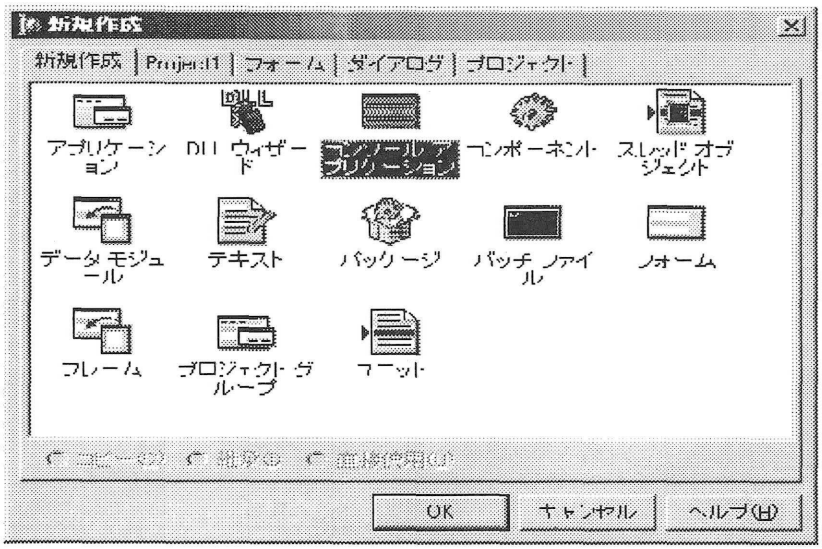


図 3 新規作成ウィンドウ

4. 新規作成のウィンドウ

ウィンドウの新規作成の手順は，以下の通りである．

(1) 新規作成ウィンドウ内の「新規作成」タブをクリックする．表示されるアイコンの中の「コンソールアプリケーション」（図 3)を選択する．[OK] ボタンをクリックする．

5. エディタを使用してプログラムを入力

新規作成されたウィンドウの例を図 4 に示す．ウィンドウは直接，プログラムを編集できるエディタになっている．エディタのウィンドウの中には，プログラムを構成する概形が最初から記述されている．

プログラムは，この雛形を埋めるように，必要な部分を入力することで，完成させる．この時，以下のことに注意して，プログラムの入力を行う．

- 入力した文字はカーソル位置に挿入される．
- 新しく行を作りたいときには，行の先頭にカーソルを移動させて，[Enter] キーを押す．
- 1 行入力したら，行の終わりで[Enter]キーを押す．
- プログラムを入力しているときに Delphi が注釈を表示することがあるが，気にせずに入力する．

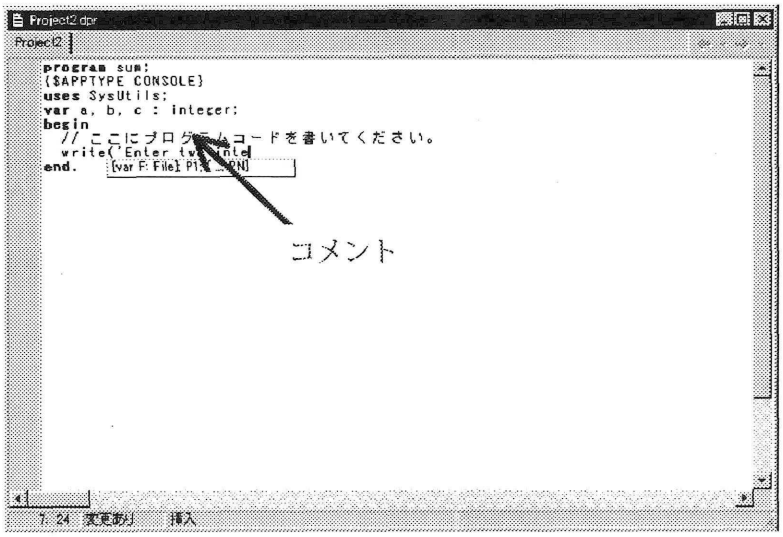


図 4 コンソールアプリケーション作成エディタ

6. 入力するプログラム

```
program sum;
{$APPTYPE CONSOLE}
uses SysUtils;
var a, b, c : integer;
begin
  // ここにプログラムコードを書いてください。
  write('Enter two integers: ');
  readln(a, b);
  c := a + b;
  writeln('sum = ', c);
  readln
end.
```

図 5　2数の和を求めるプログラム sum

「2つの整数を入力し，その和を計算する」Pascal 言語で記述されたプログラム sum を入力してみよう．このプログラムでは，まず，2つの整数を入力するように促すメッセージ’Enter two integers: ’を画面に表示し(write)，キーボードから2つの整数を入力させる．入力された数は，2つの変数 a, b に読み込まれる(readln)．変数 a と b の和を計算し，その値を変数 c に蓄える(c:=a+b)．そして，その結果(c)を表示する(writeln)．　最後の readln は，メッセージを表示

するウィンドウが，Enter キーを押すまで消えないようにするために付けている．図中の//の後の文字は，プログラムに書かれたコメントである．またプログラムを書く時には以下の決まりがあり，これを守らないとエラーになる．

- プログラムは，コメントと出力文字列以外の場所には，2 バイト文字（全角文字）を書いてはいけない．

7. 打ち間違いの修正

打ち間違いを訂正するために，「一文字消去」，「領域を選択しての削除」，そして「作業の取り消し」をよく利用する．

一文字消去

- BS(Back Space)キーを押す．
- カーソルの直前(左側)の文字が消える．

領域削除

- マウスカーソルを，削除したい最初の文字に合わせる．
- マウスの左ボタンを押したまま，削除したい所まで動かしボタンから離す．
- BS キーまたは削除キーを押すと，指定された領域が消える．

作業の取り消し

たった今，行った編集を取り消したい場合には，[編集] → [元に戻す]をクリックする．

8. プログラムの保存

プログラムを保存する手順は，以下の２つからなる．

(1)　メニューバーの[ファイル]→[プロジェクトに名前を付けて保存]を選ぶ．

(2)　ファイル名を，プログラムの内容を表す名前（例えば「sum」）に変更して，[保存]ボタンをクリックする．この際，ファイルの種類は変えてはならない．

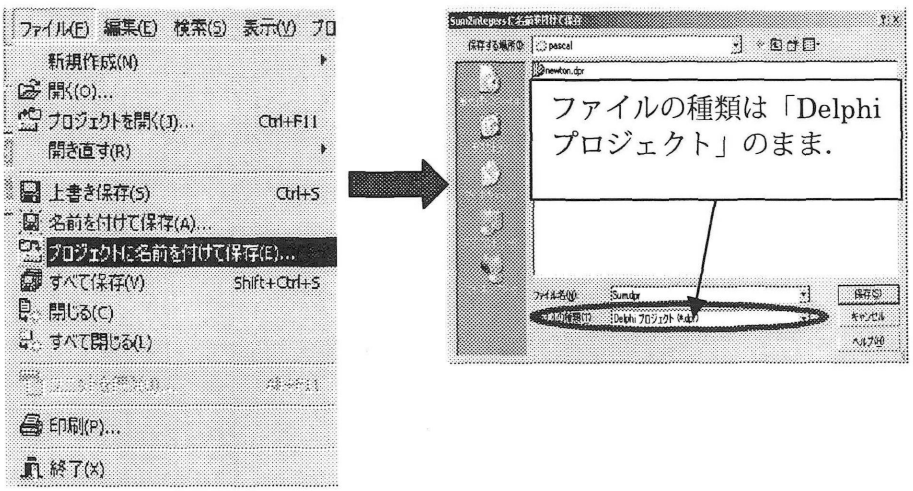


図 6 プログラムの保存

Delphi では，プログラムの開発を，**プロジェクト単位**で行う．そのメインとなるファイルが，**プロジェクトファイル**である．コンソールアプリケーションを作成する場合には，このプロジェクトファイルを，直接読み書きすることになる．

9. プログラムの検査・実行

作成したプログラムは，必ずしも正しいとは限らないので，何度も「**検査→実行**」を繰り返したり，実行結果によっては，最初からプログラムを見直すことも必要となる．

(1) メニューバーの[実行]→[実行]をクリックして，作成したプログラムのコンパイルを行う．
(2) コンパイル中にエラーが見つかれば，計算機が報告してくれる．この検査で見つかるエラーを「**文法的エラー**」と呼ぶ．
(3) エラーを訂正して，再度コンパイルする．
(4) エラーがない場合は，自動的に実行してくれる．
(5) 実行結果が正しいかを人が調べる．正しくなければ，プログラムを訂正する．このときに見つかるエラーを「**意味的エラー**」と呼ぶ．

10. コンパイラからのメッセージ

コンパイラが出すメッセージは，日本語で表示される．メッセージの書式は「**プロジェクトファイル名(行番号):エラーメッセージ**」である．

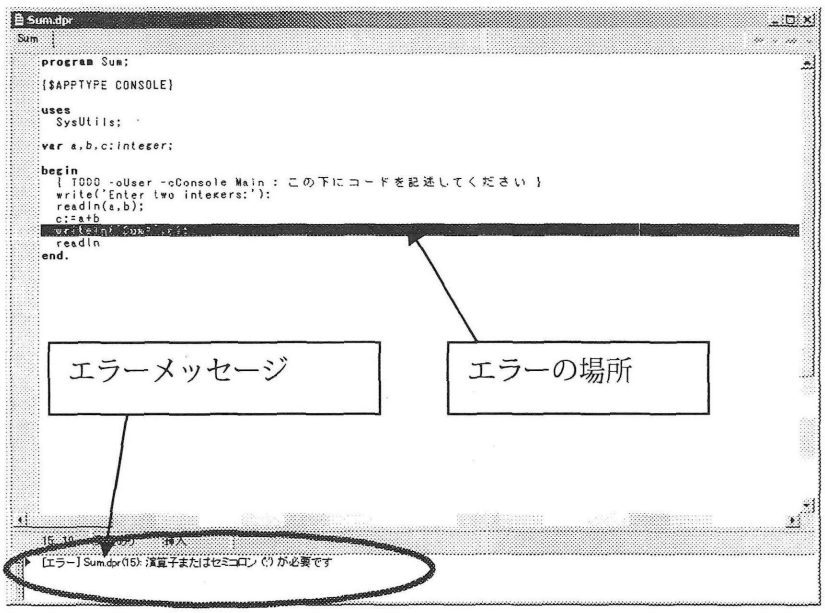


図 7 エラーがある場合のウィンドウの例

以下に，よく出るメッセージの例を記す．

- sum.dpr(15)：演算子またはセミコロン（;）が必要です
- sum.dpr(14)：':='が必要な場所に'='があります
- sum.dpr(14)：未定義の識別子：'c'
  変数'c'が宣言されていない．変数名の打ち間違い．
- sum.dpr(6)：識別子の多重定義：'sum'
  同じ名前の変数を2回以上宣言している．プログラム名と変数名で同じ名前を使用している．

11. プログラムの実行

プログラムを実行する方法は2通りある．

(1)　統合環境から実行する．

- メニューバーの[実行]→[実行]をクリックする．
- コンパイル中にエラーがなければ，実行される．

(2)　統合環境を使わずにプログラムを実行する（実行可能ファイルのアイコンをダブルクリックする）．

- プログラムをコンパイルしてエラーが出なかったら，「sum.dpr」を保存したフォルダに「sum.exe」というファイルができている．（.exeは拡張子として表示されない場合もある）
- コンパイルでエラーがないにもかかわらず「sum.exe」のアイコンが表示されない場合は，[表示] → [最新の情報に更新] をクリックする．

12. プログラムの印刷

プログラムは，Delphi の統合環境から，次の手順で印刷することができる．

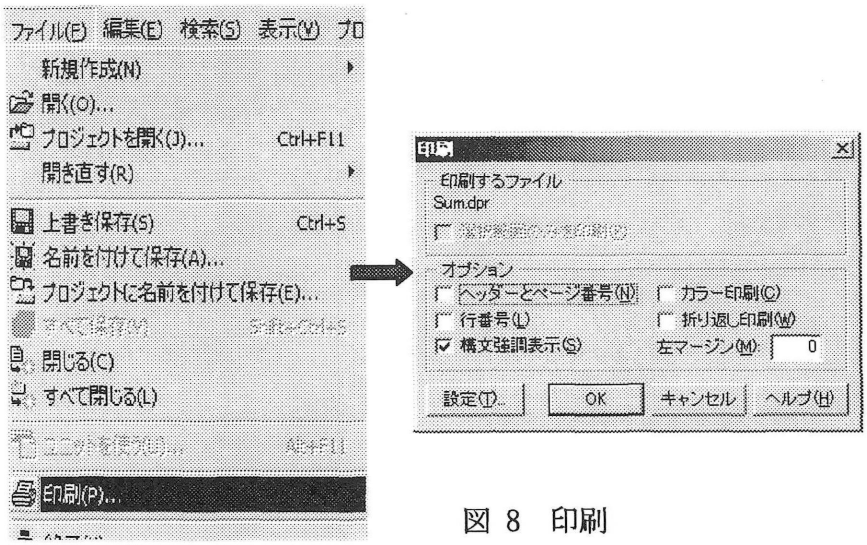


図 8　印刷

(1)　メニューバーの[ファイル]→[印刷]をクリックする.
(2)　印刷のウィンドウが表示されるので，好みに合わせて設定し，[OK]ボタンをクリックする.
(3)　割り当てられたプリンタに出力される.

13. Delphi 統合環境の終了

以下のどちらかで，統合環境を終了することができる.

- メニューバーの[ファイル]→[終了]をクリックする.
- メニューバーのあるウィンドウの右上にある[×]ボタンをクリックする.

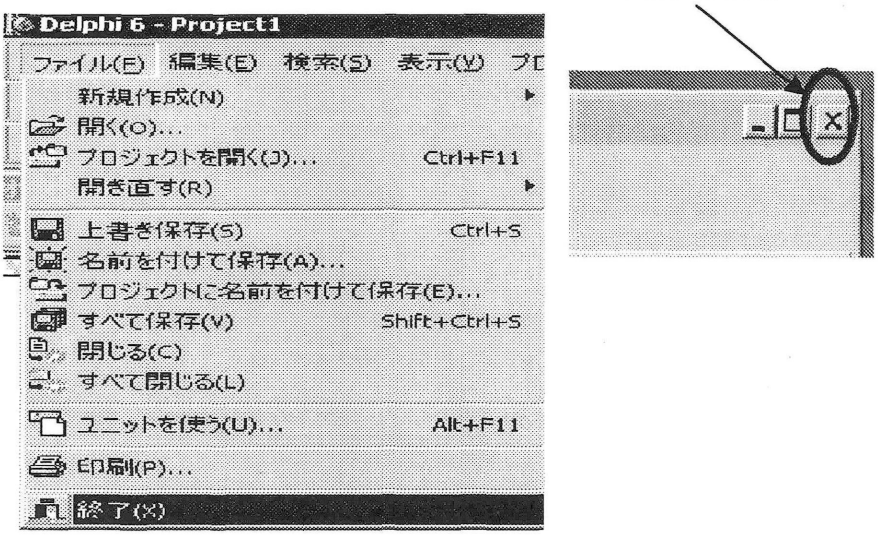


図 9　Delphi の終了方法

参考文献

[1] 情報処理基礎演習講義資料, 第 4 回 Pascal プログラミング入門, 2002, http://cl.rc.kyushu-u.ac.jp/Literacy/h14/ohp/index.html
[2] 井上仁, Delphi6 のダウンロードサービスについて，九州大学情報基盤センター広報, pp.30-31, Vol.2, No.1, 2002

Visual C++利用法入門

鈴木 孝彦[1]

本学の教育用システムのPC上では，Microsoft 製のC++プログラム開発環境「Microsoft Visual C++ (R) 6.0」が利用できます[2]。この記事では，C++ を使って Windows 上の高性能なアプリケーションを作ってみたいけれど使い方がわからない，なんだか知らないけど面白そうだから Visual C++ を使ってみたい，といった方を対象に，Visual C++ 上で簡単なプログラムを動作させる方法を紹介します。

1. はじめてプログラムを入力して実行する

```
#include <iostream>
using namespace std;

int main()
{
    cout << "Hello World!" << endl;
    return 0;
}
```

図 1 入力するプログラムテキスト

図 1 は，"Hello World!" という文字列を出力するための，簡単な C++ プログラムです。このプログラムを Visual C++ を使って入力，実行してみましょう。手順は次のようになります。

