2

演算法

本章概念

教材設計說明

了解演算法就是解決問題的方法，並學習如何透過流程圖來表達演算法，以及流程控制結構，以銜接程式設計章節。

我們身處在資訊科技時代，時常要蒐集資料，並轉換成對我們有用的資訊。例如：智慧手環會蒐集使用者的行走步數、心跳、睡眠等資料，再經由程式將這些資料轉換成健康狀況的資訊，提供使用者參考。

程式必須依據演算法所規畫的步驟執行，才能將資料轉換成有用的資訊，因此在學習程式設計之前，我們先來了解何謂演算法。

23

2-1 演算法簡介

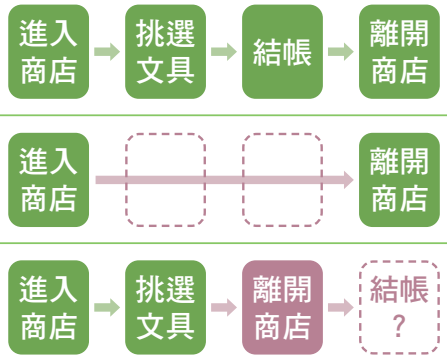
教學重點

說明演算法就是解決問題的方法，並以活動讓學生體驗指令「明確性」的重要。

1 演算法的特性

日常生活中，我們會面臨許多問題，並為這些問題找到解決的方法，這些解決問題的方法都可以視為一種廣義的演算法，例如蛋糕的作法（食譜）、數學題的解法等。當我們遵循著演算法的步驟執行時，將可以正確解決問題；若任意省略或變動其中的步驟，則可能無法順利達到目的（圖 1-2-1）。

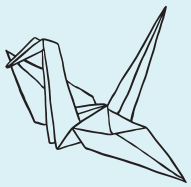
圖 1-2-1 到商店購買文具的演算法。



手腦並用

準備器材

每人準備 1 張 A4 紙。



活動方式

1. 指定一人作為操作者，操作者背對所有人，一邊摺紙，一邊敘述自己摺紙的步驟。
   例 將紙對摺、將左上角摺一個三角形……。
2. 其他人依照操作者的敘述摺紙，過程中不能提問、討論。
3. 大約進行 7 ～ 8 個摺紙步驟之後，大家互相展示摺紙的結果。

活動討論

每個人所展示的摺紙結果都相同嗎？如果不同，是什麼原因造成的？

圖 1-2-1 演算法的順序性

演算法的步驟有順序性，不可任意省略或更動，否則將無法解決問題。

例如：「炒蛋」時，應該先將蛋打散再下鍋。如果沒有打散再下鍋會變成荷包蛋。

引導建議1

本活動主要讓學生理解「指令的明確性」對於程式執行結果的重要性。不一定要摺出特殊造型的作品，因此只要經過7～8個步驟後，即可檢視全班的摺紙成果。

在手腦並用的活動中，如果操作者描述的指令不夠明確，就會造成每個人摺出來的結果不一樣，例如指令為「對摺、再對摺」，就可能有多種摺法（圖 1-2-2）。 引2

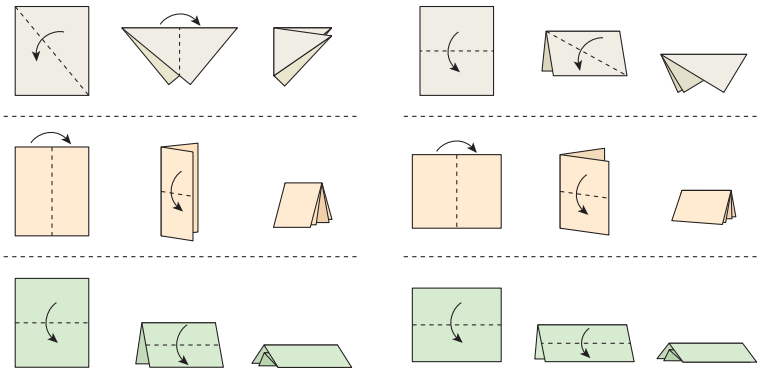


圖 1-2-2 指令不明確時，就會造成不同的執行結果。例如指令為「將紙對摺再對摺」，就可能因為每個人放置紙的方向、對摺的方向不同，而產生各種不同的結果。

上述廣義的演算法常因敘述較不精確，導致解讀不同而使結果有所差異。在資訊科技領域中，演算法有更嚴謹的定義，以確保演算法能正確執行並有效解決問題。演算法通常都具有以下特性：

演算法的五大特性

1. 輸　入：可有多個輸入資料，或是沒有輸入資料。
2. 輸　出：必須至少有一個輸出結果。
3. 明確性：每個指令必須明確，不可模稜兩可。
4. 有限性：執行演算法，必須在有限步驟內結束。
5. 有效性：又稱可行性，演算法中的每個命令都必須是可執行的步驟，用紙筆也能推演完畢，以確定能解決問題。

引導建議2

講解完後，可以再重複進行一次摺紙活動，讓學生學習如何下達明確的指令。

2-1　演算法簡介　25

教學重點

因每個人的思考方式不同，演算法沒有一定的答案，針對同一個問題，可以有很多種演算法。

2 演算法與電腦

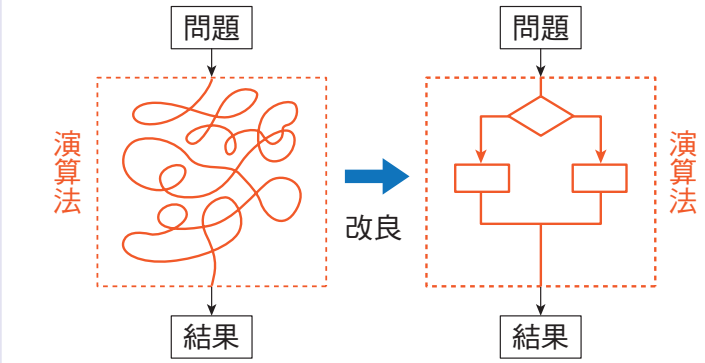
現在的電腦功能越來越強大，但它運作的背後，其實都是遵循著演算法執行的。如果想讓電腦具有特定的功能，我們就必須設計一套演算法，明確地告訴電腦：當它碰到什麼狀況時應該如何反應，或是應該執行什麼步驟。

由於每個人的思考方式不同，針對同一個問題，可能會設計出不同的演算法；而不同的演算法設計，會影響電腦處理問題的結果與效率。例如：以相同的關鍵字在不同的搜尋引擎上進行查詢，查詢的速度與所獲得的結果各有不同，這就是演算法造成的差異。而由於搜尋引擎的演算法不斷進步，讓我們查詢資料也越來越快速、越來越準確。

延伸學習　如何設計出更好的演算法？

在現實世界中，針對同一個問題，可能已經存在各種解決方法，但我們仍會不斷尋找更好、更快的方法來達成目標。

對於初學者來說，剛開始只要能夠建立可以解決問題的演算法即可，隨著知識增長與經驗累積，就能改良、設計出更快捷的演算法。



好的演算法能較有效率的解決問題。

1 電腦功能仰賴演算法

電腦功能強大的背後，主要依賴好的演算法。例如：修圖app要把照片裡的眼睛變大、把臉變小、把皮膚變白，而照片裡的哪些部位是眼睛？哪些是臉？哪些是皮膚？這些都是電腦依循演算法步驟執行的結果。

