



電腦、網際網路與 全球資訊網簡介



C++程式設計藝術(第七版)(國際版)





本章綱要

- 1.1 簡介
- 1.2 電腦硬體與軟體
- 1.3 電腦的架構
- 1.4 個人式、分散式及用戶端/伺服器的計算環境
- 1.5 網際網路與全球資訊網
- 1.6 Web 2.0
- 1.7 機器語言、組合語言和高階語言
- 1.8 C/C++的沿革
- 1.9 C++標準函式庫
- 1.10 Java 的沿革
- 1.11 Fortran、COBOL、Pascal 以及 Ada 程式語言
- 1.12 BASIC、Visual Basic、Visual C++、C# 以及 .NET
- 1.13 關鍵的軟體工程趨勢：物件技術
- 1.14 典型的 C++ 開發環境





- 1.15 C++ 與本書注意事項
- 1.16 實際體驗 C++ 應用程式
- 1.17 軟體技術
- 1.18 C++ 的未來：開放原始碼 **Boost** 函式庫、TR1 以及 C++0x
- 1.19 軟體工程案例研討：物件技術與統一塑模語言 (UML) 介紹
- 1.20 總結
- 1.21 資源網站





1.1 簡介

- ▶ 你可以從這個網址下載範例程式：
www.deitel.com/books/cpphttp7/





1.1 簡介

- ▶ 電腦 (硬體) 由軟體所控制。
- ▶ 軟體就是人撰寫的指令，可讓電腦執行動作 (action) 並做判斷 (decision)。





電腦

- ▶ 請想一個詞彙描述電腦的特性





- ▶ 快速
- ▶ 正確
- ▶ 不會累
- ▶ 多才多藝
- ▶ 記憶力超強
- ▶ ...





1.2 電腦硬體與軟體

- ▶ **電腦 (computer)** 是一種裝置，能夠以人類數百萬倍甚至數十億倍的速度，執行計算和邏輯判斷。
- ▶ 現今最快的**超級電腦 (supercomputers)** 一秒可執行數千兆 (quadrillions) 個指令！
- ▶ 電腦受一組稱作**電腦程式 (computer program)** 的指令控制來處理**資料 (data)**。
- ▶ 這些電腦程式指揮電腦去執行一連串有次序的動作，這些動作都是由**電腦程式設計者 (computer programmers)** 預先指定的。
- ▶ 電腦是由各種裝置所組成的，我們稱它們為硬體 (hardware)。
- ▶ 在電腦上執行的程式則稱為軟體 (software)。





1.3 電腦的架構

- ▶ 每部電腦實際上都可分成六個邏輯單元 (logical units) 或區域 (section) :
 - 輸入單元. 這個「接收資訊」的區域，會從輸入裝置 (input devices) 取得資訊，並將此資訊放在其它裝置上以進行處理。
 - 輸出單元. 這個用來「輸出資訊」的區域會接受經電腦處理過的資訊，並且將資訊送到不同的輸出裝置 (output devices)，讓這些資訊能夠在電腦之外使用。
 - 記憶體單元. 這是電腦中一個存取快速、但容量相對較低的「倉庫」區域。◦ 記憶體中的資訊是「揮發性的」 (volatile)。記憶體單元又稱為記憶體 (memory) 或主記憶體 (primary memory)。





1.3 電腦的架構

- ▶ **算術和邏輯單元 (arithmetic and logic unit, ALU)**。這是「生產製造」的區域，負責執行計算，也包含電腦的判斷機制。在今日的系統中，**ALU**通常是**CPU**的一部分。
- ▶ **中央處理單元 (CPU)**。它是電腦「執行管理」的區域，負責協調監督其它區域的作業。
 - 需要將資訊讀入記憶單元時，**CPU**會通知輸入單元。
 - 需要將記憶單元的資訊進行計算處理時，**CPU**就會通知**ALU**加以處理。
 - 需要將記憶單元的資訊傳送到某個輸出裝置時，**CPU**就會通知輸出單元。
 - 今天許多電腦都具備多個**CPU**，因此能同時執行許多操作，這種電腦稱為**多重處理器系統 (multiprocessors)**。**多核心處理器 (multi-core processor)** 將多個處理器放在一個積體電路晶片 中。