Visual C++ 開発環境の起動

まず，Windows のスタートボタンから「プログラム」→「Microsoft Visual Studio 6.0」→「Microsoft Visual C++ 6.0」を選択して，Visual C++ アプリケーションを立ち上げます。Visual C++ では，プログラムテキストの入力，チェック，実行，その他のプログラム開発に必要な作業が行えます。Visual C++ のようなアプリケーションをプログラムの“開発環境”と呼びます。

---

[1] 情報基盤センター　E-mail:suzuki@cc.kyushu-u.ac.jp

[2] 最新版の Visual C++ .NET でも基本的な操作は共通しています。

プロジェクトの新規作成

Visual C++ では作成したいプログラム各々について，名前をつけた“プロジェクト”を作ります。プログラムに必要な各種の設定をプロジェクトに登録することが，Visual C++ でのプログラム作成作業です。**プログラムを作るためにはプロジェクトを作る**，と覚えてください[3]。

Visual C++ のメニューから「ファイル」→「新規作成」を選択します。すると**図 2** のようなダイアログが現れて“新規作成”の選択画面となります。

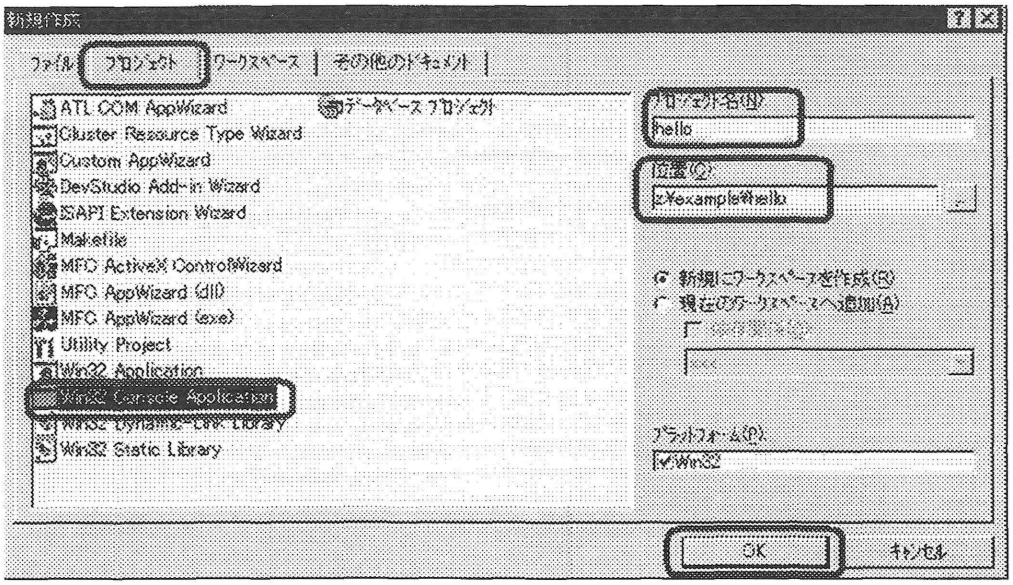


図 2 プロジェクト新規作成選択画面

ここで，「プロジェクトタブ」を選択し，「Win32 Console Application」 を選択します。さらに「プロジェクト名」と「位置」を入力します。ここでは「プロジェクト名」を hello 「位置」を Z:¥example¥hello と指定しました。プロジェクトは，「位置」で指定したディスク上のフォルダに保存されます。

このプロジェクトは“コンソールアプリケーション”，つまり，文字が表示出来る画面とキーボードだけを使うプログラムです。C++ 言語の入門書に掲載されている初歩的なプログラムの多くはコンソールアプリケーションです。

「OK」ボタンを押すと，**図 3** に示すようなウィンドウが表示されますので，「空のプロジェクト」を選択し，「終了」ボタンを押します。Visual C++ がさらにもう一つウィンドウを表示しますのでもう一度「OK」ボタンを押してください。

---

[3] プロジェクトを複数含むワークスペースというのもありますが，省略します。

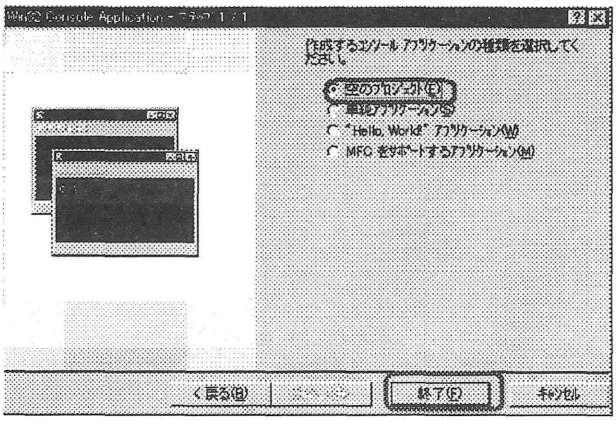


図 3 空のプロジェクト選択画面

ソースファイルの新規作成

図 1 のプログラムを「hello」プロジェクトに登録します。 Visual C++ のメニューから「プロジェクト」→「プロジェクトへ追加」→「新規作成」を選択します。すると**図 4** のようなダイアログが現れますので，「ファイルタブ」を選択，続いて「C++ ソース ファイル」を選択し，「ファイル名」と「位置」とを入力して「OK」ボタンを押します。ファイル名は **hello.cpp** 位置は **z:¥example¥hello** とします。

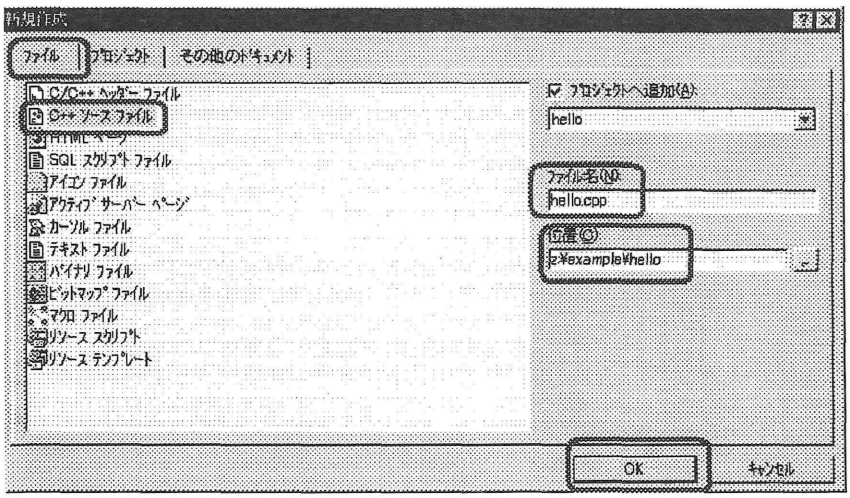


図 4 ソースファイルの新規作成

「OK」ボタンを押すとテキストの入力が可能になりますので，**図 1** のプログラムをキーボードから入力します。これでプログラムテキスト（ソースファイル）が登録されました。

ビルド

プロジェクトに登録したソースファイルから，実行可能なプログラムを作成します。Visual C++ メニューから「ビルド」→「ビルド」を選択します。ここまでの作業を正しく行っていれば，しばらくして Visual Studio のウィンドウ下部に“hello.exe - ｴﾗｰ 0、警告 0”と表示されるはずです。

実行

「ビルド」「実行」を選択すると，コンソール画面（通常の設定では 80 桁 x25 行の黒い背景の画面）が現れて

```
Hello World!
Press any key to continue
```

と表示されます。表示されたら Enter キーを押してコンソール画面を終了させましょう。

ビルドがうまく行かない場合の対応

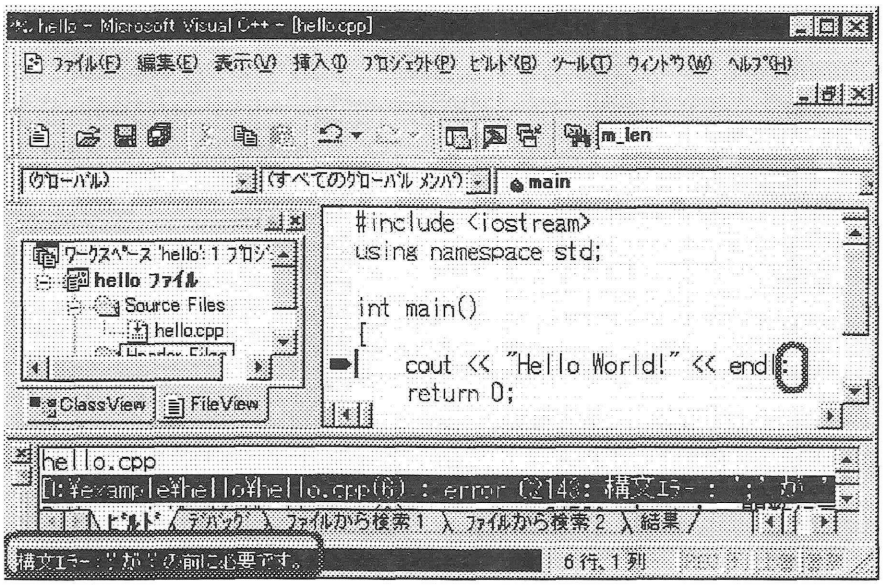


図 5 ビルドでエラーが発生した場合

ビルドを実行した際，“ｴﾗｰ 0、警告 0”と表示されずに，エラーの数が表示されたときは，（キーボードの）F4 キーを押します。すると，カーソルがエラーのある入力テキスト付

近に移動します。**図5**の例では，セミコロン「;」を入力すべき場所に，誤ってコロン「:」を入力していました。テキストを修正してもう一度ビルドしましょう。この他に，カッコの対応，二重引用符「"」の対応，つづり間違い，漢字変換モードでの入力などに注意が必要です。一個所の入力ミスでエラーが大量に発生することもあります。

すべてをチェックしてもわからない場合は，Visual C++ 開発環境を一旦終了し，新しいプロジェクト hello2 を作ってやり直してみましょう。プロジェクトの種類が「空のプロジェクト」になっていなかったり，入力したプログラムテキストに誤って 2 バイト文字が混入していることがあります。

2．ウィンドウを表示するプログラムを作成する

次に，ウィンドウを表示するプログラムを作ってみましょう。手順は少し面倒ですが，難しいことはありません。ウィンドウを表示する基本的な機能は Visual C++ が自動的に用意します。そのウィンドウの上に“コントロール”と呼ばれる部品（ボタンやテキスト）を追加登録し，それぞれのコントロールに対応するプログラムコードを記述します。

プロジェクトの新規作成

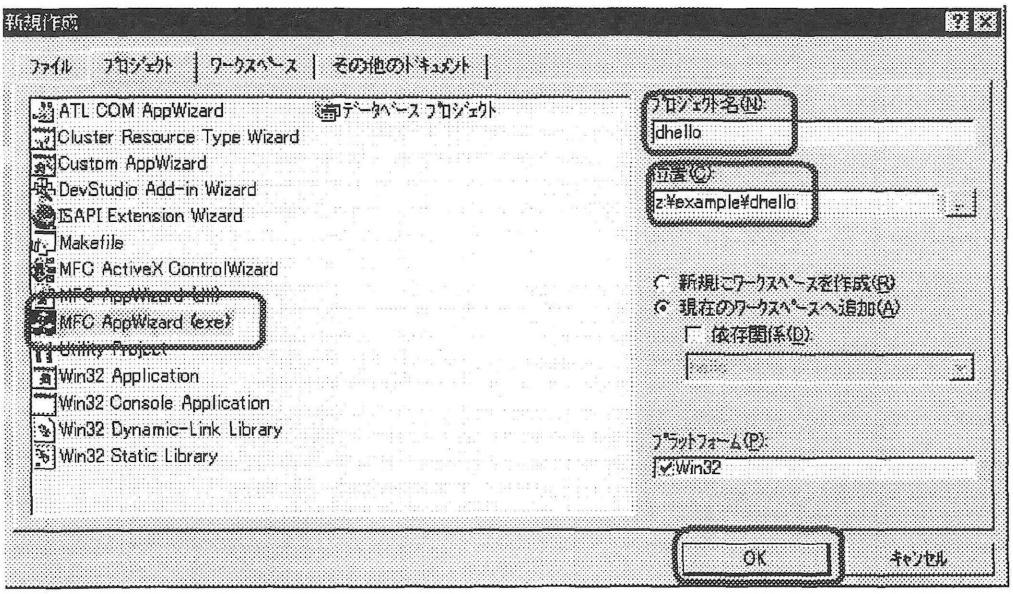


図 6 dhello プロジェクトの新規作成

コンソールアプリケーションと同じようにプロジェクトを生成します。ただし，プロジェクトの種類として「MFC AppWizard (exe)」を選択し，プロジェクト名として dhello を

入力します。位置は z:¥example¥dhello とします（図 6)。「OK」ボタンを押すと，作成するアプリケーションの種類の選択画面になります。ここで，「ダイアログベース」に選択を変更し，「終了」ボタンを押します（図 7)。

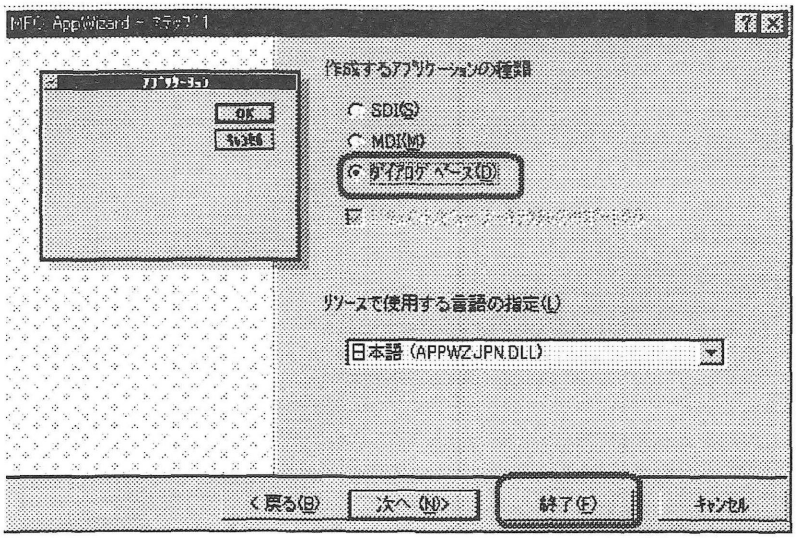


図 7 ダイアログベースアプリケーションの選択

ダイアログは，メニューも特別な描画画面も持たないウィンドウです。図 2,3,4,6,7 はダイアログの例です。なお，図 5 （Visual C++ 開発環境本体）のようなウィンドウは MDI（Multi Document Interface）といいます。

テストビルドと実行

MFC AppWizard のプロジェクトでは，Visual Studio がウィンドウを表示するために必要な最小限のプログラムコードを自動的生成します。そのため，既にこの段階で，Visual C++ のメニュー 「ビルド」「実行」を選択すると，プログラムが実行され図 8 に示すようなウィンドウ（ダイアログ）が表示されます。

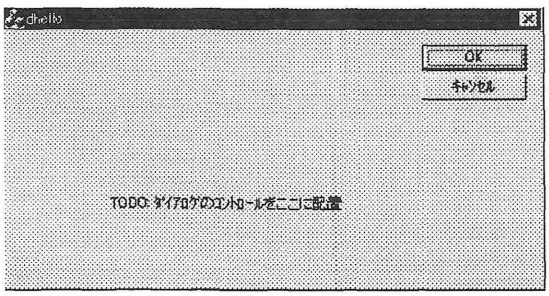


図 8 ウィンドウを表示するプログラムの実行テスト

ウィンドウの「OK」ボタンまたは「キャンセル」 ボタンを押すとプログラムは終了します。

プログラムの変更

次に，プログラムに変更を加えます。一つめのボタンを押すと Hello World! と表示し，もう一つのボタンを押すと“Hello World!” の文字数(12 文字)を表示する**図 9** のようなウィンドウを作ってみましょう。

