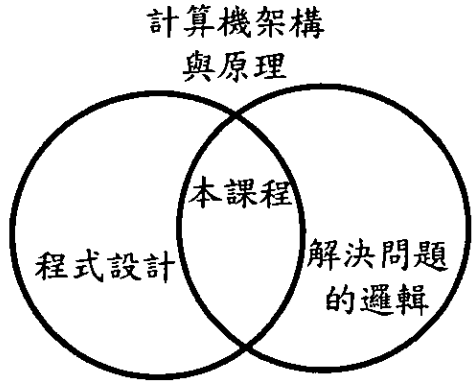
柒、 核心科目課程

7.1 計算機概論課程

計算機概論課程是大一入門課程，主要介紹計算機運作之基本觀念與基礎知識，並訓練學生能運用所學程式語言撰寫程式以解決一些簡單問題，若程式撰寫的基本功夫沒練好，將影響學生學習其他課程之學習興趣，故可依學生程度而調整上課時數，且在課堂檢討學生常犯錯誤，使其不害怕寫程式。本章首先敘述課程設計理念，進而敘述課程大綱。

7.1.1 課程設計理念



教育目標：

1. 訓練學生瞭解計算機的基本運作原理與如何設計程式以解決問題。
2. 訓練學生熟悉兩種程式語言(包含一種傳統的循序式語言與物件導向式程式語言）的結構及使用，及運用物件導向觀念設計程式。
3. 訓練學生具備程式偵錯與測試能力。

課程內容：

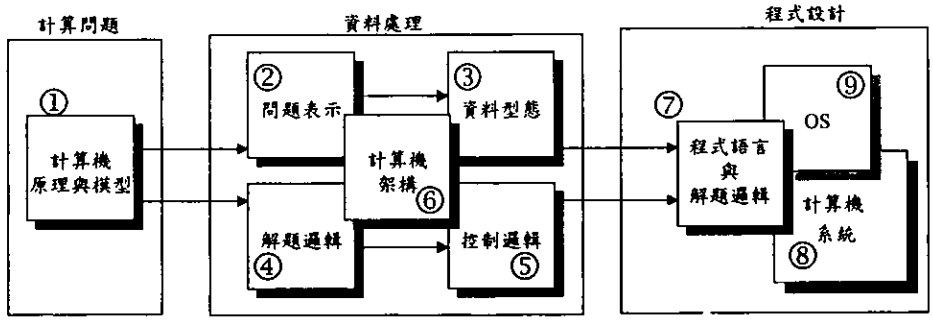
1. 電腦基本概念：硬體基本概念，作業系統，重要系統架構演進、多工、多緒、多處理器概念，執行環境概念，電腦網路。
2. 程式語言結構：Names, Scopes 及 Bindings，控制流程，資料型態，副程序及 Control Abstraction，建構可執行之程式。
3. 物件導向觀念：軟體元件、類別與物件、屬性、操作、方法、訊息、封裝、繼承。
4. 應用程式設計能力訓練。

設計構想與進行方式：

1. 課程的進行由淺顯易懂的計算機原理，配合實務操作，並經由程式語言的引導，讓學員能從解決問題的程式語言中，交互驗證計算機的運作。希望微觀(micro view)的從計算機資料處理，到宏觀(macro view)的problem-solving 的過程，來介紹計算機、軟體程式、人與計算的關係。
2. 課程分成理論授課與實務操作兩部分，必須搭配一起進行，同時給予學生適度的作業練習與程式撰寫，以驗證所學
3. 適度介紹程式偵錯(debugging)的技巧與管理，並需要每週至少一小時與學生一起分析解沃程式設計的錯誤與困難，並於適當時間實施期中期末測驗，以驗證學習成效。

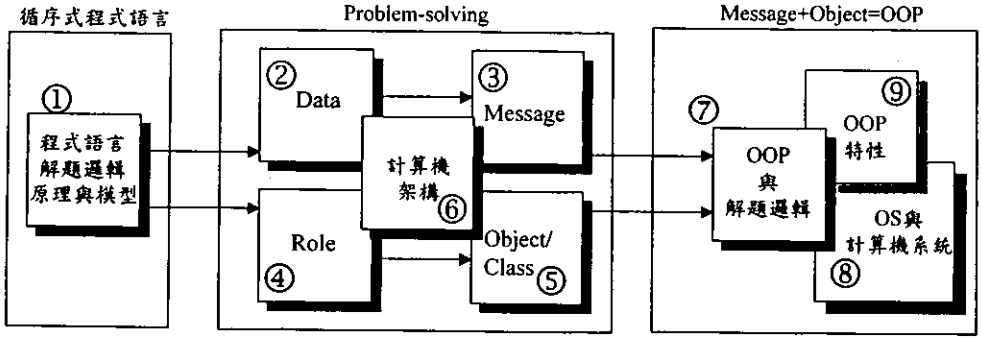
課程分成下列兩部分，分於上下學期實施：

- PART-I: (1) 從人類解決問題的需求出發，強調「計算(computing)」為解決問題的基本元素，並藉此介紹計算機的各部功能。(2) 解決問題的資料處理方式介紹問題的表示，並在計算機架構下對應資料型態。(3)以資料處理的解題邏輯，介紹計算機的處理方式(如內儲程式計算機架構)，並介紹計算機的控制邏輯功能。(4)介紹(2)及(3)的同時，需同步以程式語言的資料表示方式與控制邏輯加以對照式驗證，依照資料處理由小而大，由簡而繁的順序介紹程式如何進行資料的處理。(5)然後，由程式設計的角度看計算機四大元件的運作原理，讓學生知道在處理資料的過程中，程式如何在計算機中被處理（檔案，OS，網路等）。使學員能夠以一種循序式程式語言自行撰寫簡單的程式。概念如下圖。



- PART-II: (1) 從 PART-I 中介紹的第一種循序式程式語言的限制與問題解決(problem solving & modeling)的角度，來(2)介紹解決問題不是以程式為思考的核心，而是以角色(role)與訊息(message)溝通的方式來進行。(3)由解決問題時的問題描述與解題邏輯對應至物件與訊息，在計算機系統的架構下，逐步導出物件導向式程式語言的基本原理。(4)課程引用同一個例子貫穿整個物件導向程式設計的介紹：由物件(object)概念逐漸介紹到類別(class)，經由實例說明繼承(inheritance)及資料封裝(encapsulation)等物件式程式語言的重要特色。(5)以 OOP 的特色由簡而繁說明簡單物件與物件間，至複雜物

件間的繼承與交互合作的關係。同時，適度介紹其在計算機中運作方式與循序式程式的異同。概念如下圖。



分述如下

- PART-I: 從人類解決問題的需求出發，介紹計算機的沿革，強調「計算(computing)」為解決問題的基本元素，並藉此介紹計算機的各部功能。然後，由程式設計的角度看計算機四大元件的運作原理。從簡單的數字計算，到資料龐大而複雜的問題，引導學員逐步瞭解循序式程式語言（如 VB, C 等）的發展與特色，更重要的是讓學員知道在處理資料的過程中，程式如何在計算機中被處理（檔案，OS，網路等）。使學員能夠以一種循序式程式語言自行撰寫簡單的程式。課程設計共分為分成 14 個單元，共 80 小時，每週上課 5 小時:

Unit 1: 序論

- 簡介人類對計算機的需求並介紹計算機擅長於處理大量、重複計算的工作。介紹電腦的誕生及演變過程，並說明為何各種問題需要轉換成計算問題。各種電腦應用系統之硬體、軟體、分類（超級電腦、迷你電腦、個人電腦、PC/Notebook/PDA）之 簡介。同時需搭配動手認識電腦的實習，目的在於引發學生對計算機的興趣，並給予梗概式的認識。

Unit 2: 從人 problem-solving 的需求看計算機

- 點出計算(computing)與人 problem-solving 的關係。介紹各種不同資料形式與其運算操作的種類，並進一步說明操作型態與計算機器的設計需求。

Unit 3: 電腦/計算機系統

- 說明電腦/計算機系統由硬體、軟體、系統、人所組成。計算機四大功能運算、儲存、輸出、輸入的功能與之間的關係。同時以此概念式的方塊圖說明計算問題如何被處理。

Unit 4: 計算問題的表示方式

- 簡介表示問題的數字系統與數碼表示方式，說明 digit/bit/byte 與各種數字系統及轉換方式。介紹數字以外的問題表示方式，如符號系統表示，聲音與影像的表示。強調問題必須表示成數字型態的問題(呼應計算機四大功能& the black box)才能被計算機處理。

Unit 5: 計算機『運算』概念

- 延續前一單元說明資料如何被計算機處理的需求、概念、與演進。開始說明何謂『程式』與程式的基本元素(程式基本方塊)，並以范紐曼(Von Neumann) 『內儲程式』概念計算機作為說明的基礎。同時搭配一種程式語言作為對照式的驗證與練習，給予基本計算數字型態的資料處理指令的練習，讓學生對「計算程式」有基本認識。

Unit 6: 計算機『儲存資料』概念

- 說明在 Unit 4 所表示出來的資料如何儲存在計算機中。說明計算過程對儲存體的需求與概念，包含資料儲存與程式的儲存。資料儲存部分介紹各種 Data types 在記憶體的位置與表示方式；程式的儲存部分介紹程式在記憶體中的位置與讀取到執行的過程。同時程式語言的練習，將各種不同資料型態表示、處理並儲存在程式中，作為對照式的驗證與練習，讓學生對「資料儲存」、「資料處理」與程式間的關係有基本認識。

Unit 7: 計算機的『控制單元、輸入、輸出』

- 介紹計算機的控制單元、輸入、輸出等功能。同時藉由控制的概念說明處理資料時，程式如何進行輸入輸出的動作。程式部分則介紹簡單的程式指令，布林代數運算及 IF...THEN...ELSE 的程式控制流程，並以此進行 I/O 的練習。

Unit 8: 計算機資料處理—I (資料處理流程)

- 介紹計算機進行資料處理的流程，強調資料處理是計算機的主要功能，並說明資料處理的型態與 problem-solving 的類型間的關係。藉此介紹進階的資料表示方式如 Array/Queue/Stack 等類型與原。再者，說明此種資料結構的適用時機與 problem-solving 對應各種控制結構 control structures(For, While, Do, Switch, etc.)。旨在讓學生瞭解計算機程式處理資料時對資料與程式控制間的關係。

Unit 9: 計算機資料處理—II (模組化程式設計)

- 延續 Unit 8 的資料處理觀念，說明將經常用到的資料處理步驟進行模組化的原因、優點與種類。介紹模組化程式設計的觀念，變數或參數的使用範圍與在記憶體中的關係。讓學生將前的程式加以模組化，並呼叫系統提供的現有的函數或副程式，藉由程式的練習，比較模組化程式的優點與特色。

Unit 10: 整合性描述

- 描述電腦的五大單元如何共同完成（程式的）計算工作，需對應至

程式與資料在五大單元的關連性，計算機提供的 BIOS+ROM+RAM 等硬體支援的提供。藉此讓學生從頭思索 Unit-1~9 所述計算機、程式、資料間的互動關係，並進行較大型的練習問題。

Unit 11: 資料處理—III (大量資料的處理)

- 延續 Unit 8 的資料處理觀念，說明計算機處理大量資料的時機與方式。藉著檔案系統的介紹讓學生瞭解檔案類型、檔案變換、檔案操作（建立、新增、刪除、修改等）與程式的關係。

Unit 12: 作業系統概念

- 介紹程式設計師應瞭解的作業系統概念與所提供的服務與資源。從對計算機資源的概念談程式對 OS 服務的需求，如 IO, file systems 等等。從程式的生命週期（開啟、執行、等待、保護）介紹單一程式與 OS 的關係，同時回顧計算機的五大功能中經由 OS 保護或控制的部分。

Unit 13: 作業系統與程式的互動

- 進一步從介紹程式設計師應瞭解的多工，多緒，多處理器的概念與此種環境下作業系統概念與所提供的服務與資源。並由此介紹重要系統架構演進(如中介軟體，組譯器與編譯器，遠端呼叫,主從式架構計算環境與各種 OS 的種類與適用時機)
- 開始讓學生進行整合式的程式練習。

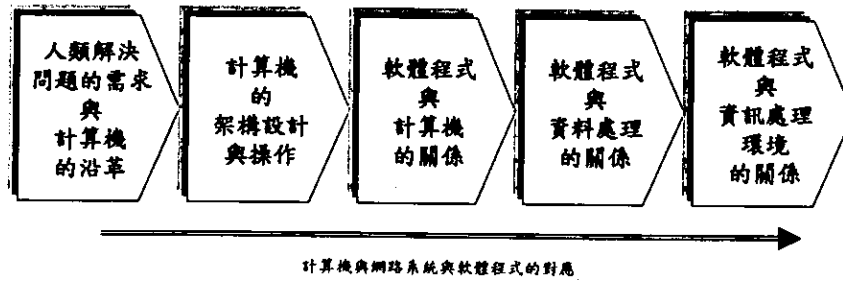
Unit 14: 電腦網路與網際網路

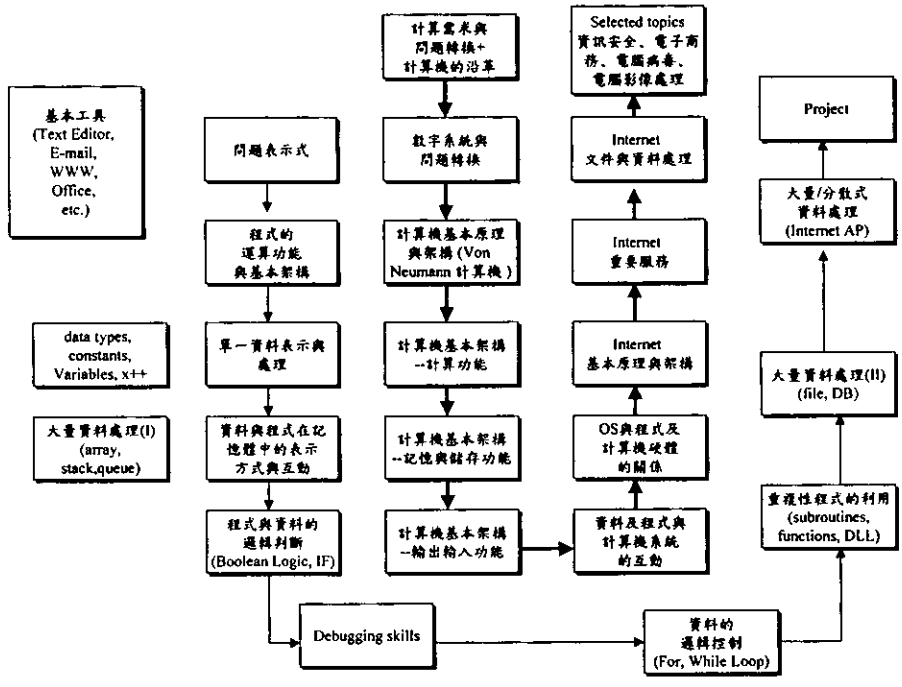
- 從「資源共享」的角度談對網路的需求與概念。經由共同約定式的通訊協定到標準化約定的通訊協定談 protocol 與網路類型（LAN/MAN/WAN/Wireless）、網際網路的沿革、Internet 的發展歷史及台灣 Internet 的沿革、IP/router，Domain name、標記語言(markup language)如 HTML 與 XML 的概念與各種網際網路的應用與重要服務。實習部分則讓學生練習以現有軟體連上 Internet，並實際架設製作網頁（站），並利用目前所學的程式技術(Java Applet, VB Script, Java Script)結合現有程式模組或函式庫撰寫簡單的 Internet 應用小程式。