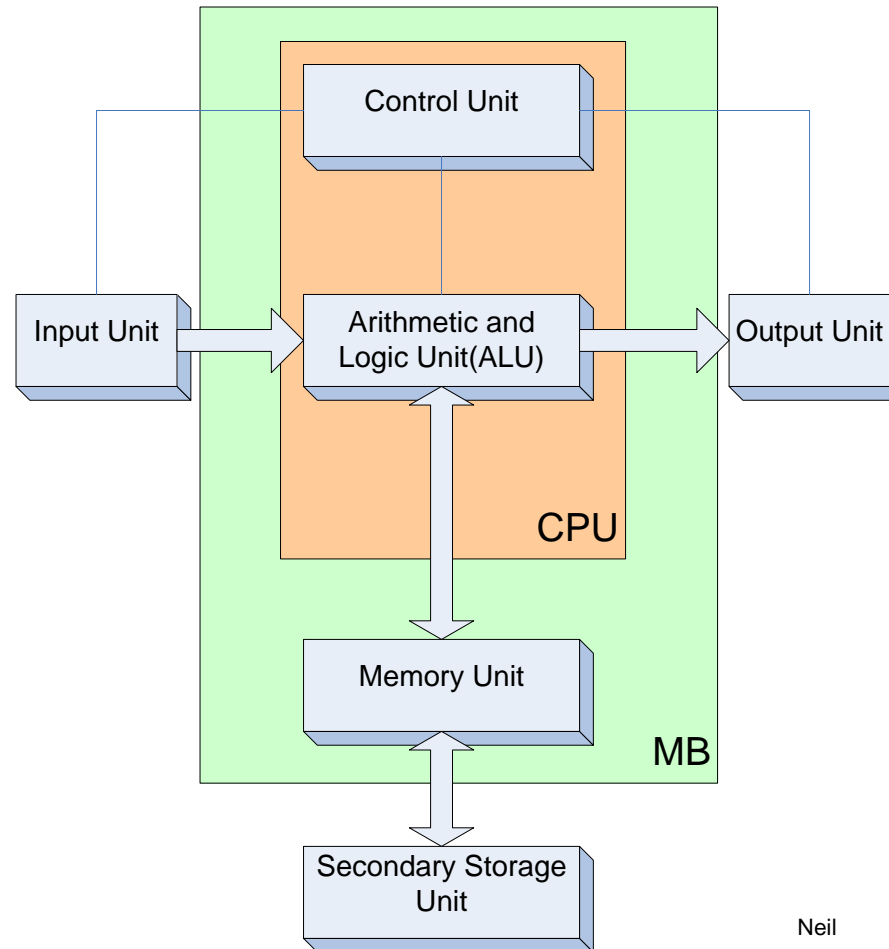
1.3 電腦的架構

- **輔助儲存單元 (secondary storage unit)**：這是電腦長期、高容量的「倉庫」區域。儲存在輔助儲存裝置中的資料為「永續性」(**persistent**)的。





Computer Organization

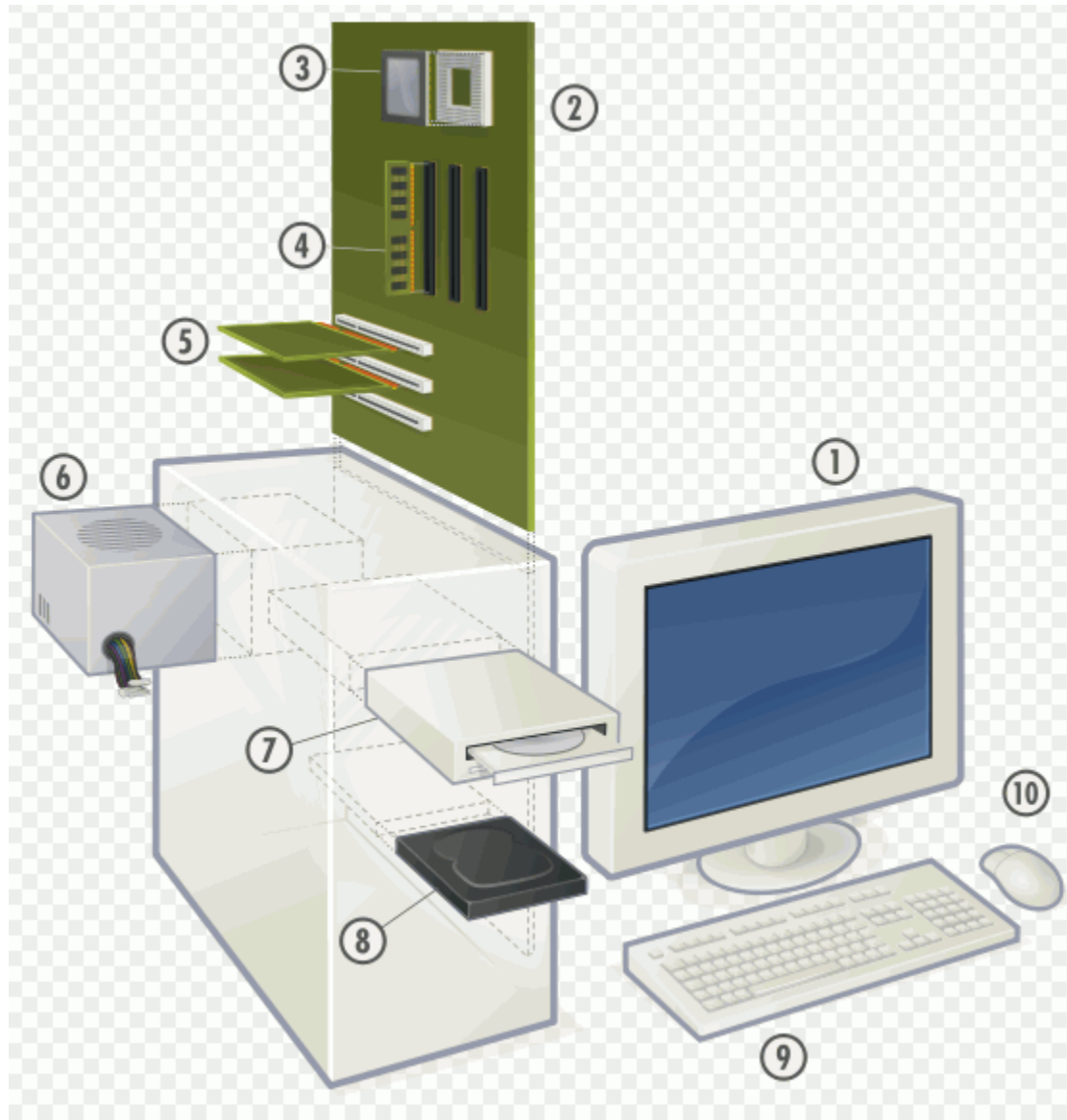


Neil





Computer Organization



1.7 機器語言、組合語言和高階語言

- ▶ 程式設計者可用各種程式語言撰寫指令，有些程式語言可由電腦直接讀取，有些則須經過中間**轉譯 (translation)** 步驟
- ▶ 語言可分成三大類：
 - 機器語言
 - 組合語言 (assembly language)
 - 高階語言



