Visual C++

经典例程分析

乌尼尔　董海军　编著

中国电力出版社
www.infopower.com.cn

Visual C++

经典例程分析

乌尼尔　董海军　编著

中国电力出版社

内 容 提 要

本书以实例为中心，循序渐进、由浅入深地介绍了如何利用 Visual C++进行编程。全书共分九章，从简单的应用程序分析到大型软件的开发，本书均有详细论述。所选实例实用性很强，而且包含最新流行技术，如网络编程、网络数据通信、网络数据库、数据安全和数据加密、网络安全等。本书最后一个程序是一个大型的管理系统，具有相当的实用价值。

本书适用于有一定 C 或 C++基础的读者。

图书在版编目（CIP）数据

Visual C++ 经典例程分析/乌尼尔，董海军 编著.-北京：中国电力出版社，2001.

ISBN 7-5083-0561-2

Ⅰ.V… Ⅱ.①乌…②董… Ⅲ.C 语言 -程序设计 Ⅳ.TP312

中国版本图书馆 CIP 数据核字（2001）第 12490 号

中国电力出版社出版、发行

（北京三里河路 6 号 100044 http://www.infopower.com.cn）

三河市实验小学印刷厂印刷

各地新华书店经售

*

2001 年 5 月第一版 2001 年 5 月北京第一次印刷

787 毫米×1092 毫米 16 开本 23.25 印张 524 千字

定价 35.00 元

序　言

Visual C++ 为我们提供了一个功能强大、使用灵活的开发环境，是 Windows 应用程序开发平台中的佼佼者。但学习 Visual C++ 并不容易。本书以实例为中心详细地讨论了如何使用 Visual C++ 开发应用软件，以及与此有关的概念和方法，内容涉及了从入门到精通的各个阶段，从多个侧面，以多种途径，全面、系统地阐述了 Visual C++ 环境下应用程序设计方法，可以帮助读者在较短的时间内掌握 Visual C++ 的使用方法和技巧。

软件介绍

Visual C++是微软公司(Microsoft)在 Microsoft C/C++ 7.0 的基础上开发的新型软件。Microsoft C/C++ 7.0 标志着面向对象技术的成熟和完善，开创了以面向对象技术为主导的软件设计的新时代。随着操作系统的发展，Visual C++ 1.0 的出现取代了 Microsoft C/C++ 7.0；它给出了开发 Windows 应用程序的一个 C++ API，即 MFC 2.0 库。MFC 2.0 库是建立在 Microsoft C/C++ 7.0 提供的 MFC1.0 基础之上的，并对 1.0 库进行了扩充，增加了许多新功能，如支持 DLL、更加容易实现 OLE 的接口等。更为重要的是，Visual C++ 1.0 提供了一个基于 Windows 的调试器，而且它所提供的 Windows 风格的编辑器也很受用户欢迎。

后来，Microsoft 公司又陆续推出了 Visual C++ 的后续版本，配备了 Code View for Windows，用来调试 Visual C++ 设计的 Windows 程序；把资源设计集成到 Visual C++ 的工作台（Workbench）中，可用 Resource 菜单访问来设计 Windows 应用程序的界面。目前的最新版本是 Visual C++ 7.0，它系统稳定性好、代码效率高、编译速度快、界面友好，从而成为 C/C++ 产品中最有竞争力的一个软件。

本书导读

本书以实例为中心，循序渐进、由浅入深地介绍了如何利用 VC++ 进行编程。例程的选择注重其实用性，从简单的应用程序分析到大型软件的开发，本书均有详细论述。所选实例不仅有开发一般性应用程序所需技术，而且包含最新的流行技术，如网络编程、网络数据通信、网络数据库、数据安全和数据加密、网络安全等。本书最后一个程序是一个大型的实用系统，不仅提供了一个较大的平台让大家来分析大型应用程序，而且其本身也具有相当价值。

本书共有十个例程，按其内容划分为九章，各章内容如下：

第一章　控制台应用程序与 Win32 应用程序

第二章　对话框应用及其数据交换机制

第三章　设计 VC 式的 MDI
第四章　动态控件应用及定制通用对话框
第五章　图像列表和程序间通信
第六章　定时器的设计和 OCX 控件的使用
第七章　分页表应用实例
第八章　超链接应用实例
第九章　综合开发实例

