

第1章 Windows编程基础

- C++语言基础知识
- Windows编程概念
- Windows编程方法
- Windows编程特点

C++语言基础

- 数据类型
- 常量、变量与指针
- 控制语句
- 函数与调用
- 类与对象
- 类继承与派生
- 函数与操作符重载
- 多态性与虚函数

数据类型(1)

■ 基本类型

✓ char、int、float、double、bool

■ 派生类型

✓ short、long、signed、unsigned

■ 枚举类型

✓ enum <类型名> { <枚举值表> }

数据类型 (2)

■ 数组类型

- ✓ 一维数组与多维数组
- ✓ 字符串 (字符数组+\0)

■ 结构类型

- ✓ struct <类型名> { <成员表> }

数据类型 (3)

■ 自定义类型 (Win32类型)

数据类型	说明	数据类型	说明
BITMAP	位图结构	MSG	消息结构
LOGBRUSH	画刷结构	POINT	点结构
LOGFONT	字体结构	RECT	矩形结构
LOGPEN	画笔结构	WNDCLASS	窗口类结构
COLORREF	颜色值	LRESULT	回调返回值

常量

■ 整型常量

- ✓ 十进制表示(20)、八进制表示(024)、十六进制表示(0x14)

■ 浮点型常量

- ✓ 科学表示法(314e-2)

■ 字符型常量

- ✓ 转义字符(\r、\n、\t、\024)

变量

■ 匈牙利表示法(前缀)

前缀	数据类型	说明
ch	char	字符
b	bool	布尔值
n	int	整数
w	word	16位无符号数
h	handle	窗口对象句柄
lpsz	LPTSTR	字符串的32位指针

指针与引用

■ 指针

- ✓ 指针变量存储对象的地址(*与&)
- ✓ 初始化时需要分配空间
- ✓ 指针可被同类对象左值初始化

■ 引用

- ✓ 引用是被引用对象的别名
- ✓ 对引用的修改是对被引用对象的修改

基本运算

■ 运算符与表达式

✓ $a*b+c$

■ 运算类型

✓ 赋值运算、算术运算、关系运算、逻辑运算、位运算

控制语句(1)

■ 条件语句

✓ if语句

```
if(expression) { statement 1; }  
else { statement 2; }
```

✓ switch语句

```
switch(expression)  
{ case constant: statement 1; break;  
  .....  
  default: statement n; }
```

控制语句(2)

■ 循环语句

✓ for语句

```
for(expression 1;expression 2;expression  
3)  
{ statement; }
```

✓ while语句

```
while(expression) { statement; }
```

✓ do...while语句

```
do{ statement; }while(expression)
```

控制语句(3)

■ 转移语句

- ✓ break - 中止循环或switch
- ✓ continue - 中止后续操作，返回循环
- ✓ goto - 跳转到指定语句
- ✓ return - 中止函数执行，返回调用函数

函数与调用

■ 函数定义

类型标识符 函数名(形参列表)

{ 函数体 }

■ 函数调用

函数名(实参列表)

or 变量名=函数名(实参列表)

与C语言相比，C++语言的最大特点是

- ☐ A 支持图形用户界面
- ☒ B 增加面向对象特征
- ☐ C 改进消息循环机制
- ☐ D 完善线程处理模式

面向对象和Windows编程(1)

■ 面向对象编程

- ✓ 在类中封装数据与处理函数，类的实例称为对象
- ✓ 可维护、易修改、可重用

面向对象和Windows编程(2)

■ 封装性

- ✓ 将逻辑上相关数据相互联系，数据访问仅通过已定义接口

■ 继承性

- ✓ 有联系类层次关系模型，通过添加、修改等操作实现重用

■ 多态性

- ✓ 允许不同类对象对同一消息作出响应，实现“一种接口、多种方法”

类与对象(1)

- 类(class)将数据和函数封装起来
- 类中包含2种成员：数据成员和函数成员，函数成员可访问数据成员

```
class 类名称
{
    type variables; //数据成员
public:
    type functions; //函数成员
}
```

类与对象 (2)

- 类成员可定义为3种类型：
 - ✓ 私有类型 (private)：只允许类本身声明的函数访问
 - ✓ 公有类型 (public)：任何外部函数都能访问
 - ✓ 保护类型 (protected)：只有派生类中函数能访问

类与对象(3)

- 对象(object)是类的实例
- 对象的定义方法:
 - ✓ 类名 对象名
 - ✓ 类名 *p=new 类名
- 对象的使用方法:
 - ✓ 对象名. 成员函数名(数据成员名)
 - ✓ 对象名->成员函数名(数据成员名)

类与对象(4)

例1-1

■ 类与对象举例

```
//声明类example
class example
{
    int i;
public:
    int j;
    void input();
};
void example::input()
{
    i=10;
}
```

```
//声明类对象
example obj;
example *pp=new example;
```

```
//使用类对象
obj.j=12;
pp->j=12;
obj.input();
pp->input();
obj.i=100;
```

类与对象 (5)

■ 构造函数 (constructor)

- ✓ 初始化对象或分配内存。构造函数名与类名相同，每个类可有多个构造函数，编译器通过参数识别

■ 析构函数 (destructor)

- ✓ 析构函数名是类名前加“~”。每个类仅一个析构函数，无参数，不返回值

类与对象 (6)

- 类中定义的函数是内联函数，类外定义的函数是非内联函数
- 编译器在调用内联函数处装入函数代码，影响代码大小和执行速度
- 较短函数定义在类中，较长函数定义在类外
- `Inline`用于强制定义内联函数

类与对象(7)

- 友元函数可访问类中的私有和保护型数据

```
class 类名称
{
    type variables;           //数据成员
public:
    friend type functions;    //友元函数
}
```

类继承(1)

- C++提供类继承机制，通过增加、修改类成员来扩充类
- 被继承的类称为基类(base)，继承的类称为派生类(derived)
- 派生类的声明方式：
class 派生类名称:派生方式 基类名称
- 派生方式包括public、protected与private

类继承(2)

- 无论哪种派生方式，基类的私有成员不能被派生类继承，保护和公有成员可被派生类继承

派生方式	protected	public
私有派生(private)	private	private
保护派生(protected)	protected	protected
公有派生(public)	protected	public

类继承(3)

