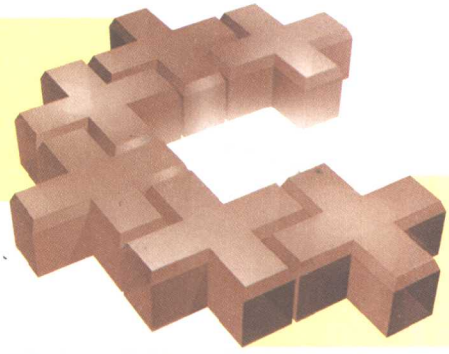
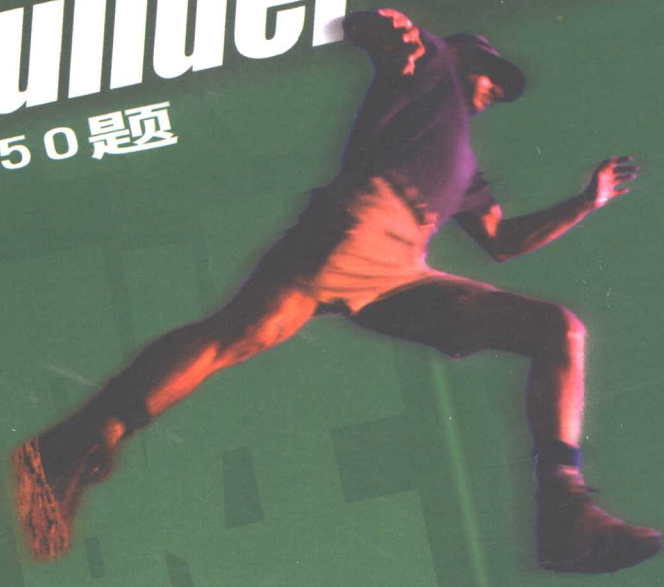


战胜 C++ Builder

必做练习50题

钟光辉 编著



北京大学出版社
http://cbs.pku.edu.cn

战胜 C++Builder 必做练习 50 题

钟光辉　编著

北京大学出版社

北　京

内 容 简 介

C++Builder 5 是 Borland 公司（Inprise 公司）最新推出的功能强大的面向对象、可视化的应用程序开发工具。本书力求全面介绍 C++Builder 编程开发的基本知识，并重点介绍 C++Builder 5 的新特点和使用技术。全书共包含 50 个实用性和代表性很强的实例，通过这些例子，对 C++Builder 的大量组件作了详细的讲解，并举例说明它们的使用。另外对一些比较常用的非可视化的组件类也做了介绍。其目的是帮助用户快速掌握 C++Builder 这个功能强大的编程工具，掌握并且精通 Windows 应用程序的开发方法。

本书示例丰富，语言清晰明朗，适合于使用 C++Builder 编程的初、中级读者。

本书含 1 张教学光盘。

图 书 名：战胜 C++Builder 必做练习 50 题
图书写作者：钟光辉
图 书 责 编：段志刚　黄庆生
光 盘 责 编：王　原
本 版 号：ISBN 7-900636-88-9/TP·65
出 版 者：北京大学出版社
地　　址：北京市海淀区中关村北京大学校内　100871
电　　话：出版部 62752015　编辑部 62765013　发行部 62750672
网　　址：http://cbs.pku.edu.cn　　E－mail:xxjs@pup.pku.edu.cn
印 刷 者：河北省滦县印刷厂
发 行 者：北京大学出版社
经 销 者：新华书店
787 毫米×1092 毫米　16 开本　15.625 印张　397 千字
2001 年 11 月第 1 版　2001 年 11 月第 1 次印刷
定　　价：29 元（含光盘 1 张）

前　言

C++ Builder 是 Borland 公司推出的快速应用程序的开发工具，支持最完整的 C++语言规则，是一个真正可视化 C++语言开发工具，它的主要特点有：