Unit 15: 資訊安全與電腦應用的未來趨勢

- 介紹資訊安全與電腦應用的未來趨勢。

各單元進行的順序的關係如下圖所示：





- PART-II: 從 PART-I 中介紹的第一種循序式程式語言的限制與問題解決(problem solving & modeling)的角度，來介紹解決問題不是以程式為思考的核心，而是以角色(role)與訊息(message)溝通的方式來進行。逐步導出物件導向式程式語言的基本原理。課程引用同一個例子貫穿整個物件導向程式設計的介紹：比較分析其利用循序式程式語言解題時的限制，由物件(object)概念逐漸介紹到類別(class)，經由實例說明繼承(inheritance)及資料封裝(encapsulation)等物件式程式語言的重要特色。同時，適度介紹其在計算機中運作方式與循序式程式的異同。本課程僅介紹基本而重要的 OOP 技術，進階部分則在「軟體系統發展」課程中補充介紹。課程設計共分為分成 14 個單元，共 80 小時，每週上課 5 小時(上課學生必須先修 PART-I 課程)：

Unit 1: 以傳統程式語言發展軟體系統的限制

- 說明傳統程式語言發展軟體系統的限制，以導出物件導向式程式語言的需求。需以一個由傳統程式語言發展的軟體程式為例，說明其在設計、開發、維護、上的缺點與限制。可以讓學生用以前開發的程式，分析說明相對於本單元的限制。

Unit 2: Objects + Messages=OOP

- 介紹角色扮演(role-playing)式的問題解決策略，說明解決問題的角色與其所具有的屬性(attribute)，對應 Data structure+algorithm=program 的觀念，說明 Objects + Messages= OOP 的重要性，而物件(object)對應角色是解決問題的基本單元。用一個較生活化的實例說明本單元的內容的重要觀念。

Unit 3: From Role Model/Object to CLASS

- 從物件的多元性與從屬關係導出類別(class)的觀念，並以實例說明

類別的定義與組成元素。以 Unit-2 的實例說明物件與類別的關係。

Unit 4: Problem Solving with Objects and Classes

- 說明以 OOP 方式解決問題時的步驟，並對應 Unit-1 的傳統程式語言發展軟體系統的限制，以說明 OOP 的優點。

Unit 5: Programming with Objects

- 開始說明以 OBJECT 為程式設計的基本單元，與物件導向程式的組成元素。說明如何建置一個 OOP 的程式，以 Unit-2 的實例說明如何在程式中定義並使用一個物件。再者，說明物件封裝屬性與繼承（完全相同屬性）的基本觀念。

Unit 6: Inheritance I—the relationship between Object and Class

- 介紹「繼承」的觀念與處理相同/差異的 properties，只談到屬性完全相同的屬性（垂直繼承+不衝突的屬性），以 Unit-2 的實例說明如何在程式中定義並使用物件的繼承關係。

Unit 7: Summary –I

- 對應 Unit-1 的傳統程式語言發展軟體系統的限制，以說明 OOP 的優點。

Unit 8: Life-cycle of Objects

- 說明軟體物件的生命週期與資料範圍(scope)，用以說明物件將資料隱藏起來的處理方式，與其在計算機系統如何被處理（物件起始化、執行、結束、多執行緒等重要觀念）。以 Unit-2 的實例說明程式中的物件的生命週期。

Unit 9: Use Package – I & Integration I

- 使用系統提供的軟體物件進行程式的輸出入(Window-based GUI, file I/O, etc.)。以 Unit-2 的實例說明如何在程式中加入 GUI 的物件。同時可以讓學生以至目前為止的 OOP 技術，設計一個較大型的作業。

Unit 10: Inter-Access of Properties among Objects

- 談物件屬性的公開與私密性（private/public），說明物件之間橫向與縱向的屬性呼叫與對應的方法(methods)，以 Unit-2 的實例說明程式中的物件如何使用父系物件的屬性，與其他類別的屬性。

Unit 11: Inheritance II—the relationship between Object and Class

- 說明屬性繼承發生衝突時的處理方式，介紹 subclass/superclass 的定義與使用，其相關的程式技巧。以 Unit-2 的實例說明程式中的物件增加與父系物件不同或衝突的屬性。

Unit 12: Abstract Class v.s. Normal Class

- 介紹概念式物件的意義與使用時機。說明介面物件與多重繼承的觀念。以 Unit-2 的實例說明程式中的物件如何使用概念式物件。

Unit 13: Use Package II --GUI & Container

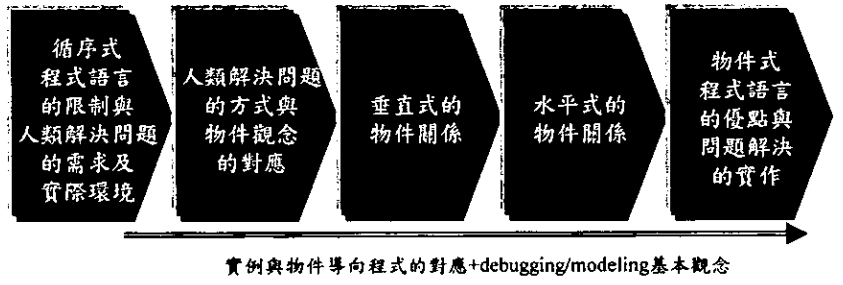
- 使用更多系統提供的軟體物件(.io, .util, .net)。以 Unit-2 的實例說明如何在程式中加入 GUI 與網路功能的物件。說明 event delegation model, container 等觀念同時可以讓學生以至目前為止的 OOP 技

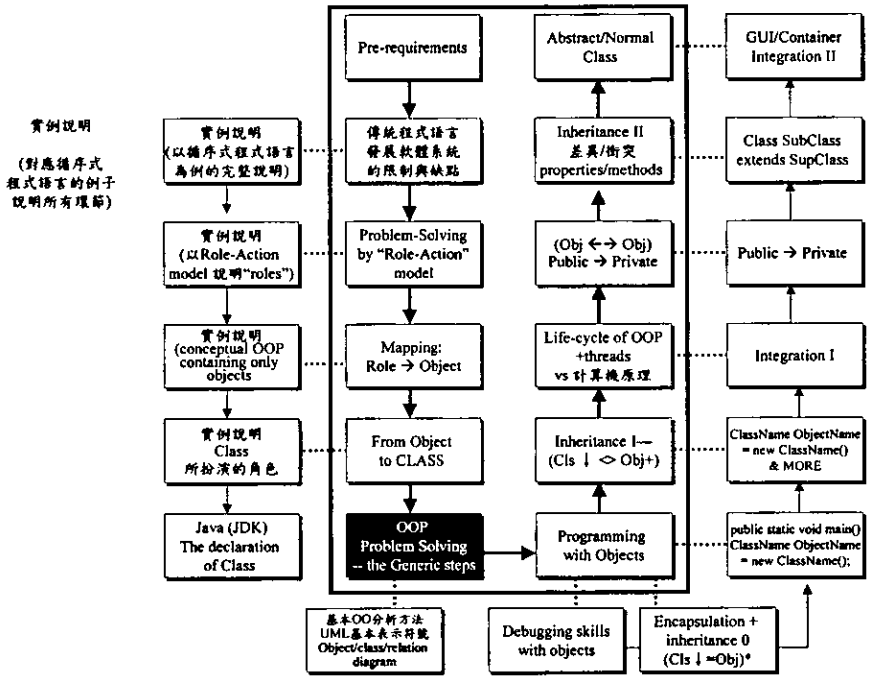
術，設計一個大型的 project。

Unit 14: Summary II

- 對應 Unit-1 的傳統程式語言發展軟體系統的限制，以說明 OOP 的優點。

各單元進行的順序的關係如下圖所示：





7.1.2 課程大綱

計算機概論（一）課程大綱

Core Knowledge	授課時數(Hr)	Programming Skills+LAB	實習時數(Hr)
1. 序論 ■ 對計算機的需求—人、資料(大量、重複計算) ■ 電腦的誕生及演變過程 ◆ 齒輪與碼表的模型 ◆ 速度(Hz)與散熱問題 ■ 問題的轉換—(各種問題→計算問題) ■ 硬體、軟體、電腦的分類 ◆ 超級電腦、迷你電腦、個人電腦、PC/Notebook/PDA ■ 各種電腦應用系統 ■ 簡介與其他	3	● 基本工具認識安裝(e.g., text editor) ● LAB: 各種應用軟體 ● LAB: 各種電腦外觀、種類、型態、廠牌... ● LAB: PC DIY**	3
2. 從人 problem-solving 的需求看計算機 ■ 點出計算(computing)與人 problem-solving 的關係 ■ The needs of representation and operations of data on machines ■ The requirements of such a "machine"	1	● LAB:各種軟體程式 ■ 系統程式 ■ 應用程式 ■ BIOS/firmware	
3. 電腦/計算機系統--硬體、軟體、系統、人 ■ 計算機四大功能 ■ 運算、儲存、輸出、輸入(只談外觀、種類、型態、廠牌...) ■ A Black Box Model – Computer ■ 問題的轉換—(各種問題→計算問題)	2	● LAB: Windows 小算盤 ● LAB: UltrEditor: 各種資料型態的內部表示法 ● LAB: UltrEditor: 程式執行檔的內部表示法	
4. 數字系統與數碼表示方式 ■ digit/bit/byte + base 2/8/16/32 ■ 數字系統轉換 ■ 符號系統表示 ■ 聲音與影像的表示 ■ 計算機運算模型---問題必須表示成數字型態的問題 (呼應計算機四大功能& the black box)	4	● Homework Assignment:	

5.	計算機『運算』概念 ■ 需求與概念 ■ 計算機的操作-->開關-->1000110-->程式(不談語言演進) ■ 何謂『程式』? ◆ 程式的基本元素(程式基本方塊) ■ 計算機運算模型--『內儲程式』概念計算機 ◆ 需求 ◆ 范紐曼(Von Neumann) 『內儲程式』概念計算機 ◆ 何謂「程式」 ◆ 『內儲程式』概念計算機架構與運算原理 ◆ 此架構下的各種工作 ■ CPU/register (program counter)	3	● Programming-00 (建議用 VB/Java) ■ program 基本結構 ■ commands; statements ■ sequential execution ■ compiler ■ Ex: simple commands, e.g., printout 3+5 (numeric +-*÷only) (暫不談 control commands) ● Homework Assignment:	3
6.	計算機『儲存』概念 ■ 需求與概念 ■ 資料儲存 ◆ 沒有記憶體的計算機 CPU 如何運作 ◆ 記憶體—physical CKT/memory space/logic meaning ◆ Data types in memory ■ 程式儲存 ◆ 程式在記憶體中的位置 ◆ 讀取程式 ■ 程式執行—interaction among CPU/program/memory/data ■ CPU/register/memory	2	● Programming-01 ■ More commands; statements ■ data types ■ data/constants/variables/program in memory space ■ the use/call of data in programs ■ Ex: simple data operation, e.g., printout 3+5, "Red"+"Apple", "a"→ "A" (暫不談 control commands) ● Debugging Skills in Programming with dumping memory for viewing constants and variables--I (配合記憶體與程式執行的觀念) ● Homework Assignment:	3
7.	計算機『控制單元、輸入、輸出』概念 ■ 需求與概念 ■ 程式如何輸入輸出資料 ◆ 控制單元 ◆ 輸入—執行的步驟 ◆ 輸出—執行的步驟 ■ 資料處理—0 (資料的表示與輸出入處理)	2	● Programming-02 ■ Simple calculation & variable assignment (y=x+1; x=x+1) ■ 布林代數運算元與運算子 ■ 布林代數運算 in VB (>=< IF…THEN…ELSE) ■ Simple I/O control (暫不談其他 control commands) ● Debugging Skills in Programming with dumping memory for viewing constants and variables+check points--II (配合記憶體與程式執行的觀念) ● Homework Assignment:	6
8.	資料處理—I (資料處理結構與流程)	6	● Programming-03:大量資料的處理	3

● 資料處理是計算機的主要功能 ● 資料處理的型態 v.s. problem-solving ■ 處理結構性資料的方式 ◆ Array/Queue/Stack ◆ 類型與原理 ◆ 適用時機＋比較		I ■ Array/Queue/Stack 類型與原理 ● Debugging Skills in Programming with array/queue/stack ● Homework Assignments	
9. 資料處理—I (資料處理結構與流程) ■ problem-solving 對應各種 control structures ■ 程式的 control 結構 ◆ control structure 的種類 ◆ control structure 的應用時機 ■ control structure 的原理	6	● Programming-03:大量資料的處理 I ■ 程式的 control 結構 ■ control structure 的種類 ■ control structure 的應用時機 ■ control structure 的原理 ■ 以 do loop 處理資料 ◆ 在 Memory 如何動作 ◆ For/While in do loop ■ 整合性練習 ● Debugging Skills in Programming with control structures ● Homework Assignment:	3
10. 資料處理—II (模組化程式設計) ■ what/when 模組化程式設計 ■ why no in a single program? ■ subroutine 的觀念 ■ subroutine 的種類 ■ call-by-value/reference/address ■ modulized programming ■ scope of program/variable ■ 主從程式間的關係（對應 program counter/memory space）	6	● Programming-04:大量資料的處理 II ■ Procedure/function 的類型 ■ 將前面的程式碼加以模組化 # ● Debugging Skills in Programming with modulized structures ● Homework Assignment:	2
11. 整合性描述 ■ 電腦的五大單元如何完成（程式的）計算工作? ■ 對應至 code/data/extra/stack segment 觀念 ■ BIOS+ROM+RAM, etc.: relationships and block diagrams ■ 對應 unit-3 的觀念	1		0
12. 資料處理—III (大量資料的處理) ■ what/when 大量資料的處理 ■ why no in a single program with many constants and variables? ■ 檔案類型 ■ File Handle/File System	2	● Programming-05:大量資料的處理 III ■ IO by Files ■ File handle (creation/deletion/ modification) ■ Data to/from files ■ File 的轉換 by programs ● Debugging Skills in Programming with files ● Homework Assignment:	2