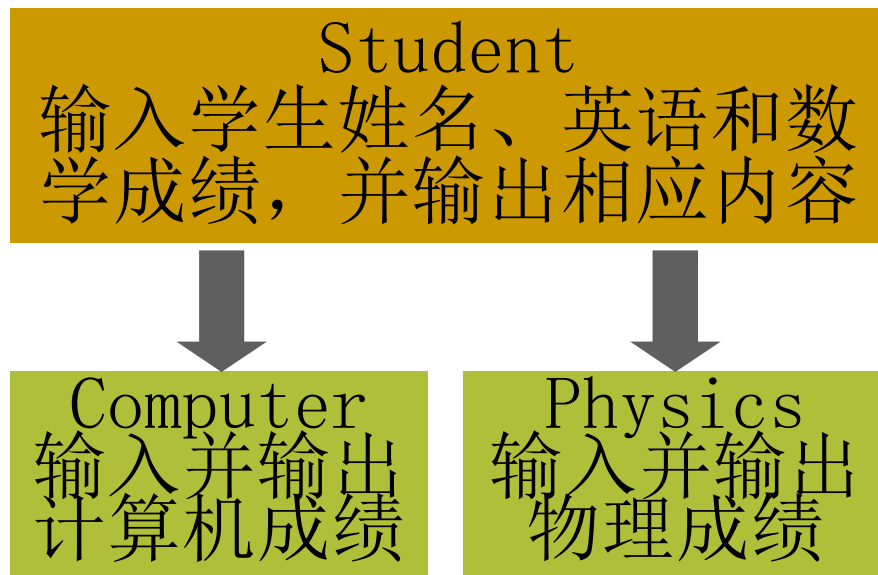
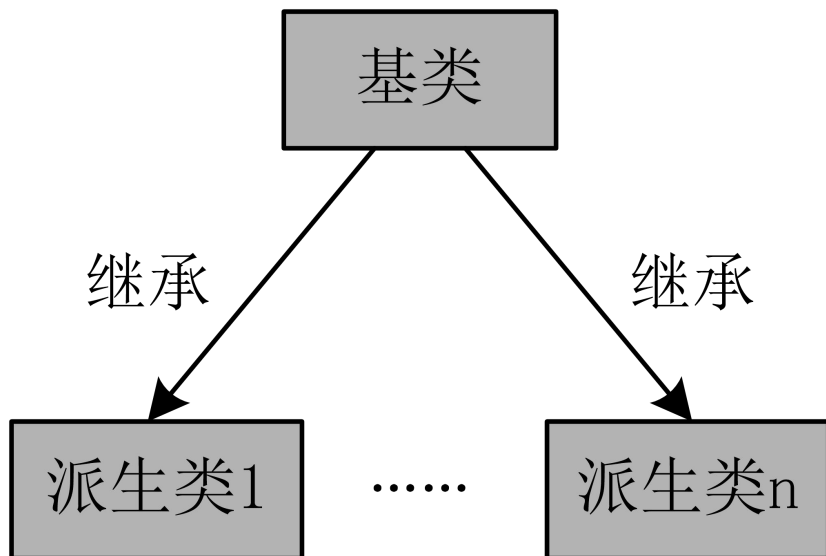
■ 基类与派生类举例

```
class parent                                //基类
{ private: int a;
  protected: int b;
  public: void change() {a++; b++; }
};
class son: public parent                    //派生类
{ private: int c;
  public: void modify() {b++; c++; }
};
```

类继承(4)

例1-2

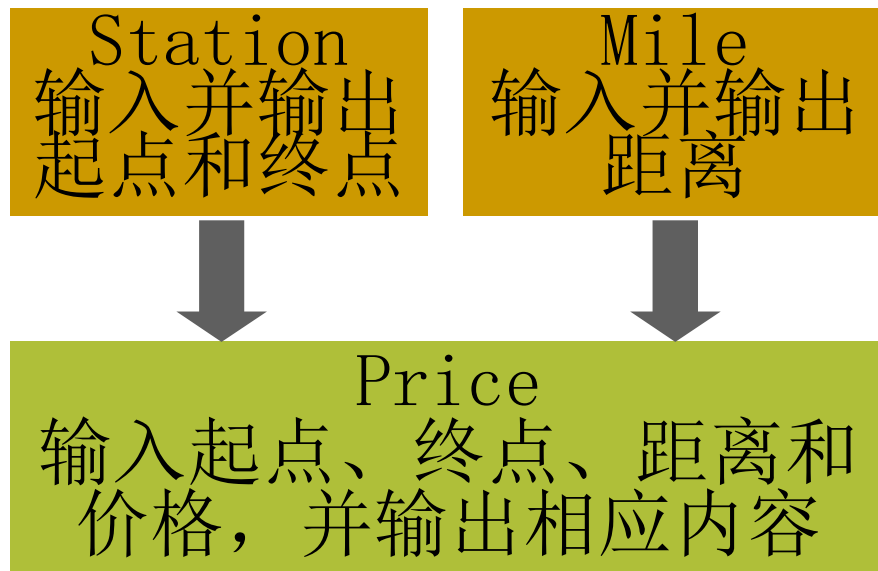
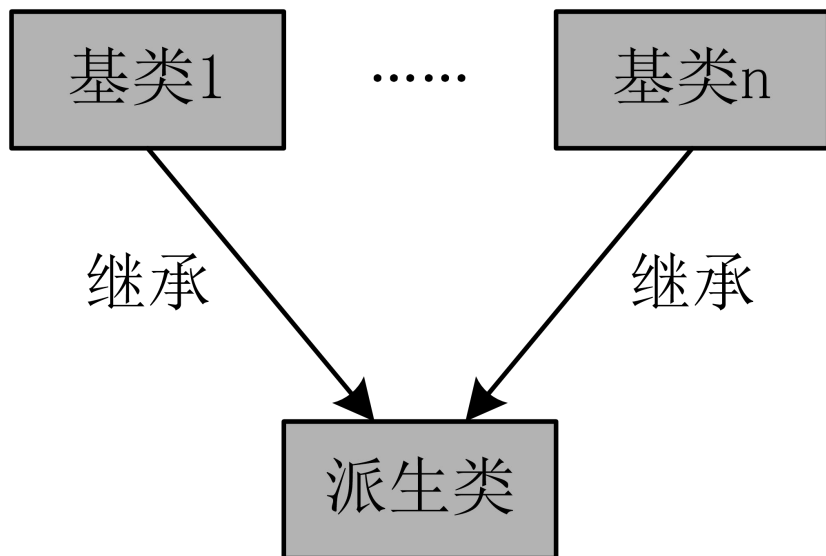
■ 多重派生



类继承(5)