機器語言 (machine language)

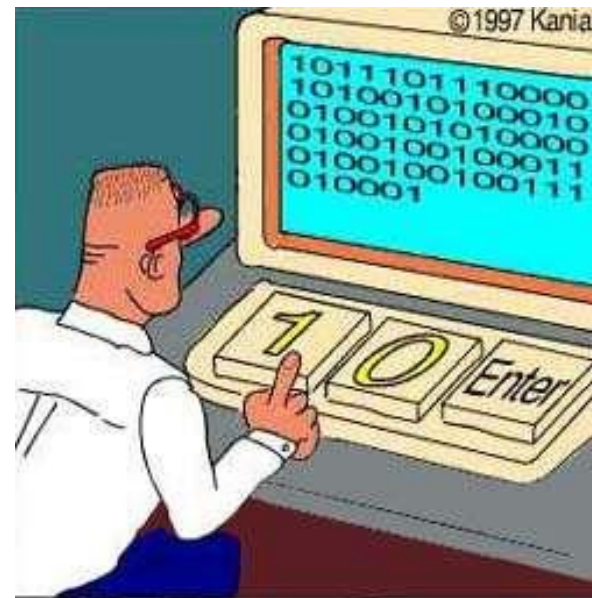
- ▶ 任何電腦都只能直接讀懂它自己的**機器語言 (machine language)**。
 - 機器語言是電腦的「母語」，此語言由電腦的硬體設計所定義。
- ▶ **機器語言與機器有關 (machine dependent)**
 - 用機器語言寫程式很慢，程式又臭又長又容易出錯。

- Example:

+1300042774

+1400593419

+1200274027



Real programmers code in binary.





組合語言 (assembly languages)

- ▶ 縮寫字構成指令。
- ▶ 組合語言指令 → 「組譯器」(assembler) → 機器語言。
- ▶ 即便是最簡單的工作，仍要許多指令。

- ▶ Example:

- LOAD BASEPAY
- ADD OVERPAY
- STORE GROSSPAY





高階語言 (high-level languages)

- ▶ 為了加快程式設計的過程，就發展了高階語言 (high-level languages)
- ▶ 只需單一敘述 (statement) 就能完成不少工作。
 - ▶ Example:
 - `grossPay = basePay + overTimePay`
- ▶ 高階語言指令 → 「編譯器」(compilers) → 機器語言。
- ▶ 「直譯器」(Interpreter) 程式可直接執行高階語言（不需要經過編譯程序），但速度比經過編譯的程式慢得多。





1.9 C++ 標準函式庫

- ▶ C++ 程式是由**類別 (class)** 和**函式 (function)** 組成。
- ▶ 大部分C++程式設計者都會用**C++標準函式庫 (C++ Standard Library)** 中現成豐富的類別和函式。
- ▶ 在C++「世界」裡我們得學習兩個部分。
 - 首先要學C++語言本身
 - 接著要學如何使用C++標準函式庫中的類別和函式。
- ▶ 軟體公司則會提供許多特殊功能的類別函式庫。





軟體工程的觀點 1.1

請使用「建構區塊」建立程式。避免重寫軟體。盡量利用現成的區塊。這叫做**軟體再利用 (software reuse)**，是物件導向程式設計的中心思想。





軟體工程的觀點 1.2

以 C++ 寫程式時，通常會使用以下的建構區塊：C++ 標準函式庫的類別與函式、您與同僚建立的類別與函式，以及各種知名第三方函式庫中的類別與函式。





增進效能的小技巧 1.1

請多利用 C++ 標準函式庫的函式與類別，而不要自己寫，因為這些程式都寫得很棒，執行效率很高，如此可增進程式效能，亦可縮短程式開發時間。





可攜性的小技巧 1.2

請多用 C++ 標準函式庫的函式與類別，盡量不要自己寫。因為所有 C++ 實作都包含了這套函式庫，如此可提升程式可攜性。





軟體工程的觀點 1.3

網際網路上可以找到許多可重複使用的軟體元件的擴增類別庫。這些類別庫有許多是免費的。





1.14 典型的 C++ 開發環境

- ▶ C++系統通常由三個部分組成：程式開發環境、程式語言及C++標準函式庫。
- ▶ C++程式通常得經過六個階段：編輯 (edit)、前置處理 (preprocess)、編譯 (compile)、連結 (link)、載入 (load) 和執行 (execute)。



