CH 07
程式語言與軟體工程

李官陵 彭勝龍 羅壽之

學習目標

- 電腦世界中，需要一種能與電腦溝通的語言：「程式語言」(programming language)
- 透過程式語言，將工作內容描述於程式 (program) 中，讓電腦執行
- 不同時空背景與需求，程式語言演進出許多種類
- 說明程式語言的分類與應用領域、電腦如何執行程式、如何撰寫程式

人與人用「語言」溝通

人與電腦用「程式語言」溝通

大綱

- 演進
- 應用領域與分類
- 程式轉譯與執行
- 撰寫程式
- 執行流程描述
- 開發與維護

程式語言的演進

- 機器語言 (machine language)
  - 電腦硬體內部處理對象為0與1所組成的數字
  - 所有執行指令用數字表示
  - 程式也是用數字描述，不容易直接理解

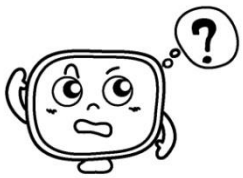
```
4D5A 9000 0300 0000 0400 0000 FFFF 0000
B800 0000 0000 0000 4000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 8000 0000
0E1F BA0E 00B4 09CD 21B8 014C CD21 5468
6973 2070 726F 6772 616D 2063 616E 6E6F
7420 6265 2072 756E 2069 6E20 444F 5320
6D6F 6465 2E0D 0D0A 2400 0000 0000 0000
5045 0000 4C01 0700 ECB8 5046 0000 0000
0000 0000 E000 0E02 0B01 0238 981C 0600
007C 0600 004A 0000 4012 0000 0010 0000
0030 0600 0000 4000 0010 0000 0002 0000
0400 0000 0100 0000 0400 0000 0000 0000
7C0E 0700 0004 0000 18D3 0600 0300 0000
0000 2000 0010 0000 0000 1000 0010 0000
0000 0000 1000 0000 00A0 0600 E101 0000
00B0 0600 980D 0000 00C0 0600 0020 0000
```

計算100+20

完全看不懂

程式語言的演進(續)

- 組合語言 (assembly language)
  - 利用英文縮寫字代替數字的編碼指令
  - 加法與減法的指令用add與sub表示
  - 將組合語言翻譯成機器語言，需要組譯器 (assembler) 的軟體



計算100+20

```
.model small
.data
  opr1 dw 0064h
  opr2 dw 0014h
  result dw 01 dup (?),'$'
.code
  mov ax,@data
  mov ds,ax
  mov ax,opr1
  mov bx,opr2
  clc
  add ax,bx
  mov di,offset result
  mov [di], ax
  mov ah,09h
  mov dx,offset result
  int 21h
  mov ah,4ch
  int 21h
  end
```

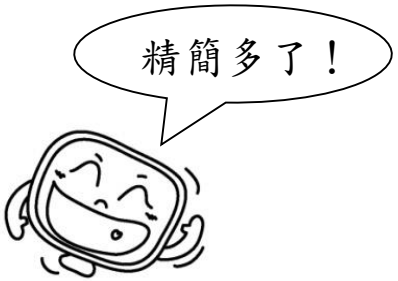
程式語言的演進(續)

- 高階語言 (high-level language)
  - 表達方式接近人類使用的自然語言 (natural language)
  - 透過 $x + y * (x - 10)$ 直覺的數學運算式
  - 利用if-else的句子，表示當某個條件成立與不成立時，需要執行的工作
  - 需要更聰明與更複雜的轉譯程序，將程式翻譯成機器語言

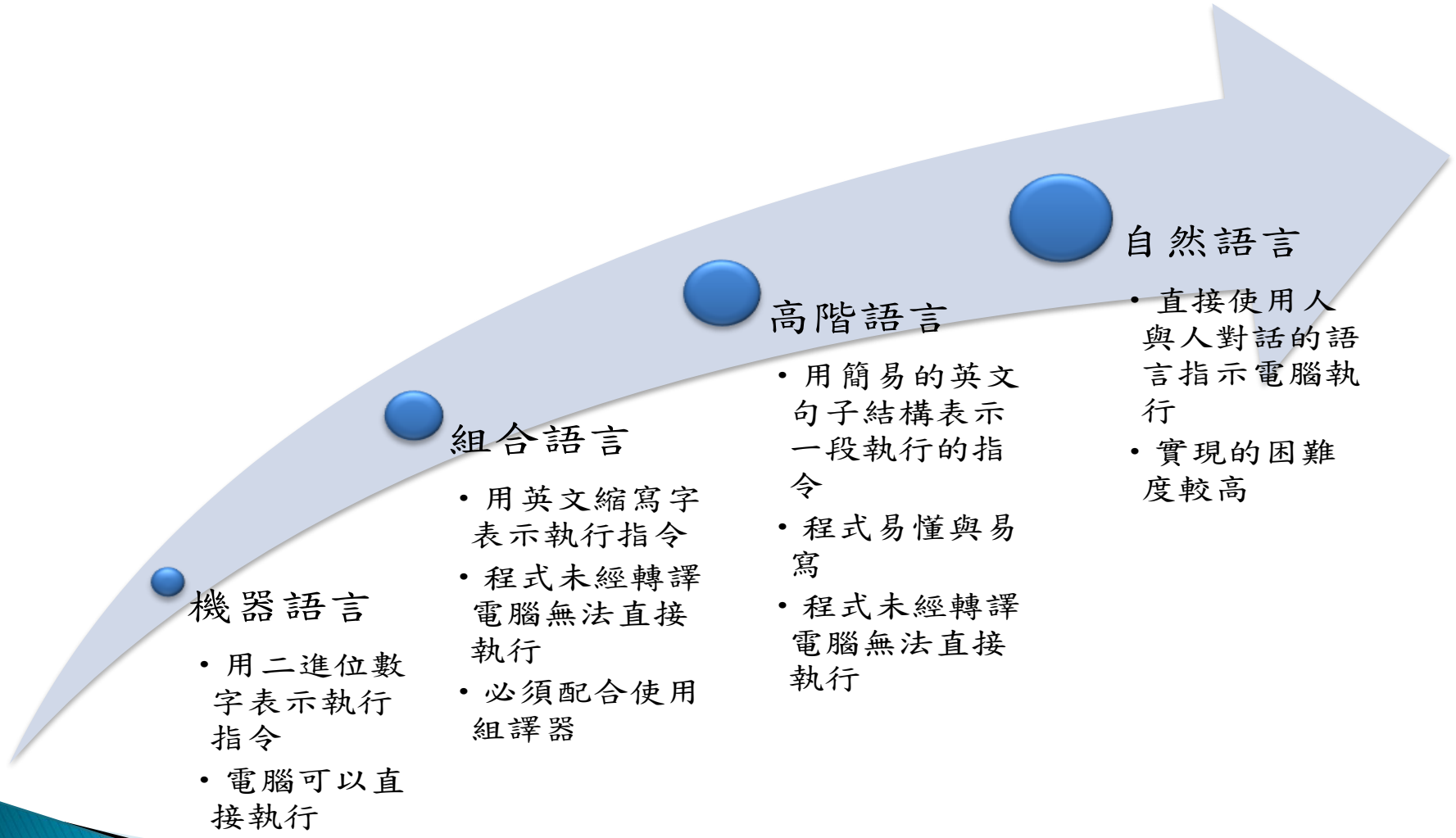
```
#include <stdio.h>
#include <stdlib.h>
int main ()
{
  int opr1 = 100;
  int opr2 = 20;
  int result;
  result = opr1 + opr2;
  printf ("sum = %d\n", result);

  return 0;
}
```