- ◆ 包含众多的组件
- ◆ 有着强大的数据库功能
- ◆ 包含了实用的 Internet 组件
- ◆ 具有平滑的 ActiveX/OCX 接口
- ◆ 支持多媒体的 OLE 编程
- ◆ 具备扩展的中文编程能力

在使用传统的开发工具，比如 Visual C++、Borland C++等进行 Windows 应用程序开发时，需要程序员具有比较高的编程技巧，并且其操作复杂、难度大、应用程序开发效率低，这就使得很多初学者望而却步。在这个背景下，面向对象的可视化的快速应用程序开发得到了飞速的发展。如 Microsoft 的 Visual Basic，Borland 公司的 C++ Builder 和 Delphi，Powersoft 公司的 PowerBuilder 等。这些快速应用程序开发工具提供了大量的组件，通过接近口语化的编程语言，利用可视化的开发环境，使 Windows 应用程序开发得到了极大的简化，提高了编程的效率。开发 Windows 应用程序对于一个初学者来说已经不是一件困难的事情了。

Borland 公司早在 1993 年就推出了第一个快速应用程序开发工具 Delphi 1.0，它提供大量的可以重用的组件，形成一个可视化的组件库，大大方便了 Windows 的开发。Delphi 1.0 一经推出，就以其技术优势取得了巨大的成功。Borland 公司随后在 1995 年推出了 Delphi 2.0，在 1997 年推出了 Delphi 3.0，在 1998 年推出了 Delphi 4.0，到现在已经出现了 5.x 版本。

C/C++作为世界上使用最为广泛的计算机编程语言，在软件开发中具有巨大的影响力。因此，Borland 公司在 1997 年基于 C/C++语言推出了又一个优秀的快速应用程序开发工具 Borland C++Builder 1.0，它继承了 Delphi 的组件库和易于使用等优点，并且充分利用了 C++语言的功能强大灵活的特点，代表了 C++语言的发展方向。Borland C++Builder 一经推出，立即受到广大 C/C++语言使用者的重视。随着其版本的不断更新，其功能也越来越强大。到目前为止，Borland 公司已经推出了 Borland C++Builder 5.x 版本。

本书通过 50 个练习例子，对 C++Builder 的大量组件作了详细的讲解，并举例说明它们的使用。另外对一些比较常用的非可视化的组件类也做了介绍。其目的是帮助用户快速掌握 C++Builder 这个功能强大的编程工具，掌握并且精通 Windows 应用程序的开发方法。

本书由钟光辉主编，另外，余晓鹏、张伟华、何广、张石勇、战祥森、张松伟、吴绍伟、孙科峰、渠继永、覃文圣、牟南、贾鹏、李蘅、钱辰、王晓龙、邓瑞峰、满晓宇、罗捷、肖健、解灵运等也参加了本书编写工作。

编　者

2001 年 11 月

目　录

练习 1　Hello World!

练习目的

从这个练习开始我们将带读者步入 C++Builder 的世界，先从最简单的练习开始做起。人们都已经习惯了 Windows 系统的窗口操作界面，接触的程序都是在窗口上表现出来的，因此生成一个窗口是最最简单的。在这里我们来编写第一个 C++Builder 程序 —— 生成一个标准的 Windows 窗口，在窗口上显示“Hello World!”。通过这个练习，读者可以看到，C++Builder 的集成开发界面是非常友好的。

练习原理

为了在窗口中显示“Hello World!”，先要学习使用 Standard 页中的 Label（标签）控件，当把鼠标放在每个控件的上面时，都会显示每个控件的名称。Label 控件一般都是用来作为别的控件或控件组的标签，它是最简单也是最常用的控件之一。它将产生一个无方框的控件，为静态控件，它不接受键盘的输入，但可以对鼠标的输入进行响应。下面介绍它的一些重要属性：