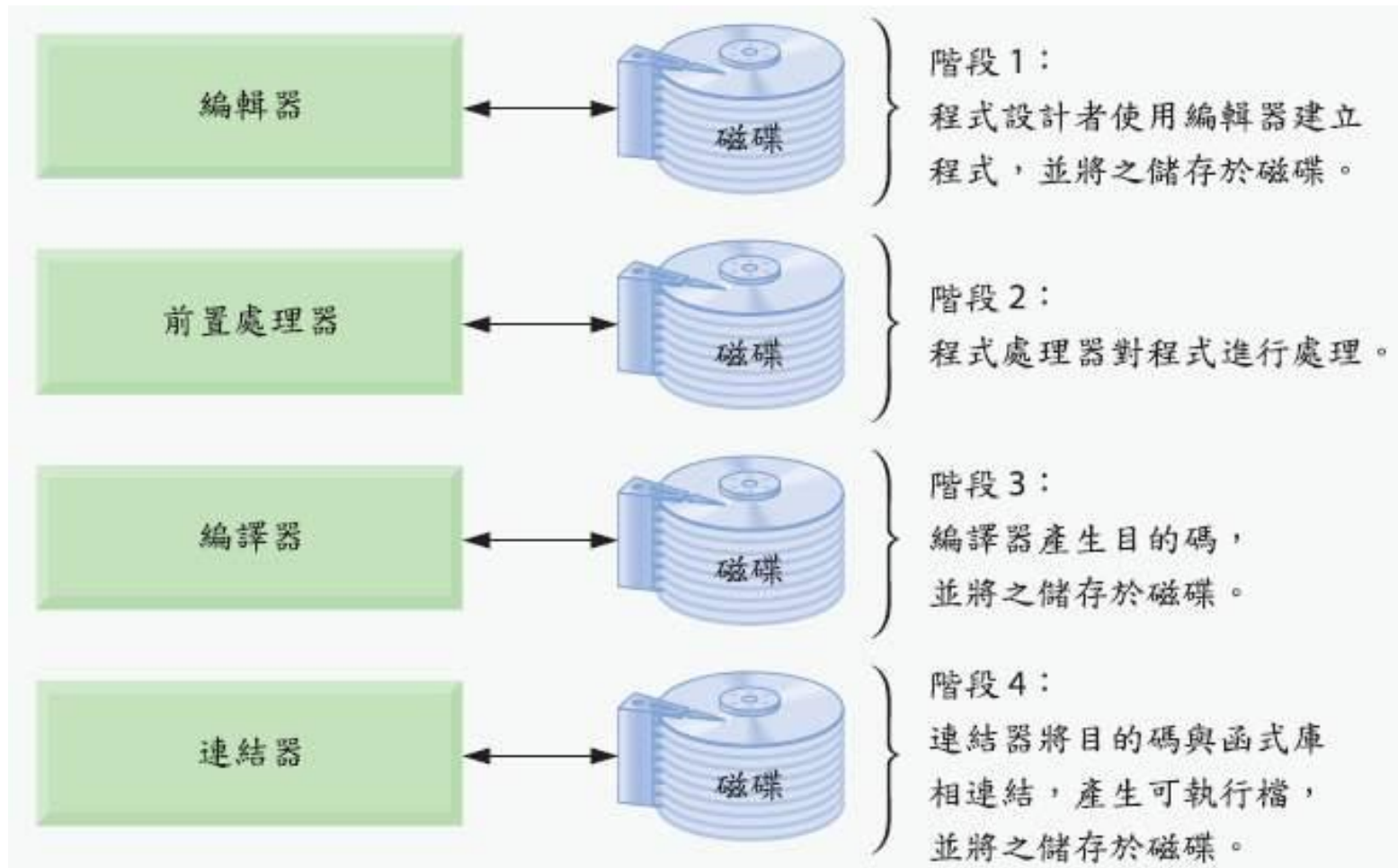


圖 1.1 典型的 C++ 開發環境



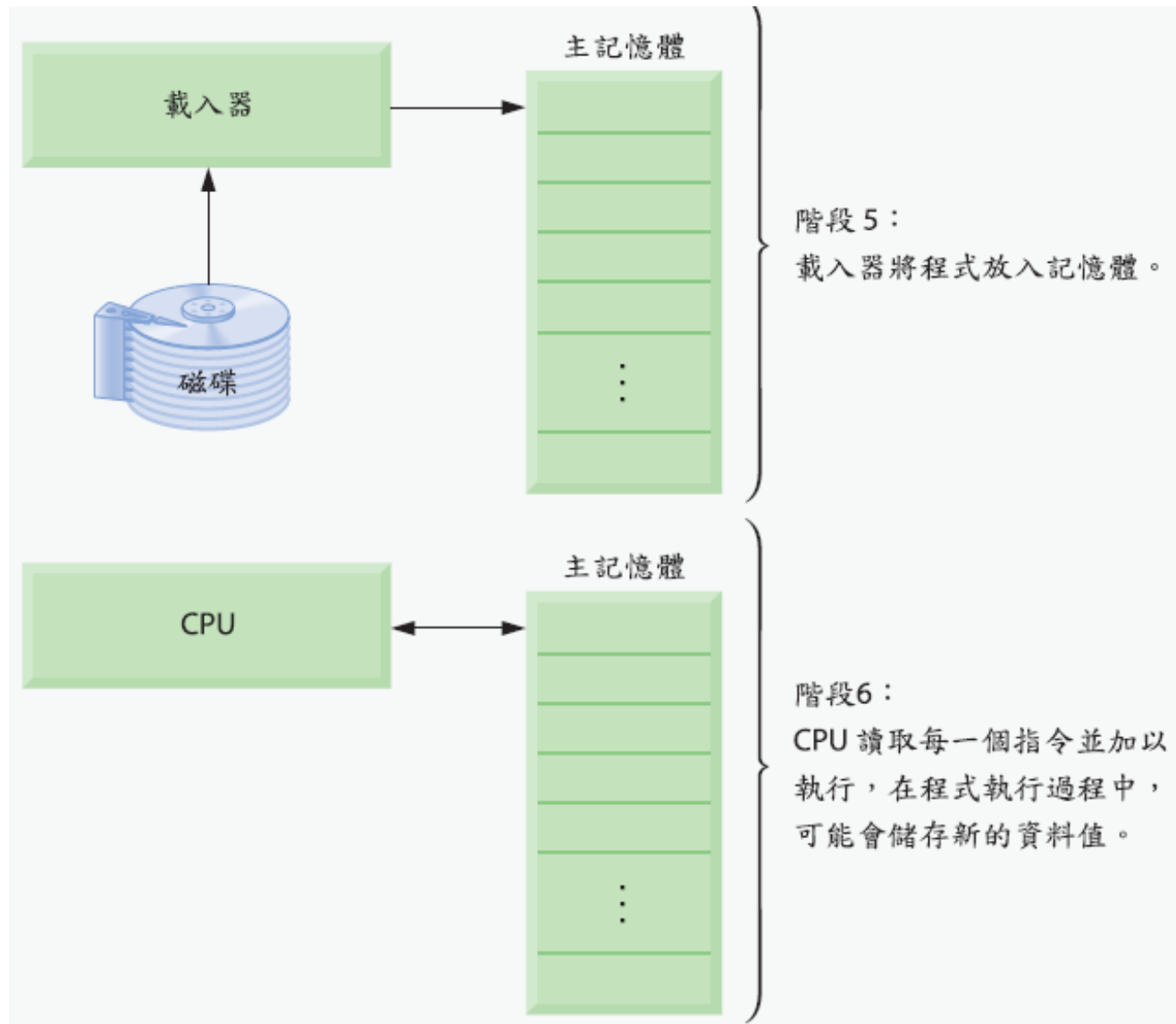
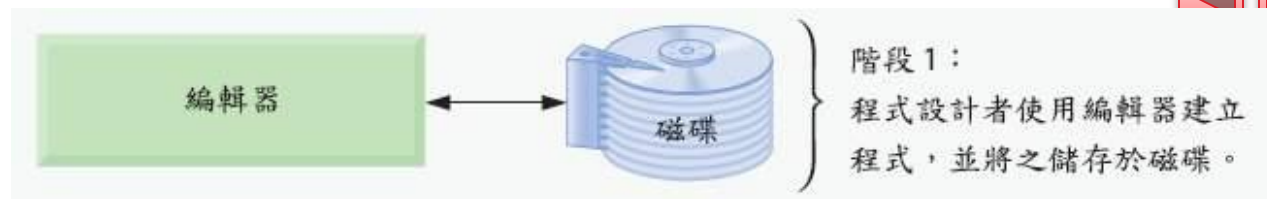