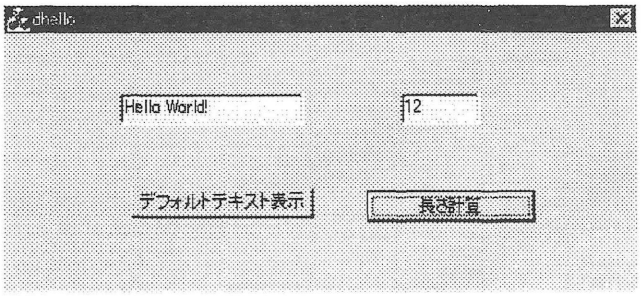


図 9 これから作成するプログラムのウィンドウの様子

ダイアログエディタの実行

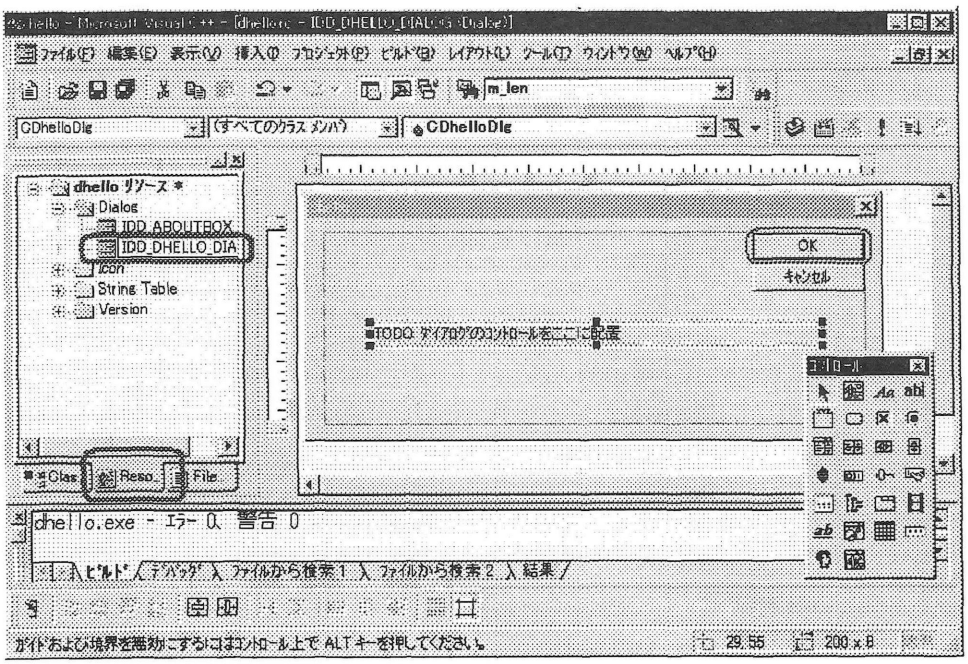


図 10 ダイアログエディタ

テストプログラムを終了させると，Visual Studio のウィンドウは，**図 10** のようになります。こうならない場合は，画面左下の「リソースタブ」を選択し，「Dialog」 アイコンを展開，さらに「IDD_DHELLO_DIALOG」 をクリックしてください。**図 10** の画面はウィンドウ上のコントロール（ボタンやメニュー）を編集する画面（ダイアログエディタ）です。

デフォルトコントロールの削除

ダイアログエディタの編集画面に表示されているウインドウの「OK」ボタンをマウスでクリックして選択し，キーボードの DELETE キーを押すと，ウィンドウから「OK」ボタンが削除されます。同様に，「キャンセル」ボタンおよび “ //TODO ...” テキストも削除します。

コントロールの追加

コントロールツールバー（**図 11**）で “ボタン” を選択し，ダイアログエディタ上のウィンドウのボタンを配置したい場所をクリックします。するとウィンドウ上にボタンが追加されます。

ボタンをもう一つと “エディットボックス” を 2 つ追加します。エディットボックスとは，テキストの入力，表示を行うための部品です。なお，設定によっては，コントロールツールバーの外見が**図 11** と異なる場合がありますので注意してください。この段階で，ダイアログエディタの編集画面は**図 12** のようになります。

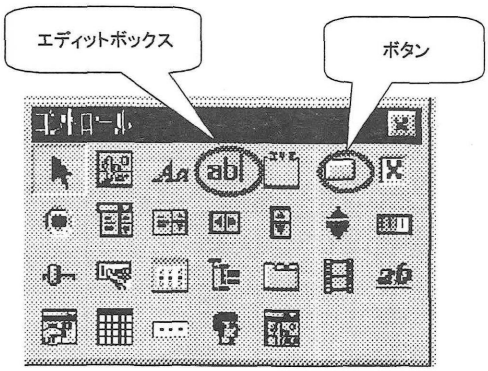


図 11 コントロールツールバー

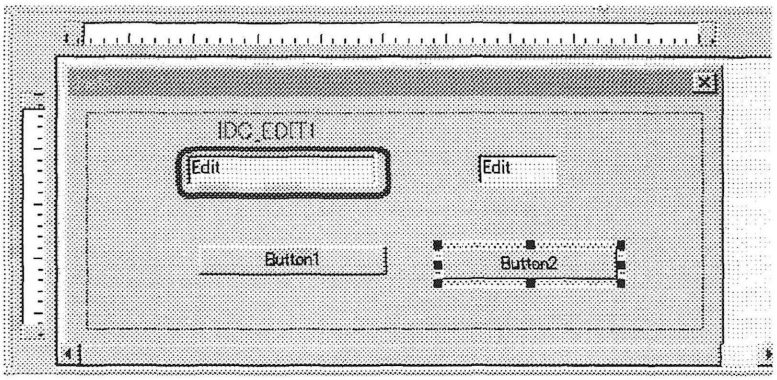


図 12 ダイアログエディタでのボタンとエディットボックスの追加

入力用変数の追加

ダイアログエディタの編集画面の上部左側に表示されているエディットボックス（**図 12** の IDC_EDIT1)をマウスで選択後，右クリックし表示されるメニューから ClassWizard を選択します（**図 13**)。IDC_EDIT1 は，ウィンドウ上のコントロール(ボタンやテキスト入力部）それぞれを識別するための名前（コントロール ID）です。「コントロール ID」で，IDC_EDIT1 が選択されていることを確認した後「メンバ変数タブ」を選び「変数の追加」ボタンを押します。変数名を m_hello と入力し，「OK」ボタンを押して終了します。この変数(m_hello)は，エディットボックスに入力，表示されるテキストを記憶するために使われます。

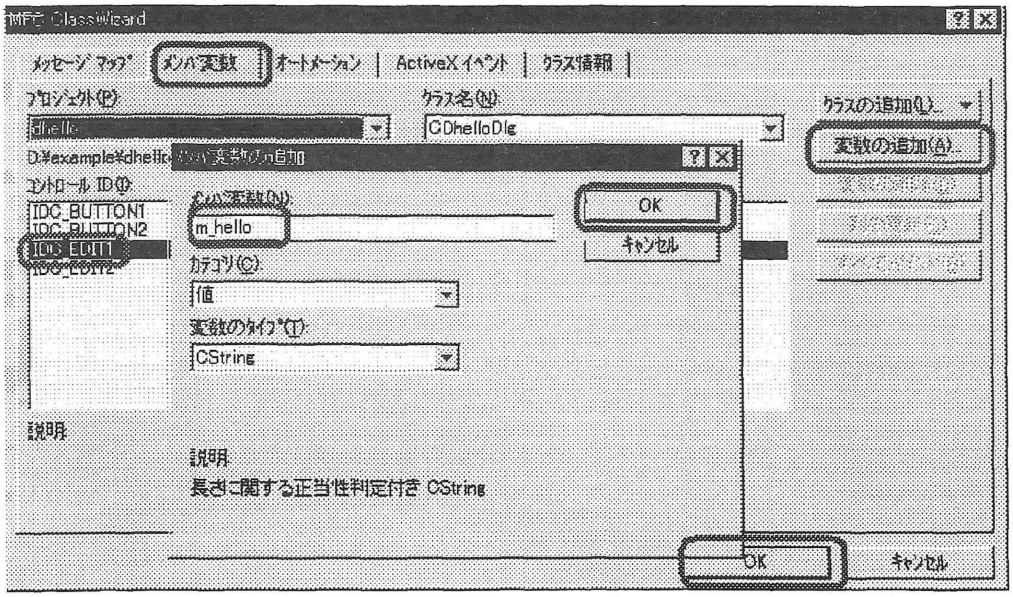


図 13 ClassWizard でのエディットボックスへの入力用変数追加

出力用変数の追加

同じく，右側のエディットボックス（IDC_EDIT2）に変数を追加します。変数名を m_length と設定します。設定画面は図 13 と同様です。ただし今度は，コントロール ID として，IDC_EDIT2 が選択されているはずです。

デフォルトテキスト表示ボタンの設定

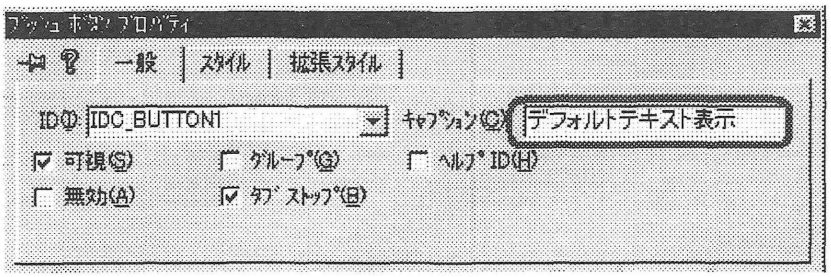


図 14 デフォルトテキスト表示ボタンのキャプション設定

ダイアログエディタ上で Button1 ボタンをマウスで選択，右クリックして表示されるメニューから「プロパティ」を選択します。「キャプション」を 「Button1」から「デフォルトテキスト表示」に変更します（図 14）。

デフォルトテキスト表示動作の設定

「デフォルトテキスト表示」ボタンをマウスで選択し（左）クリックして出てきたダイアログの「OK」ボタンを押すと，ボタンを押したときの動作を C++コードで指定できます。入力内容は，図 15 の枠で囲まれた部分です。m_hello は，図 13 で設定した，エディットボックス IDC_EDIT1 に表示される入力テキスト用の変数です。UpdateData(FALSE) によって，m_hello の内容がエディットボックスに表示されます。

長さ計算ボタンと長さ計算動作の設定

ダイアログエディタに戻って（図 10 参照），Button2 についても同様に，プロパティの「キャプション」を「長さ計算」に変更します（指定方法は図 13 参照）。「長さ計算」ボタンを押したときの動作は，図 16 のように指定します（入力方法は図 15 参照）。

図 16 では，1 行目の UpdateData(TRUE) で，エディットボックスに表示されているテキストを m_hello 変数に読み込みます。2 行目では m_hello テキストの長さを調べて（GetLength）文字列に変換し（Format） m_length に設定します。3 行目で m_length の内容がエディットボックスに表示されます。

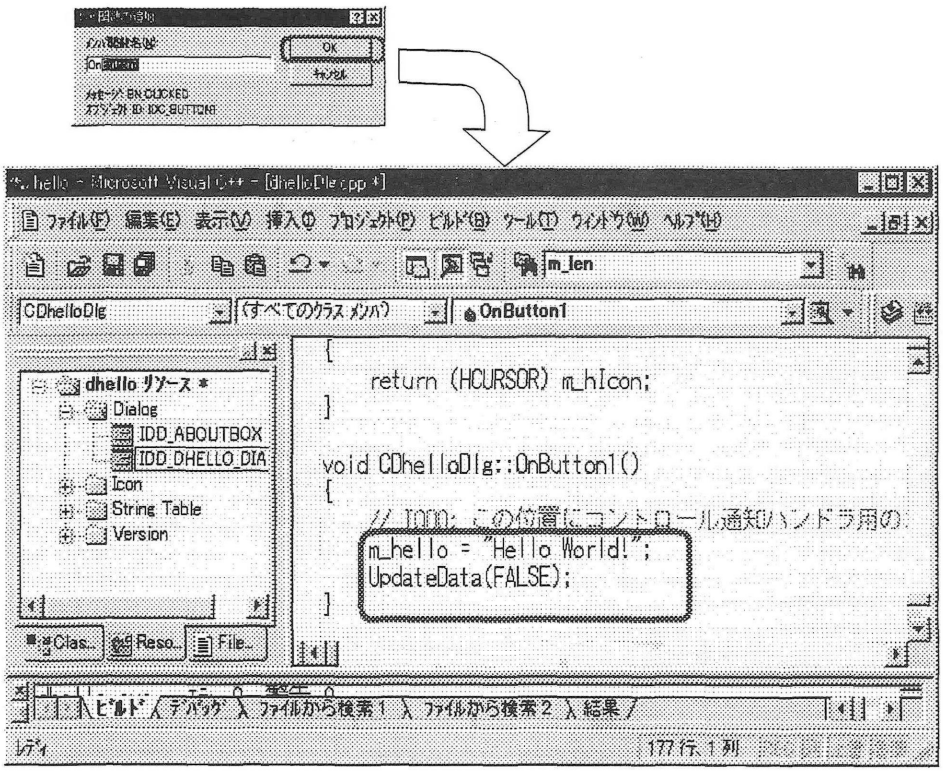


図 15 「デフォルトテキスト表示」ボタンを押したときの動作指定

```
UpdateData(TRUE);
m_length.Format("%2d",m_hello.GetLength());
UpdateData(FALSE);
```

図 16 長さ計算動作の設定コード

ビルドと実行

Visual Studio のメニューから 「ビルド」「実行」を選択すると，作成したプログラムを実行しウィンドウが表示されます。「デフォルトテキスト表示」ボタンを押すと，左側のエディットボックスに Hello World! と表示されます。「長さ計算」ボタンを押すと，右側のエディットボックスに テキスト “Hello World!” の文字数，すなわち 12 が表示されます。

左側のエディットボックスのテキストを適当に書き換えてから「長さ計算」ボタンを押

すと，その時表示されているテキストの文字数が表示されます。ウィンドウ右上の[x]をクリックすると，プログラムが終了します。**図 17** に実行の様子を示します。

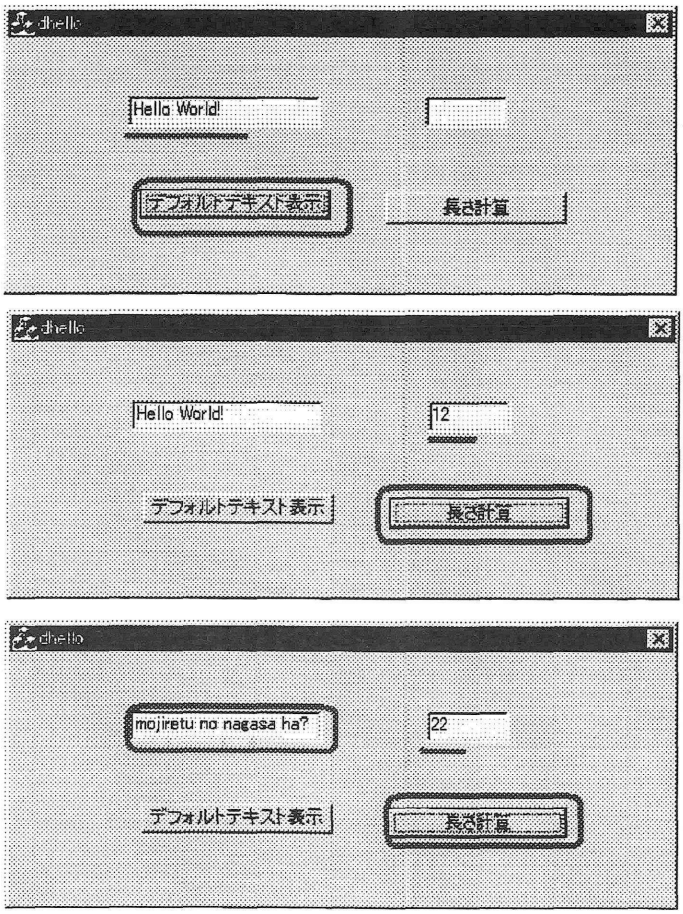


図 17 プログラム実行画面

3. 次に何をすれば良いか

ここまでで，Visual C++ を使って，コンソールアプリケーションと簡単なウィンドウを表示するプログラムの作成手順を示しました。それでは，自分の思う通りの Windows 上のプログラムを作れるようになるためには，これからどうしたらいいでしょう。大きく分けて，

- 一般的なプログラミングの方法についての知識
- C++ の文法についての知識
- Visual C++ と Windows の機能についての知識

の三つが必要になります。それぞれについて，多くの入門書が出版されていますので，それらを購入するのも良い方法です[4]。ここではもう少し手軽に出来る方法を紹介します。

Web で検索する

Web 上の検索エンジン（http://www.google.co.jp 等）を使って， Web 上の 資料を探してみましょう。検索キーワードとして {"C++"， "講座"}，{"Visual C++"，"入門"} などで検索すると，きっと役に立つ Web ページが見つかるはずです。

Web ページは，書籍の入門書に比べて作成者の好みが反映された内容になりがちです。自分の必要とする内容が書いてあるページを探しましょう。なお，この記事を書くに当たって， Web ページ http://www.nara-edu.ac.jp/~asait/visual_cpp/intro_cpp.htm が参考になりました。