引導建議

想一想，如何數出「一疊紙共有幾張」？

可請學生分享自己會怎麼數，其他學生是否有不同的答案？比較不同的數法哪一種會比較快。例如：

(1) 1張1張數：1、2、3……

(2) 2張2張數：2、4、6……

(3) 以「2張、3張」計算5：

手：2、3…2、3…2、3……

口：5　10　15

3 演算法的表達

解決問題時，我們常會把演算法以文字或圖像表達出來，以作為與他人溝通討論或是後續進行程式設計時的依據。演算法沒有固定的呈現方式，只要能夠清楚表達，並符合演算法的五大特性即可。常見的表達方式有文字、流程圖、虛擬碼 3 種。

1 文字

演算法可以利用人類語言，如中文、英文、日文等文字來描述。這種表達方式最簡單，但敘述較為冗長，且容易因為每個人的表達和認知差異而造成誤解。

舉例

求甲、乙兩數相加之總和，以文字表示：

步驟 1. 輸入「數字甲」。
步驟 2. 輸入「數字乙」。
步驟 3. 將「數字甲」及「數字乙」相加，
　　所得即為「總和」。
步驟 4. 輸出「總和」。

補 文字描述演算法的誤差

例 求兩個正整數相加的結果。
1. 輸入兩個正整數。
2. 兩數相加。
3. 所得即為總和。

根據以上演算法可以獲得正確的結果。

例 求兩個正整數相除的結果。
1. 輸入兩個正整數。
2. 兩數相除。
3. 所得即為解答。

雖然上方描述看似正確，但是第2點並未說明誰是除數、誰是被除數，因此在運算時容易造成混淆。

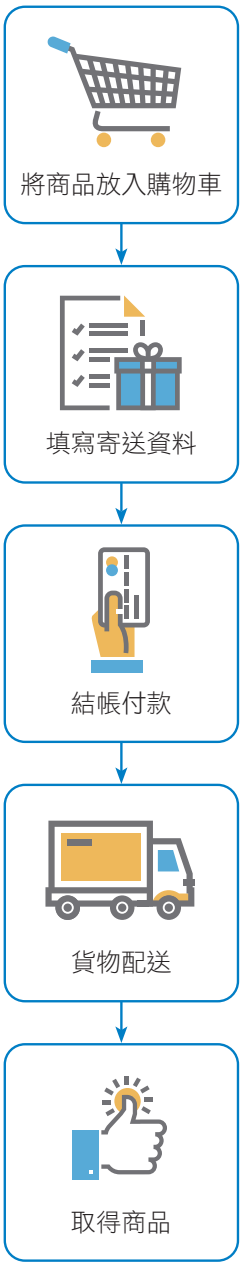


圖 1-2-3 常見的網路購物流程。

2 流程圖

流程圖（flowchart）是利用各種圖形與箭頭等符號，來表示執行一件事情的步驟與順序，不但適用於資訊科技領域，也廣泛應用在工程、設計、商業領域，就連日常生活中，也常看到流程圖的相關應用（圖 1-2-3）。

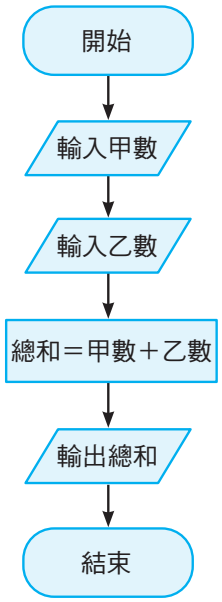
●流程圖的優點

以流程圖來表示演算法，比文字敘述更容易理解，因此被廣泛應用。其優點為：

1. 流程步驟的可讀性較高。
2. 方便除錯。
3. 容易修改流程。

舉例

求甲、乙兩數之總和，以流程圖表示：



引導建議

可引導學生討論在哪裡還有看過流程圖？藉由日常生活常見的流程圖，帶出演算法的表達方式。

28 資訊科技篇

●繪製流程圖

為了方便所有人員都能正確解讀，並依循流程圖來進行作業，因此流程圖必須採用統一且具有特定意義的符號（表 1-2-1）。

在繪製流程圖之前，應先了解每個符號代表的意義與用法，並且依照流程圖繪製原則（圖 1-2-4）來繪製。

流程圖繪製原則

1. 使用標準符號，以便閱讀和分析。
2. 文字簡潔且明確可行。
3. 繪製方向由上而下，由左至右。
4. 流程線條避免交叉或過長。

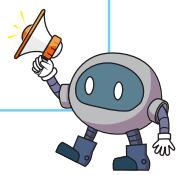


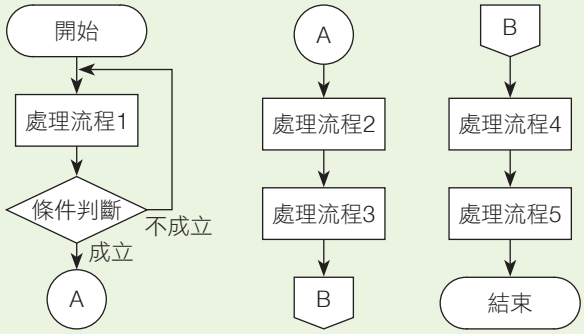
圖 1-2-4 流程圖繪製原則。

表 1-2-1 流程圖圖示說明

符號	名稱	說明
	開始／結束	流程的開始或結束
	輸入／輸出	輸入或輸出資料
	處理程序	要執行或處理的程序
	決策判斷	針對條件進行判斷，以決定執行的流向
	迴圈	重複執行指定的次數
	流向線	流程進行的方向
	註解	表示附註說明之用
	副程式	已定義流程的組合
	文件	輸入或輸出文件

