1-1 什么是 Windows SDK程序？
答：使用软件开发工具包开发出来的 Windows 应用程序叫做 Windows SDK程序。
1-3 打开 windows.h 文件，看一下 Windows 系统的句柄是什么数据类型的？ 答：整型。
1-5 什么是事件？
答：能触发程序做出相应反应的因素或动作叫做“事件”。
1-6 如何显示和更新窗口？
答：调用函数 ShowWindow显示窗口，调用函数 UpdateWindows 更新窗口。
1-7 什么是消息循环？
答：在创建了窗口的应用程序中， 应用程序将不断地从消息队列中获取消息， 并将消息指派给指定的窗口处理函数来处理， 然后再回来从消息队列获取消息，这个不断重复的工作过程叫做消息循环。
1-9 说明 Windows 应用程序的主函数、 窗口函数与 Windows 系统之间的关系。
答：Windows 应用程序的主函数和窗口函数都是系统调用的函数， 主函数是在应用程序启动时由系统首先调用的函数， 而窗口函数是主函数在消息循环中获得消息并把消息派送给系统之后， 由系统调用的用来处理消息的函数。
2-1 在窗体类 CFrameWnd中需要封装哪些成员？
答：在窗体类 CFrameWnd中要封装 窗口句柄 、窗口类的定义 、注册窗口类 、创建窗口 、显示更新窗口 。
2-2 应用程序类 CwinApp 应该具备那些主要功能？
答：创建、显示应用程序的窗口和建立消息循环。
2-3 在 MFC程序设计中，如果要建立拥有自己风格的主窗口，应该重写什么函数。
答：继承 CWinAPP类并需要重写该类的成员函数 InitInstance 。
3-3 简述构成文档 / 视图结构应用程序框架的四个 MFC派生类，并说出它们的功能。
答：假如工程名称为 MyPrj 则 MFCAppWizard 会自动创建一下四个派生类来构成应用程序的框架。
.CFrame 类的派生类 CMainFrame；
.CWinApp 类的派生类 CMyPrjApp；
.CDocument 类的派生类 CMyPrjDoc；
.CView 类的派生类 CMyPrjView。
其中，CMyPrjDoc 类对象用来存储和管理应用程序中的数据；CMainFrame 对象与 CMyPrjView 对象构成了应用程序的界面，CMainFrame对象只是 CMyPrjView 对象的容器，而 CMyPrjView 类的对象是用来显示文档与接收用户事件的； CMyPrjApp 类的对象是应用程序的全局对象， 它是应用程序中各对象的容器， 负责创建应用程序界面和消息循环。
3-4 在文档 / 视图结构的应用程序中，视图类对象是如何获取文档类对象中数据的？
答：是依靠视图类的成员函数 GetDocument 来返回文档对象指针， 然后再通过该指针访问文档类的数据成员或函数成员。
3-5 在 MFC对程序窗口功能的划分中你受到了什么启发？
答：由于简单的 MFC应用程序框架没有把数据的存储部分和与用户的交互部分分开， 所以类违背了面向对象程序设计的“单一职责原则”，从而使窗口类笨重杂乱，没有灵活性。而在文档 / 视图结构中则由于遵循了“ 单一职责原则 ”，从而使文档类和视图类既有分工又有合作，代码清晰，程序架构灵活。
3-6 什么叫类信息表？它在对象动态创建中起什么作用？
答：类中存放了类信息的一个 CruntimeClass 结构类型数据。其中的主要内容为类名称和指向对象构建函数的指针， 建立该表的目的就是为了能在运行期根据类名称调用构建函数来动态创建对象。
3-7 MFC所说的对象动态创建与 C++中的对象动态创建有什么区别？对象动态创建的核心是多少？
答： MFC所说的对象动态创建指的是在程序运行期间根据类名称创建一个对象；而 C++所说的对象动态创建是为待创建的对象动态分配存储空间。
4-1 为什么要使用 DC?
答：为了屏蔽硬件输出设备的多样性， Windows 系统为程序员提供了一个可以操作这些硬件却与硬件无关的接口， 于是就可以把对不同设备的操作方法统一起来。
4-3 如何把绘图工具载入设备描述环境？
答：使用 CDC 的成员函数 SelectObject 把绘图工具载入设备描述环境。
4-4 如何使用 CDC类提供的绘图方法绘图？
答：首先使用语句 CDC*pDC;创建一个 CDC类对象的指针， 然后就可以用下面格式的语句来调用 CDC类提供的各种方法了： pDC->方法名（参数）；
5-1 解释下列语句出的含义。
