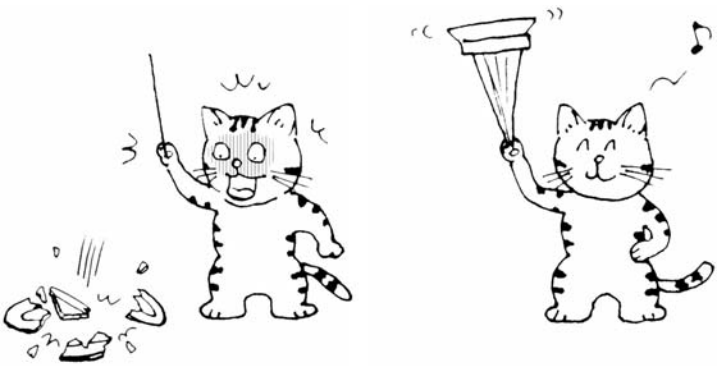
1.2 Windows 程式的起跑點

那麼，讀者知道「轉盤子」嗎？（喂！明明是程式書籍，怎麼忽然扯到轉盤子啊！？）轉盤子一開始很難，直到可以轉兩、三圈為止，需要一大段艱辛的練習。聽說大部分學習的人就是在學習到「轉兩、三圈」的技術時受挫放棄了。但是，其實有個比較簡單的學習方法。先用厚紙板在盤子下面作出一個比較高的腳(?)，這樣只要練習兩三分鐘，大概就可以成功轉起盤子了。接下來就慢慢一點一點把厚紙板減低一點，慢慢練習就可以了。

使用 C 語言寫 Windows 程式也一樣。一開始的門檻很困難，有很多人就是在這個階段受到挫折的。但是本書要帶領讀者以最簡單的方式跨過這道門檻，接下來的世界，就會越來越有趣了。

那麼，我們就來使用 C 語言寫出最簡單的 Windows 程式吧。這個程式什麼都不做，只要開啟一個視窗就好。照說一般學習新的語言要寫的第一個程式，老規矩是顯示出一個「Hello World」，不過，我們也不管它！



要用把普通的盤子拿來轉滿難的，但是…

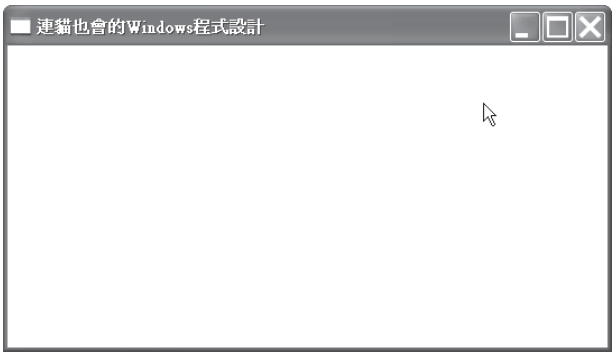


圖 1-1　什麼事都不做的程式

首先，請觀察圖 1-1 。雖然只是一個簡單的視窗，但仔細看，已經具備很多功能了。首先，拖曳視窗的標題列可以移動視窗。另外，拖曳視窗的外框，可以調整視窗的大小。視窗並具備了「最小化按鈕」、「最大化按鈕」與「關閉按鈕」。另外，按下左上角的應用程式圖示按鈕，會顯示出系統選單。

明明是什麼事都不做的程式，但卻已經擁有那麼多的功能了。如果這些事情全部要自己做，會是多恐怖的程式開發作業啊。不過，這倒是一點都不用擔心。因為「車到山前必有路」，請看下面的程式碼：

```
// sample01.cpp

#include <windows.h>

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);
ATOM InitApp(HINSTANCE);
BOOL InitInstance(HINSTANCE, int);

char szClassName[] = "sample01";    //視窗類型
```

```
int WINAPI WinMain(HINSTANCE hCurInst, HINSTANCE hPrevInst,
                   LPSTR lpsCmdLine, int nCmdShow)
{
   MSG msg;
   BOOL bRet;

   if (!InitApp(hCurInst))
        return FALSE;
   if (!InitInstance(hCurInst, nCmdShow))
        return FALSE;
   while ((bRet = GetMessage(&msg, NULL, 0, 0)) != 0) {
        if (bRet == -1) {
             break;
        } else {
             TranslateMessage(&msg);
             DispatchMessage(&msg);
        }
   }
   return (int)msg.wParam;
}

// 登錄視窗類型

ATOM InitApp(HINSTANCE hInst)
{
   WNDCLASSEX wc;
   wc.cbSize = sizeof(WNDCLASSEX);
   wc.style = CS_HREDRAW | CS_VREDRAW;
   wc.lpfnWndProc = WndProc;   // 視窗函式名
   wc.cbClsExtra = 0;
   wc.cbWndExtra = 0;
   wc.hInstance = hInst;  // 實體
   wc.hIcon = (HICON)LoadImage(NULL,
        MAKEINTRESOURCE(IDI_APPLICATION),
        IMAGE_ICON,
        0,
        0,
```

