**第一次作业**

**C++语言中函数的作用是什么？**

函数是一段可重用的代码，用于完成特定的任务。函数可以接受参数和返回值，并且可以与其他代码分离。使用函数可以使代码更易于维护和阅读，同时提高代码的复用性和可扩展性。函数可以定义在类中或在全局空间中，并且可以按值传递参数、按引用传递参数或按指针传递参数。

**在C++中如何进行类的定义？**

可以使用关键字class来定义一个类。类的定义通常包括以下几个部分：

1. 类名：用于标识该类的名称。

2. 成员变量：描述该类对象所具有的属性或状态。

3. 成员函数：描述该类对象所能够执行的操作或行为。

4. 构造函数和析构函数：用于初始化和清理该类对象。

5. 访问修饰符：用于控制类成员的访问权限，包括public、private和 protected 。

下面是一个简单的类定义示例：

class Person {

public:

// 成员函数

void setName(const string& name);

string getName() const;

private:

// 成员变量

string m\_name;

};

在上面的代码中， Person 是类名， setName 和 getName 是成员函数， m\_name 是成员变量。 public 访问修饰符表示这些成员函数可以从类外部访问，而 private 访问修饰符表示 m\_name 成员变量只能从类内部访问。

**构造函数和析构函数的功能是什么？如何创建构造函数和析构函数？**

构造函数用于在创建对象时初始化其成员变量，为对象分配空间并执行必要的初始化操作。析构函数则在对象被销毁时自动执行一些清理操作，例如释放内存和关闭文件等。

下面是一个简单的类示例，包含构造函数和析构函数：

class MyClass {

public:

// 构造函数

MyClass() {

cout << "Creating an object of MyClass" << endl;

}

// 析构函数

~MyClass() {

cout << "Destroying an object of MyClass" << endl;

}

};

在上面的代码中，构造函数 MyClass() 使用与类名相同的名称，并且没有返回类型，它将在对象创建时自动调用。这里的构造函数是默认构造函数，因为它没有参数。当对象被销毁时，析构函数 ~MyClass() 将自动调用。同样地，析构函数也没有返回类型，名称以 ~ 开头，并且不需要任何参数。

**C++中派生的含义是什么？**

派生是一种面向对象编程的重要概念，用于描述如何从一个或多个现有的类中创建新类。

派生类（也称为子类）可以继承一个或多个基类（也称为父类），并且可以扩展或重写其父类的成员函数和数据成员。这使得程序员可以使用已有的代码来构建新的功能，同时避免了代码的复制和粘贴。

派生类通过使用关键字 class 来声明，并在类名后面跟着冒号和一个或多个基类名字。例如：

class BaseClass {

public:

void baseFunc();

};

class DerivedClass : public BaseClass {

public:

void derivedFunc();

};

在上面的示例代码中， DerivedClass 是一个派生类，它公开继承自 BaseClass 。因此， DerivedClass 不仅包含自己的成员函数 derivedFunc() ，还能够访问 BaseClass 中的成员函数 baseFunc() 。

**C++中重载的作用是什么？**

重载是一种允许同名函数具有不同参数列表的机制。使用重载，程序员可以定义多个具有相同名称但不同签名（参数数量、类型或顺序）的函数。

重载的作用主要体现在以下两个方面：

1. 提高代码的可读性和可维护性：当一个函数需要处理不同类型或数量的参数时，使用重载可以使函数调用更清晰和易于理解，避免了创建多个具有不同名称的函数。

2. 增强了函数的灵活性和适用性：通过重载，可以为不同情况下的函数调用提供不同的实现，并且可以根据不同的需求进行定制。这也使得函数更加通用化和可复用。

下面是一个简单的函数重载示例：

void print(int num) {

cout << "The integer is: " << num << endl;

}

void print(double num) {

cout << "The double is: " << num << endl;

}

int main(){

int a = 5;

double b = 3.14;

print(a);

print(b);

return 0;

