

---

## 《深入淺出 MFC》2/e 電子書開放自由下載聲明

### 致親愛的大陸讀者

我是侯捷（侯俊傑）。自從華中理工大學於 1998/04 出版了我的《深入淺出 MFC》1/e 簡體版（易名《深入淺出 Windows MFC 程序設計》）之後，陸陸續續我收到了許許多多的大陸讀者來函。其中對我的讚美、感謝、關懷、殷殷垂詢，讓我非常感動。

《深入淺出 MFC》2/e 早已於 1998/05 於臺灣出版。之所以遲遲沒有授權給大陸進行簡體翻譯，原因我曾於回覆讀者的時候說過很多遍。我在此再說一次。

1998 年中，本書之發行公司松崗（UNALIS）即希望我授權簡體版，然因當時我已在構思 3/e，預判 3/e 繁體版出版時，2/e 簡體版恐怕還未能完成。老是讓大陸讀者慢一步看到我的書，令我至感難過，所以便請松崗公司不要進行 2/e 簡體版之授權，直接等 3/e 出版後再動作。沒想到一拖經年，我的 3/e 寫作計劃並沒有如期完成，致使大陸讀者反而沒有《深入淺出 MFC》2/e 簡體版可看。

《深入淺出 MFC》3/e 沒有如期完成的原因是，MFC 本體架構並沒有什麼大改變。《深入淺出 MFC》2/e 書中所論之工具及程式碼雖採用 VC5+MFC42，仍適用於目前的 VC6+MFC421（唯，工具之畫面或功能可能有些微變化）。

由於《深入淺出 MFC》2/e 並無簡體版，因此我時時收到大陸讀者來信詢問購買繁體版之管道。一來我不知道是否臺灣出版公司有提供海外郵購或電購，二來即使有，想必帶給大家很大的麻煩，三來兩岸消費水平之差異帶給大陸讀者的負擔，亦令我深感不安。

---

因此，此書雖已出版兩年，鑑於仍具閱讀與技術上的價值，鑑於繁簡轉譯製作上的費時費工，鑑於我對同胞的感情，我決定開放此書內容，供各位免費閱讀。我已為《深入淺出 MFC》2/e 製作了 PDF 格式之電子檔，放在 <http://www.jjhou.com> 供自由下載。北京 <http://expert.csdn.net/jjhou> 有侯捷網站的一個 GBK mirror，各位也可試著自該處下載。

我所做的這份電子書是繁體版，我沒有精力與時間將它轉為簡體。這已是我能為各位盡力的極限。如果（萬一）您看不到檔案內容，可能與字形的安裝有關——雖然我已嘗試內嵌字形。anyway，閱讀方面的問題我亦沒有精力與時間為您解決。請各位自行開闢討論區，彼此交換閱讀此電子書的 solution。請熱心的讀者告訴我您閱讀成功與否，以及網上討論區（如有的話）在哪裡。

曾有讀者告訴我，《深入淺出 MFC》1/e 簡體版在大陸被掃描上網。亦有讀者告訴我，大陸某些書籍明顯對本書侵權（詳細情況我不清楚）。這種不尊重作者的行為，我雖感遺憾，並沒有太大的震驚或難過。一個社會的進化，終究是一步一步衍化而來。臺灣也曾經走過相同的階段。但盼所有華人，尤其是我們從事智慧財產行為者，都能夠儘快走過灰暗的一面。

在現代科技的協助下，文件影印、檔案複製如此方便，智財權之尊重有如「君子不欺暗室」。沒有人知道我們私下的行為，只有我們自己心知肚明。《深入淺出 MFC》2/e 雖免費供大家閱讀，但此種作法實非長久之計。為計久長，我們應該尊重作家、尊重智財，以良好（至少不差）的環境培養有實力的優秀技術作家，如此才有源源不斷的好書可看。

我的近況，我的作品，我的計劃，各位可從前述兩個網址獲得。歡迎各位寫信給我（[jjhou@ccca.nctu.edu.tw](mailto:jjhou@ccca.nctu.edu.tw)）。雖然不一定能夠每封來函都回覆，但是我樂於知道讀者的任何點點滴滴。

## 關於《深入淺出 MFC》2/e 電子書

《深入淺出 MFC》2/e 電子書共有五個檔案：

檔名	內容	大小 bytes
dissecting MFC 2/e part1.pdf	chap1~chap3	3,384,209
dissecting MFC 2/e part2.pdf	chap4	2,448,990
dissecting MFC 2/e part3.pdf	chap5~chap7	2,158,594
dissecting MFC 2/e part4.pdf	chap8~chap16	5,171,266
dissecting MFC 2/e part5.pdf	appendix A,B,C,D	1,527,111

每個檔案都可個別閱讀。每個檔案都有書籤（亦即目錄連結）。每個檔案都不需密碼即可開啓、選擇文字、列印。

## 請告訴我您的資料

每一位下載此份電子書的朋友，我希望您寫一封 email 給我（[jjhou@ccca.nctu.edu.tw](mailto:jjhou@ccca.nctu.edu.tw)），告訴我您的以下資料，俾讓我對我的讀者有一些基本瞭解，謝謝。

姓名：

現職：

畢業學校科系：

年齡：

性別：

居住省份（如是臺灣讀者，請寫縣市）：

對侯捷的建議：

-- the end



欲흥 그 후 20년



深入淺出 MFC  
2nd Edition



## Visual C++ 整合開發環境

如果 MFC 是箭，Visual C++ IDE（整合開發環境）便是弓。

強壯的弓，讓箭飛得更遠。

看過重量級戰鬥嗎？重量級戰鬥都有「一棒擊沉」的威力。如果現實生活中發生重量級戰鬥 -- 使人生涯結束、生活受威脅的那種，那麼戰況之激烈不言可知。如果這場戰鬥關係到你的程式員生涯，鈴聲響起時你最好付出高度注意力。

我說的是 application framework。換個角度來說，我指的是整合型（全套服務的）C++ 軟體開發平台。目前，所有重要廠商包括 Microsoft、Borland、Symantec、Metaware 和 Watcom 都已投入這個戰場。在 PC 領域，最著名的 application framework 有兩套（註）：MFC（Microsoft Foundation Class）和 OWL（ObjectWindow Library），但整合開發環境（IDE）卻呈百家爭鳴之勢。

註：第三套可以說是 IBM VisualAge C++ 的 Open Class Library。VisualAge C++ 和 Open Class Library 不單是 OS/2 上的產品，IBM 更企圖讓它們橫跨 Windows 世界。

在這一章中，我將以概觀的方式為你介紹 Visual C++ 的整合環境，目的在認識搭配在 MFC 週遭的這些強棒工具的操作性與功能性，實地了解這一整套服務帶給我們什麼樣的便利。除非你要以你的 PE2 老古董把程式一字一句 co co co 地敲下去，否則 Visual C++ 的這些工具對軟體開發的重要性不亞於 MFC。我所使用的 Visual C++ 版本是 v5.0（搭配 MFC 4.21）。

## 安裝與組裝

VC++ 5.0 採 CD-ROM 包裝，這是現代軟體日愈肥胖後的趨勢。記憶體最好有 16MB，跑起來才會舒服些；硬碟空間的需求量視不同的安裝方式（圖 4-1f）而定，你可以從畫面上清楚看到；只要硬碟夠大，我當然建議採用 Typical Installation。

Visual C++ 5.0 光碟片中有 AUTORUN.INF 檔，所以其 Setup 程式會在 Windows 95 和 Windows NT 4.0 的 autoplay 功能下自動執行。Setup 程式會偵測你的環境，如果沒有找到 Internet Explorer (IE) 3.01，它會建議你安裝或更新之（圖 4-1a）。VC++ 5.0 碟片中附有 IE 3.01 (英文版)。為什麼要先安裝 Internet Explorer 呢？因為微軟的所有 Visual Tools（包括 Visual C++、Visual Basic、Visual FoxPro、Visual J++、Visual InterDev 等）都集中由所謂的 Visual Studio（圖 4-1c）管理，而這些工具有一個極大的目標，就是要協助開發 Internet 應用軟體，所以它們希望能夠和 Internet Explorer 有所搭配。

如果你原已有 Visual C++ 4.x，Setup 程式會偵測到並給你一個警告訊息（圖 4-1e）。通常你可能會想保留原有的版本並試用新的版本（至少我的心態是如此），因此你可能擔心 Visual C++ 5.0 會不會覆蓋掉 4.x 版。放心，只要你在圖 4-1f 中指定安裝目的地（子目錄）和原版本不同，即可避免所謂覆蓋的問題。以我的情況為例，我的 Visual C++ 4.2 放在 E:\MSDEV 中，而我的 Visual C++ 5.0 安裝在 E:\DEVSTUDIO 中。



圖 4-1a Visual C++ 5.0 建議你安裝最新的 IE 3.01（英文版）。



圖 4-1b 當你安裝 IE 3.01 (英文版) 時，可能會和你現有的 IE 中文版有些版本衝突。我的經驗是依其建議，保留現有的檔案。

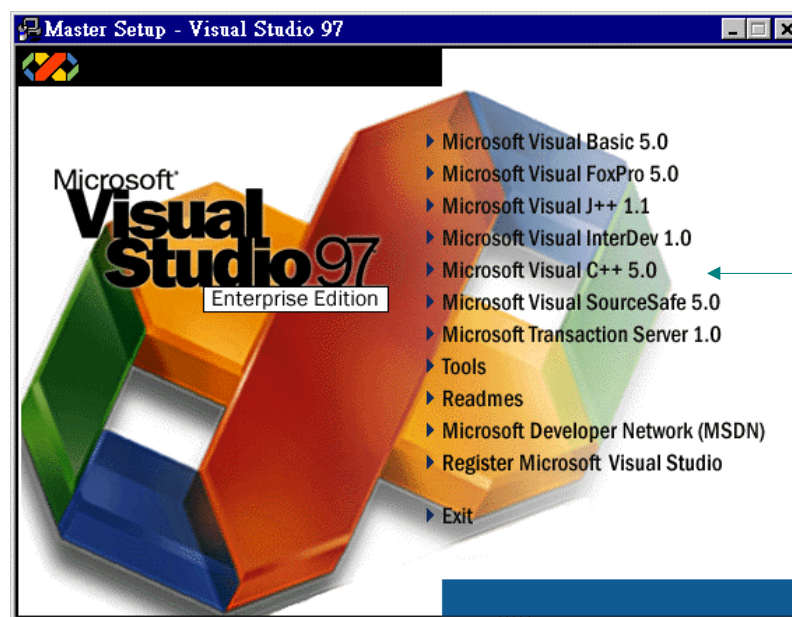


圖 4-1c Visual C++ 5.0 Setup 程式畫面。請把滑鼠移到右上角第五個項目 "Microsoft Visual C++ 5.0" 上面，並按下左鍵。



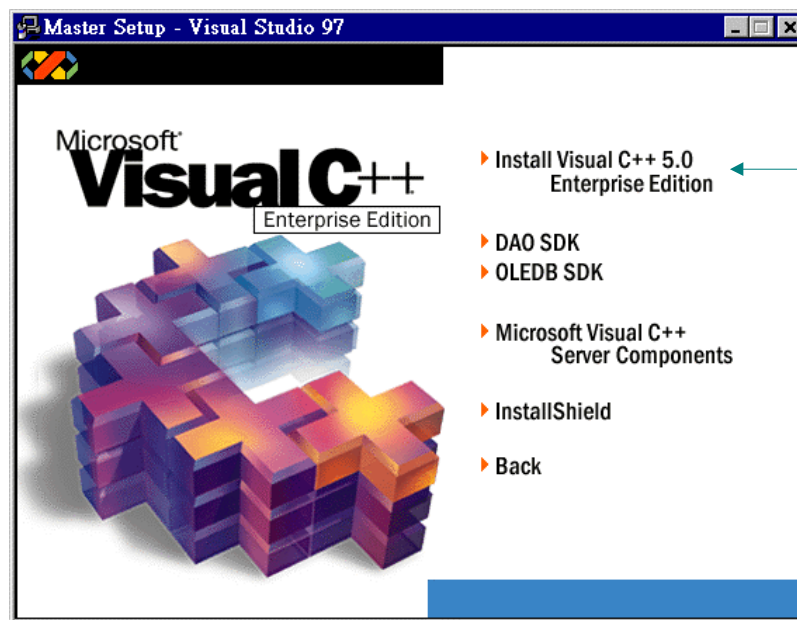


圖 4-1d 你可以安裝 Visual C++ 5.0 中的這些套件。其中 InstallShield 是一套協助你製作安裝軟體的工具。

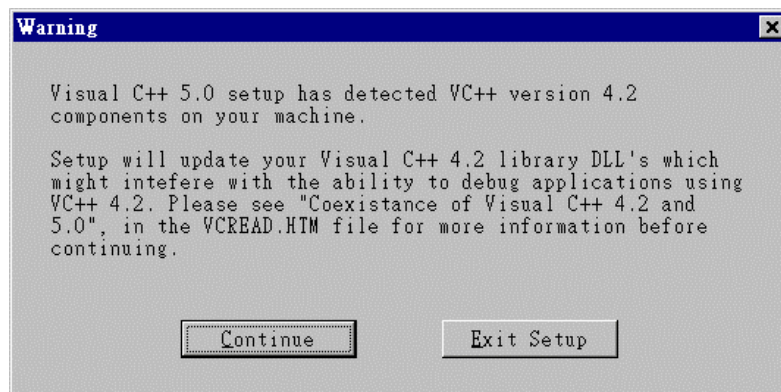


圖 4-1e Setup 程式偵測到我已經有 Visual C++ 4.2，於是提出警告。

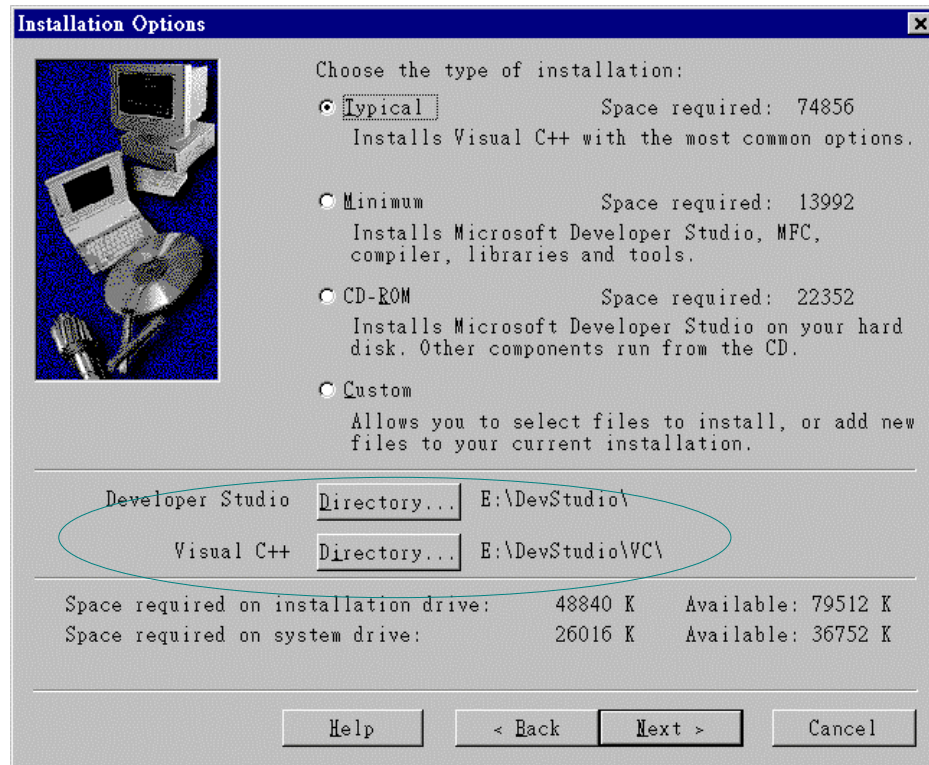


圖 4-1f Visual C++ 提供四種安裝方式。中央偏下的【Directory...】鈕允許我們設定安裝目的地（硬碟目錄）。

早期的 Visual C++ 版本曾經要求你在 AUTOEXEC.BAT 中加入這行命令：

```
SHARE /L:500 /F:5100
```

為的是讓 DOS 藉著 SHARE.EXE 的幫助支援「檔案共用與鎖定功能」。如今已不需要，因為 Windows 95 及 Windows NT 已內建此項能力。

這個整合環境並不要求你設定什麼環境變數，它自己內部會在安裝時記錄該有的路徑。如果你習慣以命令列的方式在 DOS 環境（也就是 Windows 95 或 Windows NT 的 DOS 視窗）下編譯連結，那麼你必須小心設定好 PATH、LIB、INCLUDE 等環境變數。如果你有許多套開發工具，為每一個環境準備一個批次檔是個不錯的作法。下面是個例子：

```
rem file : envir.bat
cls
type c:\utility\envir.txt
```

其中 `envir.txt` 的內容是：

```
(1) CWin95 & Visual C++ 1.5
(2) CWin95 & Visual C++ 2.0
(3) CWin95 & Visual C++ 4.0
(4) DDK
(5) CWin95 & Visual C++ 5.0
```

每當欲使用不同的工具環境，就執行 `envir.bat`，然後再選擇一個號碼。舉個例，`3.BAT` 的內容是：

```
rem 3.bat
rem Win95 & Visual C++ 4.0
@echo off
set TOOLROOTDIR=E:\MSDEV
rem
set PATH=E:\MSDEV\BIN;D:\WIN95;D:\WIN95\COMMAND
set INCLUDE=E:\MSDEV\INCLUDE;E:\MSDEV\MFC\INCLUDE
set LIB=E:\MSDEV\LIB;E:\MSDEV\MFC\LIB
set MSDevDir=E:\MSDEV
set
```

`5.BAT` 的內容是：

```
rem e:\devstudio\vc\bin\vcvars32.bat
@echo off
rem
rem e:\devstu~1 == e:\devstudio
set PATH=E:\DEVSTU~1\VC\BIN;E:\DEVSTU~1\SHARED~1\BIN;D:\WIN95;D:\WIN95\COMMAND
set INCLUDE=E:\DEVSTU~1\VC\INCLUDE;E:\DEVSTU~1\VC\MFC\INCLUDE;E:\DEVSTU~1\VC\ATL\INCLUDE
set LIB=E:\DEVSTU~1\VC\LIB;E:\DEVSTU~1\VC\MFC\LIB
set
```

其中大家比較陌生的可能是 `VC\ATL\INCLUDE` 這個設定。ATL 全名是 ActiveX Template Library，用以協助我們開發 ActiveX 控制元件。關於 ActiveX 控制元件的開發設計，可參考 *ActiveX Control Inside Out* (Adam Denning/Microsoft Press) 一書 (**ActiveX 控制元件徹底研究** / 侯俊傑譯 / 松崗出版)。至於 ActiveX controls 的應用，可參考本書第 16 章。

上述那些那些環境變數的設定，其實 VC++ 早已為我們準備好了，就放在 \DEVSTUDIO\VC\BIN\VCVARS32.BAT 中，只不過形式比較複雜一些。

如果你也喜歡（或有必要）保留多套開發環境於硬碟中，請注意出現在 DOS 提示號下的編譯器和聯結器版本號碼，以確定你叫用的的確是你所要的工具。圖 4-2 是 Microsoft 軟體開發工具的版本號碼。

VC++	編譯器	聯結器	NMAKE	RC.EXE	MFC
Microsoft C/C++ 7.0	7.00	S5.30	1.20	3.10	1.0
Visual C++ 1.0	8.00	S5.50	1.30	3.11	2.0
Visual C++ 1.5x	8.00c	S5.60	1.40	3.11	2.5
Visual C++ 2.0	9.00	I2.50	1.50	3.50	3.0
Visual C++ 4.0	10.00	I3.00	1.60	4.00	4.0
Visual C++ 4.2	10.20	I4.20	1.61	4.00	4.2
Visual C++ 5.0	11.00	I5.00	1.62	5.00	4.21

\* 聯結器 S: Segmented Executable Linker  
I: Incremental Linker

圖 4-2 Microsoft 編譯器平台的演化

Visual C++ 提供三種版本：學習版、專業版和企業版。三者都提供 C/C++ 編譯器、MFC、以及整合開發環境，可以協助建立並除錯各類型應用軟體：

- MFC-based EXE
- MFC-based DLL
- Win32 Application (EXE)
- Win32 Dynamic Link Library (DLL)
- Win32 Console Applications
- MFC ActiveX Controls

- ATL COM (ActiveX Template Library Component Object Model)
- ISAPI (Internet Server API) Extension Application
- Win32 Static Library

圖 4-3 是 VC++ 5.0 專業版安裝完成後的程式群組，打開 Win95 的【開始/程式集】便可看到。

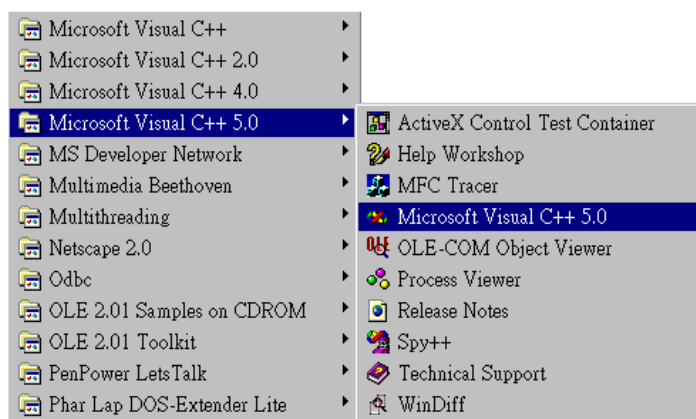


圖 4-3 VC++ 5.0 專業版安裝完成後的程式群組 (group)

VC++ 5.0 安裝完成後重要的檔案分佈如下。可能有些在你的硬碟，有些在光碟片上，因不同的安裝方式而異：

```

MSDEV <DIR>
  BIN <DIR>          各種 EXE、BAT、DLL。
  DEBUG <DIR>        MFC 除錯版本（各種 DLLs）。
  HELP <DIR>         各種 Help 檔案。
  CRT <DIR>          C runtime 函式庫的原始碼。
  ATL <DIR>          ActiveX Template Library
    INCLUDE <DIR>    ATL 的含入檔（表頭檔）
    SRC <DIR>        ATL 的原始碼
  REDIST <DIR>       這是可以自由（免費）傳播的檔案，包括你的應用程式售出後，
                    執行時期所需的任何 DLLs，如 MFC42.DLL、ODBC DLLs、
                    DAO DLLs。還包括微軟公司附贈的一些 OCXs。

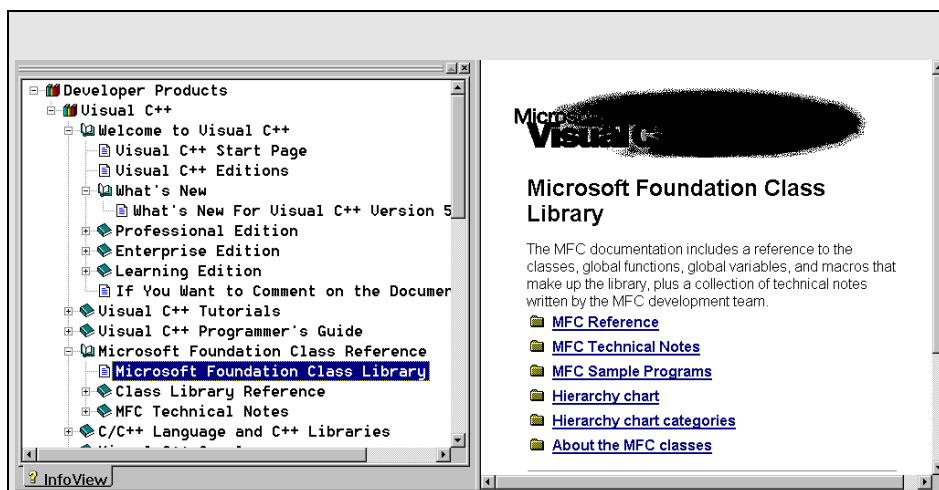
SAMPLES <DIR>       豐富的範例程式（請看附錄 C）
  APPWIZ <DIR>
  ATL <DIR>
  COM <DIR>
  ENT <DIR>
  MFC <DIR>
  SDK <DIR>

INCLUDE <DIR>       各種 .H 檔案。包括 C/C++ 函式表頭檔、WINDOWS.H 等等。
LIB <DIR>            各種 .LIB。包括 C/C++ runtime、Windows DLLs import 函式庫。
MFC <DIR>
  INCLUDE <DIR>     以 AFX 開頭的 .H 檔案（MFC 的表頭檔）。
  LIB <DIR>         MFC 的靜態函式庫（static library）。
  SRC <DIR>         MFC 的原始碼（.CPP 檔）。

