第一章

計算機概論與C語言簡介

目標

在本章中，我們將回顧一些計算機概論的基本知識，藉由這些基本知識，進而學習電腦的程式設計。除此之外，我們也將針對C語言的編譯器及執行環境做一些說明，加強讀者在往後章節中練習範例時所需要的基本知識。

大綱

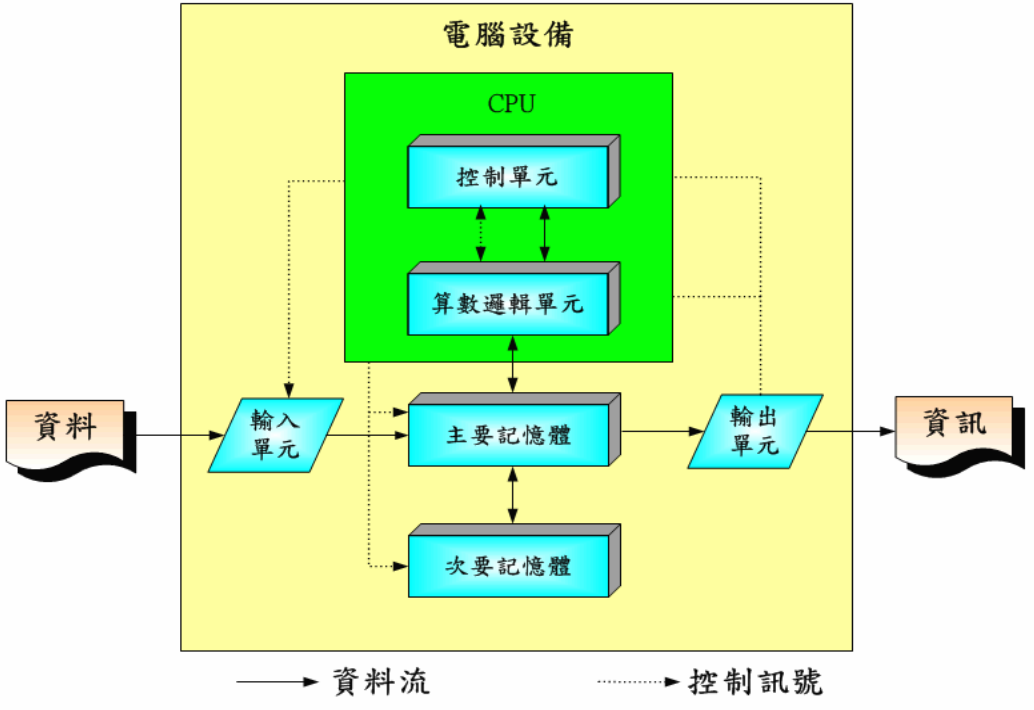
- 1.1電腦硬體
- 1.2電腦軟體
- 1.3程式語言
- 1.4C語言簡介

大綱(續)

- 1.5程式開發流程與編譯器
- 1.6本章回顧及下章預覽

1.1 電腦硬體

- 電腦硬體就是您可以看到的電腦設備（拆開外殼），不過若光是這樣形容電腦硬體未免過於簡單。實際上，若從功能面加以區分，可將電腦硬體分爲5大單元（如下圖），5個單元分別負責不同的工作。



1.1.1 算數邏輯單元

- 算數邏輯單元（簡稱ALU）是執行程式中各類運算的實體單位。這些運算則可以分爲兩大類：算數運算與邏輯運算。算數運算包含加、減、乘、除等等的數值運算，而邏輯運算則包含AND、OR、NOT、移位等位元／位元組的邏輯運算。

1.1.2　控制單元

- 控制單元（簡稱CU）的功能爲控制流程及協調輸入、輸出、記憶、算數邏輯等4大單元的運作。控制單元中包含(1)記錄指令運作順序的微程式（microprogram）、(2)取得下一指令的邏輯電路 (3)驅動元件的解碼器（decoder）及(4)眾多選擇器（multiplexer）。

1.1.3 記憶體單元

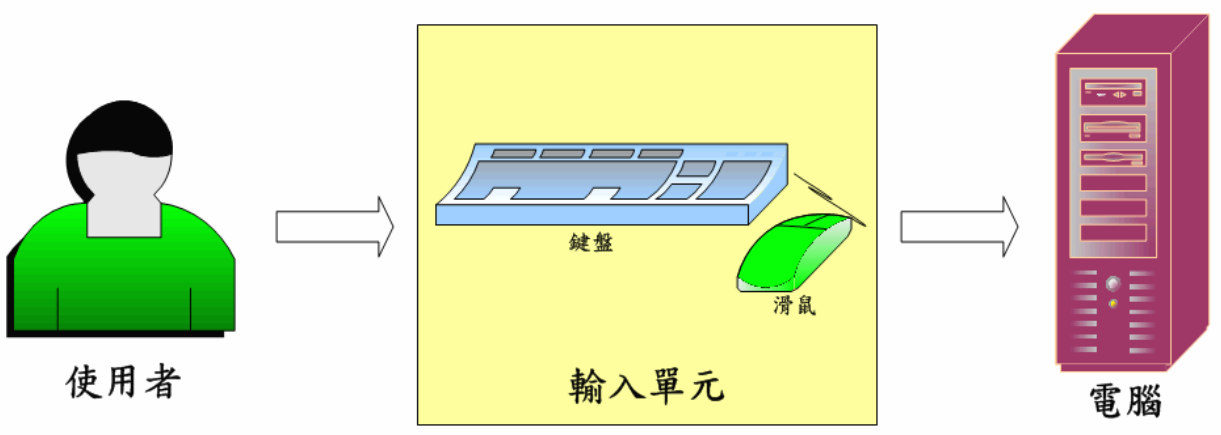
- 記憶體單元分爲主記憶體(main memory)與輔助記憶體(secondary memory)，主要功能是用來儲存資料
  - 主記憶體（又稱爲內部記憶體），目前以半導體元件制成，特性爲存取速度快、成本高。主記憶體依照存取特性又可以分爲隨機存取記憶體(Random Access Memory；簡稱RAM)及唯讀記憶體(Read Only Memory；簡稱ROM)。

