

單元

5

電腦與



電腦解題與演算法

- 13-1 電腦可以協助解決哪些問題
- 13-2 電腦解題簡介
- 13-3 電腦解題規劃－演算法
- 13-4 認識資料結構



※電腦解題程序與實作

- 14-1 電腦解題程序
- 14-2 電腦解題實作



☑同場加映

- 水平式邏輯思考
- 變數的概念
- 電腦解題工具

問題解決



電腦解題與演算法

電腦每秒可執行超過十億次以上的運算，是幫助人類解決問題的好工具。你知道在哪些領域適合使用電腦來解決問題嗎？要如何善用電腦來協助我們解決問題呢？本章將介紹「電腦解題」與「演算法」的相關概念，培養同學邏輯思維及運用電腦解決問題的能力。

13-1 電腦可以協助解決哪些問題

早期電腦主要是應用在大量資料的計算上，而現今電腦已被廣泛地應用在各個領域，例如**科學研究**、**軍事發展**、**醫學實驗**、**氣象預測**、**商業計算**、**影音休閒**、**生活應用**（圖13-1）……等，以下列出電腦在這些領域常見的應用實例：

- **科學研究**：太空探測、行星軌道計算、替代能源開發等。
- **軍事發展**：彈道計算、飛彈導航、戰略模擬等。
- **醫學實驗**：基因研究、遠距醫療、電腦斷層掃描等。
- **氣象預測**：颱風預測、地震預測、雨量預測等。
- **商業計算**：金融稅務、風險精算、投資分析等。
- **影音休閒**：電腦遊戲、電腦繪圖、動畫繪製等。
- **生活應用**：網路訂票、網路購物、自動櫃員機、網路資料搜尋等。

這些都靠我！



圖13-1 電腦解決問題的應用範例



(<https://irs.thsrc.com.tw/IMINT>)

網路訂票

(<http://www.google.com.tw>)

網路資料的搜尋

(<http://photino.cwb.gov.tw/>)

颱風預測

13-2 電腦解題簡介

要利用電腦來解決問題，有2項重要的前置作業，首先要利用邏輯性的思考，找出解決問題的方法；接著再以循序漸進的方式，規劃出解決問題的步驟，分別說明如下。

13-2.1 垂直式邏輯思考

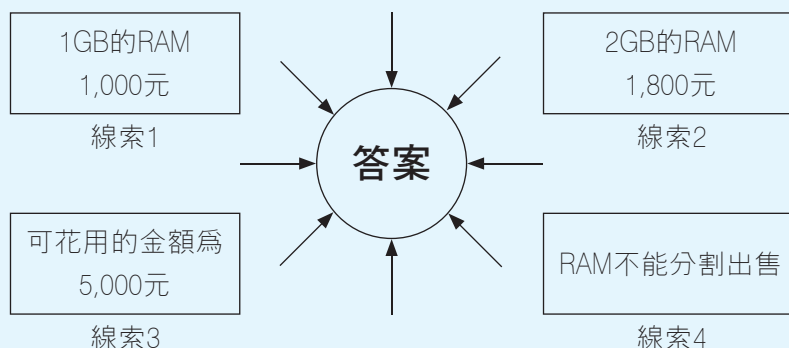
垂直式邏輯思考（Vertical thinking）又稱為「收斂式思考」，這種思考方式是透過反覆的思考與求證（如分析、評估、判斷、比較）等過程，以抽絲剝繭的方式，找出解決問題的方法。

垂直式邏輯思考適合應用在科學研究、數學運算及電腦解題等邏輯性思考的問題上；當我們要使用電腦來解決問題時，經常需要使用這種思考方式。圖13-2為一個使用垂直式邏輯思考來求解的簡單範例。

問題

某家商店1GB的RAM 1,000元，2GB的RAM 1,800元，請問如何用5,000元買到最多容量的RAM？

1. 根據問題，可以得知以下4個線索：



2. 計算1GB與2GB RAM的單位價格，比較何者便宜？

容量	單位價格	費用
1GB	$1,000/1GB = 1,000$	較貴
2GB	$1,800/2GB = 900$	較便宜

透過分析比較
找答案

3. 因為2GB RAM的單位價格較便宜，所以應盡量多買2GB的RAM。

答案：買2GB的RAM 2條、1GB的RAM 1條，共花用4,600元。



圖13-2 垂直式邏輯思考問題的範例



水平式邏輯思考

另一種人類的思考方式稱為**水平式邏輯思考**（Horizontal thinking），又稱為「發散式思考」。它是一種不受既有事物或觀念拘束的思考方式，常用來激發創意或尋求新的見解（圖13-3）。

水平式邏輯思考較適合應用在藝術創作、創意研發等領域。生活中有些問題，使用水平式思考，會比使用垂直式思考更合適。例如：怎麼設計出一句響亮的廣告詞？如何推銷賣不出去的商品……等。