13. 作業系統概念 ■ 需求與概念 ■ computational resources (分配與控制) ■ 再談何謂「程式」與 program counter ■ services provided by OS ◆ IO, file systems, etc. ■ 單一 CPU 系統與程式 ■ creation/execution/protection/waiting/termination of a program ■ 單一程式與 OS 的關係 ◆ program 的生命週期(對應 unit-9 modulized programs) ◆ flow-char/control sequence ◆ 程式執行環境設定 ■ 系統軟體	2	● LAB: Windows (task manager/ MyComputer/ server/ monitor) ● Programming-06 ■ Use services provided by OS (e.g., printing, file systems, sound, video, etc.) ■ Windows API/DLL ● Homework Assignment:整合型練習—using array/stack/queue + files + sub-routines + OS services to take care different size of data ● Integrated project (given by some existing algorithms)	2
14. 作業系統與程式的互動 ■ 多工, 多緒, 多處理器概念與程式間的關係 ■ Program vs OS (多重程式與多工多處理器 OS 的關係, program 的生命週期 ■ 重要系統架構演進 ◆ 中介軟體, 組譯器與編譯器 vs 作業系統 ◆ 遠端呼叫 ◆ 主從式架構計算環境與程式執行 ◆ OS 的種類與適用時機 (Windows/Unix/Linux/Mac OS, etc)	2		
15. 電腦網路與網際網路 ■ 需求與概念 ■ 網路的興起--網路的需求+資源共享 ■ Protocol 概述—共同約定式→標準化約定 ■ 網路類型 --LAN/MAN/WAN/Wireless ■ 網際網路的沿革&Internet 的發展歷史 ■ ARPAnet，NSFNET，台灣 INTERNET 的沿革，TANet, TANet2 ■ Internet 上的身分識別 ◆ IP/router，Domain name	6	● LAB: Access to the Internet ● LAB: 網路的架設實務 ■ File server/Printer Server ■ 網路線/網路卡 ■ HUB/Modem/ADSL ■ 在 Windows 的相關設定 ● LAB: ■ 建立 Homepage ■ 作業系統環境之建立與設定 ■ 區域網路與網際網路設定 ■ 伺服器管理 ● Homework Assignment: ■ Programming-07: Access Internet with Java Applet (Java Script/VB Script) ■ Programming-08: connecting	4

■ Internet 上的文件 ◆ The concept of markup language ◆ HTTP/HTML ◆ XML—concept & applications ■ Internet 上的重要服務與伺服器管理與重要功能介紹 ◆ firewall/telnet/ftp/SSH/RPC/ email/news/proxy/search engine/ICQ/E-mail/ ftp/BBS/Plugins/其他		to the Internet (optional:: Connect to Internet or Intranet, Access to file server/printer server (using VB/Java components))	
16. 資訊安全與電腦應用的未來趨勢 ■ 資訊安全 ■ 電腦應用的未來趨勢	1		0
時數建議—四學分五小時（16 週共 80 小時）	49		31

計算機概論（二）課程大綱

Core Knowledge	授課時數(Hr)	Programming Skills+LAB	實習時數(Hr)
0. Pre-requirements ● Procedure-based language 基本觀念與實作 ● Program running on a computer (CPU, Memory, I/O, OS) ● Von Neumann's Computer Architecture ● 程式語言--review ■ 機器語言/組合語言/高階語言 ■ 4GL/自然語言 ■ Compiler/Interpreter ■ Programming Language – Definition, Usage, Declaration	0	● Step-by-Step statement; ● Constant & Variable ● Data type ● Data type transformation 未來可對應到 ◆ data type →(Class) ◆ variable →(Object) ◆ DTT→(Polymophism)* ● Statement (Std-in and Std-Out) ● Control Structures ◆ If else ◆ Do-loop (For/While) ● Array/Queue/Stack ● Procedures/Functions (by reference and by value) ● File (FILESTREAM) ● Java environment* ■ JDK、javac、java(virtual machine)、path environment setup、Java Code Convections	0
1. 以傳統程式語言發展軟體系統的限制 ● One Big Main File ● Main File + Functions/Procedures ● Reusable codes ● Complete and complex ALGORITHM design ● Cooperative development ● Data Structure + algorithm = Program ● 缺點 ● Program management ● Debugging ● Reusability ● Development Hour/Man ● Hard to design complete ALGORITHM ● The needs of programming languages	3	以實例說明之 Ex. ● A simple Lotto/Workflow program implemented by traditional languages. (需搭配完整的程式實例) ● Homework Assignment:	2
2. Objects + Messages=OOP ● Problem-solving by roles, not only by algorithms ● Modeling the roles by objects	6	以人類的問題解決方式與環境說明之 ● 分析 Lotto/Workflow problem (不需談程式)+其中的 role/object	3

● Role playing by "attributes" ● The mission of roles = handling data ● Role actions(intra-/inter-roles) = program ● The Role Model = Object ● Problem-solving by OOP = ● 只有 Object 是不夠的, An object is an instance of Class		● the role model of object in problem solving ● A "conceptual" OOP containing only objects	
3. From Role Model/Object to CLASS ■ Various objects ■ Class, why? ■ The definition of Class ■ The component of Class ■ Data members (properties) ■ Member functions (methods)	6	I: 以人類的思考邏輯說明之 ● 分析 Lotto/Workflow program 中的 Class (不需談程式) II: In Java (JDK) ● The declaration of Class ● Lotto/Workflow Class in Java	3
4. Problem Solving with Objects and Classes ■ Generic steps ■ 對應 Unit-1 Class/Object 的優點是什麼?	2	● 問題轉換 ● Lotto/Workflow problem→OOP 的步驟 (不需談程式)	1
5. Programming with Objects ■ The definition of Application ■ The declaration of Application ■ The definition of Object ■ The construction of Object ■ Simple program in object/class ■ Encapsulation + property inheritance 0 ● Object 封裝屬性，方法的特性 ● 基本繼承的觀念	6	以傳統程式語言的觀點說明對 inheritance 的需求 I: 以人類的思考邏輯說明之 ● 分析 Lotto/Workflow program 中的 Class/Object (不需談程式) II: In Java (JDK) ● The entrance of program execution ● public static void main() ● ClassName ObjectName= new ClassName(); ● Ex: Handling Lotto/Workflow Objects in Java ● Debugging Skills in Programming with objects --I (配合記憶體與程式執行的觀念) ● Homework Assignment:	2
6. Inheritance I—the relationship between Object and Class ■ What/Why inheritance ■ 繼承什麼東西--definition ■ 處理相同/差異的 properties ■ 只談到屬性完全相同的屬性垂直繼承+不衝突的屬性	5	I.以人類的思考邏輯說明之 分析 Lotto/Workflow program 中 Class/Object 的 inheritance 關係 (不需談程式) II: In Java (JDK) Object/Class Inheritance in the Lotto/Workflow problems using Java Debugging Skills in Programming with object inheritance --I (配合記憶體與程式執行的觀念) Homework Assignment:	2
7. Summary I -- OOP 對應傳統循序式	1		0

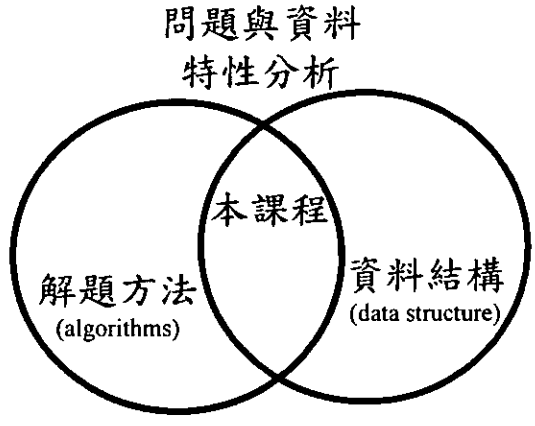
語言的優勢			
8. Life-cycle of Objects ● The Life-Cycle & Life-Scope of Object ● Data driven/Event driven ■ The invoking of objects by data/events ■ Initialization, Survival, Finalization ● The scope of objects ■ Data hiding ■ Access control of properties ■ Access control of methods ● The control of objects in a view of OS ■ Objects in memory spaces ■ Data/message passing among objects ● Threads（the needs and basic conceptual model）for managing objects in OS ● 對應 Von Neumann 計算機架構 ● OOP 的程式架構 （對應 unit I-12）	6	I.以計算機的運作角度說明之 Data/Even driven in the Lotto/Workflow problems(不需談程式) 說明 class/object/property 在計算機的四大元件+OS 的概念模型（不需談程式) II: In Java (JDK) ■ Constructor ■ this&this() ■ Destructor ■ Exercises ■ Debugging Skills in Programming with object (配合記憶體與程式執行的觀念) ■ Homework Assignment:	2
9. Use Package – I ● File Access using Java Packages ■ java.io	3	■ File objects, FileStreamReader, FileStreamWriter ■ GUI version "Hello, World" ■ 將 Lotto/Workflow 的問題資料儲存起來 Integration I ■ 完整的 JAVA 作業/project ■ Learning Management & Summarization of Bugs	2
10. Inter-Access of Properties among Objects ● Public → Private ● Private/Public properties ■ Access "Properties" ■ Private/Public--definitions ■ The access of Object's properties ■ The access of Object's methods	6	I. 以人類的思考邏輯說明之 ■ 分析 Lotto/Workflow program 中 Class/Object 之間(垂直＋水平) private/public 屬性關係(不需談程式) II: In Java (JDK) ■ (public v.s private, not including protected) ■ setProperties & getProperties	1
11. Inheritance II—the relationship between Object and Class ● 處理差異/衝突的 properties/methods ● The definition of subclass and superclass ● The declaration of subclass ● The component of Subclass	6	I. 以人類的思考邏輯說明之 ■ 分析 Lotto/Workflow program 中 Class/Object 的 inheritance 關係(不需談程式) II: In Java (JDK) ■ Class SubClassName extends SupClassName {...}	1

■ Data members(properties) ■ Member functions (methods) ■ The inherited data members ■ The inherited member functions		■ Inheritance in the Lotto/Workflow problems using Java ■ Exercises	
12. Abstract Class v.s Normal Class ● Definition ● Interface ● Multiple Inheritance	3	I. 以人類的思考邏輯說明之 分析 Lotto/Workflow program 中需要 Abstract Class 的時機 (不需談程式) II: In Java (JDK) Shape(abstract class) v.s. Triangle, Circle, Square (concrete class)	1
13. Use Package II --GUI & Container ● Using Java Packages ■ java.io, java.util, java.net, etc. ■ Swing I ● Event delegation model ● GUI Components ■ Containers, etc. ■ Swing II	3	GUI/Internet version of the Lotto/Workflow problems	1
14. Summary II -- OOP 對應傳統循序式語言的優勢	2	Integration II ■ 完整的 JAVA 作業/project ■ Learning Management & Summarization of Bugs	1
時數建議—每週 5 小時（共 80 小時）	58		22

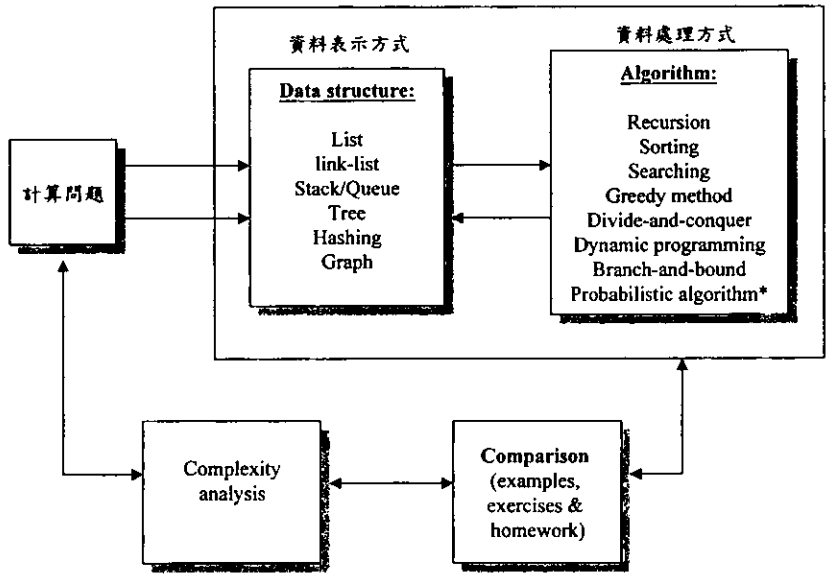
7.2 資料結構與演算法

7.2.1 課程設計理念

「資料結構與演算法」課程主要訓練學生如何運用適確的資料結構與解題策略來解決問題，是程式設計人員必備的核心知識與技術。程式除了需能解決問題外，當資料量龐大時，其執行的效能更是設計考量的重點，程式效能與選用之資料表示方式及演算法有密切關係。本課程的目標，在使學員充分了解各種常用的資料結構及演算法，使其正確的應用在程式設計上以發展出高品質的程式。



本課程的設計著重實際應用，避免艱深的問題解析。課程中儘量使用與生活相關的範例解說，配合程式實作與比較，使學員能活用資料結構與演算法。實作練習中並以物件導向的概念，要求學員將程式寫成元件，以供日後使用。