1.1.3 記憶體單元

- RAM的成本較低，但是無法於電力消失時保存資料，故爲揮發性記憶體的一種。ROM成本較高，但卻可以在無電力的狀況下保存資料，傳統的ROM只能寫入資料一次，因此通常只會把啓動電腦所需要的小程式儲存在ROM裡面，例如BIOS就是使用ROM做爲記憶體。
- 輔助記憶體（又稱爲外部記憶體），目前以磁性物體或光學材料組成，例如：硬碟機、軟碟片、光碟片。輔助記憶體的存取速度相對於主記憶體慢了數十倍以上，但製作成本則比主記憶體低了數十倍以上，因此適合儲存大量的資料。

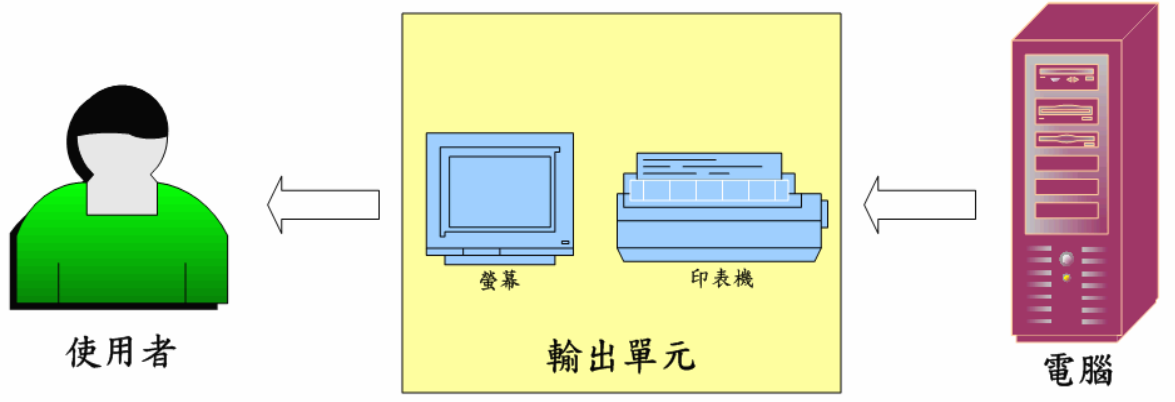
1.1.4　輸入單元

- 輸入單元是「具有輸入功能的週邊設備」，例如鍵盤、滑鼠、搖桿等等。使用者可以藉由這些輸入裝置與電腦取得溝通的管道。



1.1.5　輸出單元

- 輸出單元恰與輸入單元相反，所有「具有輸出功能的設備」皆屬輸出單元的元件，主要功能是將程式執行的結果（文字、聲音、影像）輸出或顯示。常見的輸出裝置有螢幕、印表機等等。某些週邊設備同時具有輸入與輸出的功能，例如：觸控式螢幕、會震動的搖桿等等。



1.2 電腦軟體

- 電腦軟體分爲資料與程式兩大類，事實上不論是哪一類，都是以0、1的二位元表示法儲存在電腦設備中（例如：儲存於硬碟機中）。
- 而程式又可以分爲系統程式（System Program）與應用程式（Application Program）。

  - 系統程式一般爲較接近硬體底層的低階程式，例如：作業系統(Operating System)、編譯程式(Compiler)、組譯程式(Assembler)，連結程式(Linker)等都屬於系統程式。
  - 應用程式則是架構在系統程式之上，依據某種特殊需求而開發出來的軟體，例如：Office、帳務系統、電腦遊戲等等。

1.3 程式語言

- 語言的用途是做爲人與人溝通的橋樑，例如：和美國人交談就要用英文溝通。同樣地，人若要和電腦溝通的話，就必須使用電腦『懂』的語言，這種語言稱爲程式語言（Programming Language）。而一般我們用來與人溝通的語言則稱爲自然語言（Natural Language）。