問題

某家商店1GB的RAM 1,000元，2GB的RAM 1,800元，請問如何用5,000元買到最多容量的RAM？

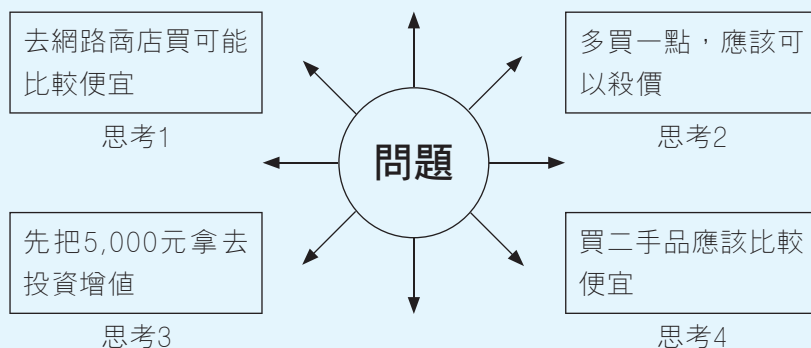


圖13-3 水平式邏輯思考問題的範例

換個角度
想答案



以下有兩道題目，請同學運用水平式邏輯思考，找出最符合題目邏輯的答案^註。

Q₁ 世上最大的馬鈴薯長在哪裡？

參考答案：土裡

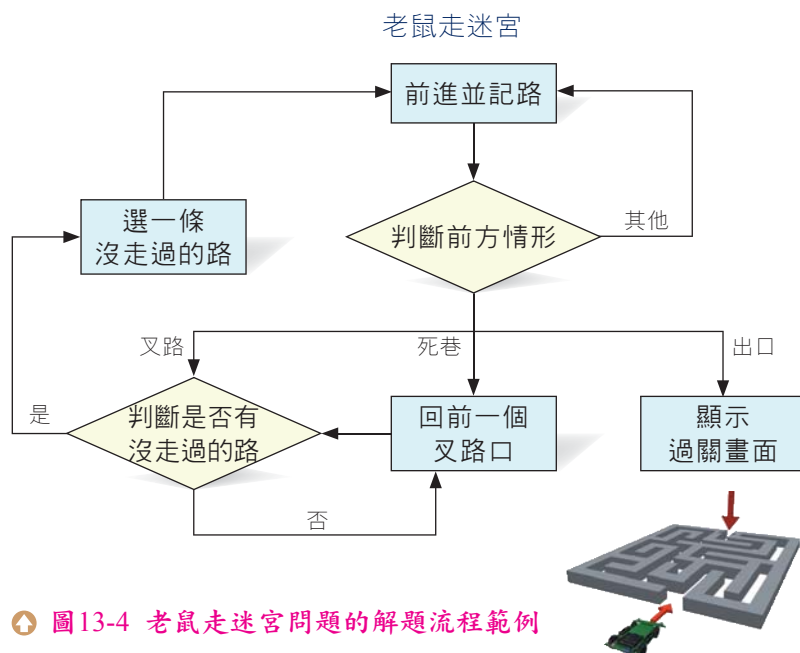
Q₂ 什麼地方所有男人都一樣帥？

參考答案：黑暗中

13-2.2 循序漸進的解題流程

電腦不像人類具有獨立思考的能力；我們必須先透過「垂直式邏輯思考」來找出解決問題的方法，再規劃出解決問題的明確指令或步驟，電腦才能依循這些步驟來解題。圖13-4是將老鼠走迷宮問題的解題步驟以流程來表示的範例。

如果我們所要解決的問題較為龐大或複雜，可採循序漸進的方式，先將問題分割成幾個較小的問題，再針對每一個較小的問題，一一規劃出解題的步驟，以有系統地解決問題。



- ___ 1. 下列哪一種思考方式，最適合用在電腦解題的領域上？ (A)水平式思考 (B)跳躍式思考 (C)發散式思考 (D)垂直式思考。
- ___ 2. 下列有關垂直式邏輯思考的敘述，何者正確？ (A)又稱發散式思考 (B)是透過思考與求證，找出問題的答案 (C)是不受既有觀念拘束的思考方式 (D)適合用在創意研發的領域。
3. 有肉粽、春捲、肉包、水餃四道菜，已知：
 - (1) 第一道菜是肉粽或水餃
 - (2) 第二道菜不是春捲，第三道菜不是肉包
 - (3) 第四道菜是肉包或水餃
 - (4) 第三或第四道菜是水餃
 請問，肉粽、春捲、肉包、水餃各是第幾道菜？



13-3 電腦解題規劃－演算法

假設你和3位朋友要從新竹到阿里山玩兩天一夜，總預算為8,000元（不含餐費與個人花費），要如何規劃行程，才不會讓費用超出預算呢？試試看用電腦來查詢資訊及試算費用，以作為旅遊規劃的參考（圖13-5）。

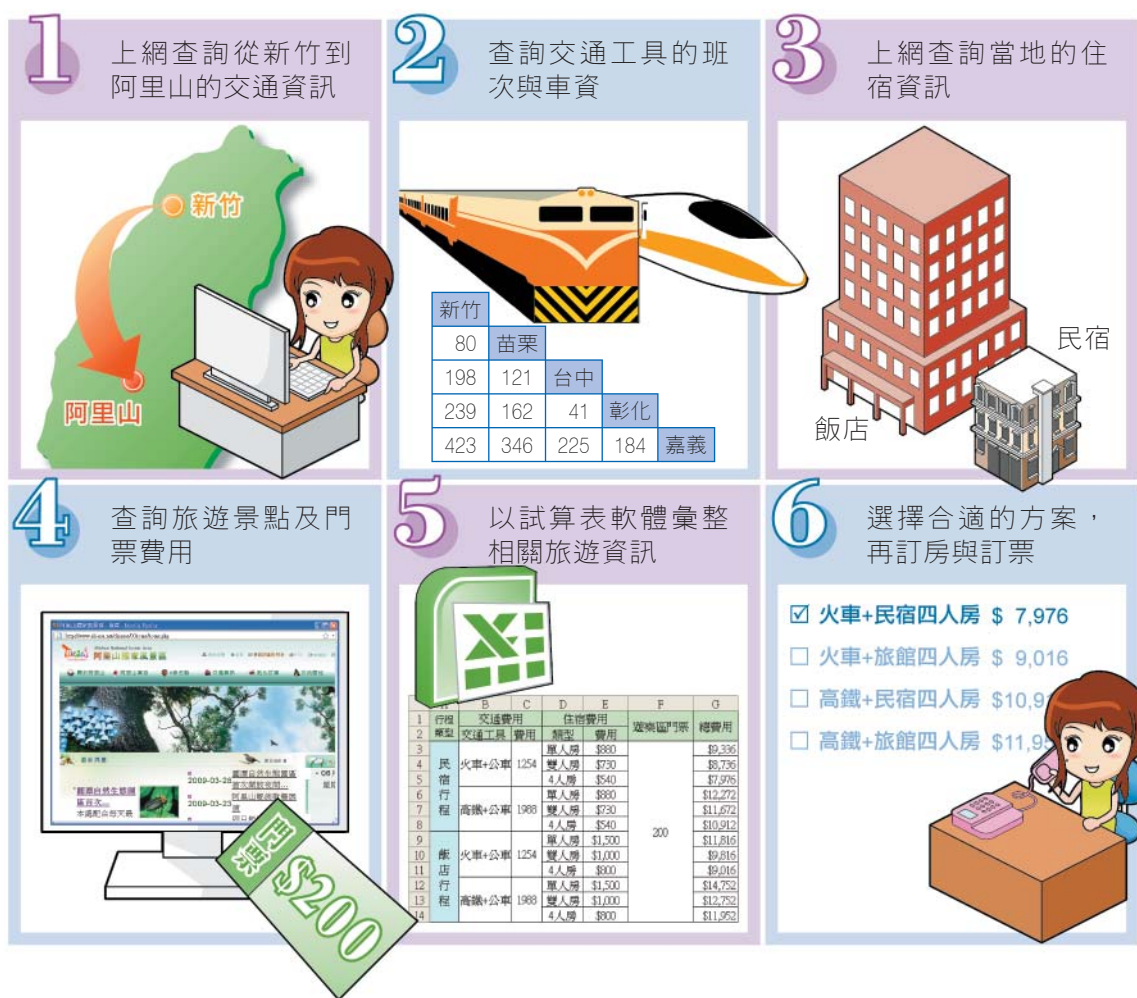


圖13-5 規劃旅遊行程的示意圖

很多人都需要做類似的旅遊規劃，因此有許多網站為了商機，會提供規劃旅遊行程的程式，我們只要輸入來回地點和預算，電腦便會自動提供一些適合的方案供我們選擇。

程式設計師在開發這種程式時，必須事先規劃出明確的步驟，才能依步驟來解決問題。以下將介紹規劃電腦解題步驟時用的「演算法」，讓同學瞭解其特性、表示法及基本結構。

13-3.1 演算法簡介

演算法（algorithm）是一組用來解決特定問題的有限指令或步驟，我們可以依循這些指令，在有限的步驟內逐步解決問題或完成特定工作。演算法中的每一個步驟都必須非常明確，不可以模稜兩可。

