

# 第三章 Windows 应用程序

- 基础知识

Windows的  
程序设计语言

VC  
VB  
VJ

都是“面向对象”  
的程序设计语言

对象是Windows  
的规范部件

窗口  
菜单  
按钮  
对话框  
程序模块

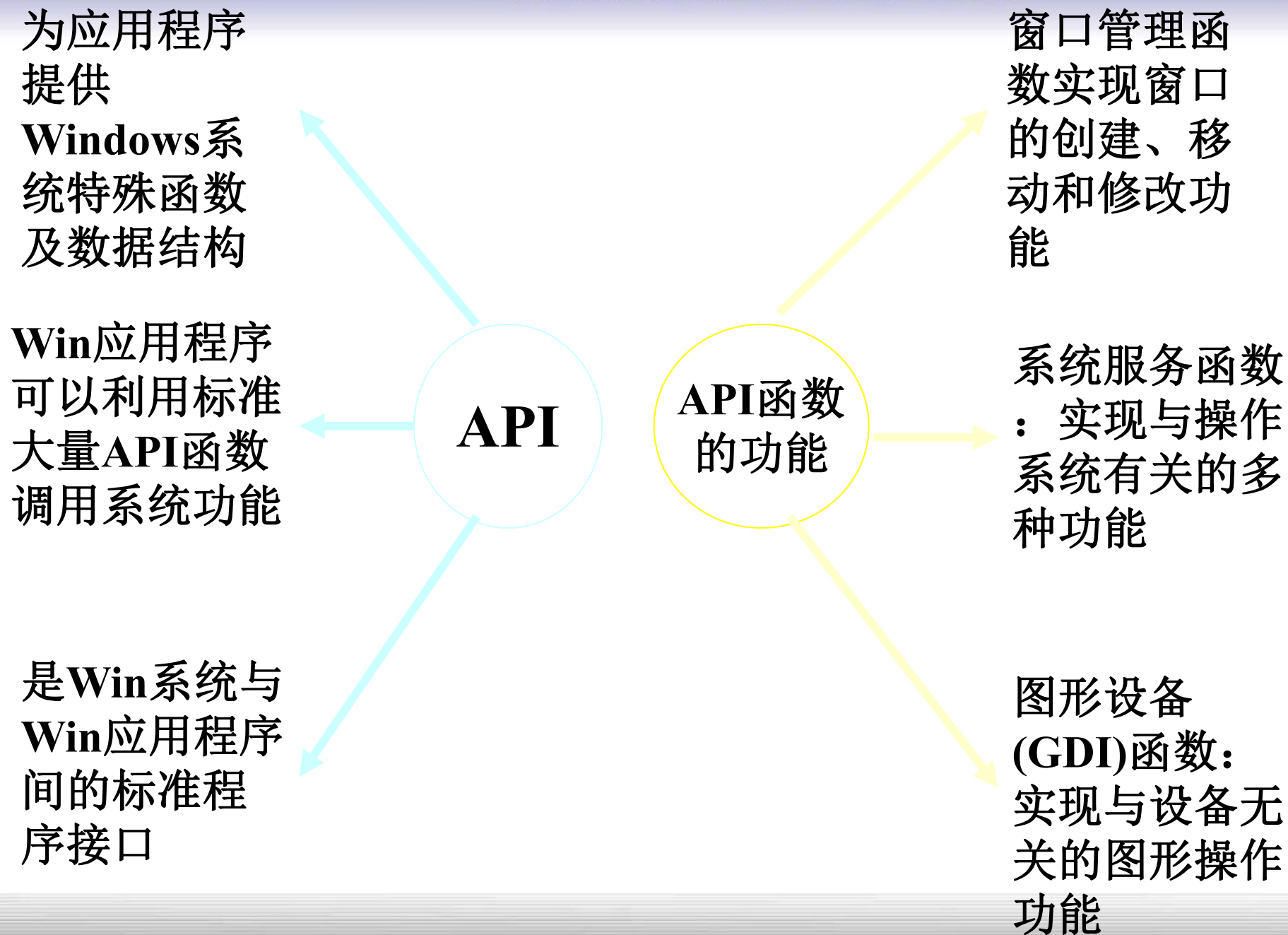
编写Windows  
程序相当一部分工  
作是在创建对象和  
为对象属性赋值