}

在上面的代码中，我们定义了两个名字都为 print 的函数，一个用于打印整数，一个用于打印双精度浮点数。在 main() 函数中，我们分别使用这两个函数来打印一个整数和一个双精度浮点数，由此演示了函数重载的功能。

**第二次作业**

**Windows编程中窗口的含义是什么？**

一个应用程序的窗口一般包含下列成分：

控制菜单框、下拉菜单、标题栏、工作区、窗口边界、最大化按钮、最小化按钮、垂直滚动条、水平滚动条。

编写一个Windows应用程序首先应创建一个或多个窗口，随后应用程序的运行过程即是窗口内部、窗口与窗口之间、窗口与系统之间进行数据处理与数据交换的过程。

**事件驱动的特点是什么？**

Windows程序设计是针对事件或消息的处理进行。

Windows程序的执行顺序取决于事件发生的顺序，程序的执行顺序是由顺序产生的消息驱动的，但是消息的产生往往并不要求有次序之分。

事件驱动编程方法对于编写交互式程序很有用处，它避免了死板的操作模式。

**Windows应用程序中的消息传递是如何进行的？**

Windows应用程序利用Windows消息(Message)与应用程序及系统进行信息交换。

消息

消息号：由事先定义好的消息名标识

字参数(wParam) ：用于提供消息的附加信息

长字参数(lParam) ：用于提供消息的附加信息

**句柄是什么？有何作用？**

句柄是一个8字节长(相对64位操作系统)的数值，用于标识应用程序中不同的对象和同类对象中不同的实例。

应用程序通过句柄访问相应的对象信息。

**Windows应用程序的最基本构成应有哪些部分？**

1. 程序入口点（Entry Point）：是应用程序的启动点，当操作系统加载应用程序时，会自动调用程序入口点代码。

2. 消息循环（Message Loop）：是应用程序的核心组成部分，它负责检查、接收和处理从操作系统发送过来的消息，并将其分派给相应的窗口或控件进行处理。

3. 窗口过程（Window Procedure）：是窗口的消息处理函数，当窗口收到消息时，窗口过程会被调用，并对消息进行处理，并根据需要返回一个结果给操作系统。

4. 用户界面（User Interface）：包括各种控件和窗口组成的用户交互界面，例如按钮、文本框、列表框、菜单等。

5. 应用程序逻辑（Application Logic）：是应用程序的业务逻辑部分，它定义了应用程序需要完成的功能和处理流程。

**Windows应用程序常用的消息有哪些？**

窗口管理消息

初始化消息

输入消息

系统消息

剪贴板消息

控制处理消息

控制通知消息

滚动条通知消息

非用户区消息

MDI消息

DDE消息

应用程序自定义的消息

**第三次作业**

1. **GDI是什么的英文缩写？**

Graphics Device Interface

**2、 什么是设备的无关性？**

设备的无关性是指应用程序或操作系统能够不受硬件设备类型、品牌或型号的限制而运行。它是一种能够在不同的设备上运行的能力，而不需要针对每个设备编写独立的代码。在计算机图形学中，设备的无关性通常是指图形设备无关性，即应用程序能够使用相同的代码生成可在多种图形设备上渲染的图像。这样，开发人员就可以编写只需一次编译和调试，就能在不同的平台和设备上运行的程序。

**3、 GDI的主要任务是什么？起到什么样的作用？**

GDI负责系统与用户或绘图程序之间的信息交换，并控制在输出设备上显示图形或文字， 是Windows系统的重要组成部分。

开发人员只要建立与输出设备的关联，让系统加载相应的设备驱动程序即可。

**4、 什么是设备描述表？它的作用是什么？**

设备描述表是一种数据结构，用于描述硬件设备的属性和特性。在Windows系统中，设备描述表通常包含在驱动程序中，可以通过操作系统提供的API来访问。

设备描述表的作用是告诉操作系统如何与硬件设备进行交互。它包含了硬件设备的各种信息，例如设备类型、厂商ID、设备序列号、支持的功能和性能参数等。根据这些信息，操作系统可以选择适当的驱动程序来管理硬件设备，并向应用程序提供正确的接口。在Windows系统中，设备描述表通常使用一种名为“设备信息集”（Device Information Set）的结构来组织，每个设备都有一个唯一的标识符。应用程序可以使用基于设备描述表的API来遍历设备信息集，查找并枚举系统中可用的设备，并获取有关这些设备的详细信息。