圖 1.1

典型的 C++ 開發環境



1. 編輯

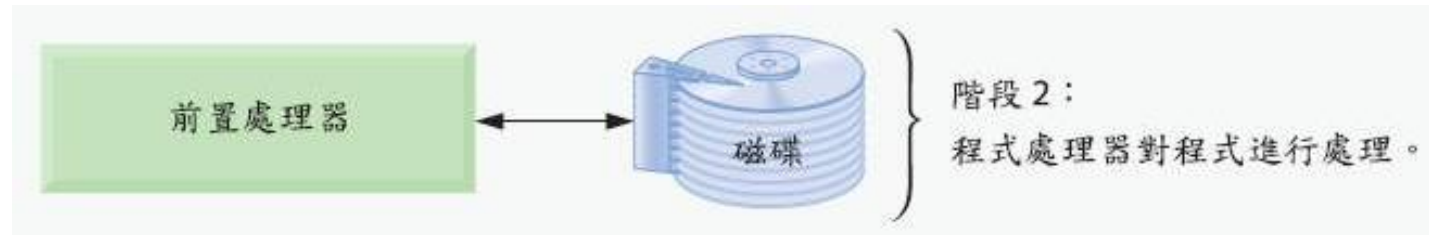


- ▶ 第一階段是以**編輯器**（**editor program**，簡稱**editor**）編輯檔案。
 - 您可用編輯器鍵入C++程式 [通常稱作**原始碼 (source code)**]、進行必要修正，並將程式儲存。
 - **C++原始碼檔案的副檔名通常是 .cpp、.cxx、.cc 或 .C** (注意，**C**是大寫)，表示該檔案含有C++原始碼。
 - **UNIX系統中最常用的兩種編輯器就是 vi 和 emacs。**
 - **C++軟體套件，如Microsoft Visual C++** (msdn.microsoft.com/vstudio/express/visualc/default.aspx) 將編輯器整合在程式開發環境中。



2. 前置處理

- ▶ C++系統中有個**前置處理器 (preprocessor)**，在編譯器進行轉譯前會自動執行此程式。
- ▶ C++前置處理器會按照一種叫做「**前置處理指令 (preprocessor directive)**」的指令進行動作，該指令表示編譯前要對程式執行某些操作。
- ▶ 這些操作通常會把其它要編譯的文字檔含括進來，並進行各種文字取代動作。
- ▶ 我們會在前幾章討論最常用的前置處理指令；附錄E「前置處理器」會詳細介紹前置處理程式的功能。
- ▶ 在第三階段，編譯器會將C++程式碼轉譯成機器碼（也稱為目的碼）。