例1-3

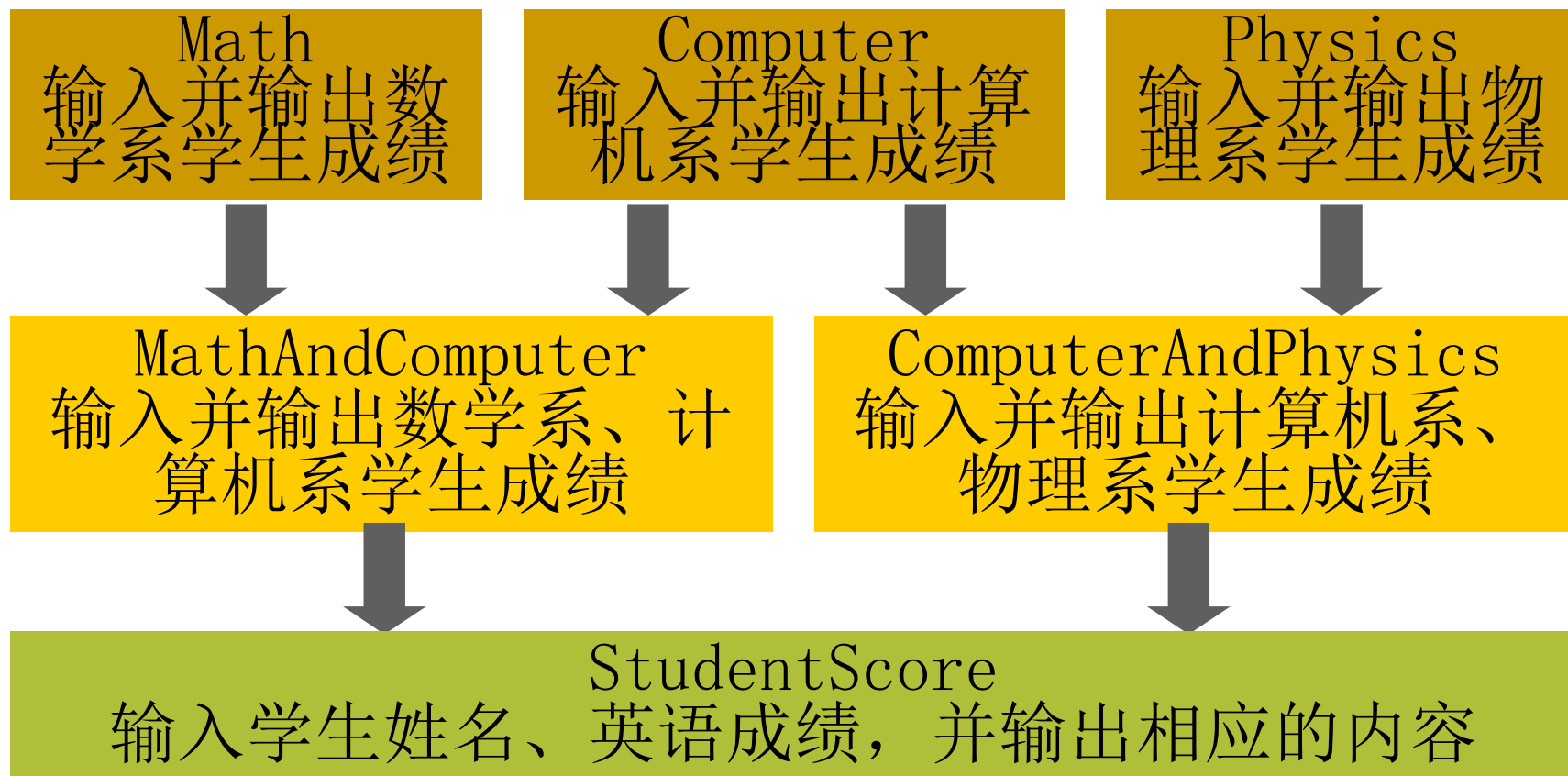
■ 多重继承



类继承(6)

例1-4

■ 多继承与多层次



函数重载

- 函数重载声明多个同名函数，完成不同功能，并带不同类型、数量的参数及返回值

//函数重载定义

```
int Function(int i)
{ return(2*i); }
float Function(float f)
{ return(2*f); }
```

//函数重载使用

```
int a=1;
float b=1.0;
iSum=Function(a);
fSum=Function(b);
```

操作符重载

例1-5

- 操作符重载为已有操作符赋予新功能，与原操作符的本来含义不冲突
- 操作符重载的声明方式：
函数类型 operator#(形参)
- 在使用重载的操作符时，根据操作符位置判断具体的执行

多态性与虚函数(1)



例1-6

- 虚函数是基类中声明为virtual，并在派生类中重新定义的函数
- 虚函数的功能：不修改基类的源代码，可修改基类的行为

多态性与虚函数(2)

