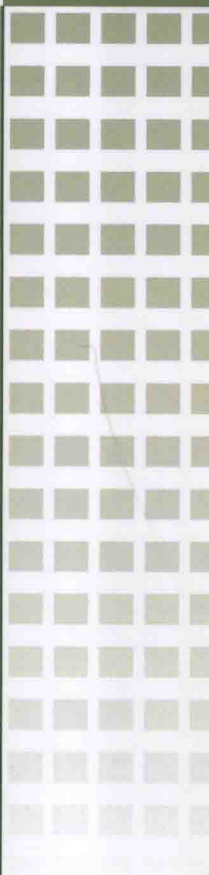
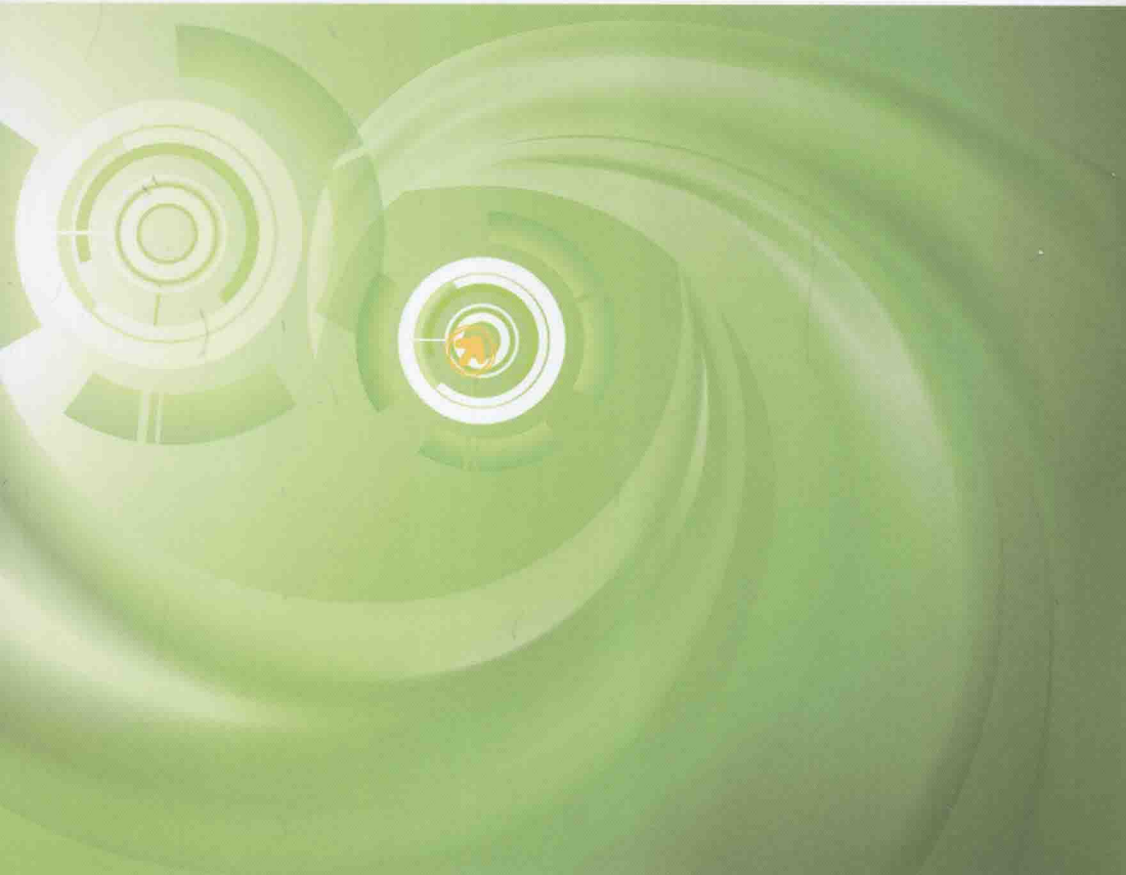