計算100+20



程式語言的演進(續)



程式語言的應用領域

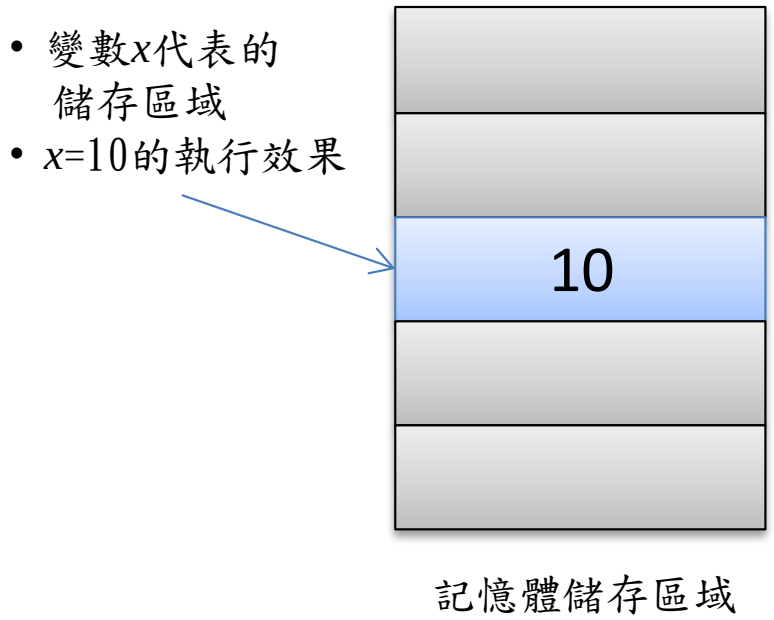
- 處理工程科學問題
  - 計算複雜的數學運算，推估飛彈的路線，或是進行飛機的模擬飛行控制等
  - 1957年誕生的Fortran語言提供方便的矩陣與三角函數等多種的數學運算
- 實現人工智慧 (artificial intelligence)
  - 讓電腦更聰明能學習人類的思考模式
  - 程式語言能方便表達邏輯觀念與推理思考
  - 1958年誕生的Lisp語言與1972年出現的Prolog語言

程式語言的應用領域(續)

- 協助財金資料管理
  - 利用電腦儲存與管理客戶資料
  - 記錄與計算每筆金錢交易後的帳戶餘額
  - 1959年出現的COBOL語言，提供方便的指令，製作財務報表
- 電腦軟體系統開發
  - 電腦運作需要軟體的配合做為系統管理
  - 軟體執行要求高的執行效率與對硬體元件直接的控制
  - 1972年出現的C語言，允許在高階程式中，直接嵌入組合語言，方便存取電腦的硬體設備

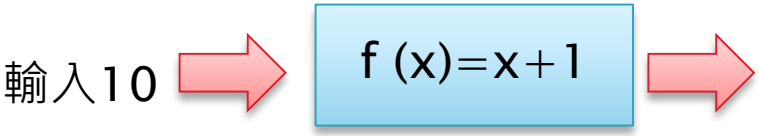
程式語言的分類

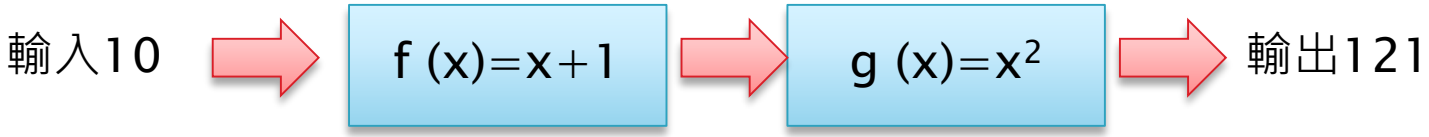
- 命令式語言 (imperative language)
  - 利用變數 (variable) 表示一個可儲存資料內容的某塊電腦記憶體區域
  - 透過設定 (assignment) 的指令，變更變數的內容
  - $x = x + 1$表示取出變數x的儲存內容，再加上1，接著儲存回變數 x 的儲存區域
  - Fortran、COBOL與C語言



程式語言的分類 (續)

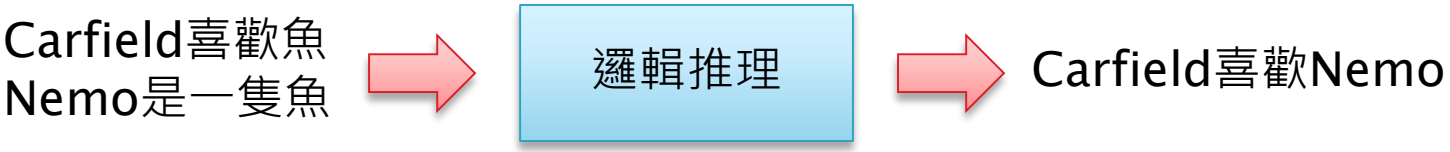
- 函數式語言 (functional language)
  - 函數可將輸入的資料，經過某種運算輸出結果
  - 這種程式語言需要定義許多函數來完成工作的任務
  - 例如定義函數 $f(x) = x + 1$，函數 $g(x) = x^2$
  - Lisp語言





