

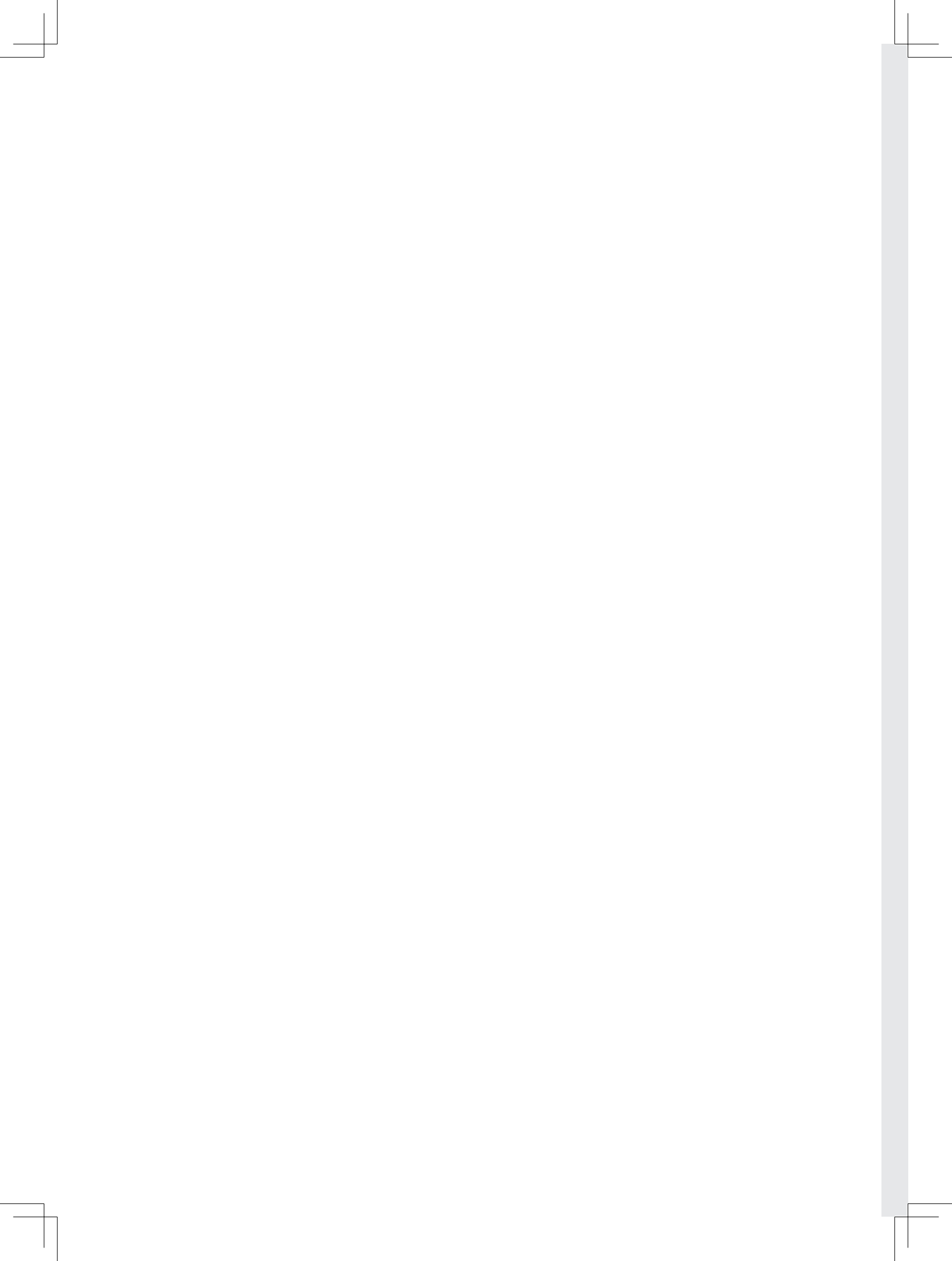
十二年國民基本教育
技術型高級中等學校電機與電子群

程式設計實習

(全一冊)

洪國勝、孫銘宏、蔡懷文 編著
鍾享旭、蔡懷介、陳蘊慈

泉勝出版有限公司
www.goodbooks.com.tw



編輯大意

- 一、本書係依據民國 107 年 8 月教育部發布之技術型高級中等學校電機與電子群「程式設計實習」課程綱要編輯而成。
- 二、本書全一冊，供技術型高級中等學校電機與電子群所屬類科等，第一學年第一學期三學分教學之用。
- 三、本書教材範例豐富，所有指令都有範例解說，讓程式不只是程式，而是可以解決日常生活問題的工具。
- 四、充分的自我練習。我們有充分的自我練習，這樣學生才能充分練習，達到舉一反三的效果。
- 五、跨領域學習。本書的範例涵蓋數學、英文、資訊科技與生活科技，完全呼應與配合 108 課綱的跨領域教學與學習。
- 六、電機與電子群的「程式語言實習」是銜接二年級的「單晶片微處理機實習」的基礎課程，目前所有的嵌入式單晶片微處理機系統的程式語言都是 C 語言，所以本書前面十一個章節選擇 C 語言為程式設計工具。其次，物件導向已經是時代趨勢，C++ 是繼承 C 語言的物件導向語言，因此本書第十二章導入物件導向程式設計語言時，選擇 C++ 語法進行編撰。
- 七、本書章節前加註 ※ 者，為進階主題或範例，教師可依實際教學進度酌予刪減。
- 八、本書雖力求嚴謹，但疏漏之處在所難免，尚祈先進惠予指正。

♂ 本書學習表現如下：

- 一、認識程式語言的架構，具備符號辨識的能力。
- 二、了解以演算法為基礎的程式設計方法，並能以系統思考、規劃執行、科技資訊運用方式，進行專業問題之解決。
- 三、具備程式設計之技術與能力，並了解以專案開發為目標的程式設計概念，並能以團隊合作之精神，積極面對與解決職場各種問題。
- 四、認識程式設計工場設施，並了解工業安全及衛生與消防安全相關知識，建立工作職業安全及衛生知識的理解與實踐，探究職業倫理的基礎素養，並展現良好的工作態度與情操。
- 五、能思辨勞動法令規章與相關議題，省思自我的社會責任。

♂ 教學注意事項：

- 一、本科目為技能領域實習科目，得依據相關規定實施分組教學。
- 二、在課程授課中，要使學生能了解使用軟體的智慧財產權問題，培養公民意識與社會責任。

目錄

§ 第1章 工場安全衛生及程式應用

1-1 實習工場設施環境及機具設備的認識	1-2
1-2 工業安全及衛生、消防安全的認識.....	1-4
1-3 程式應用的認識	1-12
1.4 應用實例的解析	1-13

§ 第2章 程式架構的認識與實作

2-1 程式語言與演進	2-2
2-2 專案架構、程式架構及語言架構	2-4
2-3 程式架構.....	2-8
2-4 程式設計步驟	2-11
2-6 開發環境介面	2-12

§ 第3章 資料的種類與資料型態

3-1 資料種類.....	3-2
3-2 資料型態 (Data Type)	3-4

§ 第4章 變數與常數

4-1 識別字與保留字	4-2
4-2 變數與常數宣告與應用	4-4
4-3 資料型態轉換	4-8
4-4 資料型態應用實例.....	4-9

§ 第5章 基本輸出/輸入函式

5-1	printf()	5-2
5-2	scanf()	5-8
5-3	亂數	5-12
5-4	聲音的產生	5-15

§ 第6章 運算子及運算式

6-1	運算子	6-2
6-2	運算式	6-22
6-3	運算子與運算式應用實例	6-24
6-4	專案除錯	6-33

§ 第7章 流程指令

7-1	if...else	7-2
7-2	switch...case	7-11
7-3	流程指令應用實例	7-19

§ 第8章 迴圈指令

8-1	for 迴圈	8-2
8-2	while 迴圈	8-10
8-3	巢狀迴圈	8-15
8-4	迴圈應用實例	8-21

§ 第9章 陣列

9-1	一維陣列	9-2
9-2	多維陣列	9-11
9-3	字元陣列	9-19
9-4	結構	9-25
9-5	陣列應用實例	9-33

ㄟ 第10章 指標

10-1 指標與單一變數	10-2
10-2 指標與陣列	10-4
10-3 指標與字串	10-13

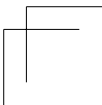
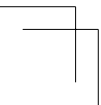
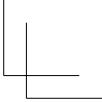
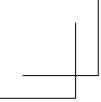
ㄟ 第11章 公用函式及自訂函式

11-1 數值函式	11-3
11-2 字元函式	11-12
11-3 字串函式	11-18
11-4 日期與時間函式	11-27
11-5 自訂函式	11-31
11-6 函式應用實例	11-44

ㄟ 第12章 物件導向程式設計

12-1 C++ 程式設計	12-4
12-2 類別	12-13
※12-3 繼承	12-26
※12-4 應用實例	12-35

ㄟ 附錄1：關鍵字索引



工場安全衛生及程式應用

學習綱要

- ☞ 1-1 實習工場設施環境及機具設備的認識
- ☞ 1-2 工業安全及衛生、消防安全的認識
- ☞ 1-3 程式應用的認識
- ☞ 1-4 應用實例的解析

學習目標

- ☞ 1. 瞭解實習工場的各種設施並且安全的使用及發揮各種設施的最大效能。
- ☞ 2. 瞭解工業安全及衛生的相關法規及注意事項以確保生命及財產安全。
- ☞ 3. 瞭解消防安全的重要性並培養意外發生時的緊急應變能力。
- ☞ 4. 培養學生良好的工作習慣並瞭解程式應用如何帶給人們便利的日常生活。
- ☞ 5. 瞭解程式的應用。
- ☞ 6. 認識程式的應用實例。

1-1 實習工場設施環境及機具設備的認識

工場安全教育是要讓從事工業相關之從業人員，透過教育訓練而具有工業安全之知識與技能，讓所有從業人員在安全的工作環境中工作，加上適當的安全防護措施，用科學與安全的方法從事工作，以防止工場意外事故之發生，減少損失。另外，實習課程的教學必須分門別類在各種不同的專門實習工場實施，在一般職業類科學校的實習工場中，主要的設施有：實習儀器設備、場地照明及通風設備、人員機具安全防護設備、及各類標示牌等等，如圖 1-1。

 單晶片微處理機實習工場環境及機具說明。

各校實習工場設備如圖 1-2，因地制宜各有特色，通常單晶片微處理機實習工場主要建置設備有下列各項：

► 實習儀器設備

1. 個人電腦	2. 教學廣播系統
3. 作業系統	4. 應用軟體
5. 投影機	6. 印表機
7. 有線網路 / 無線網路	8. 交換機
9. 伺服器	10. 不斷電系統 (UPS)

► 場地照明及通風設備

1. 日光燈	2. 緊急照明燈
3. 通風扇	4. 空調

► 人員機具安全防護設備

1. 滅火器	2. 漏電、過載斷路保護器
3. 煙霧偵測器	4. 急救箱
5. 緊急求救鈴	6. 緊急廣播系統
7. 消防栓	8. 緩降機

圖 1-1 實習工場各類標示牌



圖 1-2 單晶片微處理機實習工場

1-2 工業安全及衛生、消防安全的認識

工業安全（職業安全）與工業衛生（職業衛生）以及消防安全的基本目標是一樣的，皆是致力於維護工作人員的安全與健康，避免意外事故的發生，以及災害發生時的緊急應變措施，所以工（職）業安全與工（職）業衛生以及消防安全是先進國家非常重視的課題。工業安全是指「透過各種安全防護措施，以避免工業災害的發生。」工（職）業安全課程包括：意外事故與職業傷害、機具安全與維護、電器安全與維護等。工（職）業衛生是指「分析工（職）業環境對工作人員健康影響的一切因素，進而利用科學方法去預防和減少工作人員產生疾病和傷害。」工（職）業衛生課程包括：危害物的管理與回收處理、急救、工作環境清潔等。我國憲法第 153 條規定：「國家為改良勞工及農民之生活，增進其生產技能，應制定保護勞工及農民之法律，實施保護勞工及農民政策。」並且明文規定「勞動基準法」為國家實現此一基本國策所制定之法律。

 工業安全與衛生。

► 工業安全及衛生之標示顏色相關規定

1. 紅色表示：危險或禁止標示。（例如：緊急控停止鈕、禁煙標示等。）
2. 橙色表示：可能發生立即災害或傷害的地方所用之標示。（例如：爆裂物、電擊傷害等。）
3. 黃色表示：注意或警告標示。（例如：易燃物、墜落、跌倒等。）
4. 綠色表示：無危險或衛生有關標示。（例如：逃生門、急救箱。）
5. 藍色表示：提醒注意標示。（例如：控制按鈕、指示燈。）
6. 白色表示：一般標示。（例如：方向指示。）
7. 紫色表示：放射性危險警告或污染標示。（例如：放射性物質現場警告標示。）
8. 黑色表示：輔助或提醒用標示。（例如：文字標語。）

► 工業安全及衛生之意外發生與急救處理

一般事故的發生，大多數是人為因素所造成的，只要透過教育訓練，確定每個工作人員具備安全觀念和知識，並且消除不安全之機具設備的因素與環境因素，便可以使意外事故發生減少到最低。為了防止事故的發生，必先探究事故發生的原因，一般意外事故發生的主要原因有三：

1. **人員的因素**：意外事故發生中大部分是肇因於工作人員的粗心大意、不熟悉作業流程或是精神狀態不佳所造成的。
2. **機具設備的因素**：機具設備若缺乏安全防護，或是未落實定期保養也是意外事故發生的主要因素之一。
3. **環境因素**：工場照明不良、通風不良、噪音太大、粉塵、有毒氣體、不適當的建築結構與隔間或是不整潔的廠房也是意外事故發生的主要因素之一。

另一方面，對於意外事故的發生我們除了須盡力防範之外，也必須了解事故的發生仍然無法絕對避免，因此，急救處理便是給予意外事故發生時受傷者的緊急照料，以維持傷患之生命現象，除去傷害因素，阻止傷勢惡化並給予傷患心理上之鼓舞及信心，直到受傷者獲得醫護人員的救治為止。簡單來說，急救處理的目的就是救命，當意外事故發生時，請依圖 1-3 學校場所意外通報流程快速通報學校，若有人員受傷，通常會依傷患受傷的類別來進行急救。各受傷類別之急救方法分述如下：

1. 外傷急救

- (1) 使患者平躺，並將出血部位設法抬高。
- (2) 以清潔的紗布止血，綁緊繃帶，控制出血，但不可過緊以免影響血液循環，使肌肉組織壞死。
- (3) 盡快送外科醫院診治。

2. 骨折急救

- (1) 閉鎖性骨折：斷骨端未刺穿皮膚。
 - a. 使患者平躺，盡可能將傷肢置於自然的位置。
 - b. 使用木板固定患部，並用繃帶或布條固定之。

- (2) 開放性骨折：斷骨端刺穿皮膚。
 - a. 用壓力繃帶止血。
 - b. 使患者平躺，盡可能將傷肢置於自然的位置。
 - c. 使用木板固定患部，並用繃帶或布條固定之。

3. 灼傷急救

人體的皮膚因受到熱、冷、火、化學藥品等傷害，稱之為灼傷，灼傷的急救方法應以「沖」、「脫」、「泡」、「蓋」、「送」五大步驟 施行。

- (1) 沖：以流動的冷水沖洗灼傷部位 15 ～ 30 分鐘。
- (2) 脫：於冷水中小心脫除灼傷部位的衣物。
- (3) 泡：於冷水中浸泡 15 ～ 30 分鐘，以降低傷口疼痛。
- (4) 蓋：以乾淨的毛巾蓋住傷口，防止細菌感染。
- (5) 送：快速送醫治療。

4. 大量出血急救（需注意血液循環的問題）

- (1) 直接壓迫傷口止血法。
- (2) 壓迫止血點止血法。
- (3) 止血帶止血法

5. 昏迷無心跳無呼吸急救

當傷患傷患呼吸停止無心跳時，如沒有即時急救必定死亡，因此必須立即施行心肺復甦術 (Cardio Pulmonary Resuscitation, CPR)(注意：沒有經過完整訓練並取得合格證書者，不可隨意對傷患進行 CPR 急救)，依據 2011 年我國衛生署函文說明的新版 CPR 急救法，分為六個程序，簡稱「叫叫 CABD」，如圖 1-4 所示：

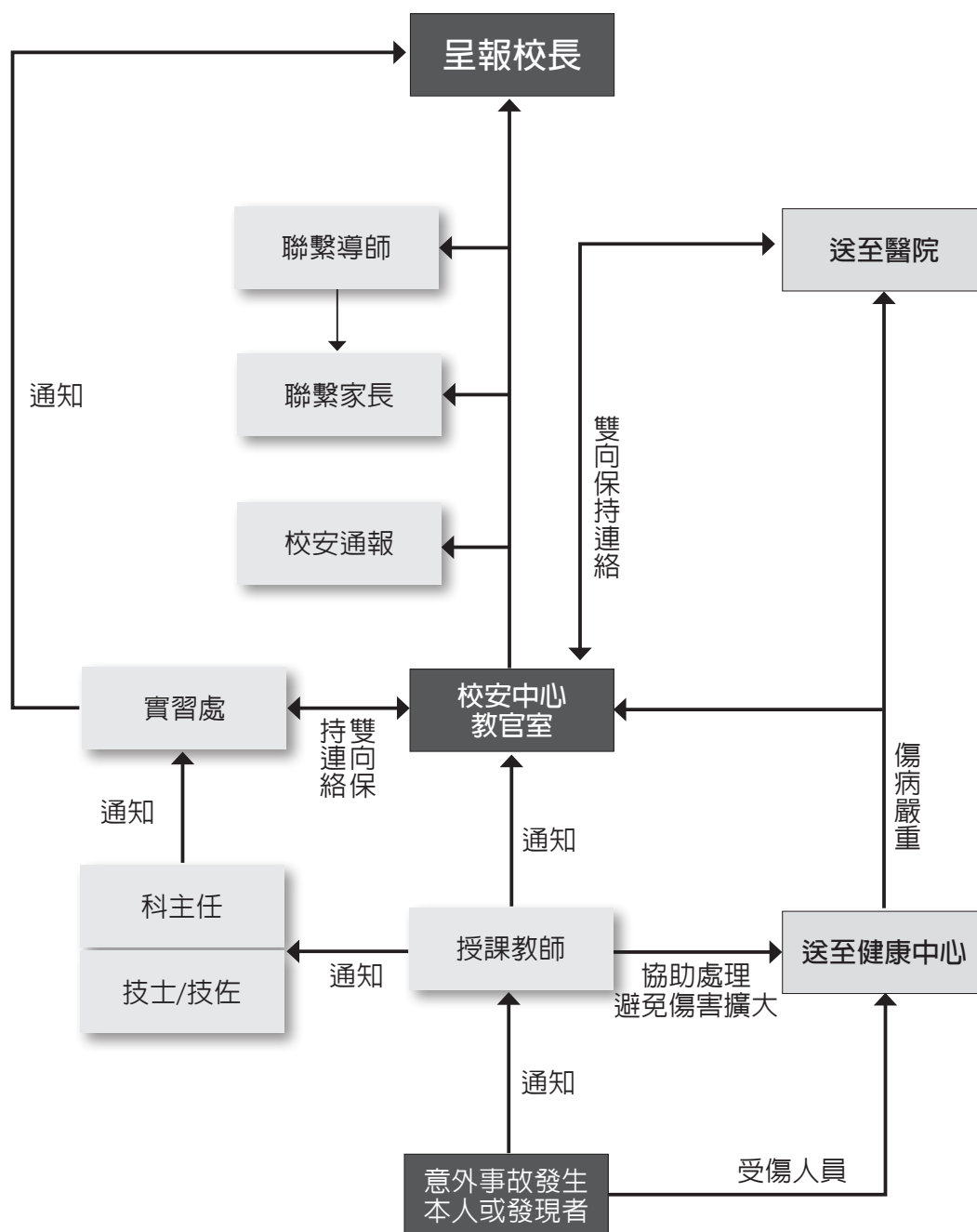


圖 1-3 學校實習場所意外通報流程圖

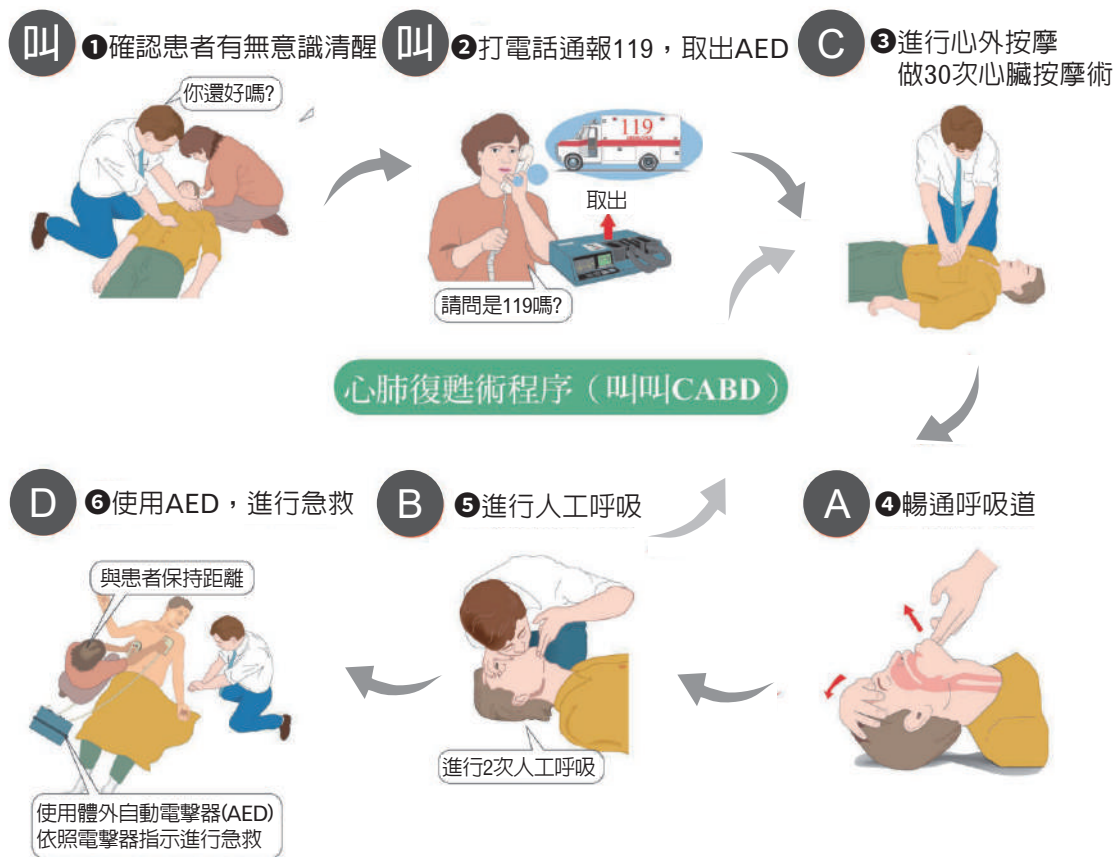


圖 1-4 心肺復甦術程序 (摘自苗栗縣政府消防局—救護 e 點靈叫叫 CABD 之急救程序)

♂ 消防安全。

火災為工業安全之意外災害中最嚴重的災害之一，它所造成的災害損失都很大，尤其是人員的傷亡。因此防火、滅火及火災的應變，為所有人員都必須學習的科目。

► 火災的種類與滅火方法

火災的種類依據火災燃燒物之性質可將火災分為下列四類，如表 1-1。

► 表 1-1 火災分類表

火災類別	火災名稱	燃燒物	適當滅火方法
A 類 (甲類)	一般火災	由一般可燃性固體所引起的火災。如木材、紙張、衣服、塑膠、橡膠等。	使用水或含有大量水分的溶劑滅火最有效。
B 類 (乙類)	油類火災	由可燃性液體或可燃性氣體與油脂類物質所引起的火災，如汽油、機油、燃料油、溶劑、酒精等。	此類火災禁止使用大量的水來灌注滅火，因為油浮於水面上，水的流動會助長火焰的蔓延。正確的滅火方式常用乾化學劑或二氧化碳、泡沫等方式來滅火。
C 類 (丙類)	電器火災	由電力設施、電氣設備所引起的火災，如馬達、開關設備、接線盒、變壓器等。	此類火災滅火必須使用非導電性滅火劑如乾化學劑、二氧化碳等。泡沫及水均不宜使用，因水具有導電性，可能使救火人員觸電，但是如果確定總電源切斷後，可視為甲 (A)、乙 (B) 類火災。
D 類 (丁類)	金屬火災	由易氧化之金屬所引起的火災，如鉀、鈉、鈦、鋁、鎂等。	此類火災滅火需要特種技術並使用特殊金屬化學乾粉的滅火器才能撲滅。
輔助記憶法： 憶猶未盡 (一油電金) (一般火災、油類火災、電器火災燒焦味、金屬火災)			

各類火災因燃燒物不同，可選用的滅火器對照表，如表 1-2。

► 表 1-2 各類火災滅火器對照表

適用滅火劑 火災類別					乾粉		
	水	泡沫	二氧化碳	鹵化烷 (乾化學劑)	ABC 類乾粉	BC 類乾粉	D類乾粉(特殊 金屬化學乾粉)
甲 (A) 類火災	○	○			○		
乙 (B) 類火災		○	○	○	○	○	
丙 (C) 類火災			○	○	○	○	
丁 (D) 類火災							○

► 滅火器的裝設規定

根據我國「各類場所消防安全設備設置標準」第 31 條，滅火器應依下列規定設置：

- (1) 設有滅火器之樓層，自樓面居室任一點至滅火器之步行距離在二十公尺以下。
- (2) 固定放置於取用方便之明顯處所，並設有以紅底白字標明滅火器字樣之標識，其每字應在二十平方公分以上。但與室內消防栓箱等設備併設於箱體內並於箱面標明滅火器字樣者，其標識顏色不在此限。
- (3) 懸掛於牆上或放置滅火器箱中之滅火器，其上端與樓地板面之距離，十八公斤以上者在一公尺以下，未滿十八公斤者在一點五公尺以下。

► 火災的預防

1. 去除可燃物質。
2. 防止高溫及無法散熱的問題。
3. 嚴禁工作人員吸煙。
4. 養成良好工作方法及習慣，慎選慎用高溫設備。
5. 做好電力線路及設備之檢查及維護。
6. 做好防災教育訓練。

► 火災的應變

火災應變能力絕不是與生俱來的，它是經過教育訓練與不斷的演習才能夠具備的一種能力。當火災發生時才能藉由正確的使用滅火裝置於第一時間滅火，並能適時的疏散逃生。另外，工場應設置消防組織，對所有從業人員實施消防訓練，其訓練內容應包括：