对象特征：具有规范形态和操作模式

编程方法 { 传统编写法-->API  
交互式方法-->MFC



采用交互式方法时，可视化开发平台给出了许多选用的对象，程序员可选择所需对象并确定其属性，由此搭建起应用程序的“大框架”，并可根据需要进一步编写必要的细节代码段，最后构成完整的应用程序



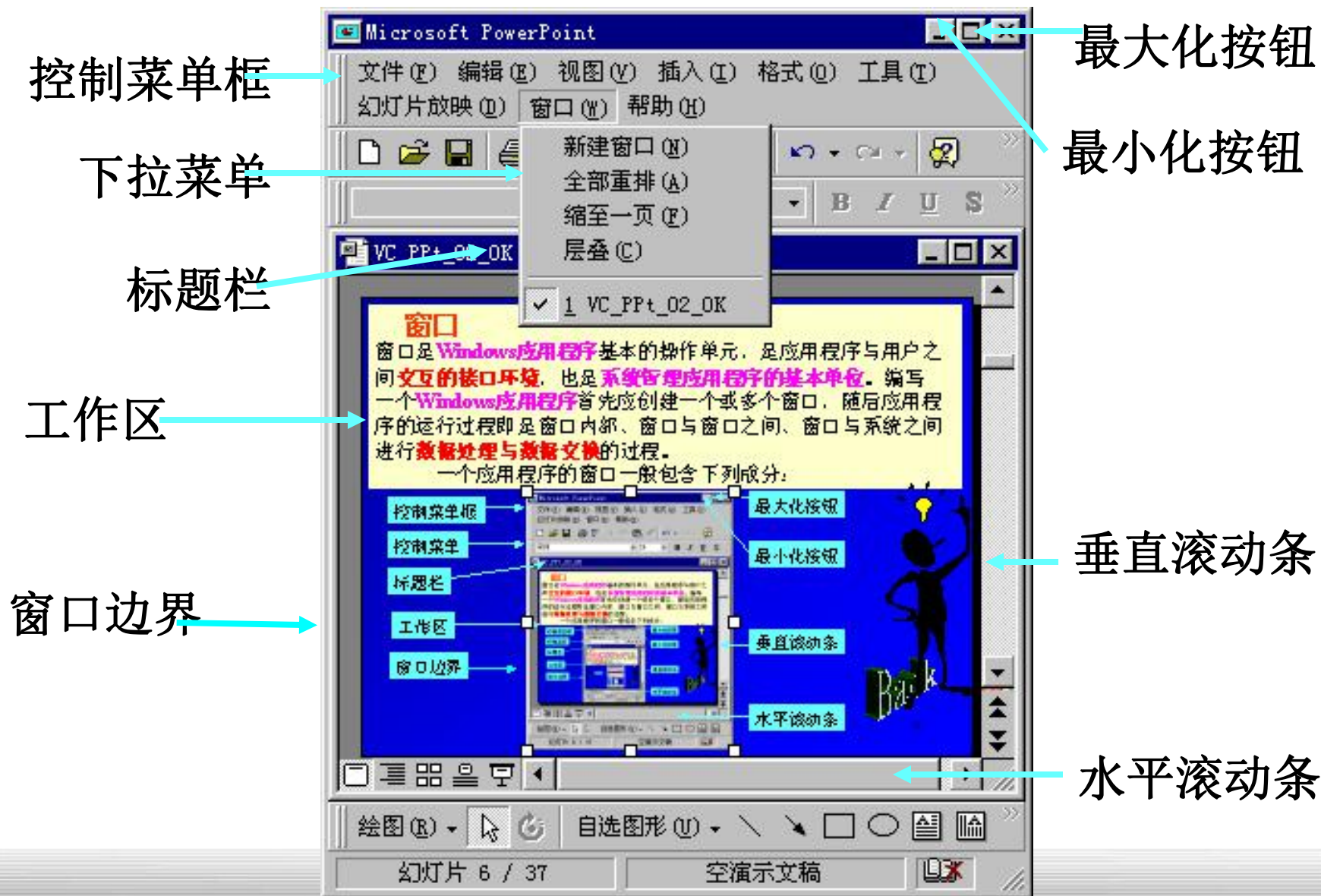
利用Windows API函数编写Windows应用程序必须首先了解以下内容：

- (1) 窗口的概念
- (2) 事件驱动的概念
- (3) 句柄
- (4) 消息



# 1. 窗口

一个应用程序的窗口一般包含下列组成部分：







系统管理  
应用程序  
的基本单位

应用程序与用  
户之间交互的  
接口环境

Win应用程序  
基本的操作单元

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。

消息是描述事件发生的信息  
(如按下鼠标或键盘)

## 2.事件驱动

**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。

### 3. 句柄

句柄是一个4字节长的数值，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

窗口  
按钮  
图标  
滚动条  
输出设备  
控制  
文件

#### 常用句柄类型及其说明

HWND  
HBITMAP  
HICON  
HMENU  
HFILE  
HINSTANCE

窗口句柄  
位图句柄  
图标句柄  
菜单句柄  
文件句柄  
当前实例句柄

HDC  
HCURSOR  
HFONT  
HPEN  
HBRUSH

设备环境句柄  
光标句柄  
字体句柄  
画笔句柄  
画刷句柄



## 4. 消息

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

附加信息与具体消息号的值有关，在Win中消息用结构体MSG表示

```
typedef struct tagMSG
```

```
{HWND  hwnd;  窗口句柄，为null，则可检索所有驻留在消息队列中的消息
```

```
  UINT  message;消息值，由Windows.h头文件中的宏定义来标识
```

```
  WPARAM wParam;          包含有关消息的附加信息,不同消息其值有所不同
```

```
  LPARAM lParam;
```

```
  DWORD time; 指定消息送至队列的时间
```

```
  POINT pt;指定消息发送时屏幕光标的位置，其数据类型为