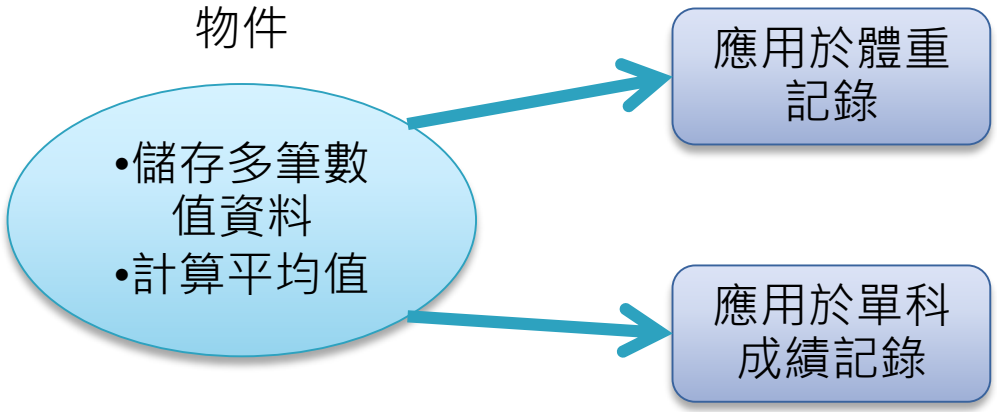
程式語言的分類 (續)

- 邏輯式語言 (logical language)
  - 處理邏輯條件的判斷與推論
  - 適合開發專家系統軟體：電腦圍棋、醫生診斷
  - Prolog語言



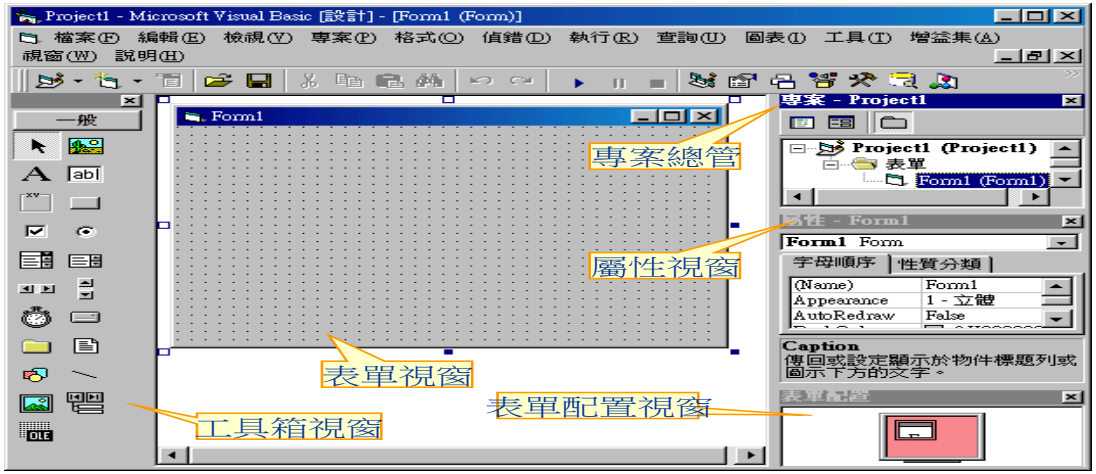
程式語言的分類 (續)

- 物件導向式語言 (object-oriented language)
  - 將產品開發的想法應用於軟體開發
  - 軟體也可以由軟體元件所建構，這個軟體元件就是物件
  - 物件除了儲存資料外，還提供資料處理的方法
  - 物件可重複利用，降低類似程式開發的成本
  - C++與Java語言



程式語言的分類 (續)

- 其他命令式語言
  - 視覺化語言 (visual language)：提供快速方式建構視窗應用程式，如Visual Basic、Visual C++與Visual C#等
  - 腳本語言 (scripting language)：電腦開機的檢查程序與環境設定、網頁的執行檢查，如JavaScript



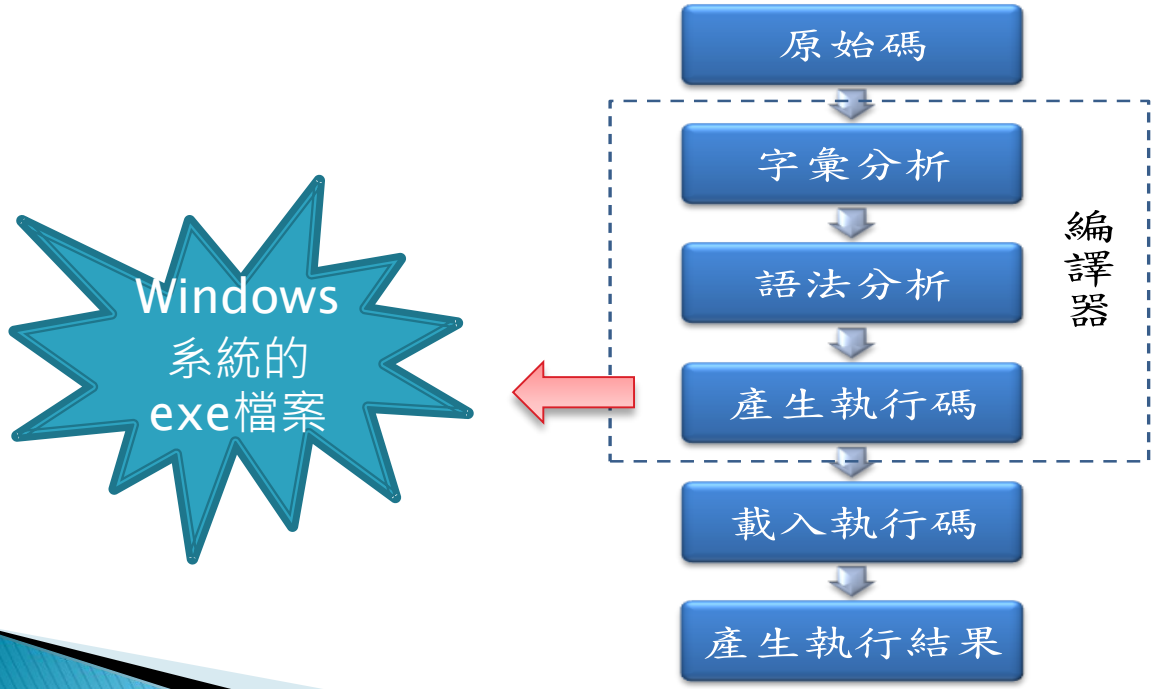
Visual Basic開發環境

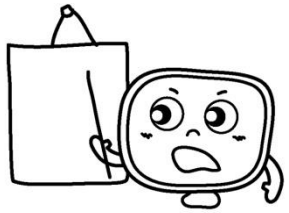
程式的轉譯與執行

- 英翻中的處理
  - 檢查單字對不對、句子文法對不對、瞭解意思後將句子轉成中文
- 程式轉譯的步驟
  - 字彙分析 (lexical analysis)：檢查是否使用不合法的字彙
  - 語法分析 (syntax analysis)：檢查是否使用不合法的敘述結構
  - 執行碼產生 (code generation)：將程式內容轉譯成機器語言碼

程式的轉譯與執行 (續)