1.3 程式語言

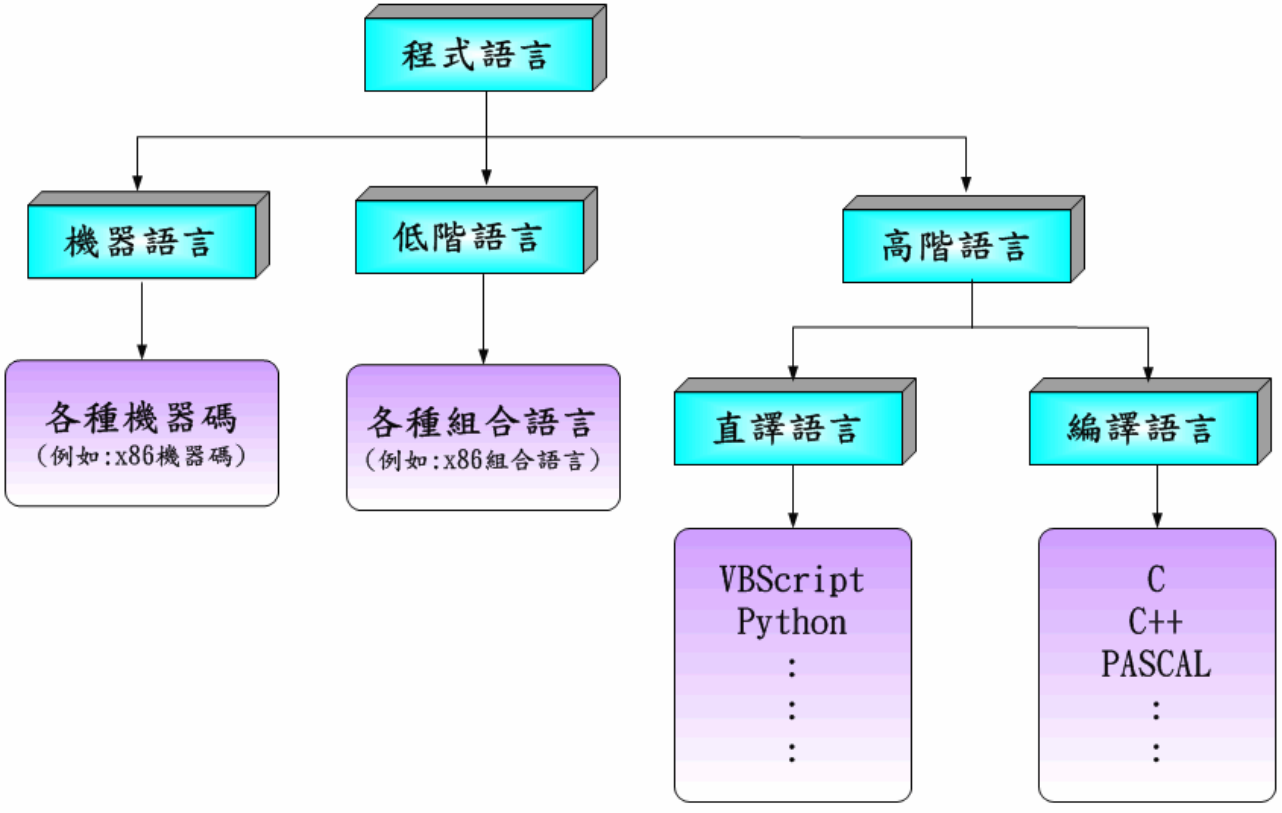


圖1-6 程式語言分類

1.3.1　機器語言

- 機器語言（Machine Language）是電腦硬體唯一看得懂的語言，換句話說，機器語言就是一連串的0、1二進位數字組合，因此又稱爲機器碼。一般人通常看不懂這些0、1所代表的特殊涵義，其實對於電腦而言，這些0、1 的組合數字，可能代表某種資料，也可能代表某個指令。由於大多數的人無法了解或記憶這一連串0、1數字所代表的涵義，因而發展了低階語言與高階語言。

1.3.2 低階語言

- 低階語言（Low-level Language）是一種接近於機器語言的表示方法，不過卻使用人類比較容易記憶的單字形式來對應一連串的0、1組合。最典型的低階語言就是組合語言（Assembly Language）。在組合語言中，使用運算子與運算元來表示一連串的0、1組合，而這些運算子則使用類似英文的縮寫以利人類的記憶與理解，例如：使用INC來代表Increment（累加指令）。

1.3.2 低階語言(續)

8051機器語言指令	8051組合語言指令	意義
00000100	INC	執行累加1
10000100	DIV	執行除法

8051組合語言指令與機器語言指令的對應

- 在上表中，很明顯的可以看出，組合語言的指令與機器語言的指令是一對一的對應關係，但是卻比機器語言容易記憶，除此之外，其他組合語言的設計都與機器語言的設計相同，我們可以從下表中更明確地看出兩者的相同與相異處：

1.3.2 低階語言(續)

- 組合語言與機器語言一對一的特性，使得組合語言可以完全掌控電腦的硬體結構，如此一來，在執行效率上自然也就完全交由程式設計師決定。不過，由於不同的CPU必須使用不同的組合語言並且必須對於該CPU的組織結構有充分認知，因此，這種低階語言仍舊無法被絕大多數的人所接受，因而發展了更接近於人類自然語言的高階語言。