```

手冊呢？C/C++ 加上 SDK 再加上 MFC 共二十來本厚薄不一的手冊不可能塞到寬僅五公分的 VC++ 5.0 包裝盒中。所有的手冊都已電子化到那片 CD-ROM 去了。像我這種看書一定得拿支筆的人，沒什麼比這更悲哀的事。不是沒有補救辦法，再花個數千元就可得到 VC++ 印刷手冊，另一個數千元可再得到 SDK 印刷手冊。

## MFC Tech Notes



VC++ 5.0 的 Online Help 中有一些好東西：為數 69 篇的寶貴技術文件。以下是一份列表。文件 1 至 17 是一般性主題，適用於 MFC 1.0 和 2.0；文件 18 和 19 專注在如何將 MFC 1.0 程式移植到 MFC 2.0；文件 20 至 36 適用於 MFC 2.0（或更高版本）；文件 37 適用於 32 位元版 MFC；文件 38 至 48 適用於 MFC 2.5（或更高版本）；文件 49 至 52 適用於 MFC 3.0（或更高版本）；文件 53 至 69 適用於 MFC 4.0（或更高版本）。某些號碼跳掉是因為 MFC 1.0 的老東西不值得再提。

1. Window Class Registration
2. Persistent Object Data Format
3. Mapping of Windows Handles to Objects
4. C++ Template Tool
6. Message Maps
7. Debugging Trace Options
8. MFC OLE Support
11. Using MFC as Part of a DLL
12. Using Windows 3.1 Robustness Features

14. Custom Controls
15. Windows for Pen
16. Using C++ Multiple Inheritance with MFC
17. Destroying Window Objects
18. Migrating OLE Applications From MFC 1.0 to MFC 2.0
19. Migrating MFC 1.0 Applications to MFC 2.0
20. ID Naming and Numbering Conventions
21. Command and Message Routing
22. Standard Commands Implementation
23. Standard MFC Resources
24. MFC-Defined Messages and Resources
25. Document, View, and Frame Creation
26. DDX and DDV Routines
27. Emulation Support for Visual Basic Custom Controls
28. Context-Sensitive Help Support
29. Splitter Windows
30. Print Preview
31. Control Bars
32. MFC Exception Mechanism
33. DLL Version of MFC
34. Writing a Windows 3.0 Compatible MFC Application
35. Using Multiple Resource Files and Header Files with App Studio
36. Using CFormView with AppWizard and ClassWizard
37. Multithreaded MFC 2.1 Applications ( 32-bit specific)
38. MFC/OLE IUnknown Implementation
39. MFC/OLE Automation Implementation
40. MFC/OLE In-Place Resizing and Zooming
41. MFC/OLE1 Migration to MFC/OLE2
42. ODBC Driver Developer Recommendations
43. RFX Routines
44. MFC support for DBCS
45. MFC/Database support for Long Varchar/Varbinary
46. Commenting Conventions for the MFC classes



47. Relaxing Database Transaction Requirements
48. Writing ODBC Setup and Administration Programs for MFC Database Applications
49. MFC/OLE MBCS to Unicode Translation Layer (MFCANS32)
50. MFC/OLE Common Dialogs (MFCUIx32)
51. Using CTL3D Now and in the Future
52. Writing Windows 95 Applications with MFC 3.1
53. Custom DFX Routings for DAO Database Classes
54. Calling DAO Directory while Using MFC DAO Classes
55. Migrating MFC ODBC Database Classes Application to MFC DAO Classes
56. Installation of MFC Components
57. Localization of MFC Components
58. MFC Module State Implementation
59. Using MFC MBCS/Unicode Conversion Macros
60. The New Windows Common Controls
61. ON\_NOTIFY and WM\_NOTIFY Messages
62. Message Reflection for Windows Controls
63. Debugging Internet Extension DLLs
64. Apartment-Model Threading in OLE Controls
65. Dual-Interface Support for OLE Automation Servers
66. Common MFC 3.x to 4.0 Porting Issues
67. Database Access from an ISAPI Server Extension
68. Performing Transactions with the Microsoft Access 7 ODBC Driver
69. Processing HTML Forms Using Internet Server Extension DLLs and Command Handlers

以下是 MFC Tech Notes 的性質分類：

■ MFC and Windows

TN001: Window Class Registration

TN003: Mapping of Windows Handles to Objects

TN012: Using MFC with Windows 3.1 Robustness Features

TN015: Windows for Pen

TN017: Destroying Window Objects

TN034: Writing a Windows 3.0 Compatible MFC Application

TN051: Using CTL3D Now and in the Future

TN052: Writing Windows 95 Applications with MFC3.1

#### ■ MFC Architecture

TN002: Persistent Object Data Format

TN004: C++ Template Tool

TN006: Message Maps

TN016: Using C++ Multiple Inheritance with MFC

TN019: Updating Existing MFC Applications to MFC 3.0

TN021: Command and Message Routing

TN022: Standard Commands Implementation

TN025: Document, View, and Frame Creation

TN026: DDX and DDV Routines

TN029: Splitter Windows

TN030: Customizing Printing and Print Preview

TN031: Control Bars

TN032: MFC Exception Mechanism

TN037: Multithreaded MFC 2.1 Applications

TN044: MFC Support for DBCS

TN046: Commenting Conventions for the MFC Classes

TN058: MFC Module State Implementation

TN059: Using MFC MBCS/Unicode Conversion Macros

TN066: Common MFC 3.x to 4.0 Porting Issues

#### ■ MFC Controls

TN014: Custom Controls

TN027: Emulation Support for Visual Basic Custom Controls

TN060: Windows Common Controls

TN061: ON\_NOTIFY and WM\_NOTIFY Messages

TN062: Message Reflection for Windows Controls

#### ■ MFC Database

TN042: ODBC Driver Developer Recommendations  
TN043: RFX Routines  
TN045: MFC/Database Support for Long Varchar/Varbinary  
TN047: Relaxing Database Transaction Requirements  
TN048: Writing ODBC Setup and Administration Programs for MFC Database Applications  
TN053: Custom DFX Routines for MFC DAO Classes  
TN054: Calling DAO Directly While Using MFC DAO Classes  
TN055: Migrating MFC ODBC Database Class Applications to MFC DAO Classes  
TN068: Performing Transactions with the Microsoft Access 7 ODBC Driver

■ MFC Debugging

TN007: Debugging Trace Options

■ MFC DLLs

TN011: Using MFC as Part of a DLL  
TN033: DLL Version of MFC  
TN056: Installation of MFC Components  
TN057: Localization of MFC Components

■ MFC OLE

TN008: MFC OLE Support  
TN018: Migrating OLE Applications from MFC 1.0 to MFC 2.0  
TN038: MFC/OLE IUnknown Implementation  
TN039: MFC/OLE Automation Implementation  
TN040: MFC/OLE In-Place Resizing and Zooming  
TN041: MFC/OLE1 Migration to MFC/OLE2  
TN049: MFC/OLE MBCS to Unicode Translation Layer (MFCANS32)  
TN050: MFC/OLE Common Dialogs (MFCUIx32)  
TN064: Apartment-Model Threading in OLE Controls  
TN065: Dual-Interface Support for OLE Automation Servers

■ MFC Resources

TN020: ID Naming and Numbering Conventions

TN023: Standard MFC Resources

TN024: MFC-Defined Messages and Resources

TN028: Context-Sensitive Help Support

TN035: Using Multiple Resource Files and Header Files with Visual C++

TN036: Using CFormView with AppWizard and ClassWizard

■ MFC Internet

TN063: Debugging Internet Extension DLLs

TN067: Database Access from an ISAPI Server Extension

TN069: Processing HTML Forms Using Internet Server Extension DLLs and  
Command Handlers

## 四個重要的工具

完全依賴整合環境，丟掉 PE2（或其他什麼老古董），這是我的良心建議。也許各個工具的學習過程會有些陣痛，但代價十分值得。我們先對最重要的四個工具作全盤性了解，再進去巡幽訪勝一番。你總要先強記一下哪個工具做什麼用，別把馮京當馬涼，張飛戰岳飛，往後的文字看起來才會順暢。

圖 4-4 是 MFC 程式的設計流程。

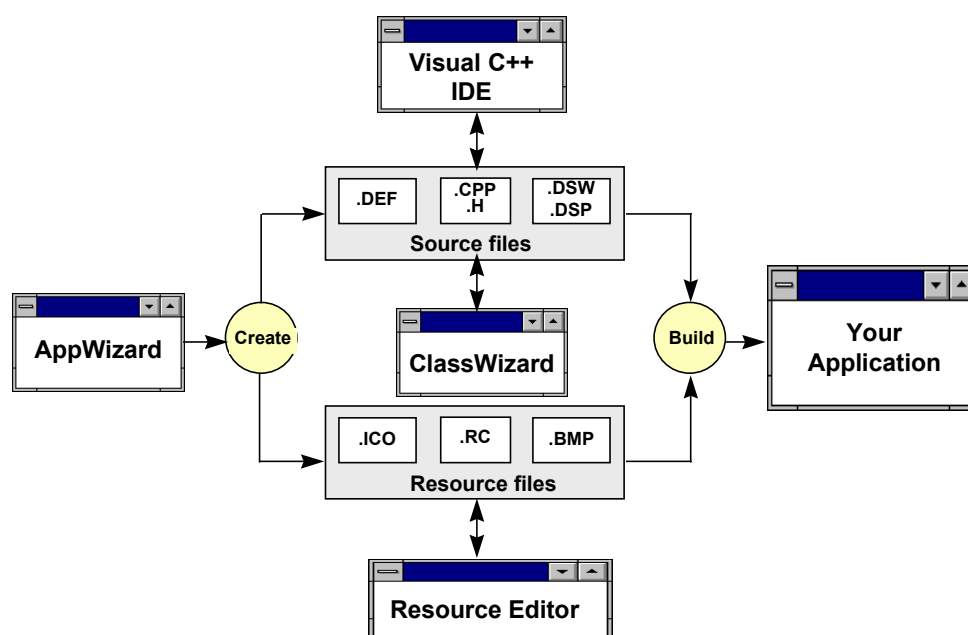


圖 4-4 MFC 程式的開發流程

- **Visual C++ 整合開發環境 (IDE)**：你可以從中明顯地或隱喻地啟動其他工具如 AppWizard 和 ClassWizard；你可以設定各種工具、編譯並聯結程式、啟動除錯器、啟動文字編輯器、瀏覽類別階層...。
- **AppWizard**：這是一個程式碼產生器。基於 application framework 的觀念，相同型態（或說風格）的 MFC 程式一定具備相同的程式骨幹，AppWizard 讓你挑選菜色（利用滑鼠圈圈選選），也為你炒菜炒出來（產生各種必要檔案）。別忘記，化學反應是不能夠還原的，菜炒好了可不能反悔（只能加油添醋），所以下手前需三思 -- 每一個 project 使用 AppWizard 的機會只有一次。
- **Resource Editor**：這是一個總合資源編輯器，RC 檔內的各種資源它統統都有辦法處理。Resource Editor 做出來的各類資源與你的程式碼之間如何維繫關係？譬如說對話盒中的一個控制元件被按下後程式該有什麼反應？這就要靠 ClassWizard 搭起鵲橋。
- **ClassWizard**：AppWizard 製作出來的程式骨幹是「起手無悔」的，接下來你只能夠在程式碼中加油添醋（最重要的工作是加上自己的成員變數並改寫虛擬函式），或搭起訊息與程式碼之間的鵲橋（建立 Message Map），這全得仰仗 ClassWizard。以一般文字編輯器直接修改程式碼當然也可以，但你的思維必須非常縝密才不會掛一漏萬。本書第四篇，當我們逐漸發展一個實用程式，你就會看到 ClassWizard 的好處。

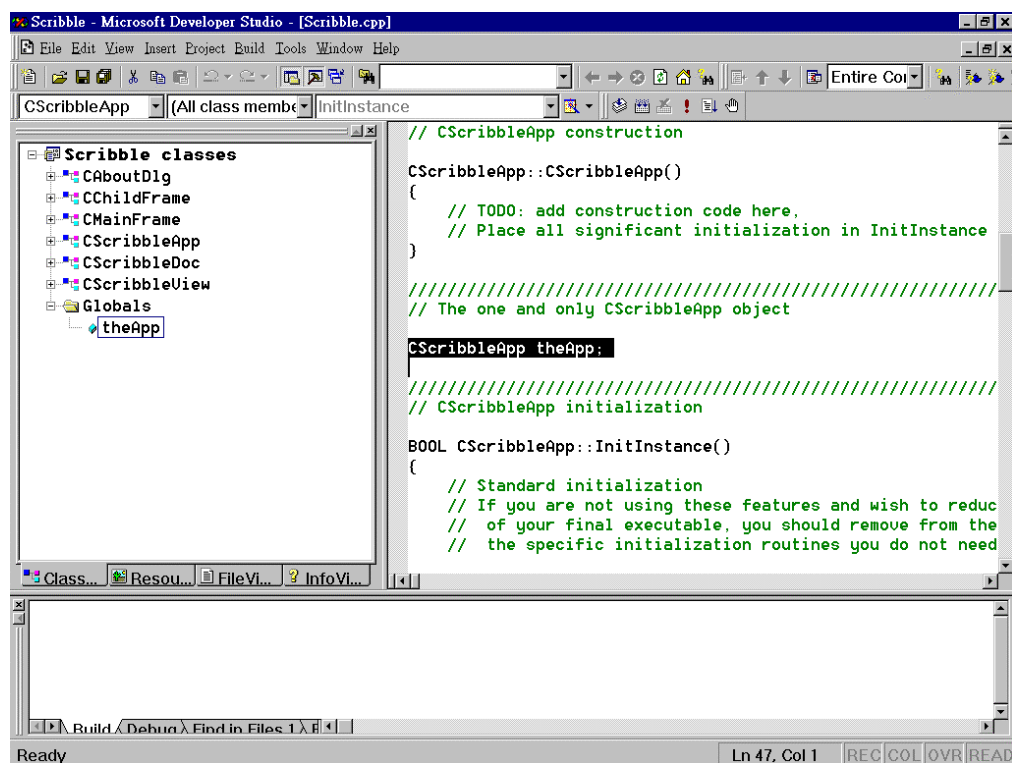
## 事務總管：Visual C++ 整合開發環境

做為一個總管，要處理的大小小事務很多。本章並不是 Visual C++ 的完整使用手冊，並不細部操作解說（完整手冊可參考 Online Help 中的 *Visual C++ User's Guide*）。基本上，如果你一邊看這些文字說明一邊實際玩玩這些工具，馬上會有深刻的印象。

以功能選單來分類，大致上 Visual C++ 整合環境有以下功能：

- **File** - 在此開啓或儲存檔案。文字檔開啓於一個文字編輯器中，這個編輯器對程式的撰寫饒有助益，因為不同型態的關鍵字會以不同顏色標示。如果你新開啓的是一個 project，AppWizard 就會暗自啓動（稍後再述）。檔案的列印與印表機的設定也在此。
- **Edit** - 這裡有傳統的剪貼簿(clipboard)功能。文字編輯器的 Find 和 Replace 功能也放在這裡。
- **View** - 對目前正在編輯之檔案的各種設定動作。例如記號 (bookmark) 的設定尋找與清除，關鍵字顏色的設定與否、特定行號的搜尋...等等。ClassWizard 可在此選單中被啓動。
- **Insert** - 可以在目前的 project 中插入新的 classes、resources、ATL objects...。
- **Project** - 可以在此操作 project，例如加入檔案、改變編譯器和聯結器選項等等。
- **Build** - 我們在這裡製作出可執行檔，也在這裡除錯。如果進入除錯模式，Build 會變成 Debug。
- **Tools** - 可以啓動 Browser、MFC Tracer、SPY++ 以及其他工具。
- **Window** - 整合環境 (IDE) 中各大大小小視窗可在此管理。
- **Help** - 線上輔助說明，包括書籍、期刊、文章、範例。有一個不錯的檢索工具。

下面就是 Visual C++ 整合環境 (IDE) 的畫面：



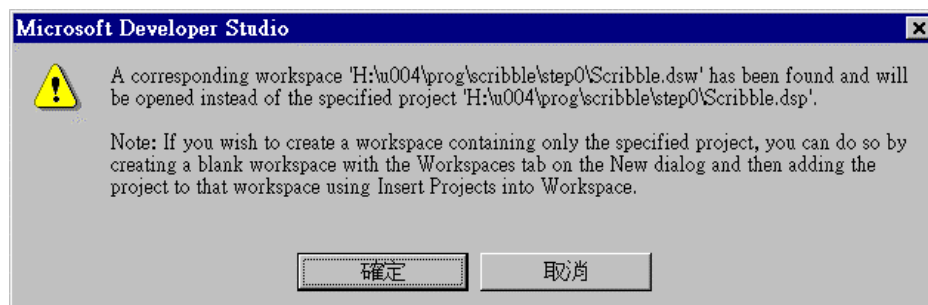
## 關於 project

開發一個程式需要許多檔案，這些檔案以一個 DSW 檔和 DSP 檔（而不再是 VC++ 4.x 時代的 .MDP 檔和 .MAK 檔）規範管理。一整組相關的檔案就是一個 project。只要你告訴 Visual C++ 在哪一個磁碟目錄下開始一個新的 project，它就會為你製作出一個 DSW 檔和一個 DSP 檔。假設我們的專案名稱是 "My"，那麼就得到 MY.DSP 和 MY.DSW。下次你要繼續工作時，在【File/Open】對話盒中打開 MY.DSW 就對了。

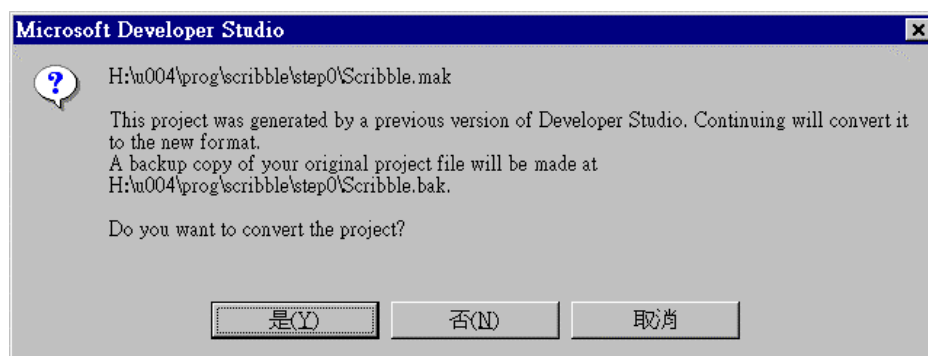
DSP 是 Developer Studio Project 的縮寫，DSW 是 Developer Studio Workspace 的縮寫。Workspace 是 VC++ 整合環境 (IDE) 的一個維護檔，可以把與該 project 有關的 IDE 環境設定都記錄下來。所以，你應該在 VC++ IDE 中選按【File/Open】後打開一個 DSW 檔（而不是 DSP 檔），以開啓 projects。如果你選擇的是 DSP 檔，而同時存在著一個 DSW



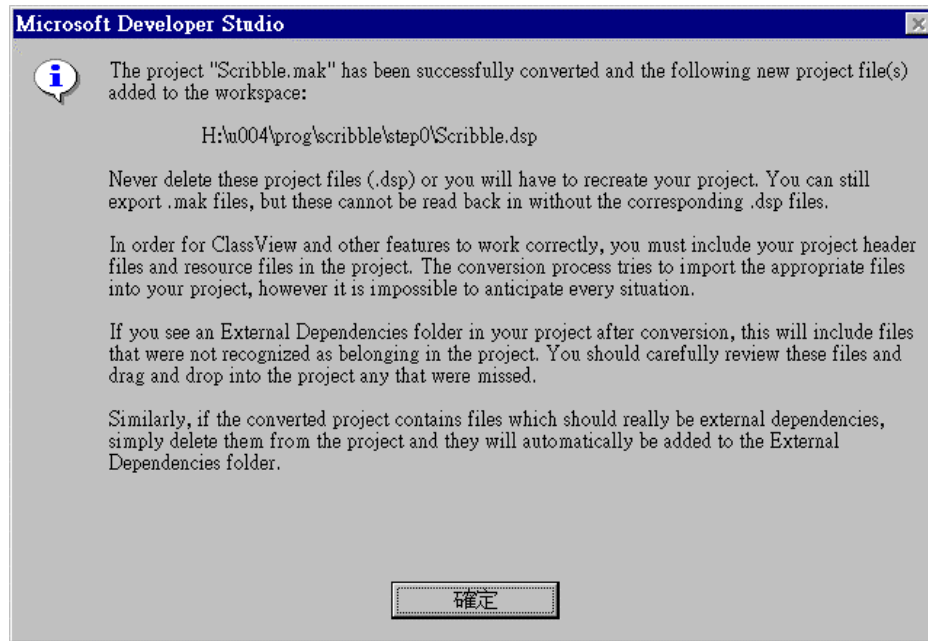
檔，你會獲得這樣的訊息：



VC++ 4.x 的老用戶們請注意，過去代表一個 project 的所謂 .MDP 檔還存在嗎？如果你是以 VC++ 5.0 的 wizards 來產生 project，就不會再看到 .MDP 檔了，取而代之的是上述的 .DSP 檔和 .DSW 檔。如果你在 VC++ 5.0 中開啓過去在 VC++ 4.x 時完成的 project（.MDP 檔），會獲得這樣的訊息：



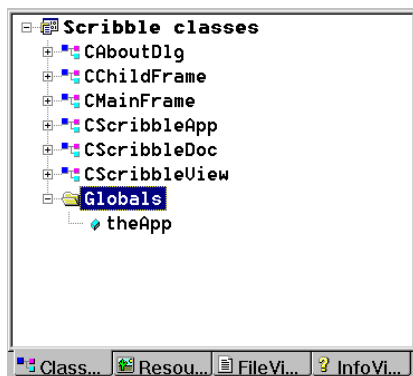
選擇【是】之後，IDE 自動為你轉換，並在完成之後給你這樣的訊息：



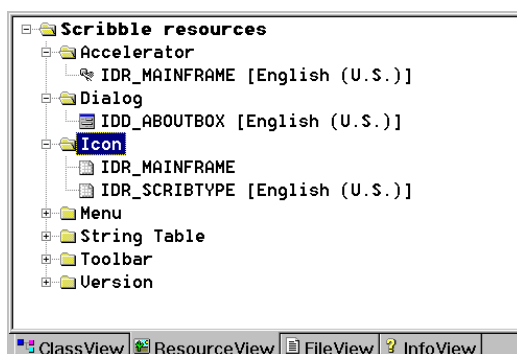
有趣的是，不論 .MDP 檔或 .DSP 檔或 .DSW 檔，我們的 makefile 寫作技巧勢將逐漸萎縮。誰還會自己費心於那些 !\$<@# 等等詭屈聱牙的奇怪符號呢？！這其實是件好事。