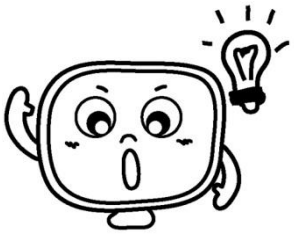
- 實現程式轉譯的方式主要有兩種：編譯 (compilation) 與直譯 (interpretation)
- 編譯器 (compiler) 的處理程序





程式的轉譯與執行 (續)

- 直譯器 (interpreter) 處理程序



程式的轉譯與執行 (續)

- 編譯方法的優缺點
  - 轉譯與執行切割開來，程式執行效率較高
  - 程式開發者必須排除所有的錯誤後，才能執行程式
- 直譯方法的優缺點
  - 轉譯與執行綁在一起，程式執行效率較低
  - 程式開發者可以陸陸續續看見執行的結果，屬於邊改邊執行的運作模式

程式的轉譯與執行 (續)

- 結合編譯與直譯的混合 (hybrid) 模式
  - 產生另一種中間程式碼， Java程式稱為bytecode
  - Java的直譯器稱為Java虛擬機器 (Java Virtual Machine, JVM)
  - 提升程式使用的可攜性

隨堂練習

- 如果進行英翻中的工作採取編譯器相同的三個階段 (字彙分析、語法分析與執行碼產生)，請問底下的訊息來自哪個階段？

1. “Excuse me” → 抱歉
2. ”You are grea” → grea指great嗎
3. ”I make help you” →有兩個動詞？

程式的撰寫

- 判斷產生的任意整數是奇數或是偶數 - C語言為例

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
  int number;  //變數宣告
  int flag;  //變數宣告
  number = rand();  //產生任意整數
  flag = number % 2;  //求餘數
  if  (flag == 0)  //表示數字為偶數
    printf("%d is an even number\n", number);  //輸出答案
  else     //表示數字為奇數
    printf("%d is an odd number\n", number);  //輸出答案
  return 0;
}
```

程式輸出結果

```
41 is an odd number
```

程式的撰寫 (續)

- 使用函數 (function)
  - 別人寫好的一段程式碼，處理特定的工作
  - 程式語言提供的各種函數，集中放置於函數庫中 (某個檔案)
  - 執行函數透過呼叫函數名稱
  - 某個函數的呼叫使用規格寫在另一個標頭檔案 (header file) 中
- 呼叫函數的前置程序
  - 引用正確的標頭檔案

```
#include <stdio.h>
#include <stdlib.h>
```

程式的撰寫 (續)

- 呼叫函數的方法
  - 例如：產生任意一個正整數

```
number = rand();  //產生任意整數
```

  - rand後面的圓括弧內無任何內容，表示這個函數不需要輸入參數 (parameter)。有些函數需要參數，如取絕對值：abs (–10)
  - number是變數，儲存函數的回傳值 (return value)
  - 雙斜線//是程式註解
  - 分號表示一行敘述的結束

程式的撰寫 (續)

- 使用變數需要事先宣告
  - 電腦需要知道變數會儲存何種資料內容，大小大概多少

```
int number;  //變數宣告
```

  - int是資料型別整數 (integer) 的意思
  - C語言變數名稱可以是混合數字與大小寫英文字母的任何組合字，但字首不得為數字
  - C語言中變數名稱相同但大小寫不同，仍視為不同的變數
    - number VS. Number

程式的撰寫 (續)

- 善加利用運算符號
  - 例如：加法 (+)、減法 (-)、乘法 (*)與除法 (/)
  - 例如：取餘數 (%)

```
flag = number % 2;  //求餘數
```

  - %是運算子，number與2是運算元
  - 組合出複雜的算術運算式： $(x + 2) * (y - 3) - 4$

程式的撰寫 (續)

- 條件判斷需要邏輯運算式

```
if  (flag == 0)  //表示數字為偶數
    … //條件成立要執行的敘述
else     //表示數字為奇數
    … //條件不成立要執行的敘述
```

  - flag == 0判斷flag變數是否等於0
  - 其他的邏輯運算子：大於 (>)、小於 (<)、不大於 (<=)、不小於 (>=)、不等於 (!=)
  - if-else控制敘述會依據邏輯運算式的判斷結果：當結果為真時，執行if後面的敘述，不然執行else後面的敘述

程式的撰寫 (續)

- 螢幕輸出答案使用printf

```
printf("%d is an even number\n", number);
```

  - 呼叫時傳入兩個參數
    - 第一個參數為雙引號標示的一段文字，用途是標示訊息顯示的樣版
    - %d 的部分會由第二個參數number儲存的數值所取代
  - \n 表示輸出完訊息後會換至新的一行
  - 雙引號標示的文字又稱字串 (string)，指一串的字元

程式的撰寫 (續)

- 程式的執行從主函數開始

```
int main()
```

  - main是程式開發者寫的主要執行函數
  - 說明main不需要參數，傳回值是整數

```
return 0;
```

  - main執行結束回傳數字0

程式的撰寫 (續)

- 讓程式自動執行多次：使用迴圈的觀念

```
int main()
{
  int number, flag, i;
  i = 0;
  while  (i < 5) {   //重複執行5次
     number = rand();  //產生任意整數
     flag = number % 2;
     if  (flag == 0)  //表示數字為偶數
         printf("%d is an even number\n", number);
     else   //表示數字為奇數
         printf("%d is an odd number\n", number);
     i = i + 1;
  }
  return 0;
}
```

程式的撰寫 (續)

- 迴圈控制伴隨邏輯運算式