- 在函数重载中，函数名相同，各函数的返回值类型与传递的参数类型不同
- 在虚函数中，函数名、函数类型和传递的参数类型相同，否则不能称为虚函数

I/O流结构

■ iostream

✓ istream: 输入流

cin>>、get、read

✓ ostream: 输出流

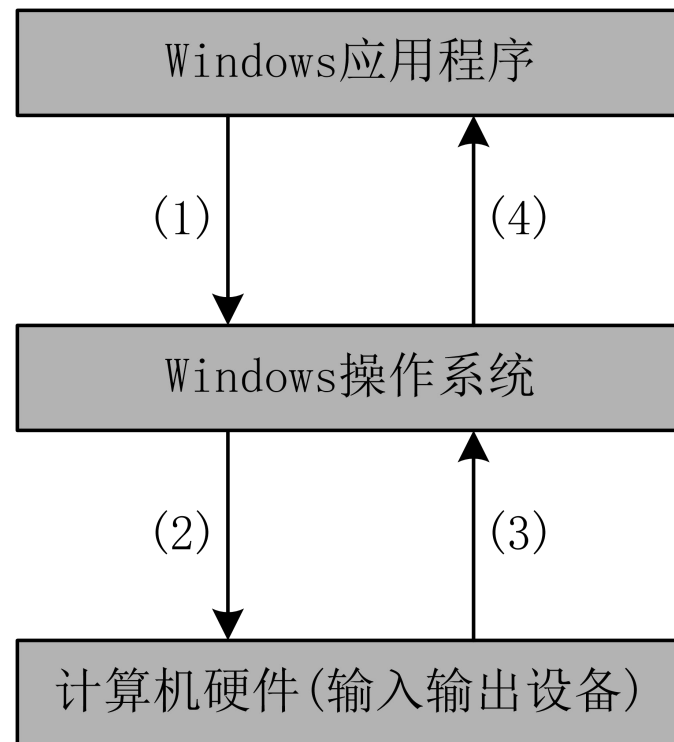
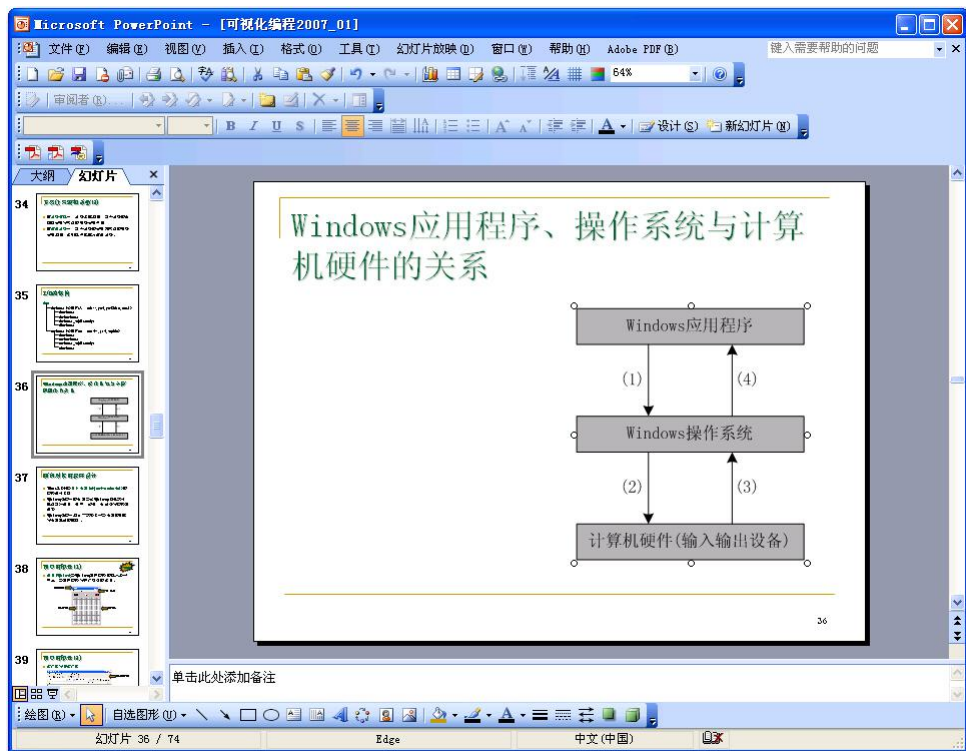
cout<<、put、write

以下关于类的哪个描述有错误？

- ☐ A 类是对相关数据与函数的封装
- ☐ B 类成员分为私有、公有与保护类型
- ☒ C 派生类可访问基类的私有成员
- ☐ D 友元函数可访问类的私有成员