本书特色

本书最大的特色就是以实例为中心。书中对每一个知识点的介绍都是通过分析实例程序来进行的。所选实例都具有一定的代表性，而且都是面向实际运用的，这样使读者学到的不是死的书本知识，而是真正实用的编程方法。

本书从实用技术、技巧开始展开，直到设想的具体实现，由浅入深，形成了一套完整的 Visual C++ 程序设计思路和方法，关于应用范例、实用技术与技巧等问题的讲述，是作者利用 Visual C++ 开发应用程序经验的总结。书中介绍了在 Visual C++ 环境下，基于客户/服务器结构开发实时数据库管理系统的方法，这些方法和使用到的技巧是作者首次提出的，对开发其他应用程序也有一定借鉴意义，由于成书时间与作者水平，书中难免会有一些疏漏，请广大读者批评指正。

作者
2000.于北航

目　录

第一章

控制台应用程序与 Win32 应用程序

Windows 应用程序都有非常类似的用户界面风格，允许用户通过点击、选择等操作实现事件的触发，并产生相应的消息交给应用程序处理，从而实现具体功能。这些消息的处理建立在 Microsoft Win32 API(Application Programming Interface)即 Win32 应用程序接口基础上，相应的 Win32 API 函数功能由动态连接库提供。直接对 Win32 API 编程是比较复杂的，虽然 Win32 API 和 Windows 系列操作系统在不断发展，但 Win32 API 不是面向对象的，因此无论是有过 Windows 编程经验的人还是从未编写过 Windows 应用程序的人，MFC 编程都是一个新的领域。

本章对于没有编写过 Windows 应用程序的人来说可以熟悉一下 MFC 的编程；对于有一定的 Windows 编程经验的人来说，可以让他们在比较熟悉的环境中逐步认识 MFC 编程的特点。

本章主要内容为：

- Visual C++ 6.0 提供的控制台应用程序。
- Win32 应用程序创建方法。

1.1　控制台应用程序

对于一些从来没有做过 Windows 应用程序的人来说，直接深入地介绍 MFC 编程似乎难度太大，他们还对手中短小精悍的 Turbo C 或是 Borland C++ 程序津津乐道，而对 Windows 应用程序中的消息处理茫然不知头绪。没有关系，Visual C++ 6.0 提供了控制台应用程序，可以像在 Turbo C 或是 Borland C++ 环境下制作一些小程序一样来编程。

下面就通过 Hello 控制台程序和另一个控制台程序实例来介绍这个方法。

1.1.1　Hello 控制台程序

在这一节我们来生成一个 HELLO 控制台程序，然后对它进行分析。这个程序的功能就是在屏幕上显示“Hello World”。程序虽然非常简单，我们还是提供给大家，并对它作一简单的分析。

Hello 控制台程序生成

我们已经知道工程的创建方法，首先需要从 Developer Studio 的 File 文件菜单中选择 New...选项，系统将弹出如图 1-1 所示对话框，要求选择新建工程或文件类型。

这里要选择 Win32 Console Application，即 Win32 控制台应用程序。项目名取 console，这样将在当前的路径下（D:\vcbook\）创建一个 console.dsw 的控制台项目，如图 1-1 所示。

