程序设计实例解析丛书

C++ Builder 4 实例解析

王小茹　赵 耀　利如安　王 昊　编著

北 京 大 学 出 版 社

北　　京

内 容 提 要

本书是“程序设计实例解析丛书”中的一本，将在 C++ Builder 4 的开发环境里，通过对各方面、各种类型实例从易到难、由浅入深地讲解，使读者尽快掌握 C++ Builder 4。本书主要讲述了以下的实例：注册表使用的演示、设计一个简易的画板、菜单的特殊效果演示、编写一个屏幕保护程序、聊天程序的设计、简单的 E-Mail 收发程序、文件浏览器设计、多媒体播放器程序实例等。另外，本书还讲述了其他许多高级实例。

对于初学 C++ Builder 语言的读者，通过阅读本书将能很快掌握 C++ Builder 语言的特点，而对于高级用户来说，通过阅读本书也能受到不少的启发。

图书在版编目（CIP）数据

C++ Builder 4 实例解析/王小茹等编著．—北京：北京大学出版社，2000.3
（程序设计实例解析丛书）
ISBN 7-301-01613-1

I．C… II．王… III．C 语言—程序设计 IV．TP312

中国版本图书馆 CIP 数据核字（2000）第 06651 号

书　　名：C++ Builder 4 实例解析
著作责任者：王小茹　赵　耀　利如安　王　昊
责 任 编 辑：黄庆生　汉　明
标 准 书 号：ISBN 7-301-01613-1/TP·93
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
网　　址：http://cbs.pku.edu.cn
电 子 信 箱：xxjs@pup.pku.edu.cn
排　版　者：南方立德（Leader）信息技术中心
印　刷　者：
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　19.25 印张　468 千字
2000 年 5 月第 1 版　2000 年 5 月第 1 次印刷
定　　价：37.00 元

目　　录

实例 1　使用不同的鼠标光标资源

◉ 主要内容

& 本例提要

鼠标是 Windows 系统的标准配置，Windows 的应用程序也几乎无一例外地需要使用到鼠标。鼠标的光标指针可以在程序中做到指示程序状态的作用，而且一个动态的鼠标也可以为程序添色不少。现在越来越多的用户都喜欢使用动态的鼠标，Windows 95/98 系统提供了 Plus! 而各种桌面主题中就使用了很多的动态鼠标。如图 1-1 所示的桌面就使用了动态的鼠标。

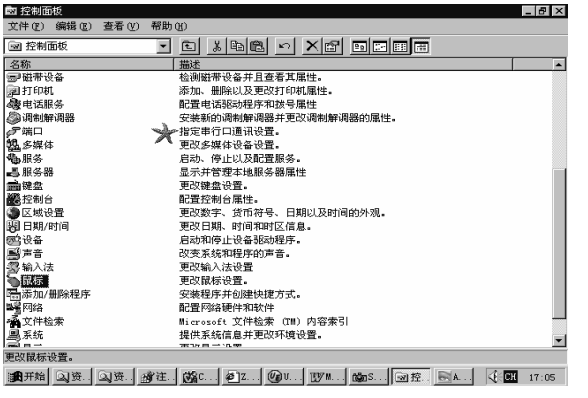

图 1-1　使用了动态光标的系统

本实例将介绍在 C++ Builder 中使用自定义的鼠标资源的各种方法。你将看到，由于 C++ Builder 的开放特性，在程序中使用自定义的鼠标资源将是很容易的事情。

☚ 技术概要

C++ Builder 程序中定义了一个全局变量 Screen，这个变量存放了所有与系统屏幕有关的参数，它是程序在初始化时自动设置的。Screen 中与鼠标有关的两个成员变量是：Cursors 数组和 Cursor 变量。Cursors 数组中存放了系统中所有可以使用的鼠标光标资源，程序也可

以向其中添加自己定义的光标资源。Cursor 变量则指定了程序（或控件）当前使用的光标资源的编号，这个编号就是程序（或控件）所使用的光标资源在 Cursors 数组中的位置。程序（或控件）只需简单地通过改变 Cursor 值就可以改变光标。程序如果要使用自己定义的光标资源，则需要把光标资源装载入程序，这是通过两个 API 函数 LoadCursor 或者 LoadCursorFromFile 来实现的。