1. 消防常識。
2. 各種消防裝置之使用。
3. 逃生與疏散。
4. 依消防任務編組實施演習。
5. 檢討消防裝置是否足夠與汰舊換新。

► 滅火器與消防栓的使用

1. 滅火器的使用：滅火器的使用步驟如圖 1-5 所示：

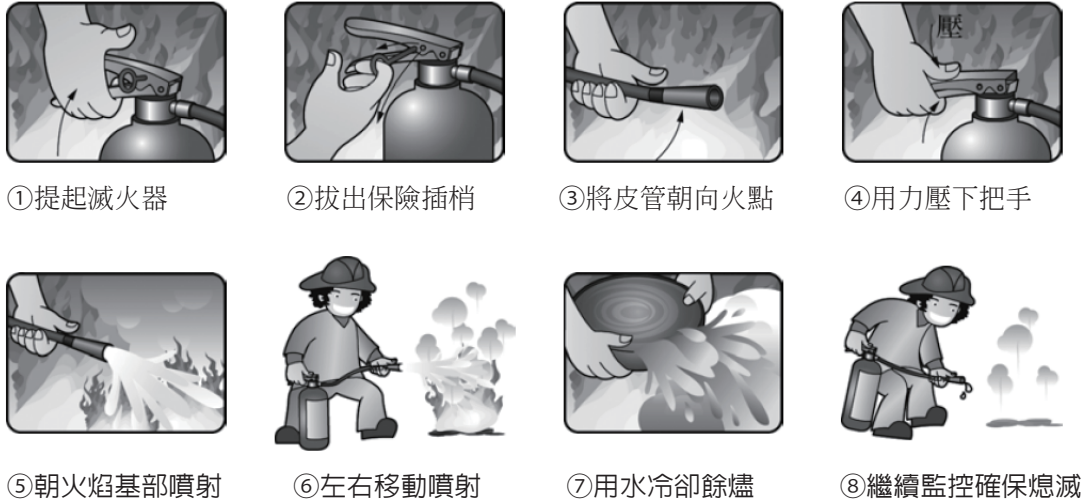


圖 1-5 滅火器使用步驟圖

2. 消防栓的使用：消防栓的使用步驟如圖 1-6 所示：

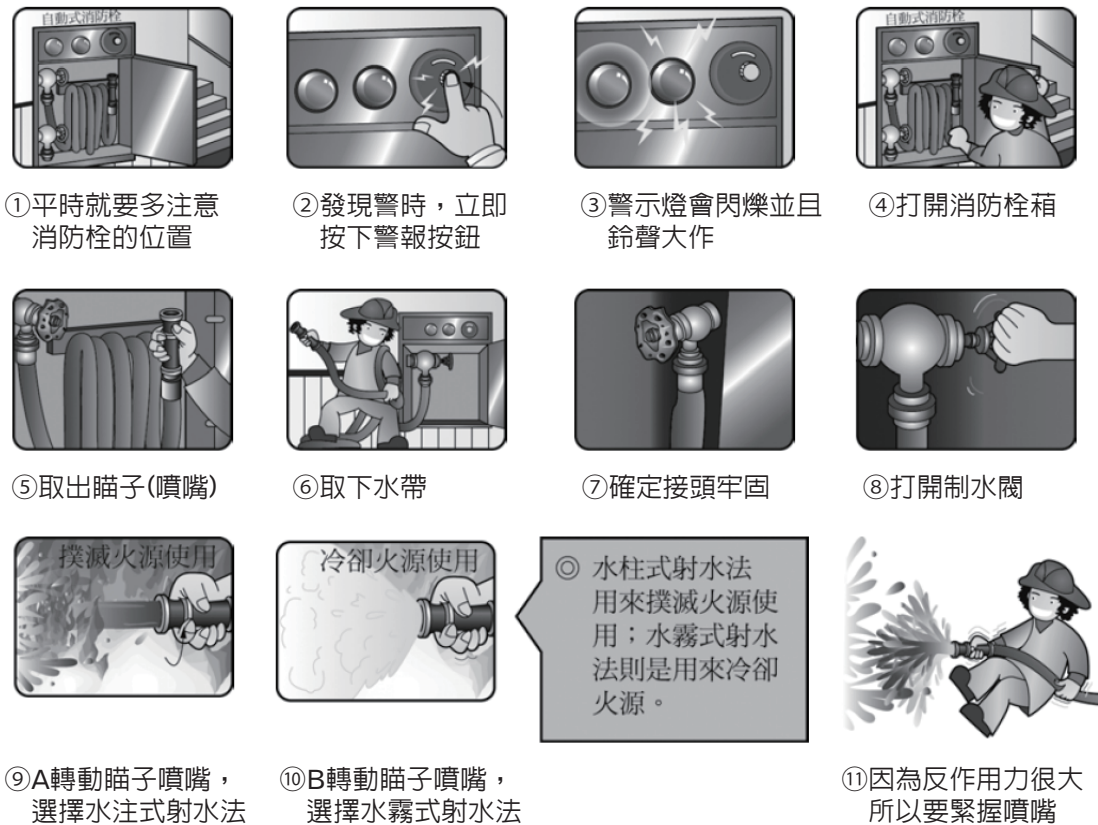


圖 1-6 消防栓使用步驟圖

1-3 程式應用的認識

隨著資訊科技的進步與普及，我們每天生活幾乎都離不開程式應用的服務之中，例如：

1. 手機與電腦的作業系統與應用程式。隨著資訊科技的進步，現在已經是人人有手機，家家有電腦或筆電，手機與電腦的作業系統與應用程式都是程式的應用。
2. 通訊軟體。隨著網路的普及，在家用手机與電腦，搭配通訊軟體 Facebook、Line 等等，就能得到全世界所有的資料、影片，這些網路傳輸軟體也是程式應用。
3. 工業自動化（Industrial Automation）。現代的工業生產線，已經大部分由機器人取代人力，機器人可自動焊接、組裝、運送，機器人與機器人也自動溝通，整個生產線的人力僅維護這些機器人的運轉，這些機器人的運轉與溝通也都是程式應用。
4. 商業自動化。現在所有的商業進銷存運轉、銀行金流、網路購物等都是靠程式應用。
5. 醫院自動化。現代醫院也不用人工傳送病歷了，所有的檢驗報告、醫囑、處方、收費等都是程式應用。
6. 個人穿戴式裝置。隨著智慧型裝置的快速發展，手機將不是被動裝置，日後手機將會協助監控我們的身體健康，發出訊息提醒自己、或直接給訊息傳給醫療單位，作為協助就醫的依據等，這也是程式應用。
7. 物聯網（Internet of Things，簡稱 IoT）時代來臨。現在每個人都有手機，人與人都可溝通，下一步將是物與物直接溝通，冰箱東西吃完了，冰箱會直接叫貨，機器人可自駕，也可直接操作電梯將東西送到我們家；地上髒了、機器人也可以自動打掃等等等，這些也都是程式應用。

1.4 應用實例的解析

♂ 模擬程式

由於程式設計功能的突飛猛進，現今訓練學習開飛機、開高鐵都有模擬程式，這樣可以在沒有危險的環境學習，且節省學習成本與時間。同樣的道理，電機電子群入門都要學習三用電表與示波器，筆者乃用所學，撰寫模擬程式。

1. **三用電表模擬程式**：三用電表的檔位很多，老師操作時，每一個學生很難看清楚。而且學生操作時，只要操作錯誤，此電表就燒掉，造成資源的浪費，所以我就撰寫此模擬程式，老師可以透過電腦廣播，每個學生就能身歷其境，當然可以安全的操作且看的更清楚。開啓三用電表模擬程式 (<http://www.goodbooks.com.tw/elect/meter.exe>) 如圖 1-7：操作方式同實體操作，先假設輸入是 8V，輸出也與實體三用電表相同，一樣有很多檔位讓學生選擇，就算選錯檔位也不會燒掉。下圖左是輸出檔位選「10V」時，指針的位置。下圖右是輸出檔位選「25V」時，指針的位置。請同學自行開啓應用程式，就能體會程式設計的用途與樂趣。

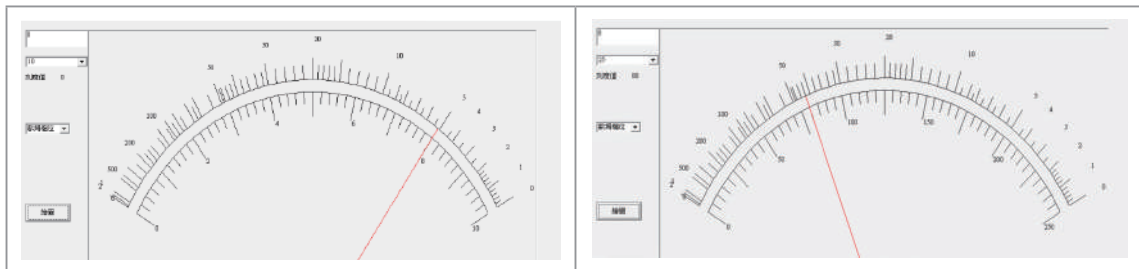


圖 1-7 三用電表模擬程式

2. **示波器模擬程式**：示波器也是一樣，老師實體教學時，後面的同學很難看清楚，自己操作時，檔位非常多，撥錯檔位也是沒有結果，很有挫敗感。開啓示波器模擬程式（<http://www.goodbooks.com.tw/elect/osc.exe>）如下圖，它可以幫助學生認識與學習示波器檔位與模擬執行結果。例如，輸入是 3V,1kHz，我們一樣有 Volts/Div 與 Time/Div 供使用者選擇，下圖左是 Volts/Div 選「1」,Time/Div 選「0.2m」的輸出結果；圖 1-8 是 Volts/Div 選「2」,Time/Div 選「0.5m」的輸出結果

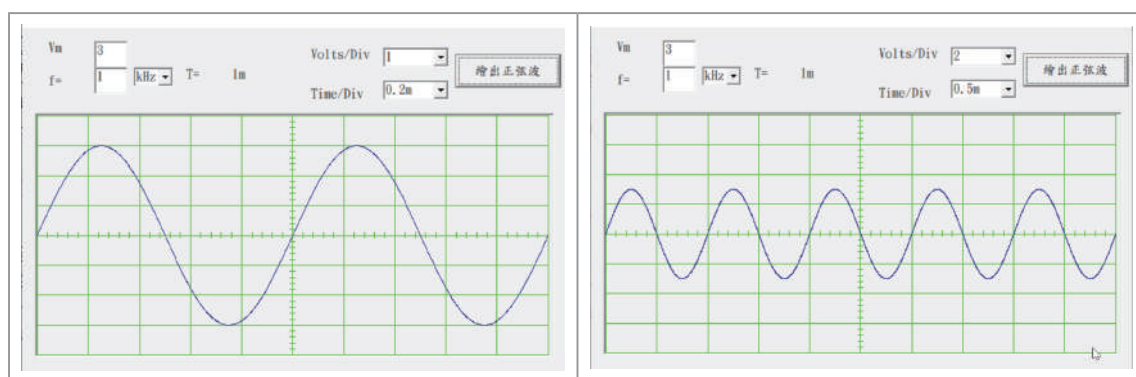


圖 1-8 示波器模擬程式

3. **英文輔助學習系統**：每個人的語言天賦不同，高材生一聽就記起來。但是大部分學生常常有聽有讀，但卻沒有進腦袋，有錢人可以請家教老師、進補習班、買 CD，沒錢沒天賦，學習成果就很糟。但是因為科技的進步，現在人人有電腦有手機，所以我就研發此程式，希望可以讓每個學生都可以很有效率的學習。其次，我的程式還比家教還好用，若一次沒聽懂，還可無限次叫電腦重念，還可叫電腦說慢一點，直到您聽懂為止，且學生能將所聽得的文句打字下來，由電腦評分。使用手機或電腦開啓此系統（<http://www.goodbooks.com.tw/eng/eng.htm>），畫面如圖 1-9：裡面程式碼也都公開，學生不僅可以學英文，也可以學程式。

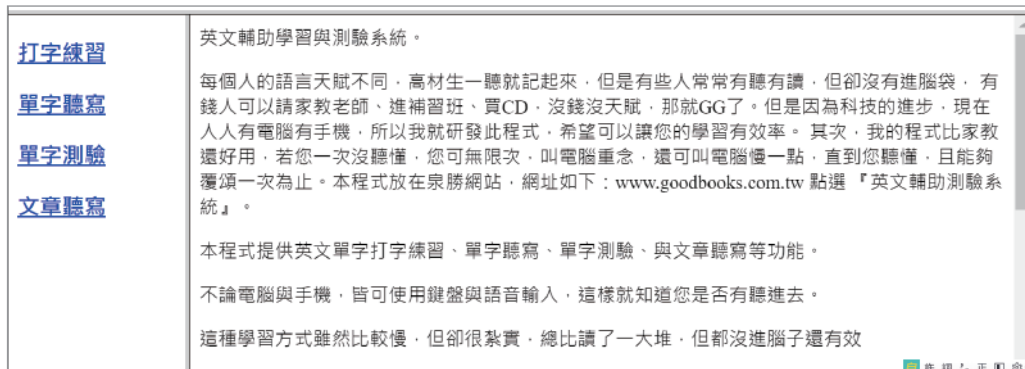


圖 1-9 英文輔助學習系統一

(1) 打字練習。

打字練習如下圖 1-10 左，它會自動出現單字，且將該單字念出來，可以讓初學者練習英文打字。此系統有提供國中、高中單字，使用者也可以自己貼上課本的單字，本應用程式可在手機、平版、電腦使用，電腦也可以評判正確與否。

(2) 單字聽寫。

單字聽寫如圖 1-10 右，與打字練習有點接近，但只有唸出單字，沒有出現單字，可以讓學生練習單字聽寫。



圖 1-10 英文輔助學習系統二

(3) 單字測驗。

單字測驗如圖 1-11 左，是以上單字聽寫的進階，但僅出現中文字，使用者要打入英文。

(4) 文章聽寫。

文章聽寫如圖 1-11 右，電腦會自動朗讀一個完整句子，使用者要輸入整個句子，可以讓學生練習文章聽寫測驗。

圖 1-11 英文輔助學習系統三

4. 數學輔助學習系統：數學輔助學習系統 (<http://www.goodbooks.com.tw/math/math.htm>) 如圖 1-12，可以讓使用者使用程式演算術數學，程式碼也都公開，學生不僅可以學數學，也可以學程式。

圖 1-12 數學輔助學習系統

5. 遊戲程式：遊戲程式如圖 1-13（<http://www.goodbooks.com.tw/pbook/pbook.htm>），程式碼也都公開，讓有心練習程式的可以觀摩遊戲程式的撰寫。

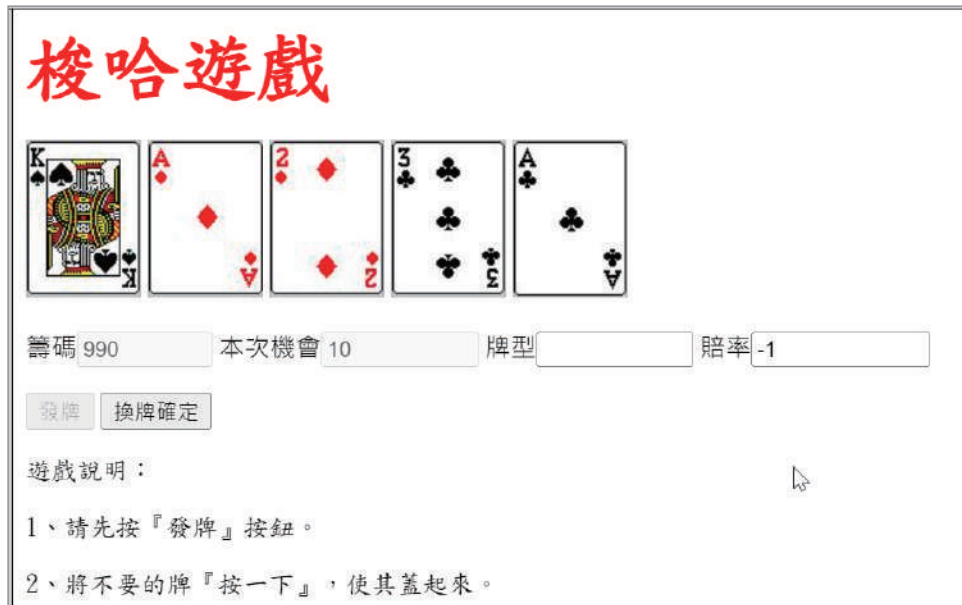


圖 1-13 遊戲程式

學後測驗

- () 1. 工場安全教育其主要且直接的目的是在 (A) 提高生產力 (B) 避免人員與設備受損 (C) 安定勞工生活 (D) 減少投資成本。
- () 2. 為了防止人員觸電所引起的傷害，下列何者錯誤？ (A) 電器設備均應有接地措施 (B) 手足潮濕，不可碰觸或操作電器設備 (C) 可以用手指測試線路或電源是否有電 (D) 危險的電力設施要有安全標誌，並有適當的限制接近設施。C
- () 3. 下列敘述中何者是正確的？ (A) 在工場中實習可以穿著寬鬆的衣服比較舒服 (B) 工場應保持通風良好，光線充足 (C) 在工場中無需保持整齊、清潔 (D) 學生在工場實習中可不必配戴個人防護器具。
- () 4. 下列對工業用標示顏色所代表之意義，何者為錯誤？ (A) 紅色表示防火設備、禁止 (B) 黃色表示注意、警告 (C) 綠色表示安全、救護設備 (D) 藍色表示放射性危險。
- () 5. 下列因素中何者不是由不安全的個人因素所造成的意外事件？ (A) 粗心 (B) 精神狀態不佳 (C) 不使用防護器具 (D) 噪音。
- () 6. 燙傷時的處理程序為 (A) 叫叫 CABD (B) 沖脫泡蓋送 (C) 牙膏與醬油塗抹 (D) 立即刺破水泡
- () 7. 下列哪一類火災是電氣火災？ (A) 甲 (A) 類 (B) 乙 (B) 類 (C) 丙 (C) 類 (D) 丁 (D) 類。
- () 8. 使用滅火器應站在何處？ (A) 上風 (B) 下風 (C) 側風 (D) 逆風。
- () 9. 下列何種滅火器不適用於撲滅丙 (C) 類電氣火災？ (A) ABC 乾粉滅火器 (B) 泡沫滅火器 (C) 二氧化碳滅火器 (D) 鹵化烷滅火器（俗稱海龍，Halon）。
- () 10. 一般 18 公斤以下滅火器，放置的高度不得超過多少公尺？（滅火器上端與樓地板之距離） (A) 1 公尺 (B) 1.5 公尺 (C) 2 公尺 (D) 2.5 公尺。

程式架構的認識與實作

學習綱要

- ☞ 2-1 程式語言與演進
- ☞ 2-2 專案架構、程式架構及語言架構
- ☞ 2-3 演算法的認識
- ☞ 2-4 程式設計步驟
- ☞ 2-5 開發環境介面

學習目標

- ☞ 認識程式語言的演進。
- ☞ 認識 C 語言的架構與專案架構。
- ☞ 認識 C 語言程式架構。
- ☞ 能以演算法寫出程式架構。
- ☞ 瞭解程式設計步驟。
- ☞ 能以 CodeBlocks 編輯、編譯與執行程式。

2-1 程式語言與演進

程式語言

人與人之間溝通的工具稱為語言，世界上因有許多民族，因其發源地不同，所以就有許多語言。例如，華語、英語及德語等。其次，人與電腦溝通的工具，則稱為電腦程式語言。那麼為什麼沒有電視語言、冰箱語言或冷氣語言呢？那是因為這些機器的功能較為簡單，只要幾個按鈕就能發揮其功能。但是電腦的功能就非常多，多到連用整個鍵盤的所有按鍵都無法表現其功能，所以必須使用一些類似單字所組成的片語與敘述來發揮其所有功能，這些單字與片語的集合就稱為電腦程式語言，簡稱程式語言。就如同人類也無法用 26 個字母表達所有感受與思維，必須藉助這些字母的排類組合，先組成單字，再由單字組合成片語與句字，才能充分表達其思維。

程式語言的演進

電腦語言依其演化先後，大致可分為低階語言 (Low-level programming language) 與高階語言 (High-level programming language)，分別說明如下：

► 低階語言

低階語言依其演化先後，又可分為機器語言 (Machine language) 與組合語言 (Assembly language)，分別說明如下：

機器語言

機器語言是由一堆 0 與 1 所組成的單字。例如，要在螢幕輸出 A，所需鍵入單字如下：

```
B2 41 B4 02 CD 21 CD 20
```

由於這些單字的來龍去脈過於複雜，絕非一般人所能體會。只因為要於螢幕顯示一個 A，就要鍵入以上 8 個單字，那如果要計算一個正方形面積，其所需鍵入的單字可能比火車還長，還不如用筆算還快，哪需要計算機呢！其次，電腦剛發明時並沒有鍵盤，要鍵入以上單字，必須用 8 個位元一組的指撥開關，分別輸入 8 次 (共 8 個單字)。例如，B2 是由 (10110010)

所組成，則必須先將 8 個指撥開關分別設定為 (on,off,on,on,off,off,on,off)，由此可見當時使用機器語言的艱辛，真是未蒙其利，先受其害。

組合語言

當時學習與使用機器語言是艱辛的，而且電腦的體積又非常龐大，通常只有作為學術機關研究用。但是當時科學家並未氣餒，為了推廣這一神奇而且好用的工具，所以有組合語言的發展。所謂組合語言就是使用一些助憶碼 (Assembly mnemonic)，這些助憶碼類似一些簡單的英文單字，以便協助使用者記憶與書寫所要完成的程式。例如，上例中要將字元 A 顯示於螢幕，所需組合語言如下：

```
MOV DL,41
MOV AH,2
INT 21
INT 20
```

其中，MOV 是 MOVE 的縮寫，41 是字元 A 的 ASCII 碼，DL 與 AH 則是 CPU 的內部暫存器，INT 21 是呼叫字元顯示副程式，INT 20 則是呼叫程式結束處理副程式。也就是程式發展已經有一脈絡可循，一切遵循一些簡單規則，方便使用者的查詢與記憶。由於有了組合語言的好工具，學習電腦終於可以突破實驗室，推廣到大學以上的資訊電子相關科系。

► 高階語言

有了組合語言，電腦使用的方便性可說向前邁向一大步，但是電腦的使用卻僅限於資訊電子相關科系，因為不同的 CPU 就要有不同的組合語言，也就是說此程式語言沒有硬體相容性。其次，這些助憶碼也要花費很多時間學習，而且每一小小功能的程式長度都要超過一頁，所以科學家並不以此為滿足，所以有高階語言的研發。例如，前例要在螢幕顯示字元 A，若使用 C 語言，則只要撰寫如下：

```
printf("%c",'A');
```

又例如，面積的計算，程式如下：

```
int a=15;
int b=8;
int c;
c=a*b;
printf("面積=%d\n",c);
```

以上程式與人類學習數學的習慣非常接近，已經可以被大部分人接受。

♂ 程式語言的選擇

電機電子群的專長是電機電子設備的控制，學習程式語言的目的主要是銜接二、三年級的嵌入式晶片控制。目前所有的嵌入式系統的程式語言都是使用 C 語言，所以本書選擇 C 語言作為電機電子群程式設計實習的語言。其次，物件導向已經是程式語言領域的標準配備，C++ 是繼承 C 語言的物件導向語言，所以本書第十二章導入物件導向程式設計語言時，選擇 C++ 闡述物件導向的程式設計。

2-2 專案架構、程式架構及語言架構

♂ 專案架構

一個可執行的 C 語言稱為一個專案，每個專案可由至少一個 main() 函式、若干函式、組合語言函式、C 標準函式庫、自訂函式庫所組成，且由 main() 函式當作起動函式，如 2-1 圖所示。

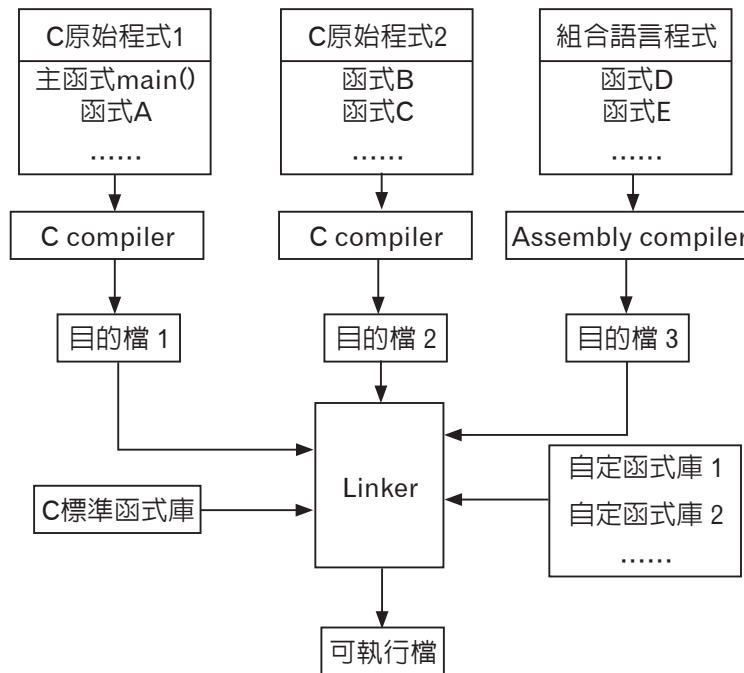


圖 2-1 C 程式專案架構

依據模組化原則，大的專案都要先分解為很多單一函式。所以實用的 C 程式都是由很多單一功能函式組合而成，因為單一功能函式容易撰寫、除錯與編譯，這些單一功能都先使用編譯程式 (Compiler) 編譯成目的檔「*.obj」，如上圖的目的檔 1、或目的檔 2。每當要改變程式功能時，只要修改對應的函式，編譯此函式，不用全部重新編譯，當然效率也高。最後階段再使用連結程式 (Linker)，連結所有目的檔「*.obj」、對應的標頭檔 (C 標準函式庫共有 29 個標頭檔) 或已經編譯完成的自訂函式庫，即可完成可執行檔。

🔗 程式架構

以上專案架構有點複雜，早期學習 C 語言程式設計，都要自己使用 Compiler 編譯程式，將程式編譯成目的程式 (「*.obj」)。若需修改，則修改後再編譯。最後再使用 Linker 連結所有目的程式 (「*.obj」) 成為可執行檔案「*.exe」。所幸，目前有 Code::Blocks 統整與管理以上瑣事。以下是使用 Code::Blocks 開啓新專案時，系統準備好的基本程式架構。

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    printf("Hello world!\n");
    return 0;
}
```

► 標頭檔

所有 C 語言程式都要載入兩個標頭檔

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

這些標頭檔內含許多由 C 語言事先編寫好的函式，例如內含許多輸出入函式供我們呼叫，只要載入它們兩個，我們就可以輕鬆作出輸出入的功能。例如：本例的

```
printf("Hello world!\n");
```

就可以輸出「Hello world!」。我們作為程式設計師，只要靈活呼叫這些函式，就可以完成許多常見的工作。

► 新增/刪除檔案

若要新增檔案，也是點選功能表「File/New/File」即可新增 C 函式或組合函式；若要修改函式，也是先點選函式名稱再修改。

► 執行程式

完成程式的編輯，最後點選「Build/Build and Run」即可編譯所有函式、連結所有目的檔，且執行程式。

以上是最簡單的 C 程式，我們剛剛強調 C 語言是由許多函式組合而成，以下則是自己新增函式的 C 程式架構，程式分為四大區塊，分別是函式引用區、全域變數宣告區、函式實作區、主程式區，如表 2-1 所示：

► 表 2-1 C 程式架構

標題	程式
函式引用區（載入所需標頭檔）	<code>#include <stdio.h></code> <code>#include <stdlib.h></code>
全域變數宣告區	<code>int s;</code>
函式實作區	<code>int add (int a, int b){</code> <code>int c;</code> <code>c = a + b;</code> <code>return (c);</code> <code>}</code>
主程式區 main()	<code>int main() {</code> <code>s = add (6, 2);</code> <code>printf("%d",s);</code> <code>return 0;</code> <code>}</code>

圖 2-2 是專案管理窗格，表示此專案，目前含兩個檔案，分別是 main.c 與 m3.c。

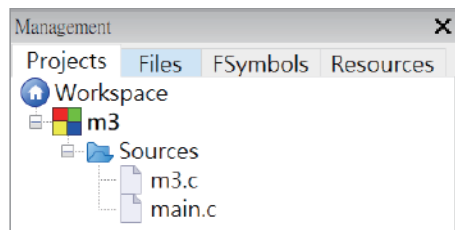


圖 2-2 專案管理窗格

圖 2-3 是以上專案的檔案結構。

名稱	修改日期	類型
bin	2022/6/16 下午 04:52	檔案資料夾
obj	2022/6/16 下午 04:52	檔案資料夾
m3.c	2022/6/28 下午 04:01	C 檔案
m3.cbp	2022/6/16 下午 04:32	CBP 檔案
m3.depend	2022/6/19 上午 06:17	DEPEND 檔案
m3.layout	2022/6/24 下午 03:22	LAYOUT 檔案
main.c	2022/6/24 下午 03:20	C 檔案

圖 2-3 專案檔案結構

.cbp 是專案檔案名稱，main.c 是一個函式檔，目的檔 (.obj) 在 obj 資料夾裡面，執行檔 (*.exe) 在 bin 資料夾裡面。

► C 語言架構

C 語言的指令分為資料型態的宣告、運算子、決策指令、迴圈指令、函式、指標，其語言指令架構如表 2-2。

► 表 2-2 C 語言指令架構表

C 語言指令架構		
指令分類	指令	本書章節
資料型態的宣告	常數 :const #define char,int,float,double,short,long,void,struct	3,4,9
運算子	算術運算 :+=, -=, *, /, %, ++, -- 關係運算 :>, <, >=, <=, ==, != 邏輯運算 : !, &&, 位元運算 : !, &, , ^, >>, <<	6
決策指令	if, else, switch, case	7
迴圈指令	for, while, continue, break	8
函式	函式	5, 11
指標	指標	10

以上指令對應章節如表 2-2 的「本書章節」，我們將會在以下各章節陸續介紹。

 自我練習

1. 以 C 語言開發程式的敘述，下列何者錯誤？（統測 111）
 - (A) 程式需要經過編譯及連結產生可執行檔，才能夠執行
 - (B) 使用 `#include` 前置處理命令時，命令結尾需要加上分號
 - (C) `main()` 是一個函式，程式執行時從 `main()` 函式開始執行
 - (D) 單行註解可用 2 個斜線 (`//`) 開頭

 2-3 演算法的認識 演算法基本概念

演算法 (algorithm) 是指電腦程式完成一項工作所需要『步驟』的集合。演算法嚴謹定義如下：

在有限 (finite) 的步驟 (step) 所構成的集合中，依照給定輸入 (input)，依序執行每個明確 (definite) 且有效 (effective) 的步驟，以便能夠解決特定的問題，而且步驟的執行必定會終止 (terminate)，並產生輸出 (output)。

 演算法的流程表示

常見的演算法流程表示有三種，分別是自然語言、虛擬碼與流程圖。分別說明如下：

► 自然語言 (Natural language processing)

自然語言就是使用我們日常生活的文字表示或已經熟悉的數學語言。例如，以下是求 9 開根號的演算法。

1. 使用迴圈從 1 到 9 分別求其平方。（此稱為循序法）
2. 若其平方大於等於 9，此數即為所求。例如：1 的平方是 1，2 的平方是 4，3 的平方是 9，9 已經大於等於 9，所以 3 即為所求，演算結束。

► 虛擬碼 (Pseudocode)

在自然語言中，有時嵌入一些慣用常用程式敘述，例如 `for` 代表迴圈，`if` 代表決策，這樣可以縮短文字長度，也會讓敘述更加清楚易懂。例如，以下是求 9 開根號的演算法。

