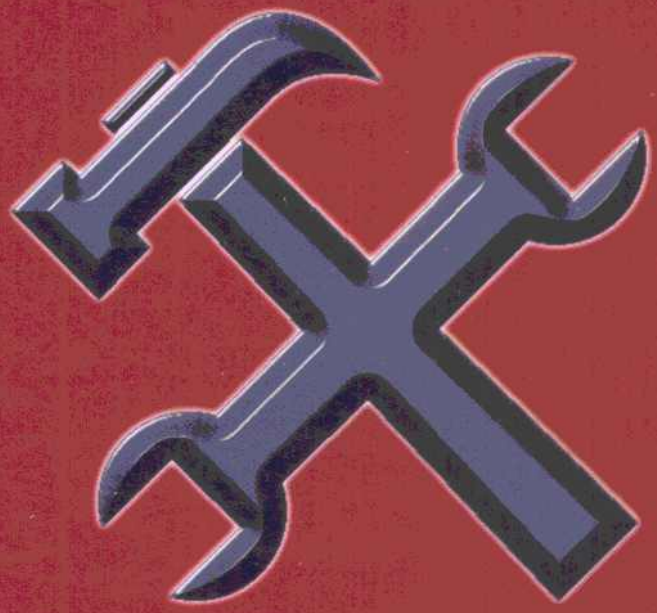
经典的实例　详细的解析
编程的精品　提高的宝典

Delphi 5 实例解析

王小茹　黄健文
林坚立　戴扬波　编著



北京大学出版社
PEKING UNIVERSITY PRESS
URL:http://cbs.pku.edu.cn

程序设计实例解析丛书

Delphi 5 实例解析

王小茹　黄健文　林坚立　戴扬波　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 Delphi 5 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 Delphi 5。本书主要讲述了以下的实例：一个小型运算器的实现、文本编辑器的设计、一个画图软件的制作、设计一个屏幕截图软件、图像放大及.INT 文件、多媒体播放器的应用、图像的特效显示、利用 CopyRect 方法实现抓图、文件浏览器等。另外，本书还讲述了其他许多高级实例。

对于初学 Delphi 5 语言的读者，通过阅读本书，将能很快掌握 Delphi 5 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

Delphi 5 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 7-301-01541-1

I. D… II. 王… III. Delphi 语言—程序设计 IV. TP312

书　　名：Delphi 5 实例解析
著作责任者：王小茹　黄健文　林坚立　戴扬波
责任编辑：黄庆生　汉　明
标准书号：ISBN 7-301-01541-1/TP·89
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：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　27.75 印张　675 千字
2000 年 6 月第 1 版　2000 年 6 月第 1 次印刷
定　　价：43.00 元

前　言

Delphi 是 Inprise 公司于 1995 年推出的 Windows 环境下可视化编程语言，在短短几年中已经发行了 100 万份以上，目前国内流行的版本为 3.0X 和 4.0X。最近 Inprise 又推出了功能更为强大的 Delphi 5。经著名的 PC Week Labs 测试证明：Delphi 5 包括了 Visual C++6.0 与 Visual Basic 6.0 的功能和优点，而且不像微软公司刚推出 Visual Studio 的工具组合只能提供个别、单一的开发环境，Delphi 5 提供较佳的环境与执行速度，其速度比 Visual Basic 6.0 快了 3～6 倍以上。的确，Delphi 将我们从复杂的 Windows 编程中解放出来，极大地提高了编程效率。

尽管目前介绍 Delphi 的书很多，但多数偏重于介绍语言本身。而本书则主要是从应用的角度来介绍 Delphi 5，对一些在应用中经常涉及的几个方面进行较深入的讨论，包括文件、图形、数据库、多媒体、游戏、网络等多方面的内容。希望对开发 Windows 应用程序的读者能起到抛砖引玉的作用。本书所介绍的例子囊括了 Delphi 的几个重要的方面，为读者提供了一定的参考。特别的是，读者可以在本书的例子中添加自己的代码，增强程序的功能；也可以把几个相关例子综合起来，写出属于自己的功能强大的程序。而且，我们也希望这本书能让更多的人喜欢 Delphi 和使用 Delphi。