提交

应用程序、操作系统与计算机硬件

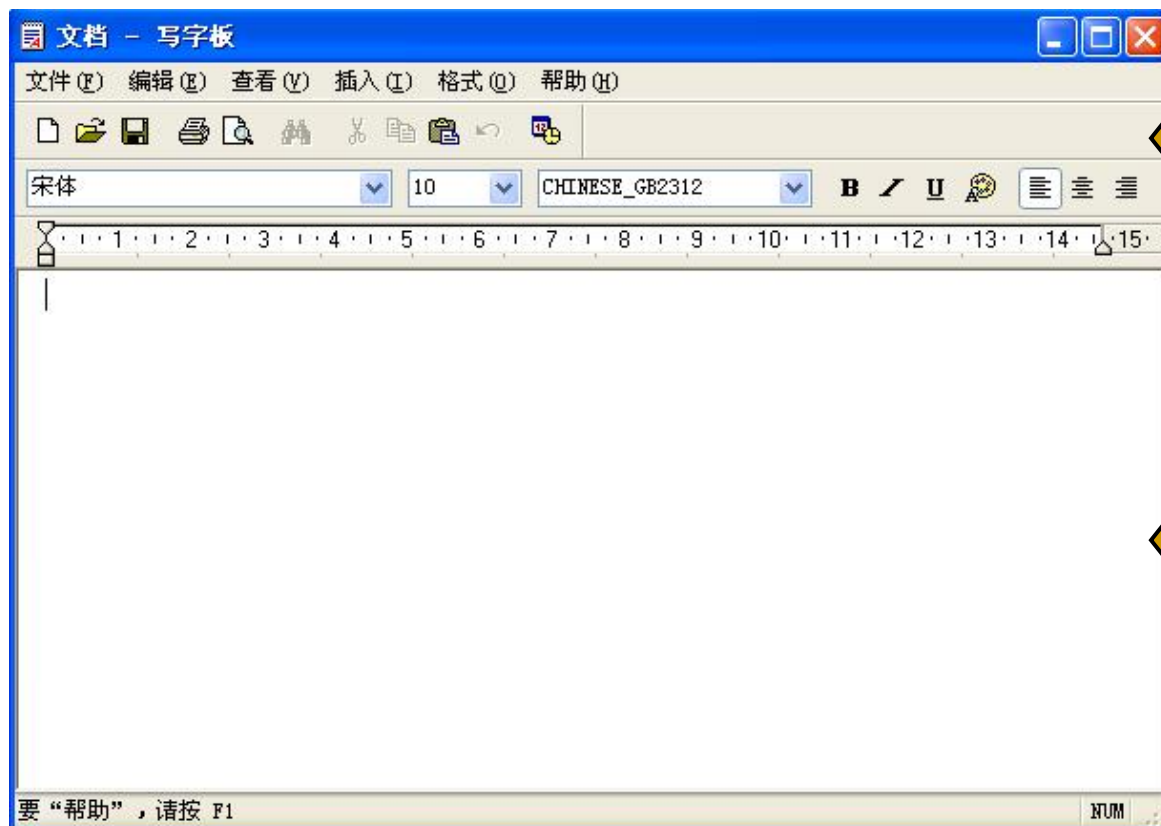


面向对象的程序设计

- Visual Studio是面向对象的编程工具
- 对象是规范化的Windows部件，包括窗口、菜单、控件、对话框等
- 大部分操作是创建对象与设置属性

窗口的概念

- 窗口(Window)是程序的用户界面

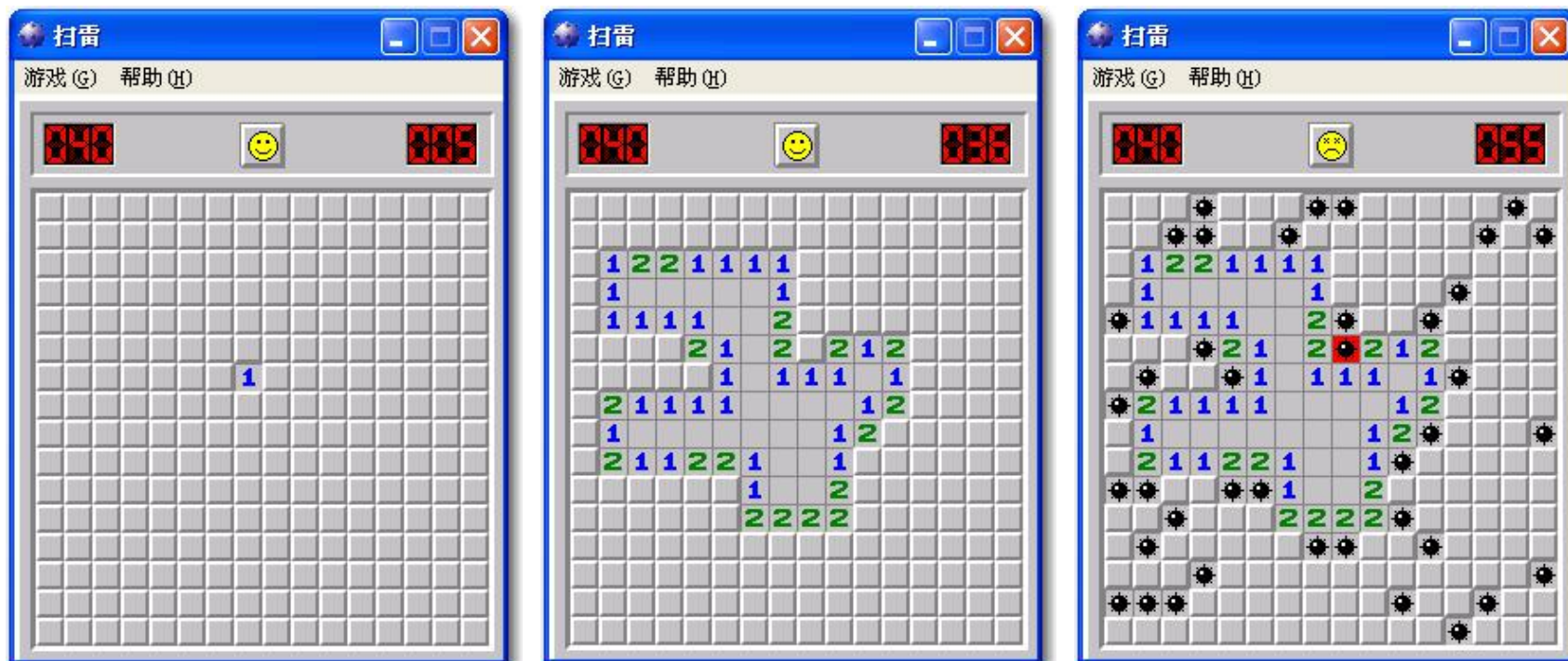


事件驱动 (1)

- Windows程序围绕事件(Event)形成, 进而引发相应处理函数运行, 称为事件驱动
- 消息是描述事件的信息, 例如按鼠标左键, Windows系统产生鼠标消息
- 程序执行取决于事件生成顺序, 即由生成消息的顺序来决定

事件驱动 (2)

■ 事件1→事件2→事件3



句柄的概念

- 句柄(Handle)是Windows系统标识对象的整数

句柄类型	说明	句柄类型	说明
HWND	窗口句柄	HDC	环境句柄
HINSTANCE	实例句柄	HBITMAP	位图句柄
HCURSOR	光标句柄	HICON	图标句柄
HPEN	画笔句柄	HMENU	菜单句柄
HBRUSH	画刷句柄	HFILE	文件句柄

消息的概念(1)

- 消息(Message)用于交换信息，结构为MSG

```
typedef struct tagMSG {  
    HWND    hwnd;        //窗口句柄  
    UINT    message;     //消息的值  
    WPARAM  wParam;      //消息附加信息  
    LPARAM  lParam;      //消息附加信息  
    DWORD    time;        //消息送至队列时间  
    POINT    pt;          //发送消息时光标位置  
} MSG;
```

消息的概念(2)

■ 系统定义的消息分类

消息前缀	说明	消息前缀	说明
WM_	窗口消息	LB_	列表框控件
BM_	按钮控件	SBM_	滚动条控件
EM_	编辑控件	TCM_	标签控件
CB_	组合框控件	TVM_	树状控件

消息的概念(3)

■ 常用的Window系统消息

消息名	说明	消息名	说明
WM_LBUTTONDOWN	鼠标按键	WM_CLOSE	关闭窗口
WM_KEYDOWN	键盘按键	WM_DESTROY	销毁窗口
WM_CHAR	非系统键	WM_QUIT	退出程序
WM_CREATE	创建窗口	WM_PAINT	绘制视图

Windows程序组成

■ 主要文件类型

文件扩展名	文件类型
.cpp	源程序文件
.h	头文件
.rc	资源描述文件
.vcxproj	主项目文件
.sln	解决方案文件

Windows程序框架(1)

- 创建窗口

- ✓ 设计窗口类、注册窗口类、创建窗口对象、显示及更新窗口

- 处理消息循环

- 编写窗口处理函数

Windows程序框架(2)

■ WinMain函数

- ✓ Windows程序的入口函数

■ WinMain函数功能

- ✓ 注册窗口类、建立窗口并初始化
- ✓ 处理消息循环，由消息队列接收消息，调用相应处理函数
- ✓ 接收WM_QUIT消息，终止程序运行

Windows程序框架(3)

■ WinMain函数声明

```
int WINAPI WinMain
(
    HINSTANCE hInstance,      //当前实例
    HINSTANCE hPrevInstance,  //前一个实例
    LPSTR lpCmdLine,          //命令行指针
    int nCmdShow              //窗口显示方式
)
```


Windows程序框架(4)

■ WinMain函数初始化

- ✓ 注册窗口 – RegisterClass()
- ✓ 创建窗口 – CreateWindow()
- ✓ 显示窗口 – ShowWindow()
- ✓ 更新窗口 – UpdateWindow()
- ✓ 加载图标 – LoadIcon()
- ✓ 加载光标 – LoadCursor()

Windows程序框架(5)