オンラインヘルプを使う

Visual Studio には，大量のオンラインヘルプ情報（MSDN ライブラリ）が附属しています。たいていの情報はオンラインヘルプの中に記述されています。ただし，あまりにも情報が大量であるため，一から探そうとしても肝心な情報にたどり着くまでに疲れてしまう場合があります。 Web 上で調べた内容の確認，補完として使うのがよいでしょう。

少し腰をすえて Windows 上の Visual C++ プログラミングを勉強したい人は，Visual C++ オンラインヘルプ上で，"scribble チュートリアル" を検索して参考にしてください。

App Wizard の生成したプログラムを参考にする

AppWizard(exe) は，自動的にウィンドウを表示するプログラムを生成します。自動生成されたプログラムがどういう構造になっているか，ボタンや他のコントロールを追加したときに，プログラムにどのようなコードが追加されるのかを確認しておきましょう。理解の助けになるはずです。

---

[4] "C++プログラミング入門"，グレゴリー サティア他著，オライリー・ジャパン， ISBN: 4873110637 。"標準講座 MFC6.0—Visual C++による効率的な Windows プログラミング Programmer's SELECTION"，ハーバート シルト著，1999，翔泳社，ISBN: 4881357042 等。

Visual Basic 入門

多川孝央（九州大学情報基盤センター）
tagawat@cc.kyushu-u.ac.jp

1. はじめに

情報基盤センターの教育用システムでは、プログラム開発用の環境として Microsoft Visual Studio6.0 を利用者用パーソナルコンピュータにインストールしています。Visual Studio には Visual Basic、Visual C++、Visual J が含まれていますが、本稿ではこのうち Visual Basic の使いかたについて紹介します。

2. Visual Basic の操作とサンプルプログラム

Visual Basic は Windows で動作するアプリケーションプログラムを開発するための言語環境です。Visual Basic の特徴は、マウス操作を中心としたグラフィカルな開発環境のもとで、高度な知識がなくても簡単に Windows のプログラムを作成することが可能なことです。ここでは、Visual Basic の利用に関連する初歩的な知識や概念を、簡単なサンプルプログラムの作成の様子を交えながら紹介します。

Visual Basic の起動

Visual Basic を起動するには、スタートメニューから「プログラム(P)」「Microsoft Visual Studio 6.0」「Microsoft Visual Basic 6.0」と順を追って選択してください。起動すると、最初に「新しいプロジェクト」のダイアログボックスが表示されます。普通の Windows アプリケーションを作成するということで、今回は「標準.EXE」を選択してください。

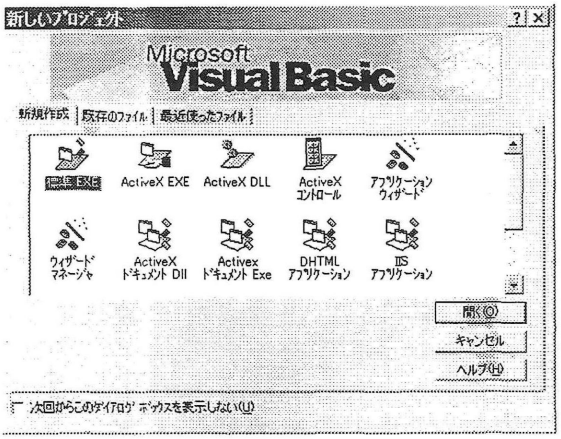


図 1　「新しいプロジェクト」選択画面

Visual Basic のウインドウが表示されます。

ここで、画面の左側にある、いくつものアイコンが並んでいる領域を「ツールボックス」、画面の右側の上にある樹形状の表示領域を「プロジェクトエクスプローラ」、その下にある表示領域を「プロパティウインドウ」といいます。

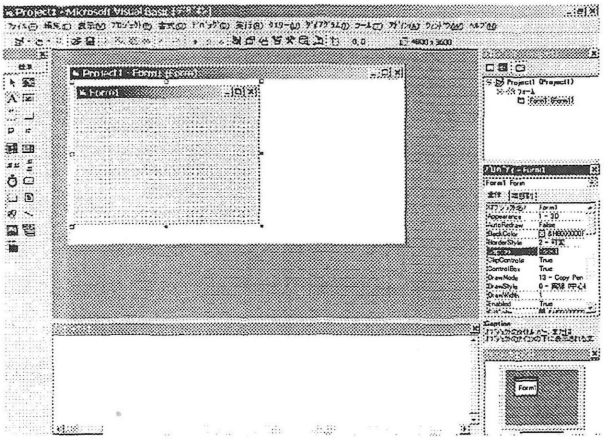


図 2　Visual Basic6.0 の画面

中央に表示されているのは「フォームレイアウトウインドウ」といいます。フォームレイアウトウインドウの中には、「フォーム」と呼ばれる四角形の領域が配置されています。このフォームは、作成するプログラムの外見の雛形になるものです。

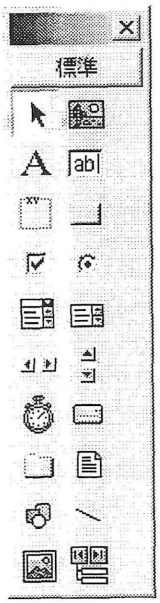


図 3　ツールボックス

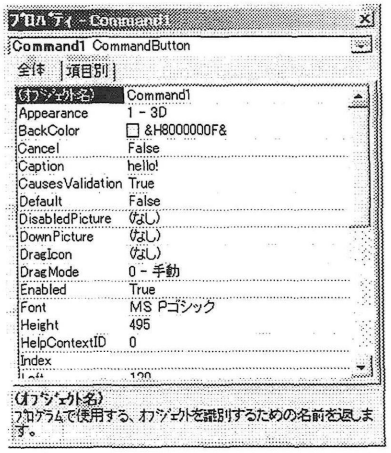


図 4　プロパティウィンドウ

ツールボックスに並んでいるアイコンを「コントロール」といいます。各コントロールは様々な機能を持っていて、プログラマはこれをフォームの上に配置することによってプログラムのユーザインタフェイスをデザインすることになります。

Visual Basic のプログラミングは、基本的にはフォームにコントロールを配置し、各コントロールに関する設定をプロパティウインドウで行い、その後ユーザがコントロールに対して操作を行ったときに実行する処理を記述し、最後にコンパイルすることによって実行可能なアプリケーションを作成するという手順になります。

プロジェクト

プロジェクトとは、Visual Basic におけるプログラミングの単位で、一つのプロジェクトで一個のアプリケーションプログラムを作成することになります。プロジェクトは、プロジェクトエクスプローラでは最上部、樹形状の幹にあたる部分に存在し、その下に一個ないし複数のフォームが配置されます。プロジェクトエクスプローラ上でプロジェクトをクリックすると、プロパティウインドウにはプロジェクトの名前だけが表示されます。「全体」と名前のついたタブの中の「(オブジェクト名)」という項目にカーソルを合わせて文字を入力すると、プロジェクトの名前が変化します。最終的に作成されるアプリケーションの名前はこのオブジェクト名になります。この例では、作成するプログラムの内容を示す helloworld という名前にします。

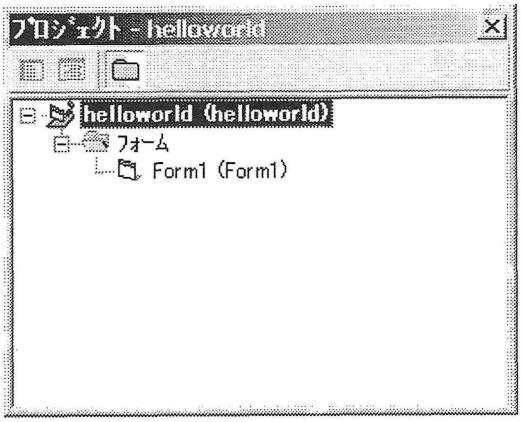


図 5　プロジェクトエクスプローラ

コントロールの配置

コントロールとは Visual Basic のプログラムの部品で、Visual Basic のデザインは、このコントロールの組み合わせによって行われます。コントロールには「ボタン」「テキストボックス」「チェックボックス」などがあります。ツールボックスのアイコンをクリックし、次にマウスのカーソルをフォームの上でドラッグすると、フォーム上にコントロールが配置されることになります。一旦配置したコントロールは、マウスでの操作によってフォームの上での位置を変更したり、大きさを変えたりすることができます。フォーム上に配置

したコントロールをクリックすると、プロパティウインドウにそのコントロールのプロパティが表示されます。このプロパティの値を設定することで、フォーム上の各コントロールの見かけや細かい動作を変更することができます。この例では、テキストボックスとコマンドボタン、ラベルを利用しますので、それらを図のようにフォーム上に置きます。

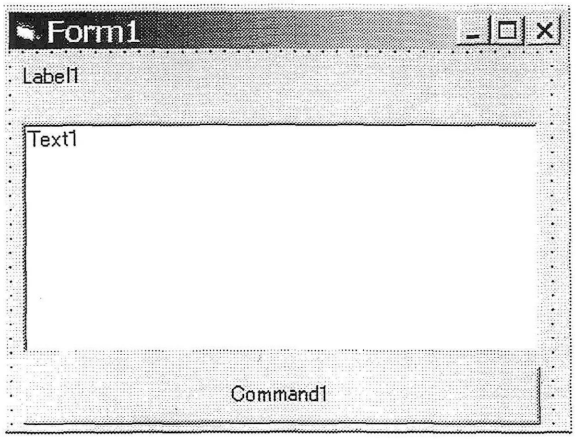


図 6　フォーム上にコントロールを配置した例

ここで、外見を整えるためにフォーム、テキストボックス、コマンドボタン、ラベルのプロパティを適切に設定します。フォームをクリックし、プロパティウインドウ上の「Caption」という項目を「Sample Program」と変更します。すると、フォームの上部の青い部分に表示される文字が「Form1」から「Sample Program」に変わります。同様に、配置したコマンドボタンの「Caption」を「Command1」から「hello!」に、テキストボックスの「Text」という項目を「メッセージはここに表示されます」に、ラベルの「Caption」を「Label1」から「テスト」に変更すると、フォームの外見は下のようになります。これで、ひとまずプログラムの外見は完成しました。

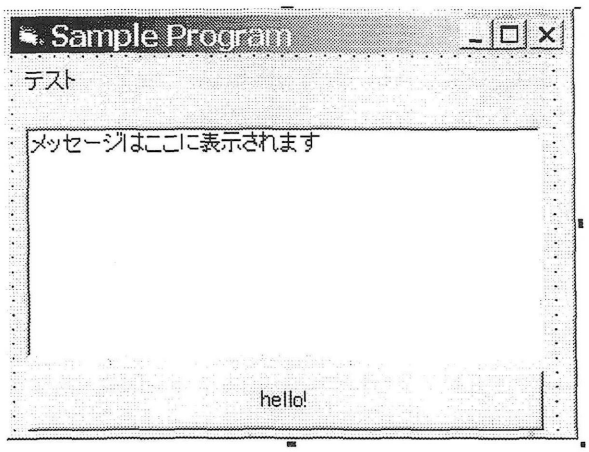


図 7　プロパティの設定でコントロールの表示内容を変更した例

コードの記述

コントロールを配置することでアプリケーションの外見を作ることができました。次は、アプリケーションの動作を決定する必要があります。これは、フォーム自身やそれぞれのコントロールに対してユーザが特定の操作を行った場合に（これを「イベント」と呼びます）どのように反応するかを記述することによって行われます。

フォーム上のコントロールをダブルクリックすると、「コードエディタウインドウ」が開きます。Visual Basic では、このウインドウに Basic 言語で処理を書くことでアプリケーションの動作を指示します。このサンプルプログラムでは、ユーザの操作はコマンドボタンをクリックするだけですので、フォームレイアウトウインドウ上のコマンドボタンをダブルクリックします。すると、下のようにコードエディタウインドウが表示されます。

詳しい説明はここでは省略しますが、「Private Sub Command1_Click()」という行と、「End Sub」という行の間に、Basic 言語で操作を記述することによって、ボタンがクリックされた際のプログラムの動作を設定することになります。このサンプルプログラムでは、

```
Text1.Text = "Hello,world!"
```

とだけ記述します。これは、「Text1.Text と変数に、"Hello,world!"という値を代入する」という操作になります。

ところで、このプログラムにおいては、Text1.Text という変数は、フォーム上に配置されたテキストボックス Text1 のテキストボックスの表示内容を指します。すなわち、この記述内容は、テキストボックスの表示内容を"Hello,world!"に変更することを意味しています。この一行をボタンがクリックされた際の動作として記述したことによって、「ボタンをクリックしたら、テキストボックスに"Hello,world!"と表示される」という機能が実現されたことになります。

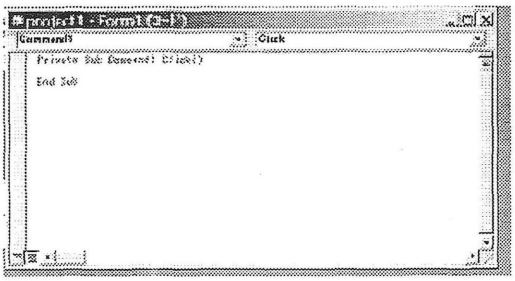


図 8　コードエディタウインドウ

動作テスト・デバッグ

フォームとコントロールを配置し、コードを記述したらひとまずプログラムは完成です。この後は出来上がったプログラムを実行可能なアプリケーションの形にする作業が残っていますが、その前に、プログラムが自分の意図した通りに動作するかをチェックする必要があります。

では、プログラムをテスト的に実行してみます。メニューから「実行(R)」>「開始(S) F5」を選択するか、キーボードの上段の「F5」と書いてあるキーを押す、あるいは、ツールバー上で右向きの三角のアイコンをクリックすると、プログラムが実行され、フォームが表示されます。この時、フォームの上に配置されたコントロールの外見は、プロパティウインドウで設定した通りになっています。

ここでコマンドボタンを押すと、テキストボックスの内容が、「メッセージはここに表示されます」から、「Hello, world!」に変化します。意図した通りの動作をしていますので、これでテストは終了です。メニューから「実行(R)」>「終了(E)」を選択するか、ツールバー上で停止を示す四角形のアイコンをクリックすることで、テスト的な実行を終了させます。

プログラムが正常に実行できないことがシステムにとって明らかな場合には、エラーの種類を示す番号とともにエラー内容を示すメッセージウインドウが表示されます。ここで、ウインドウの中の「デバッグ」と書いてあるボタンをクリックすると、コードエディタウインドウ中のエラーの発生した箇所が黄色く表示されます。ここを修正し、もういちどプログラムテストを行ってください。

実行形式ファイルの作成

プログラムが自分の意図の通りに動作することが確認できたら、実行形式のファイルを作成します。「ファイル(F)」から「helloworld.exe の作成」を選択してください。保存場所を選ぶ画面が表示されます。プログラムを、現在利用している開発環境なしで実行可能な形にしたものが、この画面の操作によって保存されることになります。ここで保存されるプログラムファイルの名称は、現在のプロジェクトの名称にあわせたものになります。Visual Basic を起動したときのままプロジェクト名を変更していない場合は、project1.exe になります。

Visual Basic6.0 で作成したアプリケーションプログラムは、必ずしも単体で完全に動作するとは限りません。多くの場合、Visual Basic 用のランタイムと呼ばれるファイル群が実行するパーソナルコンピュータにインストールされている必要があります。Visual Basic がインストールされているパーソナルコンピュータでは、ランタイムを追加してインストールする必要はありません。

利用の終了

Visual Basic での作業を終了する時には、メニューから「ファイル(F)」>「Microsoft Visual Basic の終了(X) Alt +Q」と選択してください。プロジェクトのファイルとフォームのファイル（それぞれ拡張子が vbp と frm になります）に、プロジェクトの内容とフォームの内容が保存されます。これまでに行った作業の続きを再開したい時には、Visual Basic 起動時の画面で既存のファイルを開くことを選択し、適切なプロジェクトファイルを選択すれば、前回に保存した時の状態から作業を継続することが可能です。

3. おわりに

本稿ではサンプルプログラムの作成例を交えながらVisual Basicの操作に関する初歩的な事柄を紹介しました。ただし、紙幅の都合からBasic言語によるコーディングやデバッグツールの使用法などは省略してあります。また、コントロールやイベントについても特定のものしか扱っていません。

Visual Basicでは、コントロールとイベント、プロパティの設定の組み合わせやコードの記述によって、様々な機能のアプリケーションを簡単に作成することが可能です。その方法は、本稿に示した手順と基本的には同じものです。本稿がVisual Basicの利用の入門としてお役に立てば幸いです。