**5、 图形刷新包括什么？**

1. 刷新请求
2. 对刷新请求的响应

(3) 有效的刷新方法

**6、 刷新请求有哪些？**

对象穿越

窗口移动

被覆盖后的恢复

**7、 应用程序什么情况下会发生刷新请求？同时发出什么消息？**

当用户区的内容需要刷新时，系统向应用程序消息队列发送WM\_PAINT消息，系统在应用程序的消息队列中加入该消息，以通知窗口函数执行刷新处理

**8、 什么是无效区域？**

窗口被另一个窗口覆盖的区域称为无效区域。

Windows系统为每个窗口建立了一个PAINTSTRUCT结构，该结构中包含了包围无效区域的一个最小矩形的结构RECT，应用程序可以根据这个无效矩形执行刷新操作。

**9、 有效的刷新方法有哪些？大家觉得哪种比较好？在代码中若主动要求刷新窗口，应该怎么办？**

记录事件。刷新时重新执行这个曾经发生的事件

保存副本。刷新时将副本拷贝到相应的窗口中

重新绘制。将图形绘制处理程序放在消息WM\_PAINT响应模块中，刷新时重绘图形

重新绘制比较好。将图形绘制处理程序放在消息WM\_PAINT响应模块中，刷新时重绘图形。

**10、 获取设备环境常用的两种方法是什么？**

调用函数BeginPaint或GetDC

**11、 调用BeginPaint函数和GetDc有什么联系与区别？在实际编程中怎样选择使用？**

(1) 调用BeginPaint函数

应用程序响应WM\_PAINT消息进行图形刷新时，主要通过调用BeginPaint函数获取设备环境

hdc=BeginPaint(hwnd,&ps)； //ps为PAINTSTRUCT类型结构

定义方式为：PAINTSTRUCT ps;

由BeginPaint函数获取的设备环境要用EndPaint函数释放

BOOL EndPaint(HWND hwnd, PAINTSTRUCT &ps)

(2) 调用GetDC函数

如果绘图工作并非由WM\_PAINT消息驱动，则调用GetDC函数获取设备环境。

hdc=GetDC(hwnd);

由GetDC函数获取的设备环境必须用ReleaseDC函数释放

void ReleaseDC(HWND hwnd);



**12、 什么是映像模式？为什么要用映像模式**

映像模式定义了将逻辑单位转化为设备的度量单位以及设备的x方向和y方向，程序员可在一个统一的逻辑坐标系中操作而不必考虑输出设备的坐标系情况

**13、 了解几种常用的映像模式，默认是哪种映像模式。**



默认是MM\_TEXT。

**14、 了解映像模式中函数的使用（不需要书面写），可以写部分重点函数的理解。**

设置设备环境的映像模式

SetMapMode(hdc,nMapMode); nMapMode为映像模式的整型标识符

获取当前设备环境的映像模式

nMapMode=GetMapMode(hdc);

窗口区域的定义由SetWindowExtEx函数完成，其函数原型形为：BOOL SetWindowExtEx

( HDC hdc,

int nHeight,nWidth， //以逻辑单位表示的窗口区域高宽度

LPSIZE lpSize， //函数调用前窗口区域尺寸的SIZE结构地址

);

视口区域的定义由SetViewportExtEx函数完成，函数原型为：

BOOL SetViewportExtEx

(

HDC hdc,

int nHeight,nWidth,//以物理设备单位表示的新视口区域高宽度

LPSIZE lpSize

);