1.3.3 高階語言

- 組合語言雖然比機器語言更接近於自然語言，但組合語言的程式設計師必須對於執行程式的處理器有更多的了解，並且每個處理器的組合語言並不相同，因此，一種比組合語言更接近自然語言且不因更換機器而改變語法的程式語言也被發展出來，此種語言就是高階語言。
- 高階語言（High-level Language）使用更接近人類思維的方式來設計程式，當程式設計完成之後，必須通過另外一些翻譯程式的翻譯後才能夠被電腦執行。高階語言的運算子通常具有比較強大的功能，因此，單一行的高階語言程式可能被翻譯成許多的機器碼以便完成複雜的工作。

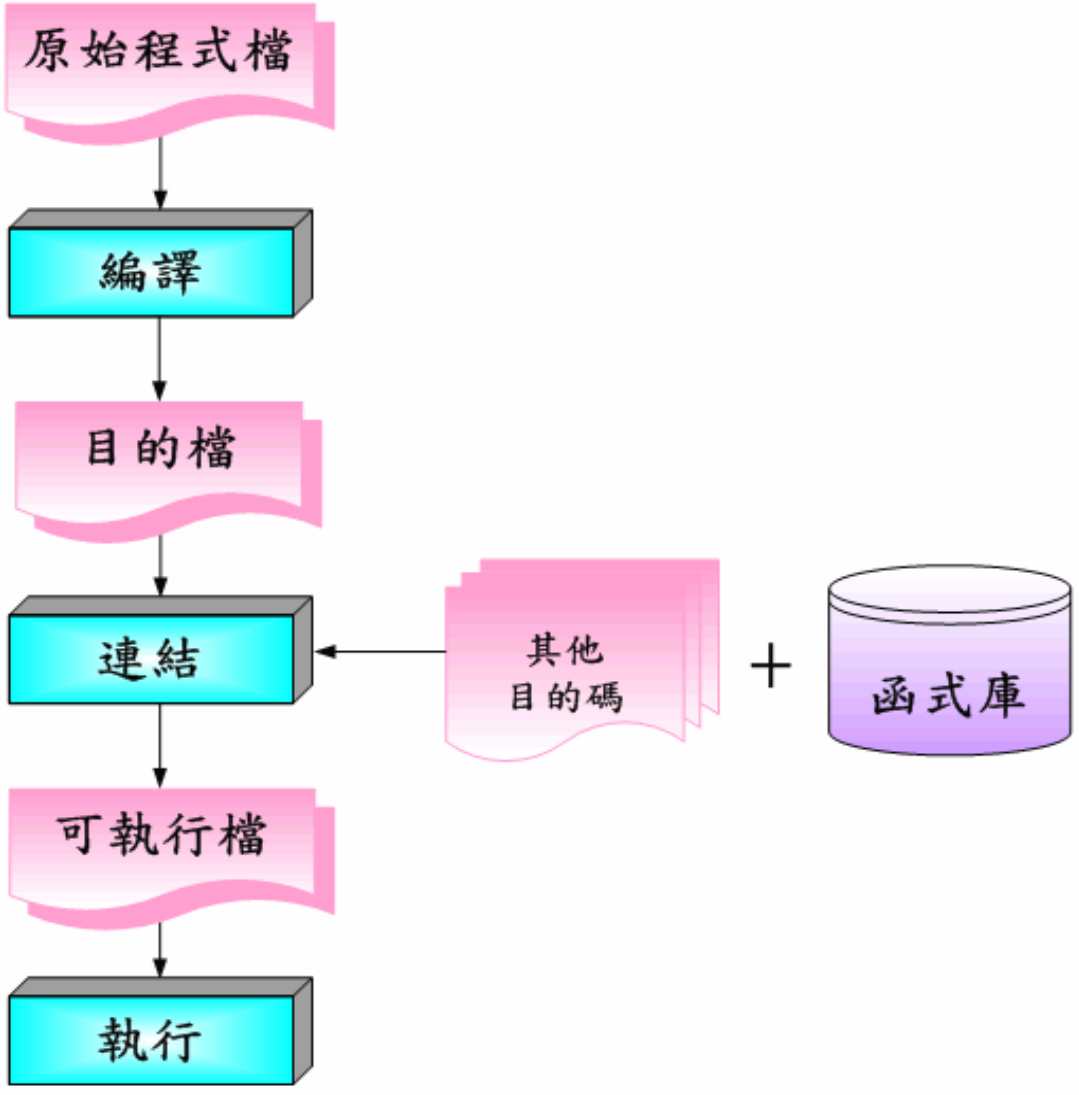
1.3.3 高階語言(續)

- 編譯式語言
  - 編譯器採用整批作業（Batch）方式來處理程式翻譯的工作，換句話說，當我們將程式設計完畢並交由編譯器翻譯之後，編譯器會將翻譯結果存成一個目的檔（object file），而這個目的檔可經由連結其他目的檔及程式庫之後形成可執行檔（execute file），並由電腦直接執行。常見的編譯式語言有C、C++（編譯器爲gcc、g++、VC++、BCB）、Pascal、Object Pascal（編譯器爲Delphi、Kylix）等等。

1.3.3 高階語言(續)

- 編譯式語言
  - 編譯式語言使用模組化技巧，也就是把某些具有特殊功能的片段程式獨立成一個個的函式，並將之集合成一個函式庫檔案，如此一來，就可以讓需要使用該功能的程式透過連結的方式加以結合，縮短撰寫程式的時間。例如：ANSI-C的math函式庫中就包含許多求三角函數的函式，所以我們並不需要自行撰寫求三角函數的詳細步驟，只要在需要求三角函數值時，引用Math函式庫即可。一個基本編譯式語言的程式處理流程如下圖。