（1） CString s ；（2） CString s(“ Hello,Visual C++6.0 ”)；（3）CString s(‘ A’,100) ；（4）CString s(buffer,100) ；（5）CString s(anotherCString) 。
答：（1）构造一个长度为 0 的字符串对象。（2）构造一个名称为 s 的字符串对象，并把字符串初始化为 Hello,Visual C++6.0 。（3）构造一个名称为 s 的字符串对象， s 字符串的内容是 100 个 A。（4）构造一个名称为 s 的字符串对象， s 字符串的内容是 buffer 的头 100 个字符，再加一个 NULL。（5）构造一个名称为 s 的字符串对象， s 字符串的内容和 anotherCString 字符串的内容相同。
5-2 执行：
Cstring s(Cstring(“ Hello,world ”).Left(6)+Cstring(“ Visual C++”).Right(3)) ；语句后， s 字符串中的内容是什么？
答： Hello,C++ 。
5-3 现有语句 Cstring s(“ My,name,is,C++ ”)；若想将 s 字符串中的“ ，”号全部更换成“ ”，将如何编写语句？
答：s.Replace(‘ ,’,‘ ’) ;pDC->TextOut(1,1,s) ；
5-4 CString 创建时只分配 128B 的缓冲区，如何分配更大的缓冲区？
答：使用 GetBuffer() 函数。例如： CString s ；s.GetBuffer(1024) ；
6-2 分别说明什么是 SDI 界面的程序和什么是 MDI 界面的程序？
答：用户使用应用程序时， 如果该程序一次只能打开一个文档， 那么这种程序就叫做 SDI 界面的程序，反之就叫做 MDI界面的程序。
6-3 在使用 VC++提供的应用程序向导 MFC AppWizard 生成程序框架时， 有哪几个机会允许程序员选择应用程序窗口的样式？
答：一是在 MFCAppWizard-Step 1 时，选择 SDI、MDI 和基于对话框界面的窗口样式。二是在 MFC AppWizard-Step 4 中，可以确定窗口上诸如工具条、状态条、外观等一些选择。三是在 MFC AppWizard-Step 4 选择 Advanced 按钮后弹出的对话框中，选择窗口的样式。
6-5 如何用 MFC提供的程序设计向导实现具有可拆分窗口的界面程序？
答：在 MFC提供的程序设计向导 MFC AppWizard 的第四步中，即在 MFC AppWizard-Step 4 of 6 对话框中按下 Advanced 按钮，在随后打开的 Advanced Options 对话框中选择 Window Styles 选项卡，并在该选项卡中选择 Use split window 复选项。 这样， 由向导生成程序就会具有可拆分窗口的界面了。
6-6 文档类的成员函数 UpdateAllViews 的作用是什么？
答：通知文档所对应的所有窗口同时进行重绘。
6-7 为什么拆分窗口的显示更新必须要同步？
答：因为应用程序的所有拆分窗口显示的应该是同一个文档， 所以当文档发生变化时， 该文档所对应的窗口当然要同时更新显示以正确地反映文档的内容。
6-8 什么是无效显示区 ？
答：无效显示区一般定义为窗口用户区上的一个矩形区域， 这个区域因覆盖所有因文档发生变化而需要重绘的部分。 当程序需要重新绘制一个图形时，只要重新绘制该矩形内部的图形就可以了。
6-9 如何提高拆分窗口同步更新的效率？
答：原则上，想办法只绘制无效显示区。
7-1 鼠标消息分哪两类， 它们之间有什么区别？
答：根据产生鼠标消息时鼠标光标所处的位置，鼠标消息分为： 客户区鼠标消息和非客户区鼠标消息两类。 在应用程序窗口中， 用户可以绘图的部分叫做客户区或者用户区， 而除此之外的区域叫非客户区。 鼠标在客户区产生的消息叫客户区鼠标消息， 在非客户区产生的消息叫非客户区鼠标消息。
7-2 常用的客户区鼠标消息有哪些？
答： WM_LBUTTONDBCLK 双击鼠标左键 WM_LBUTTONDOWN 按下鼠标左键 WM_LBUTTONUP 释放鼠标左键 WM_MOUSEMOVE 移动鼠标 WM_RBUTTONDBCLK 双击鼠标右键 WM_RBUTTONDOWN 按下鼠标右键 WM_RBUTTONUP 释放鼠标右键
7-3 在程序设计中，如何使用非客户区鼠标消息？
答：首先， 在主框架窗口类的声明中手工添加非客户消息响应函数的声明， 然后在主框架窗口类实现文件的消息映射表中添加消息映射， 最后在主框架窗口类的实现文件中， 添加鼠标响应函数并实现它。