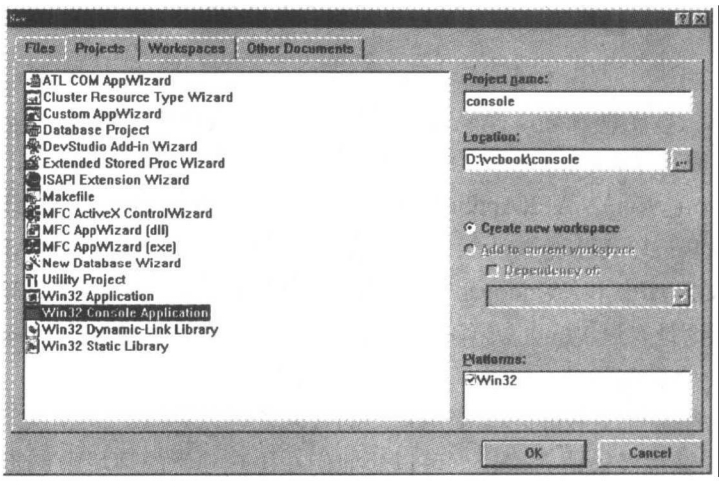


图 1-1　控制台应用程序创建

在弹出的对话框中，要求选择控制台应用程序的类型，有 4 种可选类型：

✧　空的应用程序

系统将不提供任何代码，完全由程序员编写。

✧　简单的应用程序

系统提供简单的结构代码，但是不完成任何功能。

✧　能提示“Hello World”的应用程序

能够在控制台上显示“Hello World”。

✧　支持 MFC 的应用程序

应用程序具有一个全局变量，一般名为 theApp，其类型为 MFC 的 CWinApp 应用程序类，由此可以建立与 MFC 应用程序的关系，利用该变量可以将应用程序各个独立的类联系成一个有机整体。

这里我们选择第 3 项，即生成一个能够提示“Hello World”的应用程序。按照系统提示由其自动生成应用程序，我们先不分析这个新生成的程序中有什么代码，以及各代码的作用，为了理解控制台的概念，先来看这个应用程序的执行效果如何，如图 1-2 所示。

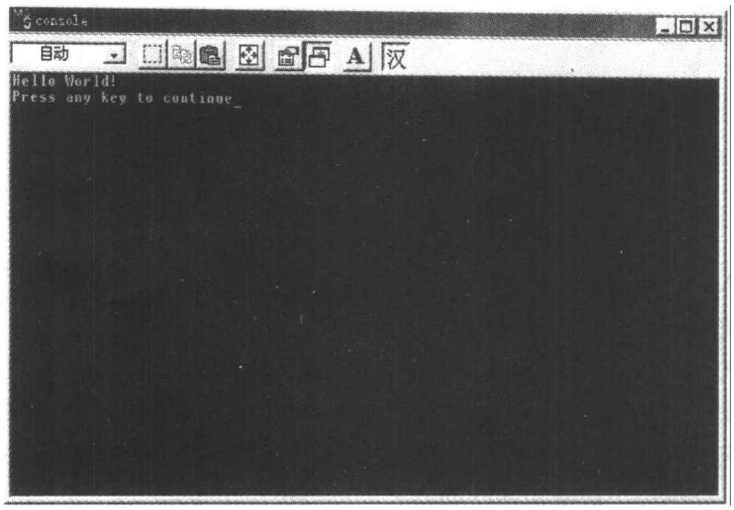


图 1-2　“Hello World”控制台应用程序的执行结果

这与以往所见到的 DOS 应用并没有什么不同，程序的运行结果显示在一个黑屏上，而且没有窗口的有关特征：程序一旦执行，用户就只能从这个简单的黑屏上查看相关的结果，而不能影响处理过程，这也是为什么称之为控制台应用程序的原因。

下面再来分析这个程序相关的代码。

控制台应用程序代码分析

首先从 Project Workspace 窗口中可以看到控制台应用程序没有资源页（Resource View），在类（Class View）页中只有全局量和全局操作一栏（Globals），而且只有一个主函数 main(int argc,char *argv[])，很明显这与以往的 C 程序没有不同，如图 1-3 所示。

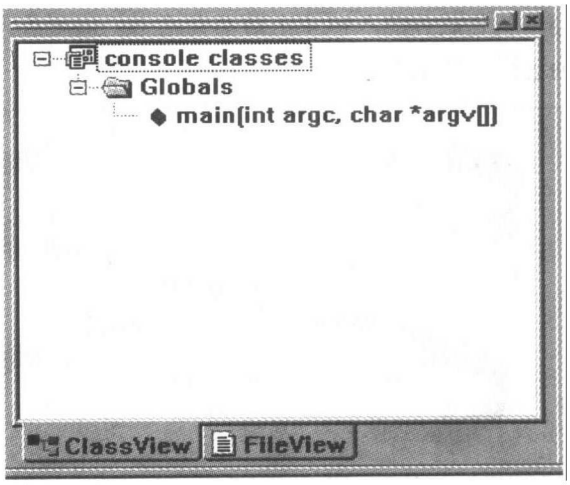


图 1-3　控制台应用代码分析

再来看这个主函数的定义：

```
int main(int argc, char* argv[])
{
printf("Hello World!\n");
return 0;
}
```