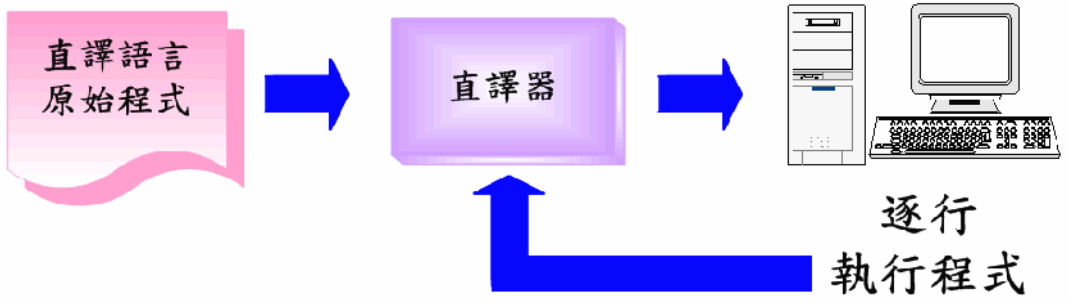
1.3.3 高階語言(續)



編譯式語言的程式處理流程

1.3.3 高階語言(續)

- 直譯式語言
  - 直譯器與編譯器處理程式的步驟不同，當我們使用直譯式語言撰寫程式完畢並使用直譯器來翻譯程式時，並不會產生目的檔或可執行檔。直譯器會將程式一行一行的讀入，逐行翻譯並送交由電腦執行，因此，每一次要執行編譯式語言程式時，都必須啓動直譯器來重新翻譯程式。



直譯式語言的程式處理流程

1.4 C語言簡介

- C語言與C++語言有著奧妙的關係，因此有人說，C++是C語言的延伸，也有人說，C++是一種全新的語言。這兩種說法都算正確，因爲C++是使用基本的C語言語法，但C++更重要的是一種全新的物件導向觀念。在本節中，我們將分別介紹這兩種語言的特色與歷史背景。

- C語言是在70年代，由貝爾實驗室的Ken Thompson與Dennis Ritchie共同發展的一套電腦語言，該版本稱爲K & R C。之後美國國家標準協會（ANSI）於1983年將C語言標準化，制訂出一套標準的C語言，稱爲ANSI-C（於1988年完成制定）。

1.4 C語言簡介(續)

- C語言的發展歷史有其特殊背景，C語言原本不叫做C語言，而是另一種程式語言B語言，Ken Thompson與Dennis Ritchie兩人爲了設計UNIX作業系統，因此將B語言加以延伸爲C語言，由於具有容易開發、效率高及可操控硬體等特性，因此同樣是由貝爾實驗室所發展的UNIX系統，系統核心的大部分程式碼都是使用C來撰寫，並且在UNIX作業系統中的公用程式及C語言的編譯器也是由C所撰寫的。

1.4 C語言簡介(續)

- 命令式型態程式語言（Imperative Paradigm Programming Language）
  - 命令式型態的程式語言是透過一連串指定敘述(assign statement)組合而成，指定敘述是表示式(expression)，將一些子運算式加以組合，只要依序執行這些敘述就可以得到結果，C語言是一種命令式型態的程式語言，其他諸如BASIC、Pascal、FORTRAN、COBOL等也都屬於此類程式語言。

```
x=x+1;
```