```

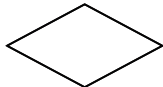
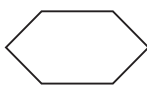
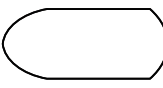
for i=1 to 9{
  y=i*i
  if y>=9 then
    print(i)
  end for
}

```

🔗 流程圖(flow chart)

流程圖 (flow chart) 是利用各種方塊圖形、線條及箭頭等符號來表達解決問題的步驟及進行的順序，常用的流程符號如表 2-3：

► 表 2-3 流程圖符號表

編號	符號	意義
1	 	起迄符號。 代表流程圖的開始或結束，此符號若是開始，則僅有一出口，若是結束則僅有一入口。
2		輸入與輸出符號。 用來填入輸入與輸出的符號，此符號有一入口與一出口。
3		處理符號。 用來填入處理程序，此符號有一入口與一出口。
4		決策符號。 用來表示決策分歧點，此符號的出口至少有兩個，分別是 true 或 false。
5		重複符號。 聲明重複指令的起點與離開條件，通常配合下面的連結符號表示重複的範圍。
6		連結符號。 可配合上面的重複符號或移至另一頁的起點。
7		副程式。(副程式又稱函式) 用來表示另一個副程式。
8		程式流向符號。 用來連結以上符號，說明程式的流向。
9		代表輸出至螢幕。 將資料由螢幕輸出。
10		代表輸出至印表機。 將資料由印表機輸出。

例如，圖 2-4 是求 9 開根號的演算法。

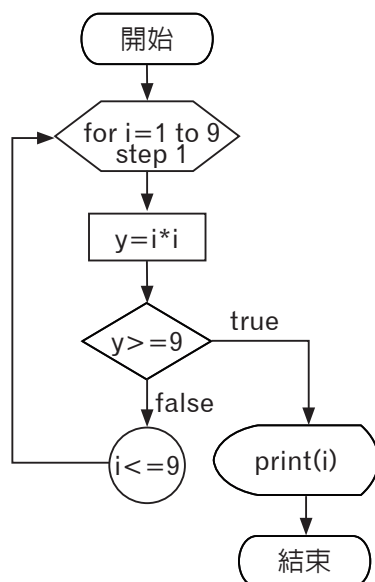


圖 2-4 開根號流程圖

又例如，圖 2-5 的流程圖，可以將輸入成績分類，分為 A、B、C、D 等。

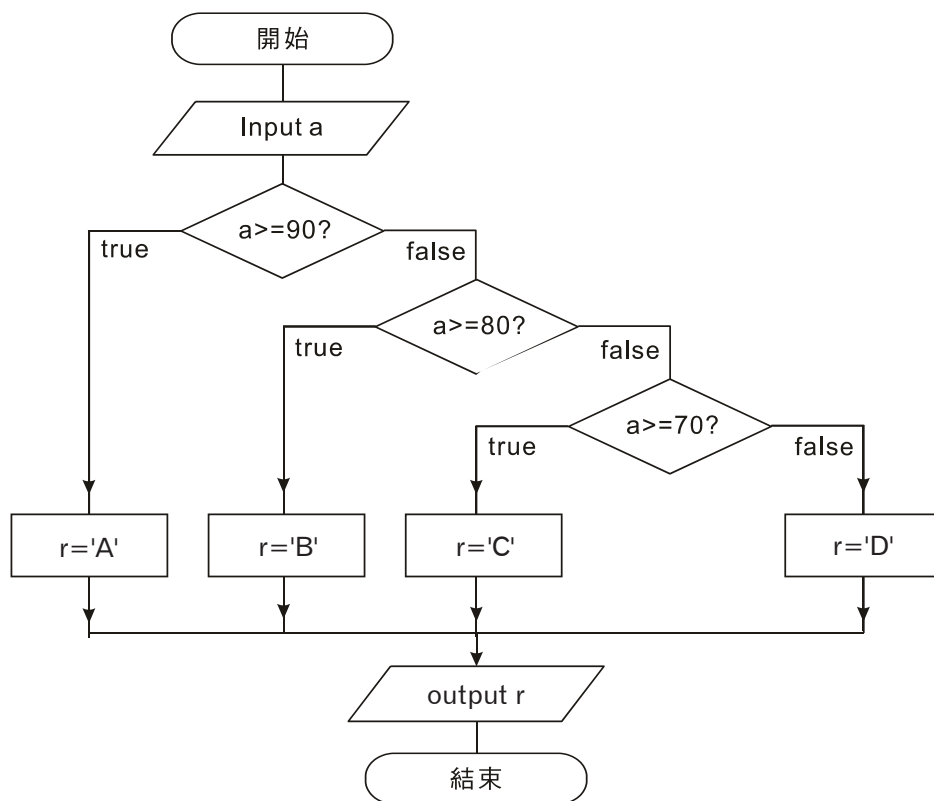


圖 2-5 成績分類流程圖

2-4 程式設計步驟

範例2-4a

已知長方形的長與寬，請寫一個程式，求其面積。

程式設計解題步驟

1. 資料的數位化

- (1) 本題若使用掌上型計算機，其方法是直接將鍵入「長 * 寬 =」，就可得到答案。例如，鍵入「15*8=」，就可得到答案「120」。
- (2) 但使用 C 語言，必須先將所要計算的值先以一個英文代號儲存，例如 a,b 或 score，以上代號在程式設計的領域，稱為「變數」。例如：

```
a=15  
b=8
```

設定變數的目的是將資料與程式分開，這樣當資料改變時，還可重複計算。計算過程都是以變數與程式指令描述，此稱為程式設計。程式設計的優點是有一致性，只要第一次對，往後都對。掌上型計算機就沒有了一致性了，每次都要按兩次，才能確認計算結果是否正確，且沒無法重複使用。以上變數的命名請看 4-1 節。

- (3) 選用資料型態。C 語言的資料型態請看 3-2 節，本例先選用 int(整數型態)，且每一個敘述都要以分號 (;) 結束，所以程式如下：

```
int a=15;  
int b=8;
```

2. 寫出演算法。本例是輸入長方形的長與寬來計算面積，所以演算法如下：

```
面積=a*b
```

3. 使用變數與程式指令，完成以上演算法。本例是計算面積，面積也要使用變數儲存，本例使用 c，所以是

```
c=a*b;
```

4. 輸出結果。

```
printf("面積=%d\n",c);
```

5. 以上全部程式如下：

```
int a=15;
int b=8;
int c;
c=a*b;
printf("面積=%d\n",c);
printf("%d",c);
printf(c)
```

6. 配合 C 語言的程式架構，全部程式如下：

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int a=15;
    int b=8;
    int c;
    c=a*b;
    printf("面積=%d\n",c);
    return 0;
}
```

2-5 開發環境介面

目前 C/C++ 語言的程式開發環境有 Code::Blocks、Dev-C++，因為 APCS（Advance Placement Computer Science, 大學程式設計先修檢測）考場採用 Code::Blocks，所以本書介紹 Code::Blocks 如下：

Code::Blocks

Code::Blocks 是目前最熱門的 C/C++ 自由整合式開發環境（IDE），Code::Blocks 官網（<https://www.codeblocks.org/>），其下載與安裝方式如下：

► 下載軟體

點選「Downloads」，畫面如圖 2-6：



圖 2-6 Code::Blocks 下載點一

點選「Download the binary release」，畫面如圖 2-7：（往下捲動）

File	Download from
codeblocks-20.03-setup.exe	FossHUB or Sourceforge.net
codeblocks-20.03-setup-nonadmin.exe	FossHUB or Sourceforge.net
codeblocks-20.03-nosetup.zip	FossHUB or Sourceforge.net
codeblocks-20.03mingw-setup.exe	FossHUB or Sourceforge.net
codeblocks-20.03mingw-nosetup.zip	FossHUB or Sourceforge.net
codeblocks-20.03-32bit-setup.exe	FossHUB or Sourceforge.net
codeblocks-20.03-32bit-setup-nonadmin.exe	FossHUB or Sourceforge.net
codeblocks-20.03-32bit-nosetup.zip	FossHUB or Sourceforge.net
codeblocks-20.03mingw-32bit-setup.exe	FossHUB or Sourceforge.net
codeblocks-20.03mingw-32bit-nosetup.zip	FossHUB or Sourceforge.net

圖 2-7 Code::Blocks 下載點二

初學者請直接點選「codeblocks-20.03mingw-setup.exe」，因為它才有內含「C/C++」編譯軟體，否則需自備編譯軟體。

► 安裝軟體

下載的檔案即是「.exe」執行檔，請在檔案總管「下載區」點按程式兩下，就可自動執行，開啓「Code::Blocks」，畫面如圖 2-8：

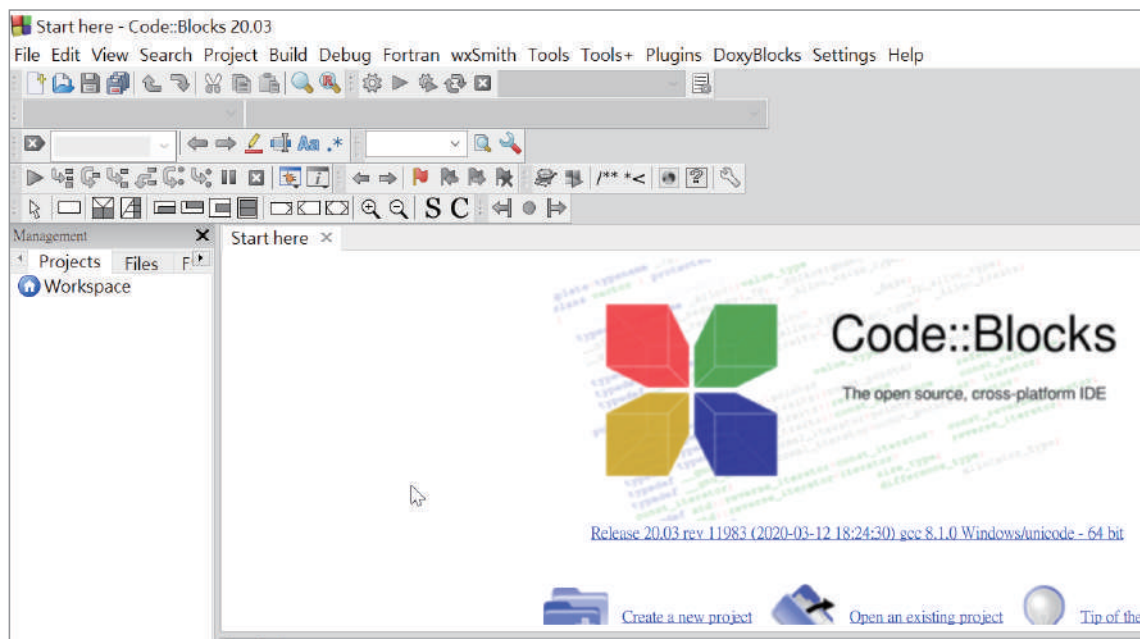


圖 2-8 Code::Blocks 初始畫面

範例2-5a

試寫一程式，將 2-4 節的面積計算程式輸入與執行。

👉 操作步驟

1. 開新專案。點選功能表的「File → New → Project」，畫面出現要求點選專案類型，本例點選『Console application』 / 『Go』，如圖 2-9。

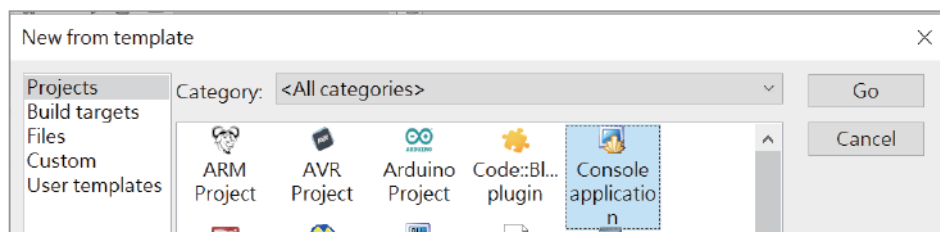


圖 2-9 開新專案一

5. 點選輸出路徑，本例使用預設值，如圖 2-13。

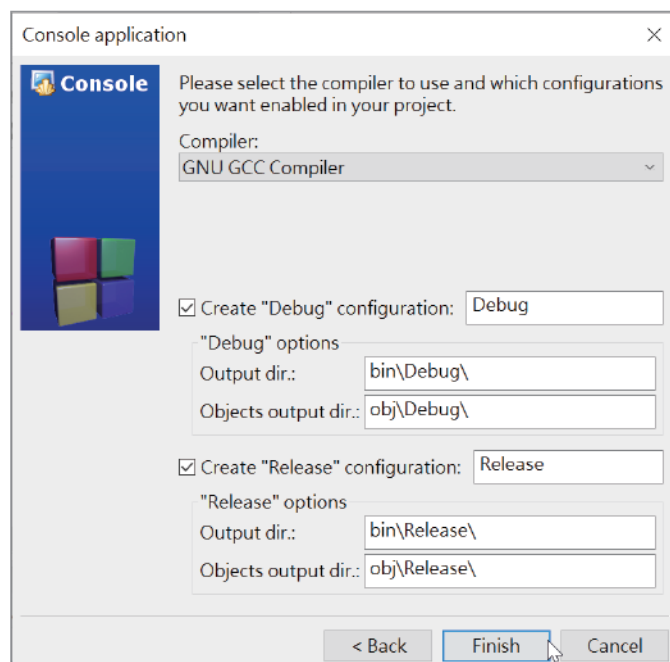


圖 2-13 開新專案五

6. 完成以上設定，並展開下圖 Sources 的 main.c，畫面如圖 2-14。

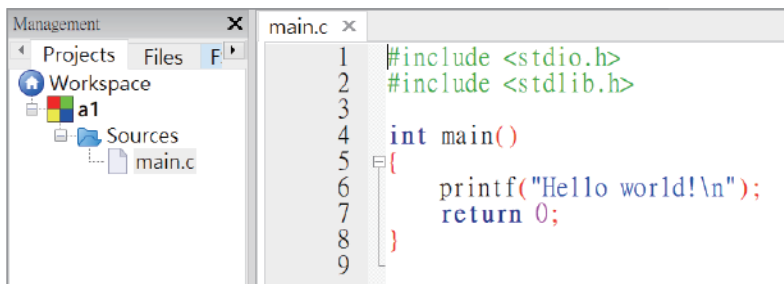


圖 2-14 main.c 程式架構

7. 編譯程式。點選功能表的「Build → Build」。
8. 執行程式。點選功能表的「Build → Run」，畫面如圖 2-15。
(以上 Step 7、8 兩個步驟，亦可點選功能表或工具列的「Build and Run」。)

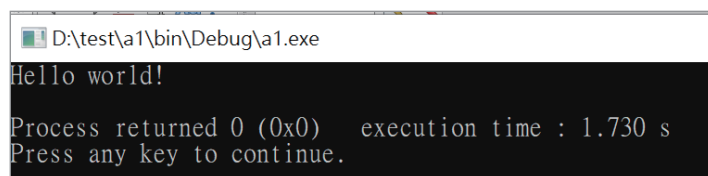


圖 2-15 執行結果

9. 以上「printf("Hello world! \n");」，就是本例所執行的程式，往後要寫任何的程式，只要將此敘述刪除，並在此位置寫下程式即可。
10. 開啓檔案總管，全部檔案如圖 2-16：（本例於 Step 4 的「Project title」輸入爲 a1，資料夾位置點選 test，系統會自動建立 a1 資料夾）。

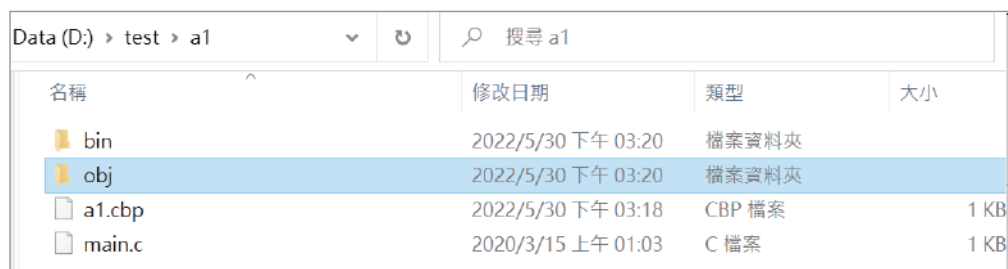


圖 2-16 C 專案檔案結構

11. 調整編輯器字型大小。點選功能表「Settings」/「Editor」，畫面如圖 2-17，繼續點選「Choose」，即可依照指示完成字型大小的設定。

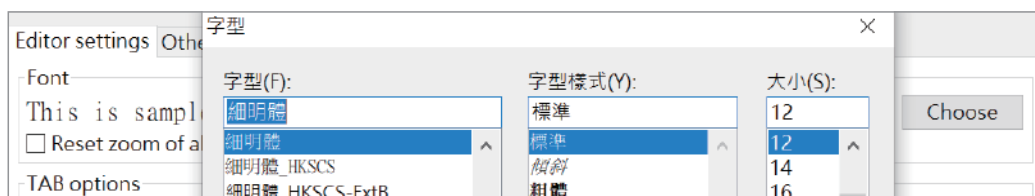


圖 2-17 Code::Block 編輯字型設定

12. 以下是開新專案時，main.c 的程式樣版。

```
#include <stdio.h>
#include <stdlib.h>
int main(){
    printf("Hello world!\n");
    return 0;
}
```

13. 以下程式可計算長方形的面積。程式執行結果

```
int a=15;
int b=8;
int c;
c=a*b;
printf("面積=%d\n",c);
```

14. 將 printf() 函式刪除，並鍵入以上程式。全部程式如下：

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int a=15;
    int b=8;
    int c;
    c=a*b;
    printf("面積=%d\n",c);
    return 0;
}
```

15. 執行程式。點選功能表「Build/Build and Run」，並觀察執行結果。若有錯誤，請修改，再執行程式。

自我練習

1. 請寫一個程式，可以指派長方體的長、寬、高，並計算表面積與體積。

資料的種類與資料型態

chapter
03

學習綱要

☞ 3-1 資料的種類

☞ 3-2 資料型態

學習目標

- ☞ 1. 能瞭解我們日常生活中資料的種類。
- ☞ 2. 能寫出 C 語言資料型態。

3-1 資料種類

C 語言所能處理的資料種類分別有數值（含整數、長整數及浮點數等等）、字元與字串、布林值等。

整數(Integers)

C 語言可以處理的整數有三種進位方式，分別是十進位 (Decimal)、十六進位 (Hexadecimal) 及八進位 (Octal)。其中十進位則以我們平常書寫數字的方式即可；十六進位應以數字 0X 或 0x 開頭（數字 0），且以 0,1,2,3,4,5,7,8,9,a,b,c,d,e,f 代表十進位的 0 到 15，英文字元 a,b,c,d,e,f 大小寫都可以，例如 0xA 或 0Xf 均為十六進制，分別代表十進位的 10 與 15；八進位則應以數字 0 開頭，例如 :072 則為八進位，等於十進位的 58。例如：請鍵入以下程式，寫出執行結果。

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int a=10;
    printf("%d\n",a); //_____
    int b=010;
    printf("%d\n",b); //_____
    int c=0x10;
    printf("%d\n",c); //_____
    return 0;
}
```

浮點實數(Floating-point literals)

數字中含有小數點或指數的稱為浮點數、實數或浮點實數。以指數為例，E 或 e 表示 10 的次方，例如 0.0023、2.3E-3 及 2.3e-3 都是表示相同的浮點數；又例如 2.3E+2 則代表 230。浮點數可使用標準寫法或科學符號法表示，例如 321.123 即為標準寫法，1.23e+4 即為科學符號表示法。

字元(Character literals)

使用一對單引號『 ' 』圍住的單一字元，稱為字元，例如 'A' 或 'a' 等。C 語言使用 8 bits 表示一個字元，所以可表示 256 個字元 ($2^8=256$)，包含大小寫英文字母、數字、標點符號及其它特殊符號。表 3-1 是 ASCII 編碼表，僅用七位元即可表示所有電腦可用字元。

► 表 3-1 ASCII 編碼表

<https://en.cppreference.com/w/cpp/language/ascii>

ASCII Chart

The following chart contains all 128 ASCII decimal (**dec**), octal (**oct**), hexadecimal (**hex**) and character (**ch**) codes.

dec	oct	hex	ch	dec	oct	hex	ch	dec	oct	hex	ch	dec	oct	hex	ch
0	0	00	NUL (null)	32	40	20	(space)	64	100	40	@	96	140	60	`
1	1	01	SOH (start of header)	33	41	21	!	65	101	41	A	97	141	61	a
2	2	02	STX (start of text)	34	42	22	"	66	102	42	B	98	142	62	b
3	3	03	ETX (end of text)	35	43	23	#	67	103	43	C	99	143	63	c
4	4	04	EOT (end of transmission)	36	44	24	\$	68	104	44	D	100	144	64	d
5	5	05	ENQ (enquiry)	37	45	25	%	69	105	45	E	101	145	65	e
6	6	06	ACK (acknowledge)	38	46	26	&	70	106	46	F	102	146	66	f
7	7	07	BEL (bell)	39	47	27	'	71	107	47	G	103	147	67	g
8	10	08	BS (backspace)	40	50	28	(	72	110	48	H	104	150	68	h
9	11	09	HT (horizontal tab)	41	51	29	)	73	111	49	I	105	151	69	i
10	12	0a	LF (line feed - new line)	42	52	2a	*	74	112	4a	J	106	152	6a	j
11	13	0b	VT (vertical tab)	43	53	2b	+	75	113	4b	K	107	153	6b	k
12	14	0c	FF (form feed - new page)	44	54	2c	,	76	114	4c	L	108	154	6c	l
13	15	0d	CR (carriage return)	45	55	2d	.	77	115	4d	M	109	155	6d	m
14	16	0e	SO (shift out)	46	56	2e	.	78	116	4e	N	110	156	6e	n
15	17	0f	SI (shift in)	47	57	2f	/	79	117	4f	O	111	157	6f	o
16	20	10	DLE (data link escape)	48	60	30	0	80	120	50	P	112	160	70	p
17	21	11	DC1 (device control 1)	49	61	31	1	81	121	51	Q	113	161	71	q
18	22	12	DC2 (device control 2)	50	62	32	2	82	122	52	R	114	162	72	r
19	23	13	DC3 (device control 3)	51	63	33	3	83	123	53	S	115	163	73	s
20	24	14	DC4 (device control 4)	52	64	34	4	84	124	54	T	116	164	74	t
21	25	15	NAK (negative acknowledge)	53	65	35	5	85	125	55	U	117	165	75	u
22	26	16	SYN (synchronous idle)	54	66	36	6	86	126	56	V	118	166	76	v
23	27	17	ETB (end of transmission block)	55	67	37	7	87	127	57	W	119	167	77	w
24	30	18	CAN (cancel)	56	70	38	8	88	130	58	X	120	170	78	x
25	31	19	EM (end of medium)	57	71	39	9	89	131	59	Y	121	171	79	y
26	32	1a	SUB (substitute)	58	72	3a	:	90	132	5a	Z	122	172	7a	z
27	33	1b	ESC (escape)	59	73	3b	;	91	133	5b	[	123	173	7b	{
28	34	1c	FS (file separator)	60	74	3c	<	92	134	5c	\	124	174	7c	
29	35	1d	GS (group separator)	61	75	3d	=	93	135	5d	]	125	175	7d	}
30	36	1e	RS (record separator)	62	76	3e	>	94	136	5e	^	126	176	7e	~
31	37	1f	US (unit separator)	63	77	3f	?	95	137	5f	_	127	177	7f	DEL (delete)

字串

C 語言中，字串使用一對雙引號『 " 』圍住，例如 "Gwosheng", "台灣" 等。

布林值

我們人類表達對與錯的語句有：對、錯、是、否、正確、錯誤。C 語言則統一使用 1 代表布林的 true，0 代表布林的 false。

 自我練習

1. 如果 X_n 代表 X 這個數字是 n 進位，請問 $D02A16 + 548710$ 等於多少？（APCS10503）
 - (A) 1100 0101 1001 10012
 - (B) 1626318
 - (C) 5878716
 - (D) F59916

 3-2 資料型態 (Data Type)

前面已經介紹人類經常使用的一些資料常數，每一種資料常數均需要不同大小的記憶體儲存，電腦爲了有效率的儲存與處理這些資料，所以有資料型態的規劃。也就是大的資料用大盒子裝，小的資料用小盒子裝，如此即可節省記憶體，並加快處理效率。反過來說，若不分資料大小，通通大盒子裝資料，那將會非常浪費記憶體，也拖慢執行效率。

C 語言的資料型態首先分爲內建資料型態及複合資料型態，內建資料型態再分爲數值 (int、long、float、double、long double)、字元 (char) 資料型態及函式專用型態 (void)，複合資料型態再分爲陣列 (array)、結構 (struct) 等型態。

 數值資料型態

表 3-2 是 C 語言的數值資料型態，unsigned 表無號整數，即非負數整數：

► 表 3-2 C 語言資料型態表

數值資料型態	佔用記憶體的大小(位元)	所能代表的數值的範圍
short	16	-32,768 ~ +32,767
int	32	-2,147,483,648 ~ +2,147,483,647
long	32	-9223372036854775808 to 9223372036854775807
unsigned short	16	0 ~ 65,535
unsigned int	16	0 ~ 4,294,967,295

數值資料型態	佔用記憶體的大小(位元)	所能代表的數值的範圍
unsigned long	32	0 to 18446744073709551615
float	32	$3.4 \times 10^{-38} \sim 3.4 \times 10^{38}$
double	64	$1.7 \times 10^{-308} \sim 1.7 \times 10^{308}$
long double	128	$3.4 \times 10^{-4932} \sim 3.4 \times 10^{4932}$

♂ 整數

我們可以使用 signed(有號)與 unsigned(無號)的 short、int 及 long 等六種型態來儲存整數。例如，一天的秒數是 86400，已經超過整數 short 的範圍，那就要用 int 表示。但真是太複雜了，因此本書整數一律使用整數預設型態的 int 型態，其所能代表的數值的範圍是 -2,147,483,648 ~ +2,147,483,647。

♂ 浮點數

我們可以使用 float、double 及 long double 等三種型態來儲存浮點常數。浮點數的預設值是 double，但若要強制使用 float，則應於數值後面加上英文字母的 F 或 f，例如，3.1415926F 或 3.1415926f。但若要使用 long double，則於數值後面加上 L 或 l，例如，3.1415926L 或 3.1415926l。以上如此細分，對初學者來說過於複雜，因此本書採用折衷的方式，浮點數一律使用 double(因為 C 語言大部分數學函式傳回 double 型態)。所以本書關於數值型態，取常用資料型態如表 3-3：也就是遇到整數取 int，遇到浮點實數取 double。

► 表 3-3 常用資料型態表

數值資料型態	佔用記憶體的大小(位元)	所能代表的數值的範圍
int	32	-2,147,483,648 ~ +2,147,483,647
double	64	$1.7 \times 10^{-308} \sim 1.7 \times 10^{308}$

♂ 字元資料型態

C 語言使用 char 資料型態來儲存一個 ASCII 字元，因此，C 語言中的 char 型態是 8 位元長。

布林型態

C 語言設計之初並沒有布林型態，後來才新增 `_Bool` 布林型態。

字串資料型態

C 語言並無字串資料型態，而是使用字元陣列，例如：

```
char a[10]="gwosheng";
```

關於字元陣列，請看 9-3 節。

自我練習

1. 請於 <https://en.cppreference.com/w> 點選『Basic concepts』，畫面如圖 3-1：

C / C language / Basic Concepts

Basic concepts

This section provides definitions for the specific terminology and the concepts used when describing the C programming language.

A C program is a sequence of text files (typically header and source files) that contain **declarations**. They undergo **translation** to become an executable program, which is executed when the OS calls its **main function** (unless it is itself the OS or another *freestanding* program, in which case the entry point is implementation-defined).

Certain words in a C program have special meaning, they are **keywords**. Others can be used as **identifiers**, which may be used to identify **objects**, **functions**, **struct**, **union**, or **enumeration** tags, their members, **typedef** names, **labels**, or **macros**.

Each identifier (other than macro) is only valid within a part of the program called its **scope** and belongs to one of four kinds of **name spaces**. Some identifiers have **linkage** which makes them refer to the same entities when they appear in different scopes or translation units.

Definitions of functions include sequences of **statements** and **declarations**, some of which include **expressions**, which specify the computations to be performed by the program.

Declarations and **expressions** create, destroy, access, and manipulate **objects**. Each **object**, **function**, and **expression** in C is associated with a **type**.