在本书的编著过程中，笔者注意尽量减少冗长无味的说明，代之以具体实用的例题演示。通过例题，引导读者把握 Delphi 的精髓所在。本书注重开发实例、开发经验、开发技巧和 Windows 高级特性开发，适合于各个层次的 Delphi 用户。对初学者来说，可迅速加入 Delphi 高级用户的行列；对有一定使用经验的读者，也可通过本书掌握 Delphi 深层次的开发方法，学会用更巧妙的办法开发出高水平的 Delphi 应用。

最后提出的一点是，本书全部实例均是在 Delphi 5 的开发环境下编译通过，均可以在 Windows 98 环境下运行。

“聪明的程序员使用 Delphi”，这并不仅仅是因为其比 Visual C++等其他高级语言更易于使用，更重要的是 Delphi 本身对程序开发提供了强大的支持，为 Delphi 开发的控件更使其如虎添翼。让我们都成为聪明的程序员，一起走进 Delphi 的世界。

本书由孙景利策划，王小茹、黄健文、林坚立、戴扬波主编，另外，陆谊、丁雨、黄少棠、瞿小玉、黄瀚华、凌贤伍、胡梦霞、姚玉霞、孙敬娜、付红梅、康孟霞、张小东、李宁、王强、赵四海、李晓峰、董团结、杨仕润、韩白、涂海滨、张旭、张志明、朱黎、周刚兵、张华开、王登峰、郑忠良、李静、刘天翠等也参加了全书的编写工作，在这里对他们表示诚挚的感谢。

限于作者水平和有限的时间，难免在内容选材和叙述上有不当之处。竭诚欢迎广大读者对本书提出批评和建议。

编　者

2000 年 4 月

实例 1 一个小型运算器的实现

主要内容

本例提要

本例子通过讲解一个小型计算器的设计，向读者介绍关于创建基于对话框的程序，使读者尽快熟悉 Delphi 的集成环境以及一些相关的基本操作，并且了解有关面向对象程序语言(Object-Oriented Language)的一些基本思想。建议读者按照本书介绍的过程，在你的电脑上直接操作。使你对 Delphi 的可视化编程有一个直观、快捷的了解，并将起到事半功倍的效果。

设计一个计算器，首先必须产生设计规则。怎样提前估计并不知道的预期结果，想要的是什么结果呢？对于本程序来说，计算器将支持标准的 4 种功能（加，减，乘，除），并且提供 8 位数的显示，还需要清除输入（C）及清除所有输入内容的按键（CE）。此外，还需在提高计算器的版本时能包括新的功能（例如标准的科学计数法功能）。

程序运行的结果如图 1-1 所示。

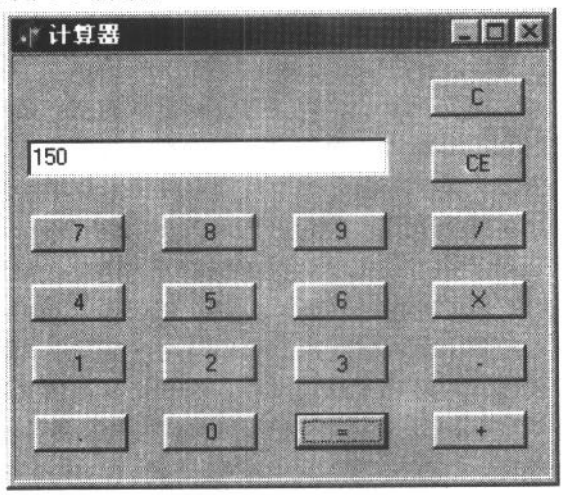


图 1-1 计算器运行结果

☞ 技术概要

Delphi 是全新的可视化编程环境，为我们提供了一种方便、快捷的 Windows 应用程序开发工具。它使用了 Microsoft Windows 图形用户界面的许多先进特性和设计思想，采用了弹性的可重复利用的完整的面向对象程序语言(Object-Oriented Language)，是当今世界上最快的编辑器、最先进的数据库技术。对于广大的程序开发人员来讲，使用 Delphi 开发应用软件，无疑会大大地提高编程效率；而且随着应用的深入，你将会发现编程不再是枯燥无味的工作——Delphi 的每一个设计细节，都将带给你一份欣喜。