1.4 C語言簡介(續)

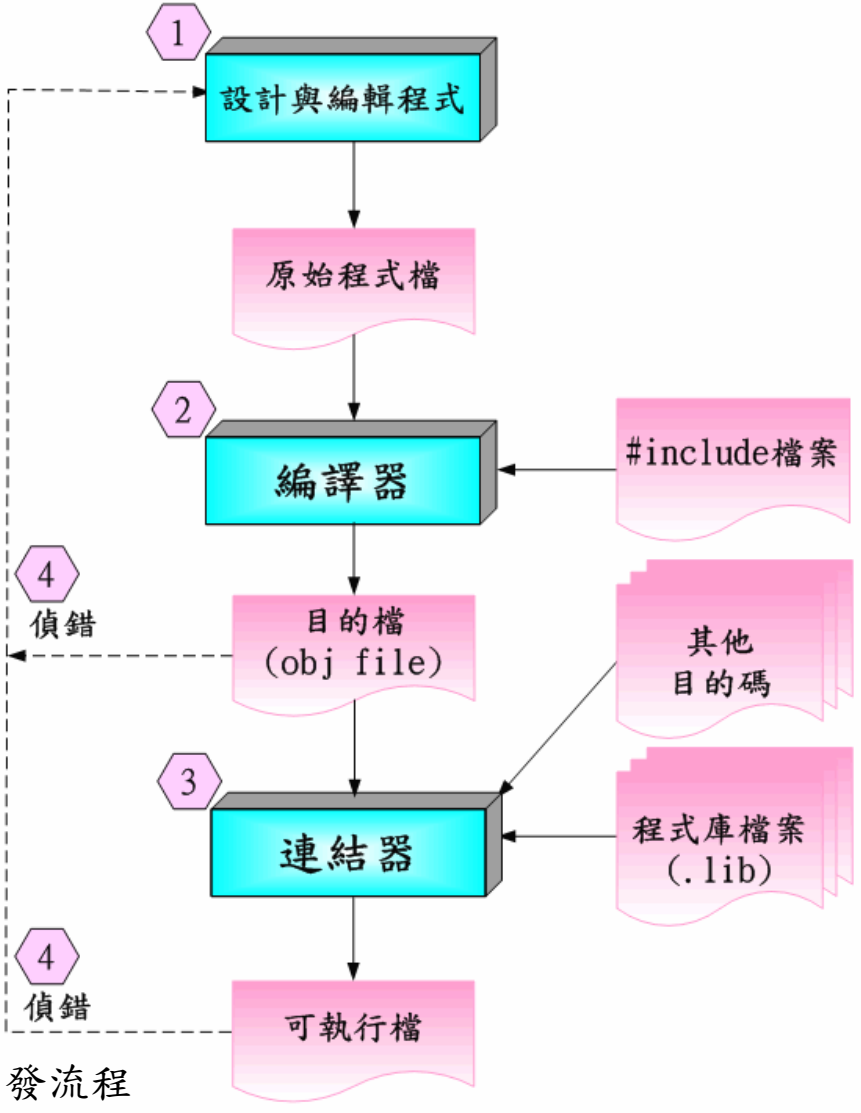
- 結構化程式語言(Structural Programming Language)
  - C語言是一種結構化程式語言，也就是採用結構化分析設計的方法，將問題由上向下，由大到小切割成許多子問題，分別尋得子問題的解答，最後再結合起來解決所要解決的問題。這種方法，也可以稱爲模組化設計。解決子問題的方法被分成一個個的模組，在C語言中則稱爲函式。在C的程式設計中，程式設計師必須設計並組合這些函式來解決問題。

1.5 程式開發流程與編譯器

- 開發C語言程式完成後，必須經過編譯才能夠被執行，在附錄中，我們列出了許多可以編譯C語言的編譯器，除此之外，在本節中，我們也將針對C語言編譯器的基本觀念加以說明，建立一些關於C語言編譯器的基本觀念。

1.5.1 C語言程式的開發流程

- 開發任何編譯式語言程式都有一定的流程（如右圖），開發C語言的應用程式也同樣可以透過下圖流程加以完成。



C語言程式開發流程

1.5.1 C語言程式的開發流程(續)

- Step1：
  - 編輯與設計程式，您可以使用任何的純文字編輯器來編輯C/C++程式（例如：Windows的筆記本、Linux的Vim或pico等等），然後存成副檔名爲『.c』與『.cpp』的原始程式檔，其中純C語言程式可存檔爲『.c』，而包含C++語法或函式庫的程式則存檔爲『.cpp』。
- Step2：
  - 使用編譯器來編譯原始程式（例如：GCC的gcc/g++），編譯結果將是一個目的檔案（object file）。
- Step3：
  - 使用連結器將目的程式及其他目的程式與程式庫共同連結成一個可執行檔，目前大多數的編譯器都具有連結器的功能。
- Step4：
  - 不論您是在編譯過程或連結過程中發生錯誤或甚至是程式本身的邏輯出現錯誤而無法達到程式需求時，都必須進入偵錯階段，重新檢查並修正原始程式中的錯誤。如此週而復始，直到程式完成您的需求爲止。

1.5.2　編譯器(Compiler)【補充】

- 傳統編譯器的功能是將原始程式轉換爲目的碼（如下圖爲C語言編譯器的工作），目的碼中則包含了機器指令、資料值及相關位址。就程式設計師而言，設計高階語言程式比較不需要考慮程式執行的機器平台，只要在不同的硬體結構，使用不同的編譯器產生適合於該平台的目的碼即可。