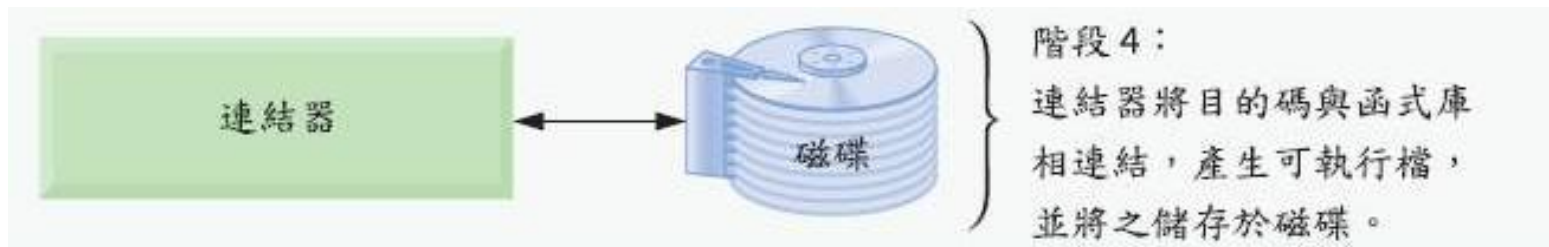
3. 編譯

- ▶ 在第三階段，編譯器會將C++程式碼轉譯成機器碼（也稱為目的碼）。



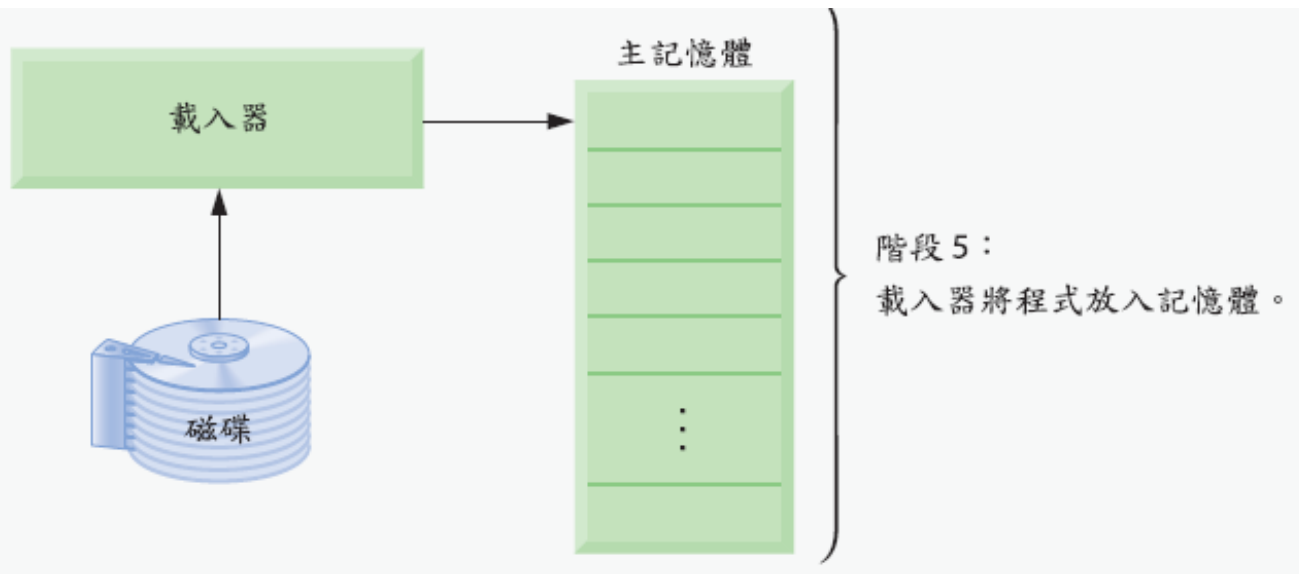
4. 連結 (linking)

- ▶ 第四階段叫做**連結 (linking)**。
- ▶ C++編譯器所產生的目的碼常包含參照別處定義的函式與資料所形成的「洞」。例如指向標準函式庫函式的參照。
- ▶ **連結器 (linker)** 會將目的碼與這些尚未加入的函式連接起來，以產生可執行的程式。



5. 載入 (loading)

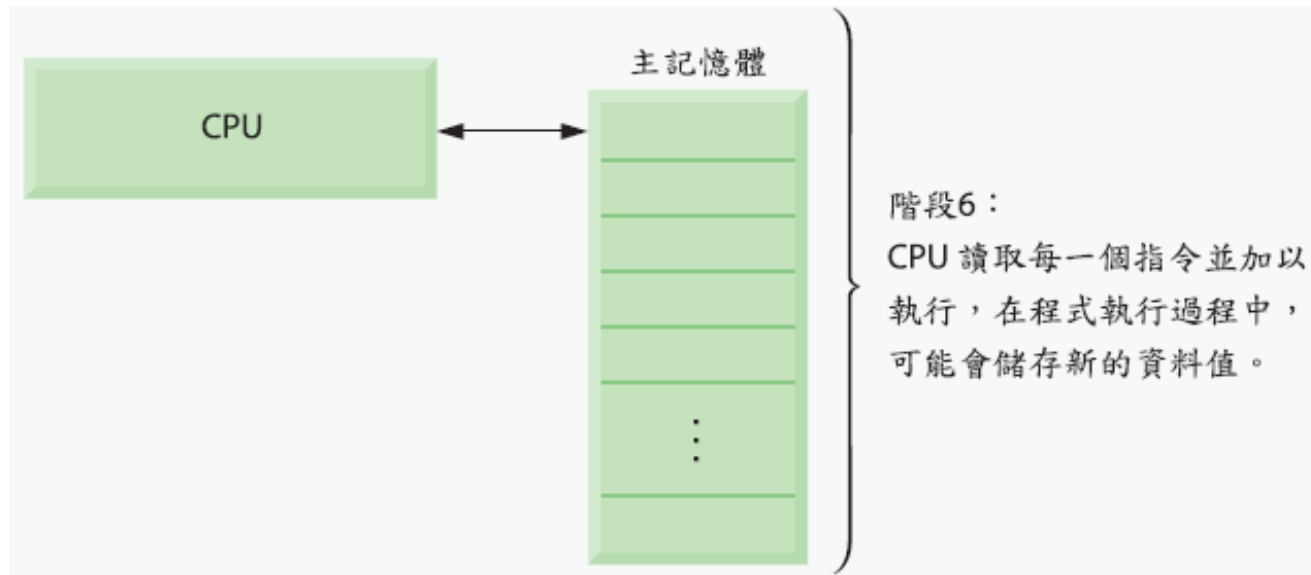
- ▶ 第五階段叫做**載入 (loading)**。
- ▶ 程式執行前，必須先被放入記憶體中。
- ▶ **載入器 (loader)** 負責這項工作，它能把可執行的影像檔從磁碟搬到記憶體中。
- ▶ 程式所用到的共享函式庫中其它元件，也須一併載入。





