2011年计算机等级考试二级Delphi辅导讲义：剪贴板和动态数据交换 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/647/2021_2022_2011_E5_B9_B4_E8_AE_A1_c97_647331.htm 导读：应用程序间的数据交换是象Windows 这样的多任务环境的重要特性。>>>>点击查看此系列辅导讲义汇总第七章 剪贴板和动态数据交换 应用程序间的数据交换是象Windows 这样的多任务环境的重要特性。作为一种基于Windows的开发工具，Delphi支持如下四种数据交换方式：剪贴板、动态数据交换 (DDE)、对象联接与嵌入(OLE)以及动态联接库(DLLs)。这中间前三种方式最为常用，OLE功能最为强大，DDE次之。而剪贴板使用最为方便。在本章，我们只讨论剪贴板和动态数据交换。利用OLE实现数据交换见下一章，利用动态联接库(DLLs)进行数据交换将在第十章中介绍。 7.1 剪贴板及其应用 本质上，剪贴板只是一个全局内存块。当一个应用程序将数据传送给剪贴板后，通过修改内存块分配标志，把相关内存块的所有权从应用程序移交给Windows自身。其它应用程序可以通过一个句柄找到这个内存块，从而能够从内存块中读取数据。这样就实现了数据在不同应用程序间的传输。 剪贴板虽然功能较为简单，且不能实现实时传输，但却是更为复杂的DDE和OLE的基础。对于一些只是偶尔需要使用其它应用程序数据的程序来说，使用剪贴板不失为一种方便、快捷的方式。 Delphi把剪贴板的大部分功能封装到一个TClipboard类中，同时把使用频度最高的文本传输功能(包括DBImage的图像传输功能)置入相应部件作为部件的方法，从而使用户可以十分方便地使用剪贴板进行编程。 7.1.1 使用剪贴板传输文本 剪贴板传输文本主

要是应用如下的三个方法:CopyToClipboard、CutToClipboard和PasteFromClipboard。包含这些方法的部件如下表所示。

表7.1 包含剪贴板方法的部件

方法	部件
CopyToClipboard	TDBEdit TDBMemo TDBImage TEdit TMemo TMaskEdit TOLEContainer TDDEServerItem
CutToClipboard	TDBEdit TDBMemo TDBImage TEdit TMemo TMaskEdit
PasteFromClipboard	TDBEdit TDBMemo TDBImage TEdit TMemo TMaskEdit

除TDBImage外，其余全是有关文本的控件。 在把文本传输到剪贴板之前，文本必须被选中。 若选TMaskEdit的AutoSelect属性为True，则当MaskEdit获得输入焦点时文本自动被选中；若选TEdit、TMemo的HideSelection属性为True，则失去焦点时，文本选中状态自动隐藏，重新获得焦点时再显示。 下面的语句把MaskEdit中选中的文本剪切到剪贴板：MaskEdit .CutToClipboard. 下面的语句把剪贴板中的文本粘贴到Memo的当前光标处：Memo.PasteFromClipboard；利用剪贴板类也可以实现文本的传输，见(7.1.2)中的介绍。 7.1.2 剪贴板类 为方便剪贴板的操作，Delphi在Clipbrd库单元中定义了一个TClipboard类，并且预定义了一个变量Clipboard作为类TClipboard的实例，从而使用户在绝大多数场合不必自己去定义一个TClipboard的实例。 利用剪贴板类可以进行文本、图像和部件的传输，剪贴板类为实现这些方法提供了相应的属性和方法。表7.2、表7.3列出了TClipboard属性和方法的意义。 表 7.2 TClipboard的属性

属性	意义

AsText 保存剪贴板的文本，只有运行时才可设置 FormatCount 可用剪贴板格式的数目 Formats 可用剪贴板格式链

表 7.3 TClipboard的方法

方法	参数	意义
Clear	无	清除剪贴板的内容
Assign	Source:TPersistent	把Source参数指定的对象拷贝到剪贴板，常 用于图形、图像对象
Open	无	打开剪贴板，阻止其它应用程序改变它的内容
Close	无	关闭打开的剪贴板
SetComponent	Source:TPersistent	把部件拷贝到剪贴板
GetComponent	Owner	从剪贴板取回一个部件并放置
Parent	:TPersistent	
SetAsHandle	Format:Word	把指定格式数据的句柄交给剪贴板
返回类型：THandle		
GetAsHandle	Format:Word	返回剪贴板指定格式数据的句柄
返回类型：THandle		
HasFormat	Format:Word	判断剪贴板是否拥有给定的格式
返回类型：Boolean		
SetTextBuf	Buffer:PChar	设置剪贴板的文本内容

剪贴板中可能的数据格式如下表。

表 7.4 剪贴板数据格式及其意义

数据格式	意义
CF_TEXT	文本。每行以CF_LF结束，nil标志文本结束
CF_BITMAP	Windows位图
CF_METAFILE	Windows元文件
CF_PICTURE	TPicture类型的对象
CF_OBJECT	任何TPersistent类型的对象