JBuilder入門

池田 大輔*

1 JBuilderとは

JBuilderとはBorland社が開発したプログラミング言語Javaの統合開発環境です。JBuilderの最新のバージョンは7で、Solaris, Linux, Windows XP/2000/NT, Max OS Xで動作します。

JBuilderにはPersonal, SE, Enterpriseの3つのラインナップがあります。このうちPersonalは個人・教育利用向けのラインナップであり、無料で利用できます[1]。本稿ではJBuilder 7 Personal(以下、単にJBuilderと書きます)のインストール方法[2]と、簡単な利用方法をサンプルプログラム作成を通して説明します。

Javaはグラフィカルなインターフェイスを持つプログラムを作成することができます。本稿では、メニューバーとツールバーを持ち、ボタンを押したら特定の動作をするプログラムを作ります(4節参照)。

また、JBuilderはプログラム開発に有用なデバッガ機能も備えています。5節で簡単に説明します。

インストールとサンプルプログラムの実行はWindows 2000上で行ないました。

2 インストール

2.1 準備

必要なファイルをBorland社のJBuilder Personalのサイト[3]からダウンロードします。このページの下のほうに、バージョン7のダウンロードページへのリンクがあります。また、JBuilder Personalのサイトには JBuilderを収録している雑誌や書籍の一覧もあります。ダウンロードページには、インストールの方法の説明とライセンスキー取得のページへのリンクがあります。

最低限必要なファイルはJbuilderそのもので、ファイル名は`jb7_windows.zip`です。"FTP"の欄が実際にダウンロードするための欄ですが、"1"と"2"の2つのリンクがあります。"2"をクリックすると、Borland社の日本法人のサイトからダウンロードします。

他に、オンラインマニュアルとサンプルプログラムがダウンロード可能です。最初にJBuilderをインストールしておけば[4]、マニュアルとサンプルのインストールはJbuilderと同様にできま

*情報基盤センター研究部　mailto:daisuke@cc.kyushu-u.ac.jp

[1]商用のプログラム開発には使用できません。

[2]教育用システムにはJBuilderはインストールされていませんので、ダウンロードして自宅のパソコンにインストールしてください。

[3]http://www.borland.co.jp/jbuilder/personal/

[4]マニュアルとサンプルのインストールの途中でJBuilderをインストールしたフォルダを聞かれます。

すので省略します。

2.2 インストール

ダウンロードしたjb7_windows.zipを適当な場所に展開します。per_installがインストーラで、setup_windowsがセットアップ方法を説明したHTMLファイルです。

インストールするにはper_installをダブルクリックします。最初にインストール画面での使用言語を指定します。日本語と英語が選択でき、デフォルトで日本語になっています。次はインストール画面の簡単な説明で、そのまま[次へ]を押します。ライセンスが表示されるので(図1参照)、よく読んだ上で[ライセンス契約の条項に同意する]にチェックを入れてから[次へ]を押します。次にインストールするフォルダを指定します(図2参照)。この時、空白を含

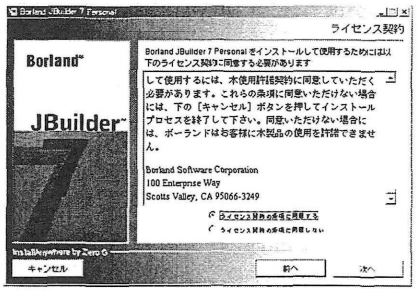


図1: ライセンスの表示

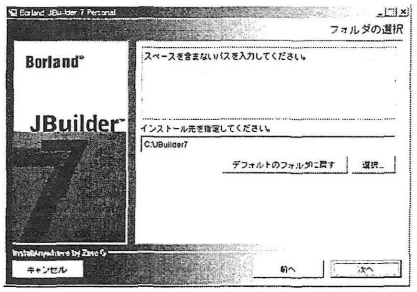


図2: フォルダの指定

んだフォルダを指定しないでください。

次はインストールする直前の確認画面が表示されます。[前へ]を押すと、前の状態に戻ります。[次へ]を押すとインストールが開始されます。

[スタート]→[プログラム]→[Borland JBuilder 7 Personal]→[Borland JBuilder 7 Personal]でJBuilderが起動しますが、初めて起動した時はライセンスキーを登録していないので、利用できません。登録の方法は2.3節を参照してください。

2.3 使用許諾コードの登録

ダウンロードページ[5]の下のほうにある「JBuilder 7 Personalのライセンスキーの取得」をクリックします。クリックした後のページに「JBuilder 7 Personalユーザ登録」というリンクがあり、これをクリックするとアンケート項目などを入力する画面になります。これらは英語で書いてありますが、「入力項目の日本語解説」というリンクもあります。

登録するには、職業やBorland社の製品のうちどれを持っているか、などのアンケート[6]に答えなくてはなりません。アンケートに答えるとライセンスキーを送るためのメールアドレスを尋ねられます。これに答えると登録は完了で、Borland社からライセンスキーがメールで送られます。メールにテキストファイルが添付されています[7]ので、これを適当なフォルダに保存

[5] http://www.borland.co.jp/jbuilder/jb7/download/
[6] 大学の構成員にはあまり関係のない質問もあります。
[7] 筆者あてのメールではreg62.txtというファイル名でした。

します。

[スタート]→[プログラム]→[Borland JBuilder 7 Personal]→[Borland JBuilder 7 Personal] で JBuilder が起動します。初めての起動の場合は図 3 のように「使用許諾コードの取得」というウインドが現われます。ここで、下の [使用許諾コードを持っている場合] にチェックを入れます。

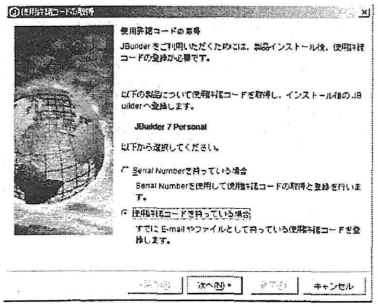


図 3: 使用許諾コードの入力 (1)

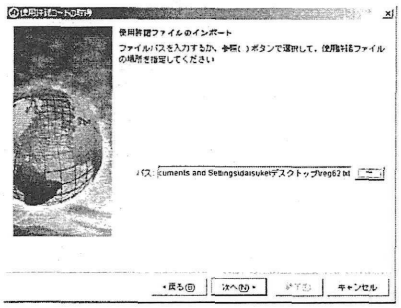


図 4: 使用許諾コードの入力 (2)

次に図 4 の [パス] に、さきほど保存したライセンスキーのテキストファイル名を書きます。横の [...] ボタンを押すとファイル選択のダイアログが現われます。

その後、再びライセンスの提示があり、これに同意すればコード登録が完了し、図 5 と図 6 のような画面が現われます。図 5 は、Java のソースファイル、クラスファイルやコンパイルし

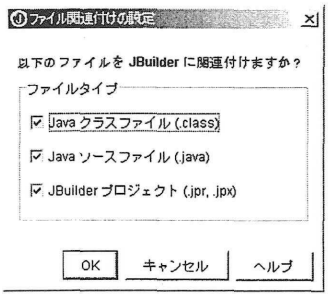


図 5: ファイル関連付けの設定

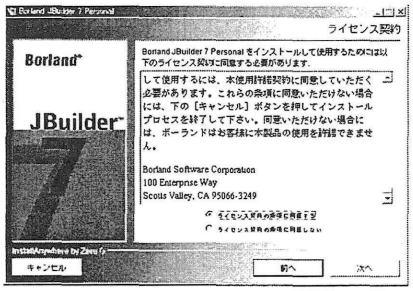


図 6: ワンポイントアドバイス

たクラスファイルなどをダブルクリックした時、自動的に JBuilder が起動するようにするための設定です。関連づけなくても問題はありません。ここではすべてにチェックを入れて [OK] を押します。以後、JBuilder を起動してもこのウインドは現われません。

図 6 は JBuilder を便利に使うためのワンポイントアドバイスです。[次回の起動時もワンポイントを表示する] のチェックをはずせば、次回以降表示されなくなります。これらのウインドを閉じると JBuilder の起動画面 (図 7) が現われます。

3 ウインド構成

図 7 が JBuilder の起動画面です。この図では左右に大きく 2 つの部分に分かれています。左側が “プロジェクトペイン” で、右が “内容ペイン” です。

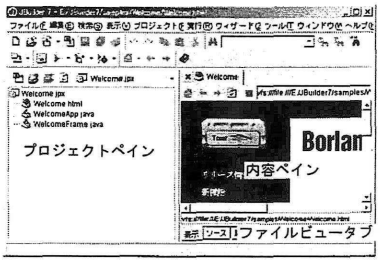


図 7: JBuilder の起動画面

JBuilder に限らず、統合開発環境では“プロジェクト”という単位でプログラムの開発を進めます。プロジェクトには、プログラムを構成する複数のソースファイルやプログラムのドキュメントなどが含まれます。プロジェクトの名前は、これらのファイルを保存するフォルダ名やファイル名の一部にもなります。

プロジェクトペインはプロジェクトに含まれるファイルを表示します。図 7 では、`Welcome.jpx` がプロジェクト名で、その下の 3 つのファイルがプロジェクトに含まれるファイルです。

内容ペインには、プロジェクトペインでダブルクリックしたファイルの中身が表示されます。内容ペインの下には“ファイルビュータブ”があります。このタブ[8]により、単純に中身を表示させたり、表示しているファイルに対する変更履歴を表示させたりできます。

プロジェクトペイン上の Java のソースファイル (ファイル名が `.java` で終わるもの) をダブルクリックすると、プロジェクトペインの下に新しく“構造ペイン”が表示されます[9]。この場合、JBuilder のウインドは図 8 のような構成になります。表示されたファイルの構造が構造ペインに表示されます。

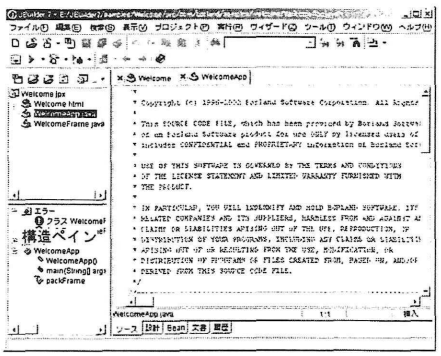


図 8: 構造ペインが表示された時のウイン ド構成

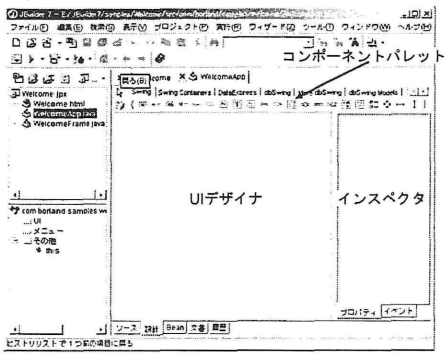


図 9: [設計] タブを押した時のウインド構成

内容ペインに Java ソースファイルを表示させている状態で、ファイルビュータブの [設計] タブを押すと図 9 のようなウインド構成になります。内容ペインは“UI デザイナ”と“インスペクタ”に分割され、上に“コンポーネントパレット”が表示されます。図 9 では、UI デザイナと

[8]表示するファイルによってタブの種類は異なります。
[9][表示] メニューで特定のペインの表示・非表示を切り替えることができます。

インスペクタには何も表示されていませんが、通常、UIデザイナにはウインドやツールバーなどのグラフィカルな部品が、インスペクタはその中で選択した部品の属性値などが表形式で表示されます。

4 Hello Worldプログラム

おなじみの“Hello World”プログラムを作成してみましょう。このプログラムは、ウインドを作成してその中に“Hello World”と表示します。ウインド内にはボタンがあり、これを押すと文字の色が変化します。実行時のイメージは図10のようになります。

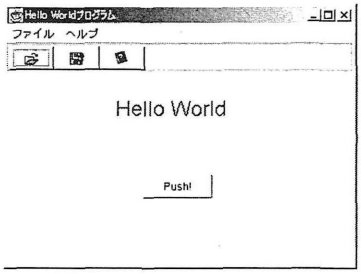


図10: サンプルプログラムの実行画面

作成の手順はおおまかに以下のようになります。

- プロジェクトの作成
- ソースファイルの追加
- ラベルの追加
- イベントの追加
- レイアウトの追加

4.1 プロジェクトの作成

まずプロジェクトを新規に作成します。[ファイル]→[新規]を選択し[オブジェクトギャラリ]を表示させます(図11参照)。ここで[プロジェクト]タブを押し、その中の[プロジェクト]アイコンを選択し[OK]を押します。[ファイル]→[新規プロジェクト]でも構いません。

プロジェクトの名前やファイルの保存場所などを決定するためのウィザード(図12参照)が開きます。ウィザードは3つのステップから成ります。

最初のステップでプロジェクトの名前や保存する場所を決めます。プロジェクトの名前を“HelloWorld”にして、[プロジェクトノートファイルの生成]にチェックを入れます。これ以外はデフォルトのまま利用します。

残り2つのステップは後で変更することが可能です。2番目のステップはデフォルトのままで利用します。3番目のステップは、[クラスJavaDocフィールド]の右側の[テキスト]の欄を適当に埋めます(図13参照)。これらは、このプロジェクトを表わすタイトルや会社名などです。

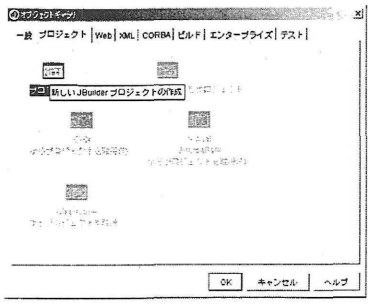


図 11: オブジェクトギャラリ

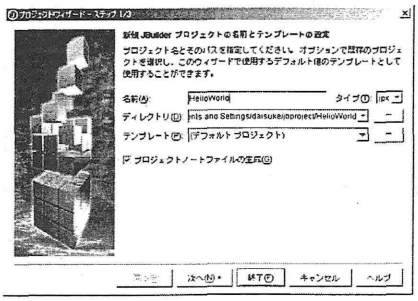


図 12: プロジェクトウィザード (1/3)

これで、プロジェクトペインに HelloWorld.jpx というプロジェクトができます。プロジェクトには HellowWorld.html というファイルのみが含まれています。このファイルが、さきほどチェックをいれたプロジェクトノートファイルです。ファイル名をダブルクリックすると閲覧でき (図 14 参照)、図 13 で入力した項目が表示されていると思います。

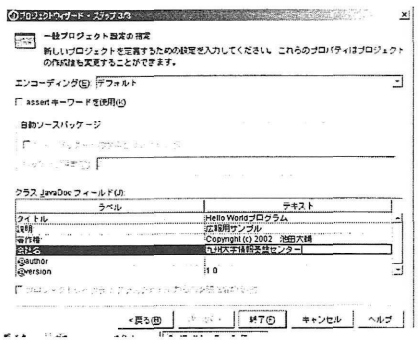


図 13: プロジェクトウィザード (3/3)

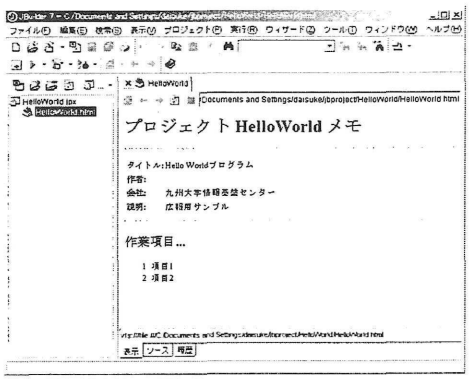


図 14: プロジェクトノートを表示

既に途中まで編集作業をやっていて、再びプロジェクトを開きたい場合は [ファイル] → [プロジェクトを開く] を選択します。図 15 のようなダイアログが開きます。ここからプロジェクトを選択してください。

4.2　ソースファイルの追加

[ファイル] → [新規] から [新規] タブを押し、オブジェクトギャラリを表示させます。[アプリケーション] を選択し、アプリケーションウィザードを開きます (図 16 参照)。パッケージ名はデフォルトのまま使います。クラス名は “HelloWorldClass” にします。クラス名の大文字と小文字は区別されます。これらの名前は好きにつけて構いませんが、クラス名の慣習として単語ごとに大文字で始めています。