**15、 画笔的操作有哪些？**

创建画笔

将画笔选入设备环境

删除画笔

**16、 了解画笔中使用的函数名字的英文意思，便于大家记忆函数，（不需要书面写出）**

**17、 了解常用的绘图函数，尤其是饼图函数，椭圆弧线函数。想一想怎么画圆，有没有画圆的函数？**

1．设置画笔当前位置的函数MoveToEx，

BOOL MoveToEx

(HDC hdc,

int X,Y, // X、Y分别为新位置的逻辑坐标

LPPOINT lpPoint //存放原画笔位置的POINT结构地址

)

2． 从当前位置向指定坐标点画直线的函数LineToEx，

BOOL LineToEx(HDC hdc,int X,int Y) //X和Y为线段的终点坐标

3． 从当前位置开始，依次用线段连接lpPoints中指定的各点

BOOL Polyline

( HDC hdc,

LPPOINT lpPoints, //指向包含各点坐标的POINT结构数组的指针

int nCount // nCount为POINT数组中点的个数

)

4． 绘制椭圆弧线的函数Arc，

BOOL Arc

(

HDC hdc,

int X1,intY1, //边框矩形左上角的逻辑坐标

int X2,int Y2, //边框矩形右下角的逻辑坐标

int X3,int Y3, //椭圆弧起始点坐标

int X4,int Y4 //椭圆弧终止点坐标

)

5． 绘制饼图，并用当前画刷进行填充

BOOL Pie

(

HDC hdc,

int X1,intY1, //边框矩形左上角的逻辑坐标

int X2,int Y2, //边框矩形右下角的逻辑坐标

int X3,int Y3, //椭圆弧起始经线的确定点坐标

int X4,int Y4 //椭圆弧终止经线的确定点坐标

)

6． 绘制矩形，并用当前画刷进行填充

BOOL Rectangle(HDC hdc,int X1,int Y1,int X2,int Y2)

7． 绘制圆角矩形，并用当前画刷填充

BOOL RoundRect (HDC hdc,int X1,int Y1,int X2,int Y2,

int nHeight, int nWidth)

8． 绘制椭圆，并用当前画刷填充

BOOL Ellipse(HDC hdc,intX1,intY1,intX2,intY2)

9．绘制多边形，并用当前画刷填充

BOOL Polygon(HDC hdc,LPPOINT lpPoints,int nCount)

**第四次作业**

1. 文本操作包括哪些步骤
2. 获得文本句柄
3. 设置字体、字符大小、字符颜色等有关属性
4. 将这些属性选入设备环境
5. 什么是物理字体、什么是逻辑字体？

物理字体是为特殊设备设计的，因而是设备相关的。

逻辑字体定义的字符集是设备无关的，它可以精确标度，因此得到广泛应用。

1. 选择系统字体的步骤是什么？

(1) 定义字体句柄变量：

HFONT hF; //hF为字体的句柄

(2) 调函数GetStockObject获得系统字体句柄

它返回的是系统的缺省字体

hF= GetStockObject( )：

(3) 调用函数SelectObject将字体选入设备环境

SelectObject(hdc,hF)；

1. 怎么设置字体的颜色？

设置字体颜色：SetTextColor(hdc,crColor)；

1. 文本输出的过程是什么？

1 获取字体信息

2 格式化文本

2.1 确定后续文本坐标

2.2确定换行时文本坐标

3 调用函数输出文本

1. 如何获取字体信息？

输出文本之前要获取字体的信息，如字符高度等，以确定输出格式和下一行字符的位置

GetTextMetrics (hdc,&tm)；//tm为TEXTMETRIC结构

调用该函数时，系统将当前字体的信息拷贝到tm标识的TEXTMETRIC结构中

1. 如何格式化文本？

(1) 确定后续文本坐标

确定后续文本的坐标,应先获取当前的字符串的宽度, 该工作由GetTextExtentPoint32函数完成，并把它存储于一个SIZE结构中。

BOOL GetTextExtentPoint32

(

HDC hdc,

LPCTSTR lpszString,//指定的字符串

int nLength, //字符串中的字符数

LPSIZE lpSize //返回字符串宽度及高度的SIZE数据结构的地址

)