See also
[C++ documentation](#) for **Basic concepts**

圖 3-1 線上手冊一

2. 繼續於上圖點選『Type』，即可得所有資料型態的詳細資料，如圖 3-2：

Type classification

The C type system consists of the following types:

- the type `void`
- basic types
 - the type `char`
 - signed integer types
 - standard: `signed char`, `short`, `int`, `long`, `long long` (since C99)
 - extended: implementation defined, e.g. `__int128`
 - unsigned integer types
 - standard: `_Bool` (since C99), `unsigned char`, `unsigned short`, `unsigned int`, `unsigned long`, `unsigned long long` (since C99)
 - extended: implementation-defined, e.g. `__uint128`
 - floating types
 - real floating types: `float`, `double`, `long double`
 - complex types: `float _Complex`, `double _Complex`, `long double _Complex`
 - imaginary types: `float _Imaginary`, `double _Imaginary`, `long double _Imaginary`
- enumerated types
- derived types
 - array types
 - structure types
 - union types
 - function types
 - pointer types
 - atomic types

圖 3-2 線上手冊二

跳脫字元序列

字元中的單引號 (')、雙引號 (") 及反斜線 (\) 在 ASCII 均已有定義其功能，若您一定要使用這些字元，則應於使用前先加一個反斜線 (\)，以跳脫其原先所定義的功能，此稱為跳脫字元序列 (Escape Sequence)。表 3-4 是一些常用的跳脫字元：

► 表 3-4 跳脫字元序列表

字元	跳脫字元序列
單引號 (')	\'
雙引號 (")	\"
反斜線 (\)	\\

例如：

```
char a[]="Gwosheng";
printf("%s\n",a);
```

結果是 Gwosheng。又例如：

```
char b[]="\Gwosheng\";
printf("%s\n",b);
```

結果是 "Gwosheng"。

自我練習

1. 鍵入以下程式，並觀察執行結果。

題號	程式	結果
1	<pre>#include <stdio.h> #include <stdlib.h> int main() { char a[]="Gwosheng"; printf("%s\n",a); char b[]="\Gwosheng\"; printf("%s\n",b); return 0; }</pre>	

2. 關於 C 程式語言的資料型態，下列敘述何者錯誤？（統測 111）
 - (A) float 資料型態可以儲存浮點數，數值精確度跟 double 資料型態相同
 - (B) 宣告 int 資料型態可以儲存整數資料
 - (C) double 資料型態可以儲存浮點數值
 - (D) 宣告 char 資料型態可以儲存字元符號

變數與常數

Chapter

04

學習綱要

- 📖 4-1 識別字與保留字
- 📖 4-2 變數和常數宣告與應用
- 📖 4-3 資料型態轉換
- 📖 4-4 資料型態應用實例

學習目標

- 📖 1. 認識識別字與保留字。
- 📖 2. 能宣告變數與常數。
- 📖 3. 能實作資料型態轉換。
- 📖 4. 能實作資料型態應用。

4-1 識別字與保留字

識別字(Identifiers)

真實的世界裏，每個人、事及物都有一個名稱，程式設計亦不例外，於程式設計時我們必須為每一個變數、常數、函式、結構及類別命名，以上所有變數、常數、函式、結構及類別等名稱，統稱為程式語言的識別字。C 語言的識別字命名規則如下：

1. 識別字必須是以字母（大小寫的 A 至 Z）或底線（_）開頭。例如，以下是一些合法的識別字。

```
a
i
sum
_sum
Income
```

以下是一些非法的識別字。

```
7eleven    /* 不能由數字開頭 */
%as        /* 不能由符號開頭 */
```

2. 識別字由字母開頭後，僅可由字母、底線及數字組合而成，但不得包含空白。例如，以下是一些合法的識別字。

```
a123
a123b
_a_b
```

以下是一些非法的識別字。

```
A=          /* 不能含有 = 號 */
sum!        /* 不能含有 ! 號 */
Age#3       /* 不能含有 # 號 */
a c         /* 不能含空白 */
c+3         /* 不得含有加號 */
```

3. 識別字的長度不可以超過 32 個字元。
4. 識別字的大小寫均視為不同，例如 Score、score 及 SCORE 皆代表不同的識別字。
5. 識別字不得使用保留字，如 if、for 等。但是，if1、fora 等則可使用。
6. 識別字亦可使用有意義的單字的組合，這樣更能傳達變數所代表的意義。例如：StudentNumber 或 AverageIncome。除非有效範圍很小（或稱生命週期極短）的變數才用 x、i 或 a 等當識別字，也千萬不要用 k23erp 等這種沒意義又難記的識別字。
7. 識別字有多個單字時，中間可以加上底線（_），例如上例的 StudentNumber 可寫成 Student_Number，若擔心單字拼錯亦可寫成 Stu_Num、stu_num、StuNum 或 stumum，其中 StuNum 又稱駝峰表示法，因為大寫字母看起來像駱駝駝峰一樣，可以避免鍵入底線的困擾、且提昇閱讀效率。
8. 不要單獨使用 l,o,O 等字母當作變數名稱，因為這些字母與阿拉伯數字 1,0 很相近，很容易讓人誤判。

♂ 保留字(Keywords)

保留字（又稱關鍵字）是任一程式語言已事先賦予某一識別字（可識別的文本或字串，稱為識別字）一個特別意義，所以程式設計者不得再重複賦予不同的用途。例如 if 已賦予決策敘述，程式設計者當然不得再定義 if 為另外的用途，表 4-1 是 C 語言的保留字，請於 <https://en.cppreference.com/w> 點選『C Keywords』。

► 表 4-1 C 語言保留字

auto break case char const continue default do double else enum extern	float for goto if inline (since C99) int long register restrict (since C99) return short	signed sizeof static struct switch typedef union unsigned void volatile while	_Alignas (since C11) _Alignof (since C11) _Atomic (since C11) _Bool (since C99) _Complex (since C99) _Generic (since C11) _Imaginary (since C99) _Noreturn (since C11) _Static_assert (since C11) _Thread_local (since C11)
---	--	---	--

4-2 變數與常數宣告與應用

變數的宣告

變數的功能是用來輸入、處理及儲存外界的資料，而變數在使用以前均要事先宣告才可使用。C 語言的變數宣告語法如下：

```
資料型態 變數名稱[=初值];
```

例如：

```
int a;
```

即是宣告變數 a 為 int 型態。又例如：

```
char b,c;
```

則是宣告變數 b,c 為 char 型態。變數的宣告亦可連同初值一起設定，如下所示：

```
double d = 30.2;
```

宣告 d 是 double 型態，並設其初值是 30.2。以下程式，宣告 e 是 char 型態，且其初值是 'a'。

```
char e='a';
```

以下敘述，同時宣告兩個整數，且指派其初值。

```
int f1=3,f2=4;
```

以下敘述可宣告變數為字元陣列，字元陣列就能代表字串。(陣列於 9-3 節有更詳細說明)

```
char g[]="Gwosheng";
```

變數宣告應用

變數經過宣告之後，編譯器即會根據該變數的資料型態配置適當的記憶體儲存此變數，所以若要提高程式的執行效率，則應盡量依照資料性質，選擇佔用記憶體較小的資料型態。在一些舊式的 Basic 語言中，變數使用前並不需要事先宣告，卻也帶來極大的困擾。以下敘述即為變數未宣告的結果，編譯器便無法回應使用者在拼字上的錯誤，而造成除錯上的困難。

```
student=studend+1;
```

上式若事先宣告 student 如下：

```
int student;
```

則編譯器遇到 studend 時，便會出現 studend 未宣告的錯誤訊息，提醒使用者補宣告或注意拼字錯誤。其次，變數宣告的優點是可配置恰當的記憶體而提高資料的處理效率。例如，有些變數的值域僅為整數，則不用宣告為 float 或 double。此即為小東西用小箱子裝，大東西用大箱子裝，才能有效運用空間與提升搬運效率。

自我練習

1. 請鍵入以下程式，編譯、除錯與執行程式。

題號	程式	結果
1	<pre>#include <stdio.h> #include <stdlib.h> int main() { int 7eleven; // 不能由數字開頭 int %as; // 不能由符號開頭 int _a; int A=; // 不能含有 = 號 int sum! ; // 不能含有 ! 號 int Age#3; // 不能含有 # 號 int a c ; // 不能含空白 int c+3 ; // 不得含有加號 int if; // 不得使用保留字 return 0; }</pre>	

```

2  #include <stdio.h>
    #include <stdlib.h>
    int main() {
        int student=0;
        student=student+1;
        printf("%d",student);
        int a=3.4;
        char b,c;
        b="aa"
        c="a";
        int d='a';
        double e=3;
        return 0;
    }

```

♂ 常數的宣告

程式設計有兩種表示常數的方式，一種是文字式 (Literal)，例如直接以 15 或 3.14159 表示某一常數；另一種是本單元的常數符號式 (Symbolic)。因為有些數字在程式中會不斷的重複出現，為了增加程式的可讀性及減少程式鍵入的麻煩，此時即可用一個有意義的符號代替，我們可以利用 `const` 或 `define` 來定義常數，則該符號的值將永遠保持在你所宣告的符號，程式中任何位置均不可能改變其值，此稱為常數符號，簡稱常數。`const` 宣告常數，語法如下：

```
const 型態 常數名稱=常數值;
```

例如，以下程式令 `PI=3.14`，常數符號通常均用大寫表示。

```
const double PI=3.14;
```

則每次要使用 3.14 時，只要填入 `PI` 即可。例如，以下程式可計算圓面積。

```
a=3*3*PI;
```

`define` 定義常數語法如下：

```
#define 常數名稱 常數值
```

例如：

```
#define PI 3.14;
```

♂ 常數宣告的應用

又例如，於銀行利息的計算，利率亦應以常數符號表示，程式中任何位置若需使用此利率時，均應以此常數符號代替，當要調整此利率時，只要在程式的最前面修改此常數符號的值即可。若未使用常數符號統一此值，則因此常數散落程式各地，而必須到處修改，萬一有些地方漏掉未修改，則無法確保此值的一致性。

👉 自我練習

1. 請鍵入以下程式，並寫出執行結果。

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    const double PI=3.14;
    double a;
    a=3*3*PI;
    printf("%f\n",a);
    const int b=4;
    b=b+1;//常數不能改變其值
    printf("%d",b);
    return 0;
}
```

2. 請鍵入以下程式，並寫出執行結果。

```
#include <stdio.h>
#include <stdlib.h>
#define PI 3.14
int main() {
    double a;
    a=3*3*PI;
    printf("%lf\n",a);
    PI=PI+1;//常數不能改變其值
    printf("%lf",PI);
    return 0;
}
```

3. 關於 C 程式語言中，使用 `define` 建立常數的方式，下列何者正確？
（111 年統測）
- (A) `define PI = 3.14;`
 (B) `define PI 3.14;`
 (C) `#define PI = 3.14`
 (D) `#define PI 3.14`

4-3 資料型態轉換

每一個變數宣告之後，即有屬於自己的型態，往後此變數均只能指派給相同或相近型態的變數儲存，若執行階段欲指定給相近型態的變數儲存，則稱此為型態轉換。在 C 語言中型態轉換又可分為隱含轉換（Implicit Conversion）與強制轉換（Explicit Conversion），分別說明如下：

隱含轉換（Implicit Conversion）

將值域小的型態轉為值域大的型態，稱為自動轉換或轉型（Convert）。此種轉換，系統可自動處理並確保資料不會流失。例如，將 `int` 轉為 `long`，則因後者的值域比前者大，所以可順利的轉換。以下敘述可將型態為 `int` 的變數 `a` 指派給型態為 `long` 的變數 `b`，且原值不會改變。

```
int a=23;
long b;
b=a;
printf("%d",b);
```

結果是 23。

強制轉換（Explicit Conversion）

將值域大的變數轉為值域小的變數（如 `long` 轉為 `int`），則稱此為強制轉換或稱為鑄型（cast）。強制轉換的語法如下：

```
變數2（值域小） = （變數2的型態） 變數1（值域大）；
```


以上可將變數 1（值域大）的型態強制轉為變數 2（值域小）。例如，以下敘述可將型態是 long 的 c 變數指定給型態是 int 的 d 變數。

```
long c=23;
int d;
d=(int) c;
printf("%d\n",d);
```

結果是 23。強制轉換的風險比較大，有可能資料流失或溢位。例如，以下敘述將 float 型態強制轉換為 int 型態，將造成小數部份的流失。

```
float e=3.4f;
    //浮點數預設為double，加上f，表示此3.4為float，請複習3-2節浮點數
int k;
k=(int) e;
printf("%d\n",k); //3
```

結果 3。其次，關於數值與字串互轉，則要看 11-3 的字串函式。

自我練習

1. 下列 C 語言程式碼片段執行後，x 與 y 的結果為何？（統測 111）

```
int x, a = 7, b = 2;
float y;
x = a/b;
y = (float) a/b;
```

4-4 資料型態應用實例

範例4-4a

假如有資料如下，請問如何以電腦儲存？

座號	姓名	國文成績	數學成績	總分	平均
1	洪煥維	98	95		

程式設計步驟

1. 資料數位化且以變數儲存。
 - (1) 座號取 int 型態，變數名稱取 no。
 - (2) 姓名取 char 型態，變數名稱取 name。
 - (3) 國文成績取 int 型態，變數名稱取 chi。
 - (4) 數學成績取 int 型態，變數名稱取 math。
 - (5) 總分成績取 int 型態，變數名稱取 total。
 - (6) 平均取 double 型態，變數名稱取 avg。
2. 程式設計如下：

```
#include <stdio.h>
#include <stdlib.h>
int main(){
    int no=1;
    char name[]="洪峻維";
    int chi=98;
    int math=95;
    int total;
    double avg;
    return 0;
}
```

自我練習

1. 假設有資料如下，請問如何以電腦儲存？

單字	詞性	中文
good	adj	美好的

基本輸出/輸入函式

學習綱要

- 📖 5-1 printf()
- 📖 5-2 scanf()
- 📖 5-3 亂數的取得
- 📖 5-4 聲音的產生

學習目標

- 📖 1. 能夠以 printf() 輸出資料。
- 📖 2. 能夠以 scanf() 輸入資料。
- 📖 3. 能取以 rand() 取得亂數。
- 📖 4. 能夠以 beep() 產生聲音。
- 📖 5. 能實作基本輸出與輸入函式。

C 語言採用 `printf()` 與 `scanf()` 作為基本輸出入函式。其次，本單元也要介紹亂數的取得，與聲音的產生，請看以下說明。

5-1 printf()

`printf()` 可輸出結果，其中括號內須放置一對雙引號『“”』來將欲輸出的資料當作引數傳給 `printf()` 函式。例如：

```
printf("a");
printf("aa");
```

的輸出結果是：

```
aaa
```

但若要換行與歸位，則應加上跳脫字元『\n』，例如：

```
printf("a\n");
printf("aa");
```

的輸出結果是：

```
a
aa
```

其次，如果是變數的輸出，則必須在輸出敘述中加入一個列印格式，此列印格式以控制字元『%』符號開頭，如表 5-1：

► 表 5-1 printf() 列印格式表

資料型別	列印格式
整數	%d
長整數	%ld
浮點數 float	%f
浮點數 double	%lf
字元	%c
字串	%s

然後編譯器會依照控制字元的順序，到後面引數區取對應資料來輸出。例如：

```
int a=3;
printf("%d", a);
```

『%』即表示要到後面引數區取變數來取代，『d』是將變數視為整數，所以輸出結果是：

```
3
```

若是：

```
int a=3, b=5;
printf("%d %d", a, b);
```

有兩個控制字元，那就到後面引數區依照順序取變數輸出，所以輸出是：

```
3 5
```

字串區也可以同時有一般字元與控制字元『%』，此時一般字元就直接輸出，控制字元就到後面引數區依序取變數輸出。例如：

```
int a=3, b=5;
printf("a=%d b=%d", a, b);
```

因為『"a= "』是一般字元，就直接輸出，『% d』是控制字元，就去引數區按照順序取對應變數來取代。第一個『%d』取到 a，第二個『%d』取到 b，所以輸出結果是：

```
a=3      b=5
```

輸出過程中如須換行與歸位，也是以跳脫字元序列『\n』來完成。例如：

```
int a=3, b=5;
printf("a=%d\n", a);
printf("b=%d", b);
```

的結果是：

```
a=3
b=5
```

自我練習

1. 請自行鍵入以下程式，並觀察執行結果。

```
int main() {
    int a=97;
    printf("%d\n", a);
    printf("%c\n", a);
    printf("%c\n", (char)a);
    char b='a';
    printf("%c\n", b);
    printf("%d\n", b);
    printf("%d\n", (int)b);
    char c='ab';
    printf("%c\n", c);
    char d='abc';
    printf("%c\n", d);
    char e='a ';
    printf("%c\n", e);
    return 0;
}
```

資料的對齊

輸出資料時，亦可指定資料長度，那就可以將所有資料對齊。例如：

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int a=3,b=55;
    double c1=3.4,c2=15.678;
    printf("a =%8d,b =%8d\n",a,b); //佔8格，靠右
    printf("c1=%8.2f,c2=%8.2f",c1,c2); //共8格，靠右，小數點取2位
    //請留意遇到『%』就是控制字元，就到後面按順序取對應資料輸出
    //非控制字元，就直接輸出，例如『a=』，就直接輸出。
    return 0;
}
```

的執行結果如下圖，這樣就可以對齊所有資料。

```
a =      3, b =      55
c1=    3.40, c2=    15.68
```

網路C語言使用手冊

網路 C 語言的使用手冊很多，但是，<https://en.cppreference.com/w> 是我目前找到最齊全的參考手冊，有 C 也有 C++ 如圖 5-1：

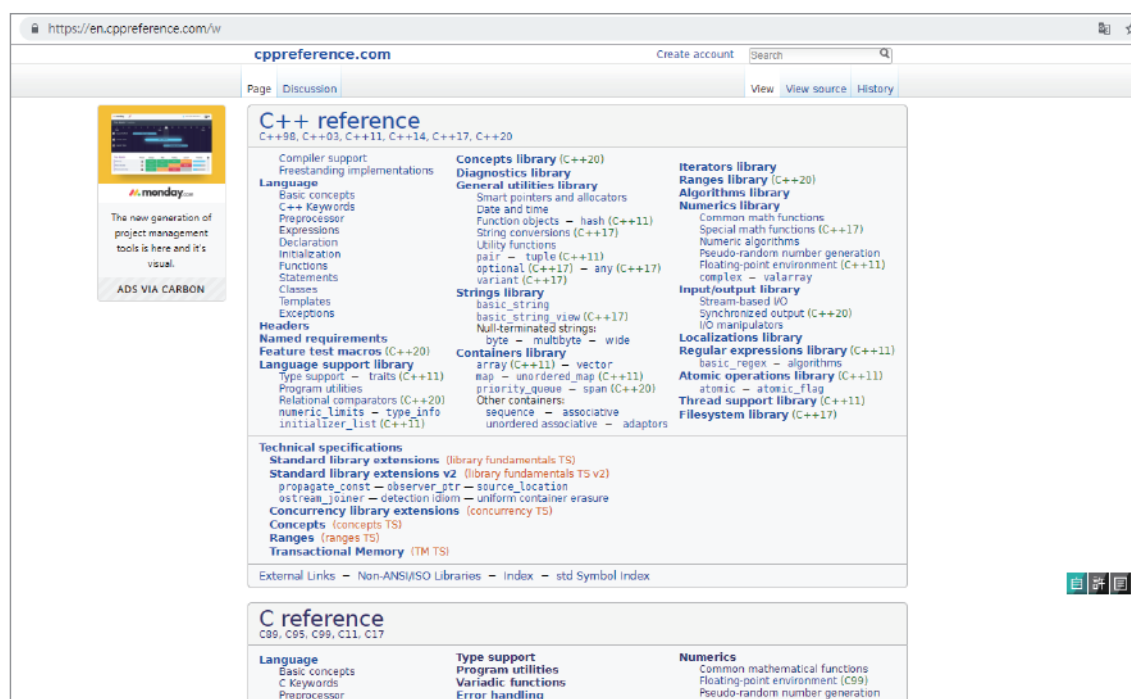


圖 5-1 C 語言網路使用手冊一

C 語言的詳細資料如圖 5-2（點選圖 5-1 的下面「C reference」）：上一章曾提到編譯器版本問題，請看下圖就會明瞭，若標示『C99』表示那是 1999 新增的指令；若標示『C11』表示那是 2011 新增的指令，沒標示則是 1990 制訂的標準。

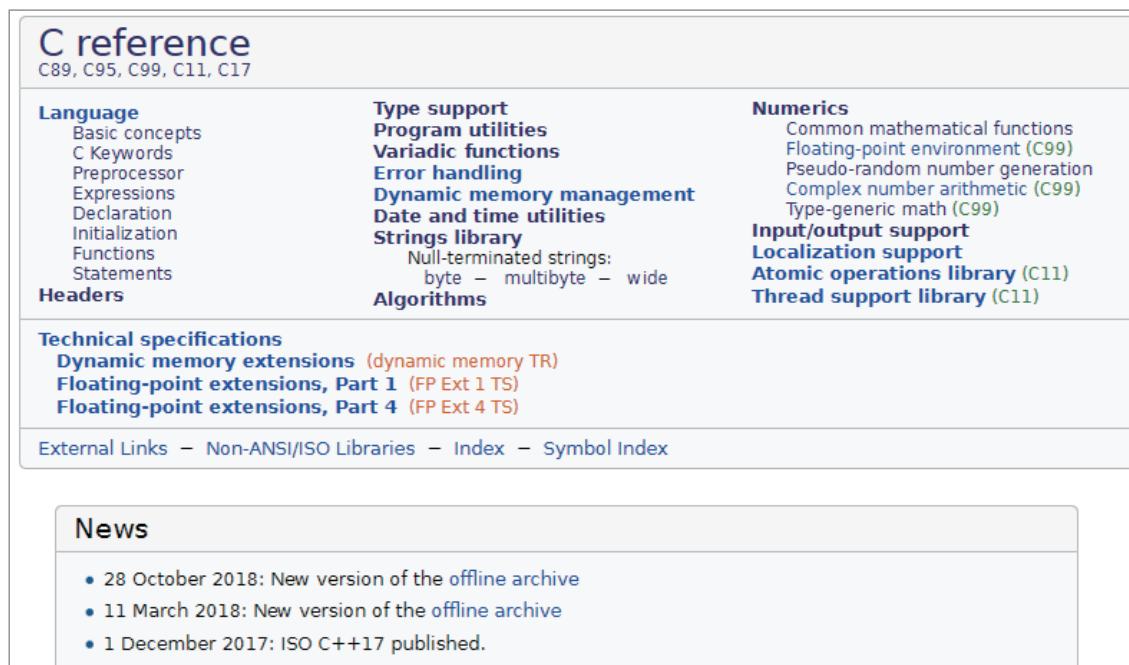


圖 5-2 C 語言網路使用手冊二

於圖 5-2 點選『input/output support』，再往下捲動，即可找到 printf() 的詳細說明，如圖 5-3：

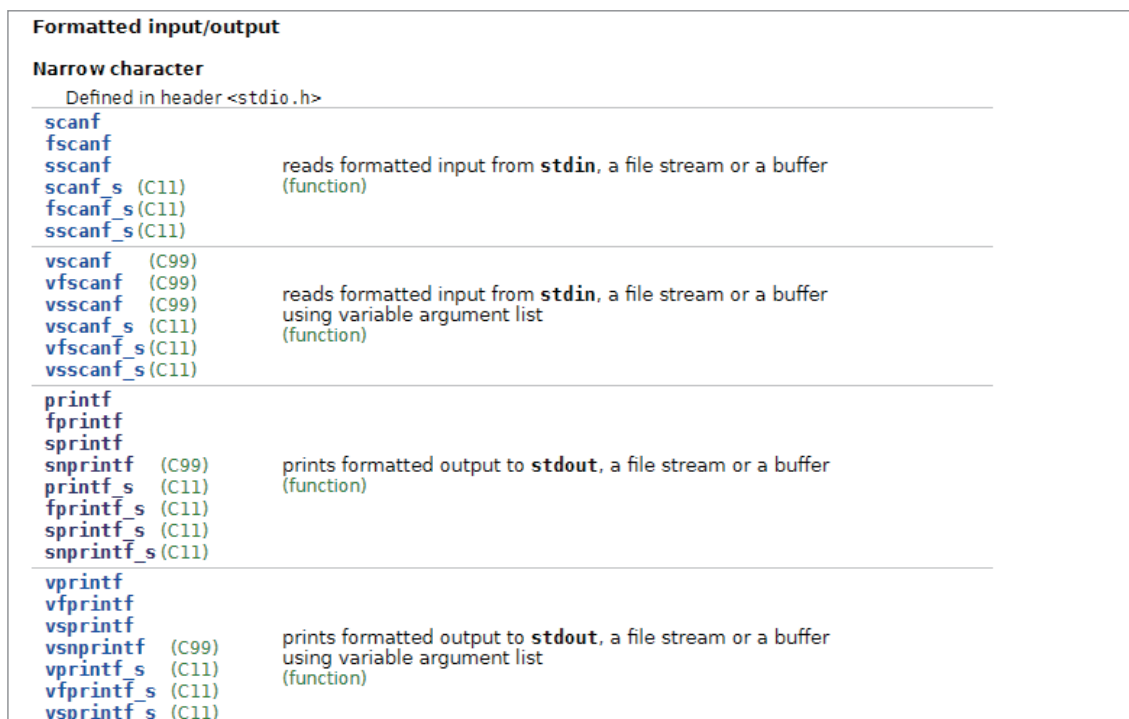


圖 5-3 C 語言網路使用手冊三

於圖 5-3 點選『printf』，畫面如圖 5-4：這裡有 printf() 的詳細用法：

C / File input/output

printf, fprintf, sprintf, snprintf, printf_s, fprintf_s, sprintf_s, snprintf_s

Defined in header <stdio.h>

int printf(const char *format, ...);

int printf(const char *restrict format, ...);

int fprintf(FILE *stream, const char *format, ...);

int fprintf(FILE *restrict stream, const char *restrict format, ...);

int sprintf(char *buffer, const char *format, ...);

int sprintf(char *restrict buffer, const char *restrict format, ...);

int snprintf(char *restrict buffer, size_t bufsz, const char *restrict format, ...);

int printf_s(const char *restrict format, ...);

int fprintf_s(FILE *restrict stream, const char *restrict format, ...);

int sprintf_s(char *restrict buffer, rsize_t bufsz, const char *restrict format, ...);

int snprintf_s(char *restrict buffer, rsize_t bufsz, const char *restrict format, ...);

(1)

(2)

(2)

(3)

(4)

(4)

(5)

(6)

(7)

(8)

(until C99)

(since C99)

(until C99)

(since C99)

(until C99)

(since C99)

(since C99)

(since C11)

(since C11)

(since C11)

(since C11)

Loads the data from the given locations, converts them to character string equivalents and writes the results to a variety of sinks.

1) Writes the results to the output stream stdout.

2) Writes the results to the output stream stream.

3) Writes the results to a character string buffer. The behavior is undefined if the string to be written (plus the terminating null character) exceeds the size of the array pointed to by buffer.

4) Writes the results to a character string buffer. At most bufsz - 1 characters are written. The resulting character string will be terminated with a null character, unless bufsz is zero. If bufsz is zero, nothing is written and buffer may be a null pointer, however the return value (number of bytes that would be written not including the null terminator) is still calculated and returned.

圖 5-4 C 語言網路使用手冊四

圖 5-4 就有如字典般的詳盡了，所有參數還有詳細解說，如圖 5-5：

Parameters

stream

buffer

bufsz

format

- output file stream to write to

- pointer to a character string to write to

- up to bufsz - 1 characters may be written, plus the null terminator

- pointer to a null-terminated multibyte string specifying how to interpret the data.

The format string consists of ordinary multibyte characters (except %), which are copied unchanged into the output stream, and conversion specifications. Each conversion specification has the following format:

• introductory % character

• (optional) one or more flags that modify the behavior of the conversion:

• -: the result of the conversion is left-justified within the field (by default it is right-justified)

• +: the sign of signed conversions is always prepended to the result of the conversion (by default the result is preceded by minus only when it is negative)

• space: if the result of a signed conversion does not start with a sign character, or is empty, space is prepended to the result. It is ignored if + flag is present.

• #: *alternative form* of the conversion is performed. See the table below for exact effects otherwise the behavior is undefined.

• 0: for integer and floating point number conversions, leading zeros are used to pad the field instead of *space* characters. For integer numbers it is ignored if the precision is explicitly specified. For other conversions using this flag results in undefined behavior. It is ignored if - flag is present.