一個好的演算法，必須具備如表13-1所示的5項特性。

表13-1 演算法必須具有的5項特性

特性	說明	圖中編號
輸入 (input)	要有輸入資料，但並非絕對必要	A
輸出 (output)	要有一個以上的輸出資料	B
有限性 (finiteness)	必須在有限的處理步驟內得到結果	C
明確性 (definiteness)	每個步驟都必須明確，不能有模稜兩可的情況	D
有效性 (effectiveness)	每個步驟都必須是可執行的	E

13-3.2 演算法實例

演算法不僅可用來解決數學或科學上的問題，也可應用在日常生活中，以解決特定的問題或完成特定的工作。例如圖13-6所示的演算法，是要加總考生各科的成績，並將計算所得的總分連同各科成績輸出。

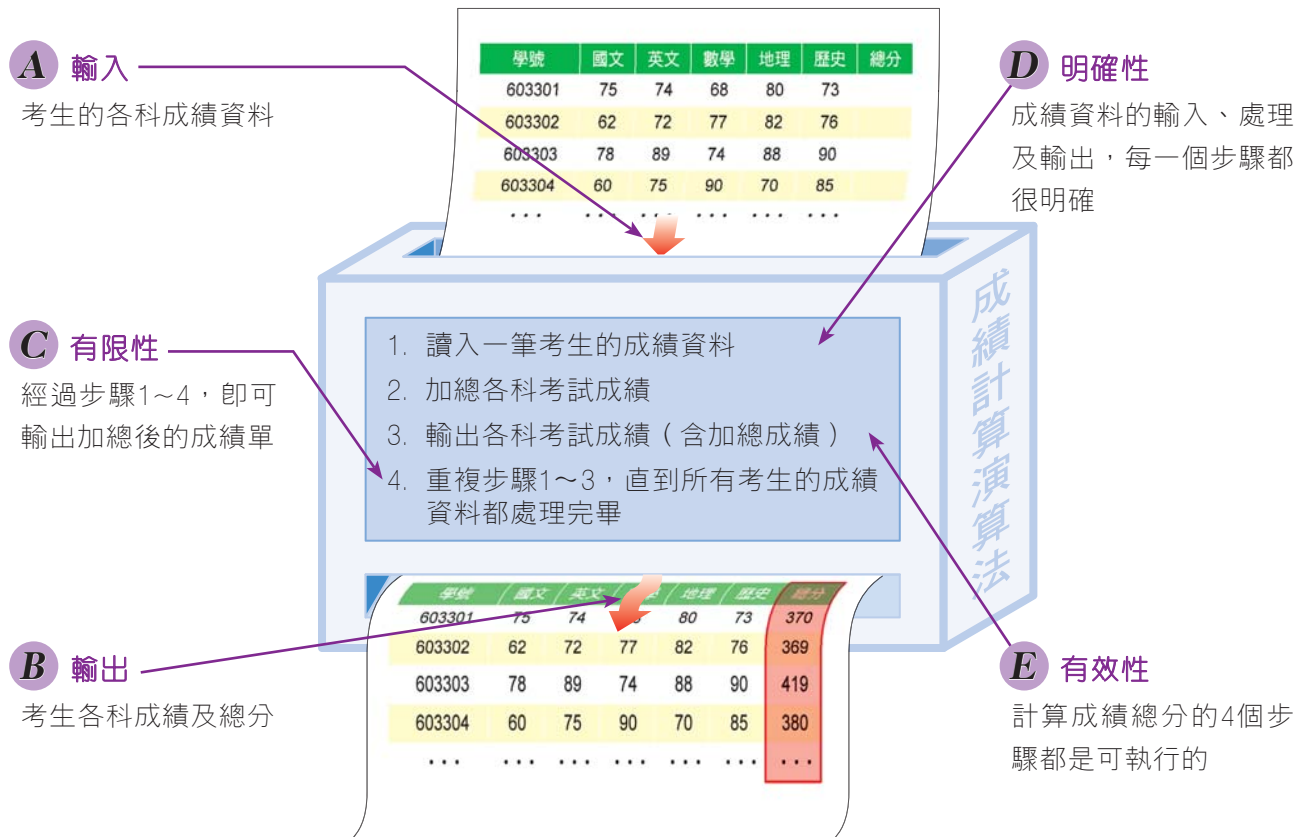


圖13-6 成績計算演算法



13-3.3 演算法的表示方法

在設計演算法時，我們常以**流程圖表示法**與**敘述表示法**兩種方式來表示處理的步驟，分別說明如下。

流程圖表示法

流程圖（flowchart）是使用簡明的圖示符號來表達解決問題步驟的示意圖，表13-2是常用的流程圖符號及其所代表的意義。

表13-2 常用的流程圖符號與說明

符號	代表意義	作用
	開始或結束	表示流程圖的開始或結束
	螢幕	表示將資料輸出於螢幕上
	輸入或輸出	表示資料的輸入或輸出
	處理符號	表示執行某些工作
	決策或判斷	表示以符號內的條件式作判斷，決定執行的流向
	迴圈符號	設定迴圈變數的初值與終值
	流向符號	表示程式的執行方向和順序
	連接符號	表示流程圖的出口或入口
	列印符號	表示資料由印表機輸出
	磁碟符號	表示由磁碟輸入或輸出資料