普通高等教育“十三五”规划教材



# Windows程序设计

陶勇 贺刚 著



科学出版社

普通高等教育“十三五”规划教材

# Windows 程序设计

陶 勇 贺 刚 著

科学出版社

北 京

## 内 容 简 介

本书围绕 Windows 程序设计的相关技术,从两个方面对 Windows 程序设计进行了深入的阐述。首先,讲解了 Windows SDK 程序设计知识,包括 Windows 的消息响应机制、GDI 绘图、消息处理、SDK 下的对话框应用程序设计;然后,讲解了 MFC 程序设计,包括 MFC 应用程序的创建及其框架、MFC 编程基础、MFC 消息映射与消息处理、MFC 文档类应用程序设计、MFC 对话框应用程序设计、ODBC 数据库应用程序设计。本书是在作者大量的工程实践的基础上编写的,不仅介绍了相关技术的原理,而且配有大量的示例。通过通俗易懂的文字,丰富、直观的配图以及经典的示例,让读者充分并深入理解 Windows 程序设计的原理与精髓。

本书可作为大学计算机专业和其他相关专业教育的教材,也可作为从事 Windows 应用程序设计及相关工作人员的参考用书。

### 图书在版编目(CIP)数据

Windows 程序设计/陶勇,贺刚著. —北京:科学出版社,2016

(普通高等教育“十三五”规划教材)

ISBN 978-7-03-049536-5

I. ①W… II. ①陶… ②贺… III. ①Windows 操作系统-程序设计-高等学校-教材 IV. ①TP316.7

中国版本图书馆 CIP 数据核字(2016)第 187048 号

责任编辑:朱敏 戴薇 王丽丽 / 责任校对:王万红

责任印制:吕春珉 / 封面设计:东方人华平面设计部

科学出版社 出版

北京东黄城根北街 16 号

邮政编码:100717

<http://www.sciencep.com>

北京京华虎彩印刷有限公司印刷

科学出版社发行 各地新华书店经销

\*

2016 年 7 月第 一 版 开本:787×1092 1/16

2016 年 7 月第一次印刷 印张:24

字数:563 000

定价:59.00 元

(如有印装质量问题,我社负责调换〈京华虎彩〉)

销售部电话 010-62136230 编辑部电话 010-62135927-2012

版权所有,侵权必究

举报电话:010-64030229; 010-64034315; 13501151303

# 前 言

Windows 程序设计是计算机专业学生的基本技能，也是计算机专业必须掌握的基本技术。本书作者长期从事计算机编程方面的教学与开发工作，在教学过程中深知全面掌握 Windows 程序设计知识体系的重要性。所以，本书从 Windows 程序设计的最基本内容——SDK 编程开始讲解，然后在此基础上讲解了 MFC 程序设计和 ODBC 数据库程序设计。本书的每个知识点都配有相应的例题，同时还提供作者课堂讲解的视频，为学习者提供了很大的便利。本书的例题、电子文档、学习视频资料的网盘地址为 <http://pan.baidu.com/s/1c1dzqes>，读者可自行下载。

本书分为 2 篇：SDK 程序设计、MFC 程序设计和 Windows 核心编程。

第 1 篇为 SDK 程序设计。其中，第 1 章为 Windows 图形界面基础，主要讲解 Windows 图形界面应用程序的结构及 Windows 消息机制。第 2 章为 GDI 基本图形，主要讲解 Windows 设备上下文及图形的绘制。第 3 章为 Windows 消息处理，包括键盘消息处理、鼠标消息处理、菜单消息处理、定时计数器消息处理、控件消息处理，这些消息处理的方式对于理解 MFC 的消息映射至关重要。第 4 章为 SDK 下的对话框应用程序设计，包括模态对话框和非模态对话框的启动、对话框中的控件，理解并掌握了 SDK 下的对话框应用程序设计，再去学习 MFC 下的对话框应用程序设计就会非常简单。