利用TClipboard实现文本的传输使用AsText属性和SetTextBuf方法。 AsText属性为非控件部件的剪贴板操作提供了方便。如： Clipboard. AsText := Form1.Caption . 把Form1的标题拷贝到剪贴板。 Label1.Caption := Clipboard.AsText. 把剪贴板中的文本写入Label1。 SetTextBuf用于把超过255个字符的字符串拷入剪贴板。 7.1.3 利用剪贴板传输图像 7.1.3.1 拷贝 Image部件上的内容和窗体上的图形可以直接拷贝到剪贴板。图像拷贝利用Clipboard的Assign方法。例如： Clipboard.Assign(Image1.Picture). 把Image1上的图像拷贝到剪贴板。 7.1.3.2 剪切 图像的剪切是首先把图像拷贝到剪贴板，而后在原位置用空白图像进行覆盖。 下面一段程序表示了图像的剪切。 procedure TForm1.Cut1Click(Sender: TObject). var ARect: TRect. begin Clipboard.Assign(Image1.Picture). with Image.Canvas do begin CopyMode := cmWhiteness. ARect := Rect(0, 0, Image.Width, Image.Height). CopyRect(ARect, Image.Canvas, ARect). CopyMode := cmSrcCopy. end. end. 7.1.3.3 粘贴 从剪贴板上粘贴图像，首先检测剪贴板上的数据格式。如果格式为CF_BITMAP，则调用目标位图的Assign 方法粘贴图像。 程序清单如下。 procedure TForm1.PasteButtonClick(Sender: TObject). var Bitmap: TBitmap. begin if Clipboard.HasFormat(CF_BITMAP) then begin Bitmap := TBitmap.Create. try Bitmap.Assign(Clipboard). Image.Canvas.Draw(0, 0, Bitmap). finally Bitmap.Free. end. end. end. try...finally为资源保护块，参第十二章。 7.1.4 建立自己的剪贴板观察程序 在这一节中我们要建立一个自己的剪贴板观

察程序，用来保存截获到剪贴板中的位图。 Windows允许用户建立自己的剪贴板观察程序，并把该程序添加到一个剪贴板观察器链中。在链中，位置靠前的程序有义务把有关剪贴板的消息传递到紧随其后的观察程序。而处于链首的程序由Windows的消息循环机制直接把剪贴板消息发送过来。 建立一个剪贴板观察程序，首先该程序必须能响应相应的Windows消息。对于那些熟悉Microsoft公司Visual Basic的读者来说，这是令他们头疼而束手无策的地方。但Delphi在这方面却有良好的表现：利用关键字message，用户可以将一个过程定义为响应特定的Windows消息。如： procedure WMDrawClipboard(var Msg:TWMDrawClipboard). message WM_DRAWCLIPBOARD. 可以响应WM_DRAWCLIPBOARD消息。类TWMDrawClipboard是消息类Message 的子类。Delphi把所有的消息都重新进行了定义，使用户在使用时可以直接引用其便于记忆的数据成员，而不必再自己动手去分解消息。虽然这并不能算作是一个重大的改进，但却体现了Delphi处处为用户方便着想的特点。 我们将要建立的程序目的是把截获到剪贴板上的位图保存下来。在本书的写作过程中，这一工作是大量存在的。虽然利用Windows工具PaintBrush(画笔），通过粘贴、保存等操作可以实现这一功能，但却存在以下一些问题： 1.程序频繁切换影响效率，当有大量位图存在时更是如此； 2.画笔有一个很讨厌的缺陷：当剪贴板上的位图比画笔界面的客户区大时，客户区外的位图被截断。因而往往需要根据所截获位图的大小来调整画笔客户区的大小，并重新进行粘贴。而如果开始就把画笔客户区调整到足够大，又会覆盖掉屏幕上一些有用的信息。 为

解决这些问题，我开发了下面的程序。程序启动时，以极小化方式运行。此时只要剪贴板中存入位图，则自动弹出一个对话框请求用户保存。如果用户希望查看确认，则可以双击运行程序图标，选择相应按钮，剪贴板中的位图就会显示在屏幕上。 部件关键属性设计如下： ClipSaveForm： Caption＝‘ Save Bitmap in Clipboard Panel1: Align = Top Image1: Align = Client SaveDialog1: FileEditStyle = fsEdit FileName = *.bmp Filter = Bitmap Files(*.bmp)|*.bmp|Any Files(*.*)|*.* InitialDir = c:\bmp Title = Save Bitmap 程序主窗口是TForm派生类TClipSaveForm的实例。TClipSaveForm通过定义一些私有数据成员和过程，使响应和处理Windows的相应消息成为可能。下面是TClipSaveForm的类定义： type TClipSaveForm = class(TForm) SaveDialog1: TSaveDialog. Image1: TImage. Panel1: TPanel. Button1: TButton. SpeedButton1: TSpeedButton. SpeedButton2: TSpeedButton. Button2: TButton. procedure FormCreate(Sender: TObject). procedure FormDestroy(Sender: TObject). procedure Button1Click(Sender: TObject). procedure Button2Click(Sender: TObject). procedure SpeedButton1Click(Sender: TObject). procedure SpeedButton2Click(Sender: TObject). private { Private declarations } MyBitmap: TBitmap. { 保存截获的位图 } View: Boolean. { 判断是否显示 } NextViewerHandle: HWND. { 下一剪贴板观察器的句柄 } procedure WMDrawClipboard(var Msg:TWMDrawClipboard). message WM_DRAWCLIPBOARD. procedure WMChangeCBChain(var Msg:TWMChangeCBChain). message WM_CHANGECBCHAIN. { 响应Windows的剪贴板消

息 } public { Public declarations } end. 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com