1. 简述二进制文件与文本文件的区别，并解释有时使用记事本打开文件时，在屏幕上会出现乱码现象？

答：二进制文件与文本文件的区别体现在如下几方面：首先它们保存数据的内容不同：二进制文件顾名思义保存的是信息的二进制表示，计算机内存的信息不进行任何加工，被直接储存于磁盘，所以二进制文件又称为内存“映像文件” 。而文本文件则储存的是计算机所能表示字符的ASCII码值，对于内存即将被保存的信息要进行转换后再保存到磁盘，比如数值10000，保存到文本文件后是字符‘1’‘0’‘0’‘0’‘0’的ASCII码值。此次，为了能按行储存信息，系统自动为文本文件添加行分割符，而二进制文件不存在按行储存的概念。记事本是一款文本编辑器，当使用记事本打开二进制文件时，它把保存的二进制数据理解为字符的ASCII码值来处理，其中有些值可能对应的是ASCII码表中的图形字符或控制字符，所以就造成了乱码的现象。

简述Win32应用程序的执行过程：

Windows操作系统为每一个Win32应用程序内部建立一个消息队列，应用程序通过该消息队列来感知操作系统循环传递来的具体消息，不断地从消息队列中提取消息，并通过处理具体的自己感兴趣的消息来实现对用户I/O操作的响应，应用程序本身并不直接访问I/O设备，我们把应用程序对消息的处理又称之为消息响应。Win32应用程序的执行就是一个不断、反复处理消息的过程，直到应用程序接受到退出消息才结束程序的执行。

应用编译预处理命令解决类的头文件重复包含的问题（即：重复定义）

在一个应用程序中，可能出现多个源程序文件中同时包含某一个类的头文件的情况，可以通过把头文件的定义放在编译预处理命令#IFDEF……….#ENDIF或#IFNDEF………….#ENDIF之间来解决。

简述并画图说明C++编译器编译、连接生成exe的全过程。

C++编译器编译程序的最小单位是cpp文件，编译器为每个cpp文件产生一个obj文件，它们是一一对应的关系；\*.h的头文件并不单独进行编译，它们是通过编译预处理指令#include被包含在\*.Cpp的文件中与cpp文件一起进行编译。最后通过连接器把\*.obj的文件连同系统库文件连接在一起生成文件扩展名为exe的可执行文件。

1. 简述C++进行异常处理的原理，并举例说明.

在一个大型软件中，由于函数之间有着明确的分工和复杂的调用关系，发现错误的函数往往不具备处理错误的能力。这时它就引发一个异常，希望它的调用者捕获这个异常并处理这个错误。如果调用者也不能处理这个错误，还可以继续传递给上级调用者去处理，这种传播可以一直继续到异常被处理为止。如果程序始终没有处理这个异常，最终它会被传到C++运行系统那里，运行系统捕获异常后通常只是简单终止这个程序。C++的异常处理机制使得异常的引发与处理不必再同一函数中，这样底层的函数可以专注于解决具体问题，而不必过多的考虑对异常的处理。上层的调用者可以在适当的位置设计对不同类型异常的处理。

下面是异常处理的具体示例：

#include <iostream>

using namespace std

int Div(int,int);

void main()

{

try

{

cout<<”3/2”<<Div(3,2)<<endl;

cout<<”8/0”<<Div(8,0)<<endl;

cout<<”7/1”<<Div(7,1)<<endl;

}

catch(int)

{

cout<<”except of deviding zero”;

}

cout<<”this is OK”<<endl;

}

int Div(int x,int y)

{

if(y==0)

Thow y;

return x/y

}

1. DOS应用程序与WINDOWS应用程序的区别：

DOS操作系统是单用户操作系统：即某一时刻只能有一个应用程序运行，所以决定了应用程序在运行期间对系统资源是独占的，不存在多进程之间对资源的争强、仲裁的情况；而WINDOWS操作系统正相反。所以DOS应用程序是面向过程的，应用程序可以对系统资源（包括I/O设备）进行直接的编程访问（越过操作系统），而WINDOWS应用程序是面向事件的，应用程序对系统资源不可直接编程访问，必须经过操作系统的系统调用（即所谓的API函数）进行。应用程序通过消息的传递机制来与操作系统进行通讯，而操作系统通过调用应用程序的事件处理过程来实现（即所谓的窗口过程）。另外，WINDOWS操作系统支持面向对象的应用程序的编程方法，而DOS操作系统不支持。

在讲本程序的消息循环之前，我想先谈一下Dos与Windows驱动机制的区别:   
  
DOS程序主要使用顺序的，过程驱动的程序设计方法。顺序的，过程驱动的程序有一个明显的开始，明显的过程及一个明显的结束，因此程序能直接控制程序事件或过程的顺序。虽然在顺序的过程驱动的程序中也有很多处理异常的方法，但这样的异常处理也仍然是顺序的，过程驱动的结构。   
  
而Windows的驱动方式是事件驱动，就是不由事件的顺序来控制，而是由事件的发生来控制，所有的事件是无序的，所为一个程序员，在你编写程序时，你并不知道用户先按哪个按纽，也不知道程序先触发哪个消息。你的任务就是对正在开发的应用程序要发出或要接收的消息进行排序和管理。事件驱动程序设计是密切围绕消息的产生与处理而展开的，一条消息是关于发生的事件的消息。

1. 简述MFC应用程序的执行过程：

MFC应用程序的执行过程与win32应用程序的执行过程是有所区别的。

由于每个MFC应用程序均有一个由CWinApp派生的子类对象,程序必须的首先构造该对象，然后才调用主函数(\_tWinmain及AfxWinmain),完成窗口的创建、注册后转而调用InitInstance；根据多态性原理，如果子类重写了该函数，系统就调用调用子类的InitInstance，否则调用父类的InitInstance。所以，对于MFC程序而言，程序的运行好似从InitInstance开始的，所以程序员可以把一些初始化工作放在InitInstance函数中完成。