- Caption 属性　此属性参数为字符串类型，是最重要和最常用的属性之一，此字符串将显示在窗口上。
- Enabled 属性　此属性参数为布尔类型，用来设置 Caption 的参数是否在窗口上激活（将正常显示），或者不激活（参数为灰色），这时不能对这个控件做任何操作。
- Font 属性　是字体属性，可以从弹出的对话框改变 Caption 参数的字体、大小、颜色等。
- Visible 属性　此属性参数为布尔类型，用来设置 Caption 参数是否可见。若设为 false 的话，则什么也看不到，但其各项操作依然有效，这点和 Enabled 不一样，不要搞混了。
- WordWrap 属性　此属性类型为布尔类型。当此参数设为 false 时，标签构件的字符串将在一行中显示；设为 true 时则可以多行显示。

作为显示平台的窗口，根本不用做任何事情。当运行 C++Builder 的时候，它会自动生成一个标准窗口。只要在上面直接进行练习就可以了。

练习过程

运行 C++Builder，可以看到有三个窗口出现。顶部为编程环境的主窗口，它在编程过程中一直存在，直到推出 C++Builder。另外两个，左边的是属性和事件句柄对象监视器，右边就是用来做显示平台的窗口，这里先不管它。

图 1-1　Standard 页

如图 1-1 所示，在主窗口中的控件栏的 Standard 页里找到 Label 控件，双击，这时下面的窗口中会加入这个控件。也可以单击控件，然后在下面的窗口的任何一个位置放置这个控件。选中这个控件，左边的对象监视器就会显示这个控件的各项属性。下面来设置一些属性来实现设定的目的。

这时左边的对象监视器可以看到，这个控件名称为 Label1，属于 Table 类。先找到 Caption 属性，右边为其缺省值 Label1，然后改为“Hello World!”。现在，如图 1-2 所示可以看到右边窗口出现了需要的结果。

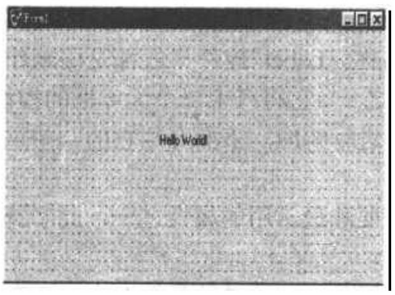


图 1-2　改变 Caption 属性的结果

下面再对标签的其他属性做一些修改，使其看起来更加漂亮。Alignment 是用于设置字符串在标签控件的放置位置的，缺省为 taLeftJustify，即左对齐，这里把它改为居中对齐 taCenter。AutoSize 缺省为 false，即控件大小不随字符串大小变化，为了使控件看起来更加紧凑，把它改为 ture。Color 设置控件的颜色，不是字符的颜色，这里不做改动。下面对字符字体进行设置，点击 Font 前的＋号图标，会下拉出下面的一些参数，里面有字体的各项属性，如图 1-3 所示。

也可以先点击 Font 项，再点击右边出现的“…”，这样可以直接弹出字体设置的对话框，如图 1-4 所示。

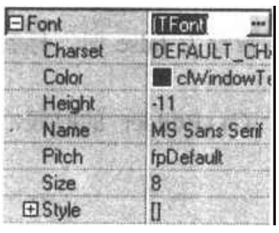


图 1-3　下拉的字体属性

图 1-4　字体设置对话框

按照上图的样子设置好各项参数，这样右边窗口将变为图 1-5 所示，这基本符合了要求。

下面再介绍一种属性 —— 提示属性，在 Hint 项输入要提示的内容，如“C++Builder 欢迎你!”，再把 ShowHint 属性改为 true，当鼠标放在这个控件上方的时候，会自动显示需要的提示内容。最后，按【F9】编译运行，并观察运行的情况，把鼠标移动到那里看看，如图 1-6 所示。