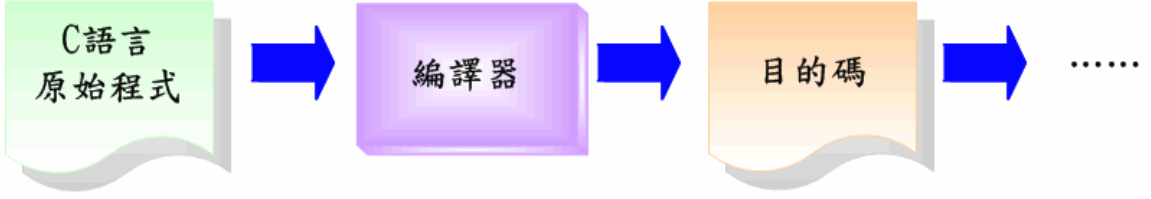
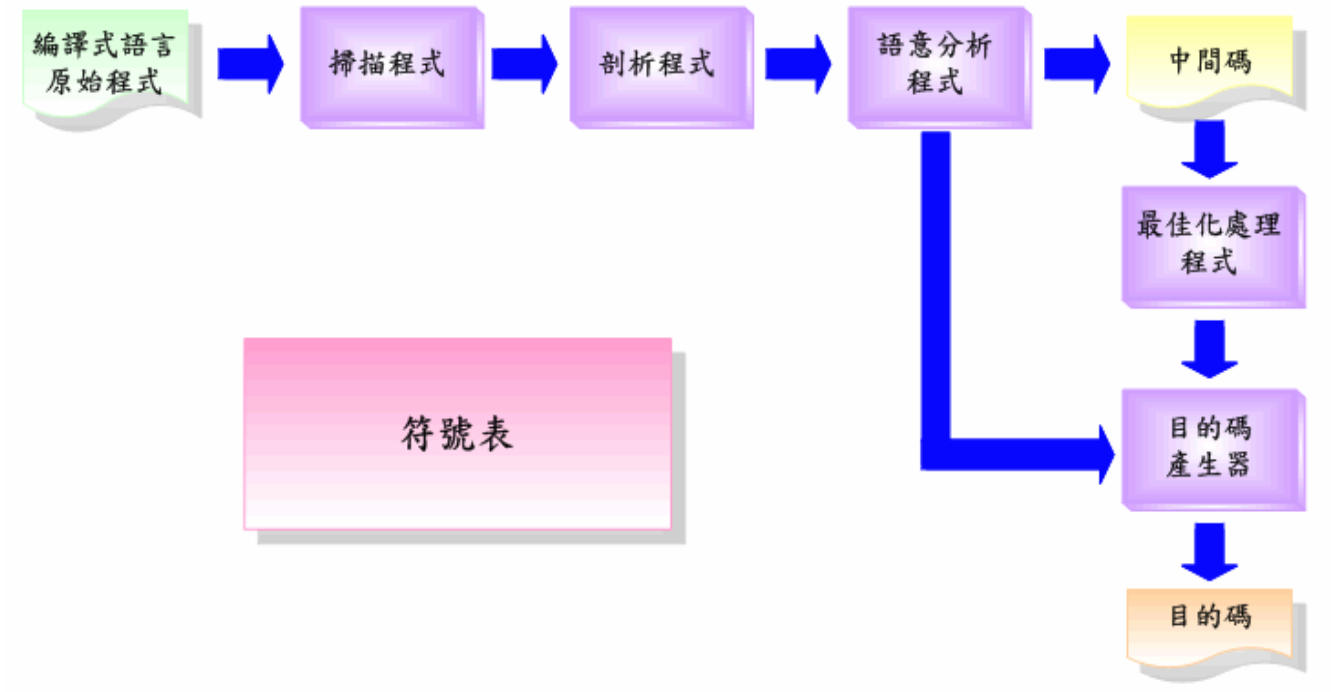


圖1-11　C語言編譯器的工作

1.5.2 編譯器(Compiler)【補充】

- 編譯器的工作主要分爲下列5個階段，並透過符號表記錄程式中相關的名稱。



編譯器的處理流程

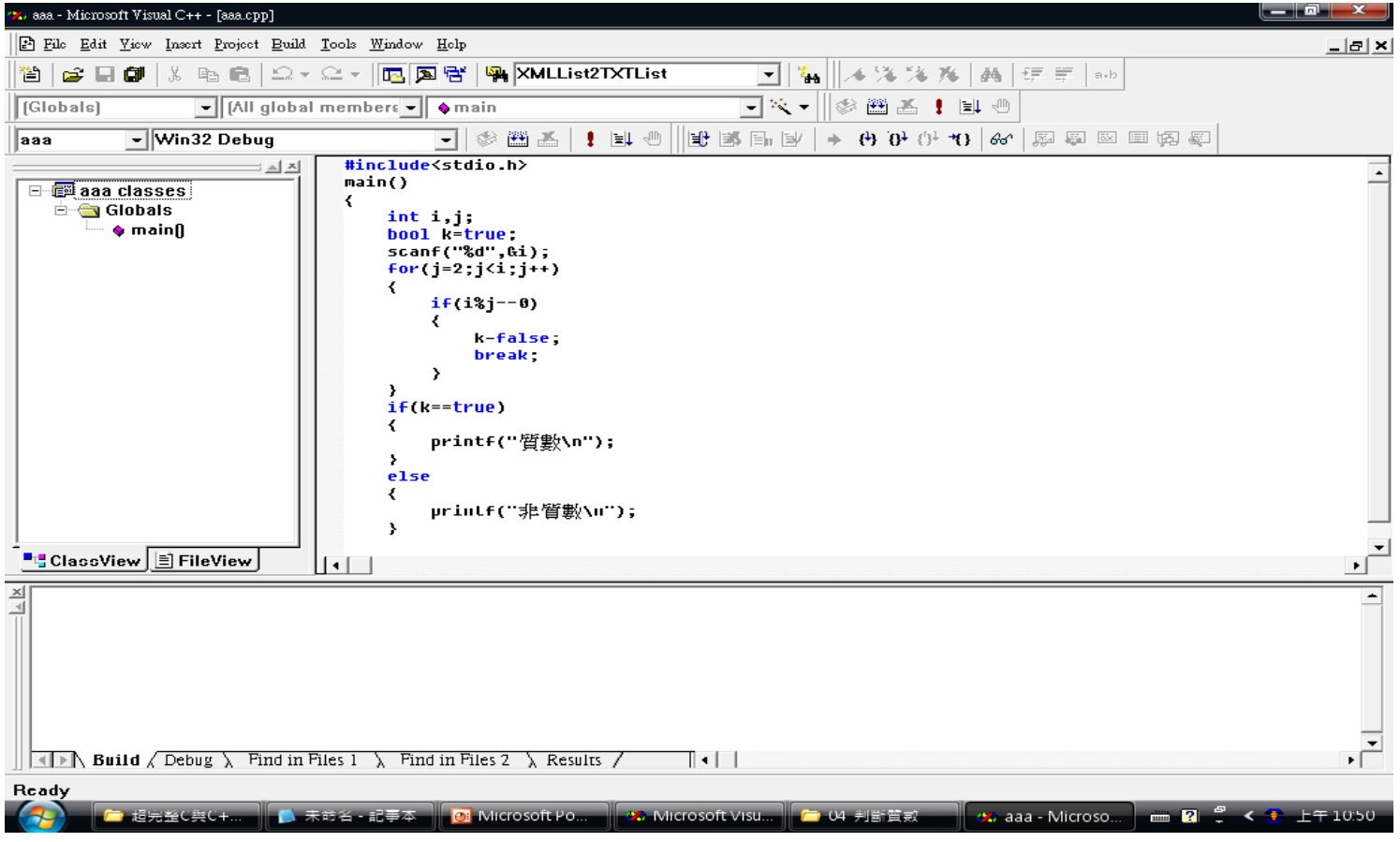
1.5.3 整合開發環境(IDE)