表 1-2-1 流程圖換頁

繪製流程圖時，若因版面不夠需要換行或換頁繪製，可使用「連接符號◯」或「換頁符號⬠」（如右圖）。

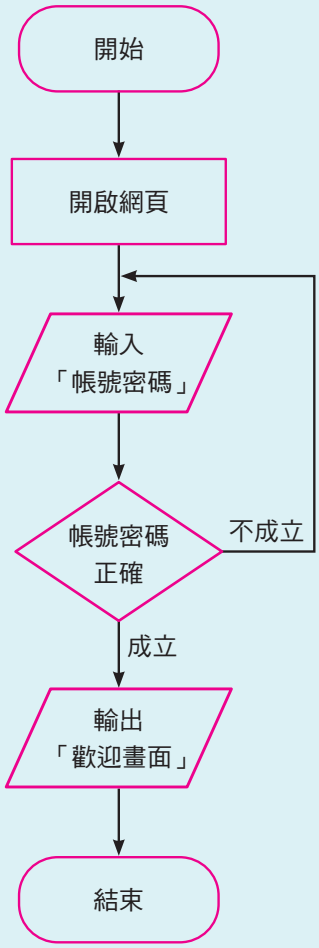


手腦並用

右圖為登入某網站時的操作流程：

使用者開啟網頁後要輸入帳號、密碼，若輸入正確，會顯示「歡迎畫面」；若輸入錯誤，則回到「輸入帳號、密碼」的步驟。

請參照表 1-2-1，試著為每一個步驟加上正確的流程圖符號。



30 資訊科技篇

●流程圖的適用範圍

流程圖可以用來處理簡單的小問題，也可以用來描述複雜的大問題。一般而言，流程圖可以先描述大致的步驟，再逐步細微化，以方便流程的理解、維護與管理（圖 1-2-5）。

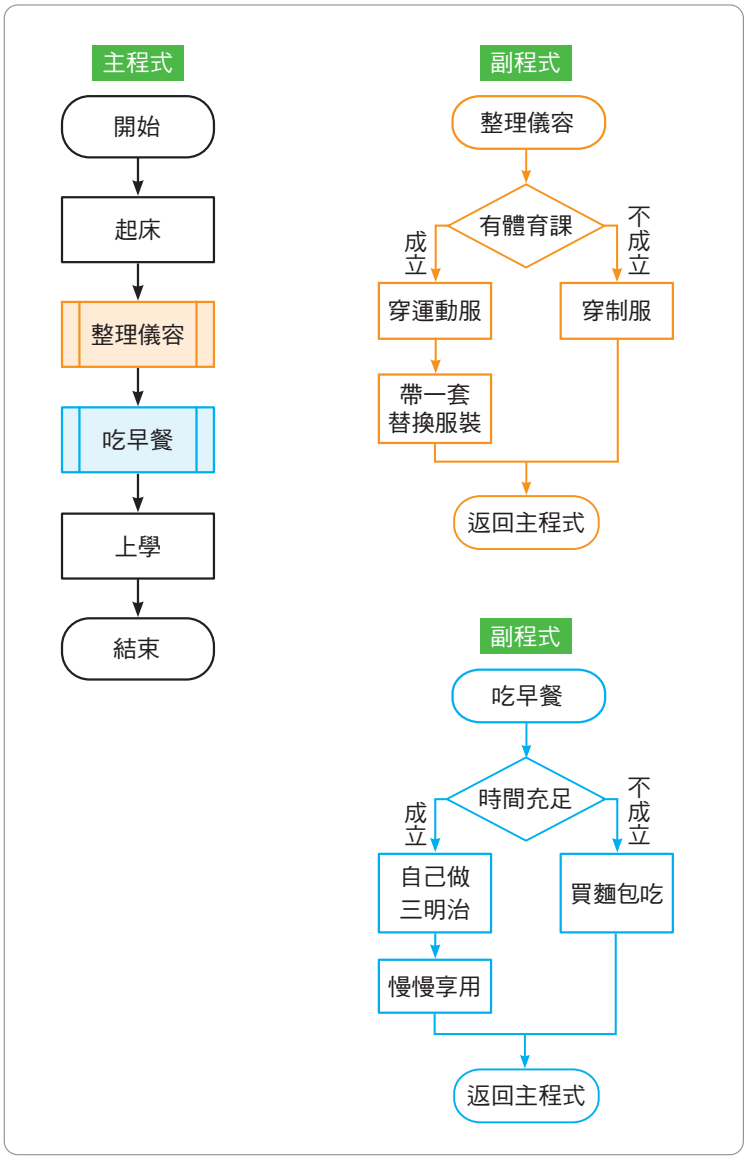
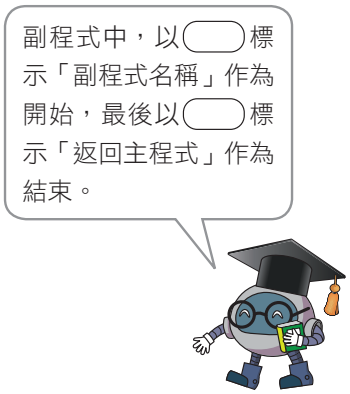


圖 1-2-5 遇到複雜的問題時，將流程圖採用模組化的方式呈現，具有更高的可讀性。



2-1 演算法簡介 31

補充影片

- 人腦也可以執行演算法 (4:57)

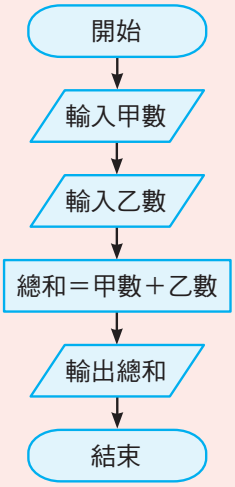
3 虛擬碼

當我們學過程式語言之後，也可以使用虛擬碼（pseudo code）來表達演算法。虛擬碼是一種介於「人類語言」與「程式語言（如 Python、C、C++）」間的表達方式，用類似程式語言的方式來描述執行步驟，兼具文字描述容易表達，以及流程圖容易理解的優點。

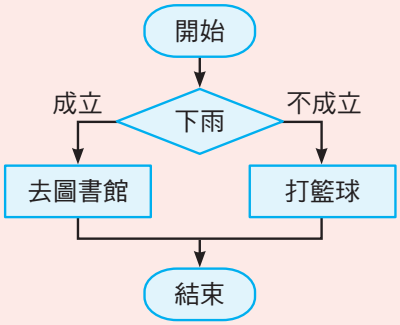
舉例

三種表示法的比較：

1. 求甲、乙兩數相加之總和。

文字	流程圖	虛擬碼
步驟 1. 輸入「數字甲」。 步驟 2. 輸入「數字乙」。 步驟 3. 將「數字甲」及「數字乙」相加，所得即為「總和」。 步驟 4. 輸出「總和」。		(1) input 甲 (2) input 乙 (3) sum = 甲 + 乙 (4) print sum

2. 依據是否下雨來決定行程。

文字	流程圖	虛擬碼
步驟 1. 判斷是否下雨。 步驟 2. 若下雨，就去圖書館。 步驟 3. 若未下雨，就去打籃球。		(1) if 下雨 (2) then 去圖書館 (3) else 去打籃球

32 資訊科技篇

2-2 流程控制結構

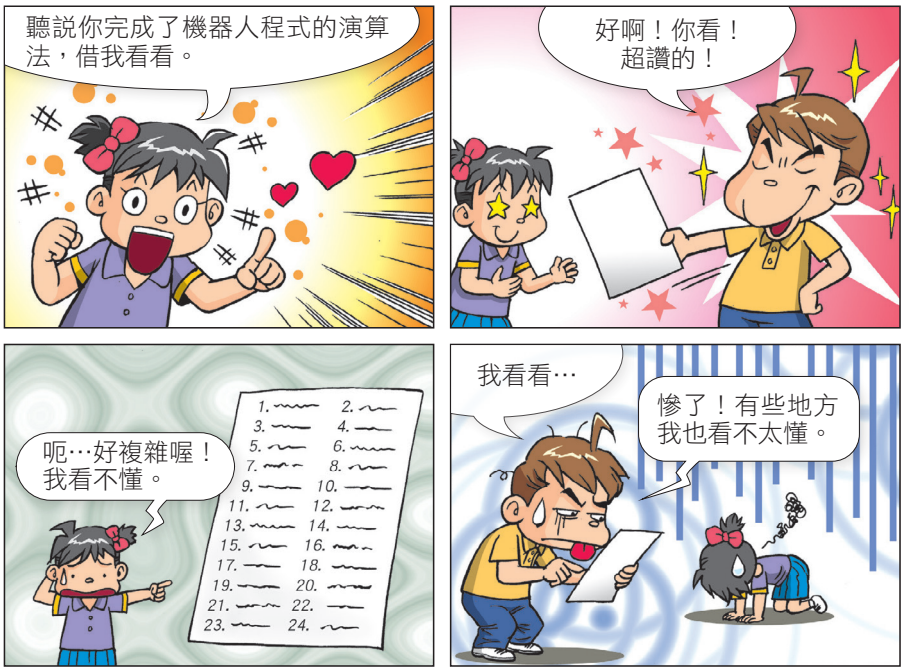


圖 1-2-6 設計演算法時，若只是隨意記錄所有的想法，當內容較多或較複雜時，將不易理解和維護。

演算法或程式如果沒有按照邏輯組織整理，當內容較多時，不僅不易理解，也很難進行維護（圖 1-2-6）。因此我們會以結構化的方式，利用循序、選擇、重複結構來進行程式設計（圖 1-2-7）。