圖 5-5 C 語言網路使用手冊五

☞ 輸入字元

輸入字元的敘述如下：

```
char a;  
scanf("%c", &a); //請留意是&
```

可將輸入的資料放入在某個變數所指定位址中，取址符號『&』用來取得欲放置輸入資料的變數位址。例如，以下程式可輸入一個字元。

```
#include <stdio.h>  
#include <stdlib.h>  
int main() {  
    char a;  
    printf("Please press any char: ");  
    scanf("%c", &a); //請留意是&  
    printf("Your char is %c", a);  
    return 0;  
}
```

請留意輸入資料要有取址符號『&』，漏掉了就是錯誤。

☞ 自我練習

1. 請鍵入以上程式，並觀察執行結果。
2. 請鍵入以下程式，並觀察執行結果。

```
#include <stdio.h>  
#include <stdlib.h>  
int main() {  
    char a,b;  
    printf("Please press any char: ");  
    scanf("%c", &a);  
    printf("Your char is %c", a);  
    printf("Please press any char: ");  
    scanf("%c", &b);  
    printf("Your char is %c", b);  
    return 0;  
}
```

以上程式，第二次輸入字元

```
scanf("%c", &b);
```

會因鍵盤緩衝問題（只要有先輸入東西，例如整數、實數都有此問題），而直接跳過，這時請將字元前放一個前導空白字元就可以。所以程式修正如下：

```
scanf(" %c", &b); //留意%c的前導空白
```

這在使用手冊有提到 <https://en.cppreference.com/w/c>，請讀者自行瀏覽。

♂ 輸入數值

常用的數值型態有 int、long 與 double 等，若要輸入這些型態的數值，則應使用不同的列印格式，例如，%d、%ld 與 %lf 等。請輸入以下程式，並觀察執行結果。

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int    a;
    long   b;
    double c;
    printf("Enter an integer:    ");
    scanf("%d", &a); //請留意是&
    printf("Enter a long integer: ");
    scanf("%ld", &b); //請留意是&
    printf("Enter a float:      ");
    scanf("%lf", &c); //請留意是&
    printf("\n Your enter..... \n");
    printf("integer      a =%d  \n",  a);
    printf("long integer b =%ld \n", b);
    printf("float        c =%f", c);
    return 0;
}
```

請留意輸入資料要有取址符號『&』，漏掉了就是錯誤。以下可輸入兩個數值，輸入資料時，資料中間以逗號分隔。

```
int a,b,d;
printf("input a,b:");
d=scanf("%d,%d",&a,&b); //請留意中間是逗號
printf("a+b=%d\n",a+b);
printf("d=%d\n",d);    //2    (傳回輸入資料正確的個數)
```

以上執行結果如下：(請自行補程式的頭尾)

```
input a,b:3,4
a+b=7
d=2
```

以下程式可輸入兩筆資料，輸入資料時，資料中間以空白分隔。

```
int a,b,d;
printf("input a b:");
d=scanf("%d %d",&a,&b); //請留意中間是空白
printf("a+b=%d\n",a+b);
printf("d=%d\n",d);    //2    (傳回輸入資料正確的個數)
```

以上程式執行結果如下：(請自行補程式的頭尾)

```
input a b:3 4
a+b=7
d=2
```

其次，scanf() 還傳回正確的資料個數，所以以上程式輸入 2 筆資料，d 值均傳回 2。若輸入資料型態不對，例如，故意輸入 a 3，由於輸入資料型態錯誤，將傳回『0』，或『EOF』，如下圖：

```
input a b:a 3
a+b=16
d=0
```

🔗 輸入字串

大部分程式語言都有字串 `string` 型態。但是 C 語言爲了加速處理速度，並沒有字串型態，而是使用字元陣列。因此在輸入字串時，我們必須利用字元陣列來完成輸入。例如，以下程式可輸入一個字串。

```
char c[20]; //字元陣列，請看第七章
printf("Please enter a string: ");
scanf("%s", &c); //
printf("Your enter is %s", c);
```

以上輸出結果如下圖：請自行鍵入程式，並觀察執行結果。

```
Please enter a string: Gwosheng
Your enter is Gwosheng
```

關於字串的使用，請看本書第 9-3 節。

👉 自我練習

1. 請鍵入以上程式，並觀察執行結果。
2. 請於 <https://en.cppreference.com/w> 查詢『`scanf()`』的詳細用法。
3. 請寫一個程式，可以輸入姓名、性別（m 代表男性，f 代表女性），且可以輸出姓名與性別。例如：輸入「焮維 m」，輸出「您的名字是焮維，性別：m」。
4. 請寫一個程式，可以輸入姓名、國文成績、數學成績，且可以輸出以上資料。例如「焮維您的國文成績爲 80, 數學成績爲 60, 總分爲 140, 平均爲 70」。

🔗 5-3 亂數

我們寫程式時，有時需要一個亂數，例如樂透開獎，撲克牌發牌等都需要產生一個亂數。所以電腦內部就以記憶體儲存一個亂數表，以供使用者取用。取用亂數的函式是：

```
rand();
```

它將傳回一個 0 到 32767 的任意整數。所以我們宣告一個整數變數暫存，如以下程式：

```
int a;
a=rand();
```

其次，此函式放在 `stdlib.h`，所以還要在標頭檔宣告如下：

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

還有，因為亂數表是固定的，那要如何取到不同亂數呢？答案是，可取系統時間當依據。因為每次程式執行時，系統的時間一定不一樣，就可取到不同的亂數。以下敘述，以當時的系統時間，當作亂數表的起點。

```
srand(time(0)); //以當時的系統時間，當作亂數表的起點
```

其次，`time()` 函式放在 `time.h`，所以也要載入以下標頭檔。

```
#include <time.h> //本例使用Time() 函式
```

範例5-3a

示範以上亂數的取得。

程式列印

```
#include <stdio.h>
#include <stdlib.h>
//步驟1，引入標頭檔
#include <time.h> //本例使用time() 函式
int main() {
    int a;
    //步驟2，以當時的系統時間，當作亂數表的起點
    srand(time(0));
    //步驟3，取亂數
    a=rand();
    printf("The Random Number is %d .\n", a);
    return 0;
}
```

以下敘述，可取 0~5 的亂數。「%」為取餘運算子，將於 6-1 節介紹。

```
int a=rand()%6;
```

以下敘述，可取 1~6 的亂數。

```
int a=1+rand()%6;
```

以下敘述，可取 97~122 的亂數。而 97~122 為小寫字元，所以可輸出小寫字元。

```
int a=97+rand()%26;
printf("%c",a);
```

👉 自我練習

1. 請將以上程式的『srand(time(0));』去掉，並觀察執行結果，是否每次都一樣。
2. 請練習於 <https://en.cppreference.com/w> 尋找『Pseudo-random number generation』，如圖 5-7：

C	Numerics	Pseudo-random number generation
Pseudo-random number generation		
Defined in header <stdlib.h>		
rand	generates a pseudo-random number (function)	
srand	seeds pseudo-random number generator (function)	
RAND_MAX	maximum possible value generated by <code>rand()</code> (macro constant)	

圖 5-7 random 說明

3. 請練習取 -5 到 8 的整數亂數，且輸出此數字。請參考 6-1 節取餘運算子『%』。
4. 請練習取 -1 到 1 的實數亂數，且輸出此數字。請參考 6-1 節取餘運算子『%』與除法『/』。
5. 請練習輸出任一大寫字元，且可由使用者也輸入此字元。

6. 若函式 `rand()` 的回傳值為一介於 0 和 10000 之間的亂數，下列那個運算式可產生介於 100 和 1000 之間的任意數 (包含 100 和 1000) ? (APCS10603)
- (A) `rand() % 900 + 100`
- (B) `rand() % 1000 + 1`
- (C) `rand() % 899 + 101`
- (D) `rand() % 901 + 100`

5-4 聲音的產生

要讓電腦發出聲音，可使用 `_beep()` 函式，此函式放在 `stdlib.h`，所以要載入標頭檔 `#include <stdlib.h>`，其語法如下：

```
void _beep(unsigned int x, unsigned int y);
```

`x` 是要發出的頻率，單位是 Hz，`y` 是發聲的時間，單位是 ms。其次，鋼琴鍵盤頻率對照表如表 5-2：

► 表 5-2 音階頻率表

	n	1	2	3	4	5	6	7	8	9	10	11	12
音階	音符	(Do)	(Do#)	(Re)	(Re#)	(Mi)	(Fa)	(Fa#)	(So)	(So#)	(La)	(La#)	(Si)
低音	頻率 (Hz)	262	277	294	311	330	349	370	392	415	440	466	494
中音	頻率 (Hz)	523	554	587	622	659	698	740	784	831	880	932	988
高音	頻率 (Hz)	1046	1109	1175	1245	1318	1397	1480	1568	1661	1760	1865	1976

例如，以下敘述可發出『Do』的音，且持續 0.5 秒。

```
_beep(523, 500);
```

範例5-4a

示範發出 Do、Re、Me、Fa…等八個音。

 程式列印

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    _beep(523,500);
    _beep(587,500);
    _beep(659,500);
    _beep(698,500);
    _beep(784,500);
    _beep(880,500);
    _beep(988,500);
    _beep(1046,500);
    return 0;
}
```

 演奏音樂

要讓電腦演奏音樂，那就要先下載一個簡譜，如圖 5-8。

| 1 1 1 3 | 5 5 5 5 | 6 6 6 [˙]1 | 5 - |
 我 有 一 隻 小 毛 驢 我 從 來 也 不 騎

圖 5-8 小毛驢簡譜

本例一分鐘 80 拍，所以 1 拍的時間是， $60000\text{ms}/80$ ，以一個四分音符為 1 拍，所以上圖一個八分音符所佔的時間是 $60000\text{ms}/(2*80)$ ，每小節是 2 拍。其次，上圖，所有音符的最小時間是八分音符，所以我就取八分音符的時間為 1 個單位。第 1 音『1 我』半拍，取 1 個單位；『5- 騎』是 2 拍，取 4 個單位。

範例 5-4b

示範演奏音樂。本範例僅作 4 小節，『我有一隻小毛驢我從來也不騎』。

 程式列印

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int t=60000/160;
    _beep(523,t);
    _beep(523,t);
    _beep(523,t);
    _beep(659,t);
    _beep(784,t);
    _beep(784,t);
    _beep(784,t);
    _beep(784,t);
    _beep(880,t);
    _beep(880,t);
    _beep(880,t);
    _beep(1064,t);
    _beep(784,4*t);
    return 0;
}
```

 補充說明

1. 請熟悉以上發音方式，第七章還會介紹如何製作電子琴。
2. 以上僅輸出一小節，程式就有點冗長，那寫完一首歌不就累垮。請放心，待介紹迴圈與陣列就會明瞭，本書於範例 9-5g，將會以陣列與迴圈重做本範例。

 自我練習

1. 請繼續完成以上音樂。
2. 請用輸出指令讓輸出音樂的同時，螢幕也輸出歌詞，那就是 KTV 了。
3. 請自己找一首歌，完成輸出歌詞與音階。

🔊 警車聲音

警車的警報聲是連續發出頻率 265Hz 與 350Hz 所形成，以下範例示範發出此聲音。

範例5-4c

示範發出警車警報聲。

👉 程式列印

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    while(1){//無窮迴圈，請看第六章
        _beep(265,300);
        _beep(350,700);
        _sleep(50);//空白
    }
    return 0;
}
```

👉 自我練習

1. 市內電話來電大都採用 1000Hz 與 500Hz 交錯形成的樂音，請寫程式完成。
2. 同上題，練習讓鈴聲可越來越急。(此題可待學完迴圈再作)
3. 音感練習。請觀察您家微波爐、洗衣機、汽車安全帶未繫、倒車警告、手機的各種聲響，並寫程式完成。(提示:若沒音感，可上網搜尋)

運算子及運算式

Chapter

06

學習綱要

- 📖 6-1 運算子
- 📖 6-2 運算式
- 📖 6-3 運算式與運算子應用實例
- 📖 6-4 專案除錯

學習目標

- 📖 1. 能寫出 C 語言運算子與其功能。
- 📖 2. 能以運算子完成運算式。
- 📖 3. 能以運算式完成一個程式敘述。
- 📖 4. 能以數個敘述完成一個程式。
- 📖 5. 能自己除錯。
- 📖 6. 能實作運算子與基本運算式實例。

前面我們已經使用 $c = a * b$ 計算長方形面積，「a」、「b」稱為運算元 (Operand)；「*」稱為運算子 (Operator)；「 $a * b$ 」稱為運算式 (Expression)，運算式是一種結合序列的運算子、運算元的敘述，用來計算產生結果；「 $c = a * b$ 」稱為敘述 (Statement)。敘述是程式組成的基本元素，是程式的主體；每個敘述執行有意義的工作。程式是由許多不同意義的敘述組成的有目的的工作程序。以上內容將在本章介紹。

6-1 運算子

所謂運算子 (Operator)，指的是可以對運算元 (Operand) 執行特定功能的特殊符號。C 語言的運算子分為五大類，分別是算術 (Arithmetic) 運算子、複合指派運算子、關係運算子、邏輯運算子、位元運算子。

⌘ 算術運算子 (Arithmetic operators)

算術運算子用來執行一般的數學運算，包括指派 (=)、取正負數 (+/-)、加 (+)、減 (-)、乘 (*)、除 (/)、取餘數 (%)、遞增 (++) 及遞減 (--) 等，表 6-1 是 C 語言的算術運算子列表：

► 表 6-1 C 語言算術運算子

運算子	定義	優先順序	結合律
=	指派或賦值	14	由右至左
a++,a--	後置遞增，遞減	1	由左至右
++a,--a	前置遞增，遞減	2	由右至左
+, -	正負號，一元運算子	2	由右至左
*	乘法運算	3	由左至右
/	除法運算	3	由左至右
%	求餘數 (Modulus)	3	由左至右
+, -	加法，減法運算	4	由左至右

以上運算子的定義、優先順序、結合律分別說明如下：

► 指派運算子 (Assignment operators)

指派運算子的符號為 (=)，又稱為賦值運算子，其作用為將運算符號右邊運算式的值指派給運算符號左邊的運算元。所以，以下敘述 `sum=a+b` 是將 `a+b` 的值先運算，再指派給 `sum`。

```
int sum = 0, a = 3, b = 5;
sum = a + b;
```

上式與數學的等號是不同的，所以不要一直糾結為什麼 0 會等於 8。其次，你是不能將常數放在指派運算子的左邊，例如：

```
8 = x ;
```

此為一個不合法的敘述，但以下敘述，將常數 8 指派給變數 `x` 是合法的。

```
x = 8 ;
```

► 五則運算

以下是一些簡單算術五則運算，請自行鍵入程式，寫出執行結果。

```
int a=5,b=4;
printf("%d\n",a+b); //9
printf("%d\n",a-b); //1
printf("%d\n",a*b); //20
printf("%d\n",a/b); //1
printf("%d\n",a%b); //1
```

► 整數除法或實數除法

C 語言的除法運算，只有被除數與除數的型態均為整數，才是整數除法，商的型態為整數。其次，只要被除數與除數有一型態為實數即為實數除法，得到實數商。例如：

```
int x=5, y=4;
double xf=5,yf=4;
printf("%d\n",x/y); //1 兩者都是 int
printf("%f\n",xf/y); //1.25
printf("%f\n",x/yf); //1.25
printf("%f\n",xf/yf); //1.25
```

自我練習

題號	題目	輸出結果
1	<pre>int main() { int a,b=7,c=2; double d; d=b/c; printf("%lf\n",d); return 0; }</pre>	
2	<pre>int main() { int a,b=7,c=2; double d=2,e; e=b/d; printf("%lf\n",e); a=b/d; printf("%d\n",a); return 0; }</pre>	

► 整數除法與取餘數的應用

整數除法與取餘數可以將一個整數分解為數個數字。例如，若要使用七段顯示器顯示 152，就要將此數字分解為 1,5,2 三個數字，其方法如下：

```
int a=152;
int a1=152/100;//1百位數
int a2=(152-a1*100)/10;//5十位數
int a3=a %10;//個位數
```

以上是已知數字長度，可以從左邊分解。以下是從末端一直先取餘數，再相除。

```
int a=152;
int a3=152%10;//個位數
a=a/10;
int a2=a%10;//十位數
a=a/10;
int a1=a ;//百位數
printf("%d,%d,%d",a1,a2,a3);
```


從末端分解的優點是，可以由商是否為零，判斷是否結束。等到學了決策與迴圈指令就能寫程式完成任意長度整數的分解。

► 取餘數

若要讓一個數字在累加的過程中，保持一定的循環，那也是用取餘數。例如：

```
while(1)    { //無窮迴圈，請看第六章
    b=a[i];
    i=(i+1) %4;
}
```

那 i 就永遠在 0,1,2,3 循環。

自我練習

1. 請輸入一個四位數，並將其分解輸出為四個數字，且求其數字和。例如，輸入 1234，那可以輸出 1,2,3,4,10。(請留意本單元自我練習都是固定長度，所以不用迴圈與決策指令)
2. 請輸入一個四位數，並將其反相輸出此四個數字，且求其數字和。例如，輸入 1234，那可以輸出 4321,10。
3. 若有一個秒數是 12345，如何分解為幾時幾分幾秒？例如，本例要能得到 3 時 25 分 45 秒。
4. 若有一個 5 位數，請問如何從最右邊分解為每兩個一組數字？例如， $a=25678$ ，那如何分別得到 25,56,67,78 等 4 個數字？其次，要如何得到 2,56,78 三組數字呢？
5. 若有一個 5 位數，如何得到最左邊與最右邊數字的和？例如，輸入 12345，結果是 $1+5=6$ 。
6. 若有一個 5 位數，如何得到最左邊兩位數與最右邊兩位數字的和？例如，輸入 12345，結果是 $12+45=57$ 。
7. 若有一個 6 位數，如何得到最左邊三位數與最右邊三位數字的和？例如，輸入 123456，結果是 $123+456=579$ 。

(補充說明：以上方法不只一種，方法沒有優劣，只要自己想出來的都是最好的。其次，以上即為 APCS 考試基礎題)

► 遞增(++)及遞減(--)

遞增 (++) 及遞減 (--) 是 C 語言特有的運算子，此二個運算子又分為前置與後置，前置是運算子在運算元之前，例如：

```
a=1;++a; printf("%d\n",a); //2
```

後置是運算子在運算元之後，例如：

```
a=1;a++; printf("%d\n",a); //2
```

原則上不論 ++a 或 a++ 都是將 a 值加 1 並放回 a。但若是：

```
a=1;b=++a; // a=a+1;b=a; a is 2 b is 2
```

和

```
a=1;b=a++ ; // b=a;a=a+1;a is 2 b as 1
```

則其 a 值均會加 1，但 b 值會有差異，前置 b 值會得到加 1 的結果，後者只能得原 a 值。

► 次方與根號

值得注意的是，C 語言的數學運算子只有以上 +,-,*,/,%。若所需運算沒有運算子，例如，要次方或根號運算，那要如何處理呢？答案是要使用數值處理函式。例如，以下 pow() 函式可計算 3 的 2 次方：

```
a = pow(3,2); //include<math.h>
```

若要執行根號運算，則應使用數值處理函式的 sqrt() 函式。如以下敘述是計算 9 的開平方。

```
x= sqrt(9); //include<math.h>
```

但以上函式都放在 math.h，所以都要在程式最前面引入 math.h 標頭檔使用，如以下敘述：

```
#include<math.h>
```

載入標頭檔 `math.h`。請留意以上敘述載入標頭檔，不用分號『;』結束。
關於函式的用法，請看第十一章。

自我練習

請將以上運算式於 C 環境編譯、執行，觀察執行結果。

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h> //本例使用數學函式 sqrt()
int main() {
    printf("%lf\n",pow(3,2));
    printf("%lf",sqrt(9));
    return 0;
}
```

複合指派運算子（Compound assignment operators）

結合指派與算術的運算子稱為複合指派運算子，此乃 C 語言特有的運算子。例如：程式設計者常會鍵入 `sum=sum+5`，程式語言乃制定此一複合指派運算子「+=」。所以上述 `sum=sum+5`，即可寫成 `sum += 5`。表 6-2 是 C 語言常用的複合指派運算子：

► 表 6-2 複合指派運算子

運算子	定義	優先順序	結合律
=	指派內容	14	由右至左
+=	相加之後再指派內容	14	由右至左
-=	相減之後再指派內容	14	由右至左
*=	相乘之後再指派內容	14	由右至左
/=	相除之後再指派內容	14	由右至左
%=	取餘之後再指派內容	14	由右至左

例如：

```
int sum=0,a=5;
sum+=5;
printf("%d\n",sum); //5
sum/=a;
printf("%d\n",sum); //1
```

♂ 比較(Comparison)運算子

比較運算子又稱為關係運算子 (Relational Operators)，用於資料之間的大小比較，比較的結果可得到 1(true) 或 0(false)，表 6-3 是 C 語言中的比較運算子符號。

► 表 6-3 比較運算子

運算子	定義	優先順序	結合律
<	小於	6	由左至右
>	大於	6	由左至右
<=	小於等於	6	由左至右
>=	大於等於	6	由左至右
==	等於	7	由左至右
!=	不等於	7	由左至右

例如：

```
int a=8,b=5;
printf("%d\n",a>b); //1
printf("%d\n",a==b); //0
printf("%d\n",a>=b); //1
printf("%d\n",a<=b); //0
printf("%d\n",a!=b); //1
```

👉 自我練習

1. 請將以上運算式於 C 環境編譯、執行，觀察執行結果。
2. 請鍵入以下程式，編譯與修正程式。

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int a=8,b=5;
    printf("%d\n",a==b);
    printf("%d\n",a=b);
    printf("%d\n",a>b);
    printf("%d\n",a<b);
    printf("%d\n",a!=b);
    return 0;
}
```

🔗 邏輯運算子(Logical Operators)

當同一個運算式要同時存在兩個以上的關係運算子時，每兩個關係運算子之間必須使用邏輯運算子連結。例如：您要找男生且年齡大於 40，此一選擇就同時含有兩個關係運算式，此時就要使用邏輯運算子給予連結。C 的邏輯運算子有否定 not『!』、邏輯 and『&&』、邏輯 or『||』，如表 6-4 所示：

► 表 6-4 邏輯運算子

運算子	定義	優先順序	結合律
!	邏輯否定 not 運算	2	由右至左
&&	邏輯 and 運算	11	由左至右
	邏輯 or 運算	12	由左至右

► !

『!』是取邏輯否定，也就是將『true』轉為『false』『false』轉為『true』。例如：

```
int a=8,b=5,c=3;
printf("%d\n", (a>b)); //1
printf("%d\n", !(a>b)); //0
printf("%d\n", !(a==b)); //1
```

► &&

『&&』是邏輯 and，兩件事都 true，結果才是 true，其餘都是 false，其真值表如表 6-5：

► 表 6-5 and 真值表

事件1	事件2	結果
true	true	true
true	false	false
false	true	false
false	false	false

例如：

```
int a=8,b=5,c=3;
printf("%d\n",a>b && b>c );//1
printf("%d\n",a>b && b==c );//0
```

► ||

『||』是邏輯 or，也就是兩件事只要有一件為 true，那就為 true；只有兩件事全為 false，才為 false。其真值表如表 6-6：

► 表 6-6 or 真值表

事件1	事件2	結果
true	true	true
true	false	true
false	true	true
false	false	false

例如：

```
int a=8,b=5,c=3;
printf("%d\n",a>b || b==c );//1
printf("%d\n",a==b || b==c );//0
```

範例 6-1a

若有一數學式，判斷 x 是否滿足 $1 < x \leq 6$ ，請轉換為 C 語言敘述。

解答

判斷 x 是否在 1 到 6 之間，數學語言是

$$1 < x \leq 6$$

C 語言是

```
int x=3;
printf("%d",x>1 && x<=6);
```

範例6-1b

輸入三角形三邊長 a, b, c ，圍成三角形的條件是，任兩邊之和要大於第三邊，如何以 C 語言表示？

 提示

任兩邊之和大於第三邊的數學語言是：

```
a+b > c and a+c > b and b+c > a
```

C 語言是

```
printf("%d", (a+b)>c && (b+c)>a && (c+a)>b);
```

完整程式如下：

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int a=3,b=4,c=5;
    printf("%d", (a+b)>c && (b+c)>a && (c+a)>b);
    return 0;
}
```

 自我練習

- 若要邏輯判斷式 $!(X1 \parallel X2)$ 計算結果為真 (True)，則 $X1$ 與 $X2$ 的值分別應為何？(APCS)
 - $X1$ 為 False， $X2$ 為 False
 - $X1$ 為 True， $X2$ 為 True
 - $X1$ 為 True， $X2$ 為 False
 - $X1$ 為 False， $X2$ 為 True
- 請產生 1 個 $-10 \sim 10$ 的亂數，判斷 x 是否滿足 $x>3$ or $x<-2$ ，滿足的輸出 1，不滿足的輸出 0，請轉換為 C 語言敘述。
- 若有一數學式，已知六個數字，同時判斷六個數字是否滿足 $\frac{a_1}{a_2} = \frac{b_1}{b_2} = \frac{c_1}{c_2}$ ，請轉換為 C 敘述。

(提示：測試資料 1: int a1=1,b1=2,c1=3,a2=3,b2=6,c3=9; 爲 1；測試資料 2: int a1=1,b1=2,c1=3,a2=3,b2=6,c3=1; 爲 0)

4. 若有一數學式，已知六個數字，同時判斷六個數字是否滿足 $\frac{a_1}{a_2} = \frac{b_1}{b_2} \neq \frac{c_1}{c_2}$ ，請轉換爲 C 敘述？

(提示：測試資料 1: int a1=1,b1=2,c1=3,a2=3,b2=6,c3=9; 爲 0；測試資料 2: int a1=1,b1=2,c1=3,a2=3,b2=6,c3=1; 爲 1)

⚡ 位元運算子(Bitwise operation)

在介紹位元運算子前，請您思考一下複選題答案應該如何數位化？例如，答案有可能 ABC，也有可能 BE，那如何將這些資料數位化，選用適當的變數型態處理呢？是不是用一個字串表示呢？例如：以

```
a="ABD"
```

表示。這樣表示當然也可以，但每個答案就要占 1 ~ 5 byte(1 byte=8bit)，但 "ABD" 若是以 11010 表示呢？"A" 就以 10000 表示，這樣就可以分別用 26(11010=2⁴+2³+2¹=26) 與 16(10000=2⁴=16) 表示 "ABD" 與 "A"，所以只要 1byte 就可以記錄所有答案。好的，以上已經完成非常精簡的資料數位化方式，那電腦如何評分呢？答案就是電腦也備有位元運算子，可以錙銖必較的處理每個位元 (bit)。

位元 (Bitwise) 運算子是將一個 8 位元的整數逐位元進行運算，此類運算子還可以分爲兩類：位移 (Shift) 運算子與布林運算子。位移運算子可以用來將各個位元向左或是向右移；布林運算子則可以逐位元進行布林運算。表 6-7 是 C 語言位元操作運算子的列表：

► 表 6-7 位元運算子

運算子	定義	結合律
~	逐位元 not 運算	由右至左
<<	逐位元向左位移	由左至右
>>	逐位元向右位移	由左至右
&	逐位元 and 運算	由左至右
^	逐位元 xor 運算	由左至右
	逐位元 or 運算	由左至右

► ~

『~』是位元 not 運算，not 是將 0 變 1，1 變 0。例如：

```
a=1;
printf("%d",~a); //-2
```

為什麼答案是 -2 呢？這有點難懂，筆者簡單說明如下，因為電腦是以 2 補數表示負數。a=1，分解為 2 進位是 00000001

```
~a= not 00000001=11111110
```

首位元是 1 表示負數，那到底負多少，就取 2 補數。2 補數的步驟是先取 1 補數再加 1。其次，所謂 1 補數是『逐位元將 1 變 0，0 變 1』，所以上面 11111110 的 1 補數是

```
00000001
```

將 1 補數再加 1，就是 2 補數，上面 00000001 加 1 就是 00000010，其大小是 2。上面首位元符號既然是『1』，那就表示其為負數，所以 11111110 代表此數是 -2，所以輸出 -2，此即為二補數的觀念。

👉 補充說明

1. 正數的數位化。正數就直接以 2 進位編碼，本例以 8 位元為例，如下表 6-8：

► 表 6-8 正數數位化

正數	二進位編碼
0	00000000
1	00000001
2	00000010
3	00000011
4	00000100
5	00000101
6	00000110
7	00000111
254	11111110
255	11111111

2. 正負數的數位化：正負數的數位化，直覺的作法，就將最高位元放 0 代表正數，放 1 代表負數，以上直覺的編法如表 6-9：

► 表 6-9 正負數十進位轉二進位直覺編法對照表

二進位編碼	數字
0000000	0
0000001	1
0111111	127
1000000	-0
1000001	-1
1000010	-2
1111110	-126
1111111	-127

由上表發現，『0000000』與『1000000』都對應到 0，這樣不是一對一函數，求反函數比較複雜，所以科學家就不斷討論與修正，發現負數的轉換需要兩個步驟，第一，求其 1 補數（將 0 變 1，1 變 0），第二，求 2 補數（1 補數加 1 稱為 2 補數。）

3. 將負數部分取 2 補數的過程如表 6-10：

► 表 6-10 負數編碼

數字	取正數	1補數	2補數
-0	0000000	1111111	0000000
-1	0000001	1111110	1111111
-2	0000010	1111101	1111110
-3	0000011	1111100	1111101
-126	0111110	1000001	1000010
-127	0111111	1000000	1000001
-128	1000000	0111111	1000000

4. 將上表合併正數，如表 6-11：（請留意正數直接編碼，負數取 2 補數編碼）

► 表 6-11 正負數編碼

數字	取正數	1補數	編碼
127			01111111
126			01111110
2			00000010
1			00000001
0			00000000
-0	00000000	11111111	00000000
-1	00000001	11111110	11111111
-2	00000010	11111101	11111110
-3	00000011	11111100	11111101
-126	01111110	10000001	10000010
-127	01111111	10000000	10000001
-128	10000000	01111111	10000000

5. 由上表可知，若負數先取 2 補數在編碼，竟然就解決正負 0 的問題，使的轉換過程為 1 對 1。而且這種轉換竟然一石二鳥，也解決減法問題，因為

5-2

可以看成

5+(-2)

也就是減法，也可用加法解決。

6. 由上表可知，剛剛

```
a=1; //a=00000001
printf("%d",~a); //~a=11111110 11111110 is -2
```

為什麼是『-2』了。

自我練習

1. 請輸入以下程式，並觀察執行結果，且自行使用以上原理推敲，看答案是否一致。

```
a=-3;
printf("%d",~a);
a=-127;
printf("%d",~a);
```

► <<

『<<』是左移運算子，例如：

```
a=1;
printf("%d",a<<1); //2    1(00000001) 向左移位1位元(00000010)，結果是2
a=1;
printf("%d",a<<2); //4    1(00000001) 向左移位2位元(00000100)，結果是4
```

► >>

『>>』是右移運算子，例如：

```
a=4;
a=a>>1;
printf("%d",a); //2    4向右移位1位元，結果是2
a=7;
a=a>>2;
printf("%d",a); //1
```

► &

『&』是位元 and 運算，and 運算真值表如表 6-12，當 a 與 b 同時為 1，c 才得到 1：

► 表 6-12 位元 and 真值表

a	b	c= a & b
0	0	0
0	1	0
1	0	0
1	1	1

例如：

```
a=1;b=5;
printf("%d",a&b); //00000001 & 00000101=00000001=1
```

► ^

『^』是位元 xor 運算，xor 運算真值表如表 6-13，當 a 與 b 不同時，c 才得到 1。

► 表 6-13 位元 xor 真值表

a	b	c=a^b
0	0	0
0	1	1
1	0	1
1	1	0

例如：

```
a=1;b=5 ;
printf("%d",a^b); //00000001 ^00000101=00000100=4
```

► |

『|』是位元 or 運算，or 運算真值表如表 6-14，當 a 與 b 有一為 1，c 就得到 1。

► 表 6-14 位元 or 真值表

a	b	c=a b
0	0	0
0	1	1
1	0	1
1	1	1

例如：

```
a=1;b=5;
printf("%d",a|b); //00000001 | 00000101 =00000101=5
```

 自我練習

1. 寫出以下程式執行結果。(統測 111)

```
int a = 9, b = 7;
printf( "%d", a^b );
```

 運算子的優先順序(Precedence)

同一敘述，若同時含有多個運算子，此時即需定義運算子的優先順序。例如：