圖13-7為使用流程圖表示法來描述日常生活中使用ATM轉帳的流程。

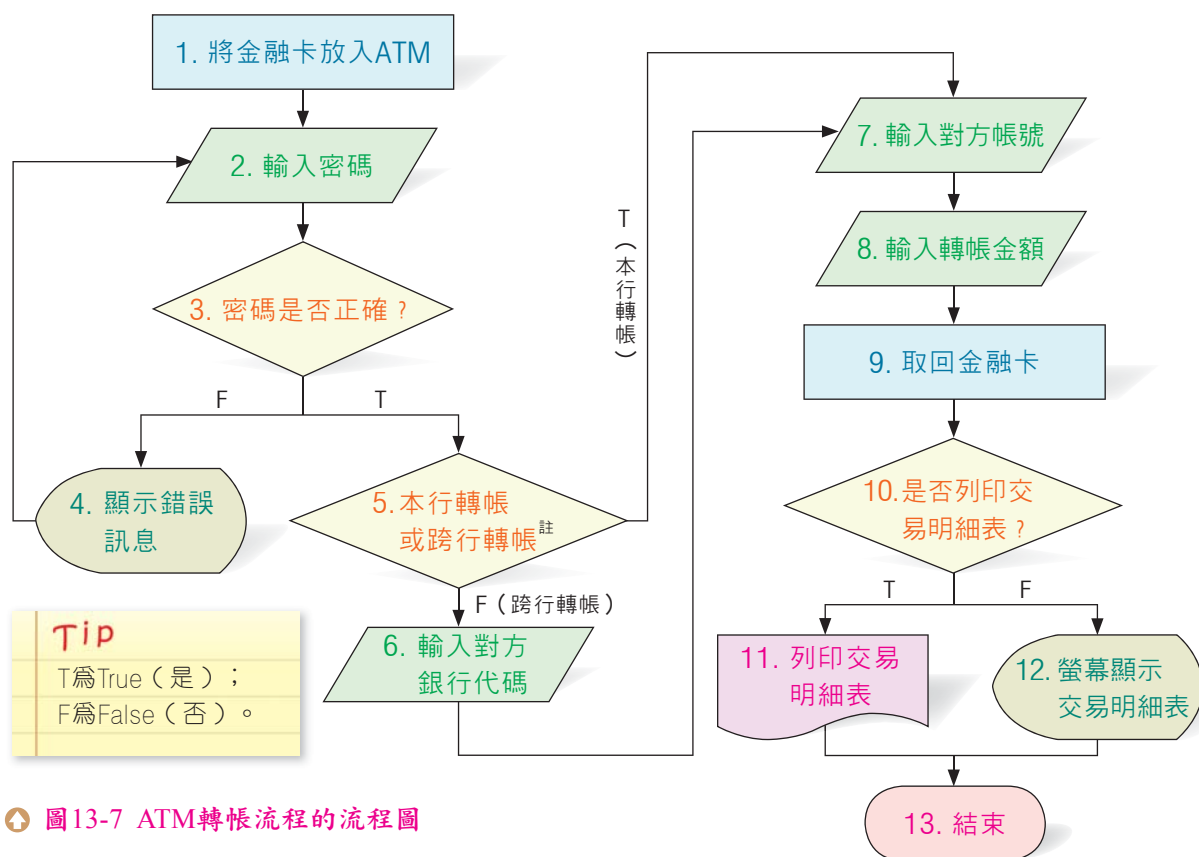


圖13-7 ATM轉帳流程的流程圖

敘述表示法

敘述表示法是使用虛擬碼，來表達演算法的處理步驟。**虛擬碼**（pseudo code）是一種以簡潔扼要的文字，來設計程式邏輯的工具。許多程式設計師為了方便直接參照虛擬碼來撰寫程式，會使用類似程式語言（如C）的語法，來撰寫虛擬碼。圖13-8是將圖13-7中的演算法以敘述表示法來描述處理的步驟。

1. 放入金融卡	7. 輸入對方帳號
2. 輸入密碼	8. 輸入轉帳金額
3. 判斷密碼是否正確，若正確，跳至步驟5；若錯誤，跳至步驟4	9. 取回金融卡
4. 在螢幕顯示錯誤訊息，並跳至步驟2	10. 判斷是否需要列印交易明細表，若需要，跳至步驟11；若不需要，跳至步驟12
5. 判斷是本行轉帳或跨行轉帳，若為「跨行轉帳」，跳至步驟6；若為「本行轉帳」，跳至步驟7	11. 列印交易明細表，並跳至步驟13
6. 輸入對方銀行代碼	12. 在螢幕顯示交易明細
	13. 結束流程

圖13-8 ATM轉帳流程的虛擬碼

註：我們透過網路購物時，若賣家的帳戶與我們的帳戶分屬於不同銀行，就必須使用「跨行轉帳」的方式來匯款。





13-3.4 演算法的基本結構

演算法包含**循序**、**條件**及**重複**等3種基本結構，一個用來解決問題的演算法，常使用到這3種基本結構，以下將分別介紹。

循序結構

循序結構（sequence structure）是一種由上而下依序執行的控制結構（圖13-9）。當電腦要依序執行動作時，就會使用循序結構。例如要利用電腦輸出「小蜜蜂」音樂，就必須使用循序結構來處理。

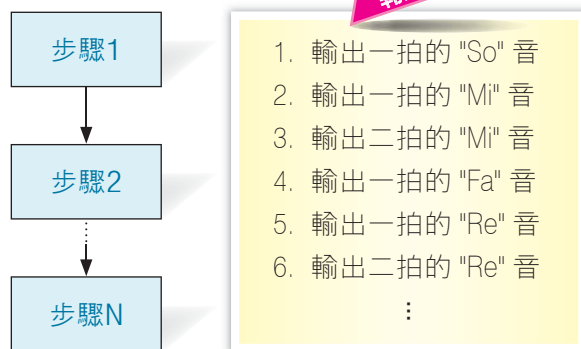


圖13-9 循序結構

條件結構

條件結構（selection structure）是一種依照特定的條件或測試的結果，來決定不同執行路徑的控制結構（圖13-10）。當電腦需要處理抉擇的問題時，就必須使用條件結構。例如要判斷數值為奇數或偶數，必須使用條件結構來處理。

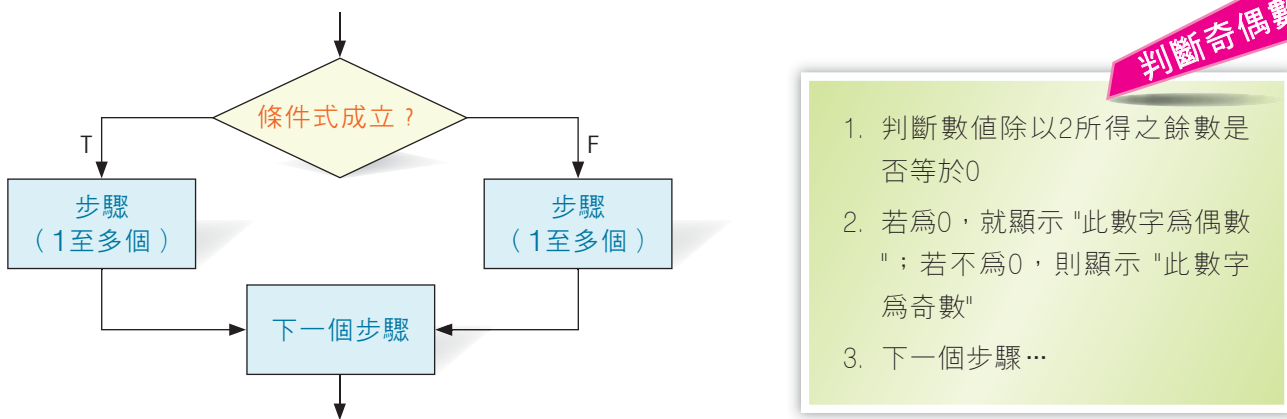


圖13-10 條件結構

重複結構

重複結構（repetition structure）是一種反覆執行解決問題的步驟，直到特定條件出現才停止執行的控制結構（圖13-11）。當電腦需要處理反覆執行的問題時，便可以使用重複結構來處理。例如要累加1至10的總和，就可以使用重複結構來處理。

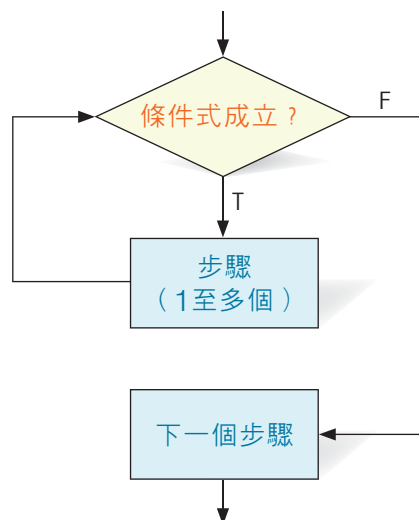


圖 13-11 重複結構

累加1~10

1. 設定數值 $i = 1$ ， $sum = 0$
2. 判斷 i 值是否 ≤ 10 ，若為是，跳至步驟3；若為否，跳至步驟5
3. 將 i 值加到 sum 中
4. 將 i 值加1，並跳至步驟2
5. 顯示 sum 值