圖 1-2-7 流程控制結構。

圖 1-2-6 「結構化」的重要性

如果每個人都用特定的方式表達，就可以節省解讀的時間，也較不易發生錯誤。

1 循序結構

循序結構是由上而下，依序按照一個指令、一個指令逐步執行，這是最基本的程式結構（圖 1-2-8）。

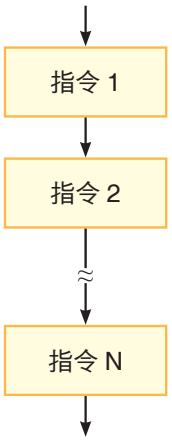
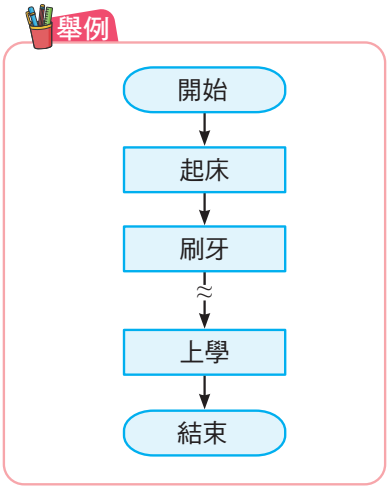


圖 1-2-8 循序結構的標準流程。



2 選擇結構

在執行過程中，若希望程式經由條件判斷的結果（成立／不成立），來決定接下來要執行何種指令，就可使用選擇結構（圖 1-2-9）。

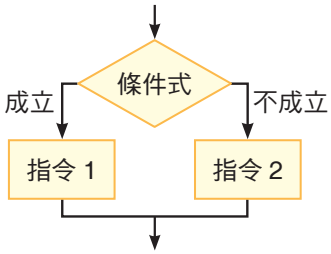
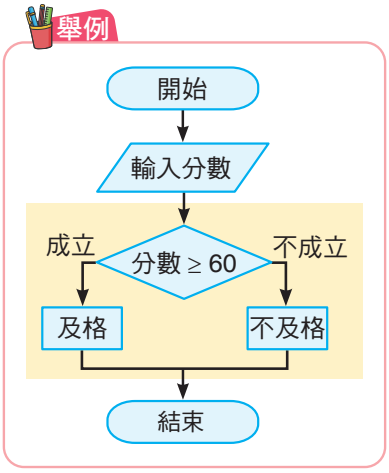


圖 1-2-9 選擇結構的標準流程。



34 資訊科技篇

3 重複結構

執行程式時，若希望讓某些指令重複執行多次，就可以使用重複結構（圖 1-2-10），不但能簡化程式，也讓程式更容易閱讀與理解。

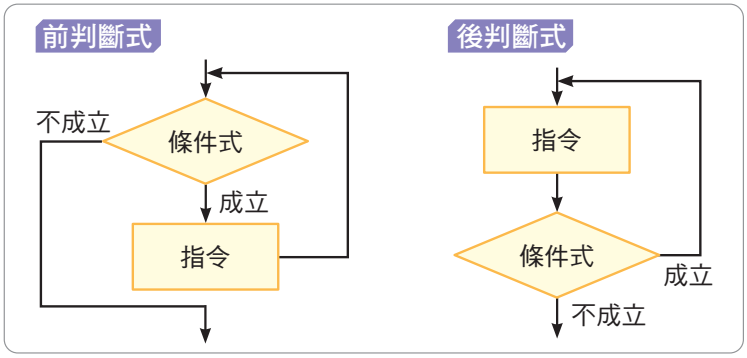
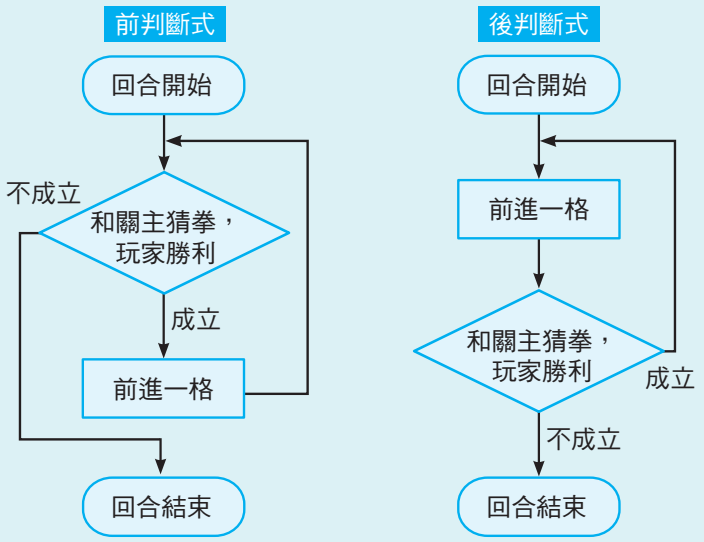


圖 1-2-10 重複結構的標準流程。前判斷式會先進行判斷再決定是否執行指令，所以可能完全不執行指令；後判斷式會先執行指令，然後再進行判斷，所以至少會執行一次指令。

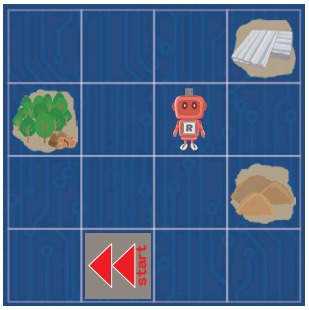
手腦並用

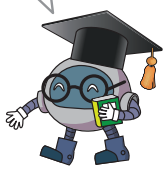
小潔設計了一個多人競賽的闖關遊戲，每位玩家輪流進行一個回合。想想看，下列兩種遊戲流程的執行結果有什麼不同？