```
x=a+b*c;
```

在數學裡，我們定義先乘除再加減。程式語言也是相同，必須定義這些運算子的優先順序，C 語言的運算子優先順序如表 6-15：

► 表 6-15 運算子優先順序表

優先順序	運算子	說明	結合律
1	++,--	後置遞增與遞減。例如 a++,a--。	由左至右
2	++,-- +,- !	前置遞增與遞減。例如 ++a,--a。 取正或取負。 邏輯否定。	由右至左
3	*,/,%	算術乘、除、取餘。	由左至右
4	+, -	算術加、減	
6	>, >=, <, <=	關係運算的大小。	
7	==, !=	關係運算的相等或不相等。	
11	&&	邏輯的 and。	
12		邏輯的 or。	
14	=, +=, -=, *=, /=, %=	指派與複合指派。	由右至左

由以上各運算子的『優先順序』可知，乘號『*』的優先順序是第 3，而加號『+』則是第 4。所以，以上敘述：

```
x=a+b*c;
```

的結果與

```
x=(a+(b*c));
```

的效果相同。同理，

```
x=a>b && b>z;
```

的結果同義於

```
x=(a>b) && (b>z);
```

自我練習

1. 鍵入、執行以下程式，並寫出執行結果。

程式	執行結果
<pre>#include <stdio.h> #include <stdlib.h> int main() { int a,b,c; a=1;b=2;b=3; a=a+a*2/2-3%2; printf("%d\n",a); a=5;b=4; c= a>2 && !(b>3) (a==3); printf("%d\n",c); return 0; }</pre>	

2. 依 C 語言之運算子優先權順序，下列運算式的結果，何者為真（true）？（統測 111）
 - (A) $!(1!=3) \parallel 1==3$
 - (B) $1!=3 \&\&!!(1==3)$
 - (C) $!(1<3) \parallel 1>=3$
 - (D) $1<3 \&\&!(1>=3)$

✎ 運算子的結合律(Associativity)

當同一敘述，相鄰的運算子擁有相同優先順序的運算子，此時即需定義運算子是『由左至右』的左結合或『由右至左』的右結合。例如：

```
x=a-b-c;
```

同樣是減號(-)，優先順序相同，此時就要靠定義結合律，減法結合律是由左至右，所以以上同義於

```
x=((a-b)-c);
```

而

```
a=b=c=2;
```

指派運算子的結合律是由右至左，所以以上式子同義於：

```
(a=(b=(c=2))) ;
```

以上式子，a、b、c的結果都是2。

👉 自我練習

1. 鍵入、執行以下程式，並寫出執行結果。

程式	執行結果
<pre>#include <stdio.h> #include <stdlib.h> int main() { int a=3,b=2,c=1; c=a-b-c; printf("%d\n",c); a=b=c=2; printf("a=%d,b=%d,c=%d\n",a,b,c); return 0; }</pre>	

2. 若 a, b, c, d, e 均為整數變數，下列哪個算式計算結果與 $a+b*c-e$ 計算結果相同？（APCS 試題）

(A) $((a+b)*c)-e$

(B) $((a+b)*(c-e))$

(C) $((a+(b*c))-e)$

(D) $(a+((b*c)-e))$

關於以上運算子的詳細說明可點選 <https://en.cppreference.com/w/c> 裡面的『Expressions』，如圖 6-1：

Expressions

An expression is a sequence of *operators* and their *operands*, that specifies a computation. Expression evaluation may produce a result (e.g., evaluation of `2+2` produces the result `4`), may generate side-effects (e.g. evaluation of `printf("%d", 4)` sends the character `'4'` to the standard output stream), and may designate objects or functions.

General

- [value categories](#) (lvalue, non-lvalue object, function designator) classify expressions by their values
- [order of evaluation](#) of arguments and subexpressions specifies the order in which intermediate results are obtained

Operators

Common operators						
assignment	increment decrement	arithmetic	logical	comparison	member access	other
<code>a = b</code> <code>a += b</code> <code>a -= b</code> <code>a *= b</code> <code>a /= b</code> <code>a %= b</code> <code>a &= b</code> <code>a = b</code> <code>a ^= b</code> <code>a <<= b</code> <code>a >>= b</code>	<code>++a</code> <code>--a</code> <code>a++</code> <code>a--</code>	<code>+a</code> <code>-a</code> <code>a + b</code> <code>a - b</code> <code>a * b</code> <code>a / b</code> <code>a % b</code> <code>~a</code> <code>a & b</code> <code>a b</code> <code>a ^ b</code> <code>a << b</code> <code>a >> b</code>	<code>!a</code> <code>a && b</code> <code>a b</code>	<code>a == b</code> <code>a != b</code> <code>a < b</code> <code>a > b</code> <code>a <= b</code> <code>a >= b</code>	<code>a[b]</code> <code>*a</code> <code>&a</code> <code>a->b</code> <code>a.b</code>	<code>a(...)</code> <code>a, b</code> <code>(type) a</code> <code>? :</code> <code>sizeof</code> <code>_Alignof</code> (since C11)

- [operator precedence](#) defines the order in which operators are bound to their arguments

圖 6-1 C 語言線上手冊一

繼續點選上圖的『[operator precedence](#)』，可得運算子的優先順序，如圖

6-2：

C Operator Precedence			
The following table lists the precedence and associativity of C operators. Operators are listed top to bottom, in descending precedence.			
Precedence	Operator	Description	Associativity
1	++ --	Suffix/postfix increment and decrement	Left-to-right
	()	Function call	
	[]	Array subscripting	
	.	Structure and union member access	
	->	Structure and union member access through pointer	
	(type){list}	Compound literal(C99)	
2	++ --	Prefix increment and decrement	Right-to-left
	+ -	Unary plus and minus	
	! ~	Logical NOT and bitwise NOT	
	(type)	Type cast	
	*	Indirection (dereference)	
	&	Address-of	
	sizeof	Size-of ^[note 1]	
	_Alignof	Alignment requirement(C11)	
3	* / %	Multiplication, division, and remainder	Left-to-right
4	+ -	Addition and subtraction	
5	<< >>	Bitwise left shift and right shift	
6	< <=	For relational operators < and ≤ respectively	
	> >=	For relational operators > and ≥ respectively	
7	== !=	For relational = and ≠ respectively	

圖 6-2 運算子優先順序

6-2 運算式

任何一個可求得一個值的式子，都稱為一個運算式，例如 `:5+3` 會傳回一個數值，所以 `5+3` 是一個運算式。以下是一些合法的運算式，這些運算式通常放在 `printf(“%d”, 運算式)`、第七章的 `if(運算式)`、第八章的 `for(運算式;)` 及 `while(運算式)`。一般而言，可以放在等號右邊的東西，都可以稱為運算式。

```
6           // 傳回6
4 + 5 ;     // 傳回9
sum > 3 ;   // 傳回 true or false
a==b;       // 傳回true or false
a + b ;     // 傳回8
```

敘述 (Statement)

凡是控制執行的順序、對運算式取值或不作任何事，均可稱為敘述。所有的敘述都要以分號(;)作結束，例如：以下式子即是一個敘述。

```
sum = sum + 1 ;
```

於 C 語言中，若前一個敘述未以分號結束，則錯誤訊息通常出現在下一個敘述的開頭。

敘述區塊 (Block Statement) 或複合敘述 (Compound Statement)

敘述區塊亦稱複合敘述，在任何可以放置單一敘述的地方，你都能放置敘述區塊。一個複合敘述是由兩個大括號組合而成，如下所示：

```
{  
    t = a ;  
    a = b ;  
    b = t ;  
}
```

請留意，大括號之後不可再加分號，且兩個括號對稱，方向都是朝程式區塊。

註解 (Comments)

適當的程式註解才能增加程式的可讀性，C/C++ 語言的註解方式有兩種，分別是多列註解與單列註解。首先，多列註解如下：

```
/* 我是註解 */
```

上式「/*」符號以後的字串視為註解，編譯程式不予處理，直到遇到「*/」為止。其次，單列註解如下：

```
//我是註解
```

同一列中，雙斜線（//）後面的文字均視為註解，編譯器均不予處理。例如，以下二式分號右邊均為註解：

```
x= x+ 3;          /* x值加3 */
sum = sum + y;    // 將y之值加至sum
```

前者，因為可超過兩列，較適合編寫較長的註解；後者，則僅能寫在同一列。

♂ 縮排與空行

除了適當的註解，程式設計應善用縮排與空行，才能使你的程式讓人易於閱讀。因為有了縮排（請使用 Tab 鍵，而不是自行按空白鍵，因為空白鍵很費力，且無法完全對齊），程式才有層次感。其次，空行則可使段落更加明顯。所以，本書範例則都遵守這些規定，請讀者自行觀察。

🔗 6-3 運算子與運算式應用實例

範例6-3a

請寫一程式，滿足以下條件。

1. 可以輸入兩個座標。
2. 計算此兩點座標距離。
3. 輸出此兩點距離。

👉 程式設計步驟

1. 寫出演算法

- (1) 已知兩點座標分別是 (x1,y1),(x2,y2)，則其距離的公式的數學語言是：

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

- (2) 以上公式的 C 語言運算式如下：

```
d=sqrt(pow((x1-x2),2)+pow((y1-y2),2));
```

2. 資料數位化。本例 x_1, x_2 取 `int`， d 取 `double`，變數先指派，程式如下：

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h> //本例使用sqrt,pow數學函式
int main() {
    int x1,y1,x2,y2;
    double d;
    //input
    x1=3;y1=0;
    x2=0;y2=4;
    //process
    d=sqrt(pow((x1-x2),2)+pow((y1-y2),2));
    //output
    printf("The distance of the two points is:%f\n",d);
    return 0;
}
```

3. 輸入資料

- (1) 本例使用 `scanf("%d,%d",&x1,&y1)` 連續輸入兩個數字，數入數字時要加上逗號『`,`』隔開。
- (2) 若是使用 `scanf("%d %d",&x1,&y1)` 連續輸入兩個數字，數入數字時要加上空白『』隔開。
- (3) 若是使用 `scanf("%d %d",x1,y1)` 連續輸入兩個數字，漏掉『`&`』，程式會出現錯誤。

程式列印

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h> //本例使用sqrt,pow數學函式
int main() {
    int x1,y1,x2,y2;
    double d;
    //input
    printf("input x1 y1:");
    scanf("%d,%d",&x1,&y1); //留意&
    printf("input x2 y2:");
    scanf("%d,%d",&x2,&y2);
```

```
//process
d=sqrt(pow((x1-x2),2)+pow((y1-y2),2));
//output
printf("The distance of the two points is:%f\n",d);
return 0;
}
```

👉 補充說明

1. C 語言數學運算子只有 +, -, *, /, %(取餘數)，其餘數學運算都要使用數學函式。關於數學函式可先翻閱本書 11-1 節，本例 pow() 是次方，sqrt() 是開根號。

👉 自我練習

1. 輸入三角形三邊長 a、b、c，求其面積。
提示：已知三邊長，求面積的演算法如下：先計算 $d=(a+b+c)/2$ ，則三角形面積 $=\sqrt{d(d-a)(d-b)(d-c)}$ ，本例假設所輸入的三角形三邊長可圍成三角形，例如輸入 3，4，5 則得三角形面積 6，但並不是任意三條線都可圍成三角形，若要判斷是否可圍成三角形，請繼續研讀下一章。
2. 請寫一個程式，可以將公斤數轉為台斤數。
3. 請寫一個程式，可以將攝氏溫度轉為華氏溫度。
4. 請寫一個程式，可以輸入三點座標，輸出其三角形面積。提示：三角形面積公式如下：

$$\frac{1}{2} \begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ x_1 & y_1 \end{vmatrix} = \frac{1}{2} |(x_1y_2+x_2y_3+x_3y_1-x_2y_1-x_3y_2-x_1y_3)|$$

例如，指派輸入 (0,0)、(3,0)、(0,4)，結果是 6。

5. 請寫一個程式，將時速 km/h 轉為秒速 m/s。
6. 請寫一個程式，將秒速 m/s 轉為時速 km/h。

範例6-3b

資料的抽象化範例。請寫一程式，可以先指派一直線方程式的係數 a, b, c ，並可輸入一點 $p(x, y)$ ，並求出此點與直線的距離。例如，指派 a, b, c 為 3, 4, 5，那直線就是 $3x + 4y + 5 = 0$ ，然後輸入一個點 $p(1, 2)$ ，那其距離為

$$d = \frac{|3x + 4y + 5|}{\sqrt{a^2 + b^2}} = 3.2 \text{。 (提示：實數絕對值函式是 fabs())}$$

解題步驟

1. 直線的數位化。國中數學老師寫

```
ax+by+c=0
```

我們理解此為一條直線方程式，這件事如何用程式語言表示呢？那就是指派 a, b, c 就好。

```
int a=3,b=4,c=5;
```

2. 點與直線的距離。任一 p 點 (x, y) 與直線的距離，就直接代入

$$d = \frac{|3x + 4y + 5|}{\sqrt{a^2 + b^2}}$$

這就是一種資料的抽象，僅抽取一個意象，計算我們要的東西。參考程式如下：

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int main()
{
    int a=3,b=4,c=5;
    int x=1,y=2;
    double d=fabs(a*x+b*y+5)/sqrt(a*a+b*b);
                                //abs()傳回整數絕對值，fabs()傳回實數絕對值
    printf("%f",d);
    return 0;
}
```

 自我練習

1. 如何判斷點是否在直線，那就是將此點指派為 x,y ，然後判斷 $ax+by+c$ 是否等於 0，這下一章討論。
2. 請寫一個程式，可以給定兩點，由電腦造出直線方程式。

範例6-3c

請寫一個程式，可以輸入一個一元二次方程式，並求其解（本例假設所輸入的方程式恰有二解）。

 程式設計步驟

1. 寫出演算法。

國中解一元二次方程式是先用配方法，配方法需要一點判斷，可減少計算，但是電腦可不用如此麻煩，電腦計算能力強，所以不怕計算，通常可直接代公式。公式法雖計算較多，但完全不用判斷，最適合由電腦完成，此即為電腦的運算思維，公式法演算法如下：

- (1) 設有一元二次方程式如下：

$$ax^2 + bx + c = 0$$

- (2) 輸入 a,b,c 三個整數。（本例假設整係數方程式，先以

```
int a=2,b=-7,c=3;
```

指派其值。

- (3) 令 $d = \sqrt{b^2 - 4ac}$ 。提示：此為數學語言，C 語言運算式是 $d = \text{sqrt}(b*b - 4*a*c)$ ；，括號、乘號均不能省。

- (4) 則其二解分別為 $x_1 = \frac{-b+d}{2a}$ ， $x_2 = \frac{-b-d}{2a}$ 。（提示：此為數學語言，

C 語言運算式是 $x1 = (-b+d)/(2*a)$; $x2 = (-b-d)/(2*a)$ ，括號、乘號均不能省。

- (5) 例如， $2x^2 - 7x + 3 = 0$ 。其解為 $x_1 = 3.0$, $x_2 = 0.5$ 。

 輸出結果

```
input a,b,c :2,-7,3
x1=3.000000
x2=0.500000
```


 程式列印

```

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int main() {
    int a,b,c;//係數
    double d,x1,x2;
    a=2;b=-7;c=3;
    //printf("input a,b,c :");
    //scanf("%d,%d,%d",&a,&b,&c);//留意&
    d=sqrt(b*b-4*a*c);//開根號
    x1=(-b+d)/(2*a);//留意所有括號不能漏掉
    x2=(-b-d)/(2*a);//留意所有括號不能漏掉
    printf("x1=%f\n",x1);
    printf("x2=%f\n",x2);
    return 0;
}

```

 補充說明

1. 本例假設輸入的係數有實數解，待下一章再判斷所輸入係數有沒有實數解。

 自我練習

1. 寫一個程式，可以輸入一個二元一次方程式，並求其解（本例假設所輸入的方程式恰有一解）。

提示：

1. 解二元一次方程式，國中會先教代入消去法與加減消去法，這都需要一點判斷，但可以簡化計算，但電腦計算能力強，可直接使用公式法，可完全不用判斷，直接計算而得，這樣最適合計算機了，此即為電腦的運算思維。（請慢慢熟悉電腦運算思維，寫程式要用電腦運算思維，不是人腦運算思維）
2. 解二元一次方程式的克拉瑪公式演算法如下：
 - (1) 設二元一次方程式如下：

$$a_1x + b_1y = c_1$$

$$a_2x + b_2y = c_2$$

(2) 輸入 $a_1, b_1, c_1, a_2, b_2, c_2$ 六個整數。(本例假設整係數方程式，可先用指派的方式)

(3) 令 $d = \begin{vmatrix} a_1 & b_1 \\ a_2 & b_2 \end{vmatrix} = a_1 b_2 - a_2 b_1$ 。

(4) 則其解分別是 $x = \frac{\begin{vmatrix} c_1 & b_1 \\ c_2 & b_2 \end{vmatrix}}{d} = (c_1 b_2 - c_2 b_1) / d$ $y = \frac{\begin{vmatrix} a_1 & c_1 \\ a_2 & c_2 \end{vmatrix}}{d} = (a_1 c_2 - a_2 c_1) / d$

(5) 例如：

$$3x + y = 5$$

$$x - 2y = -3$$

則表示

$$a_1 = 3; b_1 = 1; c_1 = 5;$$

$$a_2 = 1; b_2 = -2; c_2 = -3;$$

$$\text{則其解爲 } x = 1 \ y = 2$$

範例6-3d

請寫一程式，可以交換兩個變數的內容。

演算法則

兩個變數的內容要交換，就如同兩隻手的東西要交換位置。初學者都會這樣寫

```
a=b;
b=a;
```

這樣就如同兩隻手同時拋出去，再同時接起來。但是電腦並沒有這個能力，因為電腦需要很明確的演算法如下：

1. 先找來第 3 隻手 t 。
2. 將 a 的東西交給 t 暫存。(此時 a 手才是空了)

```
t=a;
```

3. 將 b 的東西交給 a。(此時 b 手才是空了)

```
a=b;
```

4. 將 t 的東西交給 b，而完成兩隻手東西的交換。

```
b=t;
```

5. 這就是一種電腦的運算思維，電腦運算要很明確。

6. 以下是參考程式。

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    //declare
    int a, b, t;
    //input
    printf("input a,b: ");
    scanf("%d, %d", &a,&b);
    /* swap */
    t=a;
    a=b;
    b=t;
    //output
    printf("after swap a=%d , b=%d", a, b);
    return 0;
}
```

自我練習

1. 輸入三個數，並將 1 交給 2，2 交給 3，3 交給 1，並輸出。

範例6-3e

計數器。生活上有很多計數問題，例如，遊樂場入園人數等。程式設計的計數器寫法如下：請鍵入以下程式，寫出執行結果。

```
#include <stdio.h>
#include <stdlib.h>
int main(){
```

```

int a=8,b=3,q=0,r=0;
a=a-b;
q=q+1;//計數器
printf("a=%d,q=%d\n",a,q);
a=a-b;
q=q+1; //計數器
printf("a=%d,q=%d\n",a,q);
r=a;
printf("q=%d,r=%d\n",q,r);
return 0;
}

```

自我練習

請鍵入以下程式，寫出執行結果。

題號	題目	執行結果
	<pre> #include <stdio.h> #include <stdlib.h> int main() { int a=8,b=3,s=0; s=s+a;//累加器 printf("%d\n",s); s=s+a;//累加器 printf("%d\n",s); return 0; } </pre>	
	<pre> #include <stdio.h> #include <stdlib.h> int main() { int a=21,b=9,r=0; r=a%b; printf("%d\n",r); a=b; b=r; r=a%b; printf("%d\n",r); printf("%d\n",b); return 0; } </pre>	

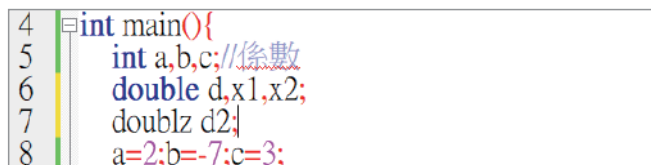
6-4 專案除錯

程式設計的過程當中，會產生程式錯誤 (Bugs) 是很正常的現象，修正錯誤的過程，稱之為除錯 (Debug)。常見的程式錯誤有以下兩種，分別是語法錯誤 (Syntax Error) 與語意錯誤 (Semantic Error)，分別說明如下：

語法錯誤

每一種程式語言都有其撰寫的語法規則，初學者難免疏漏而卡關。所幸目前的編譯器功能也提升了，當錯誤發生時，編譯器會直接顯示相關錯誤的位置與訊息，方便我們進行除錯。以下是一些鍵入程式時常犯的錯誤，整理如下，方便初學者除錯。

1. 指令拚字打錯。CodeBlock 編譯器會將所有指令，也就是保留字以藍色字體呈現。例如：圖 6-3 的 `int`、`double` 都已經以藍色字體呈現，第 2 個 `doublz` 則未呈現藍色字體，此時就很方便除錯。



```

4 int main(){
5     int a,b,c;//係數
6     double d,x1,x2;
7     doublz d2;
8     a=2;b=-7;c=3;

```

圖 6-3 編譯器解析指令回饋的錯誤訊息

2. 變數重複宣告。例如：

```
int a,b,c;
double c;
```

3. 變數未遵守識別字命名規則。例如：

```
int 3a; (變數不能以數字開頭)
```

4. 忘記在敘述行尾端打上分號 (;)，例如：

```
printf("x1=%f\n",x1)
```

則錯誤訊息如圖 6-4：

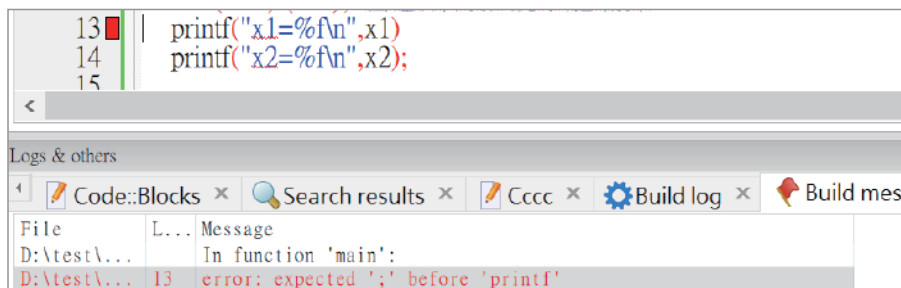


圖 6-4 編譯器解析指令回饋的錯誤訊息

且分號『;』是英數下的半形，不是中文狀態下的全形。

5. 載入標頭檔要使用「#」，且不需要「;」結束。例如：

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

6. 大小寫未區分。C 語言規定大小寫的字母是不同的字元。例如：

```
printf("Hello"); Printf("Hello"); //請留意P的大小寫
```

就不一樣了。

7. 不該出現的空白符號。例如，把

```
printf("Hello");
```

打成

```
print f("Hello");
```

也不行。

8. 拼錯字。例如將

```
printf("Hello");
```

打成

```
prinntf("Hello"); //請留意m與n或1（數字）與l（字母）
```

打字錯誤，導致指令錯誤，也會出現錯誤訊息。

9. 括弧未配成對。有一個左大括弧『{』，務必要配一個右大括弧『}』；同樣地，左小括弧『(』務必要配一個右小括弧『)』；左中括弧『[』務必要配右中括弧『]』。初學者經常會忘記打上右括弧，導致語法錯誤。
10. 引號未成對。不論雙引號『"』(用於字串)，或是單引號『'』(用於字元)的使用，都應該成雙成對出現。
11. 數學語言與程式語言。於前面範例中 $x_1 = \frac{-b+d}{2a}$ 是數學語言，程式語言如下：

```
x1=(-b+d)/(2*a);
```

請留意所有括號不能漏掉與乘號都不能省略。

⚠ 語意錯誤 (Semantic Error)

語意錯誤又稱之為邏輯錯誤，一般發生在語法正確且程式可以執行的情況下，但執行結果卻不如預期。語意錯誤時，程式語法正確所以不會顯示任何的錯誤訊息，程式設計者必須重新思考程式的邏輯錯誤點，才能排除錯誤。例如，以下程式的結果不如預期，且沒有錯誤訊息。

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int main () {
    int a,b,c; //係數
    double d,x1,x2;
    printf("input a,b,c :");
    scanf("%d,%d,%d",&a,&b,&c); //留意&
    d=sqrt(b*b-a*c);
    x1=(-b+d)/(2*a); //留意所有括號不能漏掉
    x2=(-b-d)/(2*a); //留意所有括號不能漏掉
    printf("x1=%f\n",x1);
    printf("x2=%f\n",x2);
    return 0;
}
```

此時可以將輸入值先指派，且將運算過程先輸出，程式如下：

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int main () {
    int a,b,c; //係數
    double d,x1,x2;
    //將輸入先改為註解
    //printf("input a,b,c :");
    //scanf("%d,%d,%d",&a,&b,&c); //留意&
    a=2;b=-7;c=3; //指派輸入值
    d=sqrt(b*b-a*c);
    printf("d=%lf\n",d); //輸出運算過程
    x1=(-b+d)/(2*a); //留意所有括號不能漏掉
    x2=(-b-d)/(2*a); //留意所有括號不能漏掉
    printf("x1=%lf\n",x1);
    printf("x2=%lf\n",x2);
    return 0;
}
```

由以上 d 值的輸出，此時範圍縮小，可以發現 d 值就錯了，關鍵就在這一行，就可以很快發現問題。其次，初學者經常會忽略運算子的優先順序。例如：

```
x1=(-b+d)/(2*a)
```

若寫成

```
x1=(-b+d)/2*a
```

結果就不一樣，但程式不會出現錯誤訊息。

🐞 CodeBlock Debug 功能

以上我們使用 printf() 將 d 值輸出，也可以使用 CodeBlock 的 Debug 功能。其步驟如下：

1. 指派程式的追蹤點。如圖 6-5，於「a=2;b=7;c=3;」的前面，點選滑鼠右鍵，再點選「Add breakpoint」。