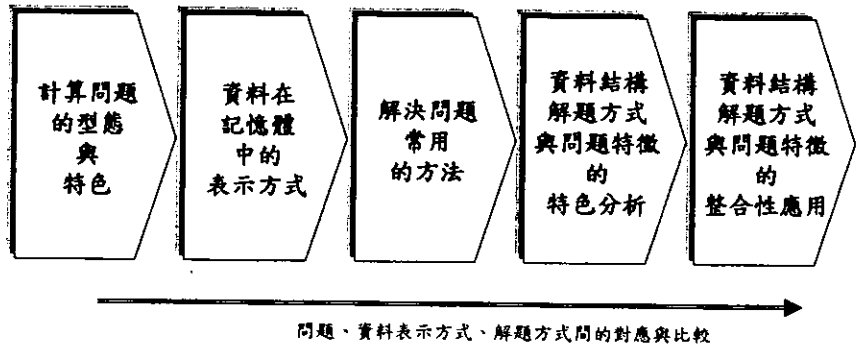
教育目標：

本課程的目標，在使學員充分了解各種常用的資料結構及演算法，且能將其正確的應用在程式設計上以發展出高品質的程式。

設計構想與進行方式：

本課程的設計著重實際應用，避免艱深的問題解析。課程中儘量使用與生活相關的範例解說，配合程式實作與比較，使學員能活用資料結構與演算法。實作練習中並以物件導向的概念，要求學員將程式寫成元件，以供日後使用。在課程進行中，前後單元有關的題材，儘量用同樣的範例說明，以供對照比較，使學員對整個課程融會貫通。

各單元進行的順序的關係如下圖所示：



課程內容：

本課程的內容涵蓋鏈結串列(Linked List)、堆疊(Stack)、佇列(Queue)、遞迴(Recursive)、樹狀結構(Tree)、資料排序(Sorting)、搜尋(Search)、圖形(Graph)、和5種基本的演算法。

Topic 1：課程開始時，以解決問題的觀點，說明資料有那些型態、需要如何處理，並以矩陣相乘的範例讓學員體會資料結構和資料處理的關係。

Topic 2：從陣列在應用上的限制開始，導入鏈結串列的需求，並比較兩者的優缺點及應用場合。

Topic 3、Topic 4：堆疊與佇列亦為密切相關的資料結構，也需要比較其個別的應用時機。

Topic 5：介紹遞迴的程式技巧，以供接續的各單元應用。

Topic 6：第六單元樹狀結構從一般樹的缺點帶入二元樹的主題，再從二元樹可能遇到的問題引入各種特殊樹; 此單元很多內容都深具實用價值，故安排較長的時間，以便充分的探討。

Topic 7：資料排序與搜尋，先介紹一些常用的排序方法，使學者了解各種不同排序方法的效能和適用場合。排序的目的是為了方便搜尋，因此接著介紹各種排序對應的搜尋演算法。

Topic 8：探討圖形。在介紹了圖形的定義與表示法等基本觀念後，以生活上的最短路徑問題說明其應用。

Topic 9：介紹解題的5種演算法，屬於進階的應用。

7.2.2 課程大綱

Core Knowledge	Programming skills/LAB	時數
1. 資料結構概論 ■ 資料與資訊 ■ 資料型態 ◆ 基本資料型態 – 字元、整數、浮點數 ◆ 延伸資料型態 - 陣列、鏈結串列、堆疊、佇列、類別 ■ 資料的操作 – 建立、取用、新增、刪除 ■ 資料結構 + 演算法 = 程式 ■ 評估程式的準則 ◆ 執行速度快 ◆ 佔用記憶體空間少 ■ 資料結構的意義 ■ 演算法的意義 ■ 陣列的意義(複習) ■ 以矩陣相乘運算為例，說明原始資料的類型及採用不同的資料結構，對程式執行速度與記憶體使用的影響。 ■ 空間複雜度的意義 ■ 時間複雜度的意義及 Big-O 表示法 ■ 當資料量很大時，時間複雜度的各種等級的關係 $\log n < n < n \log n < n^2 < n^3 < 2^n$	● 以陣列實作兩個 100×100 的矩陣相乘運算，使用兩種類型的矩陣: 1、所有元素都不為 0，及 2、大部份元素都為 0(稀疏矩陣)。	3
2. 鏈結串列 ■ 陣列結構在新增與刪除資料時的問題 ■ 鏈結串列的意義 ◆ 有序的串列 ◆ 使用動態配置的分散記憶體空間 ■ 鏈結串列的運算 – 走訪、連結、節點的插入、節點的刪除、結構的反轉 ■ 陣列與鏈結串列的比較	● 利用鏈結串列實作兩多項式相乘。就多項式缺項的多寡，比較使用鏈結串列實作與使用陣列實作的優劣。 ● 以程式實作鏈結串列的各種運算。	3

◆ 資料的新增與刪除 ◆ 記憶體空間的大小 ◆ 資料的存取 ■ 鏈結串列的延伸 ◆ 環狀單向鏈結串列 ◆ 雙向鏈結串列 ◆ 環狀雙單向鏈結串列 ■		
3. 堆疊 ■ 堆疊的定義 ■ 堆疊的基本運算 ■ 堆疊的實作 ◆ 用陣列結構 ◆ 用鏈結串列 ■ 堆疊的應用 ◆ 副程式的呼叫與返回 ◆ 運算式的轉換與求值 ■ 應注意的問題 – 資料是否超過堆疊最大容量	● 使用堆疊計算中序運算式的值。	3
4. 佇列 ■ 佇列的定義 ■ 佇列的基本運算 ■ 線性佇列與環狀佇列的實作 ◆ 用陣列結構 ◆ 用鏈結串列 ■ 優先佇列 ■ 雙佇列 ■ 佇列的應用 - 買票問題	● 實作排隊買票問題	3
5. 遞迴 (Recursive) ■ 遞迴的意義 ■ 比較遞迴與迴圈的優缺點，以實作階乘 (Factorial) 運算為例。 ◆ 執行速度 ◆ 使用記憶體空間	● 分別以遞迴與迴圈實作階乘 (Factorial) 運算。	3
6. 樹狀結構 ■ 樹的定義	● 實作二元樹的走訪。 ● 實作 AVL 樹。	12

■ 樹的表示法 – 陣列表示法、鏈結串列表示法 ■ 一般樹的缺點 – 節點的分支數差異大時，浪費儲存空間。 ■ 二元樹的基本性質 ■ 二元樹的儲存方式 ■ 二元樹的建立 ■ 二元樹的走訪 – 前序、中序、後序走訪 ■ 引線二元樹 ■ 二元搜尋樹 ■ 二元樹的應用 – 運算式處理 ■ 二元樹的缺點 – 可能產生歪斜樹 ■ AVL 樹 ■ m 元搜尋樹及 B 樹 ■ Huffman 樹		
7. 資料排序與搜尋 ■ 排序的定義 ■ 內部排序法 ◆ 氣泡排序法 ◆ 選擇排序法 ◆ 二元樹排序法 ◆ 堆積排序法 ◆ 快速排序法 ■ 外部排序法 ◆ 合併排序法 ■ 各種排序法的比較 ■ 搜尋的定義 ■ 在循序結構上的搜尋 ◆ 循序搜尋法 ◆ 二分搜尋法 ◆ 內插搜尋法 ■ 索引結構的搜尋 ◆ 直接索引 ◆ 二元搜尋樹的索引 ◆ B 樹的索引	● 實作下列一種排序。可採分組方式，各組實作不同的排序。 ◆氣泡排序 ◆選擇排序 ◆二元樹排序 ◆堆積排序 ◆快速排序 ◆合併排序 ● 實作下列一種搜尋。可採分組方式，各組實作不同的搜尋，但使用相同的資料。例如: 資料為某班學生的成績，考慮分別使用學生姓名、學號、座號為 key 的情況。 ◆循序搜尋 ◆二分搜尋 ◆內插搜尋 ◆雜湊法	10

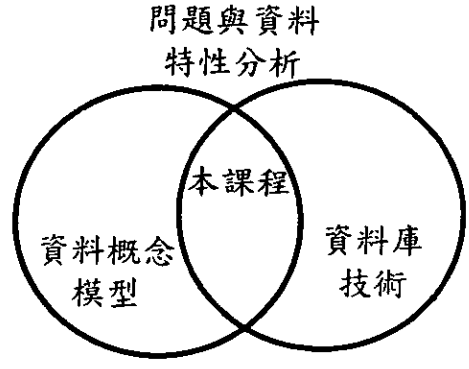
■ 雜湊法 (Hashing)		
8. 圖形 ■ 圖形的定義 ■ 圖形的表示法 ■ 表示圖形的資料結構 ◆ 鄰接矩陣(adjacency matrix) ◆ 鄰接串列(adjacency list) ◆ 加權圖的鄰接矩陣 ■ 圖形的走訪 ◆ 廣度優先走訪(Breadth First Search,BFS) ◆ 深度優先走訪(Depth First Search,DFS) ■ 擴張樹 (Spanning Tree) ■ 最短路徑 (Shortest Path) ■ 拓樸排序 (Topological Sorting) ■ 關鍵路徑 (Critical Path)	● 實作 Dijkstra 演算法找最短路徑。 ● 實作拓樸排序。	6
9. 解題方法 ■ 貪婪演算法 (Greedy Algorithms) ◆ 貪婪演算法的精神 ◆ 貪婪演算法實例一：Minimum Spanning Trees (Kruskal and Prim Algorithms) ◆ 貪婪演算法實例二：Huffman 樹 ■ 分裂征服演算法 (Divide-and-Conquer Algorithms) ◆ 分裂征服演算法的精神 ◆ 分裂征服演算法實例一：二分搜尋法 ◆ 分裂征服演算法實例二：快速排序法 ■ 動態規劃演算法 (Dynamic Programming) ◆ 動態規劃演算法的精神 ◆ 動態規劃演算法實例一：Knapsack Problem	● 實作 Game tree	20

◆ 動態規劃演算法實例二：Matrix-Chain Problem ■ 分支設限演算法 (Branch-and Bound) ◆ 分支設限演算法的精神 ◆ 分支設限演算法實例一：Game Trees ◆ 分支設限演算法實例二：3-Coloring Problem ■ 機率演算法 (Probabilistic Algorithms) ◆ 機率演算法的精神 ◆ 機率演算法實例一：Randomized Quicksort ◆ 機率演算法實例二：Random Search		
	合計	63

7.3 資料庫系統

7.3.1 課程設計理念

前言：介紹資料庫相關的技術，並著重資料庫程式設計師在資料庫軟體設計過程中所扮演的角色應注意的事項。



教育目標：

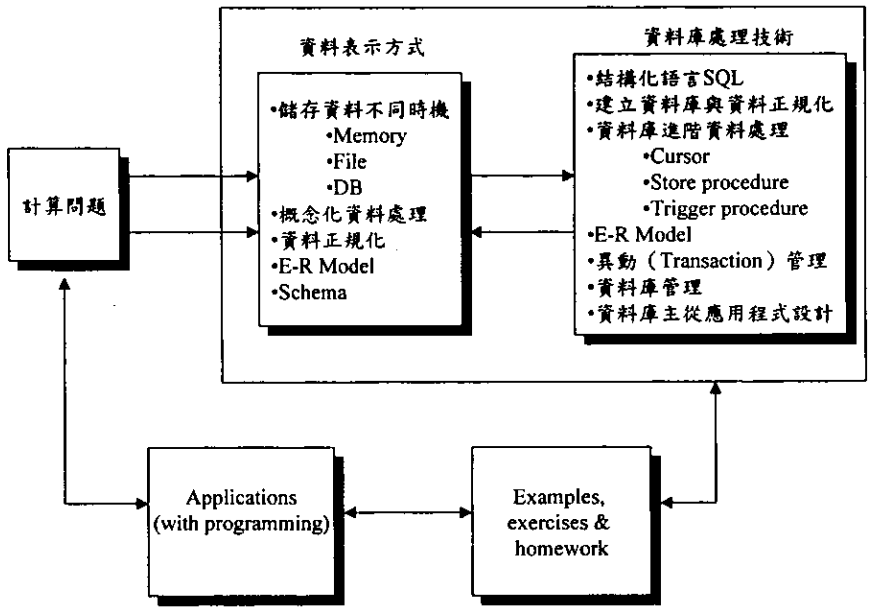
1. 訓練學生使用資料庫、建立資料庫及撰寫綱要（Schema）。
2. 學生瞭解使用資料庫時機，並設計應用程式使用資料庫。

課程內容：

(1) 資料庫系統基本觀念
(2) 結構化語言 SQL
(3) 建立資料庫與資料正規化
(4) 資料庫進階資料處理──Cursor、Store procedure、Trigger procedure
(5) 介紹使用 E-R Model
(6) 異動（Transaction）管理
(7) 資料庫管理
(8) 資料庫主從應用程式設計

設計構想與進行方式：

「資料庫系統」是當前程式設計師應該瞭解的系統；但是對系統使用方法的誤解，將使程式未受其利先受其害。由於資料庫系統的普及，甚至有免費的資料庫系統可以使用，參與的人員也越來越多，使得資料庫系統的程式開發變成是一個複雜的問題。因此如何有效地利用資料庫的技術管理資料，使資料處理更為效率，變成為一個重要的課題。



本課程強調的是程式設計師在資料庫系統應用程式開發過程中，應具備的知識與技能。希望學員完成此課程後，能夠在設計資料庫應用程式時，可以有效地利用資料庫的資料處理技巧，以最有效的方式完成設計程式的任務，解決問題。課程的基本精神在於讓學生從實作中驗證所學的技術，並從傳統的資料處理方式講起，一步一步向異質性資料庫的方向教學，希望本課程著重學生在實務上的應用，而不是在資料庫理論的教學。

本課程分成 12 個主題，共 58 小時：

Topic 1: 導論（4 hr）

首先介紹資料儲存及處理模式的演進，進而使學生瞭解檔案系統與資料庫系統的差異，並說明在什麼情況下使用資料庫系統會比使用檔案系統來的方便。同時介紹資料庫系統架構，它各層（Layer）之間的關係。

Topic 2: 資料庫系統的操作環境介紹（3hr）

介紹資料模式，主要說明放在關聯式資料庫系統上，使學生瞭解資料間的關連，進而介紹關聯表綱要及綱要與關聯表的關係。接下來說明主鍵、外來鍵在資料表中的重要性，並初略介紹資料庫的使用語法。

Topic 3: 結構化語言 SQL（I）—查詢部分(9hr)