引

請利用附件 1 的「機器人蓋城市」進行闖關活動，練習各種程式結構的概念。





參考解答

- 桌遊任務參考解答，請見 P.43-5

Ans 手腦並用

左圖「前判斷式」要猜拳勝利才會前進一格，若是第一次猜拳就失敗，則一格都沒有前進；右圖「後判斷式」會先前進一格後才開始猜拳，因此至少會前進一格。

引導建議

可帶領學生利用課本附件1，實際進行演練、操作，認識流程控制結構的運作過程。

2-3 流程圖設計實作

教學重點

帶領學生實際繪製流程圖，熟悉每個圖形所代表的意義。

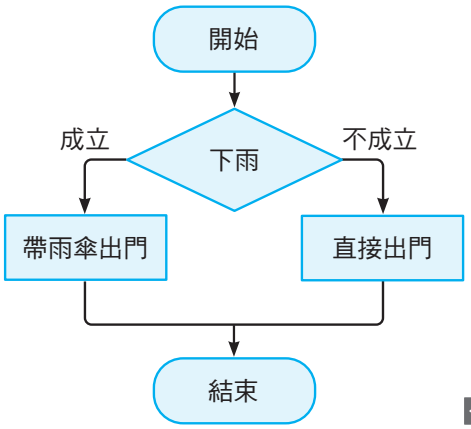


圖 1-2-11 流程圖繪製範例。

繪製流程圖時，我們可以手繪或使用流程圖軟體[1]來輔助，本節將以免費的線上軟體 Draw.io 為例，繪製「依據是否下雨來決定要不要帶雨傘出門」的流程圖（圖 1-2-11）。

註 也可以使用其他軟體如 Microsoft Word、Visio、Diagram Designer、Cacoo 等軟體來繪製。

Step1 進入 Draw.io

①連上 Draw.io 網頁

利用瀏覽器連結網址：
https://www.draw.io/

②設定中文介面

若畫面不是中文選單，可選 語言 ，設為 繁體中文 。

③選擇檔案儲存位置

此處以 裝置 為例（儲存於電腦中），依需求亦可選擇儲存於其他雲端位置。

④新增圖表

① 常見的流程圖軟體

(1) 免費軟體：
- ① Draw.io
- ② Diagram Designer
- ③ Cacoo

(2) 商業軟體：
- ① Microsoft Visio
- ② Microsoft Word
- ③ Visual Paradigm Online

⑤輸入檔案名稱

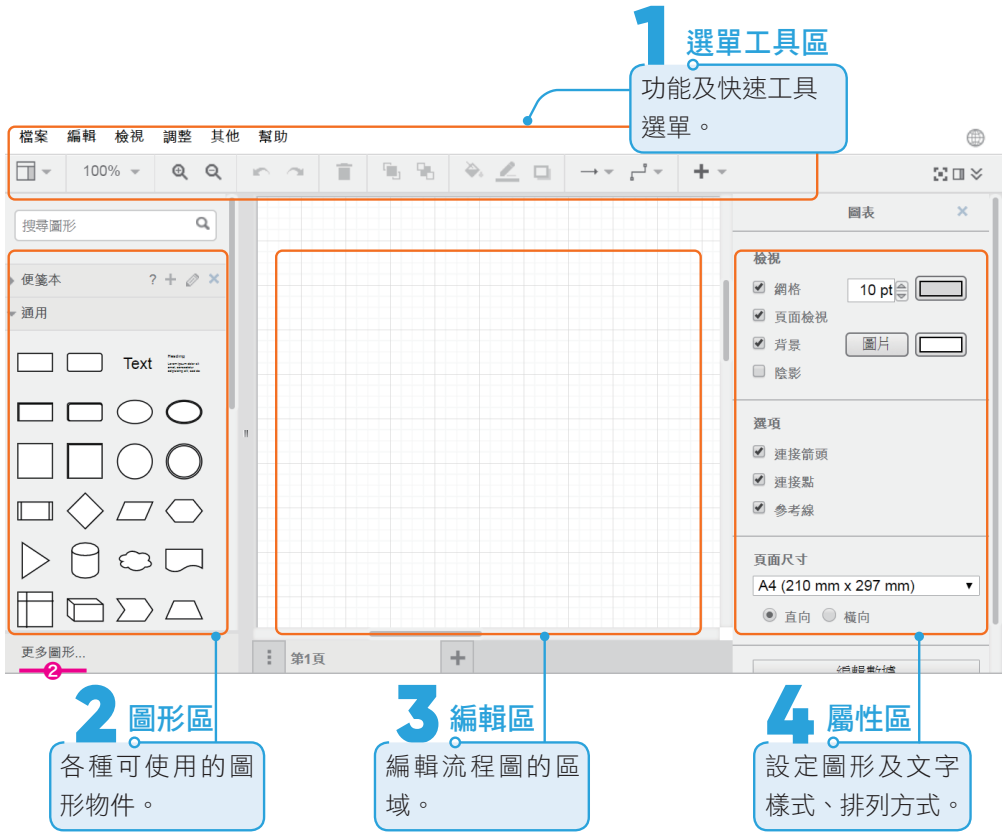
⑥選擇類別

繪製一般流程圖，選擇基本圖形即可。

⑦新增圖表

設定完成後，按新增。

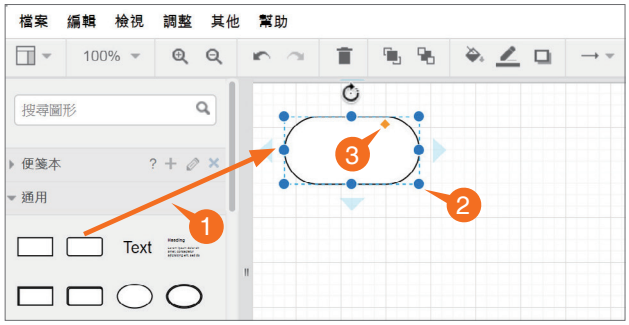
Step2 介面說明



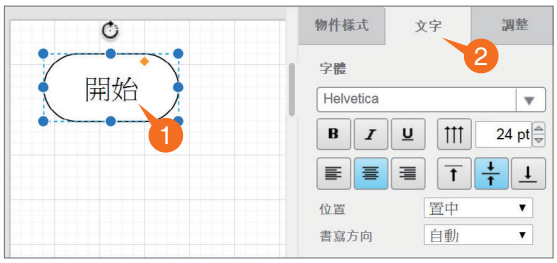
② 選擇其他圖形

若需要「基本圖形」以外的其他圖形時，可進入「更多圖形」中挑選。

Step3 物件編輯

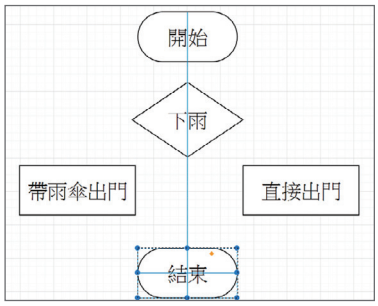


1. 繪製圖形
（以「開始符號」為例）
① 拖曳圖形到畫布上。
② 拉動物件周圍的藍色控制點，可以調整物件的大小。
③ 橘色控制點可調整圓角的弧度。



2. 輸入文字
① 圖形上雙擊滑鼠左鍵，即可輸入文字。
② 可在「屬性區」設定文字的格式。
③ 利用相同方式，完成其他圖形。[1]