```

4 int main() {
5     int a,b,c; //係數
6     double d,x1,x2;
7     //將輸入先改為註解
8     //printf("input a,b,c :");
9     //scanf("%d,%d,%d",&a,&b,&c); //留意 &
10    a=2;b=-7;c=3; //指派輸入值
11    d=sqrt(b*b-a*c);
12    x1=(-b+d)/(2*a); //留意所有括號不能漏掉
13    x2=(-b-d)/(2*a); //留意所有括號不能漏掉
14    printf("x1=%lf\n",x1);
15    printf("x2=%lf\n",x2);
16    return 0;
17 }

```

圖 6-5 程式除錯視窗

2. 逐步追蹤程式。

- (1) 點選功能表「Debug/Start/Continue」或「Debug/Run to cursor」，畫面如圖 6-6：

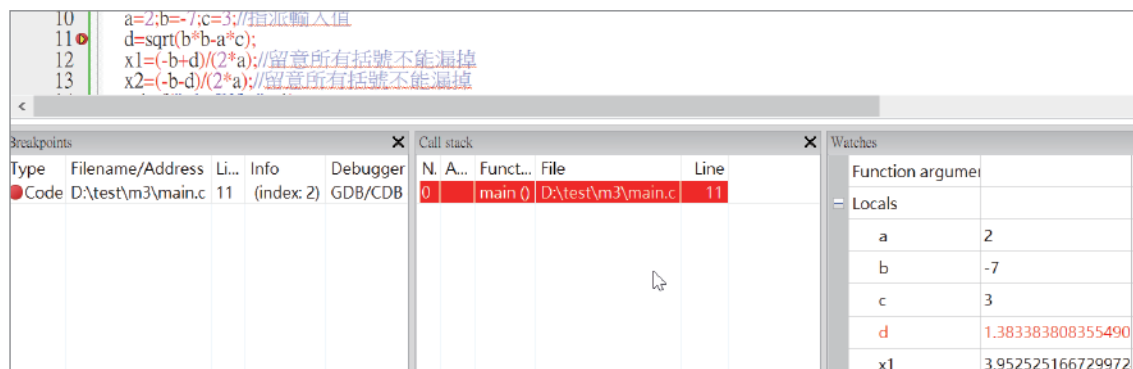


圖 6-6 程式除錯視窗

請留意編譯器會出現「Watch」窗格(若沒出現，請點選功能表「Debug/Debugging windows/Watches」)，此窗格會出現變數內容，此時即可觀察變數運算過程是否正確。

- (2) 逐步點選「Debug/Next line」，即可觀察每一運算，逐一清點每一運算式是否正確。

 自我練習

1. 請同學們兩個人一組，每個人將以上程式製作 2 個錯誤，請對方除錯。

課後習題

請先追蹤程式寫出執行結果，然後鍵入以下程式，核對自己答案。
(#include <stdio.h> #include <stdlib.h> 請自己補上)

題號	題目	結果
1	<pre>int main(){//APCS int a=2,b=3; int c=4,d=5,val; val=b/a+c/b+d/b; printf("%d\n",val); return 0; }</pre>	
2	<pre>int main(){//APCS int Total=23, Paid=1000, Change; Change = Paid - Total; printf ("500 : %d pieces\n", (Change-Change%500)/500); Change = Change % 500; printf ("100 : %d coins\n", (Change-Change%100)/100); Change = Change % 100; // A 區 printf ("50 : %d coins\n", (Change-Change%50)/50); Change = Change % 50; // B 區 printf ("10 : %d coins\n", (Change-Change%10)/10); Change = Change % 10; // C 區 printf ("5 : %d coins\n", (Change-Change%5)/5); Change = Change % 5; // D 區 printf ("1 : %d coins\n", (Change-Change%1)/1); Change = Change % 1; return 0; }</pre>	

3	<pre> int main(){//APCS _Bool x=1,y=1,z=0,a; printf("%d",!(y z) x); printf("%d",!y (z !x)); printf("%d",z (x&&(y z))); printf("%d", (x x)&&z); return 0; } </pre>	
4	<pre> //Arithmetic Operator //The Precedence of Arithmetic //The Associativity of Arithmetic #include <iostream>//C++ using namespace std; int main() { int a=1,b=2,c=3,d=4,e=5; cout<<a*b-c*-d<<endl; cout<<a/b*c+d+ ++e<<endl; a=(b=2)+(c=3); cout<<a<<endl; // 上式括號可否去掉 a=1;b=2;c=3; a+=b+=c+=5; cout<<"a="<<a<<endl; cout<<"b="<<b<<endl; cout<<"c="<<c<<endl; a=1;b=2;c=3; a=++b + ++c; cout<<a<<","<<b<<","<<c<<endl; a=b++ +c++; cout<<a<<","<<b<<","<<c<<endl; a=++b + c++; cout<<a<<","<<b<<","<<c<<endl; a=b-- + --c; cout<<a<<","<<b<<","<<c<<endl; a=1;b=2;c=3; d+=b+c; cout<<"d="<<d<<endl; d*=a+b+5; cout<<"d="<<d<<endl; return 0; } </pre>	

5	<pre>//Relational Operator #include <iostream>//C++ using namespace std; int main() { int a=1; cout<<(a>4)<<endl; cout<<(a>=0)<<endl; cout<<(a==1)<<endl; cout<<(a=5)<<endl; cout<<(a!=1)<<endl; //cout<<(a>=4)<<endl; //cout<<(a!=5)<<endl; int a=1,b=2,c=3,d=4,e=5; c=(a+3)==b; cout <<c<<endl; c=(a+1)==b; cout <<c<<endl; return 0; }</pre>	
6	<pre>#include <iostream>//C++ using namespace std; //Logital Operator //The Precedence of Logical //The Associativity of Logical int main() { int a=5,b=4,c=3,d=2,e=1; cout<<((a!=b) & (b>=3))<<endl; cout<<((a!=b) ^ !(b>=3))<<endl; cout<<(!(a!=b) !(b>=3))<<endl; cout<<((a==1)& (a>0) && (b>=2))<<endl; cout<<((a==1) (a>0) & (b>2) ^ (c==3))<<endl; cout<<((a==1) (a>0) && (b>2) ^ (c==3))<<endl; return 0; }</pre>	

7	<pre>//The Bitwise Operator，逐位元運算 #include <iostream>//C++ using namespace std; int main() { int a=3,b=-2,c=3,d=4; cout<<~a<<endl; cout<<~b<<endl; cout<<(a<<2)<<endl;// 左移 2 位元 cout<<(a>>2)<<endl;// 右移 2 位元 cout<<(b<<2)<<endl; cout<<(b>>2)<<endl; cout<<(a & b)<<endl;// 逐位元 and 運算 cout<<(a b)<<endl; // 逐位元 or 運算 cout<<(a ^ b)<<endl; 逐位元互斥或運算 cout<<(a & b c ^ d)<<endl; a<<=3; // 左移 3 位元，並放回 cout<<a<<endl; b>>=2; cout<<a<<endl; return 0; }</pre>	
8	<pre>#include <iostream>//C++ using namespace std; int main() { int a=4;b=5,t; t=a;a=b;b=t; cout<<a<<endl; cout<<b<<endl; return 0; }</pre>	
9	<pre>int main(){// 統測 111 int y, a = 45; if(a >= 60) y = a + 1; else if(a >= 50) y = a + 2; else y = a + 3; printf("%d",y); }</pre>	

流程指令

chapter

07

學習綱要

- 📖 7-1 if 指令
- 📖 7-2 select case 指令
- 📖 7-3 流程指令應用實例

學習目標

- 📖 1. 認識 C 語言流程指令的種類。
- 📖 2. 能以 if 指令完成決策敘述。
- 📖 3. 能以 select case 完成決策敘述。
- 📖 4. 能實作流程指令實例。

人類的生活必須不斷面對決策問題，連我家一個不到三歲的小孩，也常要思考他手裡的十元是要坐電動車還是買棒棒糖。程式語言是協助解決人類問題的工具，當然也有決策流程敘述，C 語言依決策流程點的多寡，分為以下兩種決策流程指令，第一是雙向分歧決策流程 `if...else...`，例如肚子餓了就吃飯，否則繼續前進；第二是多向分歧決策流程的 `switch case`，例如你身上有 5000 元，走進一家五星級的大飯店用餐，你的分歧點就很多，有自助餐、中式套餐、日本料理、泰國餐點等等分歧點。

7-1 if...else

在日常生活領域中，當出現“假如～則～，否則～”時，此種決策流程模式有兩種解決問題的方案，故稱為雙向分歧決策流程，此時可使用 `if...else` 指令。`if...else` 指令的語法如下：

```
if (運算式)
{
    敘述區塊1;
}
[else
{
    敘述區塊2;
}]
```

以上語法說明如下：

1. 運算式的值若為 1(true)，則執行敘述區塊 1；運算式的值若為 0(false)，則執行敘述區塊 2，其流程圖如圖 7-1：

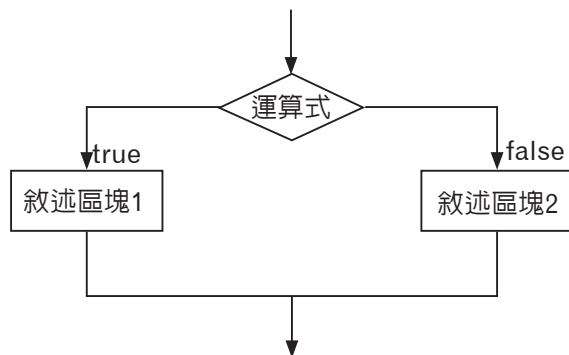


圖 7-1 if...else 流程圖

請鍵入以下程式，並觀察執行結果。

```
int main(){
    int a=66;
    char r='';
    if (a>=60){
        r='P';
    }
    else{
        r='F';
    }
    printf("%c",r) ;
}
```

2. if 或 else 後面敘述區塊若僅執行一個敘述，則 {} 可省略。例如：以上程式可以表示如下：

```
if (a>=60)
    r='P';
else
    r='F';
```

3. 程式設計語法的成對中括號的內容，表示可省略。例如，以下程式僅使用 if。

```
int main(){
    int a=66;
    char r='';
    if (a>=60){
        r='P';
    }
    printf("%c",r) ;
}
```

4. 有時爲了簡化程式的撰寫，可將否則的部分寫在 if 前面，並省略 else。例如，以下程式同義於上面程式。

```
int main(){
    r='F';
```

```

    if(a>=60)
        r='P';
}

```

5. if 或 else 後面預設僅執行一個敘述，若要執行兩個以上敘述，請加上 {}，且 {} 請要對稱排列。請鍵入以下程式，並寫出執行結果。

```

int main(){
    int a=33,b=0,c=0;
    if (a>=60)
        b=b+1;
        c=c+1;
        printf("%d\n",b) ;
    printf("%d\n",c) ;
    return 0;
}

```

```

int main(){
    int a=33,b=0,c=0;
    if (a>=60){
        b=b+1;
        c=c+1;
    }
    printf("%d\n",b) ;
    printf("%d\n",c) ;
    return 0;
}

```

6. 由以上兩段程式可以看出，左邊的程式碼，其縮排型式與程式執行流程截然不同，會造成後續維護的困擾。以下程式縮排型式的撰寫風格才與程式執行流程相符。

```

int main(){
    int a=33, b=0, c=0;
    if (a>=60)
        b=b+1;
    c=c+1;
    printf("%d\n", b);
    printf("%d\n", c);
    return 0;
}

```

7. if() 後面不用加分號 (;)，請鍵入以下程式，並觀察執行結果。

```

int main(){
    int a=33,b=0,c=0;
    if (a>=60)
        b=b+1;
        printf("%d\n",b) ;
    return 0;
}

```

```

int main(){
    int a=33,b=0,c=0;
    if (a>=60);
        b=b+1;
        printf("%d\n",b) ;
    return 0;
}

```

8. C 語言爲了強調語法的簡潔性，若決策流程的結果，只爲求得簡單的運算式，則可用以下敘述的三元運算子（?:），

```
r=(運算式1)?(運算式2):(運算式3);
```

其中『運算式 2』與『運算式 3』的括號只是增加程式的可讀性，此兩括號的有無並不影響程式的執行結果。當運算式 1 的值爲 1(true) 時，r= 運算式 2，當運算式 1 的值爲 0(false) 時，則 r= 運算式 3，例如，請鍵入以下程式，並寫出執行結果。

```
int main(){
    int a=66;
    char r;
    r=(a>=60)?('P'):( 'F');
    printf("%c\n",r) ;
    return 0;
}
```

9. 敘述區塊內可以放置任何合法敘述，當然也可以再放置 if；if 中有 if，稱爲巢狀 if，請看以下範例。

範例 7-1a

請寫一個程式，完成以下要求：

1. 輸入一個 0~100 的分數。
2. 當分數大於 90 分時，輸出 A。
3. 當分數介於 80~89 時，輸出 B。
4. 當分數介於 70~79 時，輸出 C。
5. 當分數介於 0~69 時，輸出 D。

👉 流程分析

1. 使用流程圖分析如圖 7-2：

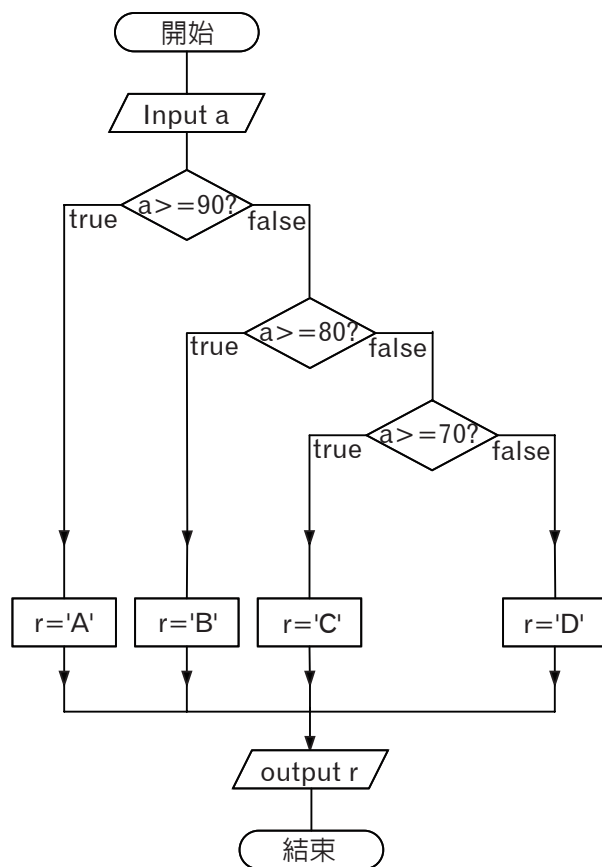


圖 7-2 流程圖

2. 以上每一個決策流程點，都有兩個分歧點，所以適用 if~else。
3. 每一個 else 後面均需進一步決策流程，所以可在每一 else 後面再放置 if。

```

#include <stdio.h>
#include <stdlib.h>
int main() {
    int a=88;
    char r;
    if(a>=90)          /* 高於90分為A */
        r='A';
    else

```

```

        if(a>=80)                /* 介於 80與90分爲B */
            r='B';
        else
            if(a>=70)            /* 介於 70與80分爲C */
                r='C';
            else
                r='D';           /* 不符合上述情況則爲D */
    printf("%c\n",r) ;
    return 0;
}

```

👉 補充說明

1. 此題有人會寫成 `if (90<a<100 )`，但 C/C++ 並沒有這種運算式，因為 `90<a<100` 是數學語言，不是程式語言。
2. 有人會這樣寫，這樣也可以，這樣可以減少一直縮排。

```

    if(a>=90)                    //高於90分爲A
        r='A'
    else if(a>=80)               //介於 80與90分爲B
        r='B';
    else if(a>=70)               // 介於 70與80分爲C
        r='C';
    else
        r='D';                  // 不符合上述情況則爲D
    printf("%c\n",r) ;

```

3. 有人會寫成

```

    if (a<100 && a>=90) r='A';
    if (a<90  && a>=80)  r='B';
    if (a<80  && b>=70)  r='C';
    if (a<70  && b>=0)   r='D';

```

這樣雖然也可以，但是其執行效率非常差。因為不管分數為何，僅有一個敘述會成立，但卻都要進行四次判斷，如此會讓電腦不斷進行無效的判斷。而且，電腦的判斷時間遠大於執行計算的時間。

自我練習

1. 程式的追蹤。

- (1) 設定程式追蹤點如圖 7-3：(於程式追蹤點點選快選功能表的「Add breakpoint」)。

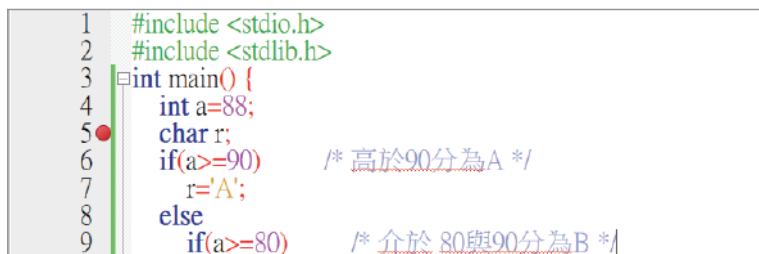


圖 7-3 程式追蹤準備視窗

- (2) 點選功能表的「Debug/Continue」。
- (3) 逐一點選「Next line」，請觀察電腦執行指令的順序。
2. 請寫一程式，由電腦產生 -3 ~ 3 的整數亂數，電腦判斷是正數、0 或負數。
3. 某一貨品定價 100 元，若購買 500 件，(含) 以上打 7 折，若購買 499 ~ 300 件則打 8 折，若購買 299 ~ 100 件則打 9 折，購買 100 件以下則不打折，試寫一程式可以輸入購買件數而得總價。

測試資料：

編號	輸入	結果
1	20	$100 * 20 = 2000$
2	200	$100 * 200 * 0.9 = 18000$
3	400	$100 * 400 * 0.8 = 32000$
4	500	$100 * 500 * 0.7 = 35000$

4. 直線標準式為 $ax+by+c=0$ ，請寫一程式，可以指派一直線係數 a, b, c 。其次，可再輸入任一點座標，並判斷所輸入點式否在直線上。例如，指派 a, b, c 分別是 1, 2, -3，那方程式就是 $x+2y-3=0$ ，其次，輸入點若是 (1, 1)，是在直線上，若是 (1, 2)，則不在直線上。
5. 請寫一程式，可以由電腦產生一個小寫字元，請判斷其是否為母音。(補充說明：字元 a, e, i, o, u 稱為母音，其餘為子音。)

6. 請寫一個程式，可以產生一個 0 到 25 的亂數，且依以下分數顯示燈號

21~25：五個燈。
16~20：四個燈。
11~15：三個燈。
6~10：兩個燈。
1~5：一個燈。
0：零個燈。

範例 7-1b

請寫一個程式，可以判斷所輸入座標的所在象限。

程式列印

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int x=3,y=4;
    char r=' ';
    if(x>0)
        if(y>0)
            r='1';          /* 第一象限 */
        else
            r='4';          /* 第四象限 */
    else
        if(y>0)
            r='2';          /* 第二象限 */
        else
            r='3';          /* 第三象限 */
    printf("x =%d,y=%d : 第 %c 象限",x,y,r);
    return 0;
}
```

補充說明

1. 此題目有人會寫成

```
if (x>0 && y>0) r='1';
if (x<0 && y>0) r='2';
if (x<0 && y<0) r='3';
if (x>0 && y<0 ) r='4';
```

這樣雖然沒有錯，但是執行效率非常差，因為電腦要不斷的比較。

 自我練習

1. 同上範例，但增加先判斷是否在原點或 x、y 軸上。
2. 同上題，但程式一執行，要求先輸入密碼，密碼若為數字 1234，才可執行本程式。

範例7-1c

打字練習。請由電腦產生一個小寫字元，讓使用者輸入，電腦並判斷是否正確。

 程式列印

前面我們已經產生 1 個字元，由使用者輸入字元，現在加上判斷，所以程式如下：

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h> //本例使用time() 函式
int main()
{
    srand(time(NULL)); //以時間當作亂數種子，請複習5-3節
    int a=97+rand()%26;
    printf("%c",a);
    char b;
    scanf("%c",&b);
    if ((int)b==a) //請留意，資料型態要相同，才能比較，所以要先轉型
        printf("right");
    else
        printf("wrong");
}
```

 自我練習

1. 心算練習。請由電腦產生兩個 1 位數，讓使用者練習加法，電腦並評判是否正確。
2. 音感練習。請由電腦發一個 Do 到高音 Do 的音，讓使用者輸入 1 到 8，電腦評判是否正確。

7-2 switch...case

一個決策流程點若同時擁有三個或三個以上的解決方案，則稱此為多向分歧決策流程。多向分歧決策流程雖也可使用 7-1 的巢狀 if else 解決，但卻增加程式的複雜度及降低程式可讀性。若此一決策流程點能找到適當的運算式，能使問題同時找到分歧點，則可使用 switch case 敘述。switch case 語法如下：

```
switch (運算式)
{
    case 常數1:
        敘述區塊1;
        break;
    case 常數2:
        敘述區塊2;
        break;
    case 常數3:
        敘述區塊3;
        break;

    [default:
        敘述區塊n;]
}
```

以上語法說明如下：

1. switch 的運算式值僅能為『整數』或『字元』。
2. case 的常數僅能『整數』或『字元』，且其資料型態應與上面的 switch 運算式相同。
3. 電腦將會依 switch 的運算式值，逐一至常數 1、常數 2 尋找合乎條件的 case，並執行相對應的敘述區塊，直到遇到 break 敘述，才能離開 switch。
4. default 可放置特殊情況，也就是沒有適當的 case，則執行 default。若省略 default，且若沒有任何 case 滿足 switch 運算式，則程式會默默離開 switch 敘述。

5. 語法中，成對中括號裡面的敘述，表示可省略。
6. 敘述區塊可放置任何合法的敘述，當然也可放置 `switch` 或 `if`。
7. 以下敘述，可將 1、2、3、4 轉為字元 'A'、'B'、'C'、'D'。

```
int main() {  
    int a=1;  
    char r;  
    switch(a) {  
        case 1:  
            r='A';  
            break;  
        case 2:  
            r='B';  
            break;  
        case 3:  
            r='C';  
            break;  
        case 4:  
            r='D';  
            break;  
        default :  
            r=' '  
    }  
    printf("%c\n",r) ;  
    return 0;  
}
```

8. 有些語言可用逗號將兩種 `case` 放在一起，但在 C/C++ 語言中每一 `case` 僅能放置一個常數，所以若兩個或兩個以上 `case`，有相同的處理方法，則應將兩個 `case` 分成兩個敘述，且共用 1 個 `break`，如以下程式。

```
case 10:  
case 9:  
    r='A';  
    break;
```

請看以下範例。

範例7-2a

試以 switch case 重作範例 7-1a。

 程式列印

1. 可使用 return 可提早離開 main() 函式。

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int a;
    char r;
    printf("input a grade: ");
    scanf("%d",&a); //&
    if(a>100 || a<0) { // 如果輸入的成績大於100或小於0，則 提早離開
        return 0;
    }
    switch(a/10) {
        case 10:
        case 9:
            r='A';
            break;
        case 8:
            r='B';
            break;
        case 7:
            r='C';
            break;
        case 6:
        case 5:
        case 4:
        case 3:
        case 2:
        case 1:
        case 0:
            r='D';
            break;
    }
    printf("the grade is %c",r);
    return 0;
}
```

2. 以上程式也可以用「…」改寫，程式如下：

```
int main() {
    int a=99;
    char r=' ';
    switch(a) {
        case 90 ...100:
            r='A';
            break;
        case 80 ...89:
            r='B';
            break;
        case 70 ...79:
            r='C';
            break;
        case 0 ...69:
            r='D';
            break;
    }
    printf("the grade is %c",r);
    return 0;
}
```

自我練習

1. 請寫一程式，將所輸入的 0、1、2…6，轉為 'S'、'M'、'T'、'W'、'T'、'F'、'S' 等字元。
2. 請寫一個程式，可以產生一個 0 到 25 的亂數，且依以下分數顯示燈號

21～25：五個燈。
16～20：四個燈。
11～15：三個燈。
6～10：兩個燈。
1～5：一個燈。
0：零個燈。

範例 7-2b

電子琴。前面我們已經介紹電腦如何使用 `_beep()` 函式發 Do、Re、Me 等音，本例則要依此製作一個電子琴。

 解題步驟

1. 使用 `scanf()` 輸入字元都還要輸入 Enter，這樣不像電子琴。所以線上查尋，發現 `getche()` 函式（放在 `conio.h`）可輸入字元，但不用輸入 Enter，所以本例使用此函式輸入字元。參考程式如下：

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h> // 本例使用 getche
int main() {
    char a;
    while(1) { // 無限迴圈，請看下一章
        a = getche(); // 輸入後，不用按 Enter
        switch(a) {
            case ('1'):
                _beep(523, 500); // Do
                break;
            case ('2'):
                _beep(587, 500); // Re
                break;
            case ('3'):
                _beep(659, 500); // Mi
                break;
            case ('4'):
                _beep(698, 500); // Fa
                break;
            case ('5'):
                _beep(784, 500);
                break;
            case ('6'):
                _beep(880, 500);
                break;
            case ('7'):
                _beep(988, 500);
                break;
```

```
        case ('8'):  
            _beep(1046,500);  
            break;  
    }  
}  
return 0;  
}
```

範例7-2c

猜拳遊戲。請寫一個程式，可以由人和電腦猜拳。

解題步驟

1. 這一任務就是寫人工智慧程式的入門了，寫程式前先想一下您和一個 3 歲小孩或沒玩過猜拳的人猜拳，請問您們如何開始此遊戲呢？首先，我們先規定有三種拳，分別是『剪刀、石頭與布』，且『剪刀贏布，石頭贏剪刀、布贏石頭，兩者相同則平手等』。『人工智慧』就是您聰明，它傻瓜，要把這些規定以程式語言表示，並由以上規定來執行遊戲與評判勝負。
2. 資料的數位化。我們人類猜拳是直接用手勢表示『剪刀、石頭與布』，但是如何讓電腦瞭解您的拳法呢，裝一個攝影機當然可以，但是也還要影像處理，這不是我們這一單元任務，我們電腦只有鍵盤，所以可以先將以上『剪刀、石頭與布』以『1,2,3』表示，此即為資料的數位化。因為用『1,2,3』，表示只要 1byte 就可以，若您用『剪刀、石頭與布』表示，當然也可以，但一個中文字就佔用 3byte，且往後的處理也是比較複雜。
3. 本例因為電腦有三種拳法，可以用一個 switch，人也有三種拳法，所以每一個 switch 裡面還有一個 switch，所以共有 9 種情況，您要讓電腦也能思考，其實就是將以上所有可能發生的情況，以程式語言先規定好，如以下程式。

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h> //本例使用Time() 函式
#include <string.h> //本例使用strcpy
int main() {
    int a,b;
    char a1[7],b1[7],r[7]; //C用字元陣列表示字串，請看9-3
    srand(time(0)); //隨機選取亂數起點
    printf( "input 0(剪刀) 1(石頭) 2(布) :");
    scanf("%d",&a); //留意&
    b=rand()%3; //產生0,1,2的亂數，需要#include <stdlib.h>
    switch (a){
        case 0:
            strcpy(a1,"剪刀"); //C字串不能a1="剪刀"，要用strcpy() 函式
            switch(b){
                case 0:
                    strcpy(b1,"剪刀");
                    strcpy(r,"平手");
                    break;
                case 1:
                    strcpy(b1,"石頭");
                    strcpy(r,"電腦贏");
                    break;
                case 2:
                    strcpy(b1,"布");
                    strcpy(r,"你贏");
                    break;
            }
            break;
        case 1:
            strcpy(a1,"石頭");
            switch(b){
                case 0:
                    strcpy(b1,"剪刀");
                    strcpy(r,"你贏");
                    break;
                case 1:
                    strcpy(b1,"石頭");
                    strcpy(r,"平手");
                    break;
```

```

        case 2:
            strcpy(b1,"布");
            strcpy(r,"電腦贏") ;
            break;
    }
    break;
case 2:
    strcpy(a1,"布");
    switch(b) {
        case 0:
            strcpy(b1,"剪刀");
            strcpy(r,"電腦贏") ;
            break;
        case 1:
            strcpy(b1,"石頭");
            strcpy(r,"你贏") ;
            break;
        case 2:
            strcpy(b1,"布");
            strcpy(r,"平手") ;
            break;
    }
    break;
}
printf("你出:%s, 電腦出:%s 結果:%s",a1,b1,r);
return 0;
}

```

補充說明

1. 本例使用亂數函式 rand()，所以要宣告 #include <stdlib.h>。
2. 本例使用時間函式 time()，所以要宣告 #include <time.h>。
3. 本例引入字串的使用，C 沒有字串 string 型態，要用字元陣列宣告，如下敘述：

```
char a1[7],b1[7],r[7]; //字元陣列，字串
```

宣告 a1、b1、r 是長度為 6 個字元的字串，且字串是用雙引號，如下敘述：


```
char a1[6]= "剪刀";
```

關於字串的用法，請看 11-3 節。

4. 字元陣列除了宣告時可直接指派字串，如以上敘述，程式中無法使用指派運算子『=』，直接指派變數，以下敘述錯誤：

```
a1="剪刀" ;
```

要用 strcpy() 函式，如以下敘述：

```
strcpy(a1,"剪刀");
```

才能重新改變變數值，請看 9-3 節。

自我練習

1. 請寫一程式，可以執行三個人的同時猜拳遊戲，並評定勝負（可電腦產生兩個亂數，使用者輸入一個）。
提示：以上兩個人，使用兩層 switch case，三個人，就使用三層。

7-3 流程指令應用實例

範例7-3a

求極大值。假如有資料如下：

3,2,5

如何求極大值。

參考解答

1. 資料數位化。本例以單一變數儲存如下：

```
int a=3,b=2,c=5;
```

2. 求極大值的演算法如下：

(1) 指派第 1 個資料為極大值。

```
int max=a;
```

- (2) 從第 2 個資料起，都要和極大值比較，若大於極大值，則取代極大值。

```
if (b> max)
    max=b;
```

3. 輸出極大值。
4. 根據以上演算法，程式撰寫如下：

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int a=3,b=2,c=5;
    int max=a;
    if (b>max)
        max=b;
    if (c>max)
        max=c;
    printf("max:%d",max);
    return 0;
}
```

自我練習

1. 若有 4 筆資料如下：

```
33,22,77,66
```

請寫一個程式，可以求其極小值。

2. 有一批資料，含有 3 個人的人名與成績，請寫一程式，可以找出最低分的人名與成績。例如：

```
aa,60
bb,30
cc,80
```

3. 請寫一程式，可以輸入 5 個數值，請去掉最大值與最小值，再求其平均。

範例 7-3b

假如有 3 筆資料如下：

8,7,3

將此三個數由小而大輸出。

演算法則

1. 分別以 a、b 及 c 表示欲排序的資料。
2. 假如 a 大於 b，則 a 與 b 交換，如下圖的 (1)。
3. 假如 b 大於 c，則 b 與 c 交換，如下圖的 (2)。
4. 假如 a 大於 b，則 a 與 b 交換，如下圖的 (3) 排序已完成，共需進行追蹤準備 3 次的比較與交換，如下圖所示，此稱為氣泡排序法（Bubble Sort）。

a	b	c
(1) a>b?	(2) b>c?	
(3) a>b?		

程式列印

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int a,b,c,t;
    printf("input a,b,c :");
    scanf("%d,%d,%d",&a,&b,&c); //留意&
    //I
    if(a>b)
        { t=a; a=b; b=t; } //兩數交換
    if(b>c)
        { t=b; b=c; c=t; }
    //II
    if(a>b)
        { t=a; a=b; b=t; }
    //output
    printf("a,b,c 由小而大排列是 %d,%d,%d ",a,b,c);
    return 0;
}
```

👉 補充說明

1. 若有 4 筆資料需要排序，則共需進行 3 個階次，6 次比較與交換，如下圖所示。

a	b	c	d
(1) a>b?	(2) b>c?	(3) c>d?	
(4) a>b?	(5) b>c?		
(6) a>b?			