7-4 如何安全地接收应用程序窗口以外的鼠标消息？

答：在一般情况下，应用程序窗口是不会接收窗口之外的鼠标消息的，如果用户想接收应用程序窗口之外的鼠标信息，必须设法捕获鼠标信息。在 Windows 中，声明了一个专门用来捕获鼠标消息函数 CWnd*SetCapture；该函数一旦被调用，则所有的鼠标消息都将发往应用程序的窗口中。在捕获鼠标消息并完成了所应该做的工作之后，应用程序应该及时释放鼠标，以使鼠标可以按系统预定的正常方式发送信息，否则将使鼠标的一些正常作用失效。释放鼠标要使用下面的这个函数：BOOL ReleaseCapture 。

7-5 什么样的窗口才能接收键盘消息？

答：在 Windows 中，有时会同时打开多个窗口。在这些窗口中只有一个是活动窗口，这个窗口一般是屏幕上位置最靠前的窗口，它的特征是其标题栏被点亮的，而不是灰色的。只有活动窗口才具有输入焦点，而 Windows 中规定只有具有输入焦点的窗口才能接收键盘消息，也就是说，只有活动窗口才能接收键盘消息。

7-6 为什么在 Windows 应用程序中不直接使用键盘的扫描码，而使用与键盘无关的虚拟码？怎样理解 Windows 中设备无关性这个概念，设备无关性对编写应用程序有什么作用？

答：键盘的扫描码是当用户直接敲击键盘上的按键时，由键盘的接口直接产生的与该键对应的一种编码。由于市面上的键盘种类很多，所以不同类型的键盘产生的扫描码有可能是不同的，也就是说，这种扫描码是与具体的键盘相关的。这样在编写程序时会有很大的不便。所以在 Windows 编程中提出了设备无关性这个概念，它是基于通用性来设计的，基于这种方法设计出来的程序是不依赖于具体的硬件的，甚至不依赖于软件。它不单单是针对键盘。另外，它还应用在网络通信等方面。因此，设备无关性为人们编写程序带来了很大的方便。

7-7 键盘消息分为哪几类？哪些键只产生按键消息，不产生字符消息？

答：键盘消息可以分成：按键消息和字符消息两类。按键消息分为系统按键消息(WM_KEYDOWN,WM_KEYUP)和非系统按键消息（WM_SYSKEYDOWN,WM_SYSKEYUP）。字符消息也同样分为系统字符消息（WM_CHAR,WM_DEADCHAR）和非系统字符消息（WM_SYSCHAR,WM_SYSDEADCHAR）。值得注意的是，系统按键消息只能产生系统字符消息，非系统按键消息只能产生非系统的字符消息。在 Windows 中一些键是只产生按键消息而不产生字符消息的，这些键包括 Shift 键、Ctrl 键、功能键、光标移动键、特殊字符键。

7-8 在程序中如何确定窗口何时具有输入焦点，何时失去输入焦点？

答：当应用程序的窗口获得输入焦点时，会发出 WM_SETFOCUS 消息；而当窗口失去输入焦点时，会发出 WM_KILLFOCUS 消息。如果一个窗口获得了输入焦点，便可以用键盘对这个窗口进行操作。

8-1 在 Windows 应用程序中，什么样的数据称为资源？常用资源有哪些？

答：资源是一种数据。在应用程序启动后，它们仍然驻留在硬盘上的可执行文件中，只是在应用程序需要时，才从可执行文件中读取它们。常用的资源有菜单、图标、字符串、快捷键、位图等。

8-2 在 Visual C++ 中，编辑资源数据可以使用哪两种方法？

答:（1）在文本编辑器中直接对资源脚本文件和资源头文件进行编辑的方法。

（2）使用 Visual C++ 的资源编辑器对资源脚本文件和资源头文件进行编辑的方法。

8-3 程序运行时，用户选中一个菜单项，会发出哪种消息？根据什么来判断消息源？

答：用户选中菜单项时，会发出 WM_COMMAND 消息，系统根据菜单项的标识 ID 来识别是哪一个菜单项发出的消息。

8-4 在程序中如何使用图标资源？

答：先用图标编辑器制作图标，以扩展名 ico 把图标文件存盘，并把这个图标文件先加入工程的资源文件夹中，然后在工程的资源头文件中定义资源的标识，在资源描述文件中声明图标文件的路径，这样就可以在程序中需要的地方使用它了。

8-5 简述在 mfc 中使用位图资源的步骤。

答：（1）使用 LoadBitmap 函数把位图资源载入位图对象。（2）用 GetBitmap 获得位图信息。（3）用以下代码把位图选入内存环境变量。CDC MemDC//定义设备环境对象 MemDC.CreateCompatibleDC(NULL)；// 创建内容设备环境 MemDC.SelectObject(&m_Bmp)；（4）用 BitBlt 函数显示位图。