Step4 物件對齊



1. 物件對齊[2]
拖曳物件至其他物件周邊時，會出現輔助線，讓物件可以彼此對齊。

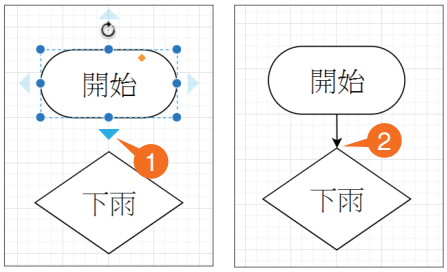
① 完成其他圖形

在製作流程圖時，我們常會重覆使用相同的符號，可以善用快速鍵複製（Ctrl＋C）及貼上（Ctrl＋V），提高繪圖效率。

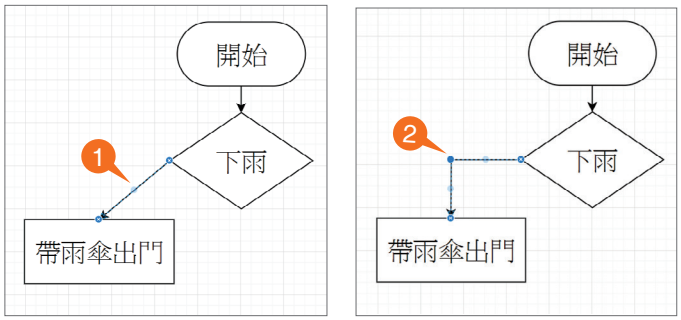
② 物件對齊

透過物件對齊，可以幫助我們快速地將物件置中對齊、左右對齊，或是平均物件間距等。

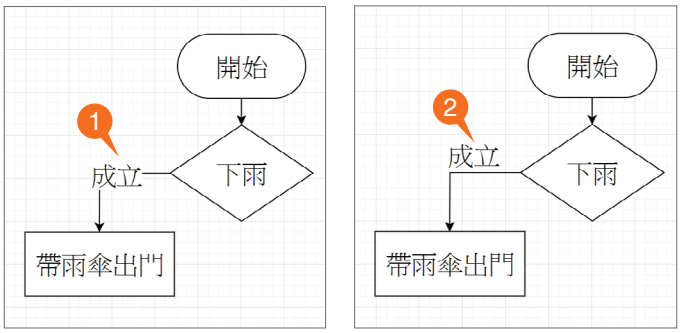
Step5 連接線



1. 連接物件
   ① 當滑鼠游標接近物件時，會出現三角形的連接點。
   ② 拖曳連接點至欲連接的物件，會產生連接線。



2. 調整連接線
   ① 拖曳連接線中間的淺藍色控制點。
   ② 調整連接線的彎曲角度及位置。



3. 輸入文字
   ① 在連接線上雙擊滑鼠左鍵以輸入文字。
   ② 將文字移動到連接線的上方。
   ③ 利用相同方式完成所有連接線。

Step6 儲存檔案

1. 儲存檔案
   繪製完成之後：
   ① 點擊 檔案 。
   ② 點擊 儲存 。

註 直接儲存，會以 xml 的格式存檔。

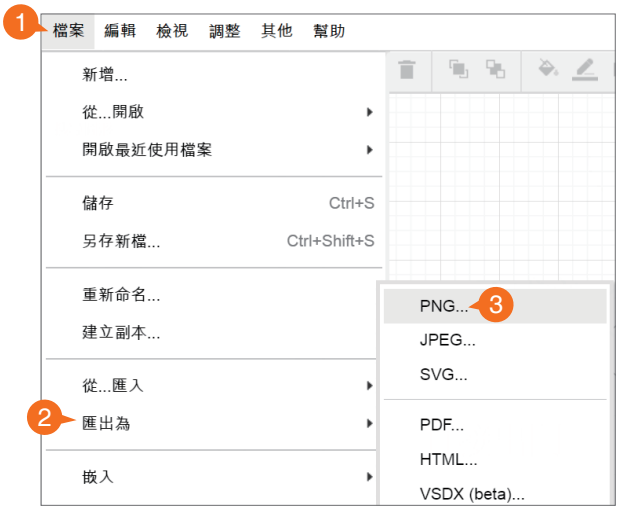
③ 輸入文字

雙擊滑鼠左鍵輸入文字時，文字會出現在連接線上，因此輸入完成後，要再將文字拖曳到適當位置。

④ 檔案儲存

若檔案已儲存成xml格式時，再次按下儲存會自動更名，不會覆蓋原始檔案。例如：原檔名為abc.xml，再次儲存時，檔案名稱會顯示abc(1).xml。

Step7 下載完成的圖檔



1. 匯出圖片
   ① 點擊 檔案 。
   ② 點擊 匯出為 。
   ③ 選擇圖片格式（以 PNG 為例）。

2. 設定圖片規格
   ① 調整 縮放 比例。
   ② 點擊 匯出 。

3. 下載檔案
   ① 輸入 檔案名稱 。
   ② 點擊 下載 。

① 圖片格式

(1) PNG
圖片的畫質不會因為壓縮而損害，但圖片的檔案較大。

(2) JPEG
圖片的檔案較小，但在壓縮的過程中，圖片會受到破壞。

第2章 學習重點

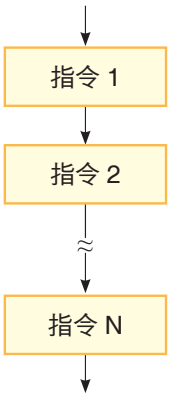
2-1 演算法簡介

1. 廣義來說，演算法就是解決問題的方法。
2. 在資訊科技領域中，演算法應具有 5 大特性：
   (1) 輸入：可有多個輸入資料，或是沒有輸入資料。
   (2) 輸出：必須至少有一個輸出結果。
   (3) 明確性：每個指令必須明確，不可模稜兩可。
   (4) 有限性：執行演算法，必須在有限步驟內結束。
   (5) 有效性：又稱可行性，演算法中的每個命令都必須是可執行的步驟，用紙筆也能推演完畢，以確定能解決問題。
3. 電腦程式都是遵循著演算法執行的。
4. 常見的演算法表達方式：
   (1) 文字：用一般語言文字來描述執行步驟。
   (2) 流程圖：用圖示符號來描述執行過程。
   (3) 虛擬碼：用類似程式語言的方式來描述執行步驟。

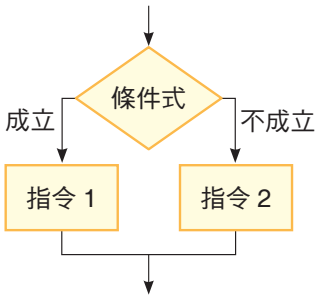
2-2 流程控制結構

1. 以結構化的方式進行程式設計，可方便理解與維護。
2. 結構化程式分為三類：

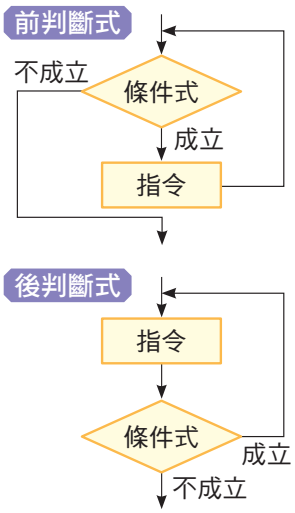
(1)循序結構：依指令先後順序由上而下，一個接著一個執行。



(2)選擇結構：依條件判斷的結果來執行指令，常應用於判斷、決策。



(3)重複結構：依需求重複執行特定指令。



2-3 流程圖設計實作

1. 我們可以手繪流程圖，或利用 Draw.io、Microsoft Word 等軟體來繪製。

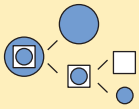
運算思維

西元 2006 年，周以真（Jeannette M. Wing）教授在美國電腦權威期刊《Communications of the ACM》中發表了《運算思維》，宣導「運算思維是利用電腦科學的基本概念進行問題解決、系統設計與人類行為理解的思維模式。」也就是說，當我們碰到問題時，可以利用運算思維來理解、分析問題，並發展出可能的解決方式。