```

```
  体  
}MSG;
```

```
typedef struct  
tagPOINT  
{LONG x;  
  LONG y;  
} POINT;
```

VC中存在几种系统定义的消息分类，不同的前缀符号经常用于消息宏识别消息附属的分类，系统定义的消息宏前缀如下：

<b>BM</b>	表示按钮控制消息
<b>CB</b>	表示组合框控制消息
<b>DM</b>	表示默认下压式按钮控制消息
<b>EM</b>	表示编辑控制消息
<b>LB</b>	表示列表框控制消息
<b>SBM</b>	表示滚动条控制消息
<b>WM</b>	表示窗口消息

**Windows编  
程常用消息**

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

## 二、Windows应用程序常用消息

### 1. WM\_LBUTTONDOWN: 产生单击鼠标左键的消息

1Param { 低字节包含当前光标的X坐标值  
高字节包含当前光标的Y坐标值

wParam包含一  
整数值以标识鼠  
标键的按下状态

→

MK_LBUTTON	按下鼠标左键
MK_MBUTTON	按下鼠标中键
MK_RBUTTON	按下鼠标右键

此外，相似的消息还有：

- WM\_LBUTTONUP: 放开鼠标左键时产生；
- WM\_RBUTTONDOWN: 单击鼠标右键时产生；
- WM\_RBUTTONUP: 放开鼠标右键时产生；
- WM\_LBUTTONDBLCLK: 双击鼠标左键时产生；
- WM\_RBUTTONDBLCLK: 双击鼠标右键时产生。

## 2. WM\_KEYDOWN: 按下一个非系统键时产生的消息

系统键是指实现系统操作的组合键，例如Alt与某个功能键的组合以实现系统菜单操作等。

如F1的虚拟键码在Windows.h文件中定义为VK\_F1

wParam: 按下键的虚拟键码，用以标识按下或释放的键

lParam: 记录了按键的重复次数、扫描码、转移代码、先前键的状态等信息。

相似的消息还有WM\_KEYUP, 在放开非系统键时产生

## 3. WM\_CHAR: 按下一个非系统键时产生的消息

wParam 为按键的ASCII码

lParam 与WM\_KEYDOWN的相同

- 4. WM\_CREATE:** 由CreateWindow函数发出的消息  
wParam: 未用  
lParam: 包含一个指向CREATESTRUCT数据结构的指针
- 5. WM\_CLOSE:** 关闭窗口时产生的消息  
wParam和lParam均未用。
- 6. WM\_DESTROY:** 由DestroyWindow函数发出的消息  
wParam和lParam均未用。