■ 设计窗口类

```
typedef struct _WNDCLASS {
    UINT style;                //窗口样式
    WNDPROC lpfnWndProc;       //窗口处理函数
    int cbClsExtra;            //类附加内存
    int cbWndExtra;            //窗口附加内存
    HANDLE hInstance;          //程序实例句柄
    HICON hIcon;               //图标句柄
    HCURSOR hCursor;           //光标句柄
    HBRUSH hbrBackground;      //背景画刷句柄
    LPCTSTR lpszMenuName;       //菜单资源名
    LPCTSTR lpszClassName;     //窗口类名
} WNDCLASS;
```

Windows程序框架(6)

- 在窗口类WNDCLASS中， `lpfnWndProc`成员指定窗口处理函数， 又称回调函数
- 当程序收到传给窗口的消息， 它调用某个函数处理该消息。调用过程由Windows系统完成，函数代码由应用程序提供

Windows程序框架(7)

■ CreateWindow函数

```
HWND CreateWindow (  
    LPCTSTR lpszClassName,    //窗口类名  
    LPCTSTR lpszTitle,        //窗口标题  
    DWORD dwStyle,            //窗口样式  
    int X, int Y,              //左上角坐标  
    int nWidth, int nHeight,   //宽度与高度  
    HWND hwndParent,          //父窗口句柄  
    HMENU hMenu,              //主菜单句柄  
    HINSTANCE hInstance,      //当前程序实例  
    LPVOID lpParam )          //传递参数指针
```

Windows程序框架(8)

■ 窗口样式

窗口样式	说明	窗口样式	说明
WS_BORDER	带边框	WS_MAXIMIZE	最大化窗口
WS_CAPTION	带标题栏	WS_MINIMIZE	最小化窗口
WS_CHILD	子窗口模式	WS_OVERLAPPED	带边框标题
WS_HSCROLL	可水平滚动	WS_VSCROLL	可垂直滚动
WS_MAXIMIZEBOX	带最大化按钮	WS_POPUP	弹出式窗口
WS_MINIMIZEBOX	带最小化按钮	WS_SYSMENU	带系统菜单

Windows程序框架(9)

■ 消息循环

- ✓ GetMessage() 从消息队列中取回消息
- ✓ DispatchMessage() 向窗口程序分发消息

```
MSG msg;  
while(GetMessage(&msg, NULL, 0, 0))  
{  
    TranslateMessage(&msg);  
    DispatchMessage(&msg);  
}
```

Windows程序框架(10)

■ 窗口处理函数

```
LRESULT CALLBACK WndProc(HWND hwnd, UINT  
message, WPARAM wParam, LPARAM lParam)  
{  
    switch(message)  
    { case WM_LBUTTONDOWN: ...  
      case WM_CLOSE: ...  
      default: ... }  
}
```

在Visual Studio中，打开整个项目所用的文件是

- ☐ A .cpp文件
- ☐ B .rc文件
- ☐ C .h文件
- ☒ D .sln文件

编写Windows应用程序

- 调用Windows提供的Win32 API
 - ✓ 大量程序代码由用户编写
- 使用MFC(微软基础类库)直接编写
 - ✓ 提供大量预先编写的类和代码
- 使用MFC和向导(Wizards)编写
 - ✓ 生成应用程序的框架结构

Windows API简介

- Windows API是应用编程接口 (Application Programming Interface)
- API提供Windows系统各种处理函数，主要函数在Windows.h中定义
- Windows提供1000多种API，基本按功能命名，例如CreateWindow、LoadIcon等
- MSDN为开发人员提供，包含开发文档与示例

调用Win32 API编程(1)