當你產生出一個 project（利用 AppWizard，稍後再提），整合環境提供了四個便利的管理視窗：

- ClassView - 可以觀察專案中所有的類別



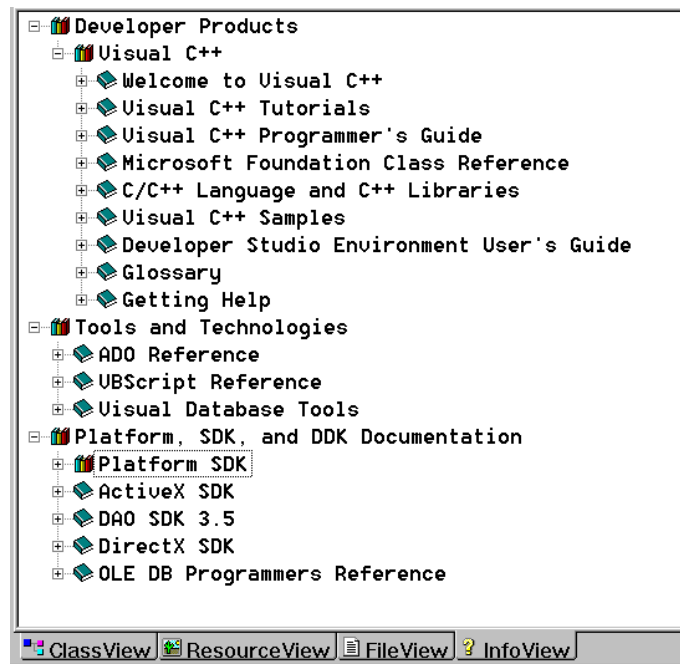
- ResourceView - 可以觀察專案中所有的資源



- FileView - 可以觀察專案中所有的檔案



## ■ InfoView - Online Help 的總目錄



## 關於工具設定

我們當然有機會設定編譯器、連結器和 RC 編譯器的選項。圖 4-5 是兩個設定畫面。

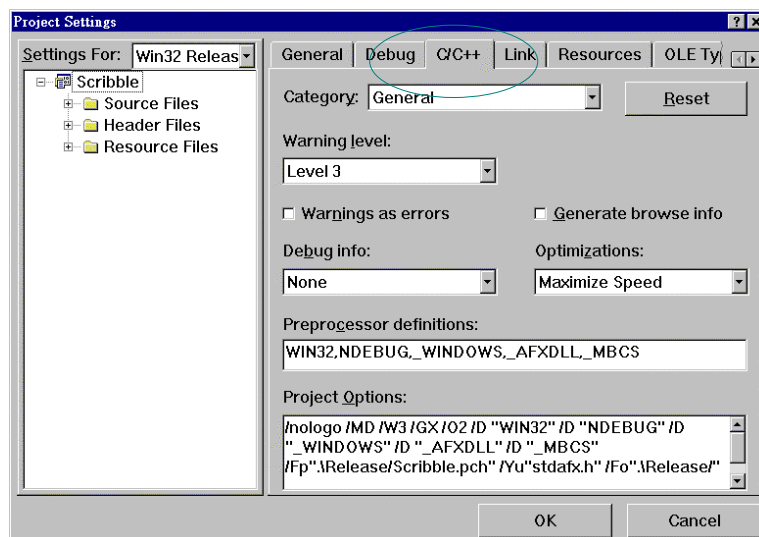


圖 4-5a 選擇 Visual C++ 的【Project/Setting...】，出現對話盒。選擇【C/C++】附頁，於是設定編譯器選項。

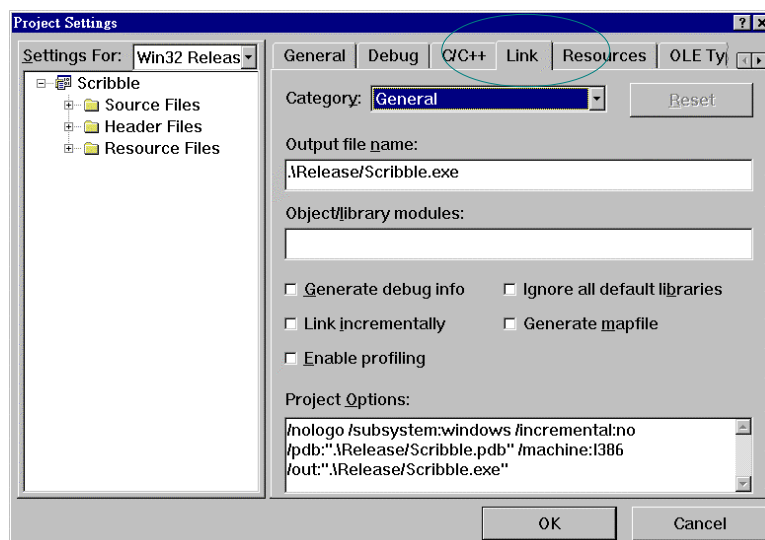


圖 4-5b 選擇 Visual C++ 的【Project/Setting...】，出現對話盒。選擇【Link】附頁，於是設定連結器選項。

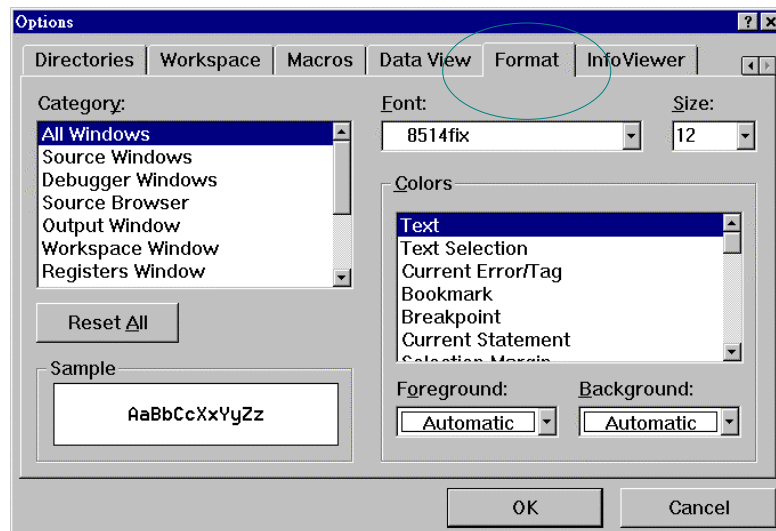
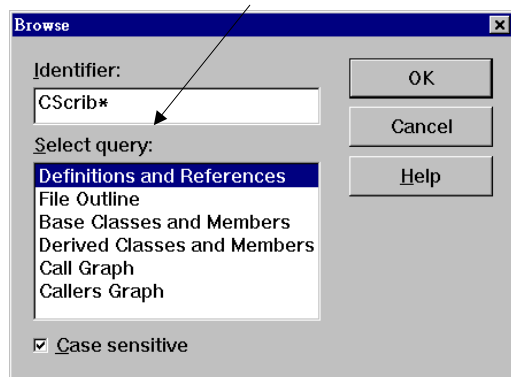


圖 4-5c 選擇 Visual C++ 的【Tools/Options...】，出現對話盒。選擇【Format】附頁，於是設定程式碼編輯器的字型與大小...

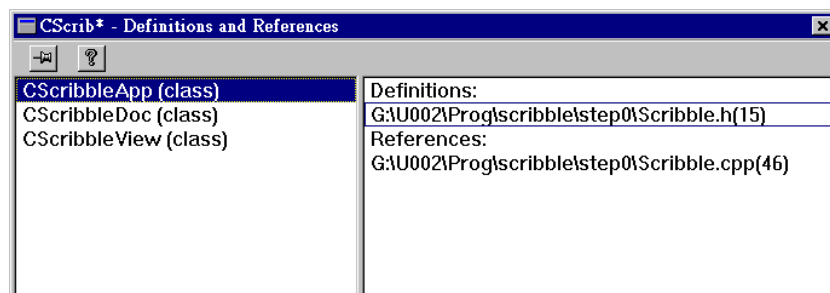
## Source Browser

好的 Browser（瀏覽器）是令人難以置信的一個有用工具，它把你快速帶到任何你所指定的符號（symbol，包括類別、函式、變數、型別或巨集）的出現地點。基本上 Browser 揭露兩件事情：位置（places）和關係（relationship）。它可以顯示某個符號「被定義」以及「被使用到」的任何位置。下面就顯示名為 CScrib\* 的所有類別：

選按 Visual C++ 的【Tools/Source Browse...】選單，出現以下對話盒。在【Identifier】欄位鍵入“CScrib\*”，並在【Select Query】清單中選擇【Definitions and References】：

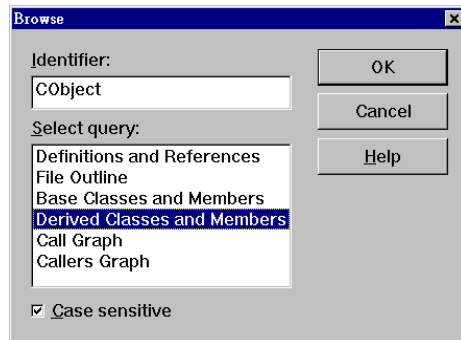


於是啟動 Browser，列出所有名為 CScrib\* 的類別。選擇其中的 CScribbleApp，右鍵单击就會填入所有它出現的位置（包括定義處以及被參考之處）。雙擊其中一個，你立刻置身其中，文字編輯器會跳出來，轉到此檔，準備為你服務。

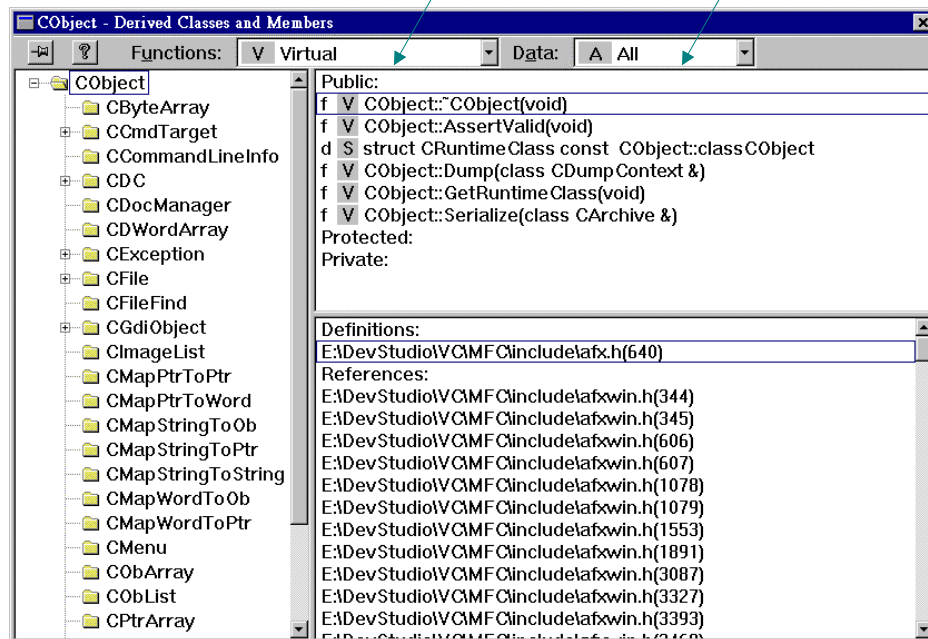


Browser 也揭露類別之間的關係以及類別與函式之間的關係。MFC 類別彼此疊床架屋，只以一般的文字編輯器（或如 grep 之類的文字搜尋器）探索這些關係，就好像划一艘小船橫渡太平洋到美利堅一樣地緩慢而遙遠。Browser 使我們在跋涉類別叢林時節省許多光陰。以下顯示應用程式中所有衍生自 CObject 的類別。

觀察應用程式中所有衍生的 CObject 的類別。請選按 Visual C++ 之【Tools/Source Browse...】選單，出現對話盒。在【Identifier】欄位鍵入 "CObject"，請注意我選擇的【Select Query】清單項目是【Derived Classes and Members】。



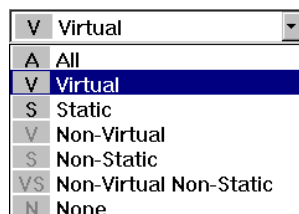
於是獲得 CObject 的所有衍生類別。請注意【Functions】欄是 Virtual，【Data】欄是 All。



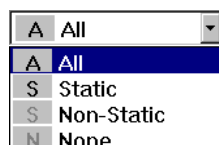
此時 Browser 出現三個窗格，左邊那個不論外觀或行為都像檔案總管裡的目錄樹，右邊兩個窗格顯示你所選定之類別的詳細資訊。



Browser 提供這些【Functions】項目供觀察:



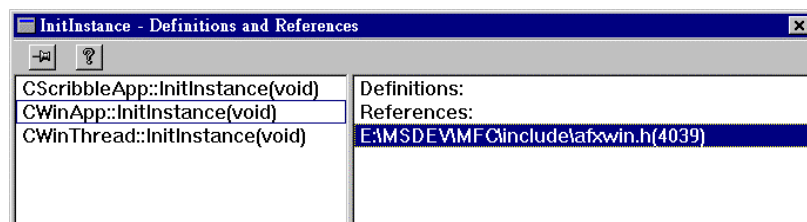
Browser 提供這些【Data】項目供觀察:



Browser 係從一個特殊的資料庫檔案 (.BSC) 取得資訊，此檔由 Visual C++ 整合環境自動產生，非常巨大。如果暫時你不想要這個資料庫，可以把圖 4-5a 中的【Generate browse Info】選項清除掉。而當你需要它時，選擇【Tools/Source Browse...】，整合環境就會問你是否要建立 .BSC 檔。

提供給 Browser 的資料 (.BSC) 很類似除錯資料，兩者都包含程式的符號資訊。不同的是，除錯資料附含在 EXE 檔案中，Browser 所需資料則獨立於 .BSC 檔，不會增加 EXE 檔案大小（但會增加程式建造過程所需的時間）。

現在我打算觀察 InitInstance 函式。我在 Browse 對話盒中鍵入 InitInstance 並選擇【Definitions and References】，於是出現如下畫面。雙擊其中的 CWinApp::InitInstance，右側顯示此函式系始定義於 e:\msdev\mfc\include\afxwin.h #4039 行；再雙擊之，編輯器於是載入此檔。以此方式觀察 MFC 原始碼十分方便。



## Online Help

我不是一個喜歡電子書的人，但是拿 VC++ 這個 Help 系統做快速查閱工作實在是不錯。圖 4-6 是其使用畫面與解說。

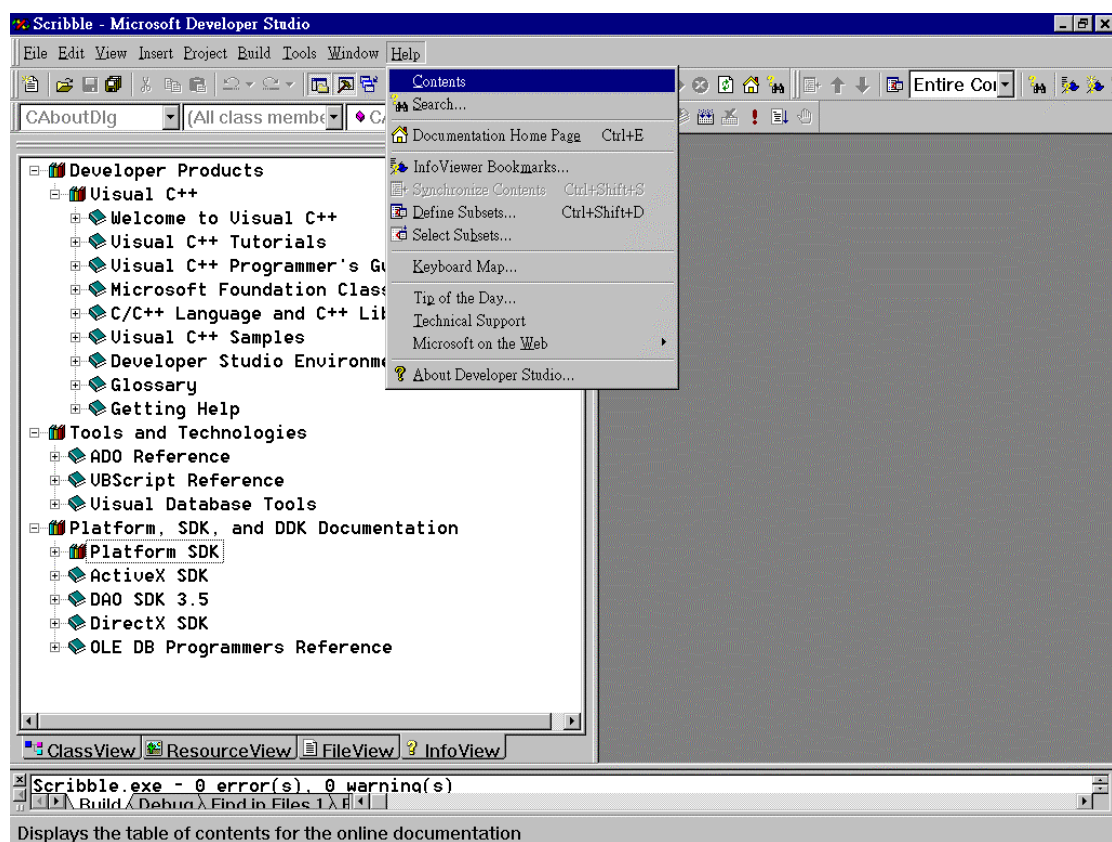


圖 4-6a VC++的 Online Help 提供各種技術資料。按下【Help/Content】，就出現圖左的資料清單，這也就是從 Visual C++ 4.0 開始新增的所謂 **InfoView** 視窗。Online Help 內容非常豐富。

讓我們試試檢索功能。選擇【Help/Search】，出現對話盒，鍵入 CreateThread，出現數篇與此關鍵字有關的網頁。選擇某一篇網頁，網頁內容將出現在另一個視窗中。

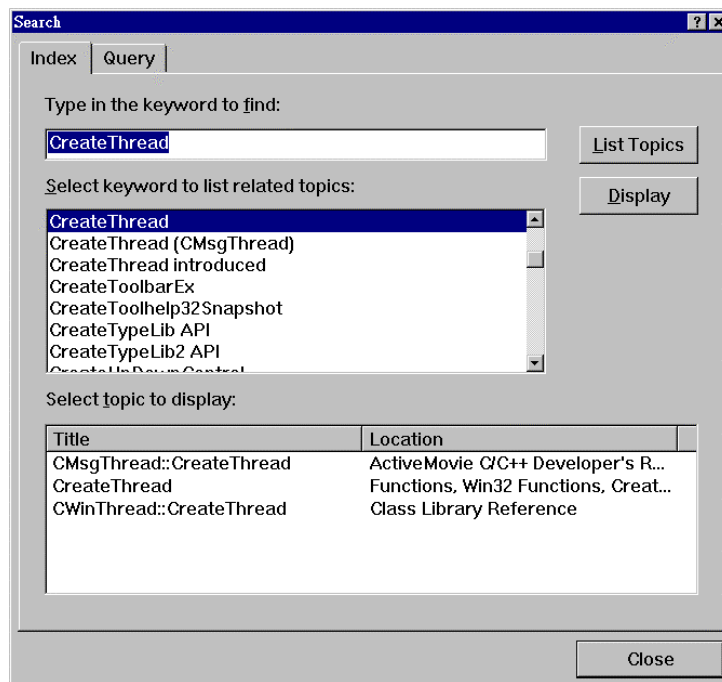


圖 4-6b 檢索功能

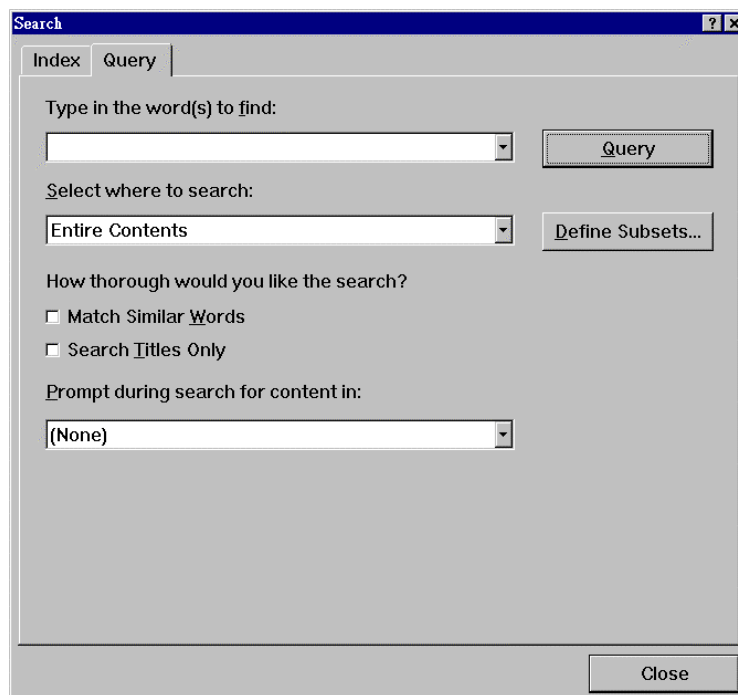


圖 4-6c 檢索功能【Search】對話盒的另一個附頁。允許你做更多搜尋設定。

## 「除錯」工具

每一位 C 程式員在 DOS 環境下都有使用「夾殺法」的除錯經驗：把可能錯誤的範圍不斷縮小，再縮小，最後以 *printf* 印出你心中的嫌疑犯，真象大白。

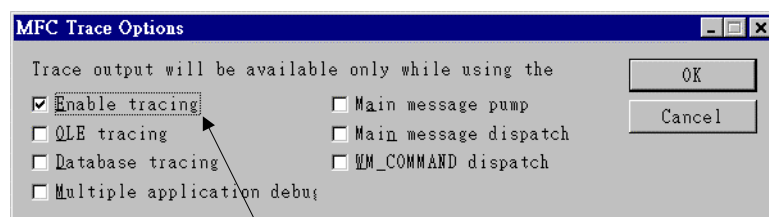
Windows 程式員就沒有方便的 *printf* 可用，唯 *MessageBox* 差可比擬。我曾經在 **Windows 記憶體管理系統篇**（旗標/1993）第 0 章介紹過一種以 *MessageBox* 和 *NotePad.exe* 合作模擬 *printf* 的方法，使用上堪稱便利。

*MessageBox* 會影響你的程式進行，自製 *printf* 又多費手腳。現在有了第三方案。你可以在程式的任何地方放置 *TRACE* 巨集，例如：

```
TRACE("Hello World");
```

參數字串將被輸出到除錯視窗去，不會影響你的程式進行。注意，*TRACE* 巨集只對程式的除錯版才有效，而且程式必須在 Visual C++ 的除錯器中執行。

爲了讓 *TRACE* 生效，你還必須先在另一個程式中做另一個動作。請選按【Tools / MFC Tracer】，得到這樣的畫面：



我們必須將【Enable Tracing】項目設立起來，然後除錯視窗才能顯示 *TRACE* 字串。

舊版的 Visual C++ 中 (v2.0 和 v1.5)，*TRACE* 巨集將字串輸出到一個名爲 DBWin 的程式中。雖然應用程式必須以 “Win32 debug” 編譯完成，但卻不需要進入除錯器就可以獲得 *TRACE* 輸出。從 Visual C++ 4.0 開始到 Visual C++ 5.0，不再附有 DBWin 程式，你無論如何需要大傢伙 (除錯器)。如果你很懷念過去的好時光，請參考 *Microsoft Systems Journal* 上的三篇文章：1995/10 的 C++ Q/A，1996/01 的 C++ Q/A，以及 1997/04 的 C/C++ Q/A。這三篇文章都由 Paul Dilascia 執筆，教讀者如何自己動手做一個可接收 *TRACE* 巨集輸出的 DBWIN 程式。

我將在本書附錄D中對 Paul Dilascia 的創意提供一些說明。

*TRACE* 很好用，美中不足的是它和 *MessageBox* 一樣，只能輸出字串。這裡有一個變通辦法，把字串和數值都送到 *afxDump* 變數去：

```
afxDump << "Hello World " << i << endl; // i 是整數變數
```

這是在 Visual C++ 中傾印 (dump) 一個物件內容的標準方法。它的輸出也是流向除錯視窗，所以你必须確定你的程式是除錯版。

其實，要在應用程式中決定自己是不是除錯版也很簡單。若程式是以除錯模式建造，`_DEBUG` 變數就成為 `TRUE`，因此這樣就可以判斷一切了：

```
#ifdef _DEBUG
afxDump << "Hello World" << i << "\n"; // i 是整數變數
#endif
```