一番下の [ヘッダーコメントの生成] は、デフォルトでチェックがはいっていると思います。

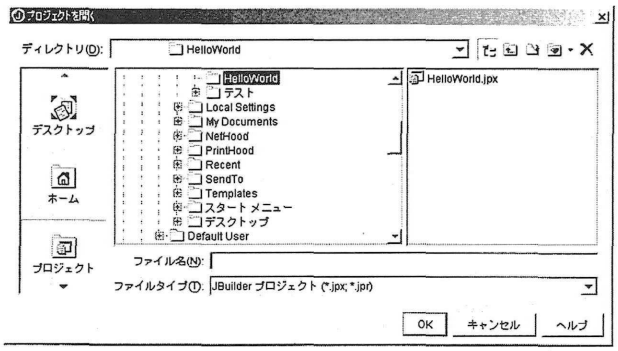


図 15: “プロジェクトを開く” ダイアログ

これにチェックがはいっていると、プロジェクト生成時に記入したプロジェクトに関する情報がソースファイルの先頭にコメントとして挿入されます。

ウィザードの次のステップ (図 17 参照) はフレームに関する設定です。フレームとはメニュー

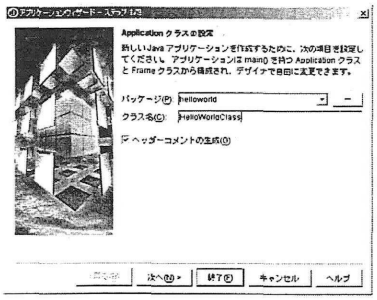


図 16: アプリケーションウィザード (1/3)

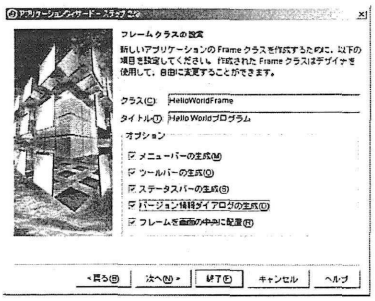


図 17: アプリケーションウィザード (2/3)

バー、ボタン、テキストフィールドなどを配置するためのグラフィカルな部品で、独立したウインドを作成するときに必ず必要となります。

クラス名は “HelloWorldFrame” とします。この名前も好きにつけて構いません。タイトルは、ウインドのタイトルバーに表示される文字列のことで、ここでは “Hello World プログラム” としましょう。ここではオプションの項目を全て選択してみます (図 17 参照)。

次のステップは変更せず、[終了] ボタンを押します。プロジェクトペインにいくつかのファイルが追加され、内容ペインに Java のソースファイルが表示されると思います (図 18 参照)。構造ペインには、このソースの構造が表示されています。プロジェクトに追加されるのは Java ソースファイルだけでなく、ツールバーのボタン用の画像なども含まれます。

間違えて追加してしまった場合は、消したいファイルをプロジェクトペイン上で選択してから、[プロジェクト](または選択したファイル上で右クリック) → [プロジェクトから削除] を選択します。

追加したソースファイルは、最低限の記述があらかじめされてるので、この時点で実行することも可能です。実行するには [実行] → [プロジェクトの実行] を選択してください。図 19 のような画面が現われます。ウインドやメニュー、ツールバーが、特に Java のソースを書かなく

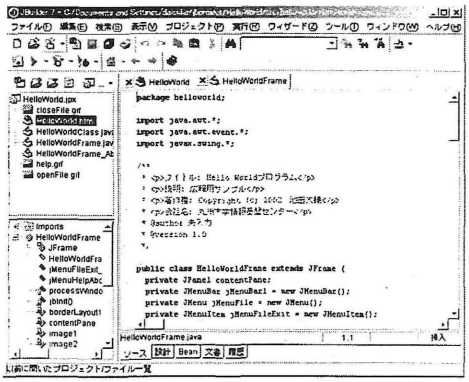


図 18: ソースファイル追加後の画面

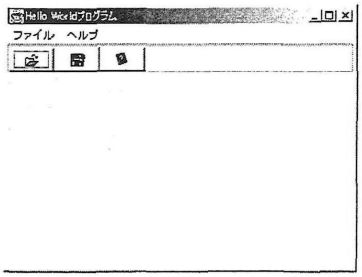


図 19: 実行画面

ても簡単に生成できることが分かります。

実行を終了するには [ファイル] → [終了] を選択してください。終了するための動作はすでに書いてあります。実行終了後は図 20 のように、一番下に “メッセージビュー” と呼ばれる領

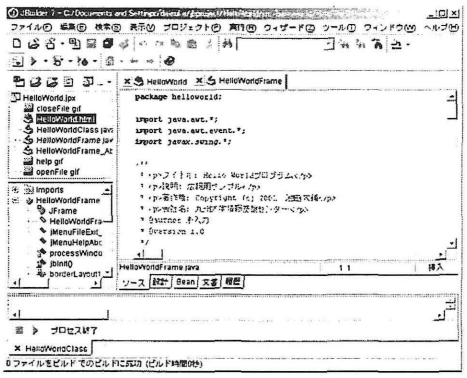


図 20: 実行終了後

域ができています。これは、実行時のメッセージが表示されるところです。実行後はタブの×ボタンで削除してください。

4.3　ラベルの追加

JBuilder では、ユーザインターフェイスのプログラミングを簡単な操作で実現できます。すでにメニューバーとツールバーができました。今度はこれにラベルを追加しましょう。

プロジェクトペインから HelloWorldFrame.java をダブルクリックし、内容ペインに表示させます。ファイルビュータブから [設計] タブを選択します。内容ペインには、現在までに追加されたユーザインターフェイスの部品が表示されています (図 21 参照)。

図 21 にはツールバーのみでメニューバーは表示されていません。メニューバーとインターフェイスは別々に扱われるためで、構造ペインの下のほうに [メニュー] フォルダがあります。

構造ペインは木構造を成しており、例えばjToolBar (ツールバー) の下にjButton (ボタン) が3つあります。実際、図21の内容ペインにツールバー上のボタンが3つ見えます。

[設計] タブを押すと、コンポーネントパレットが表示されます。タブでおおまかな分類を選び、その下のツールバーで実際に部品を選択します。

この節の目的はラベルを追加することですが、直接ラベルをフレーム (ウインド) に載せずに、まず、コンテナと呼ばれる他の部品を載せる部品を最初に追加します。コンポーネントパレットの [Swing Containers] タブを押し、ツールバー上のパネルを選択し、ウインドの中央付近でクリックして貼り付けます。パネルは一番左にある部品で、クラス名は `javax.swing.JPanel`

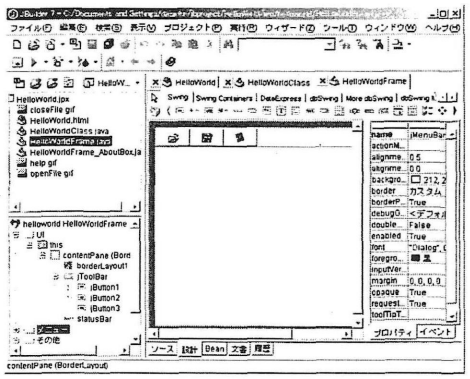


図21: ユーザインターフェイスの設計画面 (最初)

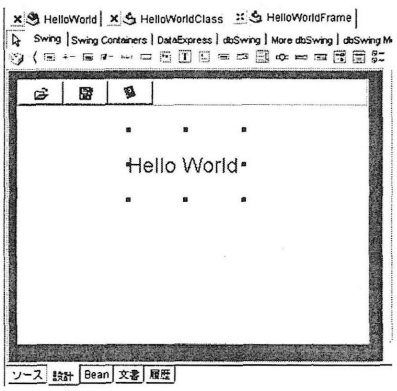


図22: ユーザインターフェイスの設計画面 (追加後)

です。

パネルを追加しても特に表示に変化は見られませんが、構造ペインに [jPanel1] というフォルダが追加され、ダブルクリックして開くと [⟨FlowLayout⟩][10]があります。また、インスペクタの [プロパティ] タブの上部に "jPanel1" と表示され、このことからもパネルが追加されていることが分かります。

インスペクタの [プロパティ] タブで部品の設定を変更します。まず、[layout] は "null" にしておいてください。こうしておかないと、お仕着せのレイアウトマネージャによって、今から追加する部品が置きたい場所におけなくなります。好きな位置に配置してから、適当なレイアウトマネージャを設定します。

他に、例えば、背景の色を変更する時は [background] の右側の欄から好きな色を選択します。ここでは "白" にしました。

次にラベルを追加します。コンポーネントパレットの [Swing] タブからラベルを選択し、パネルの適当なところをクリックして配置します。ラベルは左から5番目の部品で、クラス名は `javax.swing.JLabel` です。最終的に表1のようにプロパティを変更しました。フォントのサイズを大きくしているので、文字が全て見えるようにラベルの隅をドラッグしてサイズを変更します。また、位置も中央付近に移動させています。ここまでの作業で、インターフェイスは図22のようになりました。

[10]この部品はパネル上の部品をどのように配置するかを指定するもので、"レイアウトマネージャ" と呼ばれます。

表 1: ラベルのプロパティ

属性名	属性値
text	Hello Wolrd
font	サイズ 24
foreground	赤

4.4 イベントの追加

新たにボタンを追加し、このボタンにイベントを追加します。ここで、イベントとはキーボードやマウスなどによるユーザからの入力のことです。ここでは、ボタンがクリックされたら、先程追加したラベルの色を変更するようにしてみます。

HelloWorldFrame.javaを開き、[設計]タブを押します。コンポーネントパレットの[Swing]タブからボタンを追加します。ボタンは一番左の部品で、クラス名は javax.swing.JButton です。追加したボタンの“text”フィールドの値を例えば“Push!”とします。これがボタンに表示されます。

インスペクタの[イベント]タブを押し、“actionPerformed”フィールドの右側の欄をダブルクリックします。内容ペインが自動的に[ソース]タブに切り替わり、

```
void jButton4_actionPerformed(ActionEvent e) {

}
```

というコードが追加されます。このコードは、イベントが発生した時に自動的に呼びだされるメソッドのひな形で、この間にボタンが押されたときに行ないたいコードを追加します。

ここでは、以下のように入力してラベルの色を変えることにします。

```
void jButton4_actionPerformed(ActionEvent e) {
    jLabel1.setForeground(new Color(255,255,0));
}
```

ここで“jLabel1”は、さきほど追加したラベルの“name”フィールドの値です。変更した場合は、ここもそれに合わせてください。

残念ながらこの部分は手動で追加する必要がありますが、JBuilderには名前の補完機能があります。図 23 は、途中まで入力した段階で表示されたメソッドの一覧です。ここからクリックすると、入力の手間が省けます。他にもカッコの対応をハイライトで教えてくれたり、メソッドの引数として可能なもの(実数なのか整数なのかなど)を表示してくれたりします。

ここまで入力するとほぼ完成です。実行して“Push!”ボタンを押すと、ラベルの色が黄色に変化します。エラーがある場合は、構造ペインに“エラー”というフォルダが現われ、エラーメッセージが表示されます。また、メッセージビューにも同様のエラーメッセージが表示されます。図 24 は、手動でコードを追加した時に、行末の“;”(セミコロン)を意図的に忘れてエラーを発生させた時の画面です。メッセージビューのエラーメッセージを選択すると、ソースコードの対応する部分がハイライトされます。

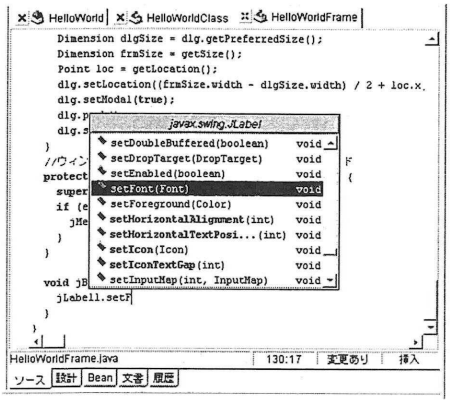


図 23: 名前の補完機能

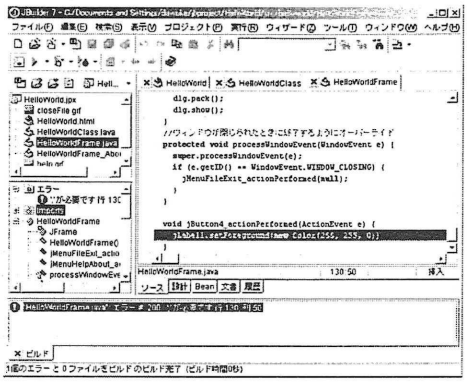


図 24: エラーメッセージ

4.5 レイアウトの追加

ボタンやラベルはコンテナクラスである JPanel 上に配置されています。4.3 節で述べたように、現在のところ JPanel のレイアウトマネージャは “null” になっています。つまり、レイアウトマネージャは使用していません。この場合、プログラムを実行してウインドのサイズを変えても、コンテナ上の部品 (この場合はボタンとラベル) の位置は変わりません。ボタンやラベルなどの部品の位置を、コンテナ上の相対的な位置で指定する場合はレイアウトマネージャを使います。

`HelloWorldFrame.java` を表示させ、ファイルビュータブの [設計] タブを選択してください。ウインド内の白い部分でクリックするか、構造ペインからパネル (“jPanel1”) を選択し、インスペクタの [プロパティ] タブを選択します。“layout” フィールドの右の欄をクリックし、適当なレイアウトマネージャを選択します。ここでは “GridBagLayout” を選択しましょう。これでプログラムの完成です。

5 デバッガ

一度プログラムを書けばそれで上手く動くということは稀でしょう。通常は、編集や変更作業と動作確認作業を何度も繰り返すことになります。この動作確認において、大きな威力を発揮するのがデバッガです。

デバッガとは、プログラム中に潜んだバグの除去作業を支援するツールのことです。デバッガは、作成したプログラムを目的に応じて止めたり、再開したりできます。また、止めた時点で変数の値を出力させ、所望の動作をしているかを確認できます。

通常の統合開発環境は、グラフィカルな操作により簡単に使えるデバッガを備えています。JBuilder のデバッガは [実行] メニューから利用できます。さきほど作った “Hello World” プログラムを対象に、デバッガを起動してみます。[実行] メニューから [ステップイン] か [ステップオーバー] を選択してみます。図 25 のようになります。実行ステータスのボタンで、実行を止めたり再開できます。

[ステップイン] と [ステップオーバー] は、実行する最小単位ごとに実行し、その都度停止し

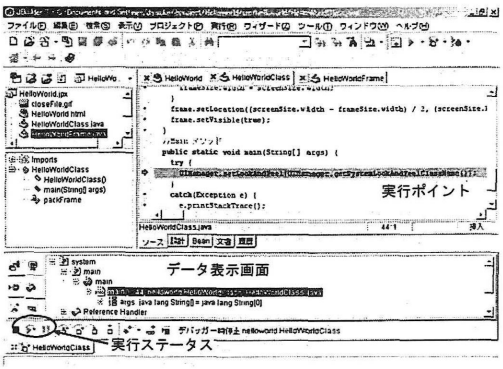


図 25: デバッガの実行画面

ます。どこを実行しているかは、青い矢印で示された行で表示されます (図 25 の実行ポイント)。どちらも基本的にソースコードの 1 行を実行しますが、メソッドの扱いに違いがあります。[ステップオーバー] はメソッドを単なる 1 行として扱いますので、メソッド呼び出し上で [ステップオーバー] を実行すると、このメソッドを全部実行し、呼びだした次の行へ移動します。一方、[ステップイン] はメソッドの最初の行を実行し、次の行へ移動します。

変数の値はデータ表示画面に表示されます。横のタブにより変数の値だけでなく、プログラムの出力やスレッドの状態などを表示させることもできます。

[実行] → [プロジェクトのデバッグ] は、明示的にどこで実行を一時停止するかを指定してある場所 (これを “ブレークポイント” と呼びます) まで実行をします。したがって、ブレークポイントを設定していない時に呼びだしてもあまり意味はありません。

ブレークポイントが設定できる行の左には、小さく青いアイコンが表示されています (図 25 参照)。ここをクリックすると表 2 にあるアイコンが表示されます。これでブレークポイントが設定されます。

表 2: ブレークポイントアイコンの種類

アイコン	意味
	未確認ブレークポイント
	確認済みブレークポイント
	不正なブレークポイント
	使用不可能なブレークポイント

6　おわりに

簡単に JBuilder Personal の使い方を紹介しました。初めてプログラミングをする時など、このように無料でダウンロードできる統合開発環境は非常に有難いと思います。また、ドキュメントも充実していて、この原稿を書くにあたり付属にチュートリアルを参考にしました。

しかし、本格的にプログラミングする場合、特に複数人でプロジェクトを開発していく場合には、JBuilder Personal では不足している機能もあります。個人的には、バージョン管理機能

はもう少し充実させて欲しいなと思いました[11]。

バージョン管理機能は上位のラインナップであるSE版やEnterprise版には実装されています。Personal版を使いこなし、機能の不足を感じたら、これらの製品の購入を考えてみるのもよいでしょう。

[11]ファイルビュータブの[履歴]から多少の情報は得られますが。

screenの利用について

笠原 義晃*

1　screenってなに?