例1-7

```
#include <windows.h>
#include <stdio.h>
LRESULT CALLBACK WinSunProc(
    HWND hwnd,        //handle to window
    UINT uMsg,         //message identifier
    WPARAM wParam,     //first message parameter
    LPARAM lParam)     //second message parameter
{
    switch(uMsg)
    {
        case WM_LBUTTONDOWN:
            MessageBox(hwnd, L"Mouse Clicked!", L"Message 1", MB_OK);
            HDC hdc;
            hdc = GetDC(hwnd);
            TextOutW(hdc, 0, 0, L"Mouse Clicked!", strlen("Mouse Clicked!"));
            ReleaseDC(hwnd, hdc);
            break;
        case WM_CLOSE:
            if (IDYES == MessageBox(hwnd, L"Close Window?", L"Message 2", MB_YESNO))
                DestroyWindow(hwnd);
            break;
        case WM_DESTROY:
            PostQuitMessage(0);
            break;
        default:
            return DefWindowProc(hwnd, uMsg, wParam, lParam);
    }
    return 0;
}
```

```
int WINAPI WinMain(
    HINSTANCE hInstance,    //handle to current instance
    HINSTANCE hPrevInstance, //handle to previous instance
    LPSTR lpCmdLine,        //command line
    int nCmdShow)           //show state
{
    WNDCLASS wndcls;
    wndcls.cbClsExtra=0;
    wndcls.cbWndExtra=0;

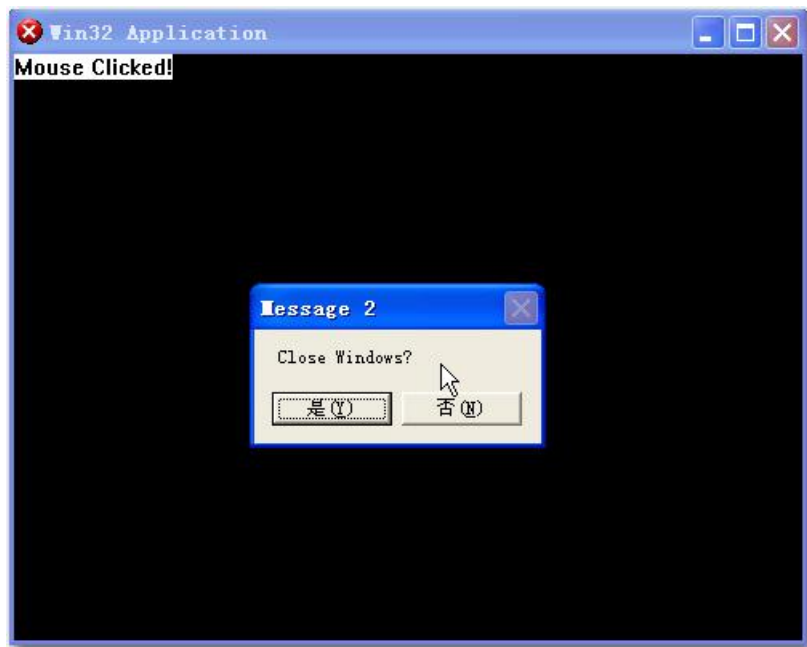
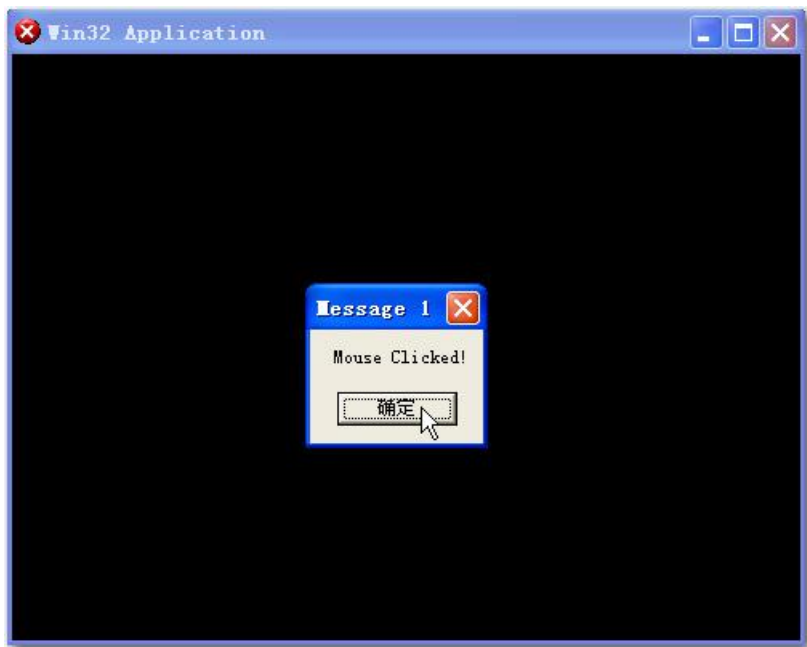
    wndcls.hbrBackground=(HBRUSH)GetStockObject(BLACK_BRUSH);
    wndcls.hCursor=LoadCursor(NULL, IDC_CROSS);
    wndcls.hIcon=LoadIcon(NULL, IDI_ERROR);
    wndcls.hInstance=hInstance;
    wndcls.lpfnWndProc=WinSunProc;
    wndcls.lpszClassName=L"Test";
    wndcls.lpszMenuName=NULL;
    wndcls.style=CS_HREDRAW|CS_VREDRAW;
    RegisterClass(&wndcls);
    HWND hwnd;
    hwnd=CreateWindow(L"Test", L"Win32 Application", WS_OVERLAPPEDWINDOW, 400, 300, 500, 400, NULL, NULL, hInstance, NULL);
    ShowWindow(hwnd, SW_SHOWNORMAL);
    UpdateWindow(hwnd);
    MSG msg;
    while(GetMessage(&msg, NULL, 0, 0))
    {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }
    return 0;
}
```

调用Win32 API编程(2)

■ 程序执行过程

第1步：鼠标左键事件

第2步：窗口关闭事件

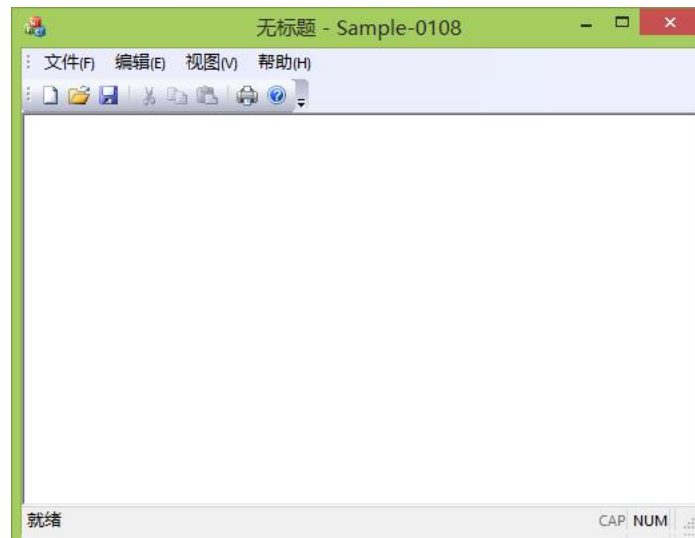
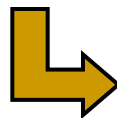
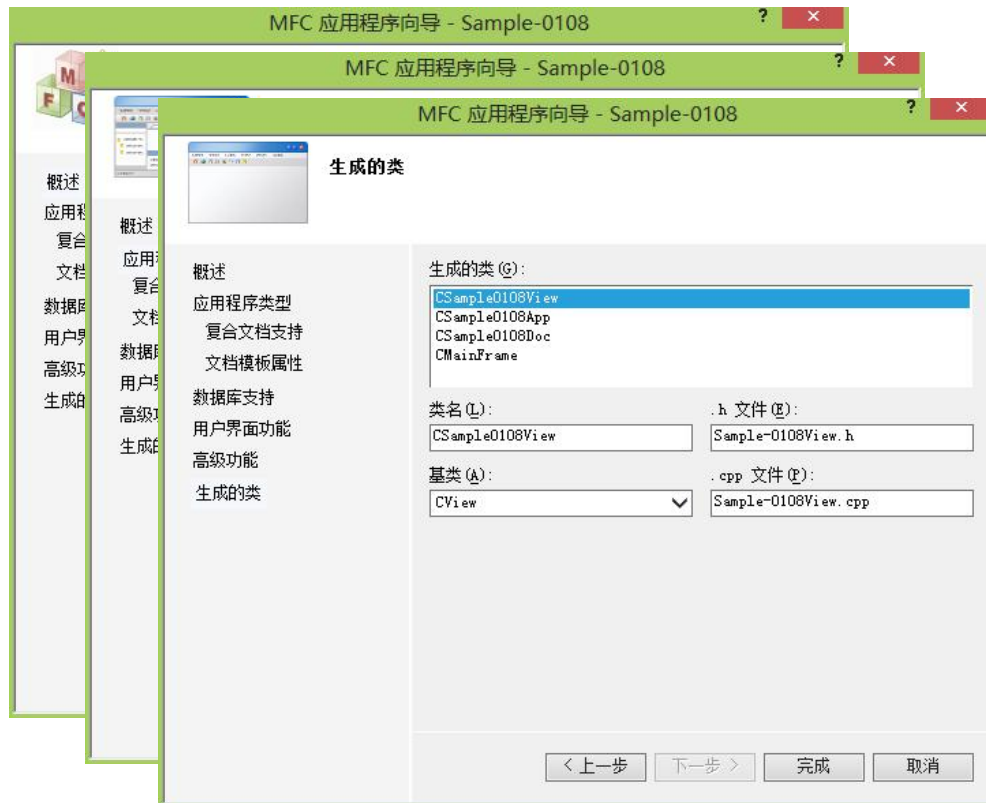