● 实例过程

Φ 建立一个新项目 Cursor

选择主菜单中“ File | New Application ”菜单项，创建一个新的应用程序 Cursor，改变程序窗体的 Name 为 CursorForm，在窗体上添加按钮后，效果如图 1-2 所示。

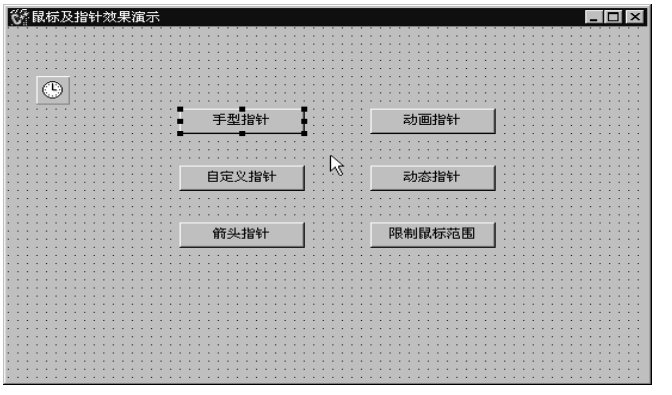

图 1-2　加入控件后的窗体布局

窗体上各控件的名称及属性如表 1-1 所示。

表 1-1　Form 上各控件的名称、属性及设置

控 件 名 称	属　　性	设　　置
TCursorForm	BorderStyle	bsSingle
(TForm)	Caption	系统 API 调用综合演示
	Name	CursorForm
Button1	Caption	手型指针
Button2	Caption	自定义指针
Button3	Caption	箭头指针
Button4	Caption	动画指针
Button5	Caption	动态指针
Button6	Caption	限制鼠标范围
Timer1	Enabled	false
(Ttimer)	Interal	150

编辑完窗体及控件的属性以后，选择菜单项“ File | Save ”保存工程文件。注意将主文件名存为 CursorMain.cpp，工程文件存为 Cursor.dpr。

Φ 添加及修改代码

（1）使用 C++ Builder 提供的 Image Editor，可以编辑自己的光标资源，窗口如图 1-3 所示。注意将光标文件存为.res 的资源文件，自定义一个光标资源并命名为 SCOPE。

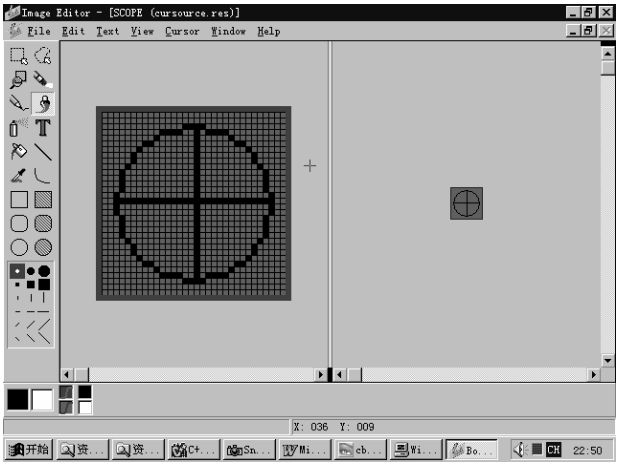

图 1-3　编辑的鼠标资源 SCOPE

将资源文件保存为 cursource.res。由于在程序里使用资源文件比较方便，所以这里使用了资源文件。程序可以使用 Image Editor 来编辑自己的资源文件。