```
while (i < 5) {   //檢查迴圈是否結束
    …   //迴圈重複執行的敘述
    i = i + 1;
}
```

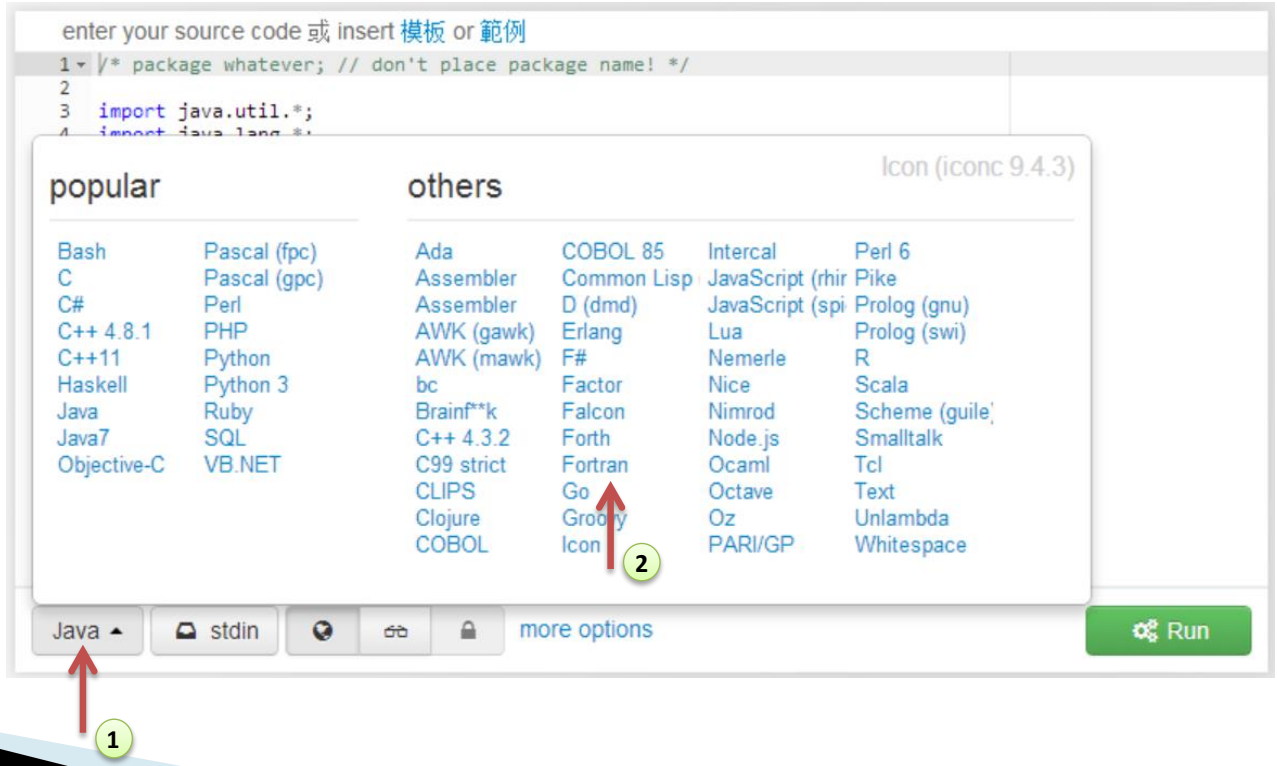
  - i < 5是判斷小於的邏輯運算式
  - while是控制迴圈的敘述，當邏輯運算式成立時，繼續迴圈的執行
  - i = i + 1讓變數i有大於5的機會，迴圈得以停止

程式的撰寫 (續)

- 學習新程式語言的法則，先瞭解底下的使用：
  - 儲存資料支援的資料型別有哪些？除了整數、浮點數與字元外，還有其他的嗎？
  - 如何表示某種資料型別的資料內容？例如浮點數的表達方式除了125.2外，利用指數的方式1.252E2也可以嗎？
  - 支援的運算子有哪些？表示的符號為何？
  - 提供哪些程式執行的控制結構？撰寫的語法為何？
  - 如何撰寫自己的函數？如何呼叫函數庫現成的函數？

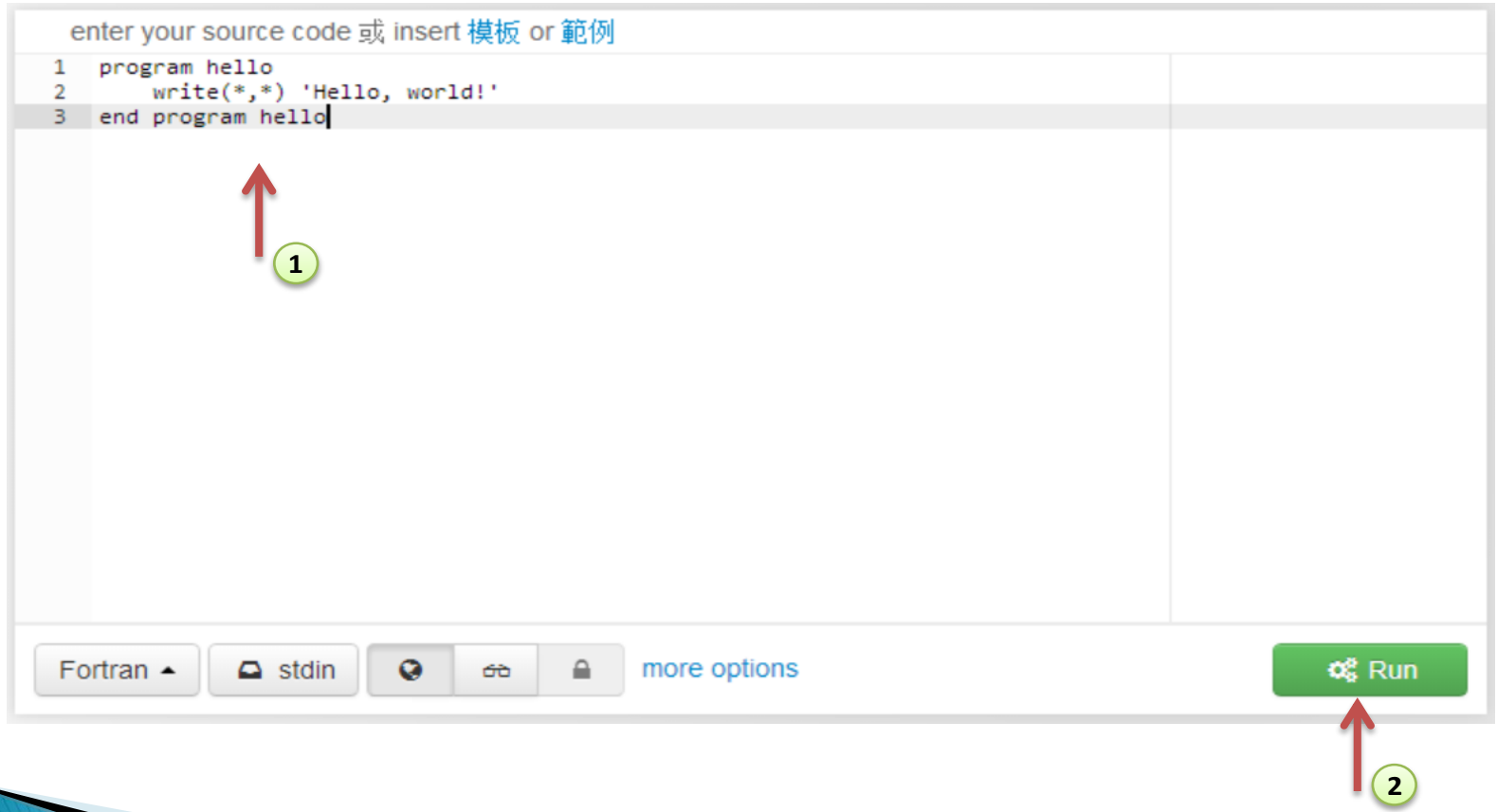
線上程式測試與執行

- 電腦不用安裝程式開發軟體，只需要網頁瀏覽器
- 網址：ideone.com
- 步驟一：選取程式語言種類



線上程式測試與執行 (續)

- 步驟二：編輯 (或複製貼上) 程式內容

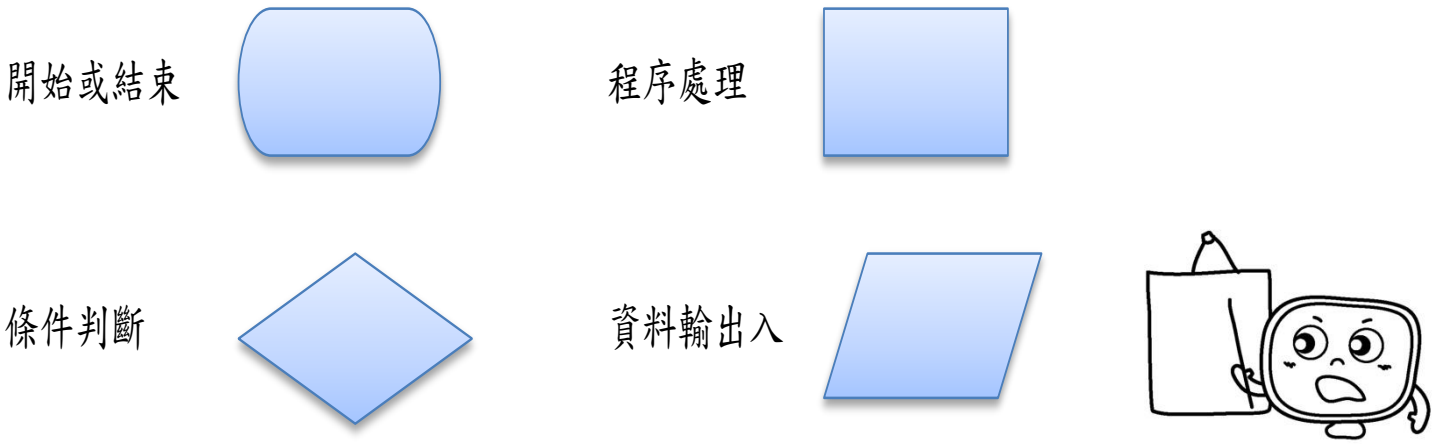


線上程式測試與執行 (續)

- 步驟三：執行程式與檢查結果

程式的執行流程

- 基本上程式的執行是依據內容的敘述，從頭至尾逐行依序進行
- 程式的執行流程，可以利用流程圖 (flowchart) 表示

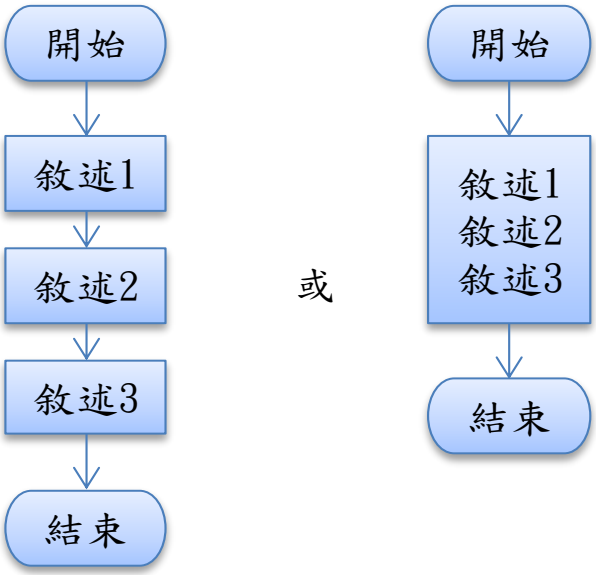


程式的執行流程 (續)

- 三種常見的執行次序：順序執行、條件執行與重複執行
- 順序執行：執行的敘述用單一程序處理表示，或是用一個程序處理內含多個敘述也可以

順序執行
(sequential execution)

敘述1;
敘述2;
敘述3;

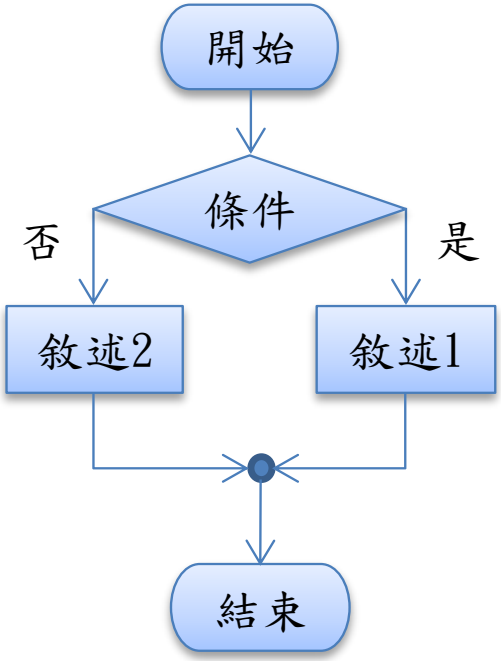


程式的執行流程 (續)

- 條件執行：利用條件判斷成立與否，決定那個敘述該被執行
  - if-else雙向選擇 (two-way selection) 的使用

條件執行
(conditional execution)