Windows程式的進入點

```
        LR_DEFAULTSIZE | LR_SHARED);
    wc.hCursor = (HCURSOR)LoadImage(NULL,
        MAKEINTRESOURCE(IDC_ARROW),
        IMAGE_CURSOR,
        0,
        0,
        LR_DEFAULTSIZE | LR_SHARED);
    wc.hbrBackground = (HBRUSH)GetStockObject(WHITE_BRUSH);
    wc.lpszMenuName = NULL;          // 功能表名
    wc.lpszClassName = (LPCSTR)szClassName;
    wc.hIconSm = (HICON)LoadImage(NULL,
        MAKEINTRESOURCE(IDI_APPLICATION),
        IMAGE_ICON,
        0,
        0,
        LR_DEFAULTSIZE | LR_SHARED);

    return (RegisterClassEx(&wc));
}

// 產生視窗

BOOL InitInstance(HINSTANCE hInst, int nCmdShow)
{
    HWND hWnd;

    hWnd = CreateWindow(szClassName,
              // 標題列會顯示出這個名稱
              " 連貓也會的 Windows 程式設計 ",
              WS_OVERLAPPEDWINDOW, // 視窗種類
              CW_USEDEFAULT,     // X 座標
              CW_USEDEFAULT,     // Y 座標
              CW_USEDEFAULT,     // 寬度
              CW_USEDEFAULT,     // 高度
              NULL, // 母視窗代碼，當這個視窗就是母視窗時設定為 NULL
              NULL, // 功能表代碼，要使用類型的功能表時設定為 NULL
              hInst, // 實體代碼
              NULL);
```

```
    if (!hWnd)
         return FALSE;
    ShowWindow(hWnd, nCmdShow);
    UpdateWindow(hWnd);
    return TRUE;
}

//視窗函式

LRESULT CALLBACK WndProc(HWND hWnd, UINT msg, WPARAM wp, LPARAM lp)
{
    switch (msg) {
         case WM_DESTROY:
              PostQuitMessage(0);
              break;
         default:
              return (DefWindowProc(hWnd, msg, wp, lp));
    }
    return 0;
}
```

回呼函式

C語言沒有「行」的觀念，不過就算我們硬用「行」作為單位，這個具有（?）許多功能的程式大概也能在100行左右寫完。（看到這個數字覺得算多還算少，就是每個人自己的問題了……）

讀者不需要急著了解這整篇程式做的是什麼，也不需要把內容背下來。這個程式就是今後的起跑點，也就是在這個什麼都不做的程式中，加上自己想要的功能後，就可以寫出自己想要的程式。在這裡首先請您敲出這篇程式碼，並build程式。

如果一切順利，請為自己鼓掌一下吧（??）。這個程式就是基本雛形，以後就是複製這些程式碼來使用。每次要寫程式前都要敲這100行，實在是太痛苦了一點。所以我們就複製這個雛型，貼上到新的程式檔中。接下來就只要加上想要的功能、動作就可以了。（這就像是轉盤子的初期秘訣。）

那麼，接下來我們就開始一點一點閱讀這段程式碼吧。

```
int WINAPI WinMain(HINSTANCE hCurInst, HINSTANCE hPrevInst,
                   LPSTR lpsCmdLine, int nCmdShow)
```

任何程式都應有其進入點（entry point）。若讀者在 MS-DOS 時代有使用 C 語言開發程式，應該會馬上聯想到 main 函式吧。在 Windows 程式中，則相當於 WinMain 函式。

仔細看這段程式碼，除了 WinMain 函式以外，其他還有 InitApp 函式、InitInstance 函式、WndProc 函式的出現。然而，程式裡卻找不到呼叫 WndProc 函式的地方。這是為什麼呢？事實上，這個函式是由系統所呼叫的。我們將這種函式稱為**回呼函式(callback function)**。並加上 **CALLBACK** 修飾子。當函式加上了 CALLBACK 修飾子，您可以想像成「這不是普通的函式喔！」就可以了。

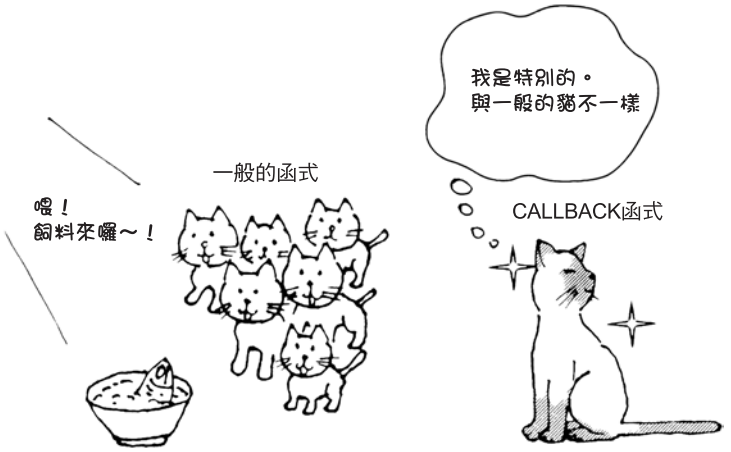
另外，在 InitApp 函式內，不知道您有沒有注意到這一行：

```
wc.lpfnWndProc = WndProc;
```

讀者應該看得出來，這裡像是在將 WndProc 函式建立了什麼關聯。

事實上 InitApp 、InitInstance 、WndProc 這些函式的名稱可以隨自己取。只有 WinMain 是固定的名稱。但因為筆者嫌麻煩，所以總是都固定用這些名稱來寫程式。這樣做，也比較容易避免錯誤。

當您可以成功 build 這段程式，開啟一個什麼事都不做的視窗，您就從這一章畢業了。當您無法順利編譯時，請詳細閱讀「附錄 A　Visual C++.Net 與 6.0 的專案與資源的建立方法」。



重 要

程式要以 WinMain 函式開始

程式中有一個供系統呼叫的回呼函式