從本單元開始詳細介紹 SQL 語法，這單元是以查詢部分。為了讓學生能充分瞭解 SQL 語法，我們將著重學生實作的部分。

Topic 4: 結構化語言 SQL（II）—異動部分(3hr)

介紹 SQL 語法中有關異動的部分。

Topic 5: 建立資料庫與資料表（6hr）

介紹建立資料庫及資料表 SQL 語法，並說明建立資料表的規劃流程、如何蒐集資料並轉換成欄位、資料的完整性、及資料庫的正規化，同時介紹資料字典及系統目錄。

Topic 6: 關聯式資料庫之進階資料處理（12hr）

介紹關聯式資料庫之進階資料處理技巧—Cursor、Store Procedure、Trigger Procedure、User Defined Function 等。

Topic 7: E-R Model（3hr）

介紹 E-R Model 的觀念、圖形符號及 E-R Model 與關聯式資料表的關係。

Topic 8: 檢視表（View）(3hr)

介紹檢視表的用途，使用檢視表的優點。同時說明如何建立、修改檢視表，在檢視表上的資料處理方式。

Topic 9: 異動管理（Transaction）管理（3hr）

介紹異動管理的目標及四個重要特性，當錯誤產生後，其回復處理的機制。最後介紹異動管理可能造成資料鎖定及鎖定死結問題。

Topic 10: 資料庫系統之管理（2hr）

介紹資料庫系統之使用者的種類及各使用者在資料庫系統上的主要工作。之後介紹資料庫管理系統所提供資料管理之功能，及善用索引（index）增加料庫系統查詢效率之方法。

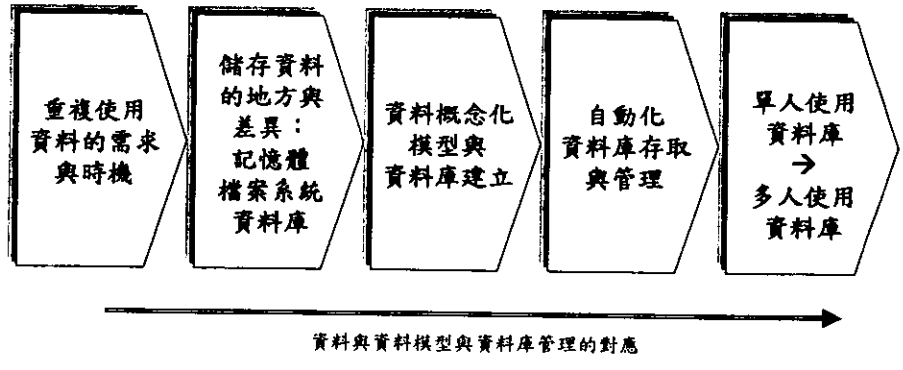
Topic 11: 發展主從式資料庫系統應用程式（6hr）

介紹主從式系統架構及主從式系統中資料庫系統如何被使用，在使用前如何設定環境變數，如何以 ODBC（JDBC）連結資料庫，並以實作驗證所學。

Topic 12: 其他資料庫系統介紹（3hr）

介紹資料模式及其他資料庫系統，如：分散式資料庫系統、物件導向資料庫系統。另外教育學生使其瞭解使用資料庫的時機及注意事項。

各單元進行的順序的關係如下圖所示：



資料與資料模型與資料庫管理的對應

7.3.2 課程大綱

Core Knowledge	Programming Skills	時數（小時）
1. 導論（3 小時） ● 資料儲存及處理模式的演進 ◆ 第一階段──紙本檔案人工處理方式 ◆ 第二階段──電腦化循序檔案處理方式 ◆ 第三階段──電腦化直接存取式檔案處理方式 ◆ 第四階段──以記錄為處理單元的資料庫系統 ● 檔案處理問題 ◆ 一人多檔處理 ◆ 多人多檔處理 ● 資料庫管理系統的需求與功能 ● 資料庫系統架構 ◆ 外部層（External Level） ◆ 概念層（Conceptual Level） ◆ 內部層（Internal Level） ◆ 各層之間的映對（Mapping）	● 設計一實作，讓學生瞭解在檔案系統中處理資料的不方便性： ◆ 一人多檔案同時處理資料 ◆ 多人多檔案同時處理資料 設計多個成績檔案，由學生練習設計程式，以瞭解使用檔案系統的問題。 ● 設計一實作，讓學生瞭解資料放在檔案、資料庫、或記憶體中的優缺點。 ● MS SQL Server 的安裝 ● MS SQL Server 的系統目錄	4
2. 資料庫系統的操作環境介紹 ● 資料模式簡介 ● 關聯表綱要簡介 ● 關聯表的內容 ◆ 認識關聯、Primary Key 與 Foreign Key ● 基本命令指令	● 讓學生瞭解在的概念： 1.利用的資料檔案，設計一個程式，以學號為 key 搜尋學生的各科成績。 2.設計一個程式，以學號為 key，可修改加退選功能。	3
3. 結構化語言 SQL（I）──查詢部分 ● 認識結構化語言 SQL ● SQL 語言的查詢用法 ◆ select 敘述的基本結構 ◆ select 子句 ◆ from 子句 ◆ Order by 子句 ◆ 聚合函數 ◆ Group by 子句 ◆ Having 子句	● 啟動 MS SQL Server ● 設定環境 ● 將成績資料轉換成資料表後，可讓學生做以下的實作： 1. 搜尋 2. 排名次 3. 算學期成績 4. 找出男、女生成績最好者 使學生瞭解結構化語言 SQL 中查詢部分的用法	9

◆ 巢狀式查詢 ◆ 聯集（Union）		
4. 結構化語言 SQL（II）——異動部分 ● SQL 語言的異動用法 ◆ Insert 敘述 ◆ Update 敘述 ◆ Delete 敘述 ◆ Select into 敘述	● 延續上個資料庫，請學生實作下列動作： 1. 修改成績 2. 加/退選學生名單 3. 學生調班 使學生瞭解結構化語言 SQL 中異動部分的用法	3
5. 建立資料庫與資料表（6 小時） ● 資料定義語言 ◆ Create DataBase 敘述 ◆ Alter DataBase 敘述 ◆ Create Table 敘述 ◆ Alter Table 敘述 ◆ Drop Table 敘述 ◆ Truncate Table 敘述 ● 規劃關聯式資料庫 ◆ 簡易的規劃流程 ◆ 蒐集資料並轉換成欄位 ◆ 資料的完整性 ◆ 資料表關聯的種類 ◆ 資料庫的正規化 ➢ 第一正規化 ➢ 第二正規化 ➢ 第三正規化 ◆ 資料字典/系統目錄	● 請學生設計圖書館借書系統所需要的資料庫系統： 使學生瞭解規劃關聯式資料庫及資料庫與資料表的產生。	6
6. 關聯式資料庫之進階資料處理 ● Cursor ◆ Cursor 簡介 ◆ Cursor 的宣告、開啟、關閉、與移除 ◆ 透過 Cursor 修改或刪除資料 ◆ 使用 Cursor 的技巧 ● Store procedure ◆ Store procedure 簡介 ◆ Store procedure 的建立、使用、與修改 ◆ 設計 Store procedure 的技巧 ● Trigger procedure ◆ Trigger procedure 簡介 ◆ Trigger procedure 的建立與	● 使用權限的授與、取回 ● 預儲程序與觸發程序的設計： 在圖書館借書系統的資料庫中建立 Cursor、Store procedure、Trigger procedure、自訂函數。	12

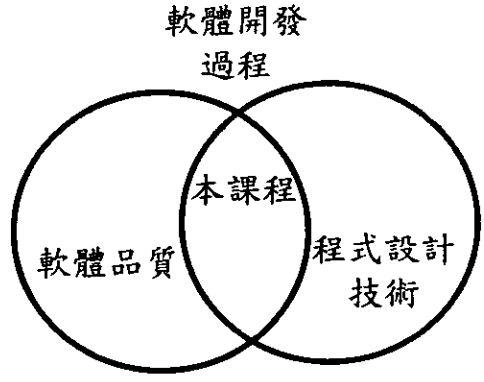
修改 ◆ 設計 Trigger procedure 的技巧 ● 自訂函數 ◆ 自訂函數的特色 ◆ 自訂函數的建立、使用、與修改 ◆ 自訂函數的使用技巧		
7. E-R Model ● E-R Model 的觀念 ● E-R Diagram ● E-R Model 與關聯式資料表的關係	● 畫出之前所設計之圖書館借書系統資料庫的 E-R Model。	3
8. 檢視表（View） ● 檢視表（View）簡介 ● 檢視表（View）上的資料處理 ● 檢視表（View）的優點 ● 建立檢視表（View） ◆ 檢視表的用途 ◆ 檢視表的建立、修改	● 利用之前所設計之圖書館借書系統資料庫建立暫時需要的檢視表（view），如: 1. 借書未還的檢視表 2. 出借書籍中所有原文書所組成的檢視表	3
9. 異動（Transaction）的管理 ● 異動管理的目標 ● 異動的四個特性 ● 回復（Failure Recovery）處理 ● 異動的並行控制 ● 資料鎖定及鎖定死結問題	● 在圖書館借書系統資料庫中，請學生實作下列動作： 1. 借書。 2. 還書。 3. 查詢。 並在這些動作中加入異動管理，利用 SQL 語法來模擬多人同時使用資料表時異動的管理	3
10. 資料庫系統之管理 ● 資料庫系統之使用者 ◆ End-User ◆ Application programs ◆ DataBase Administrator ● 資料庫管理系統的基本功能 ◆ 資料定義 ◆ 資料處理 ◆ 資料安全 ◆ 資料備份 ● 善用索引（index）加快查詢效率 ◆ 索引介紹 ◆ 索引的種類 ◆ 索引的處理 ◆ 建立索引的注意事項	● 使用權限的授與、取回 ● 資料庫系統的備份與還原	3
11. 發展主從式資料庫系統應用程式	● 設計一個主從式的應用程式，他需	6

● 主從式架構簡介 ● 存取各種資料庫的準備工作與觀念 ● 環境設定 ● 以 ODBC（JDBC）連結資料庫 ● ODBC（JDBC）API 介紹及其使用	用利用 ODBC（JDBC）連結資料庫，而應用程式可分為： 1. standalone。 2. Web application。 3. MS Access	
12. 其他資料庫系統介紹 ● 資料模式介紹 ◆ 階層式資料模式 ◆ 網路式資料模式 ◆ 關聯式資料模式 ● 分散式資料庫系統簡介 ● 物件導向資料庫系統簡介 ● 使用資料庫時機及注意事項	●	3
Total:		58

7.4 軟體發展技術

7.4.1 課程設計理念

前言：今日市面的應用軟體功能強大，數十至數百萬行的程式比比皆是，需告團隊合作方可在指定期間內完成，因此有必要訓練學生熟悉軟體文件標準及寫作技巧、軟體測試與除錯技術等等，以培育其製作高品質的大型軟體模組能力。。



教育目標：

1. 使學生了解軟體生命周期、及軟體工程之由來、目的及範圍。
2. 使學生瞭解軟體品質的意義與指標，與提升軟體品質的一些技術。
3. 訓練學生熟悉程式偵錯與測試技術，及運用進階的(OOP)程式開發技術開發複雜軟體。
4. 訓練學生使用 UML 製作軟體文件的能力。
5. 訓練學生瞭解軟體發展各階授工作內容與軟體製作階段之所需技能。

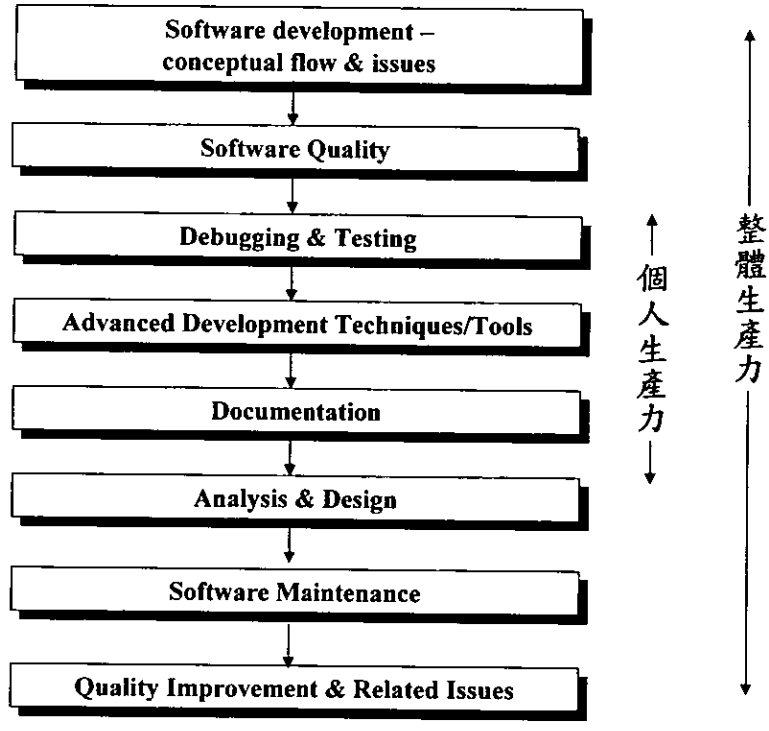
課程內容：

(1) 軟體品質基本觀念與軟體開發相關程序
(2) 軟體工程概論
(3) 軟體測試策略與技術
(4) 軟體文件種類與製作
(5) UML
(6) 軟體分析，設計維護及其他相關議題

設計構想與進行方式：

本課程強調的是程式設計師在整個軟體系統開發過程中，所扮演的角色及其相對應具備的知識與技能。希望學員完成此課程後，能夠在設計程式時，可以兼顧軟體品質，以最有效的方式完成設計程式的任務，解決問題。希望微觀(micro view)的從單一程式設計師設計單一程式的經驗與軟體品質的要求出發，進而涵蓋宏觀(macro view)的整個軟體系統的開發與品質，來強調程式設計師、軟體品質、軟體發發技術與工具的關係。課程的基本精神在於讓

學生在做中學(learning by doing)。課程進行時以學生以前設計開發的程式為對照經驗，用以驗證各單元的內容。並適度加入專題或較大型的程式系統開發，以強化學習成效。