第 2 篇为 MFC 程序设计。其中，第 5 章为 MFC 应用程序的创建及其框架，主要讲解 MFC 应用程序的创建方法及文档类应用程序的框架。第 6 章为 MFC 编程基础，重点讲解 Windows 对象和 MFC 对象的关系，这是 MFC 应用程序设计中最需要理解的部分，只有充分理解了 Windows 对象和 MFC 对象的关系，才能轻松掌握 MFC 应用程序设计的方法。另外，本章还讲解了 MFC 应用程序设计向导的使用方法。第 7 章为 MFC 消息映射与消息处理，重点讲解 MFC 的消息映射机制、Windows 标准消息的处理、控件消息处理、命令消息处理、自定义消息处理、反射消息处理。本章是 MFC 的精华，读者在学习的过程中要加倍关注。第 8 章为 MFC 文档类应用程序设计，本章重点讲解文档的串行化。第 9 章为 MFC 对话框应用程序设计，本章重点讲解控件变量关联的问题，包括值变量和控件类变量以及这两种变量之间的区别和应用环境，然后讲解几个常用控件的使用方法。第 10 章为 ODBC 数据库应用程序设计，主要讲解 CDataBase 类和 CRecordset 类的使用方法，并讲解了本书作者多年来在项目中封装的 ODBC 数据类，在此基础上讲解了一个数据库应用程序的例题。

由于篇幅有限，关于 Windows 核心编程的文件及内存管理、网络通信程序设计以及 Windows 驱动程序开发等内容本书没有提及，希望以后逐渐完善。

由于时间和能力有限，书中不足与疏漏之处在所难免，恳请广大读者批评指正，以待进一步完善。

陶勇  
贺刚

湖北民族学院计算机科学与技术系

# 目 录

## 第 1 篇 SDK 程序设计

第 1 章 Windows 图形界面基础	3
1.1 基于 SDK 的第一个 Windows 图形界面程序	3
1.2 基于 SDK 的 Windows 应用程序框架代码详解	7
1.3 Windows 应用程序的基本结构与消息机制	37
1.4 创建自己的应用程序向导	38
1.4.1 创建一个基本的基于 Win32 SDK 应用程序	38
1.4.2 在 VC++ 6.0 中创建自定义模板工程	42
第 2 章 GDI 基本图形	45
2.1 设备上下文	45
2.1.1 设备上下文概述	45
2.1.2 常见的显示设备上下文	46
2.2 Windows 图形的绘制	52
2.2.1 颜色和像素点的设置	52
2.2.2 直线的绘制	53
2.2.3 画笔和画刷的使用	60
2.2.4 字体的创建	73
2.2.5 区域的绘制	74
第 3 章 Windows 消息处理	84
3.1 键盘消息	84
3.1.1 键盘消息概述	84
3.1.2 击键消息	84
3.1.3 系统击键与非系统击键	85
3.1.4 虚拟键码	86
3.1.5 lParam 信息	87
3.1.6 字符消息	87
3.2 鼠标消息	90
3.2.1 客户区鼠标消息	91
3.2.2 非客户区鼠标消息	92
3.2.3 非客户区命中测试消息	93

3.2.4	五子棋游戏 .....	94
3.3	菜单消息 .....	102
3.3.1	菜单概述 .....	102
3.3.2	菜单的添加举例 .....	103
3.3.3	菜单消息及菜单 UI 处理 .....	105
3.3.4	托盘技术 .....	120
3.4	定时计数器消息 .....	127
3.5	控件消息 .....	135
3.5.1	向窗口中添加控件 .....	135
3.5.2	响应控件消息 .....	143
<b>第 4 章</b>	<b>SDK 下的对话框应用程序设计 .....</b>	<b>147</b>
4.1	对话框应用程序的创建 .....	147
4.1.1	对话框资源的创建 .....	147
4.1.2	对话框应用程序的启动 .....	150
4.2	对话框应用程序的窗口过程函数及消息处理 .....	158
4.3	对话框基类的封装 .....	164
4.3.1	对话框基类概述 .....	164
4.3.2	从基类派生一个主窗口 .....	169
4.4	对话框应用程序模板的创建 .....	175
4.5	对话框应用程序中的控件详解 .....	176
4.5.1	SDK 下常用的控件控制函数 .....	176
4.5.2	图形显示控件和静态文本框控件 .....	179
4.5.3	单行编辑框控件 .....	186
4.5.4	按钮控件 .....	191
4.5.5	列表框控件 .....	196
4.5.6	组合框控件 .....	201
4.5.7	列表控件 .....	209
<b>第 2 篇 MFC 程序设计</b>		
<b>第 5 章</b>	<b>MFC 应用程序的创建及其框架 .....</b>	<b>235</b>
5.1	MFC 应用程序概述 .....	235
5.2	MFC 文档视图应用程序 .....	235
5.2.1	MFC 文档视图应用程序的创建 .....	236
5.2.2	MFC 文档视图应用程序框架 .....	241
5.3	MFC 对话框应用程序框架 .....	244
5.4	MFC 应用程序的生与死 .....	247