近年、家庭へのネットワーク接続の普及や、ノートPCの普及により、自宅や出張先からネットワーク経由で大学の研究室や情報基盤センターのUNIXワークステーション等を利用する機会が増えているのではないかと思います。

自宅や出張先から大学に接続する場合、telnetやssh(WindowsならTeraTerm + TTSSHやputtyなど)で文字端末を開いて遠隔ログインし、作業する事も多いのではないでしょうか。そういう場合に「回線が不安定で作業中に接続が切れてしまった」「仕事の途中で接続を切らなければならなくなった」「大学の作業の続きを家でやりたい」といった問題が起こる事がままあります。バックグラウンドに落とせるような性質の仕事ならいいのでしょうが、対話的処理だとそうもいかず、接続を切ってしまうとその処理をしているプロセスも死んでしまいます。

そこでフリーソフトscreenの出番です。screenはUNIX系OSで動作し、物理的な一つの端末を複数のプロセス(通常は対話的シェル)で共有できるようにする、言うなれば文字端末用のウィンドウマネージャです。screenを使うと、一つの端末の中で複数個の仮想画面を利用でき、またその仮想画面とその中のプロセスを生かしたまま、自由に端末から切り離したり(detach:デタッチ)、再接続したり(attach:アタッチ)する事ができます。不意に接続が切れても自動的にデタッチしてくれますから、慌てず騒がず再度ネットワークに接続し、遠隔ログインしなおしてアタッチすれば、切れる直前の状態に元通り。やりかけの仕事を、違う時間・違う端末から続ける事も簡単にできます。もちろん、バックスクロールや仮想画面間のカット&ペーストもできますし、画面分割によるマルチウィンドウもサポートしています。

screen自体の歴史は古く、著作権表示には1987年の表示があります。筆者は学生の頃からscreenを愛用しており、もう10年以上使っていますが、存在を知らない人も多いようなので紹介する事にしました。UNIXでシェルを多用している人は是非一度利用してみてください。screen無しの生活は考えられなくなる事請け合いです。

2　コンパイルとインストール

この章ではscreenのインストールについて解説します。

まず、ソースのアーカイブファイルを入手しましょう。2002年10月現在、screenの

*九州大学情報基盤センター
E-mail : kasahara@nc.kyushu-u.ac.jp

最新バージョンは3.9.13です。screen-3.9.13.tar.gzを入手してください。screen本体の配布元はftp://ftp.uni-erlangen.de/pub/utilities/screen/です。かつてscreenは日本語などの2バイト文字に対応しておらずパッチが必要でしたが、現在のscreenには統合されているので特に手当する必要はありません。

アーカイブを入手したら、ソースを展開します。展開するとscreen-3.9.13というディレクトリができます。

```
% gzip -dc screen-3.9.13.tar.gz | tar xf -↵
```

展開できたら、コンパイルしてインストールします。configureが付属しているので簡単です。

```
% ./configure↵
% make↵
% su↵
# make install↵
```

これでインストールは終了です。なお、screenは個人ユーザ権限でのインストールも可能です。

3 使い方

3.1 設定ファイル

インストールは無事済みましたか? それでは実際に使ってみましょう。まず、必要最小限の設定ファイルを作ります。screenの設定ファイル.screenrcはユーザのホームディレクトリ直下に置きます。無くても使えない事はありませんが、少なくとも以下に示す漢字コードとエスケープ文字の設定はしておく事をお勧めします。

```
defencoding eucJP
escape ^Zz
```

「defencoding eucJP」は、入出力に使用する漢字コードの指定です。遠隔ログインなどで使う端末ソフト側に合わせてください。sjisやjisも使えます。日本語EUCがeucじゃなくてeucJPなのは、中国語EUC(eucCN)や韓国語EUC(eucKR)にも対応していてeucだけだと区別できないためです。

「escape ^Zz」は、screen自体へコマンドを送る時に入力するエスケープ文字の指定です。「escape xy」の形で書き、xがエスケープ文字、yがxを入力するのに使うキーとなります。xとyにはそれぞれ「文字1文字」「^に続けて文字1文字(Ctrl+文字の意)」「\に続けて8進数(その数値に対応するASCII文字として解釈)」「\に続けて特殊文字(^や\を字面通りに解釈)」が書けます。

この例では、エスケープ文字には^Z(Ctrlキーを押しながらz)を使う事にし、また^Z自体をscreenの中のプロセスに送りたい時には^Zに続けてzを入力するように設定しています。screenの標準設定ではエスケープ文字は^Aに、エスケープ文字の入力は^A aになっています。しかし、最近の高性能シェルやエディタでは^Aは「行頭に移動」の機能が割り振られている事が多く、これをscreenに奪われてしまうと不便ですので、他のあまり使っていないキーに変更する事をお勧めします。

ちなみに筆者は「escape ^\\\」で^\(Ctrlキーを押しながらバックスラッシュ)をエスケープ文字に指定しています(指定するバックスラッシュの数に注意)。Emacsで^Zを自分用コマンドキーに割り当ててしまっているのと、Emacsでは日本語入力にSKKを利用していて^\を使っていないからです。WnnやCannaなどではこのキーを変換モードの切替に使用するので気をつけてください。他に、^Tにしている例も聞いた事があります[1]。

以下の説明では、エスケープ文字を変更していない(^A)事を仮定して説明します。上の例のように違う文字に変更した人は適宜読み換えてください。

3.2　起動と終了

設定ファイルの準備ができたら、起動してみます。

```
% screen↵
```

screenのバージョンと著作権表示が画面に出力されますので、スペースキーかリターンキーを押してください。通常のシェルが起動するはずです。^Aがscreenに取られている以外は通常の操作方法と変わりません。^Aを入力したい時には^A aと入力してください。また、毎回著作権表示を見たくないという人は、.screenrcに

```
startup_message off
```

と書き足す事により抑制できます。

screenの起動によって生成されたシェルを終了すると、screen自体も終了します。

[1] transpose-chars(カーソル前後の2文字を交換)を多用する人は使えませんが。

```
% exit

[screen is terminating]
%
```

3.3 デタッチとアタッチ

通常 screen の説明ではこの後複数の仮想画面の作り方に行く所ですが、今回は screen の目玉である「デタッチ」「アタッチ」機能の説明を先にしようと思います。

デタッチ

screen を起動して、試しにそこで vi や Emacs みたいな対話的なコマンドを起動してみてください。普通に起動しますね? そうしたら、そのまま^A ^D または^A d と入力します (どちらでも同じ意味です)。

```
~
/tmp/vi.bPXe60: new file, iso-2022-jp: line 1
[detached]
%
```

[detached] という表示と共に、screen を起動したシェルに処理が戻ってきたはずです。screen のプロセスは終了したわけではなく、バックグラウンドで動いています。これがデタッチ操作です。おおまかに図で書くと図 1(次頁) のような感じになります。

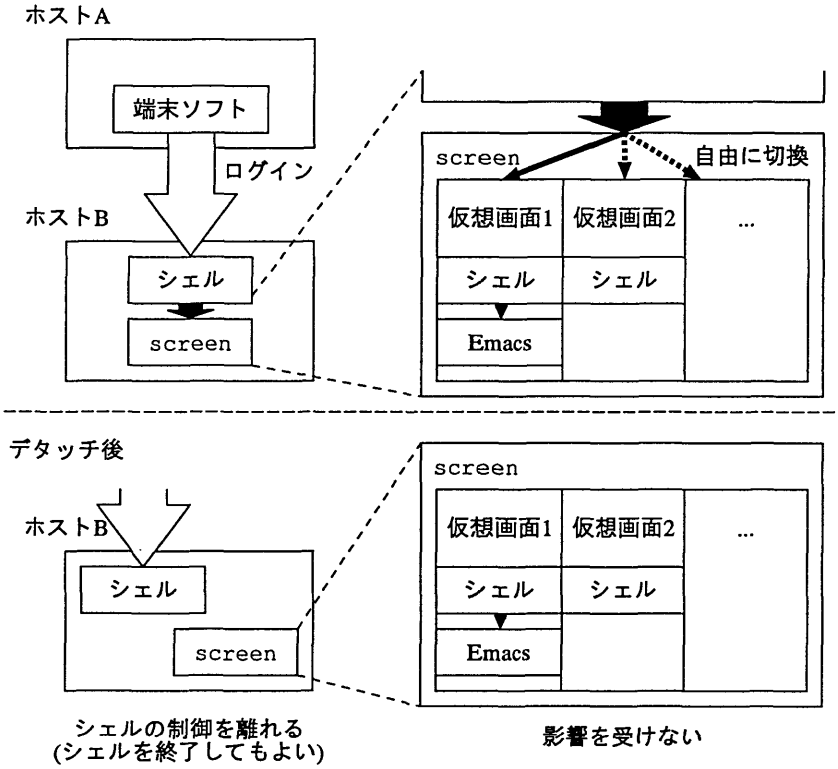
アタッチ

では、元の状態に戻してみましょう。

```
% screen -r↵
```

どうですか? デタッチする前の画面に戻ったと思います。エディタでやりかけの仕事も、デタッチした時の画面のままで復帰したはずです。これがアタッチ操作です。

図 1: screen とデタッチ



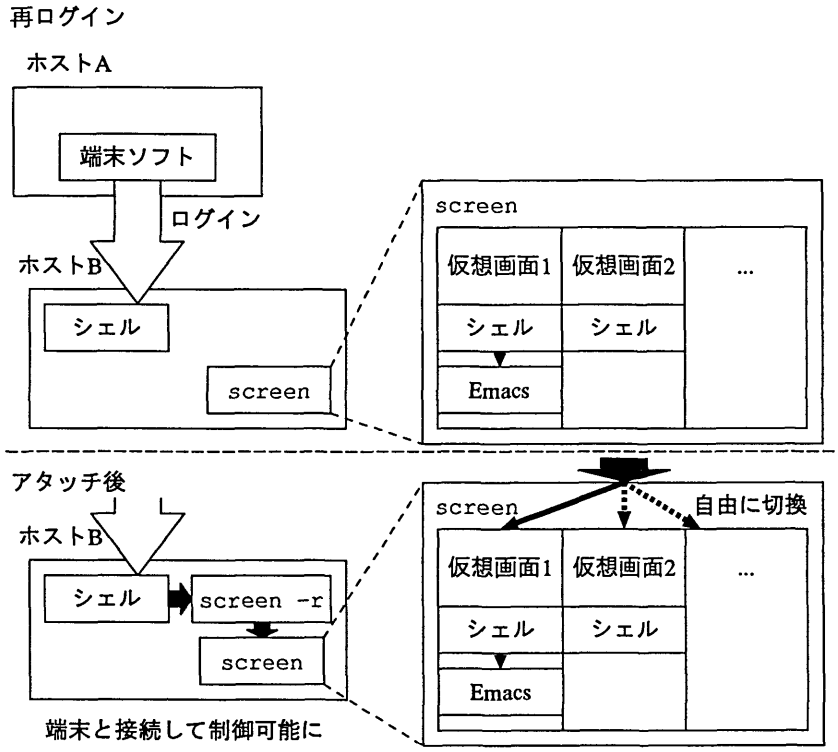
簡単でしょう? screen を使って作業中にネットワーク接続が切れてしまったりしても自動的にデタッチされますから、再度ログインして screen -r すればいいわけです。これで、ふいの接続断も怖くありません。これだけでも screen を使う価値があるというものです。もちろん、一旦デタッチしてログアウトした後、別の端末からログインしてアタッチし、作業を続ける事もできます。

おおまかに図で書くと図 2(次頁) のような感じになります。

screen 一覧

今動いている screen プロセスがいくつあるかを見るには、screen -ls と入力してください。

図 2: screen とアタッチ




```
% screen -ls↵
Your inventory:
        59164.ttyp0.elvenbow    (Detached)
1 Sockets in /tmp/screens/S-kasahara.

%
```

screen が既に端末にアタッチされていると、(Detached) の部分が (Attached) になります。

screen は同時に複数個実行する事ができて、それぞれ別々にアタッチ・デタッチできます。複数の screen がデタッチされている場合、アタッチする時にどれにアタッチするか指定する必要があります。screen -r の引数にプロセス番号を追加してください (上の例では 59164)。

リモートデタッチ

大学でscreenを使っていて、家に帰って続きを思ったけどデタッチしてくるのを忘れていた! なんて時はどうしましょう。普通にscreen -rしようとしても、既に別の端末がアタッチしているので失敗します。

```
% screen -r↵
Your inventory:
        59164.ttyp0.elvenbow    (Attached)
There is no screen to be resumed.
%
```

こういう場合は、screen -dで遠隔にデタッチする事ができます。

```
% screen -ls↵
Your inventory:
        59164.ttyp0.elvenbow    (Attached)
1 Sockets in /tmp/screens/S-kasahara.

% screen -d↵
[59164.ttyp0.elvenbow detached.]

% screen -ls↵
Your inventory:
        59164.ttyp0.elvenbow    (Detached)
1 Sockets in /tmp/screens/S-kasahara.

%
```

これで問題なく再アタッチできるようになります。接続が突然切れた時など、切れ方によってはすぐにデタッチされない事があり、そういう場合にも有効ですので覚えておきましょう。なお、screen -dを使う時にも、screenが複数動いているならプロセス番号の指定が必要になります。

マルチディスプレイモード

screenの面白い機能として、同じscreenプロセスを複数の端末で共有する機能

があります。すでに別の端末がアタッチしているscreenプロセスに対して、

```
% screen -x プロセス番号 (プロセス番号は省略可)
```

と入力する事により、既にアタッチしている端末をデタッチする事なく、追加で別の端末をアタッチする事ができます。X Windowが使える環境の人は、試しに同じホストでktermなどを2つ開き、片方でscreenを実行してもう片方でscreen -xを実行してみてください。片方の画面を操作すると、もう片方の画面も同じように更新されるのがわかるでしょう。デタッチし忘れたまま端末を離れた時に、前項で書いたリモートデタッチの代わりにこの機能を使うという手もあります。

3.4 仮想画面

デタッチ・アタッチだけでもscreenはかなり便利ですが、さらに、screenはその中に複数の仮想画面を保持し、これを自由に切り換えて使う事ができます。TeraTermなどで遠隔ログインして作業している時に、もう一つシェルが欲しくなったり、エディタとシェルを同時に使いたくなる事はありませんか? screenを利用すると、使っている端末機器に依存せず、screenのみで複数の仮想画面を使った様々な機能を利用できます。

仮想画面の作成

端末で編集作業などをしていて、急に他の事を調べたくなり、別のシェルが欲しくなる事があります。もちろん、ジョブコントロール機能や、シェルへのエスケープ機能を使う手もありますが、screenを使っているなら新しい仮想画面を作成するのが簡単です。^A ^Cまたは^A cと入力してみてください。

今まで使っていた画面が消えて、別にシェルが起動した画面に変わりました。今見ている画面が、新しい仮想画面です。この画面は今まで使っていた画面とは全く独立に動いています。仮想画面は(資源が許す限り)何個でも作成する事ができます。作成した仮想画面には順に番号がふられます(後述)。

仮想画面を作成した時に生成されたシェルを終了すると、その仮想画面は消滅し、番号は欠番になります。また全ての仮想画面が消滅するとscreen自体も終了します。

仮想画面間の移動

仮想画面を作ったのはいいけれど、前の画面に戻れないと意味がないですね。

仮想画面間の移動には以下のようなキーが用意されています。全て^A を押した後に押してください。

表 1: 仮想画面移動コマンド一覧

キー入力	動作内容
^space、^N、space、n	仮想画面を昇順に切換
^H、^P、p、backspace	仮想画面を降順に切換
数字キー (0〜9)	その番号の仮想画面に直接切換
^A(エスケープ文字)	直前に表示していた仮想画面に切換

直前に表示していた仮想画面に切り換える機能は、設定ファイルで「escape ^z^z」のように「エスケープ文字」と「エスケープ文字を入力するのに使うキー」を同じにしていると使えません。上記の例だとエスケープ文字を入力するためのキー入力が ^Z ^Z になって、重なってしまうためです。直前の端末に切り換える機能は、2 つの仮想画面を見比べる時に非常に便利なので、できれば使えるような設定にしておく事をお勧めします。