作为程序入口，main 函数是唯一而且必要的。其定义代码只有两句：即输出语句和返回语句，所使用的都是标准 C 语法。

从这个意义上讲，任何会写 C 程序的人都可以说他会编写 Visual C++ 程序，因为控制台应用程序没有涉及到 MFC 编程的概念，但要清楚现在所写的程序功能还很有限。

在控制台应用程序中，同样可以灵活地使用 Visual C++ 提供的类，下面以队列（deque）类的使用为例，介绍一个能够描述队列特点的控制台应用程序。

1.1.2 控制台程序实例

这个程序是一个关于队列操作的程序，程序本身涉及到一些队列知识。因此在介绍这个关于队列特点的控制台应用程序之前，有必要简单介绍一下队列的知识。队列是数据结构的重要内容，而要想真正编好程序，数据结构也不可忽视。

下面先介绍队列，然后再介绍这个程序的编写。

队列

队列就是我们平常所说的“排队”，很明显“排队”有先后顺序，那么队列同样有一个基本特征：FIFO（first in first out），即先进先出，这是最简单的队列类型。在 Visual C++ 提供了一个双端队列类，就是队列的入口和出口都有两个，但所有入队的元素出队时还是要按照先进先出的规则进行，如图 1-4 所示。

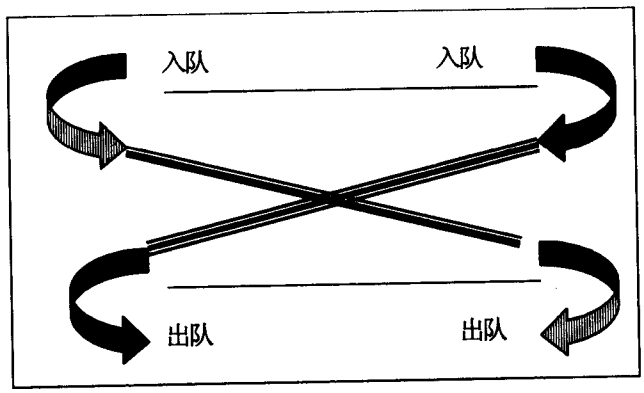
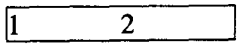


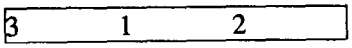
图 1-4 双端队列

Visual C++ 提供的队列类型实际上是一个队列模板，该文件可以在 Visual Studio 安装路径下的 vc98\include 子目录中找到（如 C:\Program Files\Microsoft Visual Studio\VC98\Include）。此模板定义了队列的各种操作，如由前端入队、后端入队、前端出队、后端出队以及队列长度等属性。

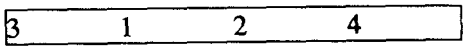
本例中我们希望检查前端入队、后端入队、前端出队、后端出队的操作结果。例如先从后端将元素 1 和 2 入队，此时队列的状态应该为：



再由前端将元素 3 入队，队列状态应该为：

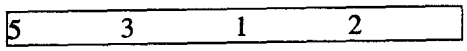


再由后端让元素 4 入队，队列状态为：

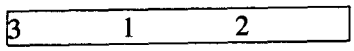


由前端入队元素 5，队列状态为：

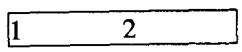
此时若从后端弹出元素，则应该弹出元素 4，队列状态调整为：



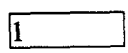
若再从前端弹出元素，弹出的元素应该为 5，队列状态为：



同样地，再从前端弹出元素 3 后的队列状态应该为：



最后，从后端弹出元素 2 后，队列状态调整为：



即队中仅存一个元素 1。下面我们就制作一个建立在双端队列（deque）基础上的控制台应用程序，检查双端队列的操作是否正常。

队列操作实现

创建了一个新的控制台应用程序后，需要将双端队列类的定义文件包含到工程中：

```
#include <deque>
```

由于这里提供的是一个队列模板，即队列元素的类型是不确定的，因此有必要声明一个具体类型的队列，定义名空间：

```
using namespace std;
```

这里将队列元素类型设置为整型，并定义了 INTDEQUE 作为元素类型为整型的双端队列类：

```
typedef deque<int>  INTDEQUE;
```