第 6 章 MFC 编程基础	249
6.1 MFC 编程与 SDK 编程	249
6.1.1 MFC 概述	249
6.1.2 MFC 对象与 Windows 对象的关系	249
6.1.3 MFC 对象和 Windows 对象的比较	250
6.1.4 MFC 对象与 Windows 对象的对应关系	251
6.1.5 MFC 窗口基类 CWnd	252
6.1.6 MFC 窗口对象的使用和销毁	258
6.2 MFC 窗口绘图	259
6.2.1 设备描述表概述	259
6.2.2 设备描述表在 MFC 中的实现	261
6.2.3 MFC 设备描述表类的使用	262
6.2.4 MFC 中的 GDI 对象	266
6.3 MFC 程序设计的 VC++ 向导	267
6.3.1 添加类	267
6.3.2 给类添加成员函数和成员变量	268
6.3.3 重载 CWnd 类的虚函数	269
6.3.4 添加 Windows 系统消息映射	269
6.3.5 添加或导入资源	271
第 7 章 MFC 消息映射与消息处理	272
7.1 MFC 消息映射概述	272
7.2 MFC 消息映射的基本概念	273
7.3 Windows 标准消息映射和消息处理	274
7.4 控件通知消息映射和消息处理	276
7.5 命令消息和命令消息处理	279
7.6 自定义消息和自定义消息处理	281
7.6.1 投递和发送消息	282
7.6.2 投递和发送消息举例	283
7.7 反射消息和反射消息处理	286
第 8 章 MFC 文档类应用程序设计	294
8.1 文档类 Windows 系统消息处理	294
8.2 文档类菜单处理	296
8.3 文档类应用程序的工具栏	300
8.4 文档的串行化	302
8.4.1 文档串行化的条件	302

8.4.2	MFC 的集合类	303
8.4.3	文档串行化类的实现	307
<b>第 9 章</b>	<b>MFC 对话框应用程序设计</b>	<b>313</b>
9.1	模态对话框和非模态对话框的创建	313
9.2	对话框中 Windows 消息处理	316
9.3	对话框菜单处理	317
9.4	对话框控件处理	319
9.4.1	对话框关联控件类型的选择	319
9.4.2	列表框控件处理	323
9.4.3	组合框控件处理	326
9.4.4	滑块条控件处理	330
9.4.5	列表控件处理	334
9.4.6	树形控件处理	339
<b>第 10 章</b>	<b>ODBC 数据库应用程序设计</b>	<b>346</b>
10.1	ODBC 数据库技术概述	346
10.2	CDatabase 类	347
10.3	CRecordset 类	348
10.3.1	快照型记录集、动态集和光标库	348
10.3.2	CRecordset 类的成员函数	349
10.4	封装 ODBC 数据库基类	351
10.4.1	CDB 类的功能	351
10.4.2	CDB 头文件	351
10.4.3	CDB 实现文件	352
10.5	学生信息管理系统的创建	358
10.5.1	数据库的创建	358
10.5.2	从 CDB 类中派生学生信息数据库处理类	359
10.5.3	学生信息管理界面程序设计	367
<b>参考文献</b>		<b>376</b>



# 第 1 篇 SDK 程序设计

SDK 是 Windows 操作系统为编程人员提供的软件开发包，由一些头文件和库文件组成。编程人员只需调用 SDK 开发包中的 API 函数，即可获得 Windows 操作系统提供的服务。

SDK 程序设计是 Windows 程序设计的基础。只有理解并掌握了 SDK 方式的 Windows 程序设计，才能在 MFC 应用程序设计的学习和使用中，知其然亦知其所以然，为后续的 Windows 核心编程、Windows 网络编程打下坚实的基础。