9-4 什么是序列化？什么是永久性对象？

答：序列化是面向对象程序设计中应对象这种数据的存储和恢复的要求而产生的一种文件读写机制。具有序列化能力的对象叫做永久性对象。

9-5 设计永久性类的时候必须使用哪两个宏？

答：宏 DECLARE_SERIAL 和 IMPLEMENT_SERIAL

10-1 简述在应用程序的窗口中使用一个控件的步骤。

答：首先在使用控件的类中声明控件，在合适的位置创建对象，然后向应用程序的消息映射中添加需要的消息，最后实现消息响应函数。

10-2 怎样才能使控件成为窗口的子窗口并且在窗口中可见？

答：为了使控件成为窗口的子窗口并且在窗口中可见，两个控制样式的常数是所有控件都必须使用的，一个是 WS_CHILD，另一个是 WS_VISIBLE，前者使控件成为应用程序窗口的子窗口，后者使控件可见。在使用多个常数指定控件样式时，应该用符号“ | ”将其进行连接。

10-3 为何创建每个控件一般都要传递 this 参数给 Create 函数？

答：因为在一般的情况之下都是为某一窗口对象创建控件，所以必须调用 Create 函数创建控件时，在控件的父窗口参数要用 this 作为参数。

10-5 控件的标识有什么用途？一般在应用程序的什么位置创建控件？

答：控件标识符的作用是用来区分应用程序中的不同控件的。一般情况下，创建控件的最佳位置在 OnCreate 成员函数。

10-6 按钮控件能创建哪三种不同的形式？

答：下压按钮、复选框和单选按钮。

11-1 什么是对话框模板资源描述文件？

答：用来描述对话框外观及对话框上控件布局的文本文件叫做对话框模板资源文件。

11-2 用户定义的对话框类派生自哪个类？

答:Cdialog 。

11-3 通常在什么地方进行对话框的初始化？

答：通常在类 CDialog 的 OnInitDig 成员函数中进行对话框的初始化。这个函数在对话框启动后，且还没有显示的时候被调用。

11-4 MFC有哪些通用对话框类？

答：CFileDialog 、CColorDialog 、CFontDialog 、CFindReplaceDialog 、CPageSetupDialog 和 CprintDialog 。

11-5 Windows 有哪两类对话框？它们的区别是什么？

答：模式对话框和非模式对话框。它们的区别为模式对话框直到退出对话框才返回应用程序，非模式对话框可以与应用程序同时工作。

基本概念：

Windows把为这种复杂对象所定义的标识叫做句柄

第一章 Windows 应用程序基础知识

1、Windows 应用程序是靠消息来驱动的，消息是一个描述事件的结构。

2、在 Windows 应用程序的主函数中，首先要注册窗口类型，然后创建并显示窗口。创建窗口后程序就进入消息循环，在消息循环中，程序不断地获得消息并将信息派送给对应的窗口函数进行处理。

3、窗口函数是处理消息的地方，它为 switch-case 结构，每一个 case 对应一段消息响应代码。

4、用函数 Windows 应用程序进行封装可以使程序的结构更为清晰。

第二章 Windows 应用程序的类封装

1、CwinApp 类是 MFC对 Windows主函数的封装，通过派生 CwinApp 可以得到自己的应用程序类，在应用程序类中主要实现了全局初始化操作，应用程序类创建了主窗口后便进入了消息循环。