(2)确定换行时文本坐标

通过计算当前行文本字符的高度与行间隔之和，即可得到换行时文本的起始坐标

字符的高度与行间隔均存储在tm指向的TEXTMETRIC结构中，换行时Y轴上文本的起始坐标cy为：

cy=tm.tmHeight+tm.tmExternalLeading;

1. 如何文本输出？

常用的文本输出函数TextOut原型如下：

BOOL TextOut

(

HDC hdc,

int X, int Y, //X, Y为用户区中字符串的起始坐标

LPCTSTR lpstring, //lpstring为显示的字符串

int nCount //nCount为字符串中的字节数

)；

**第五次作业**

1. 什么是扫描码？

扫描码是依赖于具体设备的，为达到设备无关性的要求，往往使用与具体设备无关的虚拟码，虚拟码是由Windows系统定义的与设备无关的键的标识

1. 从按键到应用程序获取消息经过什么样的过程？

键盘上的键对应一个唯一的标识值(扫描码)，按下或释放某键时产生扫描码，设备驱动程序截取键的扫描码，翻译成虚拟码，产生一条扫描码、虚拟码以及其他与击键有关的消息。消息通过设备驱动程序把消息放到系统的消息队列中，Windows从系统消息队列中取出消息，发送到相应的线程消息队列中，窗口过程取出键盘消息进行处理。

1. 什么是虚拟码？常用的虚拟码有哪些？这些虚拟码在哪里定义的？

虚拟码是一种与设备无关的键盘编码，它的值存放在键盘消息的wParam参数中，用以标识哪一个键被按下或释放，最常用的虚拟码已经在winuser.h中定义(C:\Program Files (x86)\Windows Kits\8.0\Include\um\WinUser.h)，



1. 什么叫输入焦点？窗口函数如何获得和失去输入焦点？

应用程序一般有几个窗口，但当按下某一个键时，只有一个窗口能接收到该键盘消息，接收这个键盘消息的窗口称为有“输入焦点”的窗口。

操作系统在接收到键盘输入后把消息发送给具有输入焦点（input focus）的窗口。

窗口函数通过捕获WM\_SETFOCUS和WM\_KILLFOCUS消息确定当前窗口是否具有输入焦点。

1. 键盘消息包括哪些消息？

字符消息、按键消息（非系统按键消息、系统按键消息）

1. Alt键与相关的输入键的组合产生的消息应用程序一般做处理吗？

Alt键与相关输入键的组合产生的消息，这些键一般由Windows系统内部直接处理，应用程序不处理。

1. 按键消息的wParam和lParam包含哪些内容？

wParam：包含了识别按下的键的虚键码

lParam：32位的变量

1. WinMain函数的消息循环中包含的TranslateMessage函数的功能是什么？

在WinMain函数的消息循环中包含了TranslateMessage函数，其功能是把按键消息转化为字符消息，但只有当键盘驱动程序把键盘字符映射成ASCII码后才能产生WM\_CHAR消息

1. Windows.h中定义了哪些常用的光标？

Windows中通过光标来指示当前鼠标的位置，在Windows操作系统中预定义了几种光标，并在afxwin.h头文件中加以定义



1. 用户如何自定义光标？怎样加载自定义的光标？

自定义光标保存在扩展名为.cur的文件中

光标名 CURSOR 光标文件(.cur)

采用自定义光标时，需在资源文件中定义光标资源

加载光标资源(常在定义窗口类时进行)

LoadCursor(hThisInst,lpszCursorname)

