2011年计算机等级考试二级Delphi辅导讲义：开发Delphi对象式数据管理功能 PDF转换可能丢失图片或格式，建议阅读原文

https://www.100test.com/kao_ti2020/646/2021_2022_2011_E5_B9_B4_E8_AE_A1_c97_646914.htm 导读：在本章中将介绍Stream对象和Filer对象的实现原理、应用方法以及在超媒体系统中的应用。这对于运用Delphi 开发高级应用是很重要的。>>>>点击查看此系列辅导讲义汇总第二十章 开发Delphi对象式数据管理功能 面向对象技术是九十年代的主流技术，各类应用软件如果以面向对象的方法构造并且渗透面向对象的风格将使软件具有更高的品质。在面向对象程序设计中，对象式数据管理占有很重要的地位。在Delphi中，对对象式数据管理的支持方式是其一大特色。 Delphi是一个面向对象的可视化设计与面向对象的语言相结合的集成开发环境。Delphi的核心是部件。部件是对象的一种。Delphi应用程序完全是由部件来构造的，因此开发高性能的Delphi应用程序必然会涉及对象式数据管理技术。 对象式数据管理包括两方面的内容：● 用对象来管理数据 ● 对各类数据对象（包括对象和部件）的管理 Delphi在这两方面都做的相当出色。在Delphi的早期版本Turbo Pascal 中就曾有流(Stream)、群(Collection)和资源(Resource)等专门用于对象式数据管理的类。在Delphi中，这些功能得到了大大的加强。Delphi将对象式数据管理类归结为Stream对象(Stream)和Filer对象(Filer)，并将它们应用于可视部件类库（VCL）的方方面面。它们不仅提供了在内存、外存和Windows资源中管理对象的功能，还提供了在数据库BLOB字段中对象的功能。 在本章中将介绍Stream对象

和Filer对象的实现原理、应用方法以及在超媒体系统中的应用。这对于运用Delphi 开发高级应用是很重要的。 20.1 流式对象的实现原理和应用 Stream对象，又称流式对象，是TStream、THandleStream、TFileStream、TMemoryStream、TResourceStream和TBlobStream等的统称。它们分别代表了在各种媒介上存储数据的能力，它们将各种数据类型(包括对象和部件) 在内存、外存和数据库字段中的管理操作抽象为对象方法，并且充分利用了面向对象技术的优点，应用程序可以相当容易地在各种Stream对象中拷贝数据。 下面介绍各种对象的数据和方法及使用方法。 20.1.1 TStream对象 TStream对象是能在各种媒介中存储二进制数据的对象的抽象对象。从TStream 对象继承的对象用于在内存、Windows资源文件、磁盘文件和数据库字段等媒介中存储数据。 TStream中定义了两个属性：Size和Position。它们分别以字节为单位表示的流的大小和当前指针位置。TStream中定义的方法用于在各种流中读、写和相互拷贝二进制数据。因为所有的Stream对象都是从TStream中继承来的，所以在TStream中定义的域和方法都能被Stream对象调用和访问。此外，又由于面向对象技术的动态联编功能，TStream为各种流的应用提供了统一的接口，简化了流的使用；不同Stream对象是抽象了对不同存储媒介的数据上的操作，因此，TStream的需方法为在不同媒介间的数据拷贝提供了最简捷的手段。 20.1.1.1 TStream的属性和方法 1. Position属性 声明：property Position: Longint. Position属性指明流中读写的当前偏移量。 2. Size属性 声明：property Size: Longint. Size属性指明了以字节为单位的流的的大小，它是只读的。 3. CopyFrom方法 声明：function CopyFrom(Source:

TStream. Count: Longint): Longint. CopyFrom从Source所指定的流中拷贝Count个字节到当前流中， 并将指针从当前位置移动Count个字节数，函数返回值是实际拷贝的字节数。 4. Read方法 声明：function Read(var Buffer. Count: Longint): Longint. virtual. abstract. Read方法从当前流中的当前位置起将Count个字节的内容复制到Buffer中，并把当前指针向后移动Count个字节数，函数返回值是实际读的字节数。如果返回值小于Count，这意味着读操作在读满所需字节数前指针已经到达了流的尾部。 Read方法是抽象方法。每个后继Stream对象都要根据自己特有的有关特定存储媒介的读操作覆盖该方法。而且流的所有其它的读数据的方法（如：ReadBuffer，ReadComponent等）在完成实际的读操作时都调用了Read方法。面向对象的动态联编的优点就体现在这儿。因为后继Stream对象只需覆盖Read方法，而其它读操作(如ReadBuffer、ReadComponent等)都不需要重新定义，而且TStream还提供了统一的接口。 5. ReadBuffer方法 声明：procedure ReadBuffer(var Buffer. Count: Longint). ReadBuffer方法从流中将Count个字节复制到Buffer 中， 并将流的当前指针向后移动Count个字节。如读操作超过流的尾部，ReadBuffer方法引起EReadError异常事件。 6. ReadComponent方法 声明：function ReadComponent(Instance: TComponent): TComponent. ReadComponent方法从当前流中读取由Instance所指定的部件，函数返回所读的部件。ReadComponent在读Instance及其拥有的所有对象时创建了一个Reader对象并调用它的ReadRootComponent方法。 如果Instance为nil，ReadComponent的方法基于流中描述的部件类型信息创建

部件，并返回新创建的部件。 7. ReadComponentRes方法 声明：function ReadComponentRes(Instance: TComponent): TComponent. ReadComponentRes方法从流中读取Instance指定的部件，但是流的当前位置必须是由WriteComponentRes方法所写入的部件的位置。 ReadComponentRes 首先调用ReadResHeader方法从流中读取资源头，然后调用ReadComponent方法读取Instance。如果流的当前位置不包含一个资源头。ReadResHeader将引发一个EInvalidImage异常事件。在Classes库单元中也包含一个名为ReadComponentRes的函数，该函数执行相同的操作，只不过它基于应用程序包含的资源建立自己的流。 8. ReadResHeader方法 声明：procedure ReadResHeader. ReadResHeader方法从流的当前位置读取Windows资源文件头，并将流的当前位置指针移到该文件头的尾部。如果流不包含一个有效的资源文件头，ReadResHeader将引发一个EInvalidImage异常事件。 流的ReadComponentRes方法在从资源文件中读取部件之前，会自动调用ReadResHeader方法，因此，通常程序员通常不需要自己调用它。 9. Seek方法 声明：function Seek(Offset: Longint. Origin: Word): Longint. virtual. abstract. Seek方法将流的当前指针移动Offset个字节，字节移动的起点由Origin指定。如果Offset是负数，Seek方法将从所描述的起点往流的头部移动。下表中列出了Origin的不同取值和它们的含义： 表20.1 函数Seek的参数的取值 ——————————————————————————— 常量 值 Seek的起点 Offset的取值 ——————————————————————————— SoFromBeginning 0 流的开头 正 数

SoFromCurrent 1 流的当前位置 正数或负数 SoFromEnd 2 流的结尾 负 数

10. Write方法 在Delphi对象式管理的对象中有两类对象的方法都有称为Write的：Stream对象和Filer对象。Stream对象的Write方法将数据写进流中。Filer对象通过相关的流传递数据，在后文中会介绍这类方法。Stream对象的Write方法声明如下： function Write(const Buffer. Count: Longint): Longint. virtual. abstract. Write方法将Buffer中的Count个字节写入流中，并将当前位置指针向流的尾部移动Count个字节，函数返回写入的字节数。 TStream的Write方法是抽象的，每个继承的Stream对象都要通过覆盖该方法来提供向特定存储媒介(内存、磁盘文件等)写数据的特定方法。流的其它所有写数据的方法(如WriteBuffer、WriteComponent)都调用Write担当实际的写操作。 11. WriteBuffer方法 声明：procedure WriteBuffer(const Buffer. Count: Longint). WriteBuffer的功能与Write相似。WriteBuffer方法调用Write来执行实际的写操作，如果流没能写所有字节，WriteBuffer会触发一个EWriteError异常事件。 12. WriteComponent方法 在Stream对象和Filer对象都有被称为WriteComponent的方法。Stream对象的WriteComponent方法将Instance所指定的部件和它所包含的所有部件都写入流中；Writer对象的WriteComponent将指定部件的属性值写入Writer对象的流中。 Stream对象的WriteComponent方法声明是这样的： procedure WriteComponent(Instance: Tcomponent). WriteComponent创建一个Writer对象，并调用Writer的WriteRootComponent方法将Instance及其拥有的对象写入流