2、应用程序的主窗口一般都是 CframeWnd的派生类，可以通过派生该类得到自己的主窗口类。

3、Windows 应用程序的窗口函数封装到 CcmdTarget 类中，所有希望响应消息的类都应该以 CcmdTarget 为基类来派生。

4、MFC是用消息映射表来实现消息与消息响应函数之间的映射的。MFC通过宏来声明和实现消息映射表。MFC的这种表驱动的机制使消息处理结构变得更加清晰、明了。

第三章 MFC应用程序框架

1、应用程序类、窗口框架类、视图类、文档类构成了应用程序的框架，框架的功能是通过各类之间的协调工作实现的。

2、MFC采用文档 / 视图结构来实现数据和数据表示的分离，文档视图的分离有利于数据和数据表示的单独改变。

3、MFC用类信息表存储了动态创建类对象时所需要的信息。

4、在类中使用宏 DECLARE_DYNCREATE 和

IMPLEMENT_DYNCREATE 使类具有动态创建对象的能力。

5、定义一个具有动态创建对象能力类时，必须在该类中定义一个无参数的构造函数。

6、在应用程序中，使用宏 RUNTIME_CLASS 来获得类信息表。

第四章 图形

1、 Windows 提供了图形用户接口使用户得以在窗口中绘图。

2、在 MFC 中使用 CDC 类的派生类向窗口和打印机等输出设备绘图。每个设备环境中都包含画笔、画刷、位图、调色板、字体等 GDI 对象。

3、可以通过创建 GDI 对象并将其选入设备环境来完成所需要的绘图操作。

第五章 MFC 的通用类

1、群体数据类基本上都是通过模板类实现的

2、视图类对象是用成员函数 GetDocument 获得文档类对象指针的，然后视图对象就可以通过这个指针来访问文档对象中的数据。

第六章 Windows 应用程序界面的设计

1、每次绘图操作结束后要调用视图类的成员函数 InvalidateRect() 启动 OnDraw() 函数以更新显示。

2、文档 / 视图类型的应用程序可以实现一个文档多个显示，但是在文档的内容发生改变的时候，要对所有的视图进行更新。

3、在需要时，应用程序的界面可以设计为带有滚动条的窗口形式。

第七章 鼠标和键盘

1、在应用程序的界面上，可以通过对鼠标左击、右击、移动等事件的处理来响应用户的鼠标输入。

2、鼠标消息有用户区鼠标消息和非用户区鼠标消息两种，在应用程序中主要使用用户区鼠标消息。

3、可以用消息捕获函数来捕获窗口外的鼠标消息，以完成某些特殊的操作。

4、可以通过处理字符消息、按键等键盘消息对用户的键盘操作进行响应。

5、在计算机的显示器屏幕上，如果有多个窗口存在的话，则具有焦点的窗口所对应的应用程序是具有接收用户消息能力的程序，这个程序叫做“正在活动状态的应用程序”。

第八章 资源

1、资源是与应用程序逻辑数据相隔离，用资源描述文件说明，由资源编辑器生成，可以动态加载方式供 Windows 应用程序使用的数据。资源是程序用户界面的重要组成部分。常用的资源有菜单、加速键、图标、位图等。

2、程序所需的资源使用资源描述文件来说明，并在资源文件中用标识符唯一地进行标识。

3、资源可以使用 VC++ 的资源编辑器来创建和编辑，也可以使用文本编辑器来编辑。

4、菜单的使用与 Windows 的命令消息 WM_COMMAND 相关。

5、菜单项消息映射宏的格式是： ON_COMMAND（菜单项 ID，消息响应函数名）

6、菜单项动态修改的消息映射宏的格式是： ON_UPDATE_COMMAND_UI（菜单项 ID，消息响应函数名）

7、在文档 / 视图结构的程序中，资源的加载是由应用程序类的 InitInstance 函数中通过构造 CdocTemplate 对象来完成的。

8、加速键在资源描述文件中与所对应的菜单项关联。

9、图标使用 VC++ 开发环境的菜单 Project|Add To Project|Files 添加。

10、在应用程序中，位图用 Cbitmap 对象来保存，由成员函数 LoadBitmap 来加载，在显示时需先绘制到内存 DC 中，然后再用 BitBlt 函数把它由内存 DC 复制到显示设备的 DC。

第九章 MFC 的文件处理机制

1、文件是存储在永久性存储介质上的数据的集合。在面向对象的应用程序中也涉及对象存盘的问题。对象存盘可以使用序列化的机制实现。

2、MFC 把文件的打开、关闭、读写操作封装在类 CFile 中。CFile 对象代表一个磁盘文件，使用 CFile 对象可以直接对文件进行操作。该类有一个很有用的派生类： CmemFile。

3、Carchive 是对 CFile 的再封装，它重载了插入符“《”和提取符“》”，它是一种 I/O 流，它借助 CFile 类对象完成磁盘文件数据的存取操作。

4、对象序列化是指将类对象转换成 byte/bit 流，以便于对象通过网络传输或保存在磁盘上，对象序列化是将 byte/bit 流化的对象转换成内存中的类对象的过程。MFC 使用 Carchive 对象来完成对象的序列化。

5、具有读写自身能力的对象称为永久性对象。MFC 通过宏 DECLARE_SERIAL 和 IMPLEMENT_SERIAL 给类添加动态创建对象和序列化操作所需的代码。宏 DECLARE_SERIAL 用在类声明中，宏 IMPLEMENT_SERIAL 用在类实现中。同时，该类必须从 Cobject 类或其派生类派生，并重载 Serialize() 函数．Serialize() 函数借助类 Carchive 对象实现对象的序列化。

第十章 控件

1、控件是应用程序窗口的子窗口。MFC 的控件类封装了 Windows 的标准控件和通用控件，这些控件类都派生于类 CWnd。

2、静态文本控件由类 Cstatic 封装，按钮控件由类 Cbutton 封装，编辑控件由类 Cedit 封装，进度条控件由类 CprogessCtrl 封装，微调器控件由类 CspinButtonCtrl 封装，图像列表控件由类 CimageList 封装，列表视图控件由类 ClistCtrl 封装。控件类的使用与窗口类 CWnd 的使用基本相同。

3、控件颜色的设置在 Windows 消息 WM_CTLCOLOR 的消息响应函数 OnCtlColor 中完成。其消息映射宏是： ON_WM_CTLCOLOR()

第十一章 对话框

1、对话框的基本行为由类 CDialog 封装，对话框的外观由模板资源定义。

2、2、对话框模板资源可以使用 VC++ 的资源编辑器来创建和编辑。

3、调用 CDialog 的成员函数 DoModal 可以创建并打开模态对话框。按钮 OK 和 Cancel 是对话框中系统顶置的两个按钮，分别对应关闭对话框时的确定状态和取消状态。