Delphi 实际上是 Pascal 语言的一种版本，但它与传统的 Pascal 语言有天壤之别。一个 Delphi 程序首先是应用程序框架，而这一框架正是应用程序的“骨架”。在骨架上即使没有附着任何东西，仍可以严格地按照设计运行。你的工作只是在“骨架”中加入你的程序。缺省的应用程序是一个空白的窗体(Form)，你可以运行它，结果得到一个空白的窗口。这个窗口具有 Windows 窗口的全部性质：可以被放大缩小、移动、最大最小化等，但你却没有编写一行程序。因此，可以说应用程序框架通过提供所有应用程序共有的东西，为用户应用程序的开发打下了良好的基础。Delphi 已经为你做好了一切基础工作——程序框架就是一个已经完成的可运行应用程序的窗口，只是不处理任何事情。你所需要做的，只是在程序中加入完成你所需功能的代码而已。

在空白窗口的背后，应用程序的框架正在等待用户的输入。由于你并未告诉它接收到用户输入后作何反应，窗口除了响应 Windows 的基本操作(移动、缩放等)外，它只是接受用户的输入，然后再忽略。Delphi 把 Windows 编程的回调、句柄处理等繁复过程都放在一个不可见的 Romulam 覆盖物下面，这样你可以不为它们所困扰，轻松从容地对可视控件进行编程。

面向对象的程序设计(Object-Oriented Programming，简记为 OOP)是 Delphi 诞生的基础。OOP 立意于创建软件重用代码，具备更好的模拟现实世界环境的能力，这使它被公认为是自上而下编程的优胜者。它通过给程序中加入扩展语句，把函数“封装”进 Windows 编程所必需的“对象”中。面向对象的编程语言使得复杂的工作条理清晰、编写容易。说它是一场革命，不是对对象本身而言，而是对它们处理工作的能力而言。对象并不与传统程序设计和编程方法兼容，只是部分面向对象反而会使情形更糟。除非整个开发环境都是面向对象的，否则对象产生的好处还没有带来的麻烦多。而 Delphi 是完全面向对象的，这就使得 Delphi 成为一种触手可及的促进软件重用的开发工具，从而具有强大的吸引力。

一些早期的具有 OOP 性能的程序语言如 C++，Pascal，Smalltalk 等，虽然具有面向对象的特征，但不能轻松地画出可视化对象；它们与用户交互能力较差，程序员仍然要编写大量的代码。Delphi 的推出，填补了这项空白。你不必自己建立对象，只要在提供的程序框架中加入完成功能的代码，其余的都交给 Delphi 去做。欲生成漂亮的界面和结构良好的程序丝毫不必绞尽脑汁，Delphi 将帮助你轻松地完成。它允许在一个具有真正 OOP 扩展的可视化编程环境中，使用它的 Object Pascal 语言。这种革命性的组合，使得可视化编程与面向对象的开发框架紧密地结合起来。