**7. WM\_QUIT:** 由PostQuitMessage函数发出的消息  
退出应用程序时发出的消息

**wParam:** 含退出代码,标识程序退出运行时的有关信息

**lParam:** 未用

## 8. WM\_PAINT

用户区移动或显示

用户窗口改变大小

程序通过滚动条滚动窗  
口

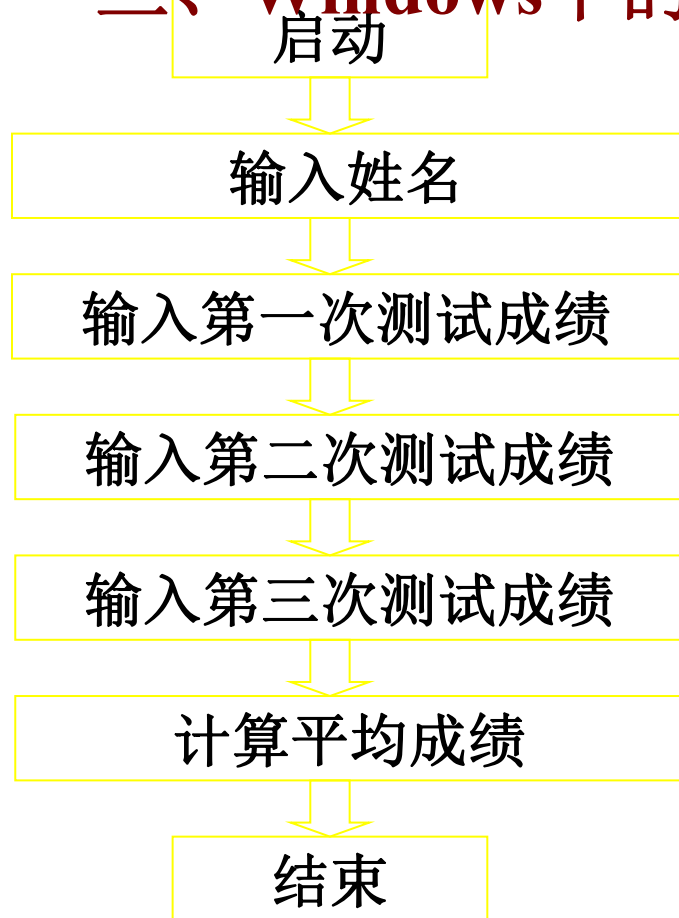
下拉式菜单关闭并需要恢复被覆盖的部分

Windows清除对话框等对象, 并需要恢复被覆盖的部分

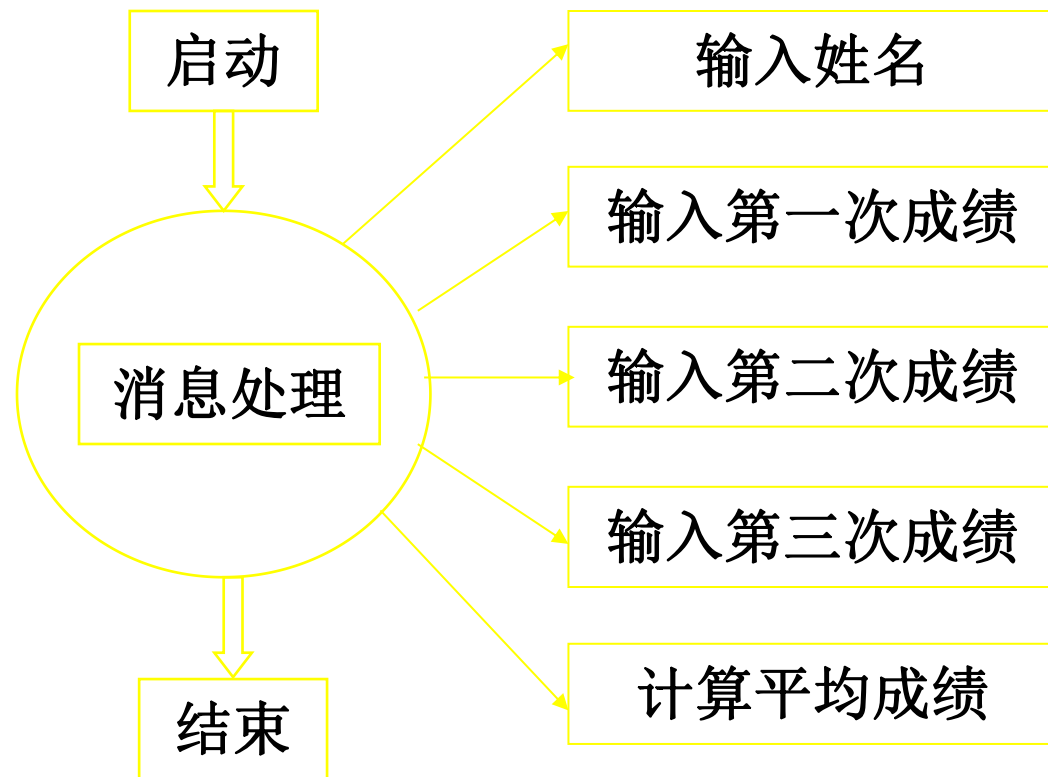


均产生  
WM\_PAINT  
消息

### 三、Windows中的事件驱动程序设计



过程驱动方法计算平均成绩

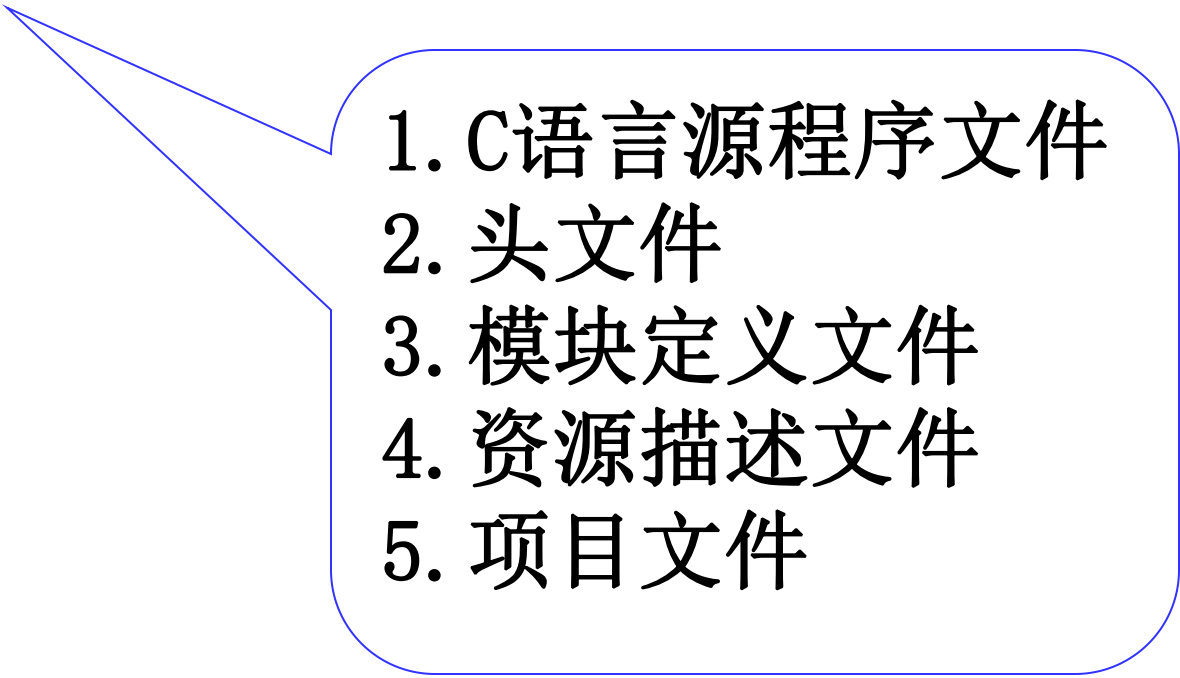


事件驱动方法计算平均成绩

## 四、Windows应用程序组成及编程步骤

### 1. 应用程序的组成

一个完整的  
**Windows**应用程序  
通常由五种类型的  
文件组成。

- 
1. C语言源程序文件
  2. 头文件
  3. 模块定义文件
  4. 资源描述文件
  5. 项目文件

## 2. 源程序组成结构

1. 所有应用程序的入口，类似Main函数，
2. 完成一系列的定義和初始化，并产生消息循环

**Windows  
应用程序**

入口函数WinMain

窗口函数WndProc

构成基  
本框架

包含各种  
数据类型、  
数据结构  
与函数等

**WinMain和WndProc是  
Windows应用程序的主体**

## WinMain函数

功能

注册窗口类，建立窗口及执行必要的初始化  