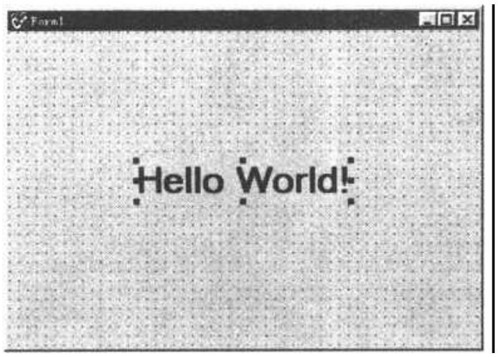


图 1-5　设置完字体的窗口

图 1-6　最后的运行情况

【练习小结】

在这个练习里面，开始接触到 C++Builder 的友好的可视化的编程平台，没有写任何程序代码，整个编程过程都由 C++Builder 自动完成。这个有点“傻瓜”式的编程虽然很简单，但在编写高水平应用程序时，还要把自己的“人工智能”与之相结合，更快更好的完成编程过程。在以后的练习可以看到，即使自己编写代码，其工作量也是很有限的，只需要完成一些程序功能方面的工作。下面点击图标来看看程序代码。

Project1.cpp 的代码如下：

```
#include <vcl.h>
#pragma hdrstop
USERES("Project1.res");
USEFORM("Unit1.cpp", Form1);
//---------------------------------------------------------------------------
WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)
{
        try
        {
                Application->Initialize();
                Application->CreateForm(__classid(TForm1), &Form1);
                Application->Run();
        }
        catch (Exception &exception)
         {
                Application->ShowException(&exception);
```

```
        }
        return 0;
}
```

Unit1.cpp 的代码如下：

```
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
```

由于没有安排任何函数操作，因此，也没有什么东西。

现在来看看 C++Builder 的文件构成。一个 C++Builder 应用程序通常包含基本的三类文件：工程文件（Project File）、窗体文件（Form File）和单元文件（Unit File）。如图 1-7 所示。

工程类文件通常包括以下三种类型的文件（这三个文件是同名的，如上图的 Project1）：

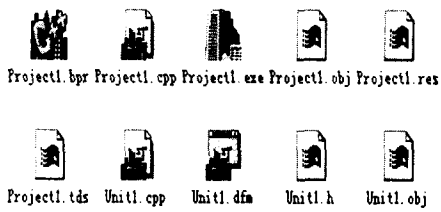


图 1-7　这个练习生成的文件

.CPP 文件含有 WinMain()函数，它就是这个应用程序的起点，可以在菜单“Project\View Source”中看到它的源代码。在通常情况下不要手工修改这个文件；

.BPR 文件包含了工程的结构信息；

.RES 文件是这个应用程序的 Windows 资源文件。

窗体类文件通常包括三种文件，即.cpp、.h、.dfm，这三个文件也是同名的，如上的 Unit1。其中.dfm 文件为描述窗体影像的二进制文件。

单元类文件通常包括.cpp 和.h 两种文件，在这个练习中，没有用到程序单元。

【思考题】

1．运行 C++Builder，熟悉其界面的组成和使用。

2．按照以上方法，自己试着修改一些属性，实现自己的个性要求，并注意修改后的变化。

3．把工程文件和窗体文件重新命名并保存，看看各个文件间的关系。

练习2　窗 体 设 计

练习目的

窗体是 C++Builder 应用程序的基础，使用窗体可以给用户的应用程序提供用户界面。窗体是个特殊化的组件，在它上面可以放置很多组件。因此，弄清楚窗体的设计对程序设计很有好处，甚至可以在程序运行的时候动态地控制窗体。