实例过程

☞ 建立一个新工程 WevCalcpro

在 File 菜单上选用 New Application 菜单项来生成新的工程文件，产生如图 1-2 所示的窗体界面。

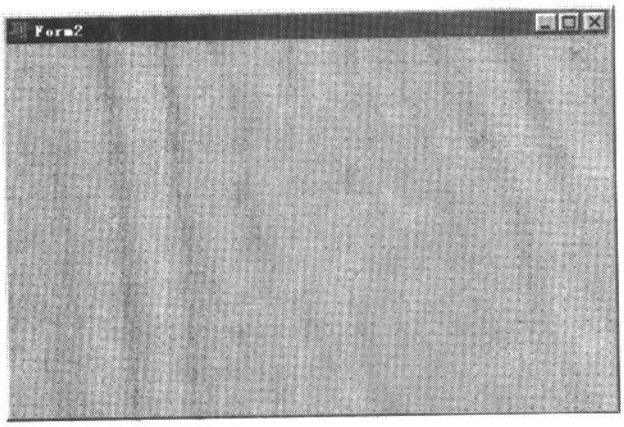


图 1-2 新建的窗体界面

然后，选取控件加入到窗体中。控件(Component)是建立 Delphi 应用程序的要素。Delphi 为用户提供了丰富的控件库，既有可视的控件(如编辑框、按钮)等，也有不可视的控件(如系统定时器、数据表等)。它们按照功能分别排列在 Component Panel 的各页上。移动鼠标到 Component Panel 上，在控件按钮上停留一两秒钟，一个黄色小提示框就会弹出，上面写有该控件的名称，我们称之为提示（Hint）。在要选择的控件上单击，则该控件按钮被按下，表示当前控件已被选中。然后，将鼠标移动到窗体上，按下左键，该控件被放到窗体中。控件的轮廓线上会显现 8 个被称为尺寸调整器（Sizing Handles）的黑色小方块。它除了供用户调整尺寸使用之外，还可以表示该对象处于当前编辑状态。此时，按【Delete】键可以将该控件删除。

按动“Standard”页标签，找到“Button”控件（图标上画有 OK 按钮）。Delphi 允许在添加多个同类型的控件时，不必每次到控件选项板上选取。按住【Shift】键，同时在“Button”控件上单击，这时“Button”控件处于按下状态，并有蓝色边框，表示已经被选择固定。依次在窗体右方的 3 个位置上单击，则会放置 3 个相同的“Button”控件。把鼠标光标移到控件选项板左侧的箭头图标处，这是“取消选择”按钮，它没有提示，按动它，会发现“Button”控件按钮恢复弹起状态。否则，每次“Form1”上的鼠标单击动作，都会导致增加一个按钮控件，一共要在表单上放置 18 个“Button” 控件。最后，利用对齐模板来对齐控件。首先将要对齐的控件选成一组，选择 Edit 菜单下的 Alignment Palette 显示对齐模板，然后按照所示的方式选择即可达到对齐的目的。

用同样的方法在“Standard”页中，找到“Edit”控件，加到窗体中。

如果控件已经对齐，为防止不小心移动控件，可以将控件位置锁定。选择 Edit 主菜单上的 Lock Controls 选项，使得控件不能进行移动操作。解锁只需再次选择此项即可。

及时地保存所做的工作至关重要。对设计者来讲，有两个文件需要保存：库单元文件（以.PAS 为后缀）和工程文件（以.DPR 为后缀）。

从 File 主菜单上选择 Save Project As...项，Delphi 会显示标题为“Save Unit1 As”的文件保存对话框，你可以在下拉式列表框中选择。最好将你的文件保存在自己的目录中。在编辑框中键入 WevCalc.pas 以保存库单元文件；然后显示标题为“Save Project As”的另一个文件保存对话框，键入 WevCalcpro.dpr。Delphi 保存这两个文件并返回当前窗口。不要把库单元和工程存成一样的文件名，Delphi 要求两者不同。

第一次保存后，以后可以随时通过 Speed Bar 中的“Save All”和“Save file”来保存工程文件和库单元文件。一般来讲，当确认文件的改变后，要同时存储这两个文件。

☞ 修改窗体及控件的属性

用 Properties 页改变控件的属性值。首先要改变各种控件的标题。先给窗口命名为“计算器”。按动 Object Inspector 上端的 Object Selector 的题条或者其右端的下拉标志，找到 Form1 项，并单击，窗体被选中。在 Object Inspector 的 Properties 页中，找到 Caption 属性并用左键选中，将其右端的“Form1 ”改为“计算器”。同时，你会发现窗体的标题已经相应地做了改变。