1. 鼠标消息包含哪些信息？

lParam包含了鼠标位置，低位包含鼠标位置的x坐标值，高位包含鼠标位置的y坐标

wParam包含了指示各种虚键状态的值

1. 在鼠标消息的处理中怎样检测Shift/Ctrl键？

case WM\_LBUTTONDOWN: //按下鼠标左键

if((wParam&MK\_CONTROL)&&(wParam&MK\_SHIFT))

… //Shift和Ctrl键都被按下

break;

case WM\_LBUTTONUP: //释放鼠标左键

…

break; …

case WM\_LBUTTONDOWN: //按下鼠标左键

…

break;

case WM\_LBUTTONUP: //释放鼠标左键

…

break;

…

1. 应用程序如果要对鼠标双击做出响应，应如何处理？怎样修改鼠标双击间隔时间？

Windows系统默认的时间间隔为0.5秒，也可以调用SetDoubleClickTime()重新设定间隔值

要使窗口函数能接收鼠标双击产生的消息在注册窗口类时必须具有CS\_DBLCLKS属性

wndclass.style=CS\_HEADRAW|CS\_VERDRAW|CS\_DBLCLKS;

若窗口不包含上述属性的定义，即使进行了双击操作，该窗口也只能接收到两条WM\_BUTTONDOWN消息或两条WM\_BUTTONUP消息。

1. 分别用什么函数捕获和释放鼠标消息？

捕获鼠标消息，SetCapture(hWnd)：可以向句柄为hWnd的窗口发送所有的鼠标消息

当该窗口不再需要捕获鼠标消息时，应及时调用ReleaseCapture()以释放鼠标，否则，其他窗口无法接收鼠标信息。

**第六次作业**

1、 Windows中的资源包括那些？

加速键、位图、光标、对话框、菜单、字符串 、工具条

2、 菜单有哪些部分组成？

菜单由以下组成部分：

(1)  窗口主菜单条

(2)  下拉式菜单框

(3)  菜单项热键标识

(4)  菜单项加速键标识

(5)  菜单项分隔线

3、 怎样定义菜单？

菜单在资源描述文件中的定义形式为：

选 项 说 明

DISCARDABLE 当不再需要菜单时可丢弃

FIXED 将菜单保存在内存中的固定位置

LOADONCALL 需要时加载菜单

MOVEABLE 菜单在内存中可移动

PRELOAD 立即加载菜单

menuID MENU [,载入特性选项]{菜单项列表}

4、 Popup是什么样的菜单？定义的形式是什么？其中的“&”的作用是什么？

POPUP语句定义弹出式菜单，其形式为：

POPUP "菜单项名" [，选项]；

选 项 说 明

MENUBARBREAK 菜单项纵向分隔标志

CHECKED 显示选中标志

INACTIVE 禁止一个菜单项

GRAYED 禁止并变灰显示

在菜单项名中加入符号“&”，以定义热键

定义弹出式菜单项“编辑”的形式如下：

POPUP “编辑（&E）”；

5、 怎么样定义菜单项？

MENUITEM 用于定义菜单项，其形式为：

MENUITEM "菜单项名" 菜单项标识（ID）[，选项]

6、 菜单的响应是通过什么消息响应的？

通过WM\_COMMAND消息进行处理。

当用户选择了一个菜单项时，Windows会将WM\_COMMAND消息发送给菜单所属的窗口。该消息的wParam参数包含了菜单项的ID值，可以通过此值来区分不同的菜单项；而lParam参数通常为NULL。接收到WM\_COMMAND消息后，窗口可以根据wParam的值，执行相应的操作，例如打开对话框、更新界面、执行特定函数等。