6. 執行 (execute)

- ▶ 最後，電腦會在CPU的控制下，以每次執行一個指令的方式開始**執行（executes）**程式。





1.14 典型的 C++ 開發環境

- ▶ 程式在第一次測試時不一定會成功。
- ▶ 前面幾個階段都可能因各種錯誤而失敗，這些錯誤本書都會討論。
- ▶ 若發生這樣的情形，就要回到編輯階段做些必要修正，再繼續後面的幾個階段，看看改對了沒。





1.14 典型的 C++ 開發環境

- ▶ C++ 中大部分程式都會輸入和/或輸出資料。
- ▶ 某些C++函式會從cin (唸作「see-in」，標準輸入串流；**standard input stream**) 取得輸入值，通常是從鍵盤進行輸入，但cin也可連結到別的裝置。
- ▶ 資料通常會輸出到cout (唸作「see-out」，標準輸出串流；**standard output stream**)，一般是電腦螢幕，但cout也能連結到別的裝置。
- ▶ 當我們說「程式印出結果」時，通常是指在螢幕上顯示結果。
 - 資料也可輸出到其它裝置，如磁碟和印表機。
- ▶ 電腦也有標準錯誤串流 (**standard error stream**)，稱為**cerr**。cerr串流用來顯示錯誤訊息。





常見的程式設計錯誤 1.1

當程式執行發生除零錯誤時，這種錯誤稱為執行時期錯誤 (**run-time error** 或 **execution-time error**)。致命的執行時期錯誤 (**fatal runtime errors**) 會讓程式無法完成工作，而必須立即終止。非致命的執行時期錯誤 (**Nonfatal runtime errors**) 可以允許程式繼續執行完畢，但通常會產生錯誤的結果。[請注意：在某些系統上，除零並不屬於致命的錯誤。請參閱您的系統文件。]





Software Engineering Observation 1.2

- ▶ Your computer and compiler are good teachers. 編譯器是你的好老師
- ▶ If you are not sure how a feature of C works, write a sample program with that feature, compile and run the program and see what happens.





1.16 實際體驗 C++ 應用程式

- ▶ 這一節我們要執行第一個C++應用程式，並與之互動。
- ▶ 這是個猜數字遊戲，從1到1000中選個數字，並提示您猜個數字。
- ▶ 若猜對，遊戲就結束。
- ▶ 若猜錯，應用程式會說您猜的數字比答案大或小。
- ▶ 猜的次數不限。
- ▶ [請注意：為了進行測試，我們將第6章「函式與遞迴簡介」中要您做的習題做個修改，建立了本程式。原本每次執行該程式時，會隨機選個不同的數字讓您猜。但我們改了程式，讓每次答案都一樣（雖然可能會因編譯器而異）。這樣當我們示範如何與你的第一個C++應用程式互動時，你就可以用跟我們一樣的數字來猜，並看到同樣的結果。]

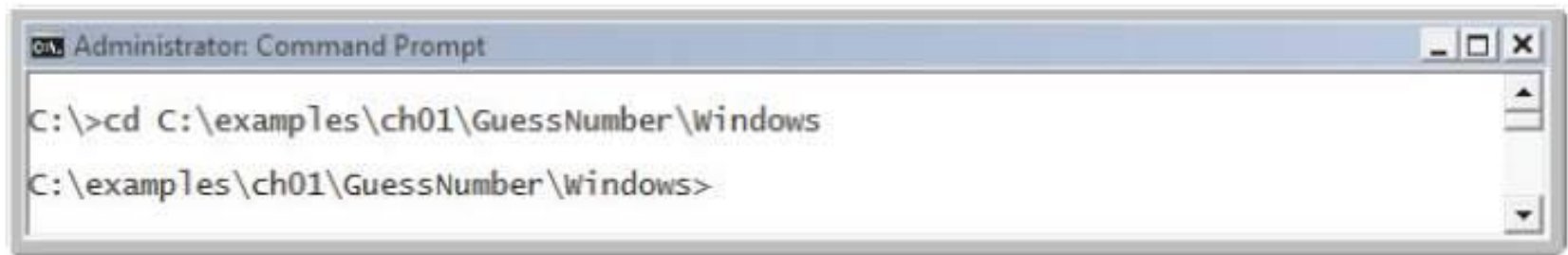




1.16 實際體驗 C++ 應用程式

- ▶ 在Windows命令提示下測試程式。





```
Administrator: Command Prompt
C:\>cd C:\examples\ch01\GuessNumber\Windows
C:\examples\ch01\GuessNumber\Windows>
```

圖 1.2 開啓命令提示 (command prompt) 視窗並切換目錄





```
Administrator: Command Prompt - GuessNumber
C:\examples\ch01\GuessNumber\Windows>GuessNumber
I have a number between 1 and 1000.
Can you guess my number?
Please type your first guess.
?
```

圖 1.3 執行 **GuessNumber** 應用程式





```
Administrator: Command Prompt - GuessNumber
C:\examples\ch01\GuessNumber\Windows>GuessNumber
I have a number between 1 and 1000.
Can you guess my number?
Please type your first guess.
? 500
Too high. Try again.
?
```

圖 1.4 輸入第一個數字，猜第一次





```
Administrator: Command Prompt - GuessNumber
C:\examples\ch01\GuessNumber\Windows>GuessNumber
I have a number between 1 and 1000.
Can you guess my number?
Please type your first guess.
? 500
Too high. Try again.
? 250
Too high. Try again.
?
```