```
if (條件)
   敘述1;
else
   敘述2;
```

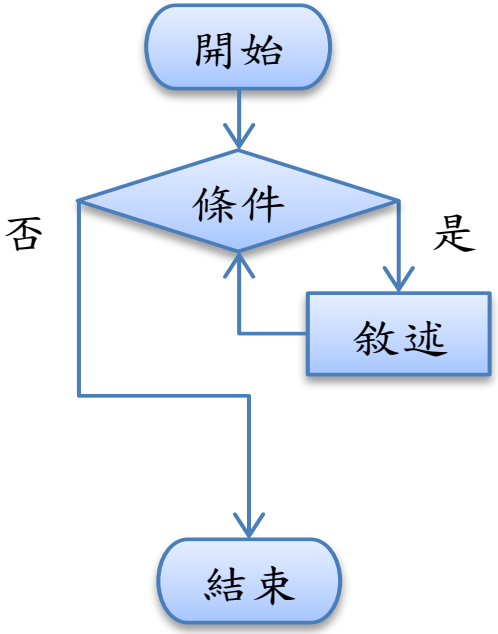


程式的執行流程 (續)

- 重複執行：利用條件判斷成立與否，決定是否重複執行某個敘述
  - while敘述的使用：”當”條件成立時繼續執行

重複執行
(repeated execution)