7、 菜单项的ID标识符是不是唯一确定的？

ID为菜单项标识，在WM\_COMMAND消息中字参数wParam中包含选中菜单项的标识。每个菜单项的标识必须唯一，标识值常在头文件中定义。

8、 菜单项中的水平分割符用不用定义ID标识符？

不用

9、 菜单资源的描述是在什么文件中做定义的？

.rc资源文件

10、 菜单资源中的ID标识符对于的数值在什么文件中定义？

.h头文件

11、 怎样加载菜单资源？

在窗口类中加载菜单

WNDCLASSEX wndclass;

wndclass.lpszMenuName=MAKEINTRESOURCE(My\_menu);

在创建窗口时加载菜单

HWND hwnd；HMENU hmenu；

…

hmenu=LoadMenu(hlnstance, MAKEINTRESOURCE(My\_menu))；

hwnd=CreateWindow( …，hmenu, …）；

动态加载菜单

hmenu2=LoadMenu(hlnstance, MAKEINTRESOURCE(My\_menu))；

SetMenu(hwnd,hmenu2)；

…

12、 怎样禁止或激活菜单项？

禁止或激活菜单项

BOOL EnableMenuItem

(HMENU hmenu,

UlNT wIDEnableItem,//被禁止或激活的菜单项标识

UINT dwEnable

)

13、 怎样设置或取消选中标志？

DWORD CheckMenuItem

(

HMENU hmenu,

UINT wIDCheckItem, //菜单项标识

UINT dwCheck //操作标识

)

14、 怎样增加菜单项？并举例理解该使用的函数的参数。

在菜单的尾部增加菜单项

BOOL AppendMenu

( HMENU hmenu,

UINT dwFlags, // 新加入的菜单项类型标识

UINT dwIDNewItem, // 新加入菜单项的ID

LPCTSTR lpNewItem // 新加入的菜单项内容

)

例如在 “文件”菜单的末尾增加一项“关于”：

AppendMenu (hmenu, MF\_ENABLED,IDM\_ABOUT, “关于(&A)”)；

15、 怎样在菜单中插入菜单项？

在菜单中插入菜单项

调用函数InsertMenu 在菜单中插入新的菜单项

BOOL InsertMenu

(HMENU hmenu, //菜单句柄

UINT wPosition, //指定新菜单项插入的位置

UINT dwFlag, //新加入的菜单项的信息

UINT dwIDNweItem, //新加入的菜单项的标识

LPCTSTR lpNewItem //新插入的菜单项的内容

)

例如，在 “文件”菜单的“退出”项之前加入 “打印”菜单项；

InsertMenu ( hmenu,IDM\_EXIT,

MF\_BYCOMMAND |MF\_ENABLED,

IDM\_PRINT,

L"打印（&P）");

16、 怎样删除菜单项？

删除菜单项

调用函数DeleteMenu 删除菜单项：

BOOL DeleteMenu

(

HMENU hmenu,

UINT wPosition, //要删除的菜单项的位置

UINT dwFlag //对wPosition的解释

)

例如，删除“文件”菜单中的“另存为”菜单项：

DeleteMenu(hmenu, IDM\_SAVEAS, MF\_BYCOMMAND)

17、 怎样修改菜单项？

修改菜单项

调用函数ModifyMenu 修改菜单中的某个项；

BOOL ModifyMenu

(HMENU hmenu,

UINT wPosition, //指定需修改的菜单项位置

UINT dwFlag,

UINT dwIDNweItem, //修改后菜单项的标识

LPCTSTR lpNewItem //修改后的菜单项名

)

例如：修改 “文件”菜单中“打开”项为“加载”项：

 ModifyMenu(hmenu,IDM\_OPEN,MF\_BYCOMMAND,

IDM\_LOAD,L"加载(&L) ")；

18、 怎样动态创建菜单？

动态地创建菜单可以使系统资源更加节省，在应用程序中动态创建菜单分两个步骤：

1．调用函数CreateMenu创建空的弹出式菜单

HMENU CreateMenu(void)