有了类的定义，可以声明一个队列对象：

```
INTDEQUE  dequetest;
```

即 dequetest 是一个元素为整型的双端队列对象。

下面的工作就要完成具体的入队和出队操作了，由队列的定义就可以充分利用下列方法：

```
void push_front(const _Ty& _X)
void pop_front()
void push_back(const _Ty& _X)
void pop_back()
```

这里参数（const _Ty& _X）表示相应类型的元素。这样前面所需要完成的元素入队和出队操作就可以用以下代码完成（注意元素类型一定要匹配）。

```
dequetest.push_back(1);      //由后端将元素 1 入队列
dequetest.push_front(3);     //由前端将元素 3 入队列
dequetest.pop_back();        //由后端出队列
dequetest.pop_front();       //由前端出队列
```

现在已经可以实现有关的操作了，下一步的任务就是定义查看结果的方法，具体地查看队列的内容，就是将队列中的元素逐个进行显示。那么首先需要确定队列的当前位置，可以通过声明一个指向队列的指针来完成：

```
INTDEQUE::iterator pdeque;
```

队列指针是在队列的起始位置和结束位置之间移动的，这里需要用队列的 begin()和 end()函数来标识其起始和结束位置。另外还可以使用 front()和 back()返回当前的队头元素和队尾元素。

具体程序代码如下：

```
#include <stdafx.h>       // 应用程序自动生成的标准头文件，并没有增加具体代码
#include <iostream>       // 处理输入输出流的库文件
#include <deque>          // 将队列模板定义文件包含到工程中
using namespace std;      // 定义名空间
typedef deque<int>  INTDEQUE;          // 定义元素为整型的双端队列类
void printcontents (INTDEQUE  deque) ;// 声明队列的输出函数

void main()                            // 主程序入口
{
INTDEQUE  dequetest;                   // 声明一个双端队列对象
dequetest.push_back(1);                // 具体入队和出队操作
dequetest.push_back(2);
dequetest.push_front(3);
dequetest.push_back(4);
dequetest.push_front(5);
printcontents (dequetest);                  // 示当前队列状态
```

```
dequetest.pop_back();
printcontents (dequetest);
dequetest.pop_front();
printcontents (dequetest);
dequetest.pop_front();
printcontents (dequetest);
dequetest.pop_back();
printcontents (dequetest);
}

void printcontents (INTDEQUE  deque)
{
    INTDEQUE::iterator pdeque;                    // 设置指针指向队列当前位置
     cout <<"The output is:"<<endl;
        for(pdeque = deque.begin();pdeque != deque.end();pdeque++)
         cout << *pdeque ;                        // 由队列的起始位置开始逐元素显示
     cout<<endl ;
     cout<<"front"<<deque.front()<<" back"<<deque.back()<<endl ;
     //指示当前的队头元素和队尾元素
}
```

