單元 1：程式設計基礎

1.1 程式設計的意義、目標與目的

1. 什麼是程式設計 (Programming)

(1) 程式是為完成某項工作，依照達成這個工作所定義或設計的邏輯順序，編寫成的一連串指令 (instruction) 的集合。程式設計的目標即是設計出能完成某項工作的程式。

(2) 高階語言的程式使用敘述 (statement) 為最小的描述單位，也就是說高階語言的程式是利用敘述，按照其語法規則及所要處理工作的順序，編排而成的一連串指令。

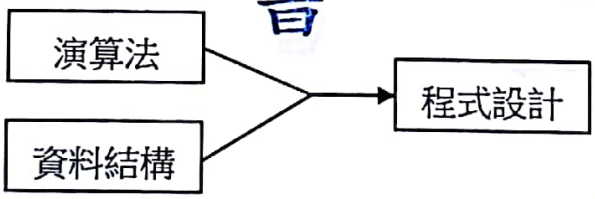
(3) 程式設計的目的，是要電腦依照我們設定好的敘述的步驟，逐步執行，以完成特定的工作。

(4) 程式設計可以是指「為解決某個問題或完成某項工作，依照所構思的解決方案，建立邏輯順序（步驟）後產生規劃藍圖，再利用程式語言依規劃藍圖撰寫程式，經過測試後能符合解決問題或完成工作的目的所從事的行為」。

2. 程式設計的流程

設計程式以解決問題，通常區分為以下幾個步驟：

(1) 分析問題 (analysis)：在設計程式以前，需要先與客戶先確認程式要完成的工作是什麼，也就是要解決的問題為何，有時簡稱為程式的功能。此外還要確認輸入、輸出的格式。依照要解決的問題，想好如何解決問題的方法，以及如何使用適當的資料結構，才能夠設計出有效率的程式。

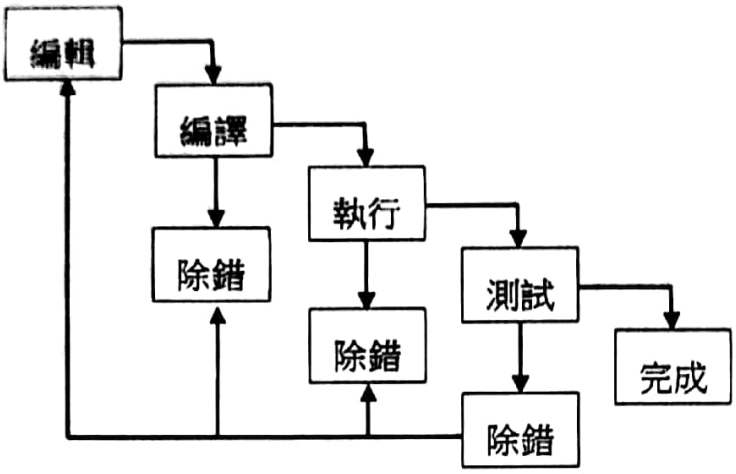


(2) 繪製流程圖或撰寫程式虛擬碼：表達演算法 (algorithms) 最常用的兩個方法是

① 流程圖 (flow chart)：以圖型化的演繹邏輯來表達程式操作順序的方法。

② 虛擬碼 (pseudo code)：類似於一種程式語言。在虛擬碼中，自然語言和高階程式語言並用，描述資料結構與演算法，兼具文字描述及流程圖優點的表示方式，是系統分析師和程式設計師之間的溝通工具。

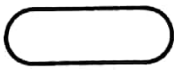
(3) 編寫程式 (coding)：根據流程圖或虛擬碼，以程式語言撰寫程式。程式經過編輯 (edit)、編譯 (compile)、執行 (run)、除錯 (debug) 與測試 (test) 等工作，獲得符合需求的可執行碼 (executable codes)。



(4) 程式安裝 (installation) 佈署 (deployment)、管理 (management) 與維護 (maintenance)：程式完成後，將程式放在客戶的硬體上，進行安裝、佈署、設定等工作，讓程式能夠完成符合客戶需求的工作。正式上限執行後，仍必須持續追蹤與維護，包含程式碼、文件說明等。

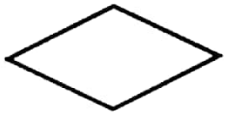
3. 流程圖

(1) 流程圖常用的符號

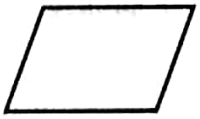
① 開始/結束： 

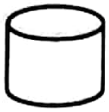
② 設定/程序： 

③ 程式執行方向：

④ 決策： 

⑤ 檔案： 

⑥ 資料（輸入/輸出）： 

⑦ 磁碟（儲存媒體）： 

⑧ 連接點： 

(2) 繪製流程圖的基本原則

① 有開始與結束符號。

② 每個符號間用帶箭頭的線連接起來。

③ 流程的方向習慣由上而下，由左至右。

④ 畫線時盡量避免交叉。

⑤ 圖形太大時，多利用連接符號，將圖分成幾個部份。

1.2 程式語言的分類

1. 簡介

(1) 程式語言 (Programming Language)：簡單的說，就是電腦懂的語言。

(2) 自然語言 (Natural Language)：人類使用的語言。

(3) 程式語言分類

① 機器語言：第一代程式語言，電腦硬體能處理的語言，只是一連串的 0、1 數字組合（機器碼）。

② 組合語言 (Assembly)：第二代程式語言，是一種接近於機器語言的表示方法，使用容易記憶的助憶碼 (mnemonics) 對應機器語言的機器碼。

③ 高階語言：第三代程式語言，設計程式的方式接近人類思維與使用的語言。

(4) 微程式 (microprogram)：根據處理器的微運算 (micro-operations) 定義，用來描述機器語言指令在 CPU 中處理動作的程式。

2. 常見的程式語言

程式語言	高低階	結構化/物件導向	特性
機器語言	低		
組合語言	低		
BASIC	高	結構化	Beginner's All-Purpose Symbolic Instruction Code
COBOL	高	結構化	Common Business Oriented Language
Pascal	高	結構化	
C	高	結構化	程式可移植性 (portability)
C++	高	物件導向	
C#	高	物件導向	
Java	高	物件導向	執行檔可在不同平台上執行
HTML	高	結構化	建立網頁的標示語言
PERL	高	結構化/物件導向	直譯、動態式的腳本語言
LISP	高	函數式類型	人工智慧語言

說明：腳本 (script) 語言是為了縮短傳統的「編寫、編譯、連結、執行」(edit-compile-link-run) 過程而建立的電腦程式語言，通常以直譯方式執行。

3. 以描述程式執行的處理過程區分，程式語言可以分成兩大類：

(1) 程序式 (procedural)：使用者需要撰寫逐步獲得結果的敘述，整個程式由一連串的指令或程序呼叫所組成。例如 BASIC、COBOL、Pascal、C/C++/C#、Java 等。

(2) 非程序式 (non- procedural)：使用者不需要撰寫要如何處理的過程，只需要定義問題的結果。這樣的語言也稱為宣告式語言 (declarative language)，例如 LISP、SQL 等。