本課程分成 8 個單元，共 48 小時：

Unit 1: Software development – conceptual flow & issues（3 hr）

- 簡介軟體系統開發過程，從單一程式設計師設計單一程式到多人參與的大型開發案可成會遭遇的問題，引導學生瞭解軟體工程的需求與核心問題，並從中瞭解程式設計師的定位。

Unit 2: Software Quality （3 hr）

- 介紹軟體品質的定義及需求，並介紹影響軟體品質不良的可能環節。同時請學生用本單元的經驗分析以前自己開發的程式的軟體品質。

Unit 3: Debugging & Testing （9 hr）

- 介紹經由 debugging 與 testing 的方式來強化軟體品質的過程與技巧。介紹各種可以提升程式設計師個人生產力的 debugging 技巧與工具，及軟體進行 testing 時的各種要求與標準。同時請學生用本單元的技巧與準則分析以前自己開發的程式的軟體品質。

Unit 4: Advanced Development Techniques/Tools （9 hr）

- 介紹數種可以提升程式設計師個人生產力的程式設計技巧與工具，例如新的技術標準(如 C#, .Net, OOP, XP 等)。強調 OOP 的進階技術，以銜接以前程式設計課程。同時請學生用本單元的技巧與準則進行幾個相關的練習。

Unit 5: Documentation （12 hr）

- 介紹文件撰寫在軟體開發工程中扮演的重要性，並特別強調與程式設計師在開

發程式的過程中有關的文件製作。利用 UML 此一工具，讓學生用本單元的技巧與準則進行以前自己開發的程式的文件撰寫。

Unit 6: Analysis & Design（6 hr）

- 簡介軟體系統的設計與分析過程。利用 UML 讓學生學會系統分析與設計相關的工作產出（如 use cases, sequence diagram, collaboration diagram, etc.）。

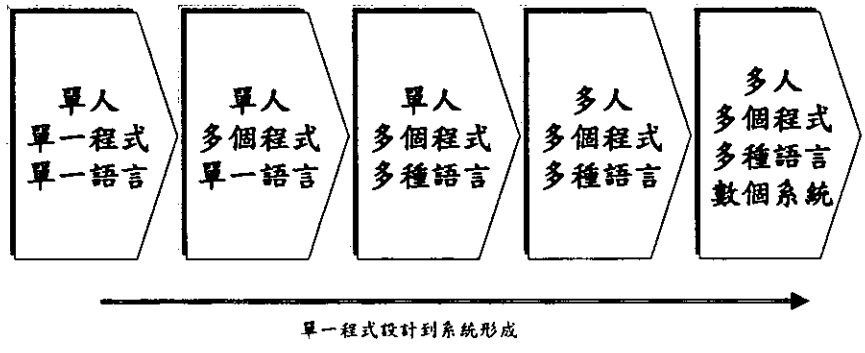
Unit 7: Software Maintenance （3 hr）

- 介紹軟體系統(或程式)完成後如何維護，內容涵蓋的是與程式設計師在開發時應該注意的相關知識與技巧。介紹軟體維護的工作種類與內容及管理程序等，如何將軟體設計的易於安裝、維護與更新，並介紹例如 CVS 此類的軟體版本控制方式。同時請學生用本單元的技巧與準則調整以前自己開發的程式的安裝與管理方式。

Unit 8: 軟體品質改善相關議題（3 hr）

- 由軟體系統開發的不同階層（個人、工作小組、組織）的觀點簡介軟體開發程序與軟體品質改善相關議題，如 PSP、TSP、CMMI 等。

各單元進行的順序的關係如下圖所示：



7.4.2 課程大綱

Core Knowledge	Programming Skill Correspondence	Hours
1. Software development – conceptual flow & issues ■ Concept & Definition ■ Software crisis (not on schedule, meet budget and requirements) ■ A programmer with one (big) program → a team of programmers with one big and complicated software system ◆ The need for SOP (standard operating procedure) for software development ◆ The needs of Documentation, process, management (of people and software) Why? ■ Life-cycle of software development ◆ Requirement phase, Specification phase, Design phase, implementation phase, test & integration phase, maintenance phase, retirement phase ◆ The tasks performed in each phase ◆ The requirement and possible errors in each phase ■ Software engineering: ◆ Why? ◆ definitions, scope ◆ the needs ◆ objective: method→process→tool ■ The role of programmers in software engineering ■ How/Why can SE improve the productivity of programmers and the quality of software?		3
2. The concept of software quality (SQ) ■ Concept, Definition & examples ■ How to justify & evaluate the quality of software ◆ Several indicators ◆ By one programmer v.s. by a team of programmers ◆ The flow/process that effect SQ ◆ The factors that effect SQ in the development of software by programmers ■ SQA(Software Quality Assurance) and V&V ■ SQA and V&V—methods and evaluation	■ Define and evaluate the quality of the programs that you have ever done.	3

■ For programmer: bug-free; debugging & testing ■ For software: meet the requirements ◆ Code review vs SQA: why & how ◆ Documentation vs SQA: why & what & how		
3. Debugging & Testing ■ Debugging: Concept, Definition & examples ◆ 除錯技術：Bugs Management 概念、除錯方法、除錯工具之觀念與使用 ◆ Bug finding/allocation ◆ Bug management ◆ Bug fixing ---------- (3hr) ■ Testing: ◆ 軟體測試功能 ◆ 軟體測試種類：功能性測試、模組化測試、安全性測試，整合測試（Graph-based 測試, etc.）、等價劃分、邊界值分析、比較測試、GUI 測試，主從系統測試、即時系統測試 ◆ 軟體測試策略：白箱測試、黑箱測試 ◆ 軟體測試步驟：測試計畫之撰寫、測試計畫之執行，單元測試、整合測試、驗證測試、系統測試 ---------- (4hr) ■ Test cases ◆ Generation & Preparation of test cases: how and where ◆ Correctness/Completeness/soundness ◆ 測試文件：測試計畫文件、測試案例 ---------- (3hr)	■ According to your programming experience ◆ Summarize the types, reasons, and solutions of bugs that you ever met ◆ List the debugging techniques that you ever did ◆ Analyze the effectiveness of new debugging techniques when being applied to your programs ◆ Test the programs that you have ever developed	9
4. Advanced Development Techniques/tools ■ 模組化程式設計 ■ 程式設計程序、模組分割、元件化程式設計 ■ New (latest) technologies in programming/system development ◆ OOP、XP、C#、.net、etc. ---------- (2hr) ■ Advanced techniques in OOP ◆ 物件導向的進階技術：more about 類別、物件與屬性、套件與介面、關聯、繼承與聚集、多形與連結 ◆ Polymorphism ---------- (2hr) ◆ Method Overriding/Overloading ---------- (2hr)	1-2 Homework Assignments	9

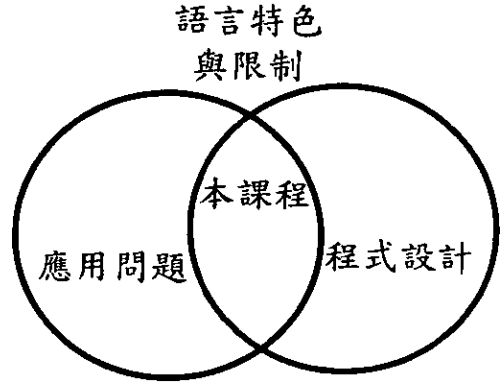
◆ Java Beam ---------- (1hr) ◆ N-tier applications with Java ---------- (2hr)		
5. Documentation (especially the ones related to programming by one or many programmers) ■ The needs, concept & definition, types of documentation ■ Documentation for development, management, and usage & deployment of software →強調與 programmer 有關的 ■ 虛擬碼、Coding Convention ■ 軟體文件種類、軟體文件格式 ---------- (3hr) ■ UML-based documentation： ◆ Concept, Definition & examples ◆ Basic notations in UML ---------- (6hr) ◆ Write your own and refer to others ◆ 使用案例圖、類別圖與物件圖、順序圖與合作圖、狀態圖與活動圖、元件圖與佈署圖、界面、合作、擴充機制、框架與樣式 ---------- (3hr) ■ ISO/IEC 12207, 15504 文件規範簡介	■ Denoting the programs that you have ever developed in UML (all documents related to programmer) ■ Read the documents of your partner ■ Documentation for your programs 實作：**Rational Rose** UML for existing software	12
6. Analysis & Design ■ Concept, Definition & examples ■ Analysis: 軟體系統開發人員應具備之軟體系統分析相關議題 ■ 軟體分析的工作內容 ■ 需求分析、領域分析 ■ UML for software analysis ■ 物件導向分析方法：情節為主之分析方法、使用案例之用法、使用案例規格、類別圖之推導，sequence diagram, collaboration diagram 之推導 ■ Design: 軟體系統開發人員應具備之軟體系統設計相關議題 ■ 軟體設計的工作內容--架構設計、元件設計、細部設計 ■ UML for software development (設計規格書：設計規格書項目、設計規格書撰寫) ■ Corresponding to Unit-5: The needs of documentation (architecture, components, details)	Sequence Diagram Collaboration Diagram UML for requirements of the outsides	6
7. Maintenance of software ■ Concept, Definition & examples ■ 軟體維護	■ According to your programming experience ◆ Maintenance	3

◆ 種類與步驟 ◆ 軟體可維護性、軟體維護程序、軟體維護評量、軟體維護計劃、軟體維護管理、軟體維護記錄 ■ 軟體組態管理 (configuration management)： ◆ 軟體組態程序管理、軟體組態識別、軟體組態控制、軟體組態稽核、軟體組態報告、軟體交付管理 ■ CVS(Current Versions System)	◆ Configuration management ◆ Upgrading & version control	
8. 軟體品質改善相關議題 ■ Concept, Definition & examples ■ 階層式軟體品質改善 ■ A person→group/team→organization: different requirements and methods ■ Personal: PSP(Personal Software Process) ■ Grouped: TSP(Team Software Process) ■ Organizational; CMMI(Capability Maturity Model Integration) ■ 軟體工程的未來趨勢		3
Total: 3*16=48		48

7.5 程式語言

7.5.1 課程設計理念

前言：學生已熟悉數種程式語言，且可運用這些語言設計程式，唯欲善用程式語言設計優質程式，實有必要程式語言的由來、類別及各自的弱點，進一步可自我學習新的程式語言，比較此新語言與其他語言之差異，使能依問題應用領域之不同而選擇適當語言。



教育目標：

1. 使學生了解程式語言之類別及應用範圍
2. 使學生了解各類程式語言演進的緣由，特性，及弱點
3. 訓練學生具有自我學習新程式語言的能力

課程內容：

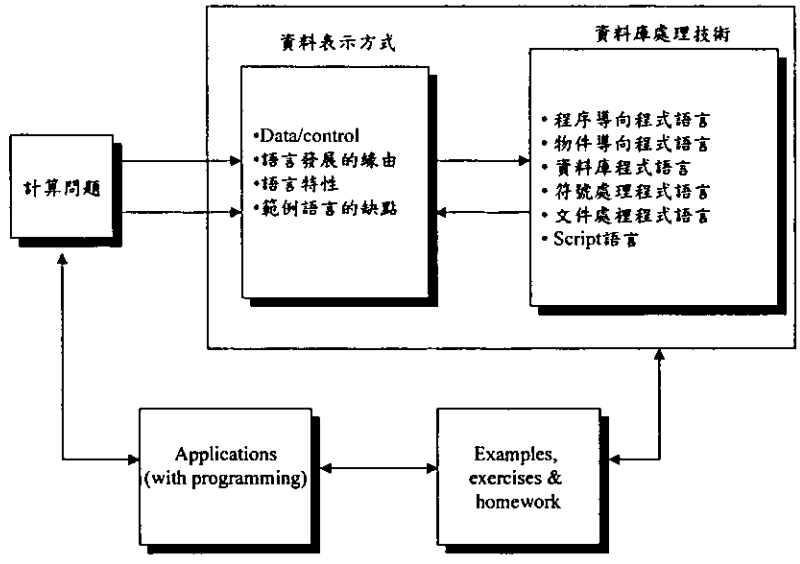
1. 程序導向程式語言
2. 物件導向程式語言
3. 資料庫程式語言
4. 符號處理程式語言
5. 文件處裡程式語言
6. Script語言

設計構想與進行方式：

課程的進行依語言特性分成：程序導向程式語言；物件導向程式語言；資料庫程式語言；符號處理程式語言；文件處裡程式語言；Script語言。每一類語言依其發展之時間由教師理論授課，而每一語言課程授課的流程：

1. 先強調語言發展的緣由：目的讓學生了解影響該程式語言演進之主要因素，如應用之影響、程式結構之影響、計算機架構之影響、程式撰寫的方便性之影響、軟體設計模式之影響。
2. 語言特性：目的讓學生了解該程式語言的一些重要性質。
3. 舉一範例：以實例說明，讓學生榮一體會與了解。

4. 語言的缺點：說明該程式語言之缺點及其使用注意事項。



本課程共分成 7 個主題，共 48 小時，各主題說明如下：

單元一、程式語言之目的，分類與演進(3hr)

1. 導論
2. 程式語言的目的
3. 程式語言的類別、性質與主要應用
4. 程序導向程式語言
5. 物件導向程式語言
6. 資料庫程式語言
7. 符號處理程式語言
8. 文件處裡程式語言
9. Script語言

單元二、程序導向語言(Procedure-Oriented Language)（16hr）

1. 背景—機器語言與組合語言
2. 機器語言
3. Von Neumann Machine
4. 組合語言
5. FORTRAN
6. COBOL
7. BASIC
8. PASCAL
9. C
10. Ada
11. VB

