 在使用DELPHI开发软件的过程中，我们就像草原上一群快乐牛羊，无忧无虑地享受着Object Pascal语言为我们带来的阳光和各种VCL控件提供的丰富的水草。抬头望望无边无际蔚蓝的天空，低头品尝大地上茂密的青草，谁会去想宇宙有多大，比分子和原子更小的东西是什么？那是哲学家的事。而哲学家此时正坐在高高的山顶上，仰望宇宙星云变换，凝视地上小虫的爬行，蓦然回头，对我们这群吃草的牛羊点头微笑。随手扯起一根小草，轻轻地含在嘴里，闭上眼睛细细品尝，不知道这根青草在哲学家的嘴里是什么味道？只是，他的脸上一直带着满意的微笑。 认识和了解DELPHI微观的原子世界，可以使我们彻底理解DELPHI的宏观应用程序结构，从而在更广阔的思想空间中开发我们的软件。这就好像，牛顿发现了宏观物体的运动，却因为搞不清物体为什么会这样运动而苦恼，相反，爱因斯坦却在基本粒子规律和宏观物体运动之间体验着相对论的快乐生活！ 第一节 TObject原子 TObject是什么？ 是Object Pascal语言体系结构的基本核心，也是各种VCL控件的起源。我们可以认为，TObject是构成DELPHI应用程序的原子之一，当然，他们又是由基本Pascal语法元素等更细微的粒子构成。 说TObject是DELPHI程序的原子，是因为TObject是DELPHI编译器内部支持的。所有的对象类都是从TObject派生的，即使你并未指定TObject为祖先类。TObject被定义在System单元，它是系统的一部分。在System.pas单元的开头，有这样的注释文

本：{ Predefined constants, types, procedures, } { and functions (such as True, Integer, or } { Writeln) do not have actual declarations.} { Instead they are built into the compiler } { and are treated as if they were declared } { at the beginning of the System unit. } 它的意思说，这一单元包含预定义的常量、类型、过程和函数（诸如：Ture、Integer或Writeln），它们并没有实际的声明，而是编译器内置的，并在编译的开始就被认为是已经声明的定义。你可以将Classes.pas或Windows.pas等其他源程序文件加入你的项目文件中进行编译和调试其源代码，但你绝对无法将System.pas源程序文件加入到你的项目文件中进行编译！DELPHI将报告重复定义System的编译错误！ 因此，TObject是编译器内部提供的定义，对于我们使用DELPHI开发程序的人来说，TObject是原子性的东西。 TObject在System单元中的定义是这样的： TObject = class constructor Create. procedure Free. class function InitInstance(Instance: Pointer): TObject. procedure CleanupInstance. function ClassType: TClass. class function ClassName: ShortString. class function ClassNameIs(const Name: string): Boolean. class function ClassParent: TClass. class function ClassInfo: Pointer. class function InstanceSize: Longint. class function InheritsFrom(AClass: TClass): Boolean. class function MethodAddress(const Name: ShortString): Pointer. class function MethodName(Address: Pointer): ShortString. function FieldAddress(const Name: ShortString): Pointer. function GetInterface(const IID: TGUID. out Obj): Boolean. class function GetInterfaceEntry(const IID: TGUID): PInterfaceEntry. class function GetInterfaceTable: PInterfaceTable. function

SafeCallException(ExceptObject: TObject. ExceptAddr: Pointer): HResult. virtual. procedure AfterConstruction. virtual. procedure BeforeDestruction. virtual. procedure Dispatch(var Message). virtual. procedure DefaultHandler(var Message). virtual. class function NewInstance: TObject. virtual. procedure FreeInstance. virtual. destructor Destroy. virtual. end. 下面，我们将逐步敲开TObject原子的大门，看看里面到底是什么结构。 我们知道，TObject是所有对象的基本类，那么，一个对象到底是什么？ DELPHI 中的任何对象都是一个指针，这个指针指明该对象在内存中所占据的一块空间！虽然，对象是一个指针，可是我们引用对象的成员时却不用写成这样的代码MyObject^.GetName，而只能写成MyObject.GetName，这是Object Pascal语言扩充的语法，是由编译器支持的。使用C Builder的朋友就很清楚对象与指针的关系，因为在C Builder的对象都要定义为指针。对象指针指向的地方就是对象存储数据的对象空间，我们来分析一下对象指针指向的内存空间的数据结构。 对象空间的头4个字节是指向该对象类的虚方法地址表（VMT ?C Vritual Method Table）。接下来的空间就是存储对象本身成员数据的空间，并按从该对象最原始祖先类的数据成员到该对象类的数据成员的总顺序，和每一级类中数据成员的定义顺序存储。 类的虚方法地址表（VMT）保存从该类的原始祖先类派生到该类的所有类的虚方法的过程地址。类的虚方法，就是用保留字vritual声明的方法，虚方法是实现对象多态性的基本机制。虽然，用保留字dynamic声明的动态方法也可实现对象的多态性，但这样的方法不保存在虚方法地址表（VMT）中，它只是Object Pascal提供的另一种可

节约类存储空间的多态实现机制，但却是以牺牲调用速度为代价的。