练习原理

窗体（Form）是一个 Tform 类对象，它的属性、事件包含了许多组件都具备的属性和事件，下面先来看看窗体的一些重要属性和函数：

- ActiveControl 属性　指定窗体上的某个组件为输入焦点。如下面的语句将使窗体上的 Label1 组件成为输入焦点：ActiveControl＝Label1。在同一时刻，应用程序只能有一个输入焦点。
- ActiveMDIChild 函数：这个函数仅在 FormStyle 属性为 fsMDIForm 时有效。它将返回多窗体程序中当前活动的子窗口。
- AutoScroll 属性　缺省为 true，这时当窗体的尺寸不足以显示窗体上所有的组件时，将自动出现滚动条。若改为 false，则必须设置 HorzScrollBar 和 VvVertScrollBar 属性才能使滚动条出现。
- BorderIcons 属性　用来指定在窗体的标题栏上出现哪些图标。它是一个集合，可以有如下元素取值：biSystemMenu 为标题栏上是否有系统控制菜单；biMinimize 为标题栏上是否有最小化按钮；biMaximize 为标题栏上是否有最大化按钮；biHelp 为标题栏上是否有问号图标。系统控制菜单是点击应用程序左上角出来的弹出菜单，如图 2-1 所示，它是系统自带的，如果没有系统控制菜单，这个属性的其他三个元素无效。

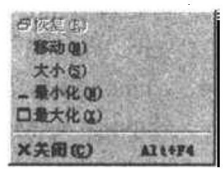


图 2-1　系统控制菜单

- BorderStyle 属性　用来设置窗体的边界风格，它有许多取值：bsSingle 为边界是单细线，不能改变窗体的大小，即使有最大按钮，也通常是禁用的；bsDialog 不能改变窗体大小，没有最大最小按钮，通常用来制作对话框；bsNone 不能改变窗体大小，没有边界，也没有最大最小按钮和系统控制菜单，通常用来制作屏幕保护或程序启

动时的封面；bsSizeable 为标准的可以改变大小的窗体；bsToolWindow 类似 bsSingle，不能改变窗体大小，通常用来作为工具框窗口使用；bsSizeToolWin 可以改变窗体大小，通常用来作为工具框窗口使用。这些属性将影响 BorderIcon 的作用。

- ClientHeight 和 ClientWidth 属性　分别用来确定窗体中可以给用户使用的区域的高和宽，以像素为单位。
- Cursor 属性　指定在窗体上鼠标的图标。
- DefaultMonitor 属性　C++Builder 支持多显示器，这个属性就是用来指定把窗体显示在哪个显示器上，dmDesktop 为不把窗体重新定向；dmPrimary 把窗体显示在主显示器上；dmMainForm 把窗体显示在与主窗体相同的显示器上；dmActiveForm 把窗体显示在与当前活动的窗体相同的显示器上。
- FormStyle 属性　用来指定窗体的风格。有以下取值：fsNormal 窗体既不是父窗体，也不是子窗体（缺省）；fsMDIChild 窗体为子窗体；fsMDIForm 窗体为父窗体；fsStayOnTop 窗体总在最前端。如果一个窗口是多窗体程序的主窗体，这个属性必须设置为 fsMDIForm。
- HelpFile 属性　用来指定这个窗体的帮助文件。
- Icon 属性　用来指定一个图标，这个图标在窗体的左上角上显示，最小化时，这个图标将代表这个窗体。如果没有指定图标，C++Builder 会使用默认的图标。
- Menu 属性　用来指定窗体的主菜单。
- KeyPreview 属性　缺省值为 false，表示击键事件只送到当前有输入焦点的组件上。若改为 true，则这些事件先送到窗体上，再送到当前有输入焦点的组件上，相当于窗体截获了原本属于组件的事件。
- Position 属性　用来设置程序运行时窗体的初始位置：poDesigned 窗体的大小和位置和设计期间一样；poDefault 窗体的尺寸和位置由 Windows 来确定；poDefaultPosOnly 窗体大小不变，但位置由 Windows 确定；poDefaultSizeOnly 窗体位置不变，但大小由 Windows 确定；poScreenCenter 窗体的大小不变，但显示在屏幕的中心位置。
- PrintScale 属性　用来设置打印窗体时是否缩放：poNone 不缩放，打印的大小和屏幕上一样；poProportional 为成比例缩放，尽量接近屏幕上的大小；poPrintToFit 为成比例缩放，尽量适合纸张的大小。
- HorzScrollBar 和 VertScrollBar 属性　用来控制可滚动组件上的水平和垂直滚动条。它们有 ButtonSize、Color、Increment、Kind、Margin、Position、Range、ScrollBarVisible、ScrollPos、Size、Smooth、Style、ThumbSize、Tracking、Visible 等属性。
- WindowState 属性　用来设置窗体的初始状态：wsNormal 窗体既不是最大化也不是最小化；wsMaximized 窗体最大化；wsMinimized 窗体最小化。