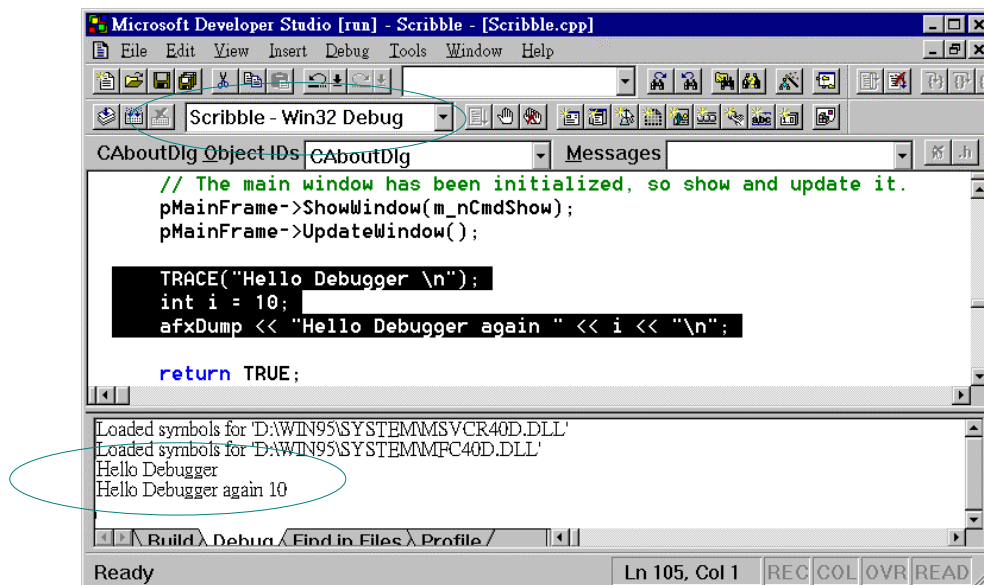


圖 4-1 左方的三行程式碼導引圖 4-1 右方除錯視窗中的輸出。

## VC++ 除錯器

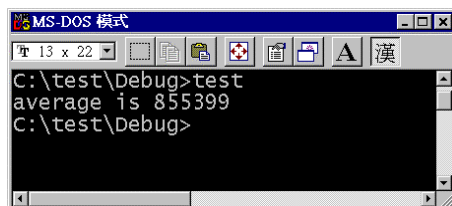
Visual C++ 整合環境內含一個精巧的除錯器。這個除錯器讓我們很方便設定中斷點（程式執行至此會暫停）、觀察變數內容、暫存器內容、並允許在除錯過程中改變變數的值。我將以一個實際的獵蟲行動示範如何使用除錯器。

欲使用除錯器，首先你的程式必須含有除錯符號，也就是說，這必須是個除錯版。很簡單，只要在 Visual C++ 整合環境上方選擇【Win32 Debug】模式，然後再進行建造

(building) 工作，即可獲得除錯版本。現在假設我有一個程式，要計算五個學生的平均成績：

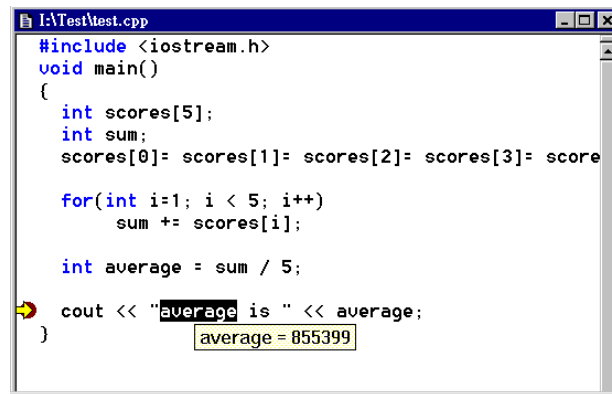
```
#0001 #include <iostream.h>
#0002 void main()
#0003 {
#0004     int scores[5];
#0005     int sum;
#0006     scores[0]= scores[1]= scores[2]= scores[3]= scores[4]= 60;
#0007
#0008     for(int i=1; i < 5; i++)
#0009         sum += scores[i];
#0010
#0011     int average = sum / 5;
#0012
#0013     cout << "average is " << average;
#0014 }
```

我預期的結果是 *average* 等於 60，而得到的結果卻是：



爲了把臭蟲找出來，必須控制程式的進行，也就是設定中斷點 (breakpoint)；程式暫停之際，我就可以觀察各個有嫌疑的變數。設定中斷點的方法是：把游標移到目的行，按下工具列上的手形按鈕 (或 F9)，於是該行前面出現紅點，表示中斷點設立。F9 是一個切換開關，在同一行再按一次 F9 就能夠清除中斷點。

爲讓中斷點生效，我必須以【Build/Debug/Go】(或 F5) 執行程式，此時整合環境上的【Build】選單變成了【Debug】。程式執行至中斷點即停下來，*average* 此刻的值應該是 60。爲證實此點，把游標放在 *average* 上，出現一個小黃標籤：



```

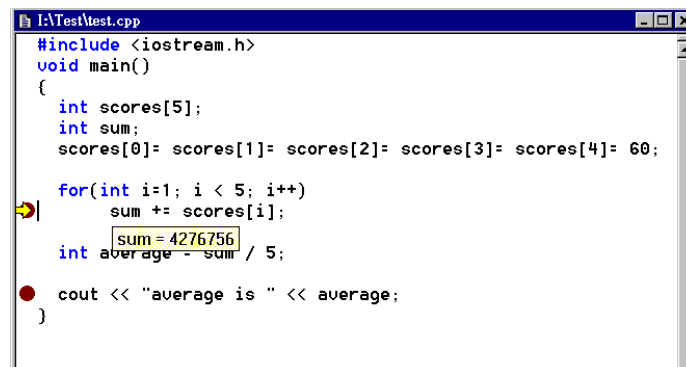
I:\Test\test.cpp
#include <iostream.h>
void main()
{
    int scores[5];
    int sum;
    scores[0]= scores[1]= scores[2]= scores[3]= score

    for(int i=1; i < 5; i++)
        sum += scores[i];

    int average = sum / 5;
    cout << "average is " << average;
}
    average = 855399

```

結果令人大吃一驚。顯然我們有必要另設中斷點，觀察其他變數。先以【Debug/Stop Debugging】結束除錯狀態，然後把中斷點設在 `sum += scores[i]` 這一行，重新 "Go" 下去。程式暫停時觀察 `sum` 的值：



```

I:\Test\test.cpp
#include <iostream.h>
void main()
{
    int scores[5];
    int sum;
    scores[0]= scores[1]= scores[2]= scores[3]= scores[4]= 60;

    for(int i=1; i < 5; i++)
        sum += scores[i];
    int average = sum / 5;
    cout << "average is " << average;
}
    sum = 4276756

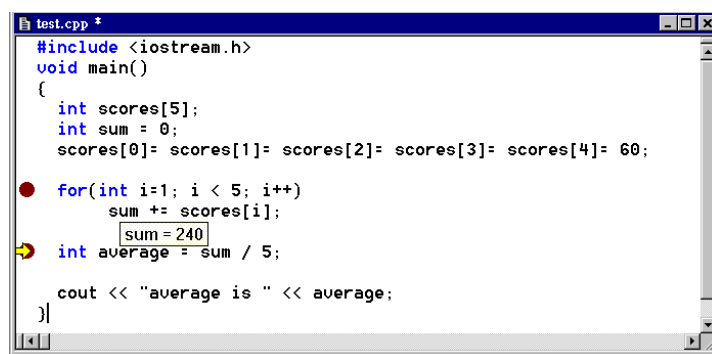
```

此時此刻程式尚未執行任何一個加法，`sum` 應該是 0，但結果未符預期。顯然，`sum` 的初值沒有設為 0，我們抓到臭蟲了。現在把程式碼第 5 行改為

```
int sum = 0;
```

重新建造，再 "Go" 一次。五次迴路之後我們預期 `sum` 的值是 300，結果卻是 240：





```
test.cpp *
#include <iostream.h>
void main()
{
    int scores[5];
    int sum = 0;
    scores[0]= scores[1]= scores[2]= scores[3]= scores[4]= 60;

    for(int i=1; i < 5; i++)
        sum += scores[i];
    int average = sum / 5;

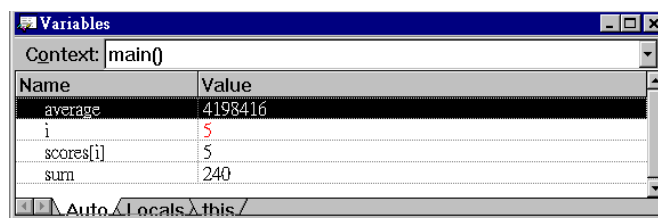
    cout << "average is " << average;
}
```

原來我竟把陣列索引值  $i$  從 1 開始計算而不是從 0 開始。

臭蟲全部抓出來了，程式修正如下：

```
#0001 #include <iostream.h>
#0002 void main()
#0003 {
#0004     int scores[5];
#0005     int sum = 0;
#0006     scores[0]= scores[1]= scores[2]= scores[3]= scores[4]= 60;
#0007
#0008     for(int i=0; i < 5; i++)
#0009         sum += scores[i];
#0010
#0011     int average = sum / 5;
#0012
#0013     cout << "average is " << average;
#0014 }
```

除錯過程中，你也可以選按【View/Variables】打開 Variables 視窗，所有的變數值更能夠一目了然：



Name	Value
average	4198416
i	5
scores[i]	5
sum	240

除了除錯之外，我還常常以除錯器追蹤 MFC 程式，以期深入了解 MFC 各類別。你知道，數萬行 MFC 原始碼光靠 Step Into/Step Over/Call Stack 這幾招，便能迅速切中「要害」。

小技巧：當你要找出與視窗 painting 有關的臭蟲時，儘量不要把欲除錯之程式視窗與 Visual C++ IDE 視窗覆蓋在一起，才不會互相影響。當然，最好你有一個 17 吋螢幕和 1024\*768 的解析度。21 吋螢幕？呃，小心你的荷包。

## Exception Handling

第 2 章最後面我曾簡介過 C++ exception handling。這裡我要再舉一個很容易練習的 MFC exception handling 實例。

開檔是一件可能產生許多 exception 的動作。檔案開啓不成功，可能是因為檔案找不到，或是磁碟空間不足，或是路徑不對，或是違反檔案共享原則（sharing violation），或是超出了可開檔數目限制。你可以在任何一個程式中練習下面這一段碼。不需要除錯版，基本上 exception handling 與除錯模式並無瓜葛。下列程式碼中的 *Output* 函式只是個代名，並不是真有這樣的 API 函式，你可以改用 *MessageBox*、*TextOut*、*TRACE* 等任何有字串輸出能力的函式。

```
#001 CString str = "Hello World";
#002
#003 TRY {
#004     CFile file("a:hello.txt", CFile::modeCreate | CFile::modeWrite);
#005     file.Write(str, str.GetLength());
#006     file.Close();
#007 }
#008 CATCH(CFileException, e) {
#009     switch(e->m_cause) {
#010         case CFileException::accessDenied :
#011             Output("File Access Denied");
#012             break;
#013         case CFileException::badPath :
#014             Output("Invalid Path");
#015             break;
```

```
#016     case CFileException::diskFull :
#017         Output("Disk Full");
#018         break;
#019     case CFileException::fileNotFound :
#020         Output("File Not Found");
#021         break;
#022     case CFileException::hardIO :
#023         Output("Hardware Error");
#024         break;
#025     case CFileException::lockViolation :
#026         Output("Attemp to lock region already locked");
#027         break;
#028     case CFileException::sharingViolation :
#029         Output("Sharing Violation - load share.exe");
#030         break;
#031     case CFileException::tooManyOpenFiles :
#032         Output("Too Many Open Files");
#033         break;
#034     }
#035 }
```

讓我簡單地做一個說明。*TRY* 區塊中的動作（本例為開檔、寫檔、關檔）如果在執行時期有任何 *exception* 發生，就會跳到 *CATCH* 區塊中執行。*CATCH* 的第一個參數是 *exception type*：如果是檔案方面的 *exception*，就是 *CFileException*，如果是記憶體方面的 *exception*，那麼就是 *CMemoryException*。*CATCH* 的第二個參數是一個物件，經由其資料成員 *m\_cause*，我們可以獲知 *exception* 的發生原因。這些原因（如 *accessDenied*, *badPath*, *diskFull*, *FileNotFound*...）都定義於 *AFX.H* 中。

## 程式碼產生器：AppWizard

有一個 Generic 範例程式，號稱為「Windows 程式之母」，恐怕大家都是從那裡跨出 Windows 程式設計的第一步。過去，當我要開始一個新的 project，我就把 Generic 的所有檔案拷貝到新的子目錄下，然後改變檔名，然後把 makefile 中所有的 "GENERIC" 字串改為新 project 名稱字串。我還必須改變 C 檔案中的視窗類別名稱、視窗標題、選單名稱、對話盒名稱；我必須改變 RC 檔中的選單、對話盒等資源；我也得改變 DEF 檔中的模組名稱和 DESCRIPTION 敘述句。這些瑣碎的事做完，我才開始在 DEF、C、RC 檔中添骨添肉。

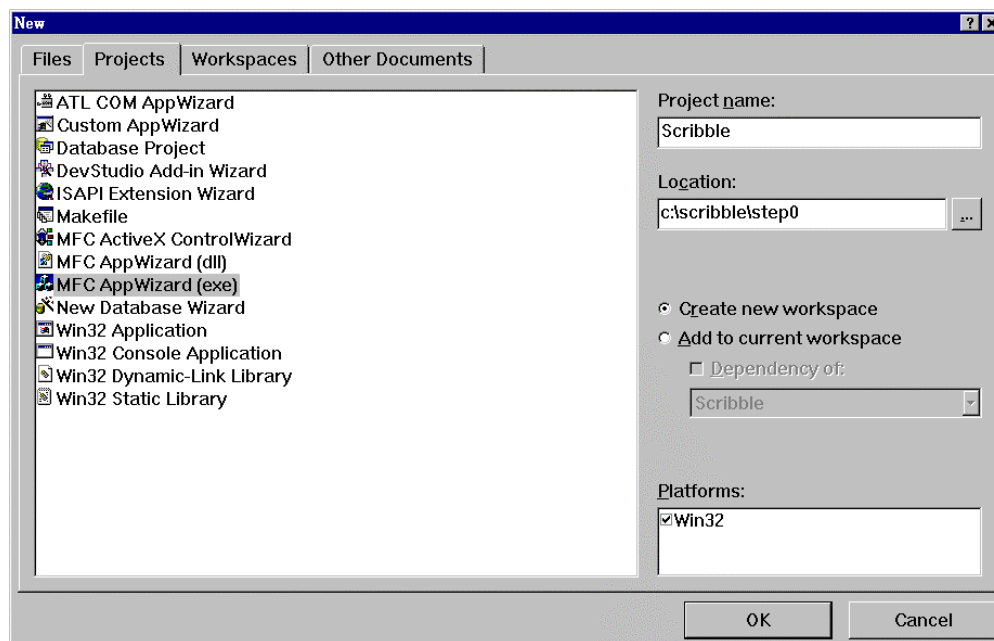
數以打計的小步驟要做 !!

有了 AppWizard，這些沉悶而令人生厭的瑣碎工作都將自動化起來。不止如此，AppWizard 可以為我們做出一致化的骨幹程式出來。以此種方式應付（我的意思是產生）標準介面十分合適。

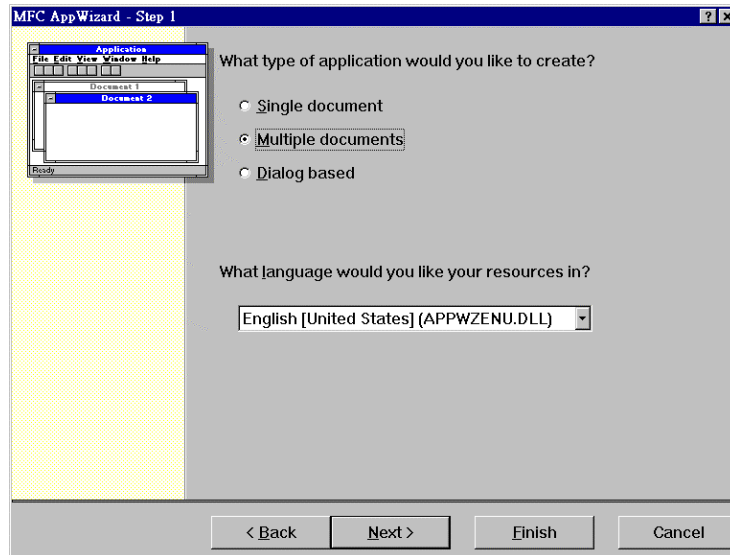
你可以從 Visual C++ 整合環境中啟動 AppWizard。第一次使用時機是當你要展開一個新的 project 之時。首先，為 project 命名並為它找一個棲身場所（一個磁碟目錄），然後選擇你想要的程式風格（例如 SDI 或 MDI）。問答題作完，劈哩啪啦呼嚕嘩啦，AppWizard 很快為你產生一個骨幹程式。這是一個完整的，不需增減任何一行碼就可編譯執行的程式，雖然它什麼大事兒都沒做，卻保證令你印象深刻。外觀（使用者介面）十分華麗，Win32 程式員窮數星期之心力也不見得做得出這樣漂亮豐富的介面來。

## 點選完 MFC 程式骨幹

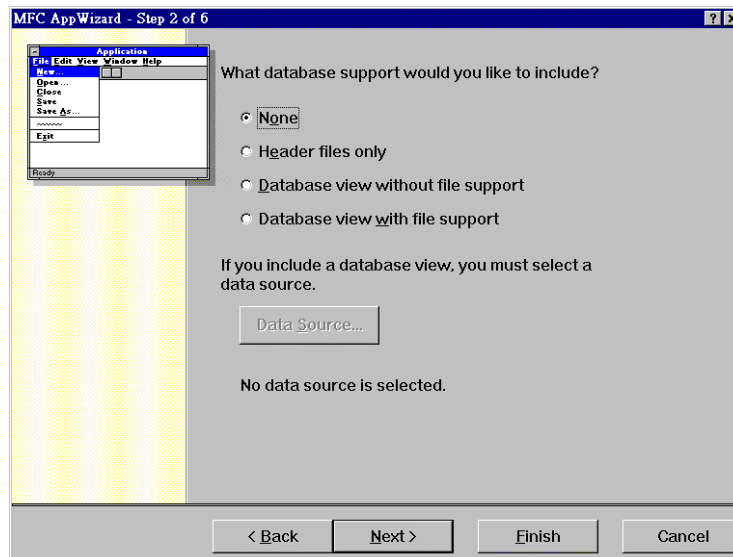
選按【File/New】，並在【New】對話盒中選擇【Project】附頁。然後再在其中選擇 MFC Application (exe)，於是準備進 Visual AppWizard 建立 "Scribble" project。右邊的磁碟目錄和 project 名稱亦需填妥。



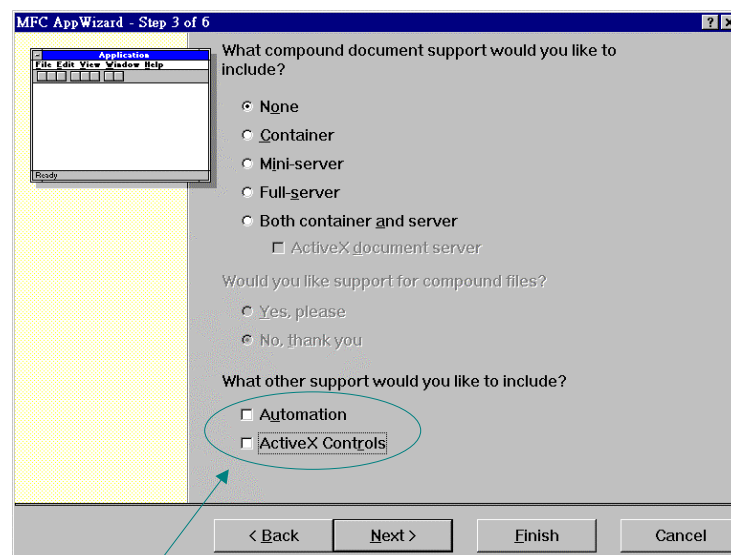
MFC AppWizard 步驟一，選擇 SDI 或 MDI 或 Dialog-based 程式風格。預設情況是 MDI。



MFC AppWizard 步驟二，選擇是否需要資料庫支援。預設情況是 None。

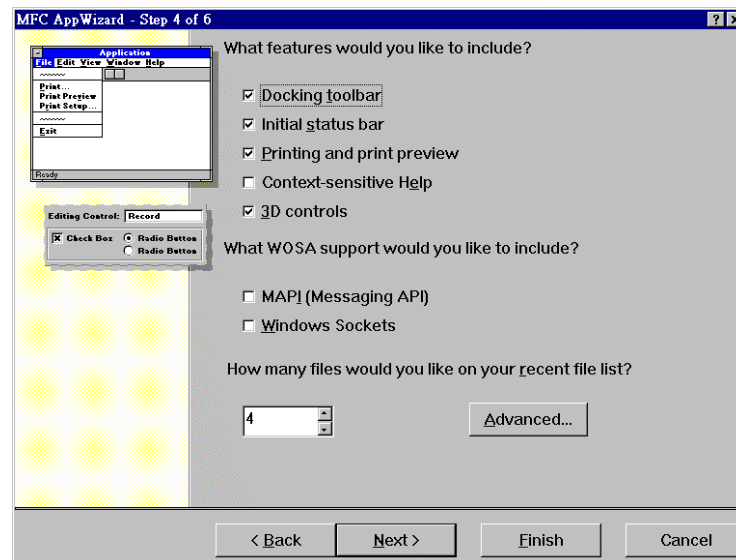


MFC AppWizard 步驟三，選擇是否需要 compound document 和 ActiveX 支援。預設情況支援 ActiveX Controls，本例為求簡化，將它關閉。

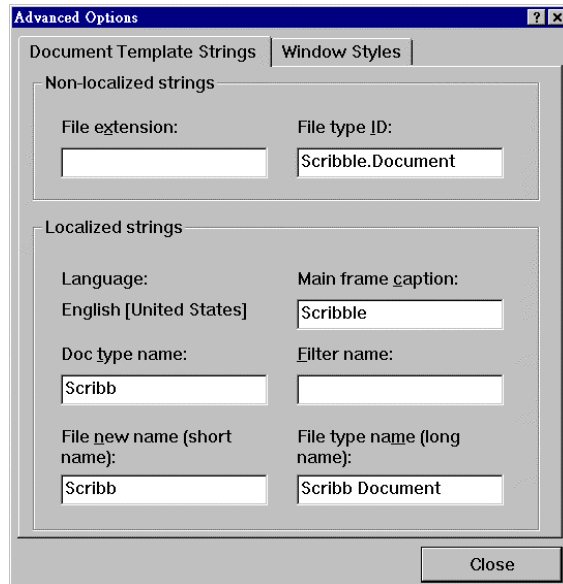


注意，在 VC++ 4.x 版中此處為 OLE Automation 和 OLE controls。

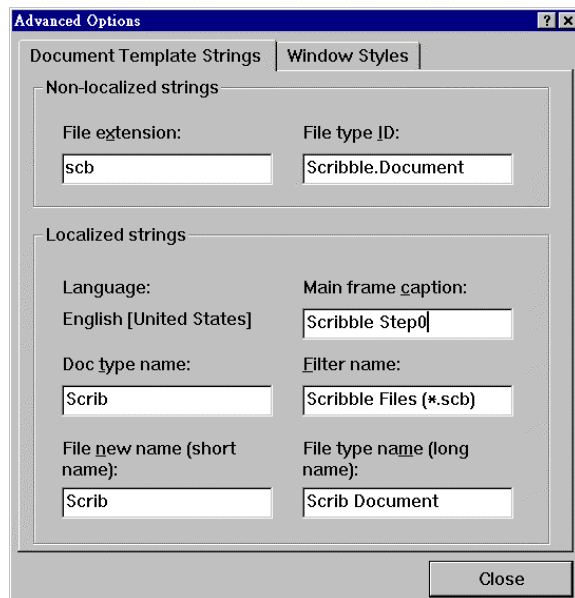
MFC AppWizard 步驟四，選擇使用何種介面。預設情況【Context Sensitive Help】未設定。