2．调用函数AppendMenu或InsertMenu在该菜单中加入菜单项。

19、 理解怎样在资源文件中定义加速键，常用的加速键有哪两种形式？并举例

在资源描述文件中定义加速键资源

加速键资源名 ACCELERATORS

加速键标识（ID），

[类型 ]

[NOINVERT] //使用加速键时，菜单项不高亮显示

[ALT][SHIFT][CONTROL] //组合键的组合方式

常用的加速键有两种形式：

①“^char”,id

例如：“文件”菜单中“保存”项的加速键可定义为：

“^S” , IDM\_SAVE

②nCode, id VIRTKEY

使用虚拟键作为加速键。

例如将F1键定义为“帮助”菜单项的加速键，其ID标识为IDM\_HELP，其形式如下：

VK\_F1,IDM\_HELP,VIRTKEY

20、 为什么要用加速键？

加速键资源是常伴随菜单使用的一种非常有用的资源。加速键是一种用于提高用户界面交互效率的技术。通过定义键盘快捷键，用户可以直接使用键盘上的某些按键来触发应用程序中的特定功能，而无需通过鼠标等其他方式寻找菜单项或按钮。

21、 怎样加载加速键资源？

加载加速键资源

调用函数LoadAccelerators加载加速键资源：

…

HACCEL hAccel；

…

hAccel=LoadAccelerators

(

hInstance, // 当前程序实例句柄

lpAcceIName // lpAccelName为加速键表名

)；

1. 怎样翻译加速键？

翻译加速键

翻译操作经常在应用程序的消息循环中进行：

while(GetMessage(&Msg,NULL,0,0))

if(!TranslateAccelerator(hwnd,hAccel,&Mag))

{ TranslateMessage(&Msg)；

DispatchMessage(&Msg)；

}

函数TranslateAccelerator是翻译操作的核心。其作用是对照加速键表，将相关的按键消息WM\_KEYDOWN和WM\_KEYUP 翻译成WM\_COMMAND 或WM\_SYSCOMMAND消息。其特点是将翻译后的消息直接发往窗口，不在消息队列中等待。

**第七次作业**

1、 什么是位图？

位图是一种数字化的图形表示形式，位图中的每个像素点由位图文件中的一位或多位数据表示。整个位图的信息被细化为每个像素点的属性值。

位图数据量一般较大，为了提高显示刷新速度，位图操作须在内存中进行。

2、 位图的操作过程是什么？

定义位图句柄：HBITMAP hBm;

加载或创建

hBm=LoadBitmap(hInstance, lpszName)

hBm=CreateCompatibleBitmap(hdc,nWidth,nHeight)

选入内存设备环境

SelectObject(hdcmem,hBm);

输出

BOOL BitBlt( HDC hdcDest, //目的设备环境句柄

int XDest,int YDest, int nWidth, int nHeitght, //位图基点及高宽

HDC hdcSrc, int nXSrc,int nYsrc, //源设备环境句柄及位图基点

DWORD dwRop //标识位图显示方式

)

3、 怎样获取位图的尺寸？

调用函数GetObject获取位图的尺寸：

GetObject

(hBitmap, //为位图句柄

sizeof(BITMAP), //BITMAP结构的大小

(LPVOID)&bm //BITMAP结构的地址

)

4、 怎样理解函数hdcmem=CreateCompatibleDC(hdc)？

用于位图操作的系统设备环境为内存设备环境。应用程序首先要通过调用函数CreateCompatibleDC向系统申请获取内存设备环境，此内存设备环境与输出设备的设备环境hdc互相兼容。

1. 怎样理解BOOL BitBlt函数？

BitBlt函数可以将源设备上下文中的矩形区域中的像素数据复制到目标设备上下文中的指定位置。

BOOL BitBlt

( HDC hdcDest, //目的设备环境句柄

int XDest,int YDest, int nWidth, int nHeitght, //位图基点及高宽

HDC hdcSrc, int nXSrc,int nYsrc, //源设备环境句柄及位图基点

DWORD dwRop //标识位图显示方式

)