练习过程

C++Builder 给用户提供了用来建立 Windows 应用程序窗体的工具，可以很容易地生成从最简单到最复杂的各种不同的窗体。

创建一个新窗体的方式有好几种：

- 运行 C++Builder，在没有改变设置的情况下，C++Builder 将生成一个空白的窗体，同时生成相关的程序单元及相关的项目文件。
- 在集成开发环境的主窗口中选择“File”菜单的“New Application”菜单项，创建一个新的应用程序。在生成应用程序的同时也将生成一个新的窗体和程序单元文件。
- 选择“File”菜单的“New Form”菜单项添加一个新的窗体到现有的项目中。

先运行 C++Builder，调整好产生的空白窗体的大小，把窗体的 Caption 属性改为“练习二”，另外在窗体中放置两个 Label 组件，如图 2-2 所示。

按【F9】运行程序，用鼠标拖动窗体大小，我们可以看到图 2-3 的效果，即窗体能根据窗体中组件的放置情况自动地出现水平和垂直的滚动条。

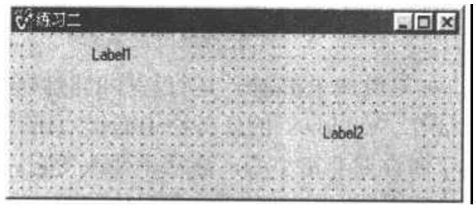


图 2-2　创建一个新窗体

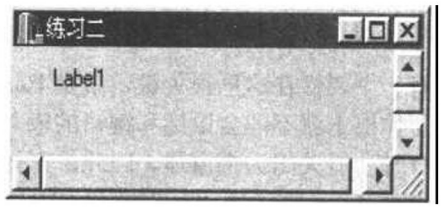


图 2-3　出现滚动条

从左边的对象监视器中可以看到，其 AutoScroll 属性为 true。把它改成 false，而且把 BorderIcon 属性中 biSystemMenu 设为 false，把 biHelp 设为 true，再次运行，可以看到，在窗体的标题栏中，没有了最大化、最小化和关闭的三个按钮，而且连左上角的图标也不见了，点击左上角也没有任何反应。拖动窗体大小，在窗体容纳不下所有的组件的时候，窗体并没有出现滚动条，如图 2-4 所示。只能按【Alt+F4】来关闭这个窗体。

把 BorderIcon 的属性改回原来的值，并把 BorderStyle 的属性由原来的 bsSizeable 改为 bsDialog，从 Cursor 属性的下拉选项中选择 crHourGlass，再次运行程序。如图 2-5 所示，虽然 BorderIcon 的属性改为缺省值，但是 BorderStyle 的属性决定了窗体的标题栏中不出现图标和最大化按钮、最小化按钮，但关闭按钮还有，而且不能用鼠标改变窗体的大小，为典型的对话框的样子。另外由于改变了 Cursor 属性，当鼠标运动到窗体上的时候，鼠标将变为指定的样子。

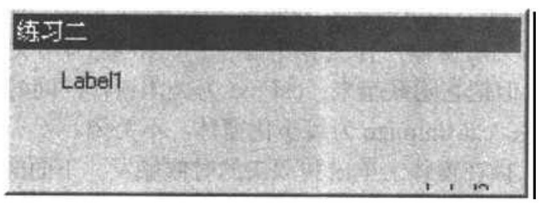


图 2-4　没有滚动条

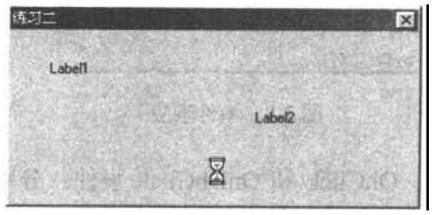


图 2-5　对话框格式

把 BorderStyle 属性改成 bsNone 时，干脆就连标题栏也没有了，根本就没有边界的痕迹，如图 2-6 所示。在运行 C++Builder 时，也可以看到类似这样的封面作用的窗体。