```
while (條件)
   敘述;
```

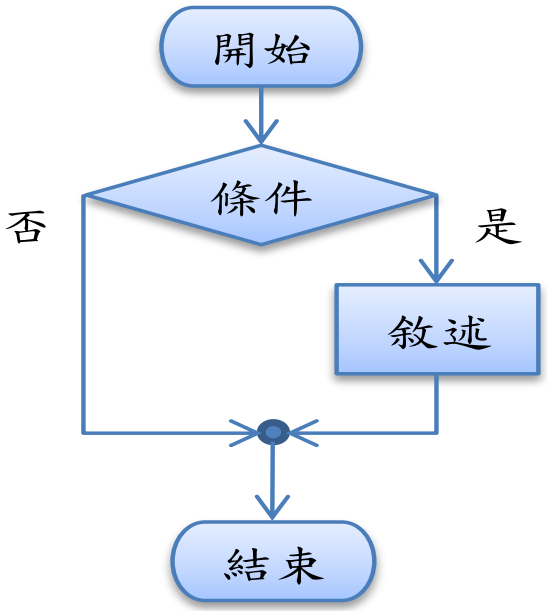


程式的執行流程 (續)

- 其他執行次序的變形
  - if敘述的單向選擇

條件執行
內含單一選擇

```
if (條件)
   敘述;
```

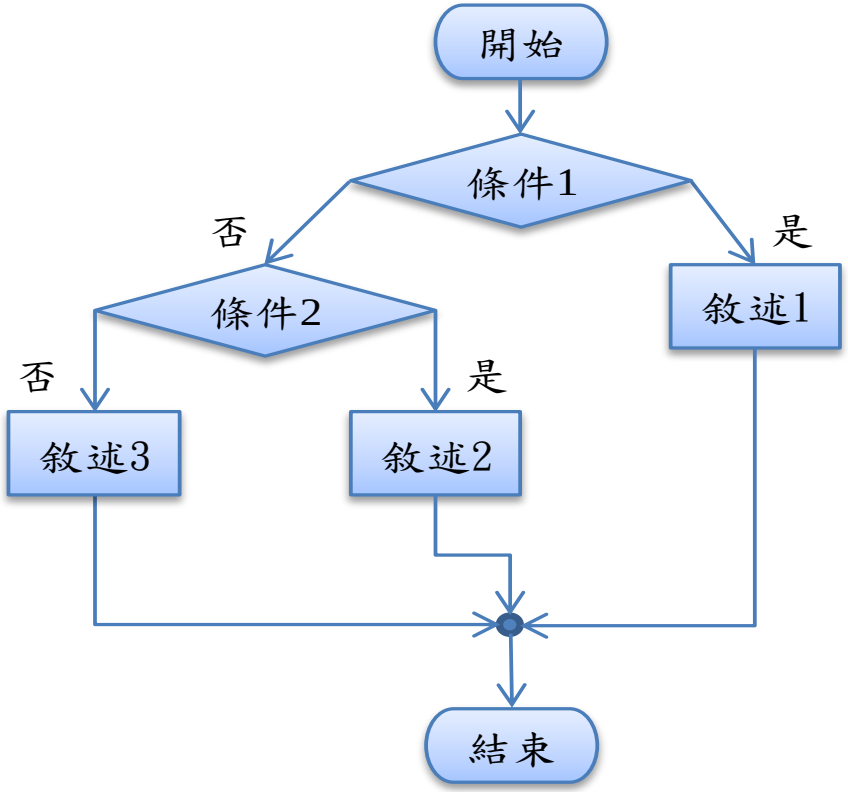


程式的執行流程 (續)

- 其他執行次序的變形
  - 巢狀條件敘述

條件執行
內含多層條件

```
if (條件1)
   敘述1;
else
  if (條件2)
     敘述2
  else
     敘述3
```

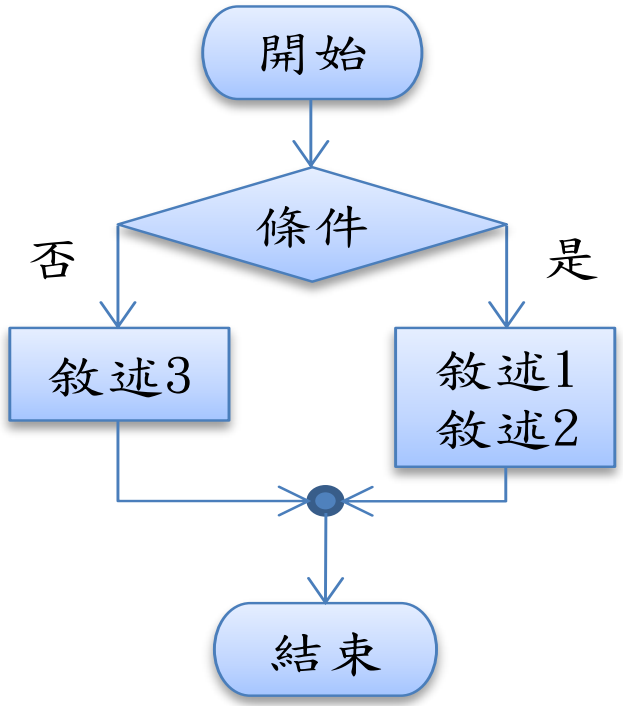


程式的執行流程 (續)

- 其他執行次序的變形
  - 複合敘述

條件執行
內含複合敘述