仮想画面の管理

仮想画面には作った順に番号が振られると書きました。番号の確認をするには、^A ^W または^A w を使います。

```
0*$ zsh  1$ zsh  2$ zsh  3-$ zsh
```

実際には白黒反転した表示になります。数字が各画面に振られた番号、*がついているのが現在表示されている画面、-がついているのは直前に表示していた画面です。zsh というのはその仮想画面のタイトルで、通常はシェルの名前になっています。$にも意味がありますが、ここでは割愛します。

このままでは全部同じタイトルでどれがどれだかわからないので、変更してみましょう。現在表示している仮想画面のタイトルは^A A(大文字の A) で変更できます。キー入力すると

```
Set window's title to:  zsh
```

というような表示が出ます。現在のタイトルの部分が編集可能ですので、書き直してリターンキーを押してください。

仮想画面の数が10を越える程になると、番号での直接移動では足らないし、^A wでも全て表示する事ができません。こういう場合には、^A "(ダブルクォート)を使うとよいでしょう。画面がウィンドウの一覧に切り換わり、カーソルキーなどで切り換え先の画面を選ぶ事ができます。

```
Num Name                                    Flags

  0 Emacs                                     $
  2 vi                                        $
  3 zsh                                       $
  4 zsh                                       $
```

3.5 バックスクロール・コピー&ペースト

ここでは、仮想画面のバックスクロールや、コピー&ペーストの方法について説明します。WindowsからTeraTermを使ったりX Windowからktermを使ったりしている場合、これらの端末ソフトにはスクロールバーがついていますし、またマウスを使ってカット&ペーストする事もできます。実際そういう機能が無い端末を使う機会というのは今ではほとんど無いような気もしますが、screenでのやり方を知っていても損はないでしょう。

コピーモードと画面操作

screenでバックスクロールやコピー&ペーストをするには、まず仮想画面をコピーモードと呼ばれる状態に切り換える必要があります。モードを切り換えてscreenの中のシェルなどに影響を与えない状態にして、カーソル移動やコピー操作を行なうわけです。

コピーモードへの移動は^A [または^A ^[(ESCキーでも可)です。画面の最下段に

```
Copy mode - Column 38 Line 4(+100) (80,24)
```

というような表示が出たと思います。数字は入力した時の仮想画面の状態によって若干異なります。Column 38 Line 4は、現在カーソルが左から38文字目の上から4行目にある事、(+100)は画面の上方にスクロールアウトした行を100行まで記憶できる事、(80,24)は仮想画面のサイズが80×24である事を示しています。

コピーモード内では、vi のカーソル移動コマンドと同様の操作でカーソルを移動する事ができます。画面の上端を越えてカーソルを動かそうとするとバックスクロールします。表 2 に主要なコマンドキーをまとめておきます。詳しくは screen のマニュアルなどを参照してください。

表 2: コピーモード内コマンド (一部)

キー入力	機能
h j k l	カーソル移動 (h から順に ←↓↑→)
0 $	行頭、行末
g G	先頭行/最終行に移動
^u ^d	半画面分ずつ上/下スクロール
^b ^f	一画面分ずつ上/下スクロール
/ ?	vi 風前方/後方文字列検索
^s ^r	Emacs 風インクリメンタルサーチ
n	直前の検索を繰り返す

コマンドキー以外のキーを押すか、ESC キーを押すとコピーモードを終了します。

```
Copy mode aborted
```

コピー

コピーモード内でカーソルを自由に動かせるようになったら、次はコピー操作です。^A [でコピーモードに入ったら、コピーしたい領域の最初の文字にカーソルを移動してスペースキーを一回押してください。

```
First mark set - Column 1 Line 5
```

最初のマークがどこに設定されたかが表示されます。これ以降カーソルを移動すると、コピー対象になる部分が反転表示されます。反転表示を見つつカーソルを移動し、コピーしたい部分が選択できたら再度スペースキーを押してください。

```
Copied 260 characters into buffer
```

コピーバッファに何文字コピーされたかが表示され、自動的にコピーモードから通常モードに復帰します。これでコピーできました。

ペースト

ペーストするには、ペーストしたい仮想画面に移動して^A] と入力してください。現在カーソルがある所にコピーバッファの内容が出力されます。ペーストされる文字列はキーから入力しているかのように入力されますので、エディタなどにペーストする時はあらかじめペーストしたい場所にエディタのカーソルを移動しておく必要があります。screenのコピーモードでペーストするわけではない事に気をつけてください。

他にも、矩形領域のコピー&ペーストや、コピーバッファに複数の部分を追加でコピーして行く操作などがありますが、ここでは割愛します。

4 おわりに

今回は、主にUNIXをシェル経由で使っているユーザを対象に、screenというフリーソフトを解説しました。screenには、今回説明した以外にも、画面分割によるマルチウィンドウ機能、画面のパスワードロック機能、バックスクロールログのファイルへの書き出し、画面のスクリーンショット記録などの機能があります。また、.screenrcでかなり細かく設定の調整が可能です。今回説明した範囲で利用していて気に入らなかったり物足りなさを感じ始めたら、マニュアルなどを参考に設定変更に挑戦してみるとよいと思います。検索エンジンで探すと日本語のページも結構見つかると思います。

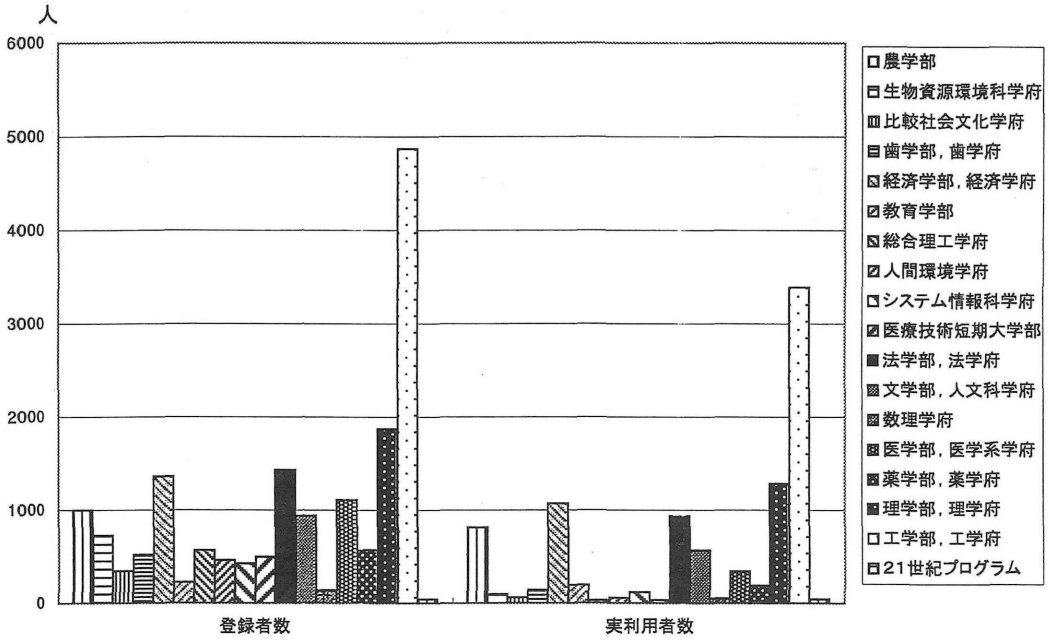
screenでより快適なUNIX生活をお楽しみください。

講 習 会 開 催 内 訳

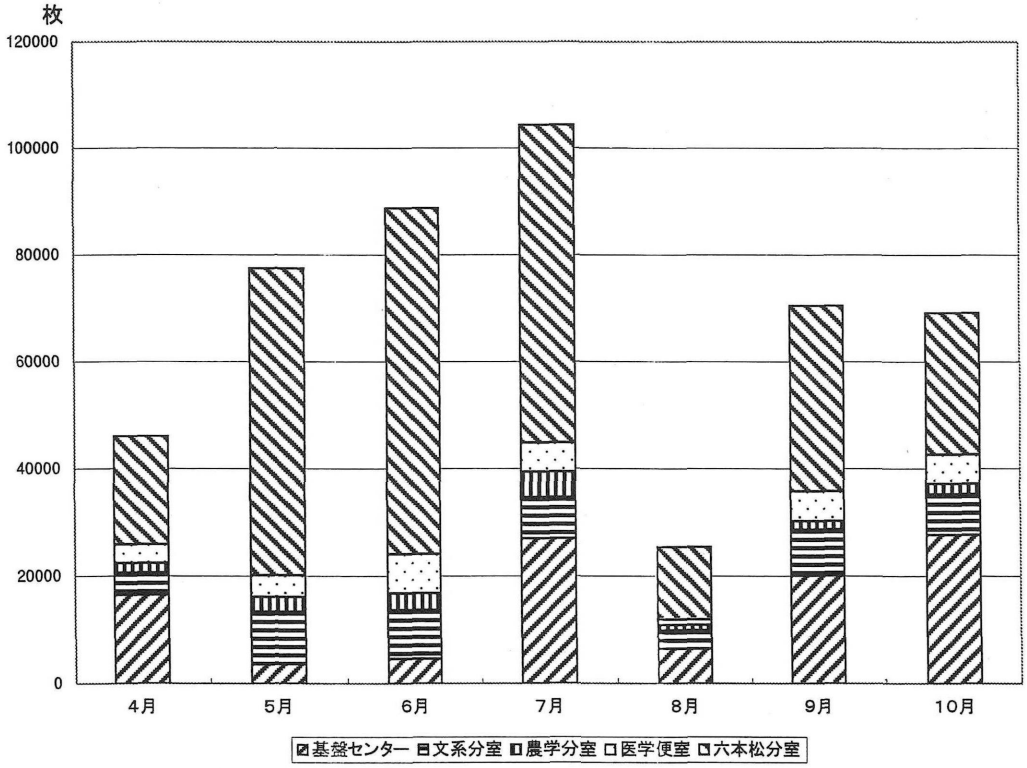
講　習　会　名	開催期日	参加人数
新入生情報処理講習会	14.4.2～4.5	約 2000
授業担当教官(授業補助者)のための利用講習会	14.4.10	20
ファイアウォール説明会	14.4.22	15
教育用システム利用入門	14.4.22～4.23	9
メール説明会	14.5.13	4
ネットワーク初級講習会	14.5.13～5.15	7
ネットワーク中級講習会	14.5.27	4
WebCT 講習会	14.6.13	21
インターネット講習会	14.8.5～8.7	52
WebCT 講習会(初級編)	14.9.25	16
WebCT 講習会(中級編)	14.9.27	13

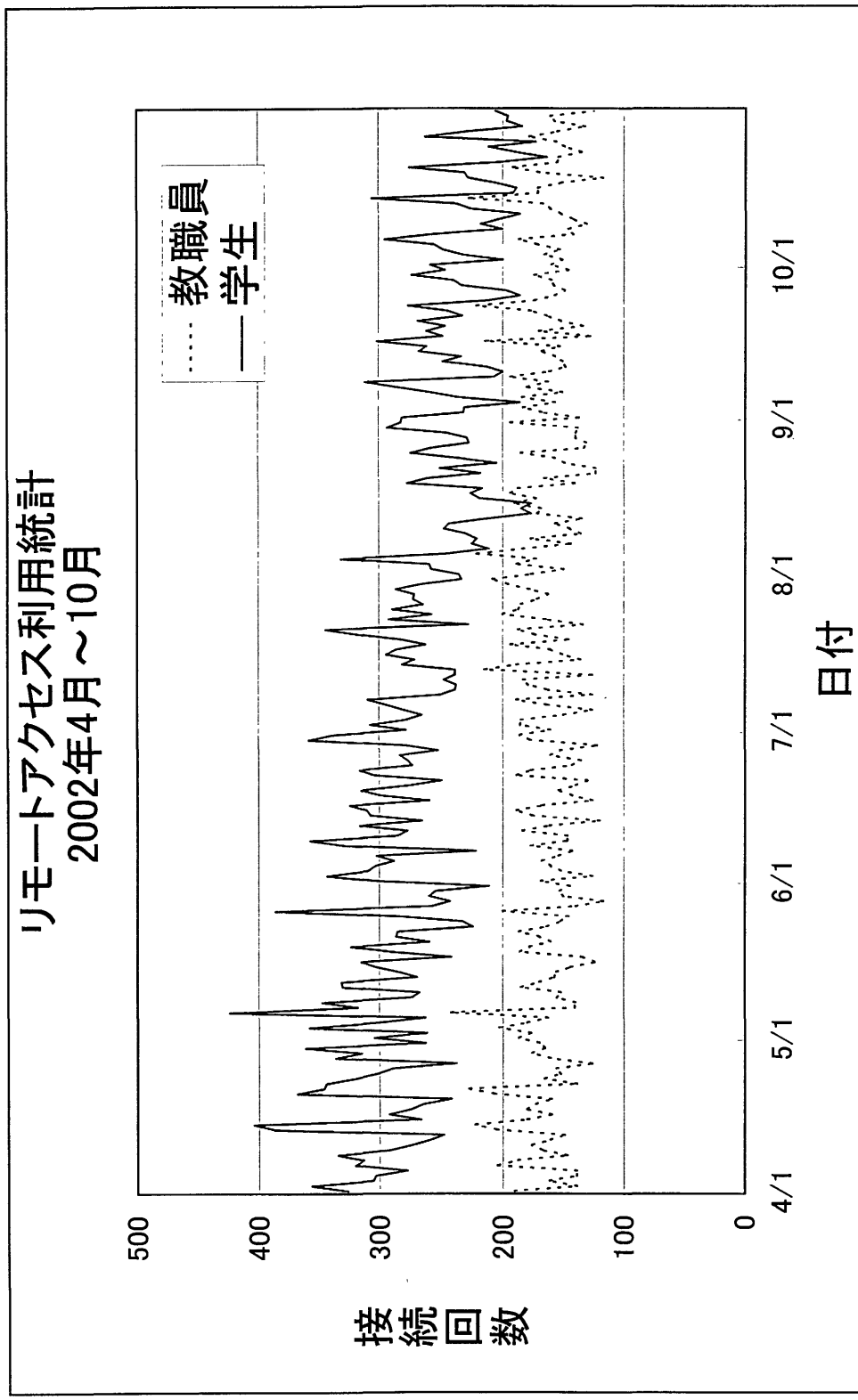
平成14年度教育用システム統計

1．学部等別登録者数および実利用者数(平成14年4月～平成14年10月)



2．月毎分室別のプリンタ出力状況(平成14年4月～平成14年10月)




リモートアクセス利用統計
2002年4月～10月
教職員
学生
接続回数
500
400
300
200
100
0
4/1
5/1
6/1
7/1
8/1
9/1
10/1
日付

人　事　異　動

○平成１４年７月１日付け昇任

文部科学技官　　橋　倉　　聡　　システム運用掛員から
システム運用掛主任へ昇任

○平成１４年８月３１日付け辞職

庶務掛　　事務補佐員　　北　野　秀　美

○平成１４年９月１日付け採用

庶務掛　　事務補佐員（パート）　松　尾　智　恵

編　集　後　記

今回は統合開発環境によるプログラミングの特集を組んでみました。解説されているものは、ほとんど教育システムで使えます。言語でいうとPascal, Visual C++, Visual Basicです。Javaを対象にしたJBuilderは教育システムでは使えませんが、無料でダウンロードして利用できるものです。インストールの仕方も解説してあります。どの記事も簡単なサンプルプログラムの作成を通して、統合開発環境の使い方を説明しています。プログラミングに興味はあっても何から始めてよいか分からないという人も多いと思います。参考にしてください。
(D. I)

先日，所用で倉敷の実家に帰省しました.
博多-新大阪間には 1 時間に 1,2 本，レールスターという新幹線が走っています.
料金は普通の新幹線と同じで，スピードはほぼ「のぞみ」並じゃないかな，と思います(でも，止まる駅が多いので時間はやっぱりこっちの方がかかる).
今では福岡の家から倉敷の実家まで door-to-door で 3 時間弱，一昔前なら新幹線に乗ってるだけでそれだけかかったのに…
ちなみに，倉敷行くのに岡山で降りるのはちょっと行き過ぎになるので，手前の福山(まだ広島県です)で降りて，快速に乗るのが私の父のお薦めです.
でもたまに福山に止まらないレールスターもあるので気をつけましょう.
僕が乗ったレールスターがそうでした…ぷぷぷ.
…いえ，特に他意はなくあまり書くことが思いつかなかったので福岡から倉敷への交通案内をしてみました. (S. T.)

九州大学情報基盤センター広報
Vol. 2, No. 3
平成14年11月 発行
編集 九州大学情報基盤センター
広報委員会
印刷 松隈印刷株式会社