选中 Button1 按钮，此时 Object Inspector 已经显示出此按钮相应的属性。将它的 Caption 属性改为“1”，把 Name 属性改为 One。同理，修改其他 Button 的 Caption 和 Name 属性。选中 Edit1 控件，把它的 Text 属性改为空，把 Name 属性改为 Screen。

程序中加入的控件以及将要使用的函数如下:

```
TForm1 = class(TForm)
  Panel1: TPanel;
  Screen: TEdit;
  Clear_All: TButton;
  Clear: TButton;
  Divide: TButton;
  Multiply: TButton;
  Subtract: TButton;
  Add: TButton;
  Eight: TButton;
  Equal: TButton;
  Three: TButton;
  Six: TButton;
  Nine: TButton;
  Seven: TButton;
```

```
    Zero: TButton;
    Two: TButton;
    Five: TButton;
    Decimal: TButton;
    One: TButton;
    Four: TButton;
    procedure ClearEntry();
    procedure FormCreate(Sender: TObject);
    procedure Clear_AllClick(Sender: TObject);
    procedure OneClick(Sender: TObject);
    procedure SubtractClick(Sender: TObject);
    procedure AddClick(Sender: TObject);
    procedure MultiplyClick(Sender: TObject);
    procedure DivideClick(Sender: TObject);
    procedure ClearClick(Sender: TObject);
    procedure EqualClick(Sender: TObject);
    procedure DecimalClick(Sender: TObject);
    procedure TwoClick(Sender: TObject);
    procedure ThreeClick(Sender: TObject);
    procedure FourClick(Sender: TObject);
    procedure FiveClick(Sender: TObject);
    procedure SixClick(Sender: TObject);
    procedure SevenClick(Sender: TObject);
    procedure EightClick(Sender: TObject);
    procedure NineClick(Sender: TObject);
    procedure ZeroClick(Sender: TObject);
```

至此，界面的设计工作已完成了。运行一下并观察效果，别忘记保存你的库单元文件、工程文件。窗体修改后的结果如图 1-3 所示。

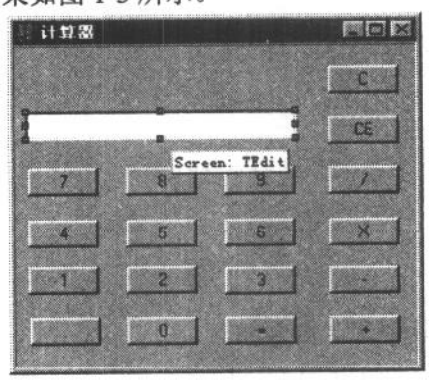


图 1-3 窗体修改的结果

☞ 编写事件处理过程

完成用户界面，只是建立了一个“骨架”，下面要做的便是给程序加入“灵魂”，也就是使它能够完成所要求的功能。

添加用户定义的常量：

```
const
  //定义用户常量
  OperatorNone = WM_USER+100;  //定义无任何操作
  OperatorAdd = WM_USER+101;   //定义加法操作
  OperatorSub = WM_USER+102;   //定义减法操作
  OperatorMul = WM_USER+103;   //定义乘法操作
  OperatorDiv = WM_USER+104;   //定义除法操作
```

对于运算按钮，必须把当前累加器的内容移到新的位置（称此新的暂时累加器为 nAccumulator），存储当前运算，并且设定一个标志，告诉程序在下一个功能键被按下时清除当前累加器。当等于“=”按钮被按下时，取出 nAccumulator 和 mAccumulator 的内容并且执行最后的运算键按下时存储的操作。

根据以上的思路，定义用户变量：

```
  public
    { Public declarations }
    mAccumulator:Double;          //暂时累加器
    nAccumulator:Double;          //当前累加器
    nDecimal:Integer;             //当前的小数点位置
    nPreviousOperation:Integer;//if True,清累加器
    nClear:Boolean;
```