（2）如图 1-4 所示，在 cursource.res 中添加 12 个鼠标光标资源，分别以 CURSOR_1 到 CURSOR_12 来命名。

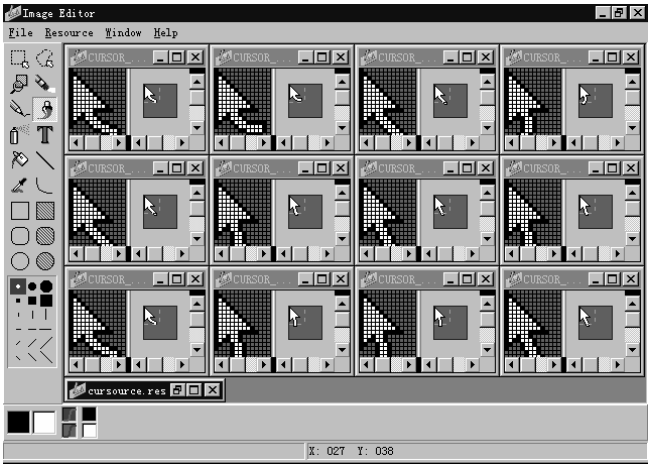

图 1-4　动态鼠标的光标资源

（3）为窗体添加 OnCreate 事件句柄函数 FormCreate，其代码如下所示：

```
//---------------------------------------------------------------
void __fastcall TCursorForm::FormCreate(TObject *Sender)
{
   Screen->Cursors[crScopeCursor]=LoadCursor((void*)HInstance,"SCOPE");
   Screen->Cursors[crAniCursor] = LoadCursorFromFile("wait.ani");
   for (int i=crAniCursor+1; i<crAniCursor+12; i++) {
      Screen->Cursors[i]      =      LoadCursor((void      *)HInstance,
(AnsiString("CURSOR_")+IntToStr(i-crAniCursor)).c_str());
   }
   CurIndex = 1;
}
```

这个函数将鼠标的光标资源句柄载入内存中。资源自载入以后将一直保留在内存中，直到程序释放了资源为止，否则系统不会自动释放资源。这将会造成内存资源的浪费。

（4）为窗体添加 OnDestroy 事件句柄处理函数 FormDestroy，这个事件句柄函数用来释放载入内存的资源。其代码如下所示：

```
//---------------------------------------------------------------
void __fastcall TCursorForm::MyUnClipCursor(TObject *Sender)
{
   TRect rect = Rect(0, 0, Screen->Width, Screen->Height);
   ClipCursor( &rect );
   Button6->OnClick = MyClipCursor;
}
```

（5）为 Button 1 添加 OnClick 事件句柄函数 Button 1 Click，其代码如下所示：

```
//---------------------------------------------------------------
void __fastcall TCursorForm::Button1Click(TObject *Sender)
{
   Timer1->Enabled = false;
   Screen->Cursor = crHandPoint;
}
```

这个函数将程序的鼠标光标换成一个手型的光标，实例的运行演示如图 1-5 所示。

（6）为 Button 2 添加 OnClick 事件句柄函数 Button 2 Click，其代码如下所示：

```
//---------------------------------------------------------------
void __fastcall TCursorForm::Button2Click(TObject *Sender)
{
   Timer1->Enabled = false;
   Screen->Cursor = crDefault;
```

```
}
```

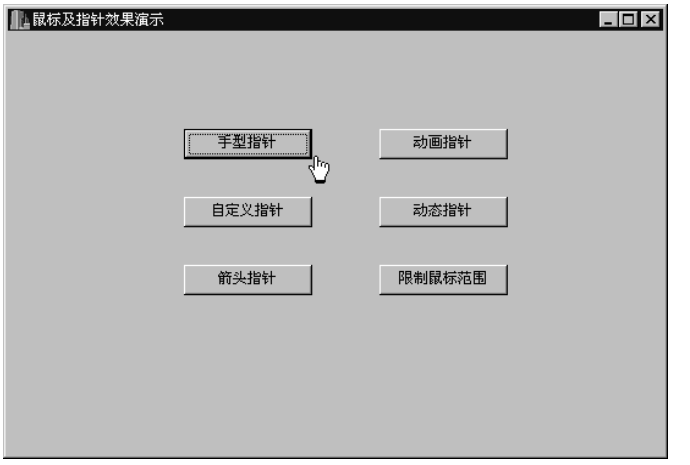

图 1-5　使用手型光标的程序实例

这个函数将程序的鼠标光标换成缺省的光标，一般是一个白色的箭头光标。程序的演示如图 1-6 所示。

图 1-6　使用缺省光标的程序实例

（7）为 Button 3 添加 OnClick 事件句柄函数 Button 3 Click，其代码如下所示：

```
//---------------------------------------------------------------
void __fastcall TCursorForm::Button3Click(TObject *Sender)
{
   Timer1->Enabled = false;
   Screen->Cursor = crScopeCursor;
}
```