如果要累加1至1,000的總和，要如何更改以上的虛擬碼呢？



變數的概念

變數 (variable) 是指一種內容不固定的資料項目，這種資料項目的值通常會隨著程式的執行而改變。例如圖13-11的虛擬碼中， i 與 sum 即是變數。

在程式語言中，變數通常是由名稱、儲存空間、儲存位址、資料型別及內容（即變數的值）等5項內涵所組成（圖13-12），說明如下：

- ✿ **名稱**：每個變數都須有一個專用的名稱，以資識別。
- ✿ **儲存空間**：變數在記憶體中所佔用的儲存空間大小。
- ✿ **儲存位址**：在記憶體中，變數內容開始存放的起始位址。
- ✿ **資料型別**：變數儲存空間允許存放的資料型別，例如整數、實數、字串等。
- ✿ **內容**：在變數儲存空間中所存放的資料，就是變數的內容，也稱為變數的值。

1. 名稱：sum
2. 儲存空間：2個bytes
3. 儲存位址：從記憶體的 $(5A34)_{16}$ 位址開始存放資料
4. 資料型別：整數
5. 內容：60

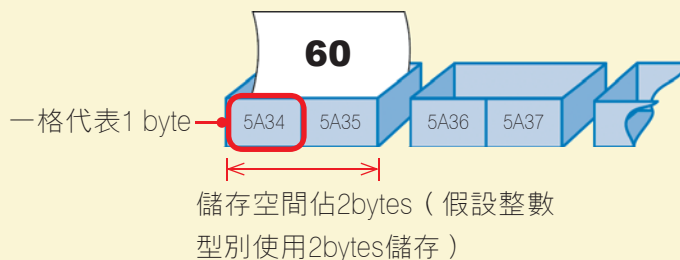


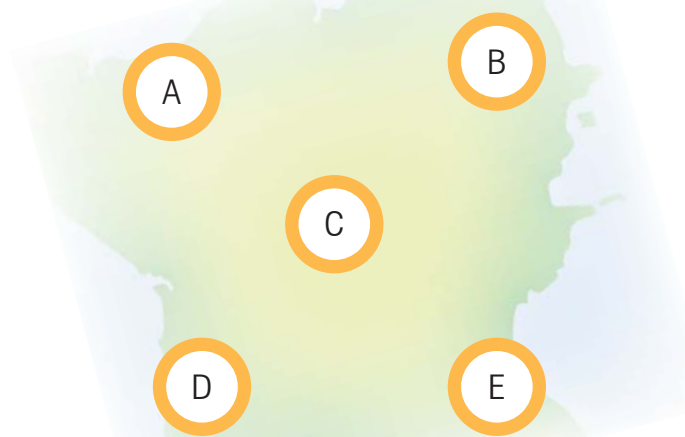
圖 13-12 整數變數內涵示意圖

13-3.5 解題策略

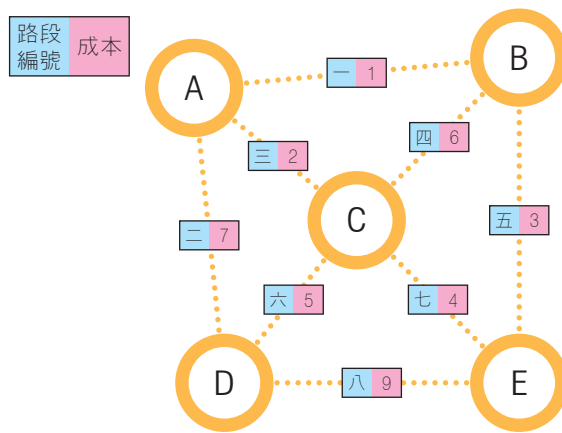
在規劃演算法之前，應該先選擇一個合適的解題策略，再進行演算法的規劃。不同的解題策略，適合用來解決不同的問題，以下介紹3種較常見的解題策略。

暴力法

暴力法（Brute Force）是電腦解題策略中最簡單也是最原始的方法，它的概念是透過逐一比對或計算，一一嘗試，以找出最佳解。某縣內有A~E五個新景點（圖13-13），要規劃興建馬路，但因經費考量，只能興建4條馬路來連接所有景點。假設可興建馬路的路段與所需成本如圖13-14所示，請問應該在哪幾條路段上興建馬路，才能使成本最低呢？



↑ 圖13-13 景點位置示意圖



↑ 圖13-14 馬路興建成本示意圖

本題共有8條可能路段，每條路段的狀態可分為 "選" 與 "不選" 2種，因此可能的組合共有 $2^8 = 256$ 種。如果使用暴力法策略來解題，我們只要一一列出這256種可能組合（表13-3），即可找出能連接所有景點，且成本最低的組合。

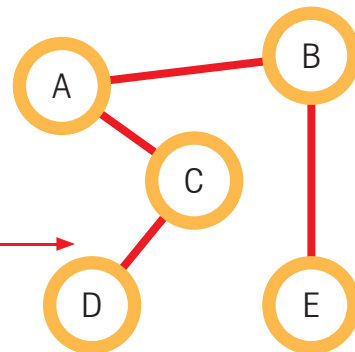


至少要選4條路段才能連接所有景點哦！

表13-3 路段與成本的可能組合

路線組合	路段編號								成本	連接所有景點
	一	二	三	四	五	六	七	八		
1	✓	—	—	—	—	—	—	—	1	×
2	—	✓	—	—	—	—	—	—	7	×
⋮										
n-1	✓	—	✓	—	—	✓	✓	—	12	○
n	✓	—	✓	—	✓	✓	—	—	11	○
⋮										
256	✓	✓	✓	✓	✓	✓	✓	✓	37	○

且成本最低，
連接所有景點，



暴力法的策略簡單易懂，但是當資料量龐大時，這種解題方法就會相當沒有效率。因此，除非沒有更理想的解題策略，我們才會使用暴力法來解題。

貪進法

貪進法（greedy method）的概念就是當面臨不同的選擇時，每次都以最小的成本或最大的效益為原則，來找出可行的解答。

延續上一個找出最低成本的問題（圖13-14），貪進法的策略是先選成本最低的路段，再選成本第二低的路段。本題共有8條路段，若使用貪進法來求解，最多只需經過8次的選取（表13-4），即可找到最佳路徑；和暴力法相比，使用此種方法來求解，效率高出許多。

在選取路段時，我們要注意不可選只會增加成本，但不會增加景點連結的路段。例如選取路段一、三之後，景點A、B、C已可連結在一起，若再選路段四，只會徒增成本（圖13-15）。