本篇分为 4 章，包括 Windows 图形界面基础、GDI 基本图形、Windows 消息处理及 SDK 下的对话框应用程序设计。另外，本篇的第 3 章在介绍鼠标消息时还附有一个俄罗斯方块的游戏设计，在本书配套的视频材料和源代码中会包含这一内容。

Windows SDK 程序设计是 Windows 程序设计的基础。因此，Windows SDK 程序设计的学习对于 Windows 编程人员来说非常重要。



# 第 1 章 Windows 图形界面基础

## 1.1 基于 SDK 的第一个 Windows 图形界面程序

利用 Microsoft Visual C++ 6.0（简称 VC++ 6.0）创建第一个基于 SDK 的 Windows 图形界面应用程序，步骤如下：

1) 运行 VC++ 6.0，选择 File→New 选项，弹出 New 对话框，如图 1-1 所示。

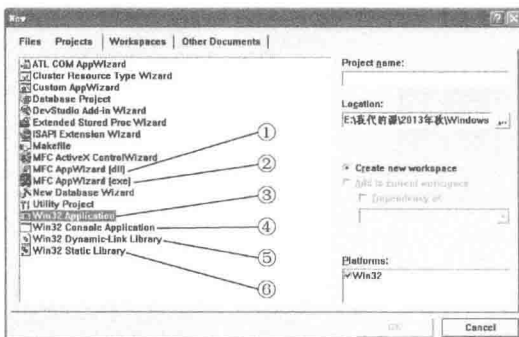


图 1-1 创建 SDK 应用程序

该对话框有 4 个选项卡：Files、Projects、Workspaces 和 Other Documents。其中，Files 与创建文件有关，如头文件、源文件；Projects 与创建工程有关；Workspaces 与创建工作区有关；Other Documents 与创建微软的其他文件有关，如 Word 文档等。

2) 选择 Projects 选项卡，可以看到该选项卡中列出了很多类型的工程，其中常用的 6 种在图 1-1 中已用数字标示。每种工程的功能如下：

- ① 用于创建 MFC 动态链接库；
- ② 用于创建 MFC 的应用程序；
- ③ 用于创建 SDK 的 Win32 应用程序；
- ④ 用于创建控制台应用程序；
- ⑤ 用于创建 Win32 动态链接库（基于 SDK 的动态链接库）；
- ⑥ 用于创建 Win32 的静态链接库（基于 SDK 的静态链接库）。

在 C 语言或 C++ 中创建的都是控制台应用程序，即第 4 种；而创建 SDK 的 Win32 应用程序时，选择第 3 种。

控制台应用程序和 SDK 的 Win32 应用程序的区别主要是 VC++ 6.0 运行时库调用的入口函数不一样。

控制台应用程序调用的入口函数是 main 函数。其函数原型为

```
main(int argc, char *argv[], char *envp[])
```

Win32 应用程序的入口函数为 `WinMain` 函数。其函数原型为

```
int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
    PSTR szCmdLine, int iCmdShow)
```

注意：如果创建的是控制台应用程序，而主函数使用的是 `WinMain` 函数，则编译会出错，因为控制台应用程序入口函数都是 `main` 函数。

在 Project name 文本框中输入项目名 `the hello program`，在 Location 文本框中输入目的路径，或者单击 Location 文本框后面的“文件浏览”按钮来选择文件路径。单击 OK 按钮，弹出图 1-2 所示的对话框。点选 `A simple Win32 application` 单选按钮，单击 Finish 按钮。