对这几个变量赋初值：

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  mAccumulator:=0.0;
  nAccumulator:=0.0;
  nDecimal:=0;
  nPreviousOperation:=OperatorNone;
  nClear:=FALSE;
end;
```

并且加入以下代码以实现清除当前累加器：

```
procedure TForm1.ClearEntry();
begin
```

```
  nAccumulator:=0.0;
  nDecimal:=0;
  nClear:=FALSE;
end;
```

单击窗体上的“1”按钮，在 Object Inspector 中，单击“Events”页标签，出现事件窗口。在本例程中，我们只关心 OnClick 事件，即按钮接收到单击左键时应用程序所作出的反应。在 Object Inspector 窗口中双击 OnClick 事件右端的值段，会使得 Delphi 激活库代码编辑器，并将光标停在该按钮所对应的过程的 begin...end 之间。“1”按钮的功能响应按下按钮“1”后，计算器所采取的动作。

在光标处键入以下的程序段：

```
procedure TForm1.OneClick(Sender: TObject);
begin
  if nClear=TRUE then
    begin
      ClearEntry();   //清除当前累加器
    end;

//判断是否为实数
  if nDecimal >0 then    //如果 nDecimal >0 则为实数
    begin
      //加上当前按钮代表的值
      nAccumulator:=nAccumulator+1.0/nDecimal;
      nDecimal:=nDecimal*10;
    end
  else     //否则，为整数
    begin
      //加上当前按钮代表的值
      nAccumulator:=nAccumulator*10;
      nAccumulator:=nAccumulator+1.0;
    end;

//向 TEdit 窗体输出结果
  Screen.Text:=CurrToStr(nAccumulator);
end;
```

同理，我们加入对其他数字按钮的 OnClick 响应事件：

```
//按钮“2”的 OnClick 响应事件
procedure TForm1.TwoClick(Sender: TObject);
```

```
begin
  if nClear=TRUE then
    begin
      ClearEntry();
    end;

   if nDecimal>0 then
    begin
      nAccumulator:=nAccumulator+2.0/nDecimal;
      nDecimal:=nDecimal*10;
    end
  else
    begin
      nAccumulator:=nAccumulator*10;
      nAccumulator:=nAccumulator+2.0;
    end;

  Screen.Text:=CurrToStr(nAccumulator);
end;

//按钮“3”的OnClick响应事件
procedure TForm1.ThreeClick(Sender: TObject);
begin
  if nClear=TRUE then
    begin
      ClearEntry();
    end;

  if nDecimal >0 then
    begin
      nAccumulator:=nAccumulator+3.0/nDecimal;
      nDecimal:=nDecimal*10;
    end
  else
    begin
      nAccumulator:=nAccumulator*10;
      nAccumulator:=nAccumulator+3.0;
    end;
```

```
  Screen.Text:=CurrToStr(nAccumulator);
end;

//按钮“4”的OnClick响应事件
procedure TForm1.FourClick(Sender: TObject);
begin
  if nClear=TRUE then
    begin
      ClearEntry();
    end;

  if nDecimal >0 then
    begin
      nAccumulator:=nAccumulator+4.0/nDecimal;
      nDecimal:=nDecimal*10;
    end
  else
    begin
      nAccumulator:=nAccumulator*10;
      nAccumulator:=nAccumulator+4.0;
    end;

  Screen.Text:=CurrToStr(nAccumulator);
end;

//按钮“5”的OnClick响应事件
procedure TForm1.FiveClick(Sender: TObject);
begin
  if nClear=TRUE then
    begin
      ClearEntry();
    end;

  if nDecimal >0 then
    begin
      nAccumulator:=nAccumulator+5.0/nDecimal;
      nDecimal:=nDecimal*10;
    end
  else
```

```
    begin
      nAccumulator:=nAccumulator*10;
      nAccumulator:=nAccumulator+5.0;
    end;

  Screen.Text:=CurrToStr(nAccumulator);
end;