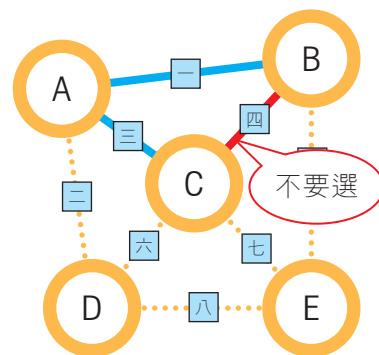
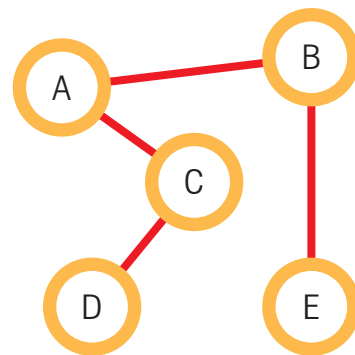


圖13-15 選取路段的原則

表13-4 用貪進法找出最低成本的路段組合

選取路段	成本	徒增成本	路段組合
一	1	否→選取	{一}
三	2	否→選取	{一，三}
五	3	否→選取	{一，三，五}
七	4	是→不選	{一，三，五}
六	5	否→選取	{一，三，五，六}
四	6	是→不選	{一，三，五，六}
二	7	是→不選	{一，三，五，六}
八	9	是→不選	{一，三，五，六}

到此已連接所有景點，
可不必再往下判斷



貪進法的概念簡單、解題效率高，常被用來解決最佳化的問題，例如求最低成本、最短路徑……等。

各個擊破法

各個擊破法（Divide and Conquer）是將一個問題分解（divide）成數個子問題，並找出共通的解決方法（= 擊破，conquer），再透過重複操作的方式來解決各個子問題，最後即可將問題解決。圖13-16是應用各個擊破法，將一串數值由小到大排序的範例。

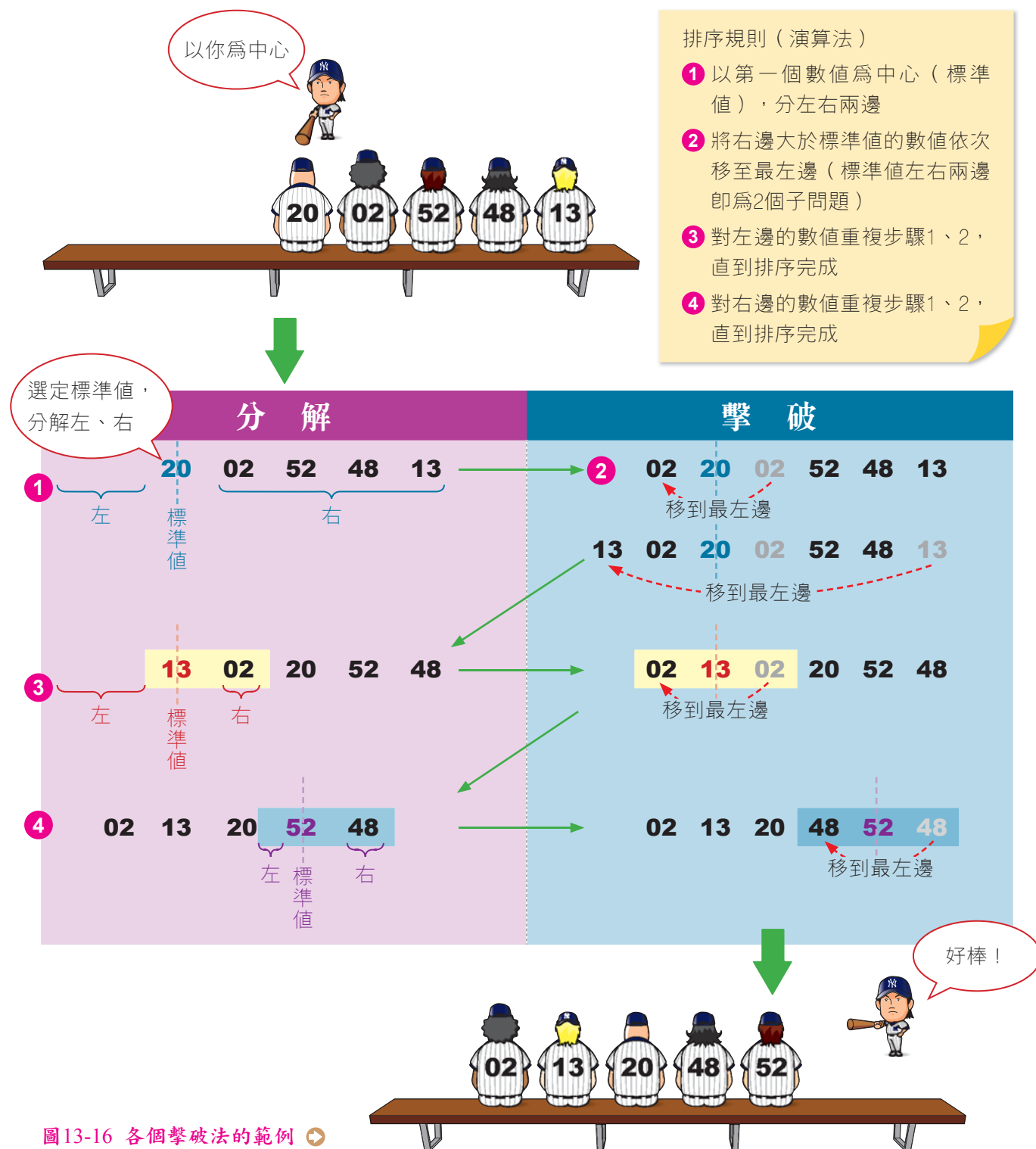
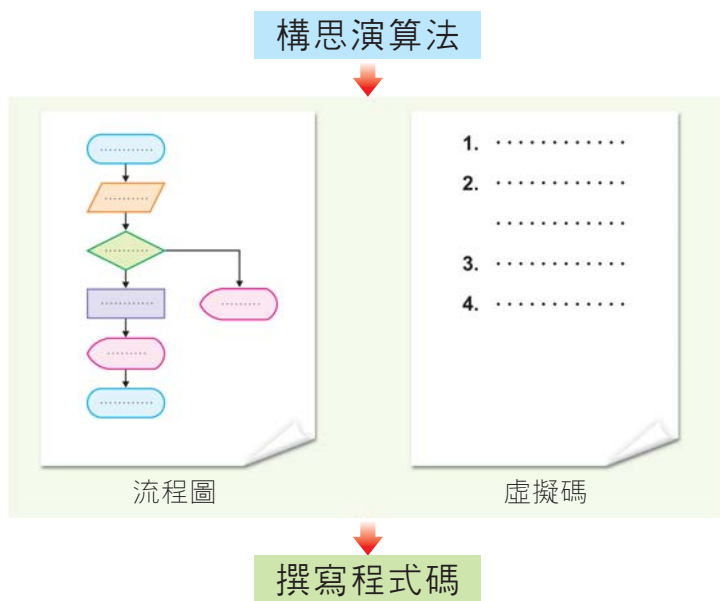