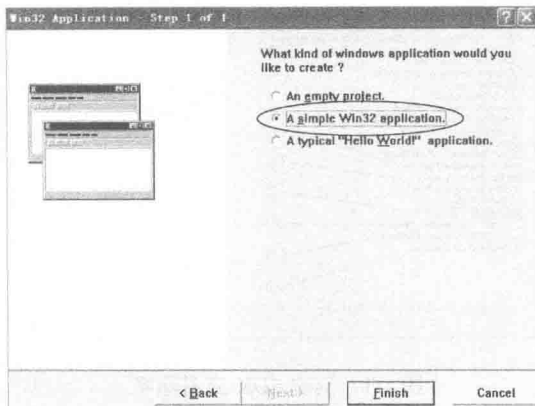


图 1-2 选择创建应用程序的类型

3) 在新创建的工程中会自动创建一个与工程名相同的 `.cpp` 文件。在文件中添加如下代码，其中黑体部分为新添加的代码：

```
#include <Windows.h> //1:必须包含头文件Windows.h
LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ; //2:声明窗口过程函数
int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
    PSTR szCmdLine, int iCmdShow) //3:Windows 应用程序的入口函数
{
    static TCHAR szAppName[]=TEXT("HelloWin"); //4:定义应用程序名变量
    HWND hWnd; //5:定义窗口句柄
    MSG msg; //6:定义消息结构体对象
    WNDCLASS wndclass; //7:定义要注册的窗口结构体
    wndclass.style=CS_HREDRAW|CS_VREDRAW; //8:窗口风格
    wndclass.lpfnWndProc=WndProc; //9:窗口过程函数
```

```

wndclass.cbClsExtra=0; //10: 该类型窗口的附加内存
wndclass.cbWndExtra=0; //11: 窗口对象的附加内存
wndclass.hInstance=hInstance;
//12: 窗口的应用程序实例句柄(产生窗口时由操作系统生成)
wndclass.hIcon=LoadIcon(NULL, IDI_APPLICATION); //13: 应用程序图标
wndclass.hCursor=LoadCursor(NULL, IDC_ARROW); //14: 应用程序光标
wndclass.hbrBackground=(HBRUSH)GetStockObject(WHITE_BRUSH); //15: 应用程序背景刷
wndclass.lpszMenuName=NULL; //16: 应用程序菜单
wndclass.lpszClassName=szAppName; //17: 窗口类型名
if(!RegisterClass(&wndclass))
//18: 在操作系统中注册第 7~17 步中定义的窗口类型
{
    MessageBox(NULL, TEXT("This program requires Windows NT!"),
        szAppName, MB_ICONERROR);
    return 0;
}
hWnd=CreateWindow(szAppName,
//19: 创建窗口, 第一个参数为注册时的窗口类型名
    TEXT("The Hello Program"),
    //20: 窗口标题
    WS_OVERLAPPEDWINDOW, //21: 窗口风格
    CW_USEDEFAULT, //22: 窗口初始位置 x 坐标
    CW_USEDEFAULT, //23: 窗口初始位置 y 坐标
    CW_USEDEFAULT, //24: 窗口的宽度
    CW_USEDEFAULT, //25: 窗口的高度
    NULL, //26: 父窗口句柄
    NULL, //27: 窗口菜单句柄
    hInstance, //28: 应用程序实例句柄
    NULL
//29: 传递给窗口过程函数的 lParam 参数指针
);
ShowWindow(hWnd, nCmdShow); //30: 显示窗口
UpdateWindow(hWnd); //31: 更新窗口
while(GetMessage(&msg, NULL, 0, 0)) //32: 消息循环
{
    TranslateMessage(&msg); //33: 消息转换
    DispatchMessage(&msg); //34: 分派消息
}
return msg.wParam;
}
LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM

```

```
lParam)
{
    //35:窗口过程函数
    HDC hdc; //36:设备上下文句柄
    PAINTSTRUCT ps;
    //37:调用 BeginPaint 获取设备句柄时需要传递的参数
    RECT rect; //38:矩形结构体对象
    switch(message) //39:处理消息
    {
        case WM_CREATE: //40:处理 WM_CREATE 消息
            PlaySound(TEXT("hellowin.wav"), NULL, SND_FILENAME|SND_ASYNC);
            return 0;
        case WM_PAINT: //41:处理 WM_PAINT 消息
            hdc=BeginPaint(hWnd, &ps);
            GetClientRect(hWnd, &rect);
            DrawText(hdc, TEXT("Hello, Windows 98!"), -1, &rect,
                DT_SINGLELINE|DT_CENTER|DT_VCENTER);
            EndPaint(hWnd, &ps);
            return 0;
        case WM_DESTROY: //42:处理 WM_DESTROY 消息
            PostQuitMessage(0); //43:发送一个 WM_QUIT 消息
            return 0;
    }
    return DefWindowProc(hWnd, message, wParam, lParam);
    //44:调用 Windows 默认的窗口过程函数
}
```