这个函数的作用是将程序的鼠标光标换成自定义的一个圆圈型光标。程序实例演示如

图 1-7 所示。

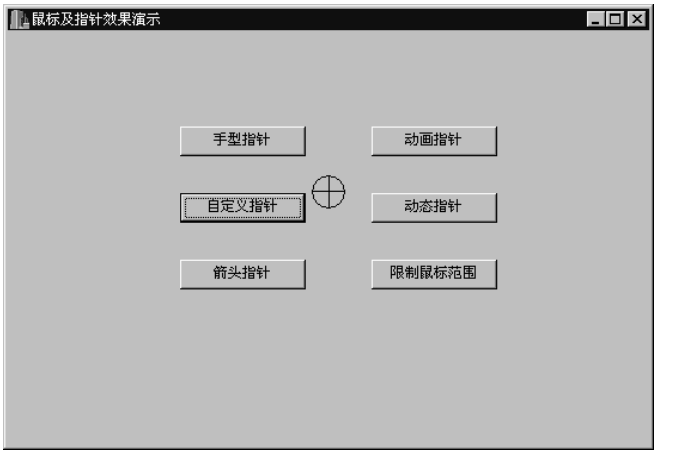

图 1-7　程序使用了自定义的圆圈型光标

（8）为 Button 4 添加 OnClick 事件句柄函数 Button 4 Click，其代码如下所示：

```
//---------------------------------------------------------------
void __fastcall TCursorForm::Button4Click(TObject *Sender)
{
   Timer1->Enabled = false;
   Screen->Cursor = crAniCursor;
}
```

这个函数使用.ani 格式的动画光标，现在越来越多的人喜欢使用这种动画光标。程序中使用动画光标和使用一般的光标其实都很简单，基本上没有什么差别。程序实例演示如图 1-8 所示。

图 1-8　使用动画光标的实例演示

（9）为 Button 5 添加 OnClick 事件句柄函数 Button 5 Click，其代码如下所示：

```
//---------------------------------------------------------------
void __fastcall TCursorForm::Button5Click(TObject *Sender)
{
   Timer1->Enabled = !Timer1->Enabled;
   if ( !Timer1->Enabled )
      Screen->Cursor = crDefault;
}
```

（10）为 Timer 1 添加 OnTimer 事件句柄函数 Timer 1 Timer，其代码如下所示：

```
//---------------------------------------------------------------
void __fastcall TCursorForm::Timer1Timer(TObject *Sender)
{
   Screen->Cursor = CurIndex + crAniCursor;
   if ((++CurIndex)>12) CurIndex %= 12;
}
```

这个函数的作用是定时地更新程序的鼠标光标，以达到动态光标的效果。这要求制作出动态的光标资源，然后程序定期地调用不同的光标就可以了。这里只演示使用动态光标的情形，如图 1-9 所示。

图 1-9　使用动态光标的一幅演示

（11）为 Button 6 添加 OnClick 事件句柄函数 MyClipCursor，其代码如下所示：

```
//---------------------------------------------------------------
void __fastcall TCursorForm::MyClipCursor(TObject *Sender)
{
   TRect rect;
   rect = Button6->BoundsRect;
```

```
    MapWindowPoints(Handle, NULL, (POINT*)&rect, 2);
    ClipCursor( &rect );
    Button6->OnClick = MyUnClipCursor;
}
```

这个函数的作用是将鼠标的活动范围限制在某一个范围之内，图 1-10 演示了光标被限制在一个按钮的范围之内的情况。Button 6 被单击后，它的 OnClick 事件处理函数将被改为 MyUnClipCursor，其作用是解除对鼠标活动范围的限制。