單元三、物件導向語言（5hr）

1. small talk
2. C++
3. Jave

單元四、資料庫語言（5hr）

1. SQL, SQL1
2. SQL2
3. SQL3

單元五、符號處理程式語言（4hr）

1. LISP
2. Prolog

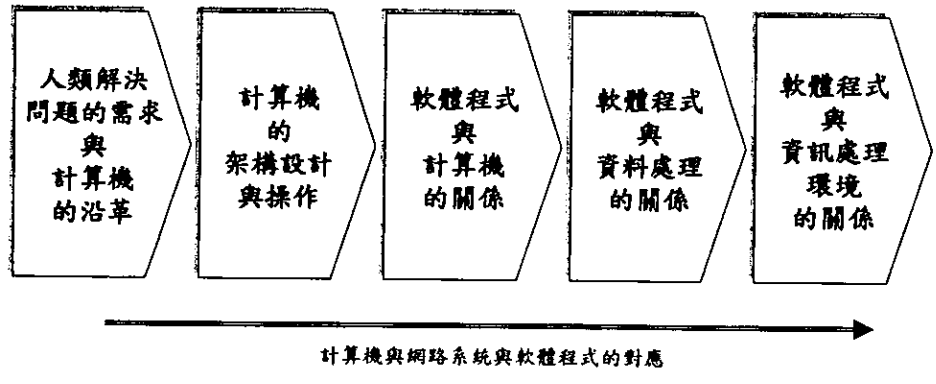
單元六、文件處裡程式語言（7hr）

1. Tex
2. Postscript
3. Latex
4. PDF

單元七、Script語言（8hr）

1. Perl
2. Tcl/tk
3. Python
4. Ruby
5. PHP

各單元進行的順序的關係如下圖所示：



7.5.2 課程大綱

Core Knowledge	Programming skills/LAB	時數
單元一、程式語言之目的，分類與演進 2. 導論 ■ 程式語言的目的 ■ 程式語言的類別、性質與主要應用 ◆ 程序導向程式語言 ◆ 物件導向程式語言 ◆ 資料庫程式語言 ◆ 符號處理程式語言 ◆ 文件處裡程式語言 ◆ Script 語言 ■ 語言的執行 ◆ compiler ◆ assembler 3. 影響程式語言演進之主要因素 ■ 應用 ■ 程式結構 ■ 計算機架構 ■ 程式撰寫的方便性 ■ 軟體設計模式 4. 程式語言的發展史		3
單元二、程序導向語言(Procedure-Oriented Language) 1. 背景—機器語言與組合語言(1956) ■ 機器語言(1 hr) ■ Von Neumann Machine ■ 動機 ◆ ■ 語言特性 ◆	● 用 JAVA 語言來模擬 FORTRAN 的 Static storage ● 用 JAVA 語言來模擬 COBOL 的報表輸出 ● 用 JAVA 語言來模擬 PASCAL 的 link list ● 用 JAVA 語言來模擬 C 的 low level control ● 用 JAVA 語言來模擬 Ada 的	16

■ 範例 ■ weakness ◆ 2. 組合語言(1956) ■ 動機 ◆ 提高程式可讀性、Relocatable Program Concept ■ 語言特性 ◆ Operation Code、Register 名稱用符號代替 ◆ 變數名稱取代位址 ◆ 巨集指令(Macro) ■ 範例 ■ weakness ◆ Relocatable Loader 之導入 — 程式起始位置之變更 3. FORTRAN(1957) ■ 動機 ◆ 近似人常用的自然語言— 第一個高階語言 ◆ Formula Translation ■ 特性 ◆ Declaration statements, execution statements ◆ Simple variables, array variables ◆ Assignment statements, control statements, I/O statements, subroutine ◆ Statements, FORMAT statements ◆ Static storage ◆ Automatic type conversion ◆ Library ◆ Comment ■ 範例 ■ Weakness ◆ Static storage ◆ No recursive feature ◆ FORMAT statements	concurrent	

- ◆ Difficult to handle input/output

4. COBOL (1959)

- ■ 動機
  - ◆ Business data processing
  - ◆ Report, numeric
  - ◆ Fortran 無法處理 string, file
- ■ 特性
  - ◆ 程式看起來很像是英文的口語
  - ◆ manipulate input/output file
  - ◆ divides programs into: identification, environment, data, procedure divisions.
  - ◆ 提供 records 的資料結構
- ■ 範例
- ■ Weakness
  - ◆ 無結構化

5. BASIC(1964)

- ■ 動機
  - ◆ Students easy to learning
  - ◆ Fortran, COBOL 變數一定要宣告
  - ◆ Fortran, COBOL 語言 compiler 如有錯很難除錯
- ■ 特性
  - ◆ Var. cannot be declared
  - ◆ Interpreter
  - ◆ Friendly error messages
  - ◆ GoSub…Return
  - ◆ Respond fast for small programs
- ■ 範例
- ■ Weakness
  - ◆ 無副程式參數傳遞的觀念
  - ◆ Interpreter (slow: 每次執行須重新編譯)

6. PASCAL (1970)

- ■ 動機

- ◆ Fortran, COBOL, BASIC 語言無結構化概念
- ◆ 之前的語言無 dynamic (pointer), record 的資料結構

■ 特性

- ◆ Variables must be defined
- ◆ 提供 dynamic (pointer)的資料結構
- ◆ well-structured
- ◆ 提供 Sets, files, records 的資料結構

■ 範例

■ Weakness

- ◆ One pass compiler--procedures 要依順序寫(被呼叫要寫在前)

7. C (1971)

■ 動機

- ◆ 之前的語言採 one pass compiler
- ◆ A general-purpose programming language
- ◆ For UNIX
- ◆ 之前的語言少提供 library routines

■ 特性

- ◆ access to low level hardware
- ◆ library routines
- ◆ UNIX system
- ◆ weak type check

■ 範例

■ Weakness

- ◆ flexible but dangerous

8. Ada (1970)

■ 動機

- ◆ 美國防部所制定
- ◆ 之前的語言無 concurrent 的能力
- ◆ 之前的語言無結構化概念
- ◆ 支援 embedded and real-time systems

■ 特性

- ◆ concurrent (parallel)

◆ semaphores ◆ bounded buffer ◆ structured ◆ run-time checking ■ 範例 ■ Weakness 9. VB (1991) ■ 動機 ◆ 之前的語言處理視窗介面難 ◆ 之前的語言與資料庫之間難連結 ■ 特性 ◆ rapid development of GUI for Windows ◆ complex database object library ◆ network-savvy ◆ multithreaded ■ 範例 ■ Weakness ◆ 程式碼大		
單元三、— 物件導向語言 1. Small talk (1969) ■ 動機 ◆ object-oriented ◆ 跨平台 ■ 特性 ◆ first OOP ◆ Everything is an object ◆ Everything is available for modification ◆ Types are dynamic ◆ Bytecodes ◆ Multithreaded ◆ Concurrency ◆ 繼承（不支援多重繼承） ■ 範例	● Java Applet	5

- Weakness
  - garbage collection 不完整
  - Bytecodes--執行速度慢

2. C++ (1983)
- 動機
  - C + O.O.
- 特性
  - Compatible with C
  - 多重繼承的觀念
  - Strict compile-time check
  - Template
  - Exception handling
- 範例
- Weakness
  - Single-threaded
  - 無 garbage collection
  - 多重繼承的觀念-- 一個物件的兩個父物件可能有衝突

3. JAVA (1995)
- 動機
  - independent of the host platform
  - for networking
  - securely
- 特性
  - portable
  - bytecodes
  - 完整的 garbage collection
  - network-savvy
  - multithreaded
  - Class library
- 範例
- Weakness
  - Bytecodes--執行速度慢

單元四、— 資料庫語言 ■ SQL(1970), SQL1(1989) ■ 動機 ◆ 針對關聯式資料庫發展的語言 ■ 特性 ◆ Data definition in SQL-- table definition, schema definition and update, user defined domain, relational constraints, relational catalogues ◆ SQL queries -- selection, insertion, deletion, update ■ 範例 ■ Weakness ◆ 缺少 Changing security settings ◆ syntax rules, create queries 較不方便 ■ SQL2(1992) ■ 動機 ◆ Enhance SOL1 ■ 特性 ◆ new reserved words, wildcard ◆ GRANT, REVOKE ◆ DISTINCT ◆ LIMIT TO ■ 範例 ■ Weakness ◆ 無法處理 Multimedia ■ SQL3(1999) ■ 動機 ◆ applications 的需求改變--multimedia ■ 特性 ◆ for objects-oriented ◆ for data-mining ◆ for spatial-data	●	5

◆ for temporal-data ◆ for on line analytical ◆ for data-warehousing ◆ for multimedia-data ◆ 將程式區分成兩部 – 1. core specification: for RDBMS, 2. option specialized packages: for applications ■ 範例 ■ Weakness		
單元五、—符號處理程式語言 ■ LISP(1958) ■ 動機 ◆ 人工智慧計算的需求 ■ 特性 ◆ rule-based ◆ lumda-calculus & functional computation ◆ list as data and program ◆ dynamic data typing (late binding) ◆ symbolic computation ◆ focused on problem solving strategies ◆ suitable for AI applications (lower level) ◆ fast prototyping ◆ stack-based interpretation ■ 範例 ■ Weakness ◆ some data is not suitable to be described as lists or functions ◆ procedure-like computation ◆ still too –lower for AI applications ◆ slow interpretation (compared with C) ◆ large memory consumption ■ Prolog(1970) ■ 動機	●	4

◆ 單一一種程式計算方式（logic interpretation）： 以 logic clauses 處理所有運算並表示所有資料型態 ■ 特性 ◆ data=program; logic clauses as data and program ◆ dynamic data typing (late binding) ◆ symbolic computation ◆ focused on problem solving strategies ◆ suitable for AI applications (higher level) ◆ fast prototyping ◆ Interpretation ■ 範例 ■ Weakness ◆ slow interpretation (compared with C) ◆ huge memory consumption		
單元六、—文件處裡程式語言 ■ Tex(1970) ■ 動機 ◆ 製作排版文件，處理數學公式與各種符號。 ■ 特性 ◆ 制訂排版指令 ◆ 以寫程式方式進行排版與文件製作（文件＝程式） ◆ 編譯式語言 ◆ 分離文件製作與輸出 (tex， dvi， ps) tex→dvi→ps→… ◆ 可跨平台 ■ 範例 ■ Weakness ◆ 不容易記憶各種排版指令 ◆ 需要進行編譯(compilation) ◆ 需安裝字型與相關函式庫		7

- Postscript(1976)
  - 動機
    - 於各種終端機顯示文件，由各種印表機輸出文件。增加文件的可攜性
  - 特性
    - 制訂螢幕輸出與印表機輸出指令
    - 分離文件製作與輸出
    - 提供各種跨硬體的文件可攜性
    - interpretation
    - 提供多種轉換成 PS 格式的工具
  - 範例
  - Weakness
    - 文件檔案龐大
    - 需安裝字型與相關函式庫

- Latex(1985)
  - 動機
    - tex 指令太多，不容易使用。LaTeX 於 tex 中加入巨集，並提供各種可重複套用的版形。
  - 特性
    - 同 tex，但加入巨集，並提供各種可重複套用的版形
    - 使用者需記憶的指令較少
  - 範例
  - Weakness
    - （仍然）不容易記憶各種排版指令
    - 需要進行編譯(compilation)
    - 需安裝字型與相關函式庫

- PDF
  - 動機
    - PS 型態文件檔案太大，不利於流通於網路上。PDF 提升網路流通的可攜性文件。
  - 特性
    - 同 PS

◆ 文件較精簡，SIZE 較小 ◆ 可嵌入 browser ◆ 提供多種轉換成 PDF 格式的工具 ■ 範例 ■ Weakness ◆ 需安裝字型與 drivers		
單元七、—Script 語言 1. Perl(1987) ■ 動機 ◆ 對問題中的文字字串進行快速、複雜的文字處理。Perl 是目前 Script Language 中最熱門的語言，由於一開始結合了 C， awk，sed， sh， and BASIC，等語言的特性，因此在 Script Language 引起十分廣泛的回響，特別是在 Unix-like 的作業系統上，傳統的網管與系統管理的工作，八成幾乎主要使用 Perl 語言來完成，另外 Perl 社群亦建立 Comprehensive Perl Archive Network (CPAN).來統一管裡各個模組，因此 Perl 成為支援性最好的 Script Language. 然而 Perl 亦繼承了 sed/awk 的負擔，整個語言的設計十分不易閱讀，程式設計師，往往一個月後，無法看懂之前所寫的程式，換言之，程式的可擴充性相對較不易。 ■ 特性 ◆ regular expression ◆ Perl takes features from other languages， such as C， awk， sed， sh， and BASIC， among others. ◆ Perls database integration interface (DBI) supports third-party databases including Oracle， Sybase， Postgres， MySQL and others. ◆ Perl works with HTML， XML， and other mark-up languages.	●	8

- Perl supports both procedural and object-oriented programming.
- Perl interfaces with external C/C++ libraries through XS or SWIG.

■ 範例

■ Weakness

- 執行效率較差
- Readable is much poor!

2. Tcl/tk(1988)

■ 動機

- 對電腦各種（周邊）設備進行處理，給予程式設計師以單一程式語言處理所有電腦相關設備。

■ 特性

- EDA/CAD: Tcl has become the defacto standard in EDA and CAD applications.
- Test Automation: Tcl is the de facto standard for automated testing.
- Dynamic Web Content: Some of the industry's most popular high-traffic web sites are Tcl-powered.
- Network and System Management: Tcl provides a platform for network and system management applications with its rapid GUI development， easy extensibility and cross-platform support.

■ 範例

■ Weakness

- The main advantage of Tcl is that *all* its variables are strings
- The main advantage of Tcl is that it has no syntax.

3. Python(1991)

■ 動機

◆ Python 係針對 Perl 的程式碼不易閱讀與不易擴充問題，重新精簡了程式的元素，例如利用縮排來取代程式區塊，去除資料型別的概念，並將物件導向觀念引進，因此相對於 Perl 來說，程式碼顯得易懂且易於擴充。以下就 Python 在幾個特色表現優異之處，一般來說 Python 的 Advanced data structures (> Tcl， Perl)、Python 的 Powerfull data-parallel arrays (> Tcl)、Readbility and modularity (> Perl)、 High-level (> C， C++， Fortan) Platform independence (> C++， Java)。

■ 特性

◆ Interpreted， High level， Object-Oriented

◆ Flexible and Extensible

◆ Introspection， self-documenting

◆ Platform Independent

◆ Open Source

◆ Rapidly gaining acceptance

■ 範例

■ Weakness

◆ The main disadvantage of python (apart from not enough people knowing about it) is that it is interpreted from bytecode and therefore executes slower than C++.