进入消息循环，根据接受的消息调用相应的处理过程  
当消息循环检索到WM\_QUIT时终止程序运行

三个基本的组成部分：函数说明、初始化和消息循环

### WinMain函数说明

WinMain函数的说明如下：

```
int WINAPI WinMain  
( HINSTANCE hThisInst,  
  HINSTANCE hPrevInst,  
  LPSTR lpszCmdLine,  
  Int nCmdShow  
)
```

注意！Win是多任务管理的，同一应用程序的多个窗口可能会同时存在，Win系统对每个窗口的执行称为一个实例，并用一个实例句柄来唯一标识

// 应用程序当前实例句柄  
// 应用程序其他实例句柄  
// 指向程序命令行参数的指针  
// 应用程序开始执行时窗口显示方式的整数值



## (2) 初始化

初  
始  
化

窗口类的定义：定义窗口的形式与功能

**LoadIcon**

**LoadCursor**

**GetStockObject**

窗口类的注册：窗口类必须先注册后使用 **RegisterClass**

创建窗口实例 **CreateWindow**

显示窗口 **ShowWindow , UpdateWindow**

### i. 窗口类定义

通过给窗口类数据结构**WNDCLASS**赋值完成,该数据结构中包含窗口类的各种属性。窗口类定义常用以下函数：

**LoadIcon**的作用是在应用程序中加载一个窗口图标。其原型为：  
**HICON LoadIcon(HINSTANCE hInstance, LPCTSTR lpIconName)**

图标资源所在的模块句柄，  
**NULL**则使用系统预定义图标

图标资源名或系统预定义图标标识名