MFC AppWizard 步驟 4 的【Advanced】帶出【Advanced Options】對話盒：

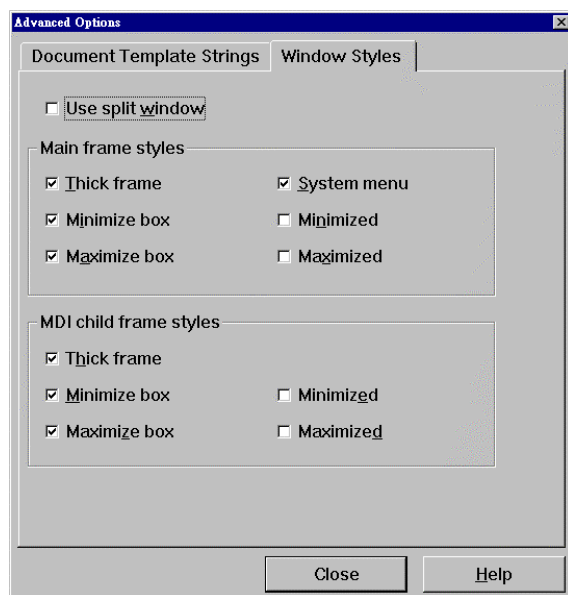


把上面修改為下面這個樣子。這些修改對程式碼帶來的變化，將在第 7 章中說明。

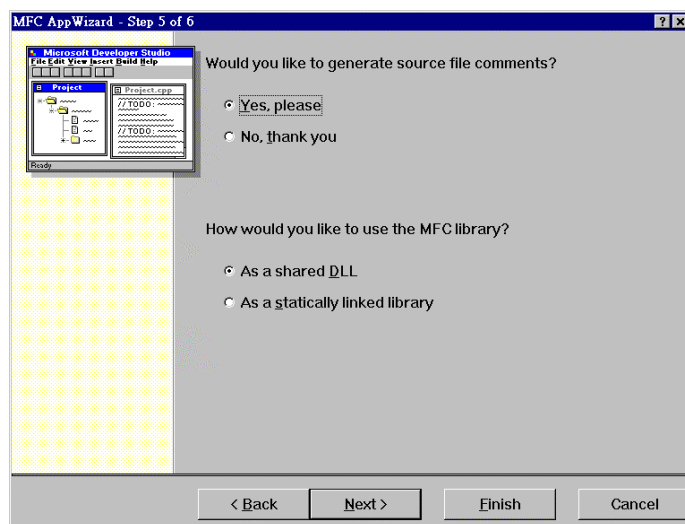




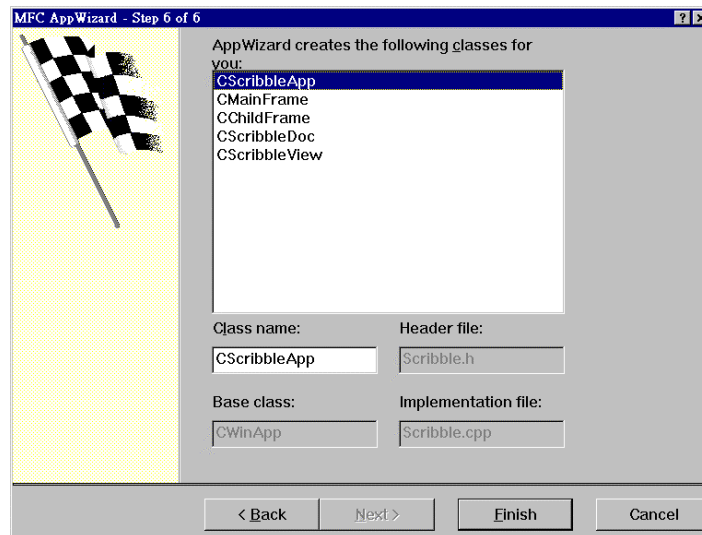
MFC AppWizard 【Advanced Options】的另 一 附頁。其中最上面的一 個複選鈕【Use split window】預設是 關閉狀態。如果要製作分裂視窗（如本書第 11 章），把它打開就是了。



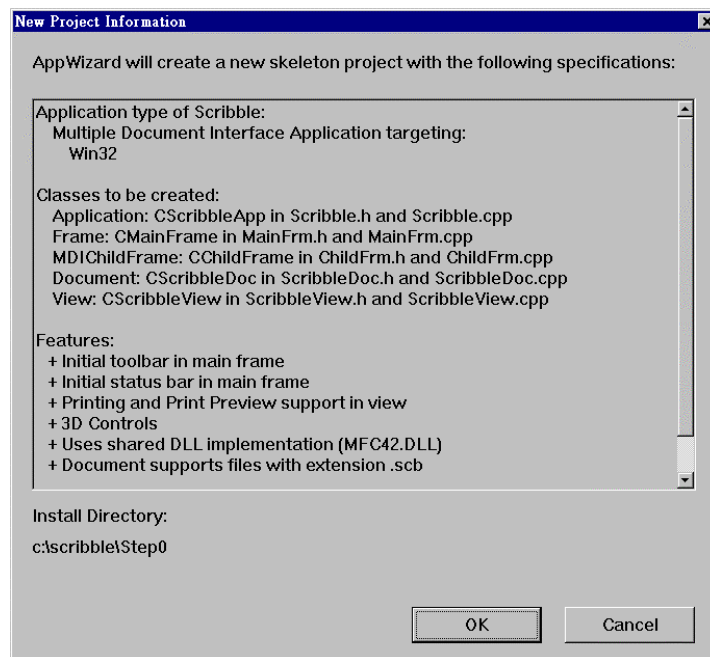
MFC AppWizard 步驟 5，提供另 一 些選項，詢問要下製為你的原始碼產生 一 些說明文字。並詢問你希望使用的 MFC 版本（動態聯結版或靜態聯結版）。



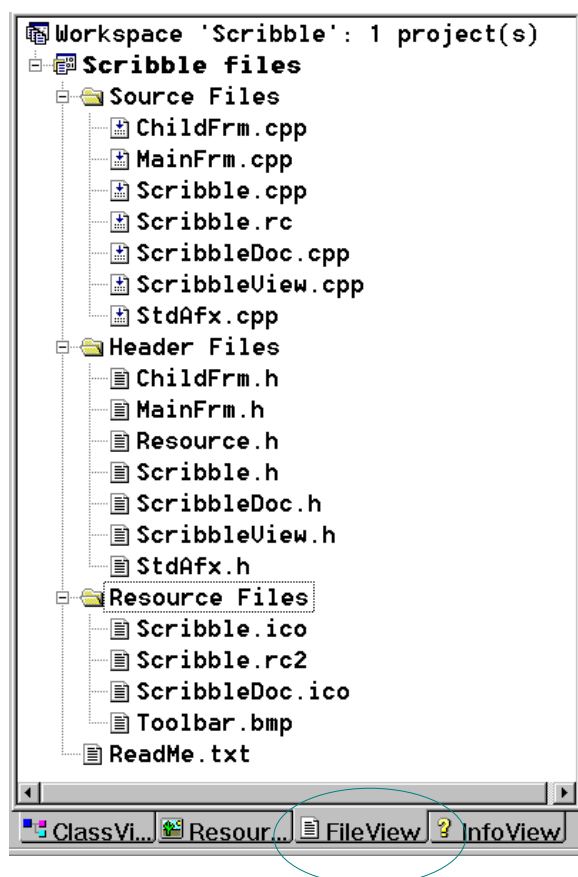
MFC AppWizard 步驟六（最後一步），允許你更改檔名或類別名稱。完成後按下【Finish】



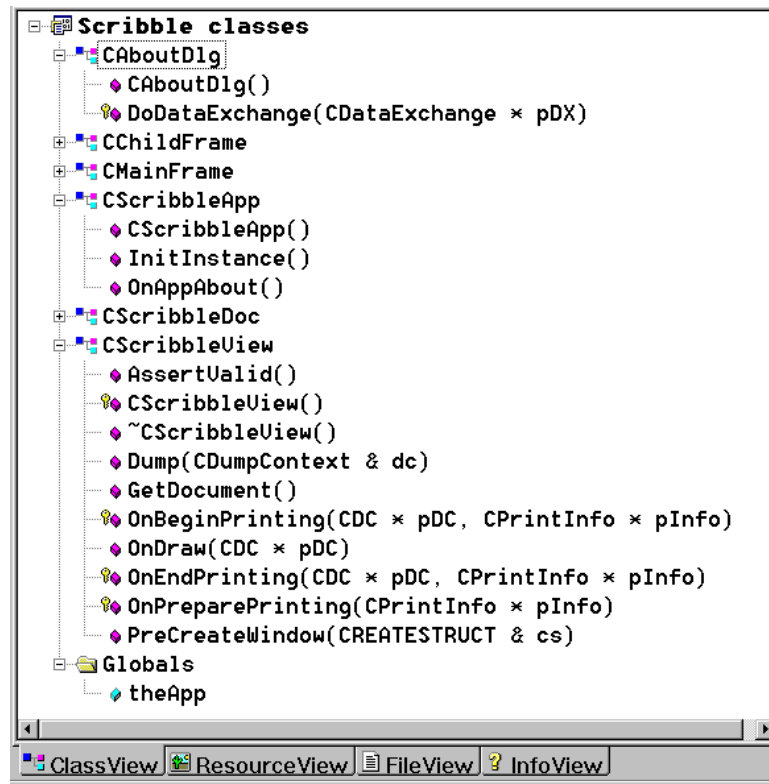
MFC AppWizard 獲得的清單（包括檔案和類別）：



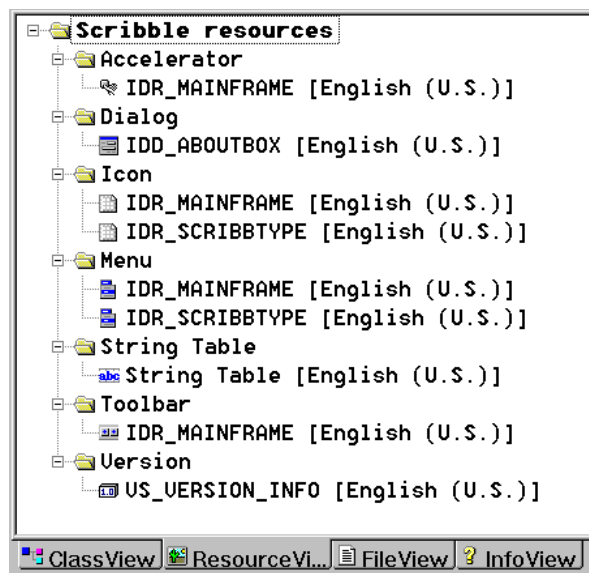
- 程式碼都還沒寫，就獲得了這許多檔案。你可以選擇【Win32 Debug】或【Win32 Release】來建造（building）程式，獲得的二進位檔案將放在不同的子目錄中。



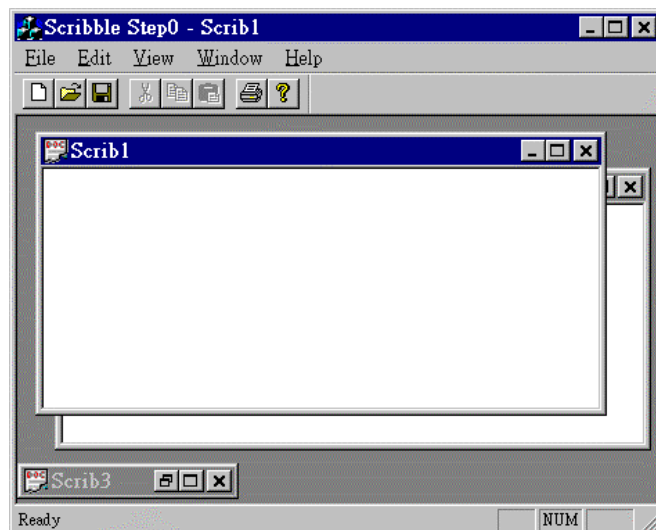
- 行程式碼都還沒寫，就獲得了這許多類別。



- 行程式碼都還沒寫，就獲得了這 麼多資源。



- 行程式碼都沒寫，只是點點按按，我們就獲得了 - 個令人驚豔的程式。基本功能 - 檔案、對話盒、印表機設定、Help、工具列、狀態列...），卻什麼都不能做（這是當然）。



AppWizard 總是為一般的應用程式產生五個類別。我所謂的「一般程式」是指 non-OLE 以及 non-ODBC 程式。針對上述的 Scribble 程式，它產生的類別列於圖 4-7。

類別名稱	基礎類別	類別宣告於	類別定義於
<i>CScribbleApp</i>	<i>CWinApp</i>	Scribble.h	Scribble.cpp
<i>CMainFrame</i>	<i>CMDIFrameWnd</i>	Mainfrm.h	Mainfrm.cpp
<i>CChildFrame</i>	<i>CMDIChildWnd</i>	Childfrm.h	Childfrm.cpp
<i>CScribbleDoc</i>	<i>CDocument</i>	ScribbleDoc.h	ScribbleDoc.cpp
<i>CScribbleView</i>	<i>CView</i>	ScribbleView.h	ScribbleView.cpp

圖 4-7 Scribble Step0( 骨幹程式 )中，各個類別的相關資料。事實上 Scribble 程式中用到了 個類別，不過只有上述 個類別需要改寫 ( override )。

你最好把哪一個類別衍生自哪一個 MFC 類別弄清楚，並搞懂 AppWizard 的命名規則。大致上命名規則是這樣的：

```
'C' + ProjectName + Classtype = Class Name
```

所有的類別名稱都由 AppWizard 自動命名，如果你喜歡，也可以在 AppWizard 的步驟六改變之。這些類別名稱可以很長很長 ( Windows 95 與 Windows NT 均支援長檔名 )。每個類別都對應一個 .H ( 類別宣告 ) 和一個 .CPP ( 類別定義 )。

AppWizard 十分周到地為我們產生了一個 README.TXT，對各個檔案都有解釋 ( 圖 4-8 )。從啟動 AppWizard 到建立 Scribble.exe，如果你是熟手，機器又不慢的話，不需要一分鐘。拿著碼錶算時間其實不具意義 ( 就像計算程式行數多寡一樣地不具意義 )，我要說的是它的確便利。

```
=====
MICROSOFT FOUNDATION CLASS LIBRARY : Scribble
=====

AppWizard has created this Scribble application for you. This application
not only demonstrates the basics of using the Microsoft Foundation classes
but is also a starting point for writing your application.

This file contains a summary of what you will find in each of the files that
make up your Scribble application.

Scribble.h
    This is the main header file for the application. It includes other
    project specific headers (including Resource.h) and declares the
    CScribbleApp application class.

Scribble.cpp
    This is the main application source file that contains the application
    class CScribbleApp.

Scribble.rc
    This is a listing of all of the Microsoft Windows resources that the
    program uses. It includes the icons, bitmaps, and cursors that are stored
    in the RES subdirectory. This file can be directly edited in Microsoft
    Developer Studio.

res\Scribble.ico
    This is an icon file, which is used as the application's icon. This
    icon is included by the main resource file Scribble.rc.

res\Scribble.rc2
    This file contains resources that are not edited by Microsoft
    Developer Studio. You should place all resources not
    editable by the resource editor in this file.

Scribble.clw
    This file contains information used by ClassWizard to edit existing
    classes or add new classes. ClassWizard also uses this file to store
    information needed to create and edit message maps and dialog data
    maps and to create prototype member functions.

////////////////////////////////////

For the main frame window:
```

```
MainFrm.h, MainFrm.cpp
    These files contain the frame class CMainFrame, which is derived from
    CMDIFrameWnd and controls all MDI frame features.

res\Toolbar.bmp
    This bitmap file is used to create tiled images for the toolbar.
    The initial toolbar and status bar are constructed in the
    CMainFrame class. Edit this toolbar bitmap along with the
    array in MainFrm.cpp to add more toolbar buttons.

////////////////////////////////////

AppWizard creates one document type and one view:

ScribbleDoc.h, ScribbleDoc.cpp - the document
    These files contain your CScribbleDoc class. Edit these files to
    add your special document data and to implement file saving and loading
    (via CScribbleDoc::Serialize).

ScribbleView.h, ScribbleView.cpp - the view of the document
    These files contain your CScribbleView class.
    CScribbleView objects are used to view CScribbleDoc objects.

res\ScribbleDoc.ico
    This is an icon file, which is used as the icon for MDI child windows
    for the CScribbleDoc class. This icon is included by the main
    resource file Scribble.rc.

////////////////////////////////////

Other standard files:

StdAfx.h, StdAfx.cpp
    These files are used to build a precompiled header (PCH) file
    named Scribble.pch and a precompiled types file named StdAfx.obj.

Resource.h
    This is the standard header file, which defines new resource IDs.
    Microsoft Developer Studio reads and updates this file.

////////////////////////////////////

Other notes:

AppWizard uses "TODO:" to indicate parts of the source code you
should add to or customize.
```



If your application uses MFC in a shared DLL, and your application is in a language other than the operating system's current language, you will need to copy the corresponding localized resources MFC40XXX.DLL from the Microsoft Visual C++ CD-ROM onto the system or system32 directory, and rename it to be MFCLOC.DLL. ("XXX" stands for the language abbreviation. For example, MFC40DEU.DLL contains resources translated to German.) If you don't do this, some of the UI elements of your application will remain in the language of the operating system.

圖 4-8 Scribble 程式的 readme.txt 檔。

別忘了，AppWizard 產生的是化學反應而不是物理反應，是不能夠還原的。我們很容易犯的錯誤是像進入糖果店裡的小孩一樣，每樣東西都想要。你應該約束自己，因為錯一步已是百年身，不能稍後又回到 AppWizard 要求去掉或改變某些選項，例如想把 SDI 改為 MDI 或是想增加 OLE 支援等等，都不能夠。欲變更程式，只有兩條路可走：要不就令 AppWizard 重新產生一組新的程式骨幹，然後回到原程式中打撈點什麼可以用的，以 Copy/Paste 方式移植過來；要不就是直接進入原來程式修修補補。至於修補過程中到底會多麼令人厭煩，那就不一而足了。所以，在開始你的程式撰寫之前，小心做好系統分析的工作。

Scribble 是第四篇程式的起點。我將在第四篇以每章一個主題的方式，為它加上新的功能。下面是 Scribble step0 的原始碼。

#### SCRIBBLE.H

```
#0001 // Scribble.h : main header file for the SCRIBBLE application
#0002 //
#0003
#0004 #ifndef __AFXWIN_H__
#0005 #error include 'stdafx.h' before including this file for PCH
#0006 #endif
#0007
#0008 #include "resource.h"          // main symbols
#0009
#0010 //////////////////////////////////////
#0011 // CScribbleApp:
#0012 // See Scribble.cpp for the implementation of this class
#0013 //
```

```

#0014
#0015 class CScribbleApp : public CWinApp
#0016 {
#0017 public:
#0018     CScribbleApp();
#0019
#0020 // Overrides
#0021 // ClassWizard generated virtual function overrides
#0022 //{{AFX_VIRTUAL(CScribbleApp)
#0023 public:
#0024     virtual BOOL InitInstance();
#0025     //}}AFX_VIRTUAL
#0026
#0027 // Implementation
#0028
#0029 //{{AFX_MSG(CScribbleApp)
#0030     afx_msg void OnAppAbout();
#0031     // NOTE - the ClassWizard will add and remove member functions here.
#0032     //      DO NOT EDIT what you see in these blocks of generated code !
#0033     //}}AFX_MSG
#0034     DECLARE_MESSAGE_MAP()
#0035 };

```

#### MAINFRM.H

```

#0001 // MainFrm.h : interface of the CMainFrame class
#0002 //
#0003 //////////////////////////////////////
#0004
#0005 class CMainFrame : public CMDIFrameWnd
#0006 {
#0007     DECLARE_DYNAMIC(CMainFrame)
#0008 public:
#0009     CMainFrame();
#0010
#0011 // Attributes
#0012 public:
#0013
#0014 // Operations
#0015 public:
#0016
#0017 // Overrides
#0018 // ClassWizard generated virtual function overrides
#0019 //{{AFX_VIRTUAL(CMainFrame)
#0020     virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
#0021     //}}AFX_VIRTUAL

```

```
#0022
#0023 // Implementation
#0024 public:
#0025     virtual ~CMainFrame();
#0026 #ifdef _DEBUG
#0027     virtual void AssertValid() const;
#0028     virtual void Dump(CDumpContext& dc) const;
#0029 #endif
#0030
#0031 protected: // control bar embedded members
#0032     CStatusBar  m_wndStatusBar;
#0033     CToolBar   m_wndToolBar;
#0034
#0035 // Generated message map functions
#0036 protected:
#0037     //{AFX_MSG(CMainFrame)
#0038     afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
#0039     // NOTE - the ClassWizard will add and remove member functions here.
#0040     //     DO NOT EDIT what you see in these blocks of generated code!
#0041     //}AFX_MSG
#0042     DECLARE_MESSAGE_MAP()
#0043 };
```

## CHILDFRM.H

```
#0001 // ChildFrm.h : interface of the CChildFrame class
#0002 //
#0003 //////////////////////////////////////
#0004
#0005 class CChildFrame : public CMDIChildWnd
#0006 {
#0007     DECLARE_DYNCREATE(CChildFrame)
#0008 public:
#0009     CChildFrame();
#0010
#0011 // Attributes
#0012 public:
#0013
#0014 // Operations
#0015 public:
#0016
#0017 // Overrides
#0018 // ClassWizard generated virtual function overrides
#0019 //{AFX_VIRTUAL(CChildFrame)
#0020 virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
#0021 //}AFX_VIRTUAL
```

```

#0022
#0023 // Implementation
#0024 public:
#0025     virtual ~CChildFrame();
#0026 #ifdef _DEBUG
#0027     virtual void AssertValid() const;
#0028     virtual void Dump(CDumpContext& dc) const;
#0029 #endif
#0030
#0031 // Generated message map functions
#0032 protected:
#0033     //{AFX_MSG(CChildFrame)
#0034         // NOTE - the ClassWizard will add and remove member functions here.
#0035         //      DO NOT EDIT what you see in these blocks of generated code!
#0036     //}AFX_MSG
#0037     DECLARE_MESSAGE_MAP()
#0038 };

```

#### SCRIBBLEDOC.H

```

#0001 // ScribbleDoc.h : interface of the CScribbleDoc class
#0002 //
#0003 //////////////////////////////////////
#0004
#0005 class CScribbleDoc : public CDocument
#0006 {
#0007     protected: // create from serialization only
#0008         CScribbleDoc();
#0009         DECLARE_DYNCREATE(CScribbleDoc)
#0010
#0011     // Attributes
#0012     public:
#0013
#0014     // Operations
#0015     public:
#0016
#0017     // Overrides
#0018     // ClassWizard generated virtual function overrides
#0019     //{AFX_VIRTUAL(CScribbleDoc)
#0020     public:
#0021         virtual BOOL OnNewDocument();
#0022         virtual void Serialize(CArchive& ar);
#0023     //}AFX_VIRTUAL
#0024
#0025     // Implementation
#0026     public:

```

```
#0027 virtual ~CScribbleDoc();
#0028 #ifdef _DEBUG
#0029 virtual void AssertValid() const;
#0030 virtual void Dump(CDumpContext& dc) const;
#0031 #endif
#0032
#0033 protected:
#0034
#0035 // Generated message map functions
#0036 protected:
#0037 //{{AFX_MSG(CScribbleDoc)
#0038     // NOTE - the ClassWizard will add and remove member functions here.
#0039     //      DO NOT EDIT what you see in these blocks of generated code !
#0040 //}}AFX_MSG
#0041 DECLARE_MESSAGE_MAP()
#0042 };
```