圖13-16 各個擊破法的範例 ➡

13-3.6 演算法與程式設計的關係

一個優良的演算法可幫助我們在有限的步驟內，以最有效率的方法來解決問題。因此在設計程式來解決某項問題之前，我們通常會先使用流程圖或虛擬碼來構思解決問題的演算法，再參照演算法的內容來撰寫程式，如圖13-17所示。



↑ 圖13-17 先構思演算法，再撰寫程式



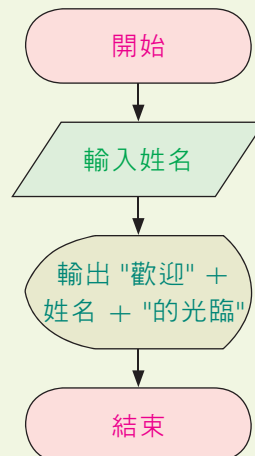
節練習



- ___ 1. 下列有關演算法的敘述，何者不正確？ (A)演算法是用來描述解決問題的步驟 (B)演算法可以利用流程圖或文字敘述的方式來表達 (C)只能應用在科學領域上 (D)演算法的推演步驟可以利用接近程式語言的語法來加以描述。
- ___ 2. 右圖的流程圖可用來表示哪一種程式基本結構的執行流程？ (A)循序結構 (B)重複結構 (C)條件結構 (D)平行結構。
3. 下面3個日常生活問題的解決，適合使用哪一種控制結構來描述？

A.循序結構
B.條件結構
C.重複結構

 - ___ (1) 外出帶傘：如果天氣預報降雨機率超過50%，就帶傘出門；否則就不帶傘出門。
 - ___ (2) 隱形眼鏡配戴：先清潔雙手，再將隱形眼鏡戴上。
 - ___ (3) 繞操場跑5,000公尺：繞長度200公尺的操場連續跑25圈。





13-4 認識資料結構

一個好的程式除了必須使用良好的演算法之外，也需使用適當的資料結構來組織資料，才能節省資料的儲存空間，並提昇資料處理的速度。

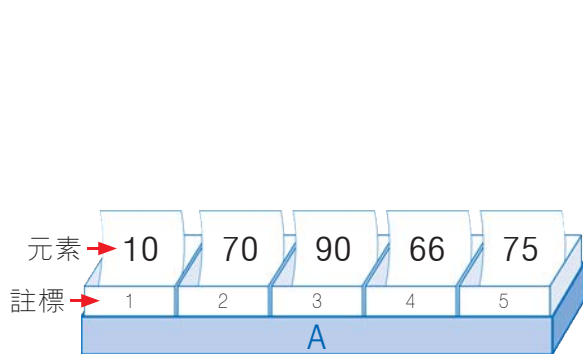
13-4.1 資料結構簡介

資料結構（data structure）是用來組織及管理資料的結構設計。使用資料結構可將資料建立成爲一個便於取用與處理的結構。以下將介紹陣列、堆疊及佇列等3種基本的資料結構。

13-4.2 陣列

陣列（array）是由一群資料型別相同且依序排列的陣列元素所組成。當我們在存取陣列中的資料項目時，必須使用陣列的註標（index）來標示所要存取的資料項目。

陣列若依其註標之個數，可分爲一維陣列（圖13-18）、二維陣列（圖13-19）、……、 n 維陣列；其中 n 表示註標之個數。



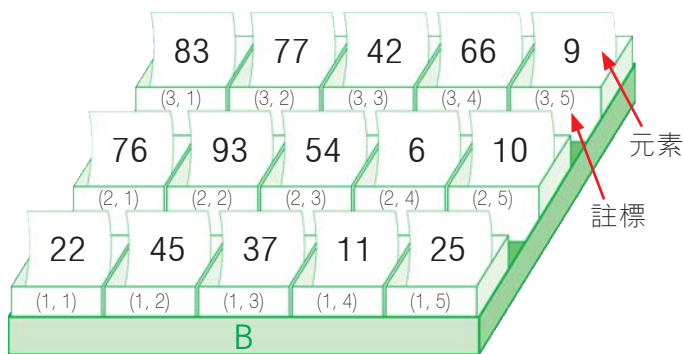
元素A(1) 存放的資料：10

元素A(2) 存放的資料：70

⋮

元素A(5) 存放的資料：75

📌 圖13-18 一維陣列範例



元素B(1,1) 存放的資料：22

元素B(1,2) 存放的資料：45

⋮

元素B(3,5) 存放的資料：9

📌 圖13-19 二維陣列範例

陣列的應用相當廣泛，例如程式中需要處理40位同學的數學成績，便可使用陣列來儲存及處理這些資料，以省去建立多個變數的麻煩。

13-4.3 堆疊

堆疊（stack）是一種具有**後進先出**（Last In First Out, LIFO）特性的資料結構。資料存取規則是：資料項目加入時，只能加到堆疊的頂端（top）；取出資料項目時，必須由頂端將資料優先取出（圖13-20）。在堆疊中加入資料的動作稱為**推入**（push），取出資料的動作則稱為**彈出**（pop）。

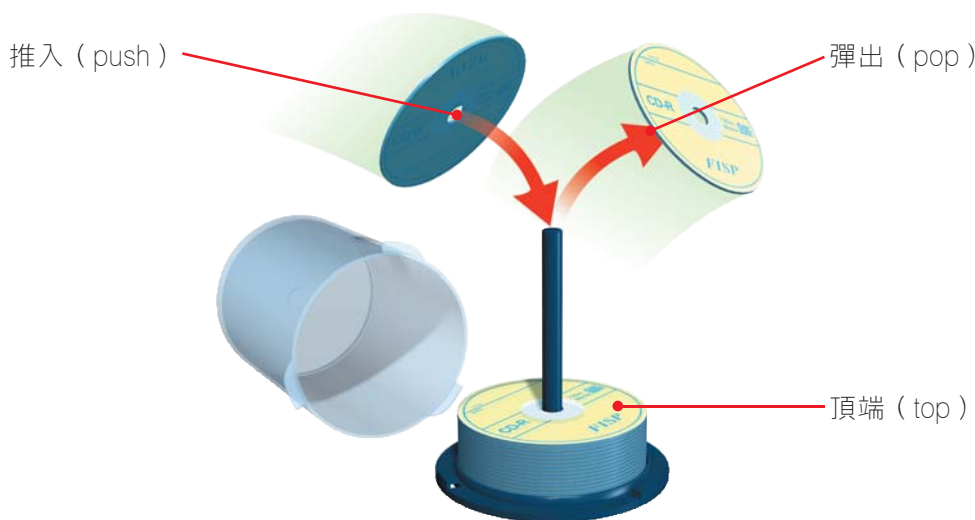


圖13-20 堆疊範例

堆疊的應用相當廣泛，例如瀏覽器的上一頁功能，可讓使用者按此鈕返回曾經拜訪過的網頁，這是因為瀏覽器軟體中有一堆疊，會依序存放（推入）使用者拜訪過的網址，每按一次**上一頁**鈕，就會從堆疊中取出（彈出）最上方的網址，回到使用者前一個拜訪過的網頁（圖13-21）。