4. Ruby(1993)

■ 動機

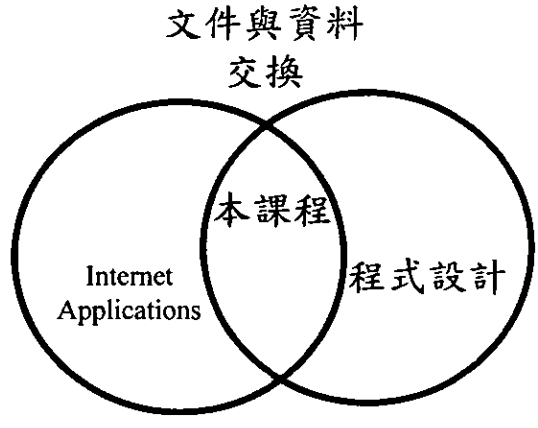
◆ Ruby(紅寶石)比起其他的 script language 來說，最大的特色來自於這個語言設計時特別強調完整的物件導向概念，除與 Perl 一樣，可用來做系統管理工具，另外也有正規表示式與強大的檔案和字串處理能力，但是確沒有像 Perl 一般不易閱讀的語法，比起 python 來說更 OO， 更有彈性， 因為 Ruby 全都是 OO.。

■ 特性 ◆ simpler syntax ◆ redefinable methods. ◆ Complete, full, pure object oriented language: OOL. ◆ OO in Ruby is carefully designed to be both complete and open for improvements. ◆ single inheritance only， *on purpose*. ◆ true closures. Not just unnamed function， but with present variable bindings. ◆ Ruby features a true mark-and-sweep garbage collector. ◆ needs no variable declarations. ◆ OS independent threading. ◆ highly portable: ■ 範例 ■ Weakness 5. PHP(1995) ■ 動機 ◆ C-like syntax, suitable for web。 ■ 特性 ◆ interpretation ◆ 跨平台 ◆ fast prototyping ◆ connect to DB ◆ widely adpoted ■ 範例 ■ Weakness ◆ 需安裝相關函式庫 ◆ 執行效率較差		
	合計	48

7.6 XML 語言

7.6.1 課程設計理念

前言：隨著網際網路的普及，人們已習慣將將資訊利用標記語言來呈現，但 HTML 標記本身有使用上的限制，因此全球資訊網路協會（W3C）統一網路科技的標準，並制訂 XML（Extensible Markup Language）延伸標記語言。XML 文件之「結構化」與「資訊內容導向化」二特色，特別是用在電子資料的傳遞、交換與分享，有助於原始資料的回溯、處理，避免不必要的重複，因而有效地提升其效率。另外由於 XML 具有「可延伸」和「跨平台」的特性，可促進各個不同的應用領域發展出各自的文件標準，以利資訊交換。本課程將介紹 XML 的應用。

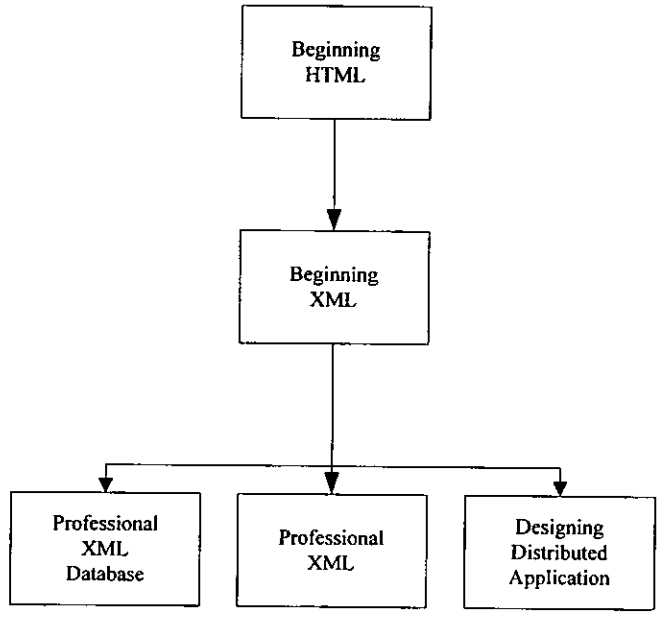


教育目標：本課程的重點在於 XML 的應用發展，包括在客戶端處理 XML、在伺服器端處理 XML，以及使用 XML 來處理資料等。

課程內容：

(1) 網際網路（internet）及全球資訊網（WWW）簡介
(2) 超文件標記語言（HTML）介紹
(3) 串接樣式表（Cascading Style Sheet -- CSS）介紹
(4) 以 XML 建立標記
(5) 驗證 XML 文件
(6) 可擴充樣式表語言轉換（XSLT）
(7) 客製標記語言（Custom Mark Language）
(8) 文件物件模組（Document Object Model, DOM）
(9) XML 的簡易應用程式介面（SAX）
(10) XML 與資料庫
(11) Web-Based 系統
(12) 簡易物件存取協定（SOAP）
(13) B2B 與微軟 BizTalk 伺服器

設計構想與進行方式：超文件標記語言（Hyper-Text Markup Language）是處理網際網路資料之語言，但其有本身上的限制。這讓 W3C 想要設計一個有「可延伸」和「跨平台」特性的標記語言──XML。因為 XML 優異的特性，使得 XML 的應用不斷的增加。為了讓學生盡快瞭解 XML 的語法及其應用，我們將本課程分成三個部分。第一部份介紹什麼是標記語言。在這裡我們將從 HTML 開始，使學生先瞭解標記語言。第二部分介紹 XML 語言及如何驗證 XML 文件，如何將 XML 文件顯示在網頁上。第三部分介紹 XML 應用的範圍，如何將 XML 檔案當作資料庫使用到在分散式系統上 XML 的應用。本課程共五十二個小時，適合在一學期開課。如果因為課程時數的限制，且在其他課程已經講述 HTML 的概念，則可從 Topic3 開始到 Topic6 約十二個小時。第三部分是選擇性的，各老師可選擇重要部分教授及延長教授的時間。整個課程的架構如下圖所示：



本課程分成 12 個主題，共 52 小時：

Topic 1:導論（2hr）

首先介紹網際網路的發展及全球資訊網路的組織。之後介紹 Markup Language 的由來及其興起的原因，他與傳統的 program language 之間有何不同。此外，為了讓學生有所瞭解，我們也介紹標準通用標記語言（SGML）。

Topic 2: 標記語言初步的介紹（4hr）

這部分先介紹 HTML 語言，讓學生對於什麼是標記語言有初步的認識。因為 HTML 是網頁最常用的語言，因此我們以此為入門的語言，並介紹 HTML 與 Browser 的關係及 Browser 的運作模式。為了讓學生對標記語言使用有所瞭解，我們將介紹 HTML 中一些常用的標記。

Topic3:樣式表介紹（2hr）

標記語言是用來儲存資料用的，對於資料的呈現則需要樣式表。這部分將介紹 HTML

的樣式表及其語法，讓學生瞭解如何修改 HTML 資料的呈現。

Topic 4:以 XML 建立標記（2hr）

當學生對標記語言初步的認識後，我們在這部分將介紹 XML 標記語言，他與 HTML 之間的差異，及它的語法。

Topic 5:驗證 XML 文件（4hr）

這部分我們將介紹驗證 XML 的目的，well-formed 及 validated XML 之間的差異，及用來驗證 XML 文件的語法（DTD 及 Schema）。

Topic 6:可擴充樣式表語言轉換（XSLT）(3hr)

介紹如何將 XML 轉換成其他格式的語言及轉換的目的。

Topic 7:客製標記語言（Custom Mark Language）(3hr)

介紹目前常用的一些標記語言如：WML、BML、ebXML。

Topic 8:文件物件模組（Document Object Model, DOM）(3hr)

介紹什麼是 DOM，為何需要 DOM，他與 XML 文件結構的關係，及如何利用程式及 DOM 來撰寫應用程式。

Topic 9:XML 的簡易應用程式介面（SAX）(3hr)

介紹 XML 文件在程式應用的另一種方式的使用。說明為什麼要 SAX 的方式及 SAX 應用程式範例。

Topic 10:XML 與資料庫（9hr）

介紹 XML 與資料庫的相似性，如果將 XML 文件當成是一個簡易的資料庫，我們又如何使用，及目前這方面的應用。

Topic 11:Web-Based 系統（6hr）

介紹目前正流行的 Web-Based 系統，內容有網路的協定、Web 上資料庫的驅動、遠端程序呼叫、分散式物件及其系統，最後介紹企業間 Web-Based 應用程式。

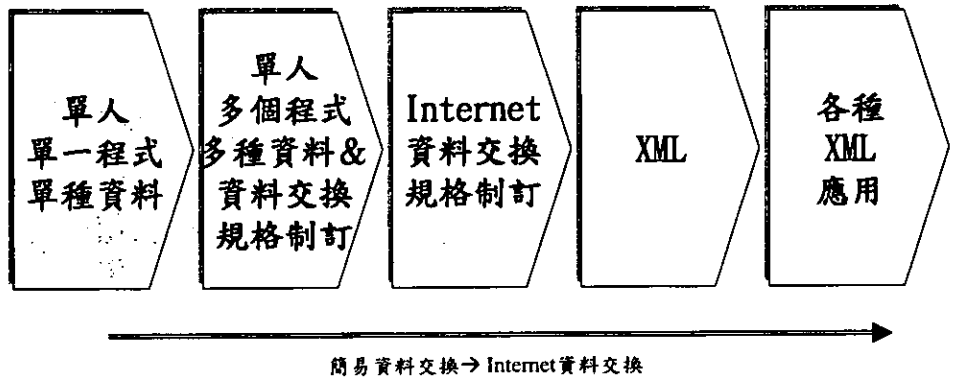
Topic 12:簡易物件存取協定（SOAP）(3hr)

介紹標記語言在分散式系統上的應用，及簡易物件存取協定，並用一個實例說明之。

Topic 13:B2B 與微軟 BizTalk 伺服器（5hr）

這部分介紹標記語言在 B2B 商務上的使用。先介紹 B2B 商務的需求，在介紹目前 Microsoft 的 BizTalk 伺服器及其範例。

各單元進行的順序的關係如下圖所示：



7.6.2 課程大綱

Core Knowledge	Programming Skills	時數（小時）
2. 網際網路（internet）及全球資訊網（WWW）簡介 ● 網際網路的歷史 ● 全球資訊網的歷史及全球資訊網協會（WWW Consortium） ● Markup Language 的由來 ■ 由 program language 到 markup language 的變化 ● 標準通用標記語言（SGML）的歷史 ● Markup Language 興起的理由及未來的趨勢		2
12. 超文件標記語言（HTML）介紹 ● Browser 與 HTML 的關係 ● Browser 如何呈現 HTML 文件 ■ HTML parser ● 標記語言 ● 編輯 HTML ■ 元件 ■ 檔頭 ■ 連結 ■ 圖像 ■ 特殊的字元 ■ 列單 ◆ 排序 ◆ 未排序 ■ 表格 ■ 表單 ■ 框架（frameset）設定 ■ <meta>標籤	● 設計一個入口網站的首頁 本實作需含有： ■ 連結 ■ 圖像 ■ 列單 ◆ 排序 ◆ 未排序 ■ 表格 ■ 表單 ■ 框架（frameset）設定	4
13. 串接樣式表（Cascading Style Sheet -- CSS） ● CSS 用途介紹 ● 內嵌樣式（InLine Styles） ● 外部樣式表連結 ● 元件內建立樣式表 ● 樣式 ◆ Positioning Element ◆ Background	● 將上一個作業的內容做樣式上的修改	2

◆ Dimensions ◆ Text Flow and Box Model ● 使用者樣式表		
14. 以 XML 建立標記 ● HTML 的不足 ■ 標記已經事先定義好而無法自行定義新的標記 ● 介紹 XML 標記 ● parser 及 Well-formed 的 XML 文件 ● XML 所用的字元 ● 標記 ● CDATA 區塊 ● XML 命名空間	● 將一份一般會議記錄轉換成 XML 的檔案。	2
15. 驗證 XML 文件 ● 驗證 XML 文件的目的 ◆ Strong type check 的介紹 ● well-formed 及 validated XML 文件的差別 ● 文件類型宣告（DTD） ◆ element type declaration ◆ attribute type declaration ◆ 條件區 ● Schema ◆ Schema 與 DTD 的差異 ◆ Microsoft XML 與 W3C XML ◆ 描述元素 ◆ 描述屬性 ◆ 資料類型	● 設計上一個作業的 DTD 檔，作為資料交換的依據。	4
16. 可擴充樣式表語言轉換（XSLT） ● XSTL 的目的 ● 什麼是轉換 ● 讀取資料：XPATH ● 轉換資料：XSLT ■ 樣版 ■ 建立元素和屬性 ■ 重複與排序 ■ 條件處理 ■ 節點複製 ■ 結合樣式表 ■ 變數 ● XML 轉換到 XML 範例	● 將之前的會議記錄之 XML 檔轉換成利用 HTML 檔	3
17. 客製標記語言（Custom Markup Language）		3

● 無線標記語言（WML） ● Java 元件標記語言（BML） ● 電子商務標記語言（ebXML）		
18. 文件物件模組（Document Object Model，DOM） ● DOM 應用的需求 ● DOM 的介紹 ● XML 文件的結構 ● DOM 的使用 ● DOM 與 XML 並用的應用程式範例	● 設計一程式顯示之前會議記錄之 XML 檔案內容。	3
19. XML 的簡易應用程式介面（SAX） ● SAX 的應用需求 ● 載入 Reader ■ Content Handlers ■ Error Handlers ■ DTD Handlers ● 驗證 ● 範例	● 利用 SAX 將之前會議記錄之 XML 檔內容以樹狀方式呈現。	3
20. XML 與資料庫 ● 資料模型化 ■ 設計 XML 文件 ■ 撰寫 Schema ● XML 資料連結 ■ 使用 Castor 內建的 introspection ■ 使用 Castor 對映檔 ■ 使用 Castor 的來源產生器 ● 查詢 XML ■ W3C XML Query 語言 ■ XQuery 與 XSLT 比較 ● 個案研究	● 設計一個簡易 XML 資料庫檔，之後用 W3C XML Query 語言來實作	10
21. Web-Based 系統 ● 應用程式之協定 ● Web 工程之各項原則 ● Web 端之資料庫驅動 ● 遠端程序呼叫 ● 輕量級分散式物件 ● 分散式物件系統 ● 企業間 Web-Based 應用程式	● 設計一 Web-based 應用程式	6
22. 簡易物件存取協定（SOAP）	● 設計一 SOAP 應用程式	5

● 介紹 SOAP ● SOAP 結合到傳送協定 ● 透過 HTTP 的同步 SOAP 實例		
23. B2B 與微軟 BizTalk 伺服器 ● B2B 商務的需求 ● 解決方案 ● 微軟 BizTalk 伺服器 ● 一個 B2B BizTalk 範例		5
Total:		52