4、对话框使用数据交换（DDX）机制实现控件与变量之间的数据交换，使用数据检验（DDV）机制检验通过控件录入的数据是否合乎规格。

5、使用 MFC ClassWizard 为对话框类添加 Membe Varaible 并与相应的控件绑定。DDX 函数具体完成控件和变量的绑定和数据交换。一对控件和变量由一个 DDX 函数绑定，并由 MFC ClassWizard 自动添加到对话框成员函数 DoDataExchange 中。DoDataExchange 被对话框成员函数 UpdateData 调用，并由其参数控制数据的交换方向。

6、MFC 还对 Windows 通用对话框进行了封装。它们分别是 CColorDialog 、CfileDialog 、CFindReplaceDialog 、CFontDialog 、CprintDialog

7、非模态对话框使用 CDialog 类的 Create 成员函数来创建和显示，使用 DistroyWindow 函数来关闭。

8、属性页是 CpropertySheet 类派生类的对象，它包含若干属性页面。属性页面是 CpropertyPage 类派生类的对象，它是一个对话框。

句柄：就是一个 4 字节长的唯一的数，用以标识许多不同的对象类型。

API 函数：用来开发 Windows SDK 应用程序的软件开发工具包是用 C 语言编写的一个大型函数库，这个库中的函数叫做 API 函数

消息映射表：在 Windows SDK 应用程序的窗口函数中，是采用 switch-case 分支结构实现消息处理的，这种方式不适合面向对象设计的要求。因此 MFC 建立了一套自己的消息映射机制——消息映射表。

类信息表： MFC 程序在不同的场合下还经常用到类的其他信息，于是 MFC 就把这些信息统统都放在映射表项中，该表即叫着类信息表。

资源：资源是一种数据。在应用程序启动后，它们仍然驻留在硬盘上的可执行文件中，只是在应用程序需要时，才从可执行文件中读取它们。

填空

1、 Windows 应用程序的主函数有哪三个主要任务？

答：注册窗口类、创建应用程序的窗口和建立消息循环。

2、常见句柄的名称： HWND 窗口句柄 HINSTANCE 当前程序应用实例句柄 HCURSOR 光标句柄 HFONT 字体句柄 HPEN 画笔句柄 HBRUSH 画刷句柄 HDC 图形设备环境句柄 HBITMAP 位图句柄 HICON 图标句柄 HMENU 菜单句柄 HFILE 文件句柄

3、消息循环的三个函数的作业： Getmessage：从消息队列中获取消息； Translatemessage: 把键盘消息翻译成字符消息； Dispatchmessage ：

把消息派送给系统，并通过系统发送给窗口。

4、MFC应用程序的界面有哪三种方式？

答：（1）单文档界面；（2）多文档界面；（3）基于对话框界面。

5、非模态对话框是使用 Cdialog 类的成员函数 Create() 来创建和显示的。模板对话框使用 CDialog 类的成员函数 DoModal() 来创建对话框。

简答

1、 在 MFC中 CDC的派生类有哪几个， 试说出它们的作用。

答： CclientDC 应用在除 WM_PAINT消息之外的消息处理函数中， 提供窗口客服区的设备描述环境。

CmetaFileDC 代表 Windows 图元文件的设备描述环境。在创建与设备无关的并且可以回收的图像时使用这个类型的 DC。

CpaintDC 在 WM_PAINT消息的处理函数 OnDraw 中使用的窗口用户区的设备描述环境。

CwindowDC提供在整个窗口内绘图的设备描述环境。

2、如何使类具有序列化能力？

答：类必须满足以下三个条件。 （1）从 Cobject 类或其派生类派生，并重写 Serialize() 函数；（2）必须在类声明文件中使用序列化声明宏 DECLARE_SERIAL()在类实现文件中使用序列化实现宏 IMPLEMENT_SERIAL()；（3）必须定义一个无参数的构造函数， 以满足动态创建对象的需要。

3、标准控件和通用控件有什么不同？

答：主要区分是目标不同。标准控件在最早的 Windows版本中就已经存在。通用控件是在后来的版本中添加进去的， 目标是使用户界面看起来更加现代化。标准控件发送的是 WM_COMMAND消息，通用控件则是 WM_NOTIFY消息。

程序

例 5-1 CString 类的应用实例

在视图类的鼠标左键按下消息中输入如下代码。

```
void CMFCexp5_1View::OnLButtonDown(UINT nFlags,CPoint point)
{
    CString strl= ”This is an easy way to perform ”;
    CString str2= ”string concantenation! ”;
        CString str3=str1+ ”” +str2;
        AfxMessageBox(str3,MB_OK|MB_ICONINFORMATION);
CView::OnLButtonDown(nFlags,point);
}
```