## SCRIBBLEVIEW.H

```
#0001 // ScribbleView.h : interface of the CScribbleView class
#0002 //
#0003 //////////////////////////////////////
#0004
#0005 class CScribbleView : public CView
#0006 {
#0007 protected: // create from serialization only
#0008     CScribbleView();
#0009     DECLARE_DYNCREATE(CScribbleView)
#0010
#0011 // Attributes
#0012 public:
#0013     CScribbleDoc* GetDocument();
#0014
#0015 // Operations
#0016 public:
#0017
#0018 // Overrides
#0019 // ClassWizard generated virtual function overrides
#0020 //{{AFX_VIRTUAL(CScribbleView)
#0021 public:
#0022     virtual void OnDraw(CDC* pDC); // overridden to draw this view
#0023     virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
#0024 protected:
#0025     virtual BOOL OnPreparePrinting(CPrintInfo* pInfo);
#0026     virtual void OnBeginPrinting(CDC* pDC, CPrintInfo* pInfo);
#0027     virtual void OnEndPrinting(CDC* pDC, CPrintInfo* pInfo);
```

```

#0028  //}}AFX_VIRTUAL
#0029
#0030  // Implementation
#0031  public:
#0032  virtual ~CScribbleView();
#0033  #ifdef _DEBUG
#0034  virtual void AssertValid() const;
#0035  virtual void Dump(CDumpContext& dc) const;
#0036  #endif
#0037
#0038  protected:
#0039
#0040  // Generated message map functions
#0041  protected:
#0042  //{AFX_MSG(CScribbleView)
#0043  // NOTE - the ClassWizard will add and remove member functions here.
#0044  // DO NOT EDIT what you see in these blocks of generated code !
#0045  //}}AFX_MSG
#0046  DECLARE_MESSAGE_MAP()
#0047  };
#0048
#0049  #ifndef _DEBUG // debug version in ScribbleView.cpp
#0050  inline CScribbleDoc* CScribbleView::GetDocument()
#0051  { return (CScribbleDoc*)m_pDocument; }
#0052  #endif

```

#### STDAFX.H

```

#0001  // stdafx.h : include file for standard system include files,
#0002  // or project specific include files that are used frequently, but
#0003  // are changed infrequently
#0004  //
#0005
#0006  #define VC_EXTRALEAN // Exclude rarely-used stuff from Windows headers
#0007
#0008  #include <afxwin.h> // MFC core and standard components
#0009  #include <afxext.h> // MFC extensions
#0010  #ifndef _AFX_NO_AFXCMN_SUPPORT
#0011  #include <afxcmn.h> // MFC support for Windows Common Controls
#0012  #endif // _AFX_NO_AFXCMN_SUPPORT

```

#### RESOURCE.H

```

#0001  //{NO_DEPENDENCIES}
#0002  // Microsoft Visual C++ generated include file.

```

```
#0003 // Used by SCRIBBLE.RC
#0004 //
#0005 #define IDR_MAINFRAME            128
#0006 #define IDR_SCRIBTYPE            129
#0007 #define IDD_ABOUTBOX             100
#0008
#0009 // Next default values for new objects
#0010 //
#0011 #ifdef APSTUDIO_INVOKED
#0012 #ifndef APSTUDIO_READONLY_SYMBOLS
#0013 #define _APS_3D_CONTROLS            1
#0014 #define _APS_NEXT_RESOURCE_VALUE    130
#0015 #define _APS_NEXT_CONTROL_VALUE     1000
#0016 #define _APS_NEXT_SYMED_VALUE       101
#0017 #define _APS_NEXT_COMMAND_VALUE     32771
#0018 #endif
#0019 #endif
```

#### STDAFX.CPP

```
#0001 // stdafx.cpp : source file that includes just the standard includes
#0002 //      Scribble.pch will be the pre-compiled header
#0003 //      stdafx.obj will contain the pre-compiled type information
#0004
#0005 #include "stdafx.h"
```

#### SCRIBBLE.CPP

```
#0001 // Scribble.cpp : Defines the class behaviors for the application.
#0002 //
#0003
#0004 #include "stdafx.h"
#0005 #include "Scribble.h"
#0006
#0007 #include "MainFrm.h"
#0008 #include "ChildFrm.h"
#0009 #include "ScribbleDoc.h"
#0010 #include "ScribbleView.h"
#0011
#0012 #ifdef _DEBUG
#0013 #define new DEBUG_NEW
#0014 #undef THIS_FILE
#0015 static char THIS_FILE[] = __FILE__;
#0016 #endif
#0017
```

```
#0018 ///////////////////////////////////////////////////
#0019 // CScribbleApp
#0020
#0021 BEGIN_MESSAGE_MAP(CScribbleApp, CWinApp)
#0022     //{AFX_MSG_MAP(CScribbleApp)
#0023     ON_COMMAND(ID_APP_ABOUT, OnAppAbout)
#0024         // NOTE - the ClassWizard will add and remove mapping macros here.
#0025         //     DO NOT EDIT what you see in these blocks of generated code!
#0026     //}AFX_MSG_MAP
#0027     // Standard file based document commands
#0028     ON_COMMAND(ID_FILE_NEW, CWinApp::OnFileNew)
#0029     ON_COMMAND(ID_FILE_OPEN, CWinApp::OnFileOpen)
#0030     // Standard print setup command
#0031     ON_COMMAND(ID_FILE_PRINT_SETUP, CWinApp::OnFilePrintSetup)
#0032 END_MESSAGE_MAP()
#0033
#0034 ///////////////////////////////////////////////////
#0035 // CScribbleApp construction
#0036
#0037 CScribbleApp::CScribbleApp()
#0038 {
#0039     // TODO: add construction code here,
#0040     // Place all significant initialization in InitInstance
#0041 }
#0042
#0043 ///////////////////////////////////////////////////
#0044 // The one and only CScribbleApp object
#0045
#0046 CScribbleApp theApp;
#0047
#0048 ///////////////////////////////////////////////////
#0049 // CScribbleApp initialization
#0050
#0051 BOOL CScribbleApp::InitInstance()
#0052 {
#0053     // Standard initialization
#0054     // If you are not using these features and wish to reduce the size
#0055     // of your final executable, you should remove from the following
#0056     // the specific initialization routines you do not need.
#0057
#0058     #ifdef _AFXDLL
#0059         Enable3dControls();      // Call this when using MFC in a shared DLL
#0060     #else
#0061         Enable3dControlsStatic(); // Call this when linking to MFC statically
#0062     #endif
#0063 }
```



```
#0064 // 侯俊傑註：0065~0068 爲 Visual C++ 5.0 新增
#0065 // Change the registry key under which our settings are stored.
#0066 // You should modify this string to be something appropriate
#0067 // such as the name of your company or organization.
#0068 SetRegistryKey(_T("Local AppWizard-Generated Applications"));
#0069
#0070 LoadStdProfileSettings(); // Load std INI file options (including MRU)
#0071
#0072 // Register the application's document templates. Document templates
#0073 // serve as the connection between documents, frame windows and views.
#0074
#0075 CMultiDocTemplate* pDocTemplate;
#0076 pDocTemplate = new CMultiDocTemplate(
#0077     IDR_SCRIBTYPE,
#0078     RUNTIME_CLASS(CScribbleDoc),
#0079     RUNTIME_CLASS(CChildFrame), // custom MDI child frame
#0080     RUNTIME_CLASS(CScribbleView));
#0081 AddDocTemplate(pDocTemplate);
#0082
#0083 // create main MDI Frame window
#0084 CMainFrame* pMainFrame = new CMainFrame;
#0085 if (!pMainFrame->LoadFrame(IDR_MAINFRAME))
#0086     return FALSE;
#0087 m_pMainWnd = pMainFrame;
#0088
#0089 // Enable drag/drop open
#0090 m_pMainWnd->DragAcceptFiles();
#0091
#0092 // Enable DDE Execute open
#0093 EnableShellOpen();
#0094 RegisterShellFileTypes(TRUE);
#0095
#0096 // Parse command line for standard shell commands, DDE, file open
#0097 CCommandLineInfo cmdInfo;
#0098 ParseCommandLine(cmdInfo);
#0099
#0100 // Dispatch commands specified on the command line
#0101 if (!ProcessShellCommand(cmdInfo))
#0102     return FALSE;
#0103
#0104 // The main window has been initialized, so show and update it.
#0105 pMainFrame->ShowWindow(m_nCmdShow);
#0106 pMainFrame->UpdateWindow();
#0107
#0108 return TRUE;
#0109 }
```

```
#0110
#0111 ///////////////////////////////////////////////////
#0112 // CAboutDlg dialog used for App About
#0113
#0114 class CAboutDlg : public CDialog
#0115 {
#0116 public:
#0117     CAboutDlg();
#0118
#0119 // Dialog Data
#0120 //{{AFX_DATA(CAboutDlg)
#0121 enum { IDD = IDD_ABOUTBOX };
#0122 //}}AFX_DATA
#0123
#0124 // ClassWizard generated virtual function overrides
#0125 //{{AFX_VIRTUAL(CAboutDlg)
#0126 protected:
#0127     virtual void DoDataExchange(CDataExchange* pDX); // DDX/DDV support
#0128 //}}AFX_VIRTUAL
#0129
#0130 // Implementation
#0131 protected:
#0132 //{{AFX_MSG(CAboutDlg)
#0133     // No message handlers
#0134 //}}AFX_MSG
#0135 DECLARE_MESSAGE_MAP()
#0136 };
#0137
#0138 CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
#0139 {
#0140     //{{AFX_DATA_INIT(CAboutDlg)
#0141     //}}AFX_DATA_INIT
#0142 }
#0143
#0144 void CAboutDlg::DoDataExchange(CDataExchange* pDX)
#0145 {
#0146     CDialog::DoDataExchange(pDX);
#0147 //{{AFX_DATA_MAP(CAboutDlg)
#0148 //}}AFX_DATA_MAP
#0149 }
#0150
#0151 BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
#0152 //{{AFX_MSG_MAP(CAboutDlg)
#0153     // No message handlers
#0154 //}}AFX_MSG_MAP
#0155 END_MESSAGE_MAP()
```

```
#0156
#0157 // App command to run the dialog
#0158 void CScribbleApp::OnAppAbout()
#0159 {
#0160     CAboutDlg aboutDlg;
#0161     aboutDlg.DoModal();
#0162 }
#0163
#0164 //////////////////////////////////////
#0165 // CScribbleApp commands
```

## MAINFRM.CPP

```
#0001 // MainFrm.cpp : implementation of the CMainFrame class
#0002 //
#0003
#0004 #include "stdafx.h"
#0005 #include "Scribble.h"
#0006
#0007 #include "MainFrm.h"
#0008
#0009 #ifdef _DEBUG
#0010 #define new DEBUG_NEW
#0011 #undef THIS_FILE
#0012 static char THIS_FILE[] = __FILE__;
#0013 #endif
#0014
#0015 //////////////////////////////////////
#0016 // CMainFrame
#0017
#0018 IMPLEMENT_DYNAMIC(CMainFrame, CMDIFrameWnd)
#0019
#0020 BEGIN_MESSAGE_MAP(CMainFrame, CMDIFrameWnd)
#0021     //{AFX_MSG_MAP(CMainFrame)
#0022         // NOTE - the ClassWizard will add and remove mapping macros here.
#0023         //      DO NOT EDIT what you see in these blocks of generated code !
#0024     ON_WM_CREATE()
#0025     //}AFX_MSG_MAP
#0026 END_MESSAGE_MAP()
#0027
#0028 static UINT indicators[] =
#0029 {
#0030     ID_SEPARATOR,          // status line indicator
#0031     ID_INDICATOR_CAPS,
#0032     ID_INDICATOR_NUM,
#0033     ID_INDICATOR_SCRL,
```

```
#0034 };
#0035
#0036 //////////////////////////////////////
#0037 // CMainFrame construction/destruction
#0038
#0039 CMainFrame::CMainFrame()
#0040 {
#0041     // TODO: add member initialization code here
#0042
#0043 }
#0044
#0045 CMainFrame::~CMainFrame()
#0046 {
#0047 }
#0048
#0049 int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
#0050 {
#0051     if (CMDIFrameWnd::OnCreate(lpCreateStruct) == -1)
#0052         return -1;
#0053
#0054     if (!m_wndToolBar.Create(this) ||
#0055         !m_wndToolBar.LoadToolBar(IDR_MAINFRAME))
#0056     {
#0057         TRACE0("Failed to create toolbar\n");
#0058         return -1;    // fail to create
#0059     }
#0060
#0061     if (!m_wndStatusBar.Create(this) ||
#0062         !m_wndStatusBar.SetIndicators(indicators,
#0063         sizeof(indicators)/sizeof(UINT)))
#0064     {
#0065         TRACE0("Failed to create status bar\n");
#0066         return -1;    // fail to create
#0067     }
#0068
#0069     // TODO: Remove this if you don't want tool tips or a resizeable toolbar
#0070     m_wndToolBar.SetBarStyle(m_wndToolBar.GetBarStyle() |
#0071         CBRS_TOOLTIPS | CBRS_FLYBY | CBRS_SIZE_DYNAMIC);
#0072
#0073     // TODO: Delete these three lines if you don't want the toolbar to
#0074     // be dockable
#0075     m_wndToolBar.EnableDocking(CBRS_ALIGN_ANY);
#0076     EnableDocking(CBRS_ALIGN_ANY);
#0077     DockControlBar(&m_wndToolBar);
#0078
#0079     return 0;
```

```
#0080 }
#0081
#0082 BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)
#0083 {
#0084     // TODO: Modify the Window class or styles here by modifying
#0085     // the CREATESTRUCT cs
#0086
#0087     return CMDIFrameWnd::PreCreateWindow(cs);
#0088 }
#0089
#0090 //////////////////////////////////////////////////
#0091 // CMainFrame diagnostics
#0092
#0093 #ifdef _DEBUG
#0094 void CMainFrame::AssertValid() const
#0095 {
#0096     CMDIFrameWnd::AssertValid();
#0097 }
#0098
#0099 void CMainFrame::Dump(CDumpContext& dc) const
#0100 {
#0101     CMDIFrameWnd::Dump(dc);
#0102 }
#0103
#0104 #endif //_DEBUG
#0105
#0106 //////////////////////////////////////////////////
#0107 // CMainFrame message handlers
```

## CHILDFRM.CPP

```
#0001 // ChildFrm.cpp : implementation of the CChildFrame class
#0002 //
#0003
#0004 #include "stdafx.h"
#0005 #include "Scribble.h"
#0006
#0007 #include "ChildFrm.h"
#0008
#0009 #ifdef _DEBUG
#0010 #define new DEBUG_NEW
#0011 #undef THIS_FILE
#0012 static char THIS_FILE[] = __FILE__;
#0013 #endif
#0014
#0015 //////////////////////////////////////////////////
```

```
#0016 // CChildFrame
#0017
#0018 IMPLEMENT_DYNCREATE(CChildFrame, CMDIChildWnd)
#0019
#0020 BEGIN_MESSAGE_MAP(CChildFrame, CMDIChildWnd)
#0021     //{AFX_MSG_MAP(CChildFrame)
#0022         // NOTE - the ClassWizard will add and remove mapping macros here.
#0023         // DO NOT EDIT what you see in these blocks of generated code !
#0024     //}AFX_MSG_MAP
#0025 END_MESSAGE_MAP()
#0026
#0027 //////////////////////////////////////
#0028 // CChildFrame construction/destruction
#0029
#0030 CChildFrame::CChildFrame()
#0031 {
#0032     // TODO: add member initialization code here
#0033 }
#0034
#0035
#0036 CChildFrame::~CChildFrame()
#0037 {
#0038 }
#0039
#0040 BOOL CChildFrame::PreCreateWindow(CREATESTRUCT& cs)
#0041 {
#0042     // TODO: Modify the Window class or styles here by modifying
#0043     // the CREATESTRUCT cs
#0044
#0045     return CMDIChildWnd::PreCreateWindow(cs);
#0046 }
#0047
#0048 //////////////////////////////////////
#0049 // CChildFrame diagnostics
#0050
#0051 #ifdef _DEBUG
#0052 void CChildFrame::AssertValid() const
#0053 {
#0054     CMDIChildWnd::AssertValid();
#0055 }
#0056
#0057 void CChildFrame::Dump(CDumpContext& dc) const
#0058 {
#0059     CMDIChildWnd::Dump(dc);
#0060 }
#0061
```

```
#0062 #endif //_DEBUG
#0063
#0064 //////////////////////////////////////
#0065 // CChildFrame message handlers
```

## SCRIBBLEDOC.CPP

```
#0001 // ScribbleDoc.cpp : implementation of the CScribbleDoc class
#0002 //
#0003
#0004 #include "stdafx.h"
#0005 #include "Scribble.h"
#0006
#0007 #include "ScribbleDoc.h"
#0008
#0009 #ifdef _DEBUG
#0010 #define new DEBUG_NEW
#0011 #undef THIS_FILE
#0012 static char THIS_FILE[] = __FILE__;
#0013 #endif
#0014
#0015 //////////////////////////////////////
#0016 // CScribbleDoc
#0017
#0018 IMPLEMENT_DYNCREATE(CScribbleDoc, CDocument)
#0019
#0020 BEGIN_MESSAGE_MAP(CScribbleDoc, CDocument)
#0021     //{AFX_MSG_MAP(CScribbleDoc)
#0022         // NOTE - the ClassWizard will add and remove mapping macros here.
#0023         // DO NOT EDIT what you see in these blocks of generated code!
#0024     //}AFX_MSG_MAP
#0025 END_MESSAGE_MAP()
#0026
#0027 //////////////////////////////////////
#0028 // CScribbleDoc construction/destruction
#0029
#0030 CScribbleDoc::CScribbleDoc()
#0031 {
#0032     // TODO: add one-time construction code here
#0033
#0034 }
#0035
#0036 CScribbleDoc::~CScribbleDoc()
#0037 {
#0038 }
#0039
```

```
#0040 BOOL CScribbleDoc::OnNewDocument()  
#0041 {  
#0042     if (!CDocument::OnNewDocument())  
#0043         return FALSE;  
#0044  
#0045     // TODO: add reinitialization code here  
#0046     // (SDI documents will reuse this document)  
#0047  
#0048     return TRUE;  
#0049 }  
#0050  
#0051 ///////////////////////////////////////////////////  
#0052 // CScribbleDoc serialization  
#0053  
#0054 void CScribbleDoc::Serialize(CArchive& ar)  
#0055 {  
#0056     if (ar.IsStoring())  
#0057     {  
#0058         // TODO: add storing code here  
#0059     }  
#0060     else  
#0061     {  
#0062         // TODO: add loading code here  
#0063     }  
#0064 }  
#0065  
#0066 ///////////////////////////////////////////////////  
#0067 // CScribbleDoc diagnostics  
#0068  
#0069 #ifdef _DEBUG  
#0070 void CScribbleDoc::AssertValid() const  
#0071 {  
#0072     CDocument::AssertValid();  
#0073 }  
#0074  
#0075 void CScribbleDoc::Dump(CDumpContext& dc) const  
#0076 {  
#0077     CDocument::Dump(dc);  
#0078 }  
#0079 #endif // _DEBUG  
#0080  
#0081 ///////////////////////////////////////////////////  
#0082 // CScribbleDoc commands
```



**SCRIBBLEVIEW.CPP**

```
#0001 // ScribbleView.cpp : implementation of the CScribbleView class
#0002 //
#0003
#0004 #include "stdafx.h"
#0005 #include "Scribble.h"
#0006
#0007 #include "ScribbleDoc.h"
#0008 #include "ScribbleView.h"
#0009
#0010 #ifdef _DEBUG
#0011 #define new DEBUG_NEW
#0012 #undef THIS_FILE
#0013 static char THIS_FILE[] = __FILE__;
#0014 #endif
#0015
#0016 //////////////////////////////////////
#0017 // CScribbleView
#0018
#0019 IMPLEMENT_DYNCREATE(CScribbleView, CView)
#0020
#0021 BEGIN_MESSAGE_MAP(CScribbleView, CView)
#0022     //{AFX_MSG_MAP(CScribbleView)
#0023         // NOTE - the ClassWizard will add and remove mapping macros here.
#0024         //      DO NOT EDIT what you see in these blocks of generated code!
#0025     //}AFX_MSG_MAP
#0026     // Standard printing commands
#0027     ON_COMMAND(ID_FILE_PRINT, CView::OnFilePrint)
#0028     ON_COMMAND(ID_FILE_PRINT_DIRECT, CView::OnFilePrint)
#0029     ON_COMMAND(ID_FILE_PRINT_PREVIEW, CView::OnFilePrintPreview)
#0030 END_MESSAGE_MAP()
#0031
#0032 //////////////////////////////////////
#0033 // CScribbleView construction/destruction
#0034
#0035 CScribbleView::CScribbleView()
#0036 {
#0037     // TODO: add construction code here
#0038 }
#0039
#0040
#0041 CScribbleView::~CScribbleView()
#0042 {
#0043 }
#0044
```

```
#0045 BOOL CScribbleView::PreCreateWindow(CREATESTRUCT& cs)
#0046 {
#0047     // TODO: Modify the Window class or styles here by modifying
#0048     // the CREATESTRUCT cs
#0049
#0050     return CView::PreCreateWindow(cs);
#0051 }
#0052
#0053 //////////////////////////////////////////////////
#0054 // CScribbleView drawing
#0055
#0056 void CScribbleView::OnDraw(CDC* pDC)
#0057 {
#0058     CScribbleDoc* pDoc = GetDocument();
#0059     ASSERT_VALID(pDoc);
#0060
#0061     // TODO: add draw code for native data here
#0062 }
#0063
#0064 //////////////////////////////////////////////////
#0065 // CScribbleView printing
#0066
#0067 BOOL CScribbleView::OnPreparePrinting(CPrintInfo* pInfo)
#0068 {
#0069     // default preparation
#0070     return DoPreparePrinting(pInfo);
#0071 }
#0072
#0073 void CScribbleView::OnBeginPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
#0074 {
#0075     // TODO: add extra initialization before printing
#0076 }
#0077
#0078 void CScribbleView::OnEndPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
#0079 {
#0080     // TODO: add cleanup after printing
#0081 }
#0082
#0083 //////////////////////////////////////////////////
#0084 // CScribbleView diagnostics
#0085
#0086 #ifdef _DEBUG
#0087 void CScribbleView::AssertValid() const
#0088 {
#0089     CView::AssertValid();
#0090 }
```

```
#0091
#0092 void CScribbleView::Dump(CDumpContext& dc) const
#0093 {
#0094     CView::Dump(dc);
#0095 }
#0096
#0097 CScribbleDoc* CScribbleView::GetDocument() // non-debug version is
#0098 {                                           // inline
#0099     ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CScribbleDoc)));
#0100     return (CScribbleDoc*)m_pDocument;
#0101 }
#0102 #endif //_DEBUG
#0103
#0104 //////////////////////////////////////
#0105 // CScribbleView message handlers
```