LoadCursor的作用是在应用程序中加载一个窗口光标

HCURSOR LoadCursor (HINSTANCE hInstance,  
LPCTSTR lpCursorName)

光标资源所在的模  
块句柄，NULL则使  
用系统预定义光标

光标资源名或系统  
预定义光标标识名

应用程序调用函数GetStockObject获取系统提供的背景刷

HBRUSH GetStockObject (int nBrush);

Win系统本身提供部分预定义的窗口类，程序员也可以自定义窗口类，窗口类必须先注册后使用。窗口类的注册由函数RegisterClass()实现。

`RegisterClass (&wndclass);` //wndclass为窗口类结构

RegisterClass函数的返回为布尔值，注册成功则返回真

### iii. 创建窗口实例

创建一个窗口类的实例由函数CreateWindow（）实现

函数原型如下：

HWND Create Window

```
( LPCTSTR lpszClassName, // 窗口类名
  LPCTSTR lpszTitle,      // 窗口标题名
  DWORD dwStyle,          // 创建窗口的样式
  int x, y,               // 窗口左上角坐标
  int nWidth, nHeight,    // 窗口宽度和度高
  HWND hwndParent,        // 该窗口的父窗口句柄
  HMENU hMenu,            // 窗口主菜单句柄
  HINSTANCE hInstance,    // 创建窗口的应用程序当前句柄
  LPVOID lpParam          // 指向一个传递给窗口的参数值的指
针
)
```