- 傳統的編譯程式開發流程有些繁複，程式設計師必須先用文書編輯器撰寫原始程式檔，然後再交由編譯器、連結器將原始程式檔編譯爲具有除錯功能的可執行檔，最後經過除錯無誤後，才重新編譯連結爲最終的可執行檔。不過這個流程通常不會一次就完成，因爲程式可能發生語法、語意等錯誤，而必須在編輯器、編譯器、除錯器間切換許多次，才能夠得到最後正確的原始程式與執行檔。

1.5.3 整合開發環境(IDE) (續)

- 整合開發環境（Integrated Development Environment；簡稱IDE）是一套整合性軟體，它將編輯器、編譯器、連結器、除錯器、執行程式等功能整合在同一套軟體之內，以方便程式設計師開發程式之用。
  - 目前大多數的視窗程式整合開發環境除了提供上述功能之外，還提供了許多的視窗套件供程式設計師使用（這些套件也都被整合在IDE之中）。例如：Borland公司所推出的Builder C++ Builder就是一套包含許多元件的視窗程式整合開發環境。

1.5.3 整合開發環境(IDE) (續)



Microsoft Visual C++ 6.0 圖

1.5.3 整合開發環境(IDE) (續)

- 【編譯器與整合開發環境】
  - 由於整合開發環境已經包含了編譯器的功能，因此我們常常會用整合開發環境的名稱作爲編譯器的名稱，例如Microsoft Visual C++ 6.0是一個整合開發環境，但我們也可以稱它爲一種編譯器，因爲它同樣具有編譯器的功能。但反之則較爲不當，例如GCC只具有編譯器的功能，而未將編輯器等其他功能納入，因此，我們只會稱GCC爲一種編譯器，而不會稱GCC爲整合開發環境。

1.5.3　整合開發環境(IDE) (續)

- 【C語言與C++語言的編譯器】
  - 除了少數的古老編譯器（例如Turbo C）之外，大部分的程式設計師都使用C++編譯器來編譯C語言程式，因爲，C++本身已經包含了所有的C語言程式語法。在本書的附錄中，我們介紹了非常多種類的C++編譯器，在本章中，我們只簡單示範、Dev C++及Microsoft Visual C++ 6.0整合開發環境。

1.6 本章回顧及下章預覽

- 在本章中，我們介紹了組織電腦系統的兩大元件，分別是硬體與軟體。其中軟體可以分爲程式與資料，而程式又可以分爲系統程式與應用程式。使用C語言不但可以開發應用程式，甚至連系統程式也可以使用C語言來開發，例如Unix作業系統就是其中一例。

1.6 本章回顧及下章預覽(續)

- 使用C語言開發程式最大的優點就是程式效率高，而且C語言又比組合語言更容易撰寫，因此獲得眾多程式設計師的喜愛。C語言從早期的K & R C演變爲ANSI C標準，使得眾多編譯器有了依循的標準，這同時使得C語言的可攜性得以提高。
- C語言的原始程式是副檔名爲『.c』的純文字檔，它必須經過編譯器、連結器的處理才能夠變成可執行檔。目前大多數的編譯器都具備了連結器的功能，以及某些在正式編譯前的前置處理功能，而這些額外功能其實都可以在編譯器的設定中加以指定。

1.6 本章回顧及下章預覽(續)

- C語言後來被衍生發展爲C++程式語言（加入了物件導向技術），因此目前大多數的編譯器都是C++的編譯器，但由於C++包含了所有的C語法，因此，我們也可以使用C++編譯器來編譯C語言程式，或甚至將原始程式儲存爲C++程式檔（副檔名爲『.cpp』）。
- 在本章中，我們實際練習了如何使用整合開發環境來開發C語言程式，但我們並不了解所輸入C語言程式有何用途，這些C語言敘述的實際功能，我們將從下一章開始陸續學習。

本章習題