運算思維簡介

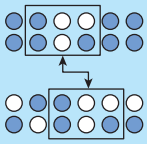
問題拆解

將複雜的大問題拆解為容易理解及處理的小部分，以方便我們檢查、解決或獨立設計。



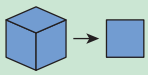
模式識別

從眾多問題中找到相似或共同的特徵，讓我們能利用相同的方法，有效地解決各個問題。



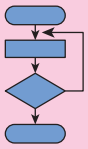
抽象化

忽視無關緊要的細節，將問題的重要關鍵特徵轉化成簡單明白的訊息。



演算法設計

針對問題的解決方案，制定出明確的執行步驟與規則。



引導建議

可用日常生活實例來引導學生思考、討論，例如：每天早上刷牙時，你知道自己進行了哪些思考步驟嗎？

1. 全家人的牙刷排在一起，哪一支是自己的牙刷？
2. 這麼多清潔用品，哪一條是牙膏？
3. 要刷多久時間？
4. 要施多少力在牙齒上？
5. 刷牙的順序是什麼……

當我們遇到複雜或困難的問題時，我們又該如何思考呢？現在要說明的運算思維是一種思考的技巧，利用特定的步驟，幫助我們解決問題。

1 問題拆解

例 認識自行車的工作原理時，可先分解成較小的幾個系統，再分別進行了解。

生活中的運算思維

周以真教授舉了一個生活化的例子來說明運算思維和我們日常生活的關係：「如果我們要做一頓飯，既要煮飯、炒菜、還要燉肉，而且不能菜都做好了，但飯還沒煮熟。我們該怎麼做呢？這三項任務都需要花費資源和時間，而我們就像是一部電腦，透過運算思維來分析問題、調配有限的資源、規畫解決方案，最終完成所有任務。」

周教授進一步指出：「運算思維是一種思考方法和基本技能，可以應用在生活、任何類型的科學和工程，甚至藝術工作上。因此，所有人都應該積極學習並使用，而非僅限於電腦科學家。」周以真教授於運算思維的見解與推廣，對推動電腦科學教育有重大貢獻。

周以真

學　　歷

麻省理工學院　電腦科學博士

研究領域

系統安全、軟體規範與驗證、
程式語言、分散式系統

經歷摘要

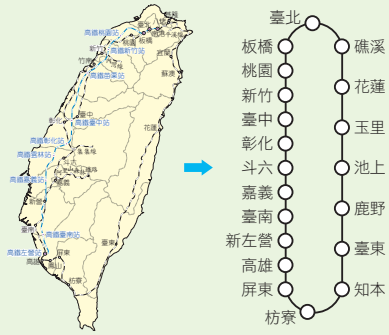
哥倫比亞大學　電腦科學教授
微軟研究院　副總裁
美國國家科學基金會——
電腦與資訊科學工程部　助理主任
卡內基美隆大學　電腦學院院長
麻省理工學院　電腦科學客座副教授

② 模式識別

例 製作兩種汽車模型時，我們可以觀察兩者之間如有共同特徵，就能一併處理，加快處理速度。

③ 抽象化

例 要繪製臺鐵「環島之星觀光列車」路線圖時，只畫出站名、先後順序，忽略其他細節。



教學活動步驟

週次	教學段落	教學主題	建議時間	說明重點
一	章扉頁	引言 頁22～23	2 分鐘	1. 說明現在的電腦功能很強大，是因為程式的運作結果；而程式要能正確運作，就要依循著「演算法」。
	2-1 演算法簡介	1. 演算法的特性 頁24～25	30 分鐘	1. 說明演算法就是解決問題的方法。 2. 說明演算法的步驟有順序性，不可任意省略或更動。 (1)以課本「買文具」為例說明。 (2)舉其他生活化的例子說明，例如： 「炒蛋」時，應該先將蛋打散再下鍋。如果沒有打散就下鍋就會變成荷包蛋。 3. 以小活動說明指令「明確性」的重要： (1)每個人準備一張A4回收紙 (2)指定一名學生為操作者，背對大家，一邊摺紙、一邊敘述動作 (3)其他學生照著指令摺紙 (4)進行幾個步驟之後，互相展示摺紙結果 (5)討論為什麼每個人摺出來的外形不一樣 4. 講解完後，再重複進行一次摺紙活動，讓學生學習如何下達明確的指令。 5. 介紹演算法的5大特性：輸入、輸出、明確性、有限性、有效性。
		2. 演算法與電腦 頁26	13 分鐘	1. 說明電腦功能強大的背後，主要依賴好的演算法。例如：修圖app要把照片裡的眼睛變大、把臉變小，而照片裡的哪些部位是眼睛？哪些是臉？這些都是電腦依循演算法的步驟執行的結果。 2. 延伸學習： (1)說明演算法沒有標準的答案，只要能解決問題就可以成立。 (2)同一個問題，可能有許多不同的演算法。 (3)演算法的基本要求是能正確解決問題，而演算法的好壞，通常可以用執行效率高低、耗費資源多少來比較。 (4)說明想要設計出好的演算法，必須累積大量的知識與經驗。

週次	教學段落	教學主題	建議時間	說明重點
二	2-1 演算法簡介	3. 演算法的表達 ①文字 頁27	5 分鐘	1. 認識以文字表達演算法的方式。 2. 說明文字演算法不易閱讀，若描述複雜的步驟會顯得冗長，且不同人的解讀可能有誤差。
		3. 演算法的表達 ②流程圖 頁28～31	30 分鐘	1. 流程圖的優點： (1)介紹流程圖主要是利用圖形和箭頭來呈現步驟。 (2)引導討論，在哪裡還有看過流程圖？ (3)與「文字演算法」比較，說明流程圖的步驟較易讀、易懂。 2. 繪製流程圖： (1)說明流程圖的繪製原則。 (2)介紹常用流程圖的符號。 (3)利用手腦並用，讓學生熟悉流程圖符號的使用。 3. 流程圖的適用範圍： 說明如果要畫複雜的流程時，可利用副程式的方式呈現，讓流程更清晰易理解。
		3. 演算法的表達 ③虛擬碼 頁32	10 分鐘	1. 說明「虛擬碼」的呈現方式。 2. 虛擬碼不是程式語言，但是可以利用類似程式語言的方式來表達演算法。 3. 比較三種表達方式的不同。
三	2-2 流程控制結構	引言 頁33	5 分鐘	1. 以生活化的例子說明「結構化」的重要性，如果每個人都用特定的方式表達，就可以節省解讀的時間，且不易發生錯誤。 2. 先簡要介紹有3種流程結構，後續會逐一說明。
		1. 循序結構 頁34	5 分鐘	1. 認識循序結構： 依指令先後順序由上而下，一個指接著一個執行，這也是最基本的結構。
		2. 選擇結構 頁34	5 分鐘	1. 認識選擇結構： 通常我們口語中提到「如果…那麼…」、「如果…那麼…否則…」，就是選擇結構的應用。
		3. 重複結構 頁35	5 分鐘	1. 說明各種重複結構，可以讓程式變得更為精簡。 2. 重複結構中，也應用到「選擇結構」，用以判斷現在要重複某些指令，或是執行接下來的指令。 3. 認識前、後判斷式： (1)前判斷式：先條件判斷。 →可能不執行指令。 (2)後判斷式：先執行指令。 →一定會執行該指令。