圖 1.5 猜第二次，並看回應





```
Administrator: Command Prompt - GuessNumber
Too high. Try again.
? 125
Too low. Try again.
? 187
Too high. Try again.
? 156
Too high. Try again.
? 140
Too high. Try again.
? 132
Too high. Try again.
? 128
Too low. Try again.
? 130
Too low. Try again.
? 131

Excellent! You guessed the number!
Would you like to play again (y or n)?
```

圖 1.6 再猜一次，猜到正確的答案



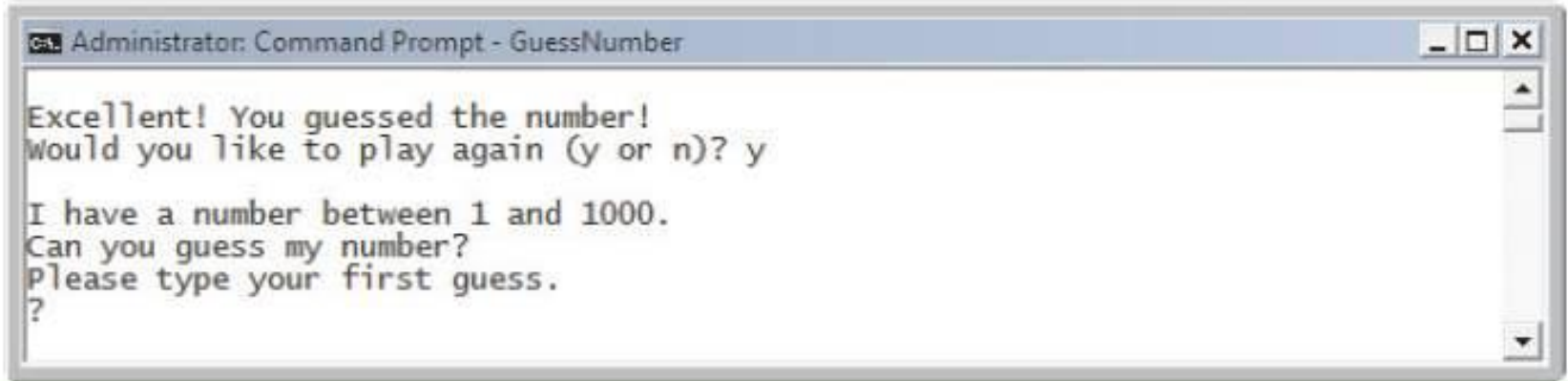
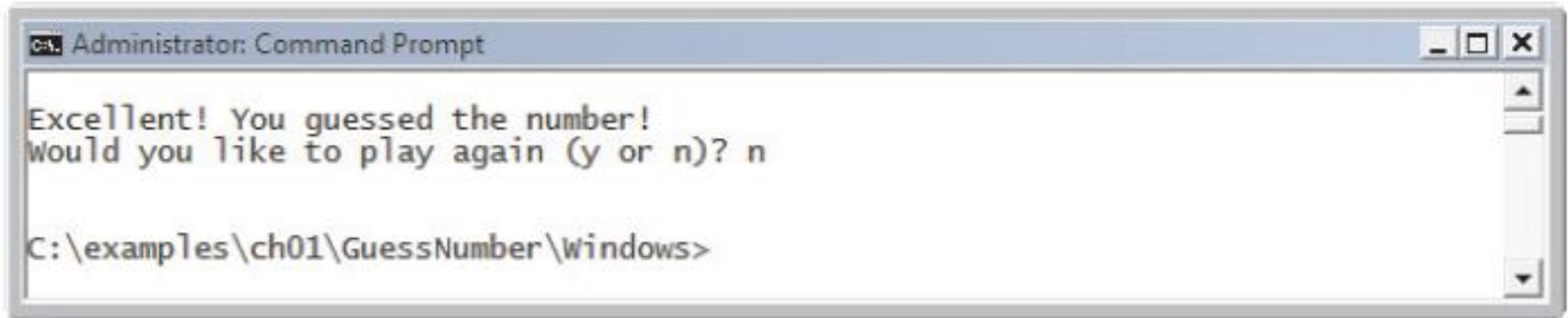


图 1.7 再玩一次





```
Administrator: Command Prompt

Excellent! You guessed the number!
Would you like to play again (y or n)? n

C:\examples\ch01\GuessNumber\Windows>
```

圖 1.8 結束遊戲





1.18 C++的未來：開放原始碼 Boost 函式庫、TR1 以及 C++0x

- ▶ 新標準的主要目標是要讓C++更容易學習，改善函式庫建構能力，並增加與C程式語言的相容性。
- ▶ 我們將在第23章中介紹下一代的C++：**Boost C++函式庫**、**Technical Report 1 (TR1)** 以及 **C++0x**。
- ▶ **Boost C++函式庫 (Boost C++ Libraries)** 是由C++社群成員所建立的免費開放原始碼函式庫。
- ▶ **Boost**已經超過80個函式庫，而且還在穩定增加。





▶ C++程式通常得經過六個階段，請寫出原文：

- 1) 連結 (`_n_`)
- 2) 載入 (`_o_`)
- 3) 前置處理 (`__e__c__`)
- 4) 編譯 (`____l_`)
- 5) 執行 (`_x_____`)
- 6) 編輯 (`__i_`)





▶ C++程式通常得經過六個階段，請依序列出：

- 1) 連結 (link)
- 2) 載入 (load)
- 3) 前置處理 (preprocess)
- 4) 編譯 (compile)
- 5) 執行 (execute)
- 6) 編輯 (edit)





▶ C++程式通常得經過六個階段，請依序列出：

- 1) 連結 (link)
- 2) 載入 (load)
- 3) 前置處理 (preprocess)
- 4) 編譯 (compile)
- 5) 執行 (execute)。
- 6) 編輯 (edit)

Ans: 6, 3, 4, 1, 2, 5