此应用程序的执行结果如图 1-5 所示：

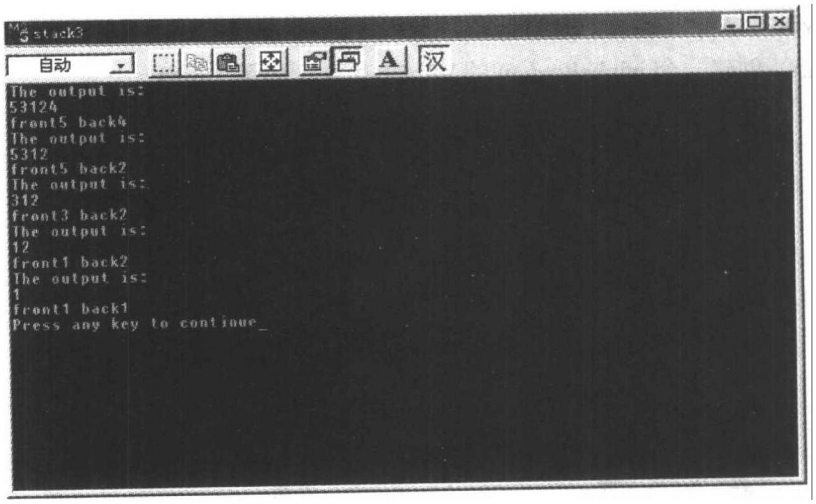


图 1-5　队列控制台应用程序执行结果

虽然完成了有关的操作，并且可以由显示屏中查看执行结果，但不可否认，这个应用程序过于“简陋”了。下一步，要向更美观和更实用的方向努力。

1.2 Win32 应用程序

一个 MFC 应用程序的结构，与以往的编程结构有些类似，但又有很大的不同，对于初学者来说可能有一些不适应。但是，Visual C++ 6.0 为用户提供了 Win32 编程风格，我们可以先分析 Win32 应用程序与 MFC 应用程序有什么关系。没有写过 Windows 应用程序的人可以由这一部分的内容初步掌握 Windows 应用程序风格，然后和具备 Windows 应用程序开发基础的程序员一起“平滑”地过渡到 MFC 编程领域中来。

下面通过两个 Win32 应用程序实例来介绍相关知识。

1.2.1 一个同样说“Hello World”的 Win32 应用程序

为了与前面的控制台应用程序对照分析，这里用 AppWizard 生成一个功能完全相同的 Win32 应用程序，然后再对它进行分析，这样可以看到它与控制台应用程序有什么不同。

Hello Win32 程序生成

首先需要从 Developer Studio 的 File 菜单中选择 New...选项，系统将弹出如图 1-6 所示下对话框，要求选择新建工程或文件类型。

选择 Win32 Application，即 Win32 应用程序。项目名取 winhello，将在当前的路径下（D:\vcbook\）创建一个 winhello.dsw 的 Win32 项目，如图 1-6 所示。

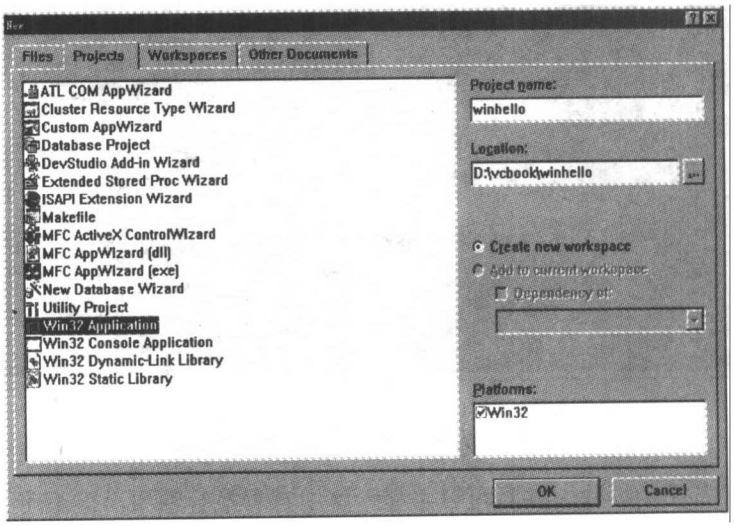


图 1-6 创建 Win32 应用程序

在弹出的对话框中，需要设置应用程序类型，有 3 种可选项：

- ❖ 一个空的项目——系统不生成任何文件，完全由程序员自行开发。

❖　简单的应用程序——系统提供简单的结构代码，但是不完成任何功能。

❖　典型的“Hello World”应用——能够在应用程序生成的窗口中显示“Hello World”。

选择第三项，系统自动生成 winhello 应用程序，它的执行结果如图 1-7 所示。

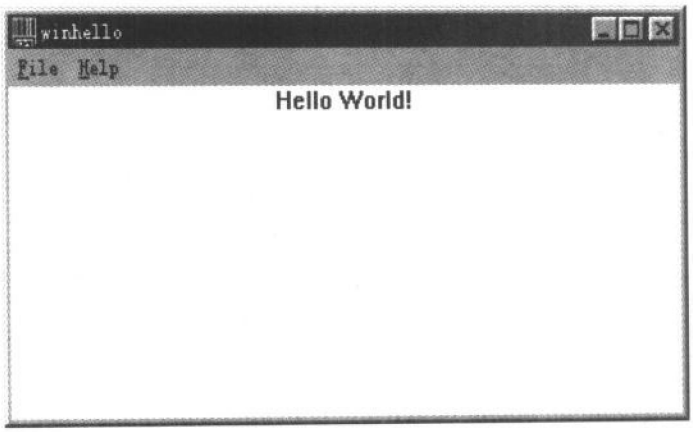


图 1-7　winhello 应用程序执行结果

可见执行结果不再出现在原始的显示屏上，应用程序有了自己的显示窗口，而且显示窗口中还有控制菜单，针对用户的操作可以产生消息，并对消息做相应的响应。但是这个应用程序功能仍很简单，不能显示较复杂的结果。例如，若需要在窗口中显示 20 次“Hello World”信息，对应用程序稍作调整后，其执行结果如图 1-8 所示。