MFC简介

- 微软基础类库 (MFC, Microsoft Foundation Class) 提供各种类, 利用向导形成程序框架, 支持Windows应用程序开发
- MFC的优势: 完整封装Windows API, 极大减少需编写代码, 摆脱句柄困扰

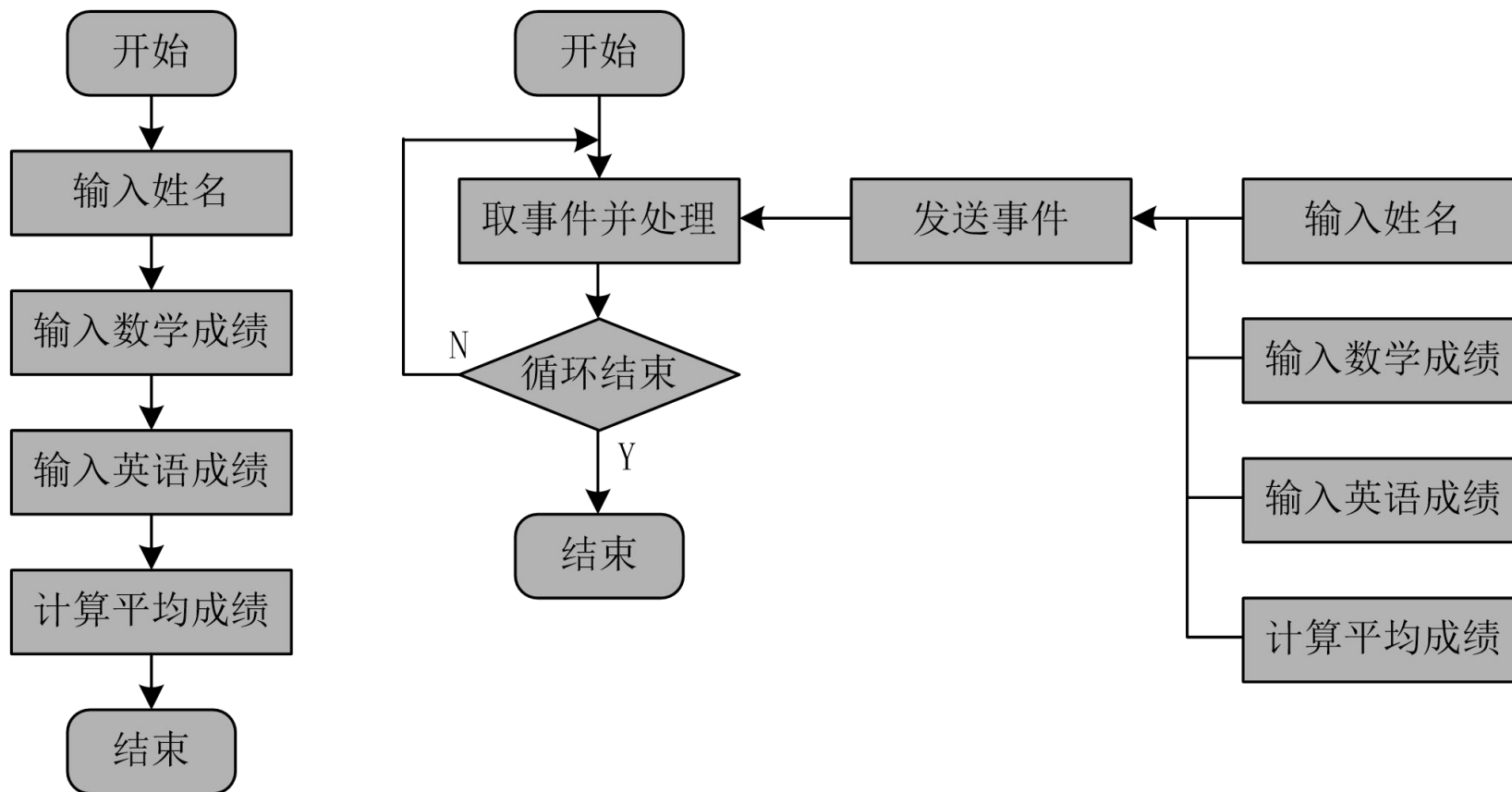
使用MFC和Wizards编程

例1-8



Windows程序设计特点(1)

■ 事件驱动机制



Windows程序设计特点(2)

■ 消息处理

- ✓ 输入消息：包括键盘、鼠标输入
- ✓ 控制消息：与控件双向通信，例如按钮等
- ✓ 系统消息：响应事件或时钟中断
- ✓ 用户消息：由用户自定义、程序主动发出，程序某部分处理

Windows程序设计特点(3)

■ 图形化输出

- ✓ Windows是多窗口系统，操作系统管理整个屏幕，应用程序仅管理部分
- ✓ Windows提供很多图形化处理函数
- ✓ Windows图形化输出与设备无关，使用图形设备接口(GDI)

Windows程序设计特点(4)

■ 用户界面设计

- ✓ Windows提供多种用户界面对象，包括窗口、菜单、图标、对话框等
- ✓ 编程者设计图形界面，编写简短代码
- ✓ 在命令行环境下，编写大量代码，完成同样工作，效果不好

Windows程序设计特点(5)

■ 资源共享模式

- ✓ Windows是多任务操作系统，应用程序共享系统资源，包括：设备、画笔、字体、图标、定时器等
- ✓ Windows程序共享资源方式，请求、使用与释放资源

Windows程序的基本运行模式是什么？

- ☐ A 程序语句顺序驱动
- ☒ B 事件与消息驱动
- ☐ C 图形用户界面驱动
- ☐ D 设备环境驱动

提交

第1次作业

- 设计学生类(Student)，数据成员是学号、姓名与成绩(数学、外语与计算机)，对学生类数组进行输入；以学号为参数搜索函数，从数组搜索并返回信息
- 定义一个二维方阵类，通过重载二元运算符“+、 \times ”，求二维方阵的加法和乘法



谢谢大家