（12）在 TCursorForm 定义中添加一个函数 MyUnClipCursor，其代码如下所示：

```
//---------------------------------------------------------------
void __fastcall TCursorForm::MyUnClipCursor(TObject *Sender)
{
    TRect rect = Rect(0, 0, Screen->Width, Screen->Height);
    ClipCursor( &rect );
    Button6->OnClick = MyClipCursor;
}
```

图 1-10　程序将鼠标的活动范围限制在一个按钮的范围以内

这个函数的作用是解除对鼠标活动范围的限制。

（13）在头文件中添加如下的常数定义：

```
#define crScopeCursor (1)
#define crAniCursor  (2)
```

在 TCursorForm 定义中添加一个 private 成员变量，它的定义如下：

```
int CurIndex;
```

编译并运行该应用程序，该实例运行后的界面如图 1-11 所示。单击不同的按钮将看到不同的鼠标光标效果。

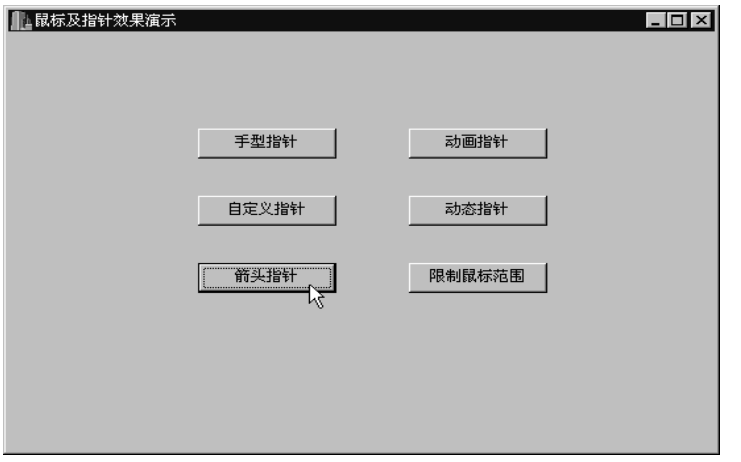

图 1-11　程序实例运行效果

♦ 实例解析

※ 载入光标资源

由于系统提供的光标资源已经被该系统载入，所以在程序中只需要直接使用就可以了。如果要使用自己的光标资源，就需要先载入它。光标资源被编辑后，可以生成.res 的资源文件，也可以生成.cur 格式的光标文件。使用资源文件时，需要在程序开始部分加上一句 #pragma resource "cursource.res"，其中 cursource.res 是本实例中编辑的资源文件名。然后定义一个光标句柄 HCURSOR，通过调用 API 函数 LoadCursor 就可以实现光标资源的载入。这个函数的定义格式如下所示：

```
HCURSOR LoadCursor(
   HINSTANCE hInstance,
   LPCTSTR lpCursorName
);
```

函数的参数很简单，hInstance 是程序的实例句柄，可以使用 C++ Builder 为程序定义一个全局变量 HInstance；lpCursorName 是资源文件中定义的光标资源的名称。函数的返回值将是光标资源的句柄。将此句柄值赋给程序的 Screen->Cursors[]数组，然后指定程序中控件的 Cursor 属性为刚才那个光标资源的索引值，就可以改变控件上的光标了。

如果是.cur 或者.ani 格式的光标文件，则调用另外一个 API 函数 LoadCursorFromFile 来载入光标资源。这个函数的定义格式如下所示：

```
HCURSOR LoadCursorFromFile (
   LPCTSTR lpFileName
```

```
);
```

参数 lpFileName 就是光标资源文件名。调用这个函数就可以直接把动画光标载入，动画光标与普通光标的使用方法一样，程序不需要再做什么额外的工作，系统会完成其他的工作。

※ 制作动态光标指针

制作动态光标指针其实很简单：先在程序中制作好要使用的光标资源，一般为一组动作连续的光标资源。程序载入了光标资源以后，使用一个 Ttimer 控件，动态地更新光标资源就可以了。