。 13. WriteComponentRes方法 声明：WriteComponentRes(const ResName: String. Instance: TComponent). WriteComponentRes方法首先往流中写入标准Windows 资源文件头，然后将Instance指定的部件写入流中。要读由WriteComponentRes写入的部件，必须调用ReadComponentRes方法。 WriteComponentRes使用ResName传入的字符串作为资源文件头的资源名，然后调用WriteComponent方法将Instance和它拥有的部件写入流。 14. WriteDescendant方法 声明：procedure WriteDescendant(Instance Ancestor: TComponent). Stream对象的WriteDescendant方法创建一个Writer对象，然后调入该对象的WriteDescendant方法将Instance部件写入流中。Instance可以是从Ancestor部件继承的窗体，也可以是在从祖先窗体中继承的窗体中相应于祖先窗体中Ancestor部件的部件。 15. WriteDescendantRes方法 声明：procedure WriteDescendantRes(const ResName: String. Instance, Ancestor: TComponent). WriteDescendantRes方法将Windows资源文件头写入流，并使用ResName作用资源名，然后调用WriteDescendant方法，将Instance写入流。 20.1.1.2 TStream的实现原理 TStream对象是Stream对象的基础类，这是Stream对象的基础。为了能在不同媒介上的存储数据对象，后继的Stream对象主要是在Read和Write方法上做了改进，。因此，了解TStream是掌握Stream对象管理的核心。Borland公司虽然提供了Stream对象的接口说明文档，但对于其实现和应用方法却没有提及，笔者是从Borland Delphi 2.0 Client/Server Suite 提供的源代码和部分例子程序中掌握了流式对象技术。

下面就从TStream的属性和方法的实现开始。 1. TStream属性的实现 前面介绍过，TStream具有Position和Size两个属性，作为抽象数据类型，它抽象了在各种存储媒介中读写数据所需要经常访问的域。那么它们是怎样实现的呢？ 在自定义部件编写这一章中介绍过部件属性定义中的读写控制。Position和Size也作了读写控制。定义如下： property Position: Longint read GetPosition write SetPosition. property Size: Longint read GetSize. 由上可知，Position是可读写属性，而Size是只读的。Position属性的实现就体现在GetPosition和SetPosition。当在程序运行过程中，任何读取Position的值和给Position赋值的操作都会自动触发私有方法GetPosition和SetPosition。两个方法的声明如下： function TStream.GetPosition: Longint. begin Result := Seek(0, 1). end. procedure TStream.SetPosition(Pos: Longint). begin Seek(Pos, 0). end. 在设置位置时，Delphi编译机制会自动将Position传为Pos。 前面介绍过Seek的使用方法，第一参数是移动偏移量，第二个参数是移动的起点，返回值是移动后的指针位置。 Size属性的实现只有读控制，完全屏蔽了写操作。读控制方法GetSize实现如下： function TStream.GetSize: Longint. var Pos: Longint. begin Pos := Seek(0, 1). Result := Seek(0, 2). Seek(Pos, 0). end. 2. TStream方法的实现 (1) CopyFrom方法 CopyFrom是Stream对象中很有用的方法，它用于在不同存储媒介中拷贝数据。例如，内存与外部文件之间、内存与数据库字段之间等。它简化了许多内存分配、文件打开和读写等的细节，将所有拷贝操作都统一到Stream对象上。 前面曾介绍：CopyFrom方法带Source和Count两个参数并返回长整型。该方法将Count个字节的内容从Source拷贝到当前流中，如

果Count值为0则拷贝所有数据。 function TStream.CopyFrom(Source: TStream. Count: Longint): Longint. const MaxBufSize = $F000. var BufSize, N: Integer. Buffer: PChar. begin if Count = 0 then begin Source.Position := 0.