#### SCRIBBLE.RC (以下之碼已經修剪，列出的主要目的是讓你瞭解共有多少資源)

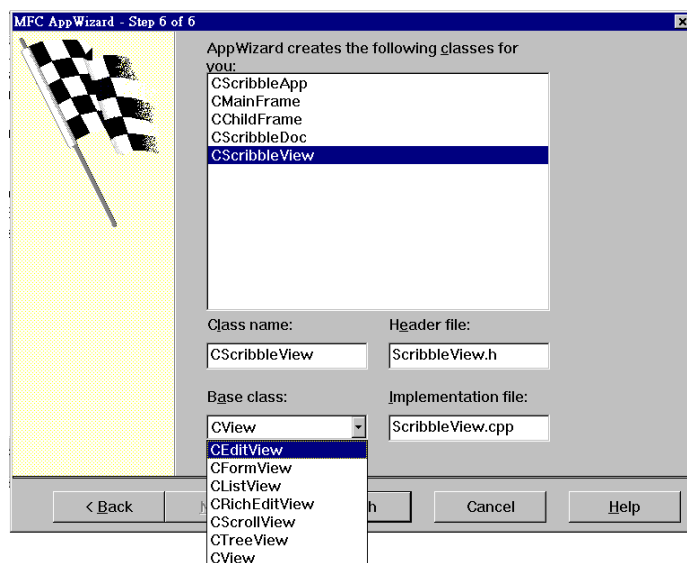
```
#0001 //Microsoft Visual C++ generated resource script.
#0002 //
#0003
#0004 #include "resource.h"
#0005 #include "afxres.h"
#0006
#0007 IDR_MAINFRAME          ICON      DISCARDABLE    "res\\Scribble.ico"
#0008 IDR_SCRIBTYPE          ICON      DISCARDABLE    "res\\ScribbleDoc.ico"
#0009
#0010 IDR_MAINFRAME          BITMAP     MOVEABLE PURE   "res\\Toolbar.bmp"
#0011
#0012 IDR_MAINFRAME TOOLBAR DISCARDABLE 16, 15
#0013 BEGIN
#0014     BUTTON      ID_FILE_NEW
#0015     BUTTON      ID_FILE_OPEN
#0016     BUTTON      ID_FILE_SAVE
#0017     SEPARATOR
#0018     BUTTON      ID_EDIT_CUT
#0019     BUTTON      ID_EDIT_COPY
#0020     BUTTON      ID_EDIT_PASTE
#0021     SEPARATOR
#0022     BUTTON      ID_FILE_PRINT
#0023     BUTTON      ID_APP_ABOUT
#0024 END
#0025
#0026 IDR_MAINFRAME MENU PRELOAD DISCARDABLE
#0027 BEGIN
#0028     POPUP "&File"
```

```
#0029         BEGIN
#0030             ...
#0031         END
#0032         POPUP "&View"
#0033         BEGIN
#0034             ...
#0035         END
#0036         POPUP "&Help"
#0037         BEGIN
#0038             ...
#0039         END
#0040     END
#0041
#0042     IDR_SCRIBTYPE MENU PRELOAD DISCARDABLE
#0043     BEGIN
#0044         POPUP "&File"
#0045         BEGIN
#0046             ...
#0047         END
#0048         POPUP "&Edit"
#0049         BEGIN
#0050             ...
#0051         END
#0052         POPUP "&View"
#0053         BEGIN
#0054             ...
#0055         END
#0056         POPUP "&Window"
#0057         BEGIN
#0058             ...
#0059         END
#0060         POPUP "&Help"
#0061         BEGIN
#0062             ...
#0063         END
#0064     END
#0065
#0066     IDR_MAINFRAME ACCELERATORS PRELOAD MOVEABLE PURE
#0067     BEGIN
#0068         ...
#0069     END
#0070
#0071     IDD_ABOUTBOX DIALOG DISCARDABLE 0, 0, 217, 55
#0072     CAPTION "About Scribble"
#0073     STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
#0074     FONT 8, "MS Sans Serif"
```

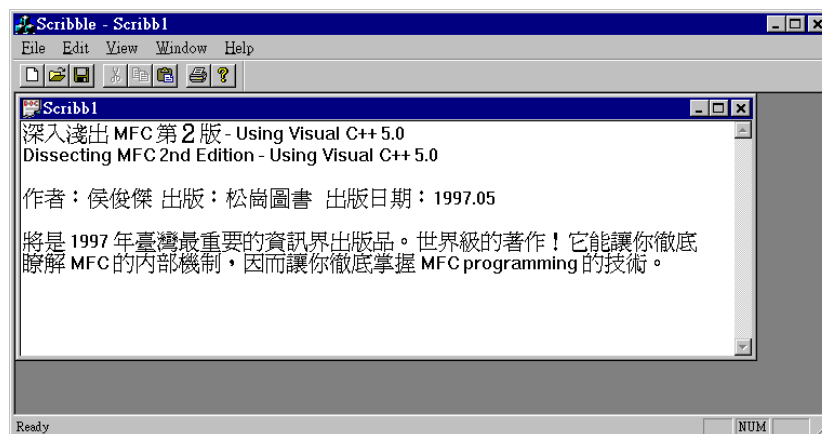
```
#0075 BEGIN
#0076      ...
#0077 END
#0078
#0079 VS_VERSION_INFO      VERSIONINFO
#0080     FILEVERSION      1,0,0,1
#0081     PRODUCTVERSION  1,0,0,1
#0082     FILEFLAGSMASK 0x3fL
#0083     #ifdef _DEBUG
#0084     FILEFLAGS 0x1L
#0085     #else
#0086     FILEFLAGS 0x0L
#0087     #endif
#0088     FILEOS 0x4L
#0089     FILETYPE 0x1L
#0090     FILESUBTYPE 0x0L
#0091 BEGIN
#0092     BLOCK "StringFileInfo"
#0093     BEGIN
#0094         BLOCK "040904B0"
#0095         BEGIN
#0096             VALUE "CompanyName",      "\0"
#0097             VALUE "FileDescription", "Scribble MFC Application\0"
#0098             VALUE "FileVersion",      "1, 0, 0, 1\0"
#0099             VALUE "InternalName",     "Scribble\0"
#0100             VALUE "LegalCopyright",  "Copyright (C) 1997\0"
#0101             VALUE "LegalTrademarks", "\0"
#0102             VALUE "OriginalFilename", "Scribble.EXE\0"
#0103             VALUE "ProductName",     "Scribble Application\0"
#0104             VALUE "ProductVersion",  "1, 0, 0, 1\0"
#0105         END
#0106     END
#0107     BLOCK "VarFileInfo"
#0108     BEGIN
#0109         VALUE "Translation", 0x409, 1200
#0110     END
#0111 END
#0112
#0113 //////////////////////////////////////
#0114 // String Table
#0115
#0116 STRINGTABLE PRELOAD DISCARDABLE
#0117 BEGIN
#0118     IDR_MAINFRAME "Scribble"
#0119     IDR_SCRIBTYPE "\nScrib\nScrib\nScribb Files
(*.scb)\n.scb\nScribble.Document\nScrib Document"
```

```
#0120 END
#0121
#0122 STRINGTABLE PRELOAD DISCARDABLE
#0123 BEGIN
#0124     AFX_IDS_APP_TITLE      "Scribble"
#0125     AFX_IDS_IDLEMESSAGE    "Ready"
#0126 END
#0127
#0128 STRINGTABLE DISCARDABLE
#0129 BEGIN
#0130     ID_INDICATOR_EXT        "EXT"
#0131     ID_INDICATOR_CAPS       "CAP"
#0132     ID_INDICATOR_NUM        "NUM"
#0133     ID_INDICATOR_SCRL       "SCRL"
#0134     ID_INDICATOR_OVR        "OVR"
#0135     ID_INDICATOR_REC        "REC"
#0136 END
#0137
#0138 STRINGTABLE DISCARDABLE
#0139 BEGIN
#0140     ID_FILE_NEW              "Create a new document\nNew"
#0141     ID_FILE_OPEN              "Open an existing document\nOpen"
#0142     ID_FILE_CLOSE             "Close the active document\nClose"
#0143     ID_FILE_SAVE              "Save the active document\nSave"
#0144     ...
#0145 END
```

好，我曾經說過，這個程式漂亮歸漂亮，可什麼也沒做。我知道 MFC 中有一個 *CEditView* 類別，具有文字編輯功能，我打算從那裡繼承我的 View（現在的你還不了解什麼是 View，沒關係）。於是我重來一次，一切都相同，只在 AppWizard 的步驟六中設定 *CScribbleView* 的【Base class:】為 *CEditView*：



這次我獲得這樣一個程式：



天啊，它不但有文字編輯功能，更有令人匪夷所思的印表功能和預視功能，也可以讀寫文字檔。

體會驚人的生產力了嗎？

注意：在 MFC AppWizard 的步驟 6 中把 *CScribbleView* 的基礎類別由 *CView* 改為 *CEditView*，會造成原始碼如下的變化（粗體部份）：

```
// in ScribbleView.h
class CScribbleView : public CEditView
{
    ...
}

// in ScribbleView.cpp
IMPLEMENT_DYNCREATE(CScribbleView, CEditView)

BEGIN_MESSAGE_MAP(CScribbleView, CEditView)
    ...
    ON_COMMAND(ID_FILE_PRINT, CEditView::OnFilePrint)
    ON_COMMAND(ID_FILE_PRINT_DIRECT, CEditView::OnFilePrint)
    ON_COMMAND(ID_FILE_PRINT_PREVIEW, CEditView::OnFilePrintPreview)
END_MESSAGE_MAP()
// ScribbleView.cpp 中所有原先為 CView 的地方，都被更改為 CEEditView

// in ScribbleDoc.cpp
void CScribbleDoc::Serialize(CArchive& ar)
{
    // CEditView contains an edit control which handles all serialization
    ((CEditView*)m_viewList.GetHead())->SerializeRaw(ar);
}
```

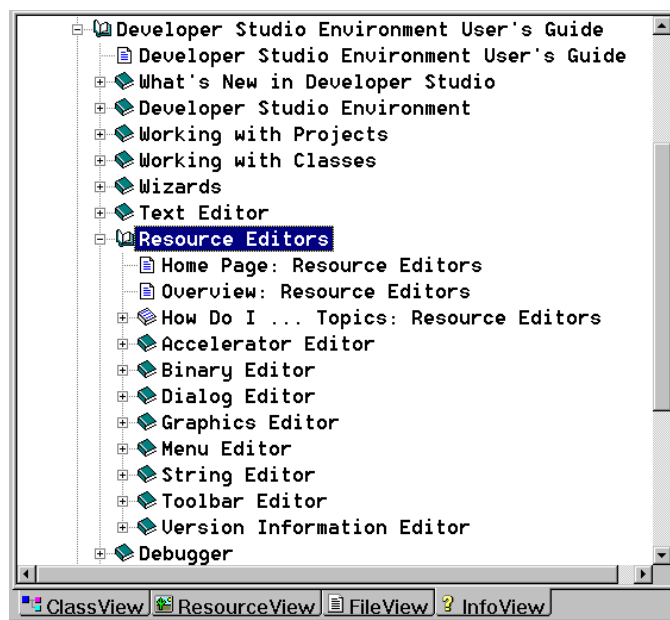


## 或可強大的資源編輯器

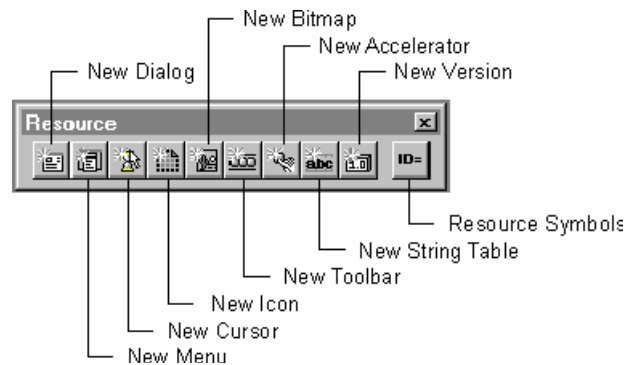
AppWizard 做出來的骨幹程式帶給我們 Windows 程式的標準 UI 介面。爲了個人需求，你當然會另外加上一些資源，這時候你得準備啓用資源編輯工具了。如果你曾經是 Visual C++ 的使用者，當記得曾有一個名爲 AppStudio 的多效合一資源編輯工具。是了，但現在不再有 AppStudio，不再有獨立的資源編輯工具，而是與 Visual C++ 整合環境做了更密切的結合。

我將對這個工具提供的各種資源編輯功能逐一簡介，並以實例展示如何在應用程式中加入新的資源項目。

資源的編輯，雖然與「正統」程式設計扯不上關係，但資源在 Windows 程式所佔的份量，衆所週知。運用這些工具，仍然是你工作中重要的一環。VC++ 的 Online 手冊上有頗爲完整的介紹；本章不能取代它們的地位，只是企圖給你一個整體概觀。以下是出現在 InfoView 視窗中的 Developer Studio Environment User's Guide 目錄：



打開一個專案後，你可以從其 **ResourceView** 視窗中看到所有的資源。想要編輯哪一個資源，就以滑鼠雙擊之。如果要產生新的資源，整合環境的工具列上有一整排的按鈕等著你按。這個「資源工具列」是選擇性的，你可以按下整合環境的【Tools/Customize】選單項目，再選擇【Toolbar】附頁（或是直接在工具列區域中按下滑鼠右鍵），從中決定要看到或不看到哪些工具列。



選按其中任何一個鈕，立刻會有一個適當的編輯器跳出來向你說哈囉。

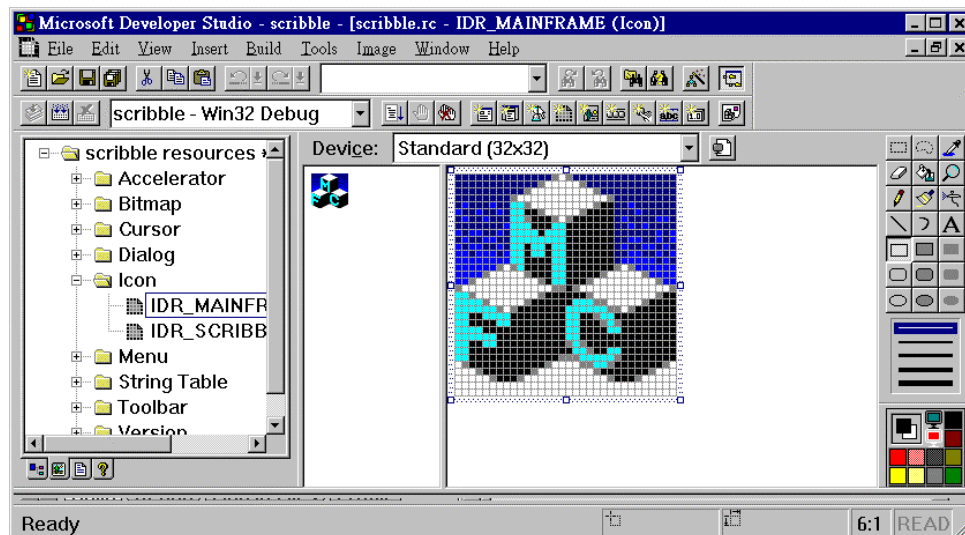
當然你可以用 PE2 老古董直接編輯 RC 檔，但整合環境的好處是它會自動處理 ID 號碼，避免重複的情況發生，新的 ID 並會自動放到你的 RESOURCE.H 檔中。總之就如我說過的，這些工具的目的在使你專注於最主要的工作上，至於各檔案間的關聯工作，枝枝節節的瑣碎事情，都由工具來完成。這，才叫作「整合性」工具環境嘛！

## Icon 編輯器

Icon、Cursor、Bitmap 和 Toolbar 編輯器使用同一個心臟：它們架構在同一個圖形編輯器上，操作大同小異。過去這個心臟曾經遺漏兩項重要功能，一是 256 色圖形支援，一是「敲入文字就出現對應之 Bitmap」工具（這種工具允許使用者將文字直接鍵入一張 bitmap 中，而不是一次一個圖素慢慢地描）。自從 Visual C++ 4.0 之後這兩項重要功能就已經完全補齊了。

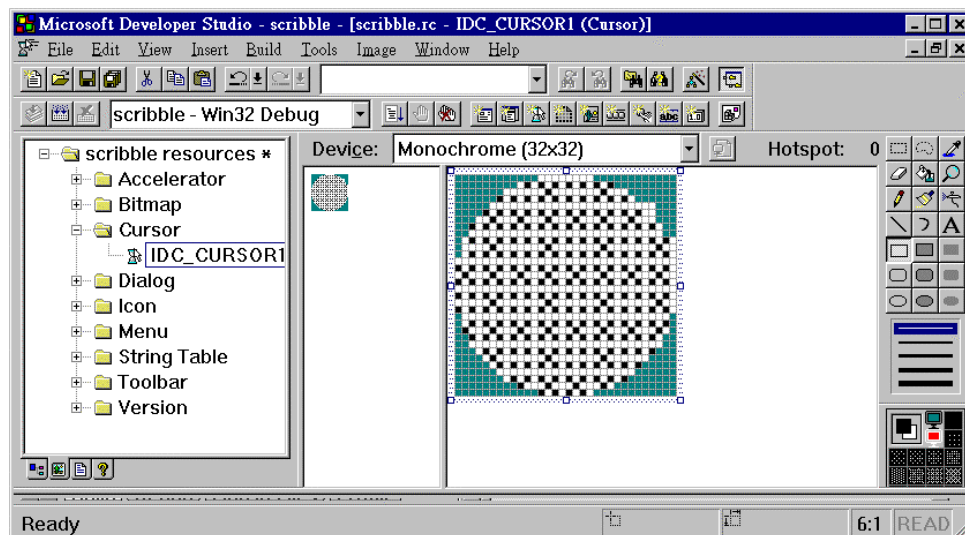
請注意工具箱（圖最右側）在不同的編輯器中稍有變化。

選按圖 7-1 ResourceView 中的 Icon，於是右側出現 Icon 編輯器。



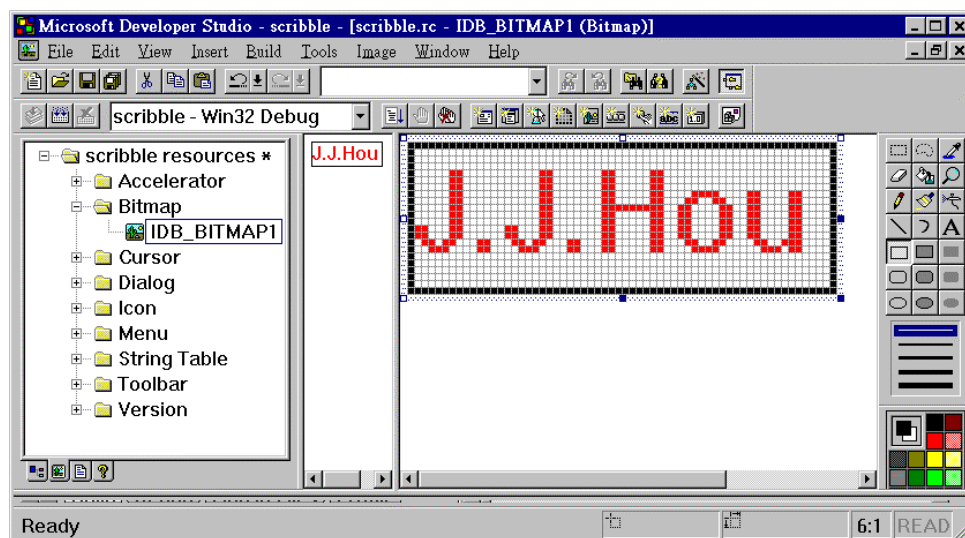
## Cursor 編輯器

選按圖 7-2 ResourceView 中的 Cursor，於是右側出現 Cursor 編輯器。



## Bitmap 編輯器

選按圖 4-1 的 ResourceView 中的 Bitmap，於是右側出現 Bitmap 編輯器。注意，本圖的 J.J.Hou 字樣並非一點一點描繪而成，而是利用繪圖工具箱（圖最右）中的字形產生器（標有 A 字形的圖示）。它不但能夠產生各種字形變化（視你安裝的字形種類而定），在 DV 環境下還能夠輸入 DV 字！不過我還沒有找到能夠調整字形大小的功能。

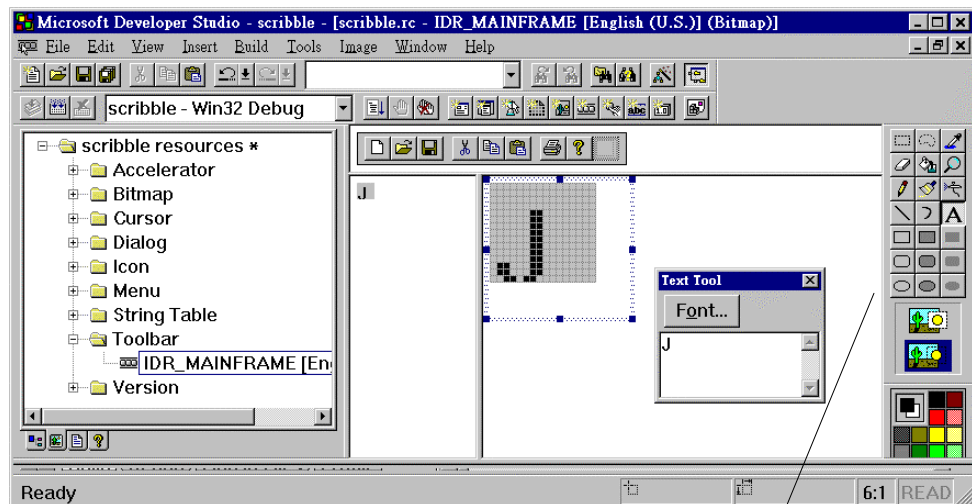


## 工具列 (Toolbar) 編輯器

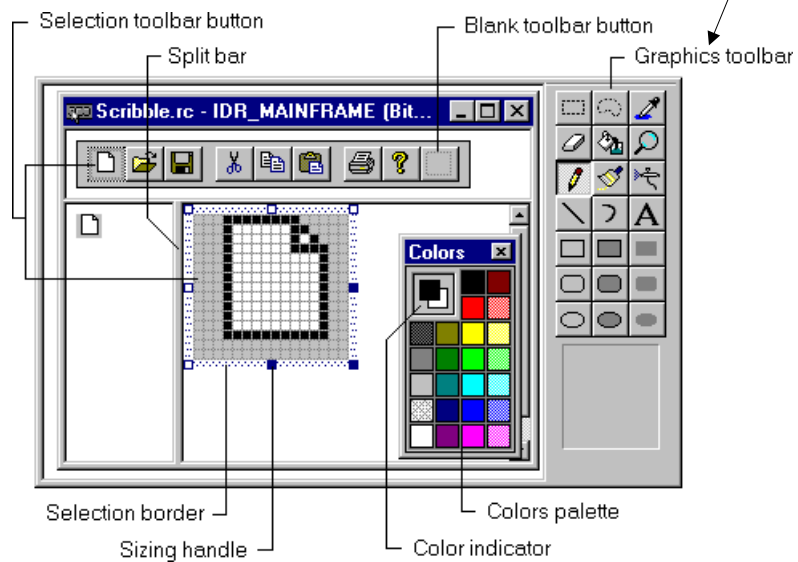
Visual C++ 早期版本沒有這個編輯器，因為，工具列原本不算是 RC 檔中的一份資源。而且，說穿了工具列其實只是靠一張由固定大小之格狀單元組成的一單張 bitmap 構成，編輯工具列其實就是編輯該張 bitmap。但是那樣一來，我們就得自己改寫程式碼中有關於工具列的設定部份，編輯程序顯得不夠一氣呵成！