//按钮“6”的 OnClick 响应事件
procedure TForm1.SixClick(Sender: TObject);
begin
  if nClear=TRUE then
    begin
      ClearEntry();
    end;

  if nDecimal >0 then
    begin
      nAccumulator:=nAccumulator+6.0/nDecimal;
      nDecimal:=nDecimal*10;
    end
  else
    begin
      nAccumulator:=nAccumulator*10;
      nAccumulator:=nAccumulator+6.0;
    end;

  Screen.Text:=CurrToStr(nAccumulator);
end;

//按钮“7”的 OnClick 响应事件
procedure TForm1.SevenClick(Sender: TObject);
begin
  if nClear=TRUE then
    begin
      ClearEntry();
    end;

  if nDecimal >0 then
```

```
    begin
      nAccumulator:=nAccumulator+7.0/nDecimal;
      nDecimal:=nDecimal*10;
    end
  else
    begin
      nAccumulator:=nAccumulator*10;
      nAccumulator:=nAccumulator+7.0;
    end;

  Screen.Text:=CurrToStr(nAccumulator);
end;

//按钮“8”的OnClick响应事件
procedure TForm1.EightClick(Sender: TObject);
begin
  if nClear=TRUE then
    begin
      ClearEntry();
    end;

  if nDecimal >0 then
    begin
      nAccumulator:=nAccumulator+8.0/nDecimal;
      nDecimal:=nDecimal*10;
    end
  else
    begin
      nAccumulator:=nAccumulator*10;
      nAccumulator:=nAccumulator+8.0;
    end;

  Screen.Text:=CurrToStr(nAccumulator);
end;

//按钮“9”的OnClick响应事件
procedure TForm1.NineClick(Sender: TObject);
begin
  if nClear=TRUE then
```

```
      begin
        ClearEntry();
      end;

    if nDecimal >0 then
      begin
        nAccumulator:=nAccumulator+9.0/nDecimal;
        nDecimal:=nDecimal*10;
      end
    else
      begin
        nAccumulator:=nAccumulator*10;
        nAccumulator:=nAccumulator+9.0;
      end;

   Screen.Text:=CurrToStr(nAccumulator);
 end;

 //按钮“0”的OnClick响应事件
 procedure TForm1.ZeroClick(Sender: TObject);
 begin
   if nClear=TRUE then
     begin
       ClearEntry();
     end;

   if nDecimal >0 then
     begin
       nAccumulator:=nAccumulator+0.0/nDecimal;
       nDecimal:=nDecimal*10;
     end
   else
     begin
       nAccumulator:=nAccumulator*10;
     end;

   Screen.Text:=CurrToStr(nAccumulator);
 end;
```

下面，编写处理运算符加号“+”的事件响应程序的代码：

```
procedure TForm1.AddClick(Sender: TObject);
begin
    //判断前一次按下的运算符
  case nPreviousOperation of
    OperatorAdd:          //前一次运算符为“+”
      begin
        //当前累加器和暂时累加器的结果相加
        nAccumulator:=StrToCurr(Screen.Text);
        nAccumulator:=mAccumulator+nAccumulator;
      end;

    OperatorSub:          //前一次运算符为“-”
      Begin
        //当前累加器和暂时累加器的结果相减
        nAccumulator:=StrToCurr(Screen.Text);
        nAccumulator:=mAccumulator-nAccumulator;
      end;
    OperatorMul:          //前一次运算符为“*”
      Begin
        //当前累加器和暂时累加器的结果相乘
        nAccumulator:=StrToCurr(Screen.Text);
        nAccumulator:=mAccumulator*nAccumulator;
      end;
     OperatorDiv:         //前一次运算符为“/”
       Begin
        //当前累加器和暂时累加器的结果相除
        nAccumulator:=StrToCurr(Screen.Text);
        nAccumulator:=mAccumulator/nAccumulator;
       end;
     OperatorNone:
       Begin              //前一次运算符为空
         //不做运算操作
         nAccumulator:=StrToCurr(Screen.Text);
       end;
  end;

 //记录这次按下的运算符
  nPreviousOperation:=OperatorAdd;
```