```
if (條件) {
    敘述1;
    敘述2;
}
else
    敘述3;
```



隨堂練習

- 請畫出底下程式片段的流程圖：

(1)
```
if  (x > 10)
    if  (y < 5)
        z = 1;
    else
        z = 2;
```

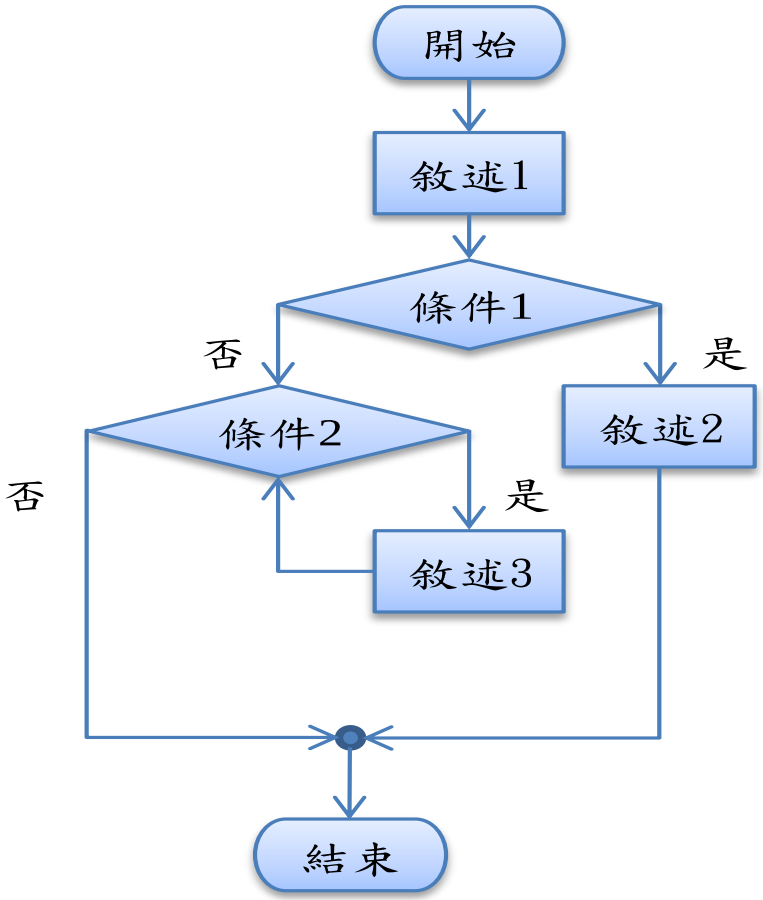
(2)
```
while  (x > 10) {
        y = y + 1;
        x = x – 1;
    }
```

程式的執行流程 (續)

- 組合三種執行次序

組合式程式

```
敘述1
if (條件1)
   敘述2;
else
  while (條件2)
     敘述3;
```



隨堂練習

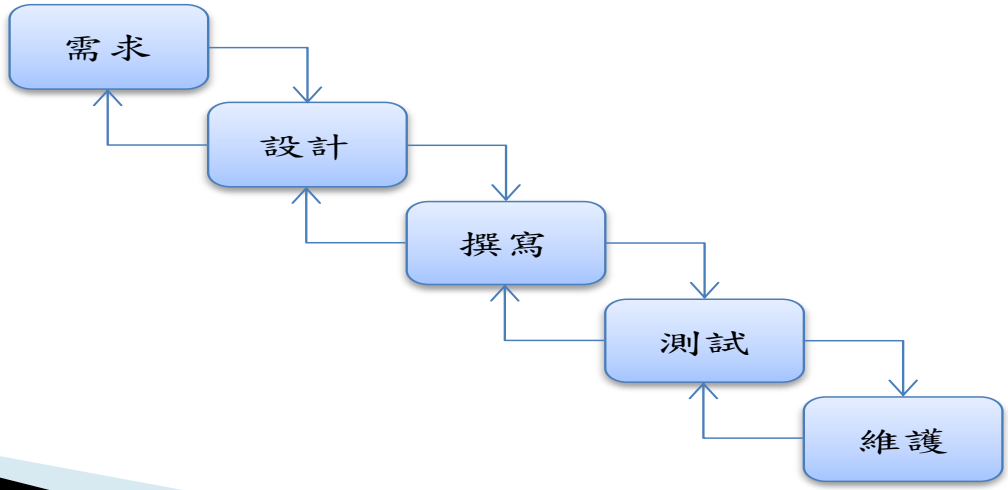
- 請畫出底下程式片段的流程圖：

(1)
```
while (x < 10) {
    if (y < 5)
        z = 10;
    x = x + 1;
}
```

(2)
```
if (x > 10) {
    z = 1;
    while (y < 5)
        y = y + 1;
}
```

程式的開發與維護

- 軟體品質太差，執行容易出錯，甚至當機
- 提升軟體品質就必須將軟體視為工程的建造，採用一套嚴謹的生產模式，這衍生了「軟體工程」(software engineering) 的觀念
- 瀑布式模型 (waterfall model)，將整個軟體開發過程區分為五個階段

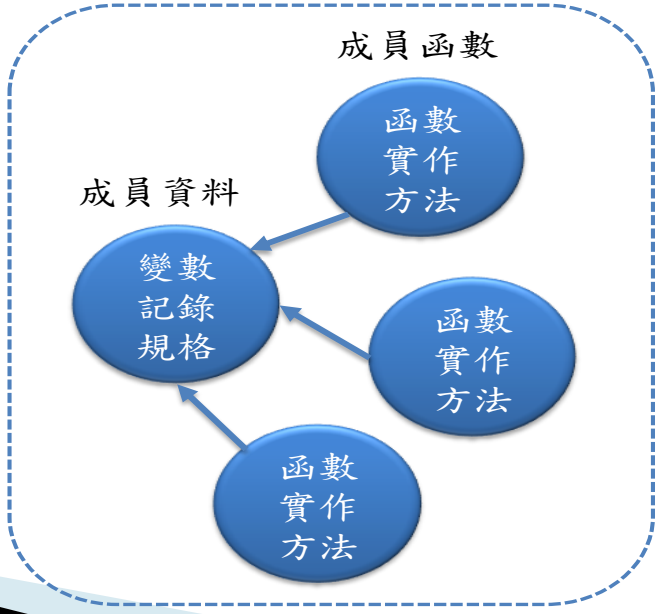


程式的開發與維護 (續)

- 需求：瞭解與分析客戶的需求，例如：需要處理的問題與達成的目標，需要哪些功能、介面如何呈現等
- 設計：依據需求開始規劃軟體架構，構思用哪種設計的方式，整個工作內容可做切割允許分工進行
- 撰寫：依據設計的規格說明開始撰寫程式，去除程式可能的執行錯誤
- 測試：整合各部分開發的程式做完整的測試，客戶可參與並確認是否滿足需求
- 維護：隨時依據客戶使用的經驗回報，修正程式排除可能的缺失

程式的開發與維護 (續)

- 物件導向式語言簡化程式的開發與維護成本
  - 支援軟體重複利用 (reusability)
- 物件實現軟體元件的規格與功能
  - 成員資料 (member data)：利用變數儲存規格資料
  - 成員函數 (member function)：透過函數提供物件的操作



程式的開發與維護 (續)

- 軟體元件是將成員資料與成員函數封裝 (encapsulate) 在一起
- 軟體元件的框架或模型用類別 (class) 的形式定義
- 透過類別可以產生規格相同、運作形式相同的許多物件 (object)
- 程式在執行時就是透過不同物件的驅動與互動來完成

程式的開發與維護 (續)

- 物件導向式語言的重複利用來自類別的繼承 (inheritance)
  - 父類別 (super-class) VS. 子類別 (sub-class)
  - 單一繼承 (single inheritance) VS. 多重繼承 (multiple inheritance)