## 常用窗口样式

标识	说明
<b>WS_BORDER</b>	创建一带边框的窗口
<b>WS_CAPTION</b>	创建一带标题栏的窗口
<b>WS_VSCROLL</b>	创建一带垂直滚动条的窗口
<b>WS_MAXIMIZEBOX</b>	创建一带最大化框的窗口
<b>WS_MAXIMIZE</b>	创建一最大尺寸的窗口
<b>WS_MINIMIZEBOX</b>	创建一带最小化框的窗口
<b>WS_MINIMIZE</b>	创建一最小尺寸的窗口
<b>WS_OVERLAPPED</b>	创建一带边框和标题的窗口
<b>WS_OVERLAPPEDWINDOW</b>	创建一带边框、标题栏、系统菜单及最大、最小化框的窗口
<b>WS_POPUP</b>	创建一弹出式窗口
<b>WS_POPUPWINDOW</b>	创建一带边框和系统菜单的弹出式窗口
<b>WS_SYSMENU</b>	创建一带系统菜单的窗口
<b>WS_HSCROLL</b>	创建一带水平滚动条的菜单



## iv 显示窗口

窗口类的显示由ShowWindow和UpdateWindow函数实现。应用程序调用ShowWindow函数在屏幕上显示窗口

ShowWindow(hwnd, nCmdshow); //nCmdshow为窗口显示形式标识



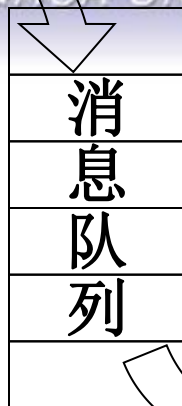
<b>SW_HIDE</b>	隐藏窗口
<b>SW_SHOWNORMAL</b>	显示并激活窗口
<b>SW_SHOWMINIMIZE</b>	显示并最小化窗口
<b>SW_SHOWMAXIMIZE</b>	显示并最大化窗口
<b>SW_SHOWNOACTIVE</b>	显示但不激活窗口
<b>SW_RESTORE</b>	恢复窗口的原来位置及尺寸

显示窗口后，应用程序调用UpdateWindow更新并绘制用户区，并发出WM\_PAINT消息。

UpdateWindow(hwnd);

### (3) 消息循环

Windows将  
产生的消息



将消息传递给  
窗口函数的相  
应过程处理

WinMain函数

消息循环的常见格式如下:

**MSG Msg;**

...

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

```
{ TranslateMessage (&Msg);
```

```
  DispatchMessage (&Msg); }
```

将消息的虚拟键  
转换为字符信息

将消息传送到  
指定窗口函数

从消息队列中读取  
一条消息，并将消  
息放在MSG结构中

返回零值，即检索  
到WM\_QUIT消息，程  
序结束循环并退出

其中函数GetMessage形式为:

**GetMessage**

(lpMSG,

//指向MSG结构的指针

hwnd,

nMsgFilterMin, //用于消息过滤的最小消息号值

nMsgFilterMax //用于消息过滤的最大消息号值

)

### 3. 窗口函数**WndProc**

**WndProc**

定义了应用程序对接收到的不同消息的响应

包含了对各种可能接收到的消息的处理过程

**WndProc**函数由一个或多个**switch**语句组成。每一条**case**语句对应一种消息，当应用程序接收到一个消息时，相应的**case**语句被激活并执行相应的响应程序模块。

窗口函数的一般形式如下:

TSINGHUA UNIVERSITY PRESS

```
LRESULT CALLBACK WndProc (HWND hwnd, UINT message,  
                           WPARAM wParam, LPARAM lParam  
)
```

```
{ ...  
  switch (message) 在消息处理程序段中一般都有对  
  { case ...      WM_DESTROY的处理,该消息是关闭窗口时发出  
    ...          的。它向应用程序发出WM_QUIT消息, 请求退  
    break;       出处理函数:  
    ...          void PostQuitMessage(int nExitCode)  
                  //nExitCode为应用程序的退出  
  
    case WM_DESTROY: 代码  
      PostQuitMessage (0);  
    default:  
      return DefWindowProc (hwnd, message, wParam, lParam);  
  }  
  return (0);
```