※ 限制光标指针的活动范围

如果程序要限制光标指针的活动范围，就可以通过一个 API 函数的调用来完成，这个函数就是 ClipCursor，函数的定义格式如下所示：

```
BOOL ClipCursor(
   CONST RECT *lpRect
);
```

参数 lpRect 是被限制的鼠标指针的活动范围，这个范围是相对于全屏幕而言的。因此如果程序是通过控件的 BoundsRect 属性来得到控件的范围，就需要调用另一个 API 函数 MapWindowPoints，将对应于窗体上的区间范围转换成对应于全屏幕的区间范围。

如果程序需要解除对鼠标活动范围的限制，可以再次调用 ClipCursor，只是需要将参数 lpRect 所表示的范围改变为全屏幕的范围。

* 小　　结

本实例演示了改变程序窗体上鼠标指针的方法，以及如何限制鼠标指针的活动范围。程序不仅可以改变鼠标指针，还可以使用自己编辑的光标资源。灵活地使用鼠标指针，可以为程序添色不少，也可以使程序更具有自己的风格。

实例 2　注册表使用的演示

◉ 主要内容

& 本例提要

在 Windows 3.x 系统中，应用程序和系统都是使用 .Ini 格式的系统配置文件来保存信息的。这样导致了系统的 .ini 文件越来越大，越来越长。在 Windows 95/98/NT 中，系统改用注册表来保存文件以及应用程序的各种信息。注册表其实就是一个数据库，应用程序也可以在注册表中存放各种需要保存的信息。图 2-1 就是使用注册表编辑器打开注册表后，显示出的注册表中的一些应用程序的信息。

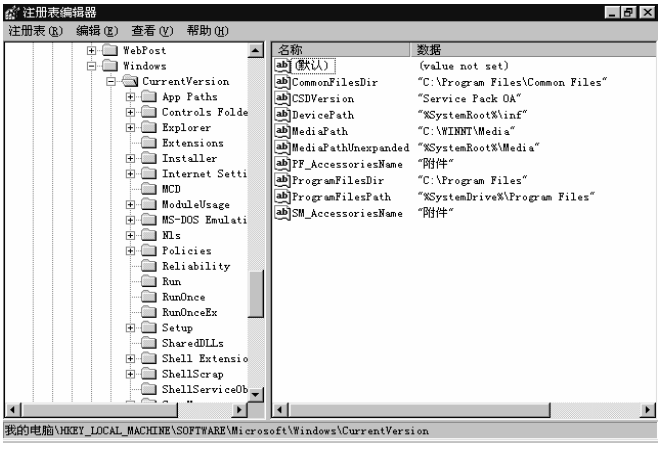

图 2-1　注册表编辑器打开的注册表

本实例将介绍在 Windows 系统的注册表中存放各种信息的方法，使用这些方法，用户可以在注册表中存放包括整数、浮点数、布尔值以及字符串等各种数据信息。

➧ 技术概要

本实例将生成 TRegistry 类的一个对象 Reg，通过此对象的方法（Method）来访问和保存需要的注册表键值信息。使用 Reg 对象时需要先对 Reg 的 RootKey 属性赋值，可以赋的

值与注册表的几个根键有关。然后就可以使用 OpenKey 来打开注册表中的某个键，访问其中的键值了。

⬣ 实例过程

Φ 建立一个新项目 Registry

选择主菜单中“ File | New Application ”来创建一个新的应用程序 Registry，改变程序窗体的 Name 为 RegistryForm，在窗体上添加按钮之后如图 2-2 所示。