自從 Visual C++ 4.0 開始，這中一切瑣事就都由工具代勞了。我將在第 7 章詳細解釋「工具列」資源如何在程式中發生效用。

選按圖 7-1 ResourceView 中的 IDR\_MAINFRAME 的 Toolbar，於是右側出現 Toolbar 編輯器。



把上圖局部放大來看：

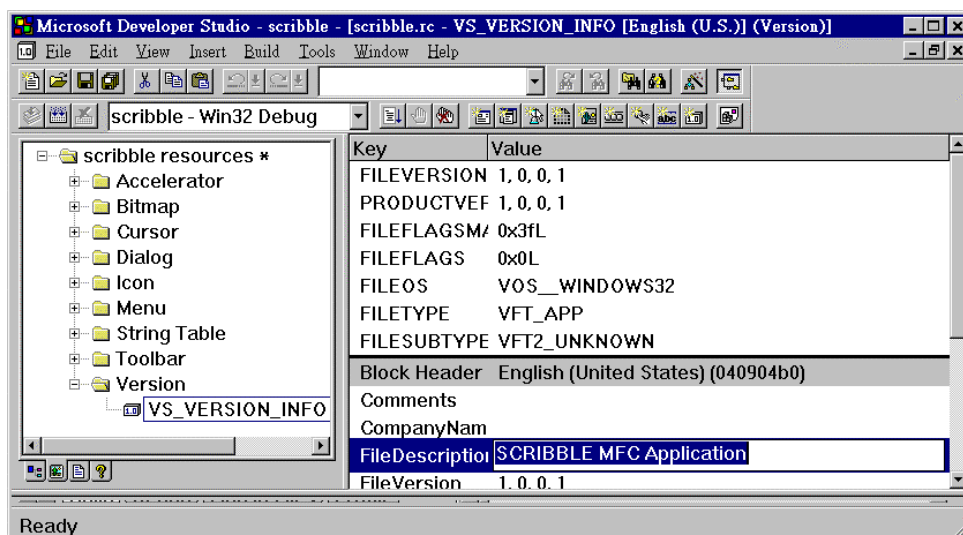


## VERSIONINFO 資源編輯器

VERSIONINFO 可幫助程式判斷存在於使用者系統中的檔案版本號碼，如此一來就不會發生「以舊版本程式改寫新格式之檔案」的遺憾了。VERSIONINFO 資源也放在 RC 檔，包含的資料可以識別版本、語言、作業系統、或含有資源之 DLL。AppWizard 會為你產生一份 VERSIONINFO 資源，但不強制你用它。下面是 Scribble.rc 檔中有關於 VERSIONINFO 的內容：

```
#0001 VS_VERSION_INFO      VERSIONINFO
#0002     FILEVERSION      1,0,0,1
#0003     PRODUCTVERSION   1,0,0,1
#0004     FILEFLAGSMASK    0x3fL
#0005     #ifdef _DEBUG
#0006         FILEFLAGS     0x1L
#0007     #else
#0008         FILEFLAGS     0x0L
#0009     #endif
#0010     FILEOS            0x4L
#0011     FILETYPE          0x1L
#0012     FILESUBTYPE       0x0L
#0013 BEGIN
#0014     BLOCK "StringFileInfo"
#0015     BEGIN
#0016         BLOCK "040904B0"
#0017         BEGIN
#0018             VALUE "CompanyName",      "\0"
#0019             VALUE "FileDescription",   "SCRIBBLE MFC Application\0"
#0020             VALUE "FileVersion",      "1, 0, 0, 1\0"
#0021             VALUE "InternalName",     "SCRIBBLE\0"
#0022             VALUE "LegalCopyright",   "Copyright \251 1996\0"
#0023             VALUE "LegalTrademarks",  "\0"
#0024             VALUE "OriginalFilename", "SCRIBBLE.EXE\0"
#0025             VALUE "ProductName",     "SCRIBBLE Application\0"
#0026             VALUE "ProductVersion",   "1, 0, 0, 1\0"
#0027         END
#0028     END
#0029     BLOCK "VarFileInfo"
#0030     BEGIN
#0031         VALUE "Translation", 0x409, 1200
#0032     END
#0033 END
```

選按圖 7-1 ResourceView 中的「VersionInfo」，於是右側出現 VersionInfo 編輯器。你可以直接在各「項目」上修改字串內容。



## 字串表格 (String Table) 編輯器

字串表格編輯器非常好用，允許你編輯 RC 檔中的字串資源 (STRINGTABLE)，這可增進國際化的腳步。怎麼說？我們可以把程式中出現的所有字串都集中在 RC 檔的字串表格，日後做中文版、日文版、法文版時只要改變 RC 檔的字串表格即可。噢當然，你還得選一套適當的 Common Dialog DLL。

AppWizard 為我們製作骨幹程式時不是加了一大套 Menu 嗎，對應於這些 Menu，有數以打計的字串資源，準備給狀態列使用。下面是 RC 檔字串表格的一小部份：

```

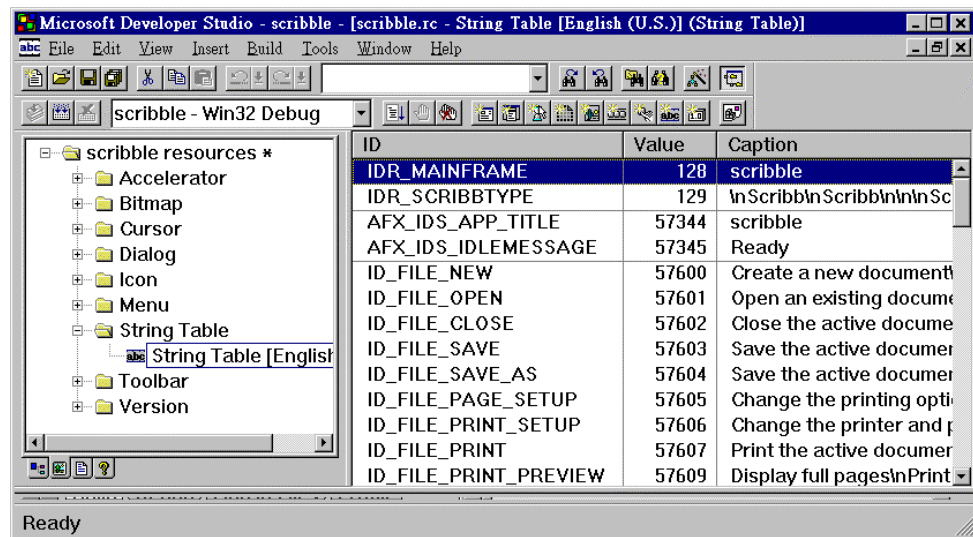
STRINGTABLE DISCARDABLE
BEGIN
    ID_INDICATOR_EXT    "EXT"
    ID_INDICATOR_CAPS   "CAP"
    ID_INDICATOR_NUM     "NUM"
    ID_INDICATOR_SCRL    "SCRL"
    ID_INDICATOR_OVR     "OVR"
    ID_INDICATOR_REC     "REC"
END
    
```

```

STRINGTABLE DISCARDABLE
BEGIN
    ID_FILE_NEW          "Create a new document\nNew"
    ID_FILE_OPEN          "Open an existing document\nOpen"
    ID_FILE_CLOSE         "Close the active document\nClose"
    ID_FILE_SAVE          "Save the active document\nSave"
    ID_FILE_SAVE_AS       "Save the active document with a new name\nSave As"
    ...

```

選按圖 4-1 ResourceView 中的 一個 String Table，於是右側出現 String Table 編輯器。你可以直接在其中一個字串上修改內容。

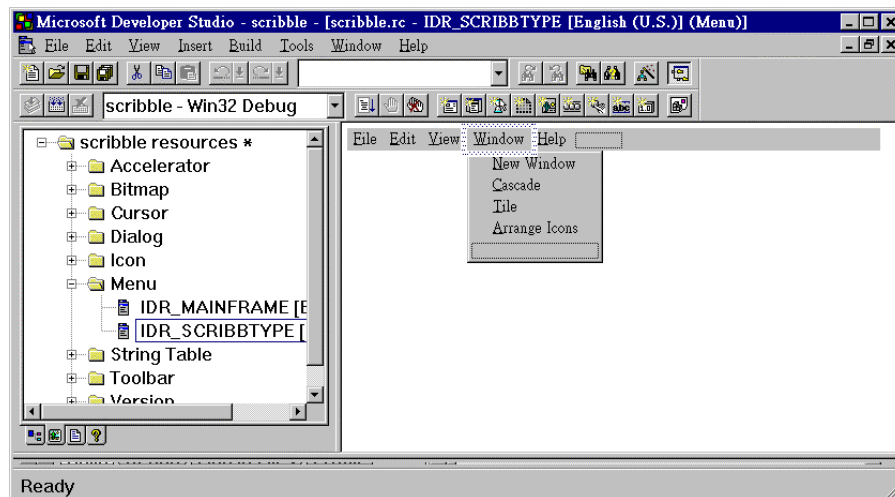


## 選單 (Menu) 編輯器

選單編輯器很好用。你可以一邊看到正在建立的選單，一邊直接在適當位置鍵入選單項目名稱，表單編輯器會把選單項目的 ID 值（當然是它自動為你產生的）放到 RESOURCE.H 的 #define 敘述中，就像字串表格編輯器所做的那樣。重新安排選單項目的位置也很容易，因為所有動作都可以滑鼠拖拉方式完成。



選按圖 1 ResourceView 中的 `Menu`，於是右側出現 Menu 編輯器。



假設我在選單上添加一份 popup 選單，內有“JJHou”和“MJChen”兩個項目。不但 RC 檔的 MENU 資源有了變化：

```
IDR_MYTYPE MENU PRELOAD DISCARDABLE
BEGIN
    ...
    POPUP "MyFamily"
    BEGIN
        MENUITEM "JJHou", ID_MYFAMILY_JJHOU
        MENUITEM "MJChen", ID_MYFAMILY_MJCHEN
    END
END
```

STRINGTABLE 也多了兩個字串定義，作為狀態列訊息：

```
STRINGTABLE DISCARDABLE
BEGIN
    ID_MYFAMILY_JJHOU "J.J.Hou is a Good man"
    ID_MYFAMILY_MJCHEN "M.J.Chen is a Good woman"
END
```

此外，RESOURCE.H 也多了兩個常數定義：

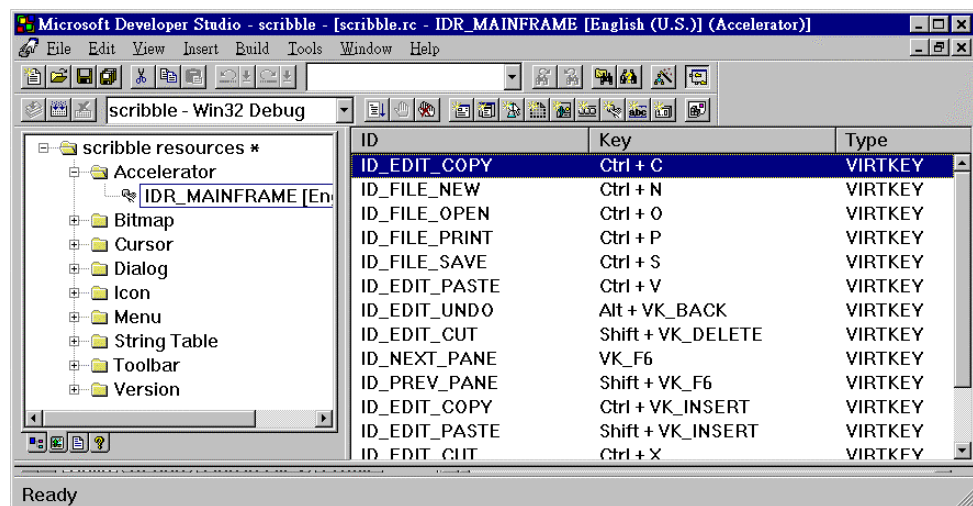
```
#define ID_MYFAMILY_JJHOU 32771
#define ID_MYFAMILY_MJCHEN 32772
```

此外也造成 .CLW 檔的變化，好讓 ClassWizard 知悉。ClassWizard 將在稍後介紹。

## 加速鍵 (Accelerator) 編輯器


AppWizard 已經為骨幹程式中的許多標準選單項目設計了加速鍵。通常加速鍵是兩個按鍵的組合（例如 Alt + N），用以取代滑鼠在層層選單中的拉下、選按動作。所有的加速鍵設定都集中在 RC 檔的加速鍵表格中，雙擊其中任何一個，就會出現加速鍵編輯器為你服務。你可以利用它改變加速鍵的按鍵組合。

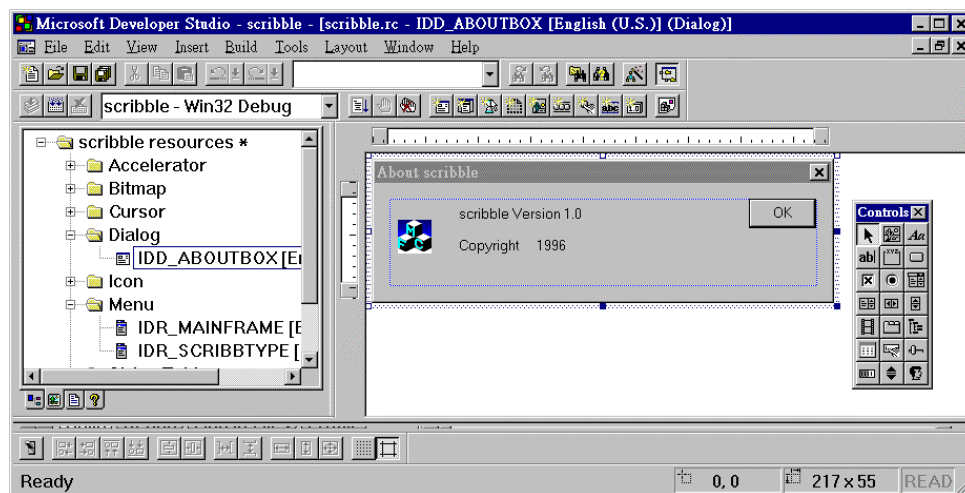
選按圖 4-1 ResourceView 中的 Accelerator，於是右側出現 Accelerator 編輯器。你可以直接在某一個項目上修改內容。



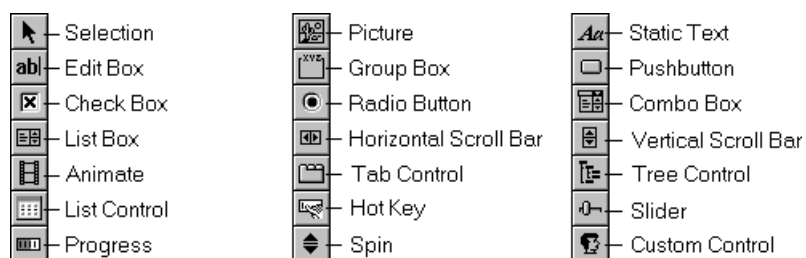
## 對話盒（Dialog）編輯器

任何一個由 AppWizard 產生出來的骨幹程式，都有一個很簡單樸素的 "About" 對話盒：

選按  ResourceView 中的 IDD\_ABOUTBOX，右側出現 Dialog 編輯器並將 About 對話盒載入。



圖右方有一個工具箱，內有許多控制元件（control）：



你可以在編輯器中任意改變對話盒及控制元件的大小和位置，也可以任意拖拉工具箱內的元件放入對話盒中。這些動作最後組成 RC 檔中的對話盒面板（Dialog template），也就是對話盒外貌的文字描述，像這樣：

```

IDD_ABOUTBOX DIALOG DISCARDABLE 0, 0, 217, 55
CAPTION "About Scribble"
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
FONT 8, "MS Sans Serif"
BEGIN
    ICON IDR_MAINFRAME, IDC_STATIC, 11, 17, 20, 20
    LTEXT "Scribble Version 1.0", IDC_STATIC, 40, 10, 119, 8, SS_NOPREFIX
    LTEXT "Copyright \251 1996", IDC_STATIC, 40, 25, 119, 8
    DEFPUSHBUTTON "OK", IDOK, 178, 7, 32, 14, WS_GROUP
END

```

## Console 程式的專案管理

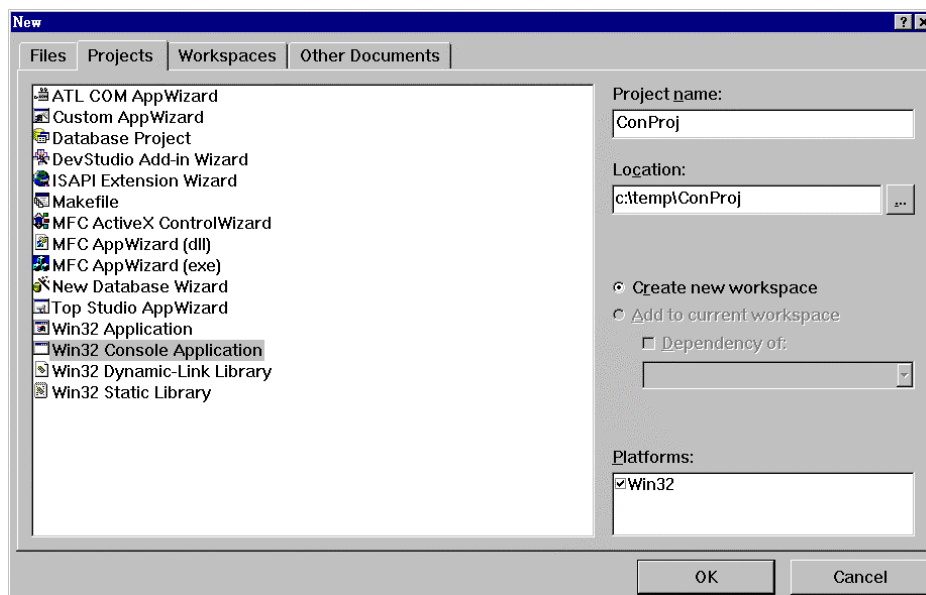
MFC AppWizard 會自動幫我們做出一個骨幹程式的所有必須檔案，建立起一個專案。但如果你想寫一個「血統單純」的純粹 C++ 程式呢？第 1 章曾經介紹過所謂的 console 程式。第 3 章的所有範例程式也都是 console 程式。

架構單純的程式，如果檔案只有一兩個，直接使用命令列就可以了：

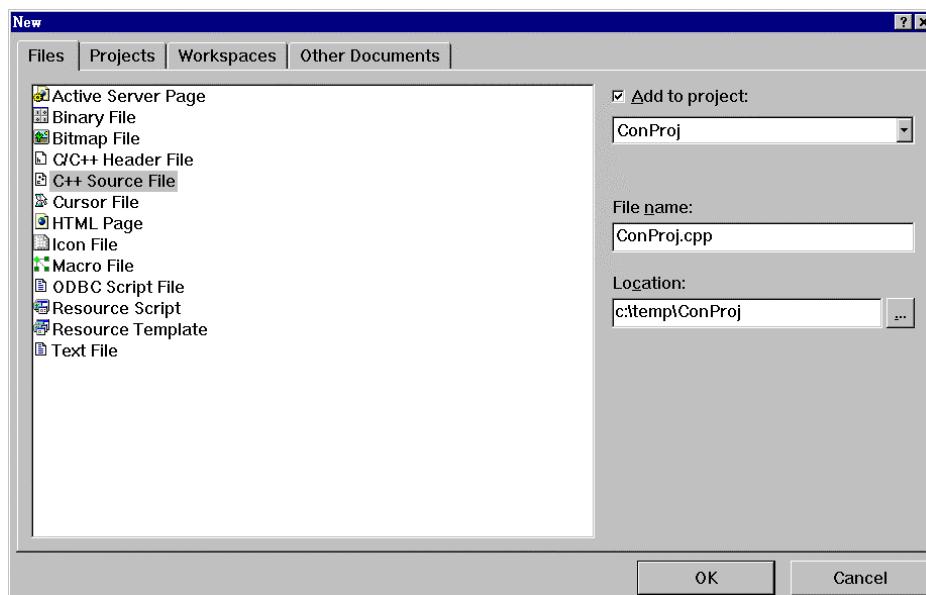
```
CL xxx.CPP <Enter>
```

如果組織架構比較複雜一點，檔案有好幾個，可以尋求專案管理員的協助。在 Visual C++ 整合環境中建立一個 console 程式專案的步驟如下：

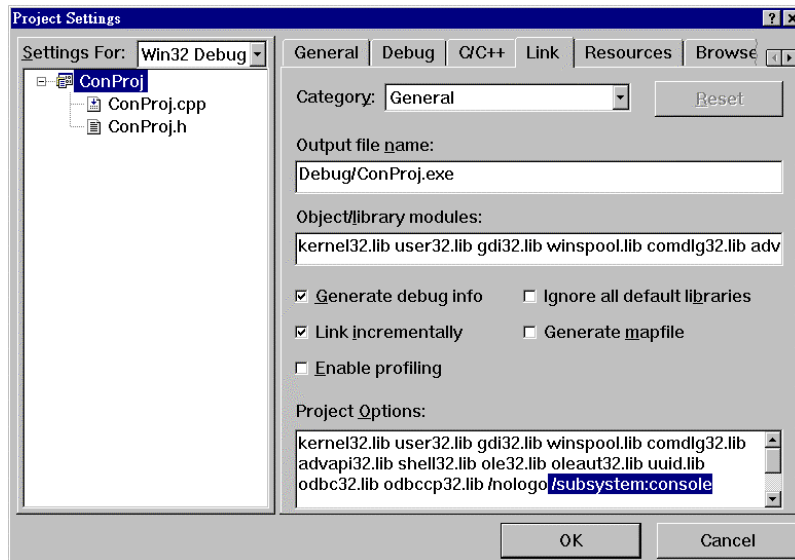
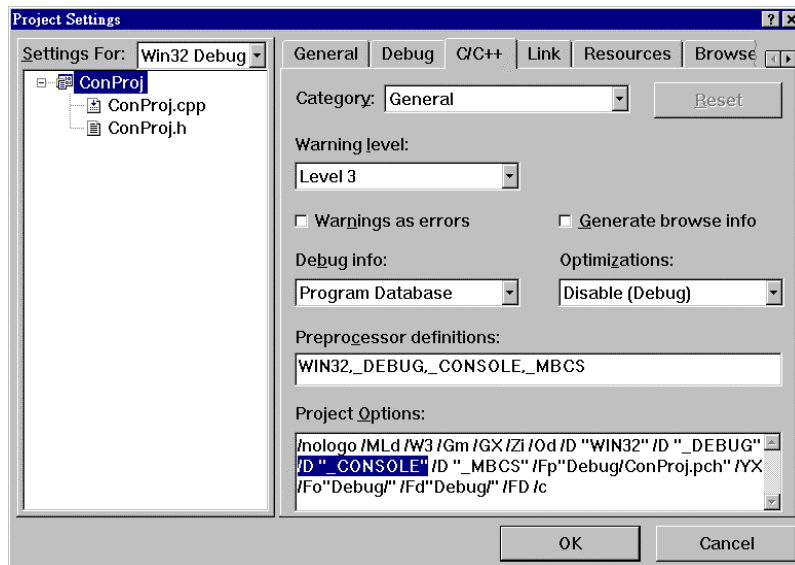
1. 選按整合環境的【File/New】，然後選擇【Projects】附頁，選按 "Win32 Console Application"，並填寫畫面右端的專案名稱和位置：



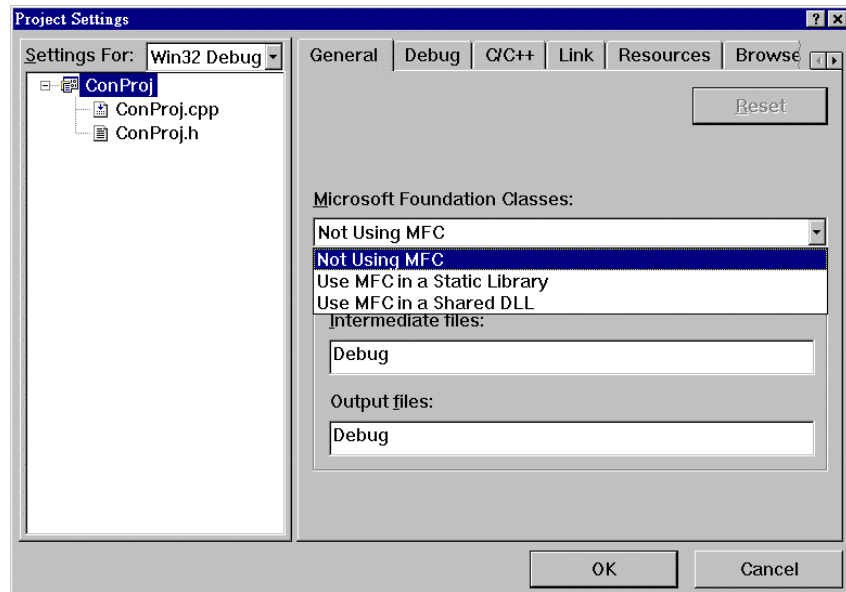
2. 按下【OK】鈕，回到整合環境主畫面，你可以選按【File/New】並選擇【Files】附頁，然後選按 "C/C++ Header File" 或 "C++ Source File" 以開啓檔案並撰寫程式碼。開啓的檔案會自動加入此專案中。



3. 你可以選按整合環境的【Project/Setting】選單項目，從中獲得並修改整個專案的環境設定。我曾經在第 1 章提過，console 程式必須在編譯時指定 `/D_CONSOLE` 常數，並在連結時指定 `subsystem:console`，這在以下兩個畫面中都可以看到（那是專案管理員自動為我們設定好的）：



第 1 章討論 console 程式時，我曾經說過，程式使用 MFC 與否，關係到 C runtime library 的單緒版或多緒版。是的，這項設定放在【General】附頁之中：



你在這裡所做的設定，會自動影響專案所聯結的 C runtime library 的版本。