5-2 编写一个程序， 当按下鼠标左键时， 在鼠标的光标位置会显示一个随机大小的矩形。

（2）在应用程序头文件 StdAfx.h 中加入包含命令。

```
#include<afxtemp1.h>
```

（3）在视图类的声明中定义一个存放 CRect 类型元素的数组 m_Rectag。

```
Class CMFCexp5_2Doc:public Cdocument
{protected:CArray<CRect,CRect&>m_Rectag;};
```

（4）在视图类的构造函数中定义 m_Rectag数组的大小。

```
CMFCexp5_2View::CMFCexp5_2View()
{m_Rectag.SetSize(256,256)
}
```

（5）在视图类鼠标左键按下消息响应函数中，将每次单击鼠标产生的矩形数据存入数组。

```
void CMFCexp5_2View::OnLButtonDown(UINT nFlags, CPoint point)
{int r=rand()%50+5;
CRectRet(point.x-r,point.y-r,point.x+r,point.y+r);
    m_Rectag.Add(Ret);
InvalidateRect(Ret,FALSE);
CView::OnLButtonDown(nFlags, point);}
```

（6）在视图类的 WM_PAINT消息响应函数中重画数组中的矩形。

```
void CMFCexp5_2View::OnDraw(CDC * pDC)
{for(inti=0 ;i<m_Rectag.GetSize();i++)
    pDC->Rectangle(m_Rectag[i]);}
```

5-3 用文档 / 视图结构程序完成例 5-2

（5）在视图类的 OnLButtonDown 函数中设置指向文档的指针并通过该指针获取文档的成员。

```
void CMFCexp5_3View::OnLButtonDown(UINT nFlags, CPoint point)
{CMFCexp5_3Doc*pDoc=GetDocument();//  获取文档指针
int r=rand()%50+5;
CRectRet(point.x-r,point.y-r,point.x+r,point.y+r);
    pDoc->m_Rectag.Add(Ret);//  向文档中数组添加元素
    InvalidateRect(Ret,FALSE);//  触发 OnDraw()函数
    CView::OnLButtonDown(nFlags, point);
}
```

（6）在 OnDraw函数中画出数组中的矩形。

```
void CMFCexp5_3View::OnDraw(CDC* pDC)
{
    CMFCexp5_3Doc* pDoc = GetDocument();//  获取文档指针
    ASSERT_VALID(pDoc);
    for(int i=0;i<pDoc->m_Rectag.GetSize();i++)

    pDC->Rectangle(pDoc->m_Rectag[i]);
}
```

7-1 编写可以在用户区中绘制一个矩形动应用程序， 在按下鼠标左键后， 这个矩形会把它的左上角移动到鼠标位置；而当按下 Shift 键的同时，按下鼠标左键，则矩形恢复原位置。

（2）在文档类中添加一个数据成员 tagRec 来存储数据。

```
Class CMFCexp7_1Doc:public Cdocument
{ Public: CRect m_tagRec;};
```

（3）在文档类的构造函数中 ，初始化数据成员。

```
CMFCexp7_1Doc:: CMFCexp7_1Doc()
{
    m_tagRec.left=30; m_tagRec.top=30;
    m_tagRec.right=350;  m_tagRect.bottom=300;
}
```

（4）在其视图类的鼠标右键按下消息响应函数中写入如下代码。

```
void CMFCexp7_1View::OnRButtonDown(UINT nFlags,CPoint point)
{
    CMFCexp7_1Doc*pDoc=GetDocument();
if(nFlags&MK_SHIFT)
{pDoc->tagRec.left=30;
pDoc->tagRec.top=30;
pDoc->tagRec.right=350;
pDoc->tagRec.bottom=300;}
else
{pDoc->tagRec.left=point.x
pDoc->tagRec.top= point.y;
pDoc->tagRec.right= point.x +320;
pDoc->tagRec.bottom= point.y +270;}
```