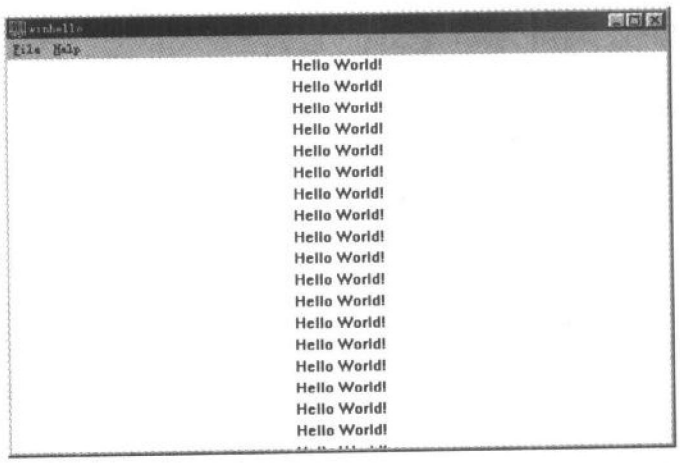


图 1-8　显示多次 “Hello World”信息的 winhello 应用程序执行结果

这个窗口仍不能看到完整的输出结果，而且无法通过按键或其他方式调整输出状态，因为这里根本就没有窗口滚动条，所以这仍是一个不完整的窗口。但与控制台应用程序相比，已经有了很大的进步。

下面分析这个初具规模的 Win32 应用程序。

Win32 程序分析

首先在 Project Workspace 窗口中选择 Class View 页，目前应用程序中实际上并没有类的定义，只有一些全局变量和全局函数，如图 1-9 所示。

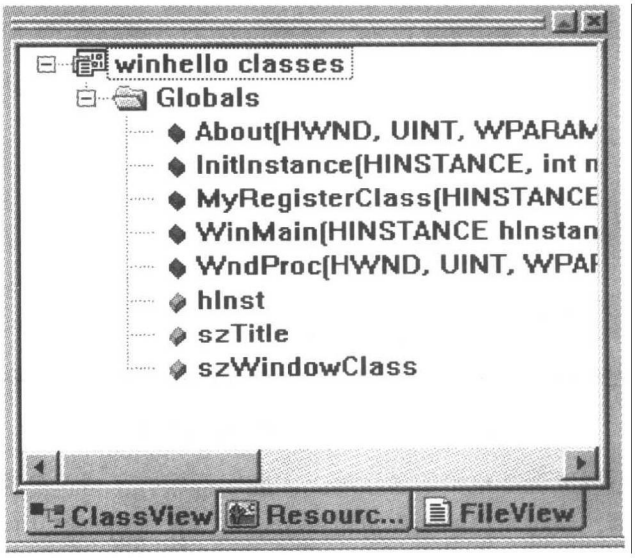


图 1-9　win32 程序分析

然后来分析函数 WinMain，它与控制台应用程序中的函数 main 一样，WinMain 在 Win32 应用程序中起到程序入口的作用。其定义如下：

```
int APIENTRY WinMain(HINSTANCE hInstance,          //实例句柄，标识当前实例
                     HINSTANCE hPrevInstance,      //实例句柄，标识上一个实例
                     LPSTR     lpCmdLine,          //命令行
                     int       nCmdShow)           //主窗口显示方式
{
  MSG msg;                                         //消息变量
  HACCEL hAccelTable;                              //快捷键表
  //设置全局字符串
  LoadString(hInstance, IDS_APP_TITLE, szTitle, MAX_LOADSTRING);
  LoadString(hInstance, IDC_WINHELLO, szWindowClass, MAX_LOADSTRING);
  //实现应用实例的登记
  MyRegisterClass(hInstance);
  // 初始化应用程序
  if (!InitInstance (hInstance, nCmdShow))
  {
    return FALSE;
  }
  //调入快捷键表
```