4) 编译, 运行上述代码, 出现图 1-3 所示的运行界面。

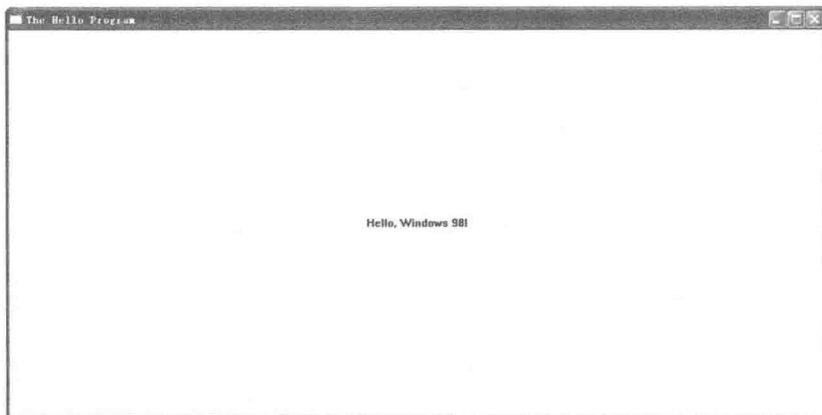


图 1-3 运行界面

下面我们将根据对上面代码的标注, 详细分析上述代码。

## 1.2 基于 SDK 的 Windows 应用程序框架代码详解

### 1. 必须包含头文件 Windows.h

Windows.h 是一个最重要的头文件，它包含了其他 Windows 头文件，这些头文件中的一些还包含了其他类型的头文件。比较重要和基本的头文件如下。

WINDEF.H: 基本数据类型定义。

WINNT.H: 支持 Unicode 的类型定义。

WINBASE.H Kernel: (内核) 函数。

WINUSER.H: 用户界面函数。

WINGDI.H: 图形设备接口函数。

这些头文件定义了 Windows 的所有资料型态、函数调用、资料结构和常数识别字，它们是 Windows 文件中的一个重要部分。

Windows 头文件很多，如果 VC++ 6.0 安装在 C 盘，那么可以在 C:\Program Files\Microsoft Visual Studio\VC98\Include 目录下打开该文件，该目录下包含了 Windows 开发的所有的头文件，如网络通信程序开发时所用的 WINSOCK.H 头文件等。

### 2. 声明窗口过程函数

窗口过程函数声明如下：

```
LRESULT CALLBACK WndProc(HWND,UINT,LPARAM,LPARAM);
```

#### (1) LRESULT

L 的意思是 LONG，RESULT 的意思是结果、返回值，LRESULT 即返回长整型结果。

LRESULT 是在 WINDEF.H 中定义的，可以在 VC++ 6.0 中右击 LRESULT，查看其定义如下：

```
typedef LONG LRESULT;
```

**注意：**typedef LONG LRESULT; 并不是创建了另一个数据类型，只是给 LONG 类型定义了另一个别名。其实，在 C 或 C++ 中也没有一种数据类型为 LONG，只有 long 类型。用上述同样的方法查看 LONG 类型的定义，其在 WINNT.H 文件中定义如下：

```
typedef long LONG;
```

所以，LRESULT 其实就是 long 类型，只是其另一个名称。

Windows 应用程序定义了很多类似的数据类型别名。下面详细介绍 Windows 的常用数据类型。查看 WINDEF.H 文件，Windows 定义的一些常用的数据类型及常量如下：

```
typedef unsigned long DWORD;  
typedef int BOOL;  
typedef unsigned char BYTE;
```

```

typedef unsigned short WORD;
typedef float FLOAT;
typedef FLOAT *PFLOAT;
typedef int INT;
typedef unsigned int UINT;
typedef unsigned int *PUINT;
typedef UINT WPARAM;
typedef LONG LPARAM;
typedef LONG LRESULT;
#define MAKEWORD(a,b) ((WORD) (((BYTE) (a)) | ((WORD) ((BYTE) (b))) << 8))
#define MAKELONG(a,b) ((LONG) (((WORD) (a)) | ((DWORD) ((WORD) (b))) << 16))
#define LOWORD(l) ((WORD) (l))
#define HIWORD(l) ((WORD) (((DWORD) (l)) >> 16) & 0xFFFF)
#define LOBYTE(w) ((BYTE) (w))
#define HIBYTE(w) ((BYTE) (((WORD) (w)) >> 8) & 0xFF)