BorderStyle 属性改为原来的 bsSizeable，Cursor 属性也改为原来的 crDefault，再来看看

确定窗体类型的属性 FormStyle 的作用。由于这里只是一个单窗体程序，所以改成 fsStayOnTop。运行，再选择别的应用程序窗口，发现，不管怎样，这个窗体总是能看到，只是不在当前的时候标题栏变灰，如图 2-7 所示。这样的特性在使用一个应用程序而又需要观察另一个应用程序的变化时特别有用。

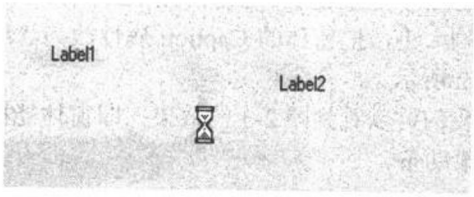


图 2-6　封面作用的窗体

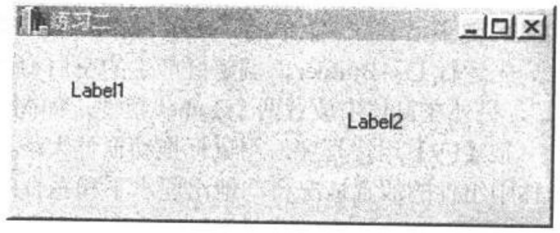


图 2-7　窗体总在最前面

有一个属性在这里很关键，就是 Enabled。如果改成了 false，运行程序的时候会发现，这个窗体根本就不理会鼠标和键盘的输入，只能在 Windows 的任务栏中右键点击，选择关闭命令时才能关闭这个窗体。因此，窗体的这个属性要非常小心，通常不要改变它。

点击 Icon 右边的“…”就会弹出一个图片编辑器，从中选择这个窗体的图标，若没有设置，C++Builder 的集成开发环境将自动选择一个图标。也可以在“Project”菜单的最后一项“Options…”的 Application 页中更改这个缺省的图标。或用代码指定窗体的图标：

```
_fastcall Tform1::Tform1(Tcomponent* Owner):Tform(Owner)
{
Icon->LoadFromFile("C:\\My Documents\\MyIcon.ico");
}
```

下面来看看与窗体有关的一些事件处理。

OnActivate 事件：当窗体变为活动窗体的时候触发。可以用它来在窗体活动前做好初始化的准备。

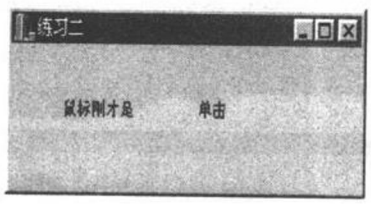


图 2-8　事件响应

OnClose 事件：当程序调用 Close 函数或选择了菜单中的 Close 命令时触发。这样就可以在关闭窗体前完成一些事情。它的声明为：void_fastcall TForm1::FormClose（TObject *Sender，TCloseAction &Action）；其中参数 Action 可以有以下值：caNone 为不关闭窗体，什么也不操作；caHide 为不关闭窗体，但把它隐藏起来；caFree 为关闭窗体，同时释放内存；caMinimize 为最小化窗体，不关闭。

OnClick 和 OnDbClick 事件：分别在鼠标在窗体上单击和双击的时候触发。下面的代码将在窗体的标签上显示鼠标是单击还是双击，如图 2-8 所示：

```
void_fastcall TForm1::FormClick(TObject *Sender)
{
Label2->Caption="单击";
}
//---------------------------------------------------------------------------
```

```
void __fastcall TForm1::FormDblClick(TObject *Sender)
{
Label2->Caption="双击":
}
```