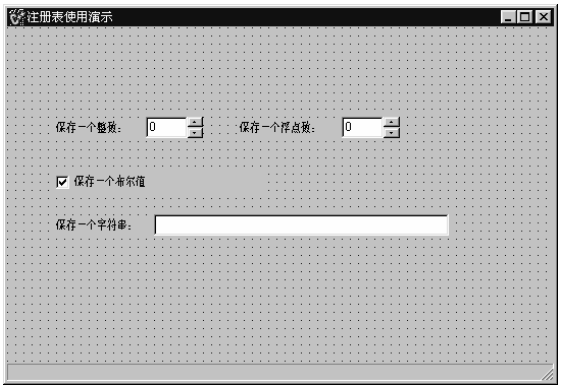

图 2-2　加入了控件以后的窗体布局

窗体上各控件的名称及属性如表 2-1 所示。

表 2-1　Form 上各控件、属性及设置

控 件 名 称	属　　性	设　　置
RegistryForm	Caption	注册表使用演示
(TForm)	Name	RegistryForm
Label1	Caption	保存一个整数
(TLabel)	FocusControl	Edit1
	Left	48
	Top	92
Edit1	Left	136
(TEdit)	MaxLength	3
	Text	0
	Top	88
	Width	41
UpDown1	Associate	Edit1
(TUpDown)	Max	100
	Min	0
	Wrap	True

（续表）

控 件 名 称	属　　性	设　　置
Label3	Caption	**保存一个浮点数**
(TLabel)	FocusControl	Edit3
	Left	228
	Top	92
Edit3	Left	328
(TEdit)	MaxLength	4
	Text	0.0
	Top	88
	Width	41
UpDown1	Associate	(**空**)
(TUpDown)	Max	100
	Min	0
	Wrap	true
CheckBox1	Caption	**保存一个布尔值**
(TEdit)	Checked	true
	Left	48
	Top	144
Label2	Caption	**保存一个字符串**
(TLabel)	FocusControl	Edit2
	Left	48
	Top	188
Edit2	Left	144
(TEdit)	MaxLength	0
	Text	(**空**)
	Top	184
	Width	289
StatusBar	Name	StatusBar
(TStatusBar)	SimplePanel	true

编辑完窗体及控件的属性以后，选择菜单项“ File|Save **”保存工程文件。注意将主文件存为** RegistryMain.cpp**，工程文件存为** Registry.dpr**。**

Φ 添加及修改代码

（1**）为窗体添加** OnCreate **事件句柄处理函数** FormCreate**，其代码如下所示：**

```
//---------------------------------------------------------------
void __fastcall TRegistryForm::FormCreate(TObject *Sender)
{
   try
   {
      TRegistry* Reg = new TRegistry;
      Reg->RootKey = HKEY_LOCAL_MACHINE;

      if (Reg->OpenKey("Software\\Registry Demo", true))
```

```
        {
            Width  = Reg->ReadInteger("Width");
            Height = Reg->ReadInteger("Height");
            Left   = Reg->ReadInteger("Left");
            Top    = Reg->ReadInteger("Top");
            UpDown1->Position  = Reg->ReadInteger("Integer");
            UpDown2->Position  = (int)Reg->ReadFloat("Float") * 10;
            CheckBox1->Checked = Reg->ReadBool("CheckBoxValue");
            Edit2->Text = Reg->ReadString("String");
        }
    }
    catch (...)
    {
    }

    double value = UpDown2->Position / 10.0;
    Edit3->Text  = FloatToStrF(value, ffFixed, 10, 1);
    UpdateStatusBar();
}
```

这个函数的作用是在窗体初始化的时候，从注册表中读取程序上次运行后保存的数据信息，并用这些数据信息来初始化窗体中一些控件的属性。

（2）为窗体添加 OnDestroy 事件句柄处理函数 FormDestroy，其代码如下所示：

```
//---------------------------------------------------------------------
void __fastcall TRegistryForm::FormDestroy(TObject *Sender)
{
    try
    {
        TRegistry* Reg = new TRegistry;
        Reg->RootKey = HKEY_LOCAL_MACHINE;

        if (Reg->OpenKey("Software\\Registry Demo", true))
        {
            Reg->WriteInteger("Width", Width);
            Reg->WriteInteger("Height", Height);
            Reg->WriteInteger("Left", Left);
            Reg->WriteInteger("Top", Top);
            Reg->WriteInteger("Integer", UpDown1->Position);
            Reg->WriteFloat("Float", UpDown2->Position / 10.0);
```