```

可以看出, **DWORD** 其实就是一个无符号的长整型的别名, 所占内存为 4 字节; **BOOL** 是 **int** 的别名, 所占内存为 4 字节 (注意 **BOOL** 与 **bool** 之间的区别, **bool** 为布尔类型, 所占内存为 1 字节); **FLOAT** 是 **float** 类型的别名。

以下面几个定义为例对数据类型进行简单介绍。

### 1) 定义 1:

```
typedef FLOAT *PFLOAT;
```

将它的格式进行改变:

```
typedef FLOAT * PFLOAT;
```

可以发现, **PFLOAT** 就是 **FLOAT \***, 即 **float \***, 也就是浮点型指针。凡是以字母 **P** 开头的数据类型均为指针类型。

### 2) 定义 2:

```
typedef int near *PINT;
```

```
typedef int far *LPINT;
```

定义 2 多了两个修饰符: **near** 为 32 位地址修饰符, 只能在进程内调用, 可在 4GB 虚拟内存空间内寻址; **far** 也为 32 位地址修饰符, 可以跨进程调用, 可在整个 64TB 虚拟内存空间内寻址。

**PINT** 和 **LPINT** 都是一个 4 字节的 **int \*** 类型, 即整型指针。

### 3) 定义 3:

```
#define MAKEWORD(a,b) ((WORD) (((BYTE) (a)) | ((WORD) ((BYTE) (b))) << 8))
```

**MAKEWORD** 宏, 字面意义为“制造字”, 即用 2 字节类型的数据构成一个字。例如,



```
MAKEWORD(1,2)
```

该宏被替换后为

```
((WORD)((BYTE)(a)|((WORD)((BYTE)(b))<<8))
```

将括号简化后该语句变为

```
(WORD)(BYTE)(a)|(WORD)(BYTE)(b)<<8
```

语句执行顺序如下：

- ① 将 a 转化为 BYTE 类型，然后将前一步转换的结果转换为 WORD 类型；
- ② 将 b 转换为 BYTE 类型，然后将前一步的结果转换为 WORD 类型，然后左移 8 位；
- ③ 将步骤①的结果和步骤②的结果按位相或。

所以 MAKEWORD(1,2)的执行过程为

- ① 00000001：将 1 转化为 BYTE 类型，即 8 位二进制数；  
0000000000000001：将 00000001 转化为 WORD 类型；
- ② 00000010：将 2 转化为 BYTE 类型，即 8 位二进制数；  
0000000000000010：将 00000010 转化为 WORD 类型；  
0000001000000000：左移 8 位；
- ③ 0000001000000001：按位相或。

在编程过程中常会用到 MAKEWORD 宏。例如，加载函数库时需要提供函数库的版本，版本通常分为主版本和次版本，版本会通过一个参数进行传递，此时就可以通过“MAKEWORD(次版本号,主版本号)”来获得一个版本参数。再如，在进行 WIN SOCKET 套接字编程时需要用到 MAKEWORD 宏，以确定 WIN SOCKET 库的版本号。

4) 定义 4:

```
#define LOWORD(l)((WORD)(l))
```

该宏的功能为获取长整型数的低十六位。

```
#define HIWORD(l)((WORD)((DWORD)(l)>>16)&0xFFFF)
```

该宏的功能为获取长整型数的高十六位。

这两个宏在编程中也会用到，当单击时，鼠标指针的坐标会通过一个 32 位数传递给窗口过程函数的 lParam 参数。可以通过如下代码得到鼠标指针的坐标 (x,y):

```
int x=(int)LOWORD(lparam);
int y=(int)HIWORD(lparam);
```

我们需要适应 Windows 定义的这些数据类型的别名，以达到看到数据类型名，即可知道该数据的功能的目的。

(2) CALLBACK 和 WINAPI

CALLBACK 和 WINAPI 与函数的调用约定有关，由 CALLBACK 和 WINAPI 修饰