圖13-21 堆疊應用的示意圖



13-4.4 佇列

佇列（queue）是一種具有**先進先出**（First In First Out, FIFO）特性的資料結構。資料存取規則是：加入資料項目時，只能從佇列的尾端（rear）加入；取出資料項目時，則只能從佇列的前端（front）取出（圖13-22）。

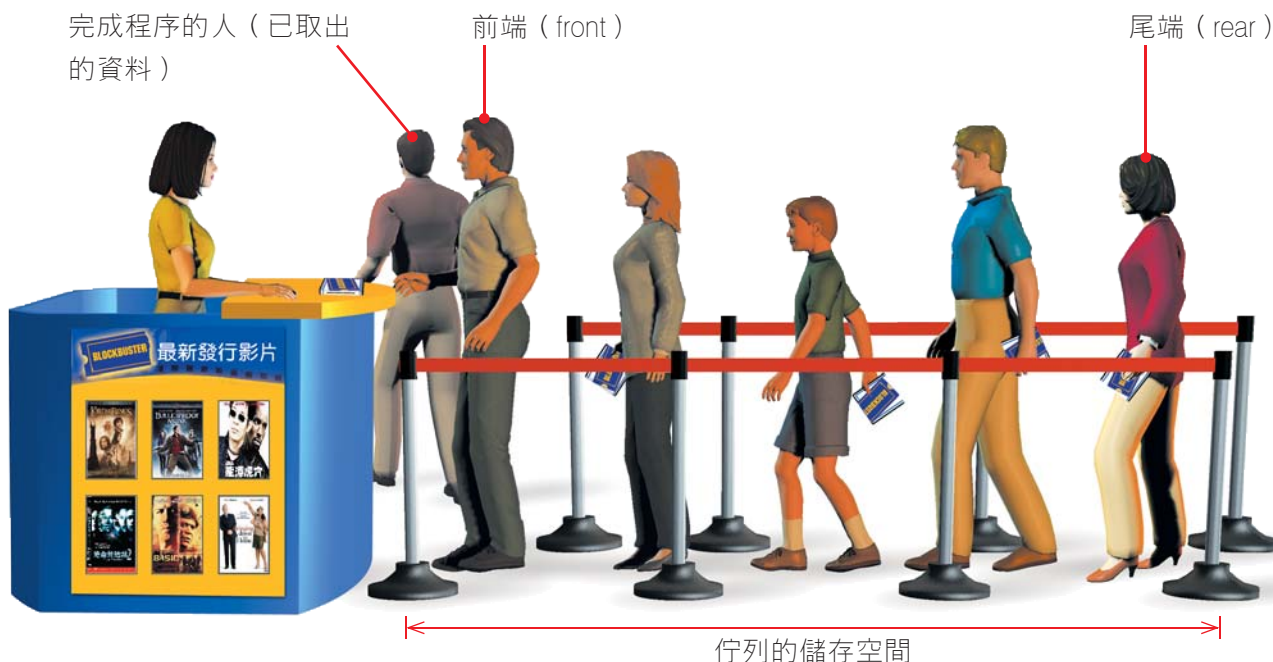


圖13-22 佇列範例

作業系統（如Windows）中的印表機排程式，會依照先後順序來印出文件，就是佇列的一種應用。



練習

1. 在資料結構中，何種存取方式是採用先進先出？ (A)佇列 (B)堆疊 (C)環狀 (D)陣列。
2. 下列哪一項非堆疊的事例？ (A)排隊買票 (B)堆積木 (C)蓋房子 (D)堆盤子。
3. 陣列A的內容如下，請問 $A(2) + A(5) =$ _____。

A					
5	3	2	4	11	15
0	1	2	3	4	5

← 元素





← 註標

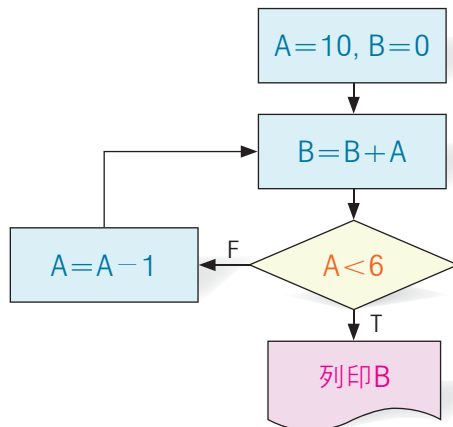


本章習題



選擇題

- ___ 1. 下列何者是使用電腦解題最不需具備的能力？ (A)垂直式邏輯思考能力 (B)循序漸進解題的能力 (C)構思演算法的能力 (D)領導能力。
- ___ 2. 下列何種流程圖的圖形不適用於循序結構之設計？ (A) (B) (C) (D).
- ___ 3. 請問下列以虛擬碼表示的演算法有什麼問題？ (A)只有一個輸出結果 (B)無法在有限步驟內結束 (C)有2個輸入值 (D)步驟不明確。
- (1) 設定數值 $i = 1$, $sum = 0$ (4) 跳到步驟(2)
(2) 將 i 值加到 sum 中 (5) 輸出 sum 值
(3) 將 i 值 + 1
- ___ 4. 一流程圖如圖(一)所示，依流程順利執行後，列印B的值為何？ (A)34 (B)40 (C)45 (D)49。
- ___ 5. 每一次面對問題時，都以最小的成本或最大的效益為原則。上述的演算法，是採用了下列哪一種電腦解題策略？ (A)貪進法 (B)暴力法 (C)黑箱測試法 (D)各個擊破法。
- ___ 6. 老鼠走迷宮的程式必須記錄老鼠走過的路徑，以判別老鼠在迷宮內遇見叉路時，可選擇尚未走過的路徑，請問下列哪一種資料結構最適合用來記錄老鼠走過的路徑？ (A)佇列 (queue) (B)堆疊 (stack) (C)二元樹 (binary tree) (D)陣列 (array)。



圖(一)

多元練習題

1. 以下是一個有趣的邏輯問題，讓同學測試自己的邏輯能力。阿達賣早餐，其中有位客人點了50元的早餐，拿出100元紙鈔結帳；店內沒有零錢，阿達便拿100元與隔壁的老闆兌換，並找錢給客人。過了一會兒，隔壁老闆說剛才的紙鈔是假鈔，阿達為表示歉意，賠了100元給隔壁老闆，請問阿達共損失了多少錢？
- ___
2. 有三對夫婦參加聚會，他們以握手、交談等方式寒暄。假設他們不會跟自己的配偶握手，也不會跟同一個人握兩次手，在用餐前，主辦人老王詢問其他5人握手的次數，結果5個人的答案都不相同。請問老王和他的太太分別與幾個人握過手？
- ___
3. 海上有五座孤島 (A ~ E)，假設要造橋讓這五座島可以相互往來，各座島間可造橋的路段及所需花用的興建成本如右圖所示，請同學找出花費成本最低的路段及其興建成本。
- ___

島與島之間的數字為造橋成本