以上的比較與交換，程式如下：

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int a=7,b=6,c=5,d=4,t;
    //階次I
    if(a>b)
        { t=a; a=b; b=t; } //兩數交換
    if(b>c)
        { t=b; b=c; c=t; }
    if (c>d)
        { t=c; c=d; d=t; }
    //階次II
    if(a>b)
        { t=a; a=b; b=t; }
    if(b>c)
        { t=b; b=c; c=t; }
    //階次III
    if(a>b)
        { t=a; a=b; b=t; }
    //output
    printf("a,b,c,d 由小而大排列是 %d,%d,%d %d ",a,b,c,d);
    return 0;
}
```

2. 若有 5 筆資料需排序，則共需進行 10 次比較與交換，如下圖所示，此即為氣泡排序法。

a	b	c	d	e
(1) $a > b?$	(2) $b > c?$	(3) $c > d?$	(4) $d > e?$	
(5) $a > b?$	(6) $b > c?$	(7) $c > d?$		
(8) $a > b?$	(9) $b > c?$			
(10) $a > b?$				

以上的比較與交換，共 4 個階次，請同學自己寫程式完成。

3. 以上待排序資料以單一變數儲存，處理 3、4 或 5 筆資料的比較與排序，其比較與交換的次數尚可克服。但是，若欲排序的資料超過 5 個，例如 20 筆資料欲排序，那程式不就很冗長？所幸，答案是否定的，請大家不用擔心，待迴圈與陣列資料結構介紹以後，程式就有精簡的寫法。
4. 測試排序程式是否正確，測試資料通常要給最惡劣的情況。例如，資料要由小而大排序，測試資料就要由大而小給，例如，9,8,7,6,5，這樣才能測出程式是否正確。也就是最惡劣都能解了，就沒有不能完成的資料。反之，您沒給最惡劣的資料，那程式是否能克服最糟的情況，就不得而知。

自我練習

- 將 4 個數由大而小排序輸出。
- 將 5 個數由小而大排序輸出。
- 假設有 4 筆資料如下：

```
aa, 9
bb, 8
cc, 7
dd, 6
```

分別代表人名與分數，請寫一程式，可依照分數由小而大輸出，且輸出應含人名與分數。

一元二次方程式

前面我們直接假設所輸入的一元二次方程式有解，因為不是隨便給三個係數，一個方程式就通通有解，本例則要先判斷是否有解。解一元二次方程式的演算法如下：

1. 設有一元二次方程式如下：

$$ax^2 + bx + c = 0$$

2. 若 $a=0$ 則輸出“輸入錯誤”。
3. 令 $d=b^2 - 4ac$ 。
4. 若 $d=0$ ，則方程式有唯一解 $x = \frac{-b}{2a}$ ；否則，若 $d>0$ ，則方程式有二解

$$x_1 = \frac{-b + \sqrt{d}}{2a}, x_2 = \frac{-b - \sqrt{d}}{2a} ; \text{否則，無實數解。}$$

5. 例如： $2x^2 - 7x + 3 = 0$ 。其解是 $x_1=3.0, x_2=0.5$ 。

以上演算分析，以流程圖說明如圖 7-4：

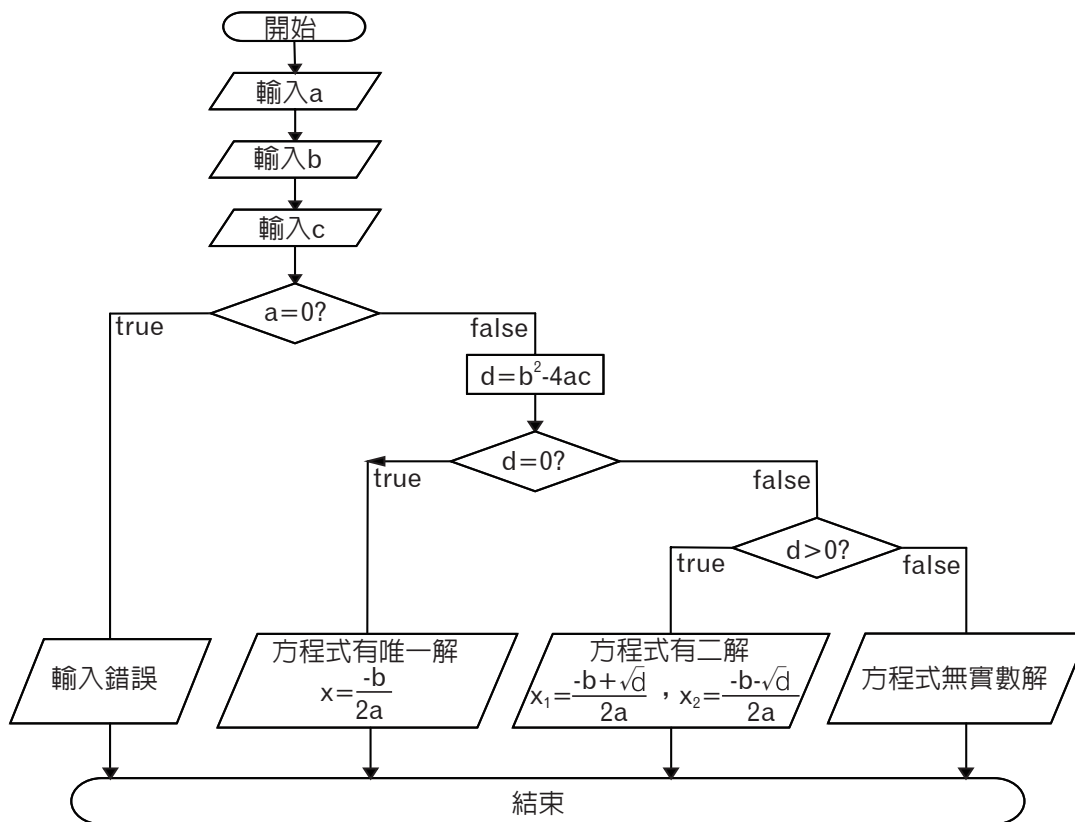


圖 7-4 流程圖

範例 7-3c

請設計一個程式，可以解一元二次方程式 $ax^2 + bx + c = 0$ 。

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int main() {
    int a,b,c;
    double d,x1,x2;
    printf("input a,b,c:");
    scanf("%d,%d,%d",&a,&b,&c); //&
    if(a==0) {
        printf("input error\n"); /* 若a=0，則列印錯誤訊息 */
        return 0;
    }
    d=pow(b,2)-4*a*c;          /* 計算d值 */
    if(d==0)
        printf("only one answer,x1,x2= %f\n",-b/(2*a));
    else
        if(d>0){
            x1=(-b+sqrt(d))/(2*a);
            x2=(-b-sqrt(d))/(2*a);
            printf("two answer,x1= %f,x2=%f",x1,x2);
            //printf("two answer,x1= d");
        }
        else
            printf("no real answer.");
    return 0;
}
```

 自我練習

1. 解二元一次方程式。解二元一次方程式的演算法如下：

(1) 設二元一次方程式如下：

$$a_1x + b_1y = c_1$$

$$a_2x + b_2y = c_2$$

(2) 令 $d = \begin{vmatrix} a_1 & b_1 \\ a_2 & b_2 \end{vmatrix}$ (表示 $d = a_1b_2 - a_2b_1$)

(3) 假如 $\frac{a_1}{a_2} = \frac{b_1}{b_2} = \frac{c_1}{c_2}$ ，則方程式無限多解，且程式結束。(代表兩重疊直線)

(4) 假如 $\frac{a_1}{a_2} = \frac{b_1}{b_2} \neq \frac{c_1}{c_2}$ ，則方程式無解，且程式結束。(代表兩平行直線)

$$(5) \ x = \frac{\begin{vmatrix} c_1 & b_1 \\ c_2 & b_2 \end{vmatrix}}{d} = (c_1 b_2 - c_2 b_1) / d$$

$$(6) \ y = \frac{\begin{vmatrix} a_1 & c_1 \\ a_2 & c_2 \end{vmatrix}}{d} = (a_1 c_2 - a_2 c_1) / d$$

(7) 例如：

$$3x + y = 5$$

$$x - 2y = -3$$

的解為 $x=1, y=2$

請設計一個程式，可以解二元一次方程式。

2. 三角形判斷。(APCS105 第二梯次試題)

若已知三角形三邊長，判斷是否構成三角形、評定三角形種類與計算三角形面積的演算法如下：

(1) 輸入三角形的三邊長 a 、 b 、 c 。

(2) 任兩邊之和要大於第三邊，才能構成三角形，否則程式結束。

(3) 若 c 是最長邊，假如 $a^2 + b^2 > c^2$ 則為銳角三角形；否則，假如 $a^2 + b^2 = c^2$ ，則為直角三角形；否則此三角形為鈍角三角形。

(4) 令 $d = \frac{1}{2}(a + b + c)$

(5) 三角形面積 $= \sqrt{d(d-a)(d-b)(d-c)}$

請輸入三角形三邊長，首先判斷是否構成三角形、其次判別三角形的種類，最後計算其面積。

範例一：輸入 3 4 5 範例一：正確輸出 3 4 5 Right (說明) $a \times a + b \times b = c \times c$ 成立時為直角三角形。	範例二：輸入 101 100 99 範例二：正確輸出 99 100 101 Acute (說明) 邊長排序由小到大輸出， $a \times a + b \times b > c \times c$ 成立時為銳角三角形。	範例三：輸入 10 100 10 範例三：正確輸出 10 10 100 No (說明) 由於無法構成三角形， 因此第二行須印出 「No」。
--	---	--

範例7-3d

複選題評分。大學學測考試的複選題共五個選項，五個選項全答對得 5 分，錯一個選項得 3 分，錯兩個選項得 1 分，錯超過兩個選項得 0 分，未作答也是 0 分，且答錯不倒扣，請寫一程式，可以協助評分。

程式設計步驟

1. 資料的數位化。答案要如何表示呢？若答案是 ABE，那用 ABE 表示嗎？這樣雖然可以，但需要 5 個 bytes，非常浪費記憶體，若將其看為二進位，那只要以 $(11001)_2 = (25)_{10}$ 表示，也就是 1 個 byte 就足夠。
2. 每個答案都有 5 個位元。本例使用 xor 逐一位元運算。例如：
答案是 ABE，則以 11001 表示，測資若為 AE，則以 10001 表示。然後兩者逐位元取 xor 運算，相同的將得到 0。取反向，計算 1 的個數，即可得到答對選項的個數，計算過程如下表所示：

標準答案	ABE	11001
測資 1	AE	10001
xor		01000
取 not		10111
計算 1 的個數		4

所以答對 4 個選項。

程式列印

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int a=25; //標準答案ABE
    int b=25; //答對5個5分
    b=17; // AE答對4個，錯1個3分
    //b=1; // E答對3個，錯2個1分
    //b=2; //D 0分
    //b=0; //未作答
    int c=0;
    int d=0; //答對幾個
```

```
int e=0;//得分
if (b==0)
    e=0;
else {
    c=a^b;//逐位元xor，位元相同是0，不同是1
    d=0;
    printf("%d\n",c);
    c=~c;//取補數，1變0，0變1
    printf("%d\n",c);
    d=d+(c & 1);//計算1的個數
    //d=d+c & 1 // 錯，因為+的優先順序大於&
    c=c>>1;//右移
    d=d+(c & 1);//位元邏輯and
    c=c>>1;
    d=d+(c & 1);
    c=c>>1;
    d=d+(c & 1);
    c=c>>1;
    d=d+(c & 1);
}
if (d==5)
    e=5;
else if (d==4)
    e=3;
else if (d==3)
    e=1;
else
    e=0;
printf("%d\n",d);
printf("%d\n",e);
return 0;
}
```

課後習題

二、填充題

題號	題目	結果
1	<pre> void main () { int count = 10; if (count > 0) { count = 11; } if (count > 10) { count = 12; if (count % 3 == 4) { count = 1; } else { count = 0; } } else if (count > 11) { count = 13; } else { count = 14; } if (count) { count = 15; } else { count = 16; } printf ("%d\n", count); } </pre>	
2	<pre> void main () { int x=20;char y; if (x==10) y = 'a'; if (x==20 x==30) y = 'b'; y = 'c'; printf ("%c",y); } </pre>	

3	<pre>void main () { int x=15;char y; if (x==10) y = 'a'; else if (x==20 x==30) y = 'b'; else y = 'c'; printf("%c",y); }</pre>	
4	<pre>void main () { int x=20;char y; if (x==10) y = 'a'; if (x>=20 && x<=30) y = 'b'; y = 'c'; printf("%c",y); }</pre>	
5	<pre>void main () { int x=20;char y; if (x==10) y = 'a'; else if(x>=20 && x<=30) y = 'b'; else y = 'c'; printf("%c",y); }</pre>	
6	<pre>int main() { int a=1; char r='0'; switch(a) { case '1': r='A'; break; default : r='X'; } printf("%c\n",r) ; return 0; }</pre>	

7	<pre> int main() { int a=1;char r='0'; switch(a) { case 1: r='A'; case 2: r='B'; break; } printf("%c\n",r) ; return 0; } </pre>	
8	<pre> int main() { int a=5; char r='0'; switch(a) { case 1: r='A'; break; case 2: r='B'; break; default : r='X '; } printf("%c\n",r) ; return 0; } </pre>	
9	<pre> int main() { int a=1; char r='0'; switch(a) { case 1: r='A '; break; case 2: r='B '; break; default : r=' X '; } printf("%c\n",r) ; return 0; } </pre>	

10	<pre>void main () { //APCS int x=20;char y; switch (x) { case 10: y = 'a'; break; case 20: case 30: y = 'b'; break; default: y = 'c'; } printf("%c",y); }</pre>	
11	<pre>void main () { //APCS for (int s=100 ;s>=0;s--){ printf("s=%d: ",s); if (s>=90) { printf ("A \n"); } else if (s>=80) { printf ("B \n"); } else if (s>60) { printf ("D \n"); } else if (s>70) { printf ("C \n"); } else { printf ("F\n"); } } printf("\n"); }</pre>	
12	<pre>int main() { int a=5,b=4,t; if (a>b) t=a;a=b;b=t; printf("%d",a); return 0; }</pre>	

13	<pre> //請留意else 是預設最接近的if int main(int argc, char** argv) { //The else default to the nearest int a=-5,b=4,t=0; if (a>0) if (b>0) t++; else t--; printf("%d\n",t); a=-5;b=4;t=0; if (a>0) if (b>0) t++; else t--; printf("%d\n",t); //If you want to change it,you need to use the curly brackets. a=-5;b=4;t=0; if (a>0){ if (b>0) t++; } else t--; printf("%d\n",t); return 0; } </pre>	
14	<pre> int main() { int a=1,b=2,c=2,d=2,e=2; if (a=='1') b=3; printf("%d",b); if (a=1) c=3; printf("%d",c); if (a='1') d=3; printf("%d",d); } </pre>	

二、程式設計

1. 假設所得稅的稅率累進法則如下：

- (1) 淨所得 30 萬以下繳納 6%。
- (2) 淨所得 30 ~ 80 萬之間，則前面 30 萬繳納 6%，超過 30 萬的部分繳納 13%。
- (3) 淨所得在 80 ~ 200 萬之間繳納 21%。(前面 30 萬繳 6%，30 ~ 80 萬之間繳 13%)
- (4) 淨所得超過 200 萬，超過的部分繳納 30%。

試寫一程式可以輸入淨所得，並計算應繳納稅額。測資如下：

淨所得	納稅額
20 萬	$20 \text{ 萬} \times 6\% = 12000$
40 萬	$30 \text{ 萬} \times 6\% + 10 \text{ 萬} \times 13\% = 31000$
100 萬	$30 \text{ 萬} \times 6\% + 50 \text{ 萬} \times 13\% + 20 \text{ 萬} \times 21\% = 125000$
220 萬	$30 \text{ 萬} \times 6\% + 50 \text{ 萬} \times 13\% + 120 \text{ 萬} \times 21\% + 20 \text{ 萬} \times 30\% = 395000$

2. 假設自來水費率如下：

100 度以下，每度 3 元。

100 ~ 300 度，超過 100 度的部分，每度 5 元。

300 度以上，超過 300 度的部分，每度 6 元。

根據以上條件，寫一程式，可輸入用水度數，得到水費。例如，測資如下表：

度數	水費
50	$50 \times 3 = 150$
200	$100 \times 3 + 100 \times 5 = 800$
400	$100 \times 3 + 200 \times 5 + 100 \times 6 = 1900$

3. 假設計程車的計費方式如下：

1000 公尺以內 60 元。超過 1000 公尺時，

(1) 日間以每 500 公尺加收 6 元，不足 500 公尺時，以 500 公尺計算。

(2) 夜間以每 300 公尺加收 6 元，不足 300 公尺，以 300 公尺計算。

請根據以上條件，寫一程式完成車資的計算。本例，日間輸入 1，夜間輸入 0，再輸入一個里程數，並計算車資。測資如下表：

模式	里程	車資	模式	里程	車資
1	500	60	0	500	60
1	1800	60+6*2=72	0	1800	60+6*3=78

4. 寫一程式輸入 x 值，並印出其所對應的值，其函數對應如下：

$$y = f(x) = \begin{cases} x+3 & x > 3 \\ x^2 & 1 \leq x \leq 3 \\ \sqrt{x} & 0 < x < 1 \\ 0 & x \leq 0 \end{cases}$$

5. 請寫一程式，可以輸入三點座標，檢查是否共線，若未共線，輸出其面積。
提示：任 3 點面積若為 0，則共線。（提示：面積公式在第 6 章）
6. 判斷任意點 D 是否在三角形 ABC 內或外。已知 ABC 三點座標，請寫一程式，可以輸入 D 點座標，並判斷任一點 D 是否在三角形 ABC 內或外。（提示： D 若在三角形內，則三角形 ABC 面積 = $DAB + DBC + DAC$ 的面積）
7. 凸或凹四邊形。已知 $ABCD$ 四點座標，請寫一程式，可以判斷四邊形 $ABCD$ 是否為凸或凹四邊形。
（提示：若為凸四邊形，則任一點，均不得在任三點所圍成的三角形內）
8. 同上題，若為凸四邊形，並計算其面積。提示：任意凸多邊形，都可以將第 5 題公式推廣，並且代入。例如，凸四邊形的面積公式是：

$$\frac{1}{2} \left\| \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 & x_1 \\ y_1 & y_2 & y_3 & y_4 & y_1 \end{array} \right\|$$

$$= \frac{1}{2} | (x_1y_2 + x_2y_3 + x_3y_4 + x_4y_1) - (x_2y_1 + x_3y_2 + x_4y_3 + x_1y_4) |$$

（請留意↘方向是相加，↗方向是相減）

迴圈指令

chapter 08

學習綱要

- 📖 8-1 for 迴圈
- 📖 8-2 while 迴圈
- 📖 8-3 迴圈指令應用實例

學習目標

- 📖 1. 認識 C 語言迴圈指令。
- 📖 2. 能以 for 指令完成重複動作。
- 📖 3. 能以 while 指令完成重複動作。
- 📖 4. 能實作迴圈指令實例。

生活上常會遇到重複執行的事件，例如，一天要上七堂課，連續作答選擇題 10 題等等。所以電腦就有所謂的迴圈，幫您完成指定的重複次數。其次，所有的程式語言都有兩種基本迴圈，那就是 for 與 while，其用法與差別請看本章說明。

8-1 for 迴圈

我們小時候常常被老師罰寫某個句子十遍，這件事若讓電腦做，那就輕鬆了，因為電腦有 for 迴圈運算。例如，以下敘述可將『洪峻維』重複寫 10 次。

```
for (int i=1;i<=10;i++)
    printf("洪峻維\n");
```

for 指令的語法如下：

```
for([計數變數=起始值] ; [迴圈運算式] ; [計數變數的改變量]) {
    [指令區塊1;]
    [break;]
    [continue;]
    [goto (標籤名稱);]
    [指令區塊2;]
}
```

以上語法說明如下：

1. 只要“迴圈運算式”結果為 1(真)，則繼續執行迴圈內的指令區塊。
2. for 運算僅預設執行 1 個敘述，若要執行兩個以上敘述，請加上 {}。請鍵入以下程式，寫出執行結果。

```
int main() {
    for (int i=1;i<=10;i++){
        printf("%d\n",i);
        printf("%d\n",i);
    }
}
```

```
int main() {
    int i;
    for ( i=1;i<=10;i++)
        printf("%d\n",i);
        printf("%d\n",i);
}
```

以上變數 *i* 的有效範圍僅在迴圈內，所以以下程式，第 2 個 `printf("%d\n",i);` 的 *i* 無效。

```
int main() {
    for ( int i=1;i<=10;i++){
        printf("%d\n",i);
    }
    printf("%d\n",i);
}
```

3. 計數變數的改變量若為遞增，請留意終值較大；若為遞減，則終值較小。請鍵入以下程式，並寫出執行結果。

```
int main() {
    for (int i=1;i<=10;i++)
        printf("%d\n",i);
    for (int i=1;i<=10;i=i+4)
        printf("%d\n",i);
    return 0;
}
```

```
int main() {
    for (int i=10;i>=1;i--)
        printf("%d\n",i);
    for (int i=10;i>=1;i=i-3)
        printf("%d\n",i);
    return 0;
}
```

4. 前面計數變數都是由小而大，使用『<=』。若是由大到小，則要用『>=』，這樣第一筆資料才會得到『true』，才有輸出結果。

```
int main() {
    for( i=10;i>=1;i--) {
        printf("%d",i);//10987654321
    }
}
```

5. 以下程式每次遞減 3。

```
int main() {
    for( i=10;i>=1;i-=3) {
        printf("%d",i);//10741
    }
}
```

6. 關係運算子通常使用『<,<=,>,>=』，若使用『==,!=』，則沒有錯誤訊息，但執行結果不同。請鍵入以下程式，寫出執行結果。

```
int main(){
    for (int i=1;i==5;i++)
        printf("%d",i);
    printf("\n");
    return 0;
}
```

```
int main(){
    for (int i=1;i!=5;i++)
        printf("%d",i);
    printf("\n");
    return 0;
}
```

7. 迴圈運算式也不能寫成：

```
int main() {
    for ( int i=1;i=10;i++)
        printf("%d",i);
}
```

因為 C 關係運算子沒有「=」，這樣是語法錯誤。其次，若寫成：

```
int main() {
    for ( i=1;i==10;i++)
        printf("%d",i);
}
```

則是語法沒錯，但結果不一樣，因為一開始 `1==10` 這件事就 `false`，所以什麼都沒有作就離開迴圈。

8. 只要迴圈運算式為 `0(false)`，就離開迴圈。請鍵入以下程式，並寫出執行結果。

```
int main(){
    for (int i=1;i>=5;i++)
        printf("%d\n",i);
    for (int i=5;i<=1;i++)
        printf("%d\n",i);
    return 0;
}
```

9. 計數變量也可在迴圈內改變。例如：

```
int main(){
    for (int i=1;i<=10 ;i++){
        printf("%d\n",i);
        i++;
    }
    return 0;
}
```

10. 程式若執行到 break，則會提早離開 for 迴圈。(請鍵入以下程式，寫出執行結果)

```
int main(){
    int i, sum=0;
    for(i=1; i<=6; i++) {
        printf("%d\n",i);
        if(i==4)
            break;    //強制離開迴圈
        printf("%d\n",i);
    }
    printf("when out of loop, i=%d",i);
    return 0;
}
```

11. 程式若執行到 continue，則會略過 continue 下面的指令區塊 2，繼續執行下一個計數變量。(請鍵入以下程式，寫出執行結果)

```
int main(){
    int i, sum=0;
    for(i=1; i<=6; i++) {
        printf("%d\n",i);
        if(i==4)
            continue;    //略過下一指令，繼續執行下一個i
        printf("%d\n",i);
    }
    printf("when out of loop, i=%d",i);
    return 0;
}
```

12. 程式若執行到 goto，則會略過 goto 下面的指令區塊 2，跳躍至（標籤名稱）所在的位置，繼續執行標籤以下的指令區塊。

```
int main(){
    int i, sum=0;
    for(i=1; i<=6; i++) {
        printf("%d\n",i);
        if(i==4)
            goto aa; //強制離開迴圈
        printf("%d\n",i);
    }
    aa:
    printf("when out of loop, i=%d",i);
    return 0;
}
```

13. 指令區塊內可以放置任何合法的指令，當然也可含 for。for 內有 for，稱為巢狀迴圈（Nested Loop）。例如，以下指令可印出 1 至 10 五次。

```
int main(){
    for (int i=1;i<=5;i++){
        for (int j=1;j<=10;j++)
            printf("%d",j);
        printf("\n");
    }
    return 0;
}
```

14. 運算式只要合理就好，不一定要用計數變數。例如：

```
int main(){
    int i=0,s=0;
    for (i=1;s<=10;i++){
        printf("%d",i);
        s=s+i;
    }
    printf("%d",i);
    printf("%d",s);
    return 0;
}
```

```
int main(){
    int i=0,s=0;
    for (i=1;i<=10 && s<=10;i++)
    {
        printf("%d",i);
        s=s+i;
    }
    printf("%d",i);
    printf("%d",s);
    return 0;
}
```


15. 計數變量若用浮點數，因為浮點數會有誤差，請鍵入以下程式，並寫出執行結果。

```
int main(){
    for (double i=1;i<=2;i=i+0.1){
        printf("%lf",i);
    }
    return 0;
}
```

範例8-1a

認識鍵盤遊戲。請寫一個程式，電腦可以連續出現 10 個字元，每出現一個字元，都由使用者輸入，並判斷是否正確，統計正確與錯誤題數。

程式列印

前面我們已經完成產生 1 個字元，使用者鍵入此字元，電腦判斷是否正確，現在要連續出現 10 個字元，只要加上 1 個 for 迴圈就可以。

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>//本例使用time() 函式
int main()
{
    srand(time(NULL));
    int r=0,w=0;
    for (int i=1;i<=10;i++){
        int a=97+rand()%26;
        printf("%c", (char)a);
        char b;
        scanf(" %c",&b);//有點bug,線上查詢後,得知要在控制字元「%」前加空白
        if (b==(char)a){
            printf("right\n");
            r++;
        }
        else{
            printf("wrong\n");
            w++;
        }
    }
}
```

```
    }  
    printf("right=%d\n",r);  
    printf("wrong=%d\n",w);  
    return 0;  
}
```

自我練習

1. 請產生 6 個 -3~3 的亂數，統計正數、0、負數的個數。
2. 請產生 10 個 1~6 的亂數，統計偶數與奇數個數。
3. 請產生 10 組 (x,y) 座標亂數，(x,y) 都介於 -2 與 2 的整數，並統計落在原點、x 軸、y 軸、四個象限的個數。
4. 請產生 10 個 1~6 的亂數，統計數字「3」出現的次數。
5. 請產生 10 個 1~6 的亂數，統計所有數字出現的次數。
6. 請產生 10 個 1~6 的亂數，統計哪一個數字出現的次數最多。
7. 心算練習。請連續產生 8 題兩個 1 位數，由使用者輸入相加結果，電腦回應對或錯，並統計正確與錯誤的題數。
8. 音感練習。請連續發出 10 個 Do 到高音 Do，由使用者輸入 1~8，電腦回應對或錯，並統計正確與錯誤的題數。
9. 請由電腦自動產生 30 個大於等於 0 且小於等於 100 的亂數 x，統計 0~59,60~69,70~79,80~89,90~100 的個數。
10. 請寫一個程式，電腦可以連續出現 50 個小寫字元，電腦統計出現母音的個數。

範例8-1b

試寫一程式，可以輸入 2 至 9 的整數，並輸出此數的九九乘法表。

執行結果

```
input a integer between 2 to 9: 6  
6 * 2= 12  
6 * 3= 18  
6 * 4= 24  
6 * 5= 30  
6 * 6= 36  
6 * 7= 42  
6 * 8= 48  
6 * 9= 54
```

👉 程式列印

每個數字都要使用 2~9 去乘，這種重複的事件，即為 for 迴圈的應用，參考程式如下：

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int a;
    printf("input a integer between 2 to 9: ");
    scanf("%d",&a); //&
    for(int i=2; i<=9; i++)
        printf("%d * %d= %d\n",a,i,a*i);
    return 0;
}
```

範例8-1c

求解輸入數值是否質數。

👉 設計步驟

1. 寫出演算法。

任一整數，除了 1 和本身外，若沒有任何數可以整除此數，則稱此數為質數。所以，本例使用 2 至該數減一的數一一試除，若無一數可整除，則稱此數為質數。其次，為提高程式執行效率，若找到 1 個可以整除，那此數就不是質數，可以提早離開迴圈。

👉 程式列印

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int a=11;
    _Bool prime=1; //布林型態
    for(int i=2; i<=a-1; i++){
        if (a%i==0) { // 留意==, 不是=
            prime=0;
            break; //找到1個就不是了
        }
    }
}
```

```

    }
    if(prime)
        printf("%d is a prime",a)        ;//1
    else
        printf("%d is not a prime",a);//0
    return 0;
}

```

自我練習

1. 請寫一程式，可以輸入兩數，並判斷其是否互質。（兩數除了 1 以外，沒有任何公因數，稱為互質。）

8-2 while 迴圈

上一節的 for 是用於程式設計階段已知迴圈次數。但有些情況，我們於程式設計階段並不知迴圈的執行次數，此時即可使用 while 指令。且有些迴圈可能一次都不執行，所以 while 指令又分為前測試迴圈（Pre-test loop）與後測試迴圈（Post-test loop）。while 的前測試迴圈語法如下圖左，後測試迴圈如下圖右。

<pre> while(運算式){ 指令區塊； break; continue; } </pre>	<pre> do{ 指令區塊； break; continue; }while (運算式)； </pre>
---	---

以上語法說明如下：

1. 不論是前測試迴圈或後測試迴圈，均是運算式值為 1（真）時，繼續執行迴圈，運算式為 0（偽）時，離開迴圈。
2. 前測試與後測試迴圈的差別為，前測試迴圈有可能一次均不執行迴圈，但後測試迴圈至少執行一次。
3. 後測試迴圈的 while（運算式）後面要加分號（；），而前測試迴圈的 while 不用加分號。
4. while 指令區塊內亦適用 break 與 continue，前者為強迫提早離開迴圈，後者可略過部份指令，提早進入條件運算式。

5. 以下程式片段可用前測試迴圈解輸出 1 ~ 10。

```
int main(){
    int i=1;
    while ( i<=10){
        printf("%d\n",i);
        i=i+1;
    }
    return 0;
}
```

以下程式片段可用後測試迴圈輸出 1 ~ 10。

```
int main(){
    int i=1;
    do{
        printf("%d\n",i);
        i=i+1;
    } while (i<=10);
    return 0;
}
```

範例8-2a

編譯器的除法運算子。請使用加減法，完成除法運算。

演算法則

兩數相乘時，程式設計階段就知道迴圈執行次數，所以使用 for。但除法就是不知道，只能說，當被除數大於除數時，就連續減去除數，能減去的次數，就是商，剩下的就是餘數。

1. a= 被除數。
2. b= 除數。
3. 商數 q=0。
4. 所謂的商就是被除數 a 共有幾個除數 b，也就是只要 a 大於等於 b，就要執行以下指令：

a=a-b；

q=q+1；

5. 本例以 8 除以 3，實際演練如下：

(1) $a=8$; $b=3$

$a \geq b$ 成立，所以執行迴圈指令

$a=a-b=5$

$q=q