为未定义处理过程的消息提供默认的处理

## 4. 数据类型

在Windows.h中定义了Windows 应用程序中包含种类繁多的数据类型

数据类型	说 明
<b>WORD</b>	16 位无符号整数
<b>LONG</b>	32 位有符号整数
<b>DWORD</b>	32 位无符号整数
<b>HANDLE</b>	句柄
<b>UINT</b>	32 位无符号整数
<b>BOOL</b>	布尔值
<b>LPTSTR</b>	指向字符串的 32 位指针
<b>LPCTSTR</b>	指向字符串常量的 32 位指针



## 5. 一些重要的数据结构

几种重要的结构

**MSG:** 包含一个消息的全部信息，是消息发送的格式

**WNDCLASS:** 包含一个窗口类的全部信息及属性

**POINT:** 定义了屏幕上或窗口中的一个点的X和 Y 坐标

**RECT:** 定义了一个矩形区域及其左上角和右下角的坐标

## 五、应用程序举例

TSINGHUA UNIVERSITY



【例3-1】创建应用程序框架。本例的目的  
在于说明创建Windows应用程序的方法及过程

```
#include<windows.h>           //包含应用程序中所需的数据类型和数据结构的定义
```

```
LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM); //窗口函数说明
```

```
//----- 以下初始化窗口类 -----
```

```
int WINAPI WinMain(HINSTANCE hInstance,  
    HINSTANCE hPrevInst, LPSTR lpszCmdLine, int nCmdShow)  
{  
    HWND hwnd ;  
    MSG Msg ;  
    WNDCLASS wndclass ;  
    char lpszClassName[] = "窗口";    //窗口类名  
    char lpszTitle[] = "My_Windows";  //窗口标题名
```

## //窗口类的定义

```
wndclass.style=0;                //窗口类型为默认类型
wndclass.lpfnWndProc=WndProc;    //定义窗口处理函数
wndclass.cbClsExtra=0;          //窗口类无扩展
wndclass.cbWndExtra=0;          //窗口实例无扩展
wndclass.hInstance=hInstance;   //当前实例句柄
wndclass.hIcon=LoadIcon(NULL, IDI_APPLICATION);
                                //窗口的最小化图标为默认图标
wndclass.hCursor=LoadCursor(NULL, IDC_ARROW);
                                //窗口采用箭头光标
wndclass.hbrBackground=GetStockObject(WHITE_BRUSH);
                                //窗口背景为白色
wndclass.lpszMenuName=NULL;     //窗口中无菜单
wndclass.lpszClassName=lpszClassName;
                                //窗口类名为"窗口"
```

```
if (!RegisterClass (&wndclass)) //如果注册失败则发出警告
{
    MessageBeep (0) ;
    return FALSE ;
}

//----- 创建窗口 -----
hwnd=CreateWindow
(
    lpzClassName,           //窗口类名
    lpzTitle,               //窗口实例的标题名
    WS_OVERLAPPEDWINDOW,   //窗口的风格
    CW_USEDEFAULT,          //窗口左上角坐标为默认值
    CW_USEDEFAULT,
    CW_USEDEFAULT,
    CW_USEDEFAULT,,
    NULL,                   //窗口的高和宽为默认值
    NULL,                   //此窗口无父窗口
    NULL,                   //此窗口无主菜单
    hInstance,              //创建此窗口的应用程序的当前句柄
    NULL,                   //不使用该值
);
```

```
ShowWindow( hwnd, nCmdShow) ;
```

```
//----- 绘制用户区 -----
```

```
UpdateWindow(hwnd) ;
```

```
//----- 消息循环 -----
```

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

```
{
```

```
    TranslateMessage( &Msg) ;
```

```
    DispatchMessage( &Msg) ;
```

```
}
```

```
return Msg.wParam;
```

```
    回系统
```

```
}
```

//消息循环结束即程序终止时将信息返回系统

## /窗口函数

```
LRESULT CALLBACK WndProc
```

```
(    HWND hwnd,  
    UINT message,  
    WPARAM wParam,  
    LPARAM lParam  
)
```



调用PostQuitMessage  
发出WM\_QUIT消息

```
{    switch(message)  
    {    case WM_DESTROY:  
        PostQuitMessage(0);  
    default:    //默认时采用系统消息默认处理函数  
        return  
    }  
    DefWindowProc(hwnd, message, wParam, lParam);  
}  
return(0);  
}
```