InvalidateRect(NULL,TRUE);
CView::OnRButtonDown(nFlags,point);
}
（5）在视图类的 OnDraw 函数中编写如下代码。
void CMFCexp7_1View::OnDraw(CDC*pDC)
{
CMFCexp7_1Doc*pDoc=GetDocumengt();
ASSERT_VALID(pDoc);
pDC->Rectangle(pDoc->m_tagRec);}
例 7-2 一个测试鼠标移动消息的程序。
(2) 在文档类声明中，添加一个点类的数据成员 m_point.
Class CMFCexp7_2Doc:public Cdocument
{ Public: Cpoint m_point;};
（3）在视图类中添加鼠标移动消息响应函数，并输入如下代码。
void CMFCexp7_2View::OnMouseMove(UINT nFlags, CPoint point)
{
CMFCexp7_2Doc*pDoc=GetDocument();
pDoc->m_Point=point;
InvalidateRect(NULL,FALSE);
CView::OnMouseMove(nFlags, point);
}
（4）在视图类的 OnDraw函数中添加如下代码。
void CMFCexp7_2View::OnDraw(CDC* pDC)
{
CMFCexp7_2Doc* pDoc = GetDocument();
ASSERT_VALID(pDoc);
CPoint point(30,30);
pDC->MoveTo(point);
pDC->LineTo(pDoc->m_Point);
}
例 7-3 编写一个程序，使鼠标的光标在标题栏或窗口边框上移动时， 在用户区显示鼠标光标的位置。
(2) 在主框架窗口类 CMainFrame的声明中， 手工添加消息响应函数的声明。
Afx_msg void OnNcMouseMove(UINT nHitTest, CPoint point)
(3) 在主框架窗口类 CMainFrame 的实现文件的消息映射表中，添加消息映射。
BEGIN_MESSAGE_MAP(CMainFrame, CFrameWnd)
ON_WM_NCMOUSEMOVE()
END_MESSAGE_MAP()
（4）在主框架窗口类 CMainFrame 的实现文件中，添加鼠标响应函数的实现。
Void CMainFrame::OnNcMouseMove(UINT nHitTest,
CPoint point)
{ CClientDC clientDC(this);
char s[20];
wsprintf(s, "X=%d Y=%d ", point.x, point.y);
clientDC.TextOut(20, 20, s);
CFrameWnd::OnNcMouseMove(nHitTest,point); }
例 7-4 当鼠标左键按下时，可以捕获鼠标消息的程序。
void CMFCexp7_4View::OnMouseMove(UINT nFlags, CPoint point)
{
char str[50];
CClientDC dc(this);
dc.TextOut(20, 20, "WM_MOUSEMOVE");
wsprintf(str, "X: %d Y: %d ", point.x, point.y);
dc.TextOut(200, 20, str);
CView::OnMouseMove(nFlags, point);
}
void CMFCexp7_4View::OnLButtonDown(UINT nFlags, CPoint point)
{
SetCapture(); // 捕获鼠标消息
CView::OnLButtonDown(nFlags, point);
}
void CMFCexp7_4View::OnLButtonUp(UINT nFlags, CPoint point)
{
ReleaseCapture( ); // 释放鼠标捕获
CView::OnLButtonUp(nFlags, point);
}
例 7-5 设计一个程序， 在用户区显示一个圆形， 当分别按下键盘上的左箭头键或右箭头键时， 可以使这个圆形向左或者向右移动。
(2) 在文档类声明中声明一个存放圆形外接矩形的数据成员。
Class CMFCexp7_5Doc:public Cdocument
{ Public: CRect m_crlRect;};
(3) 在文档类的构造函数中初始化 m_crlRect.
CMFCexp7_5Doc::CMFCexp7_5Doc()
{ m_crlRect.left=30;
m_crlRect.top=30;
m_crlRect.right=80;
m_crlRect.bottom=80;}
(4) 因为左箭头键和右箭头键都不是字符键， 因此在程序中要用虚拟键码识别这两个键。
void CMFCexp7_5View::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags)
{
CMFCexp7_5Doc*pDoc=GetDocument();
CRect clientRec;
GetClientRect(&clientRec);
switch(nChar)
{
case VK_LEFT:
if(pDoc->m_crlRect.left>0)
{
pDoc->m_crlRect.left-=5;
pDoc->m_crlRect.right-=5;
}
break;
case VK_RIGHT:
if(pDoc->m_crlRect.right
<=(clientRec.right-clientRec.left))
{
pDoc->m_crlRect.left+=5;
pDoc->m_crlRect.right+=5;
}
break;
}
InvalidateRect(NULL,TRUE);
CView::OnKeyDown(nChar, nRepCnt, nFlags);
}
（5）在 OnDraw()函数中写入如下代码。
void CMFCexp7_5View::OnDraw(CDC* pDC)
{
CMFCexp7_5Doc* pDoc = GetDocument();
ASSERT_VALID(pDoc);
pDC->Ellipse (pDoc->m_crlRect);
}
例 7-6 给例 7-5 程序增加一个功能，当分别按下 R键或者 L 键时，可以使用户区的圆形向右或向左移动。
void CMFCexp7_6View::OnChar(UINT nChar, UINT nRepCnt, UINT nFlags)
{
CMFCexp7_6Doc*pDoc=GetDocument();
CRect clientRec;
GetClientRect(&clientRec);// 获得窗口用户区的尺寸
switch(nChar)
{
case 'L':
if(pDoc->m_crlRect.left>0)
{
pDoc->m_crlRect.left-=50;
pDoc->m_crlRect.right-=50;
}
break;
case 'R':
if(pDoc->m_crlRect.right
<=(clientRec.right-clientRec.left))
{
pDoc->m_crlRect.left+=50;
pDoc->m_crlRect.right+=50;
}
break;
}
InvalidateRect(NULL,TRUE);
CView::OnChar(nChar, nRepCnt, nFlags);
}