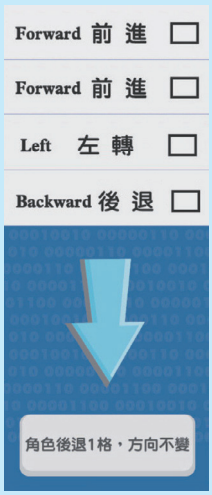
週次	教學段落	教學主題	建議時間	說明重點
（續）三	2-2 流程控制結構	3. 重複結構 頁35	5 分鐘	4. 手腦並用： 比較前、後判斷式的差別。 (1)前判斷式：可能會前進 0 格。 如果第一次猜拳就輸了，完全不前進。 (2)後判斷式：最少會前進 1 格。 每個回合中，即使第一次猜拳就輸了，還是會前進 1 格。
		【習作】 第2章 實作活動 【習作】 頁11	20 分鐘	1. 練習「等第制」流程的設計與繪製。
四	2-2 流程控制結構	桌遊： 機器人蓋城市 課本附件1	45 分鐘	1. 說明附件1桌遊的規則，介紹地圖與物件。 2. 說明控制卡牌分類，可分別對應「循序、選擇、重複」結構。 3. 說明出牌與行動規則。 (1)遊戲採回合制，各回合出牌數不限。 (2)出牌方式為上下疊合，露出標頭文字。 (3)指令執行方式：由上而下，逐一執行。 (4)每回合結束時，若停留在資源點上，即可獲得該點上的資源。 (5)若走出地圖外或障礙物上，代表闖關失敗。 4. 特別說明重複結構使用方式。 5. 複習三種流程結構，並引導學生完成三種流程結構的「小試身手」題目。 6. 讓學生自行完成「進階挑戰」、「綜合挑戰」的題目，並讓學生分享自己的解題方式。 7. 讓學生自製關卡，分組進行遊玩。
五	2-3 流程圖設計實作	活動說明 頁36	10 分鐘	1. 課本中，將利用免費的線上軟體Draw.io來完成活動。 2. 若受限於網路條件，亦可使用Microsoft Word或其他文書軟體來繪製。
		軟體操作說明 頁36～40	20 分鐘	1. 說明Draw.io的基本操作模式。 2. 可依課本範例練習繪製流程圖，或繪製習作11頁的流程圖。

週次	教學段落	教學主題	建議時間	說明重點
（續） 五	科技廣角	運算思維 頁42～43	15 分鐘	1. 引導： (1)詢問學生：我們每天都在思考問題、解決問題。例如每天早上刷牙時，你知道自己進行了哪些思考步驟嗎？ ①一整排牙刷，哪一支是自己的？ ②這麼多清潔用品，哪一條是牙膏？ ③要刷多久時間？ ④要施多少力在牙齒上？ (2)過程中好像很複雜，但我們平時刷牙時，並沒有意識到以上的思考歷程。 (3)當我們遇到複雜的問題時，可以用運算思維幫助我們解決問題。 2. 介紹運算思維： (1)問題拆解：碰到複雜的大問題時，我們往往會因為無從下手而手足無措。此時，我們可以先分析問題，將大問題拆解成多個小問題，再針對小問題進行處理，有助於解決整體問題。 例如：認識自行車的工作原理時，可先分解成較小的幾個系統，再分別進行了解。 (2)模式識別：當我們處理問題時，可以在各個小問題之間，發現相同或類似的特徵，這些特徵就稱為模式，方便我們利用相同或類似的方式處理。因此我們要發揮觀察力，找到的模式越多，就能越快、越有效的處理問題。 例如：要同時製作兩種汽車模型，可以觀察到兩者之間有共同特徵（例如：都有輪胎、後視鏡等），我們就可以一併處理，加快處理速度。 (3)抽象化：每個問題都有許多大大小小的相關細節，抽象化是只專注於問題的重要特徵，忽視無關緊要的小細節，並將關鍵特徵簡化成簡單明白的訊息，從而建立一個解決問題的表示法。 例如：要繪製臺鐵「環島之星觀光列車」路線圖時，只畫出站名、先後順序，忽略其他細節。 (4)演算法設計：依照2-1節所學的，制定清楚、明確的解決問題步驟。 3. 介紹周以真教授，鼓勵女同學也可以認真投入資訊科技領域。

桌遊任務參考解答

循序結構①：

前進2→左轉→後退

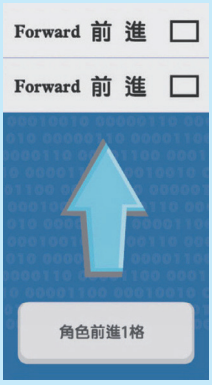


循序結構②：

回合1 迴轉→前進2→左轉→前進 / 回合2 前進2

回合1：

回合2：



選擇結構①：

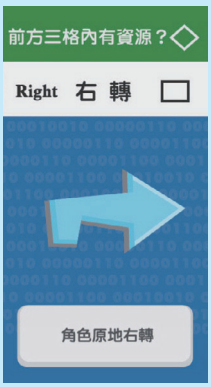
前方一格有障礙？（左轉）→前進→右轉→前方三格內有資源？

選擇結構②：

回合1 前方三格有資源？（否）→右轉

回合2 前進2→前方一格有障礙？（右轉）→前進

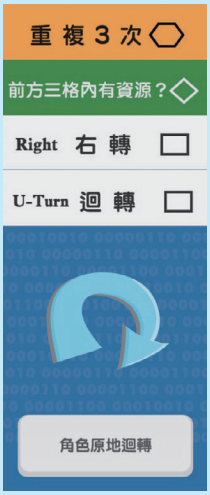
回合1：



回合2：

重複結構①：

重複3次（前方三格內有資源？→右轉→迴轉）

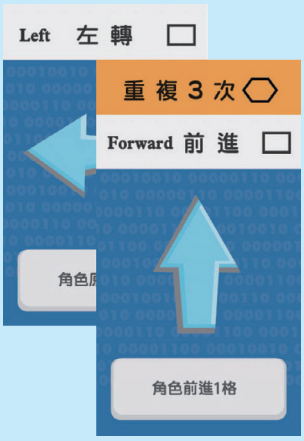


重複結構②：

回合1 左轉→重複3次（前進）

回合2 後退→重複2次｛左轉→重複2次（前進）｝

回合1：



回合2：

綜合挑戰①：

重複2次（左轉→前進）→重複3次（前方三格內有資源？→右轉）

綜合挑戰②：

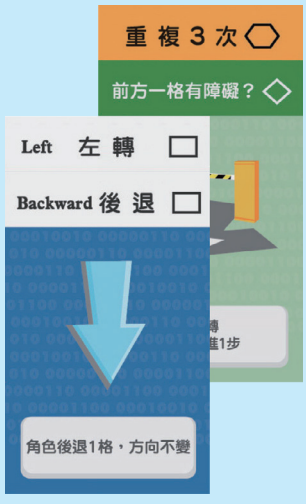
回合1 重複2次（前進→右轉→前進→左轉）

回合2 重複3次｛前方一格有障礙？（右轉）｝→左轉→後退

回合3 前方三格內有資源？

回合1：

回合2：



回合3：