另外还有一些重要的事件，如 OnCreate，它在窗体创建时触发，对于一个窗体来说，只能有一次 OnCreate 事件。其实窗体创建时还会触发以下一些事件：OnShow、OnActivate、OnPaint 等，这些事件都可以多次触发，它们的顺序是 OnCreate 最先，其他依次。可以用这些事件来做一些初始化工作。

可以把自己设计好的窗体当作窗体模板保存下来，只要在这个窗体的设计区点击鼠标右键，从弹出菜单中选择“Add To Repository”，将弹出如图 2-9 所示的对话框。在 Forms 列表框中选择要作为模板的窗体。在 Title 中输入想要在模板库中出现的标题。在 Description 中给模板做出说明，这个说明在浏览模板库时会在状态栏中出现。在 Page 中选择这个模板要放置的位置。另外，也可以点击“Browser”按钮给这个窗体模板指定一个图标。点击“OK”按钮，这个窗体就作为模板保存起来了。

点击“File”菜单的“New…”命令，选择 Forms 页，可以看到如图 2-10 所示，这个模板已经列在其中了。

也可以用 C++Builder 集成开发环境提供的模板向导来创建一个应用程序，同时也创建了一个窗体。

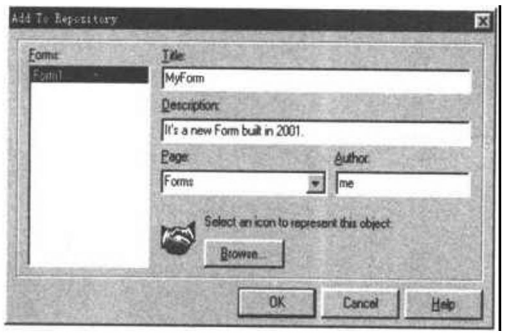


图 2-9　窗体模板保存

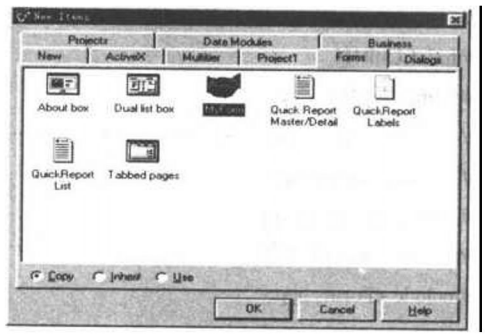


图 2-10　自己的模板

【练习小结】

这个练习中学习了窗体的很多很重要的属性，因为窗体是程序开发的基础，所以这些都要很熟悉和很好地掌握。还可以把自己认为设计得比较完美的窗体作为模板保存下来，为以后的程序开发提供一些便利。与每一个窗体相对应的是由文件*.cpp 和*.h 组成的文件组，它们就是窗体单元程序。在这个练习中在保存的时候可以给这个文件组重新命名，它们必须是一致的。下面来看看这些文件的内容。

先来看看窗体程序文件*.cpp 文件：

```
#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
```

```
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------void----------------------------
void __fastcall TForm1::FormClick(TObject *Sender)
{
Label2->Caption="单击";
}
//---------------------------------------------------------------------------
void __fastcall TForm1::FormDblClick(TObject *Sender)
{
Label2->Caption="双击";
}
//---------------------------------------------------------------------------
```

除了后面两个处理函数外，都是系统自动生成的。它首先包括了一个同名的*.h 头文件，第二部分定义了一个名为 Form1 的 Tform 类对象。

再来看看头文件*.h 的内容：

```
//---------------------------------------------------------------------------
#ifndef Unit1H
#define Unit1H
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
//---------------------------------------------------------------------------
class TForm1 : public TForm
{
__published:        // IDE-managed Components
        TLabel *Label1;
        TLabel *Label2;
        void __fastcall FormClick(TObject *Sender);
        void __fastcall FormDblClick(TObject *Sender);
private:      // User declarations
public:             // User declarations
        __fastcall TForm1(TComponent* Owner);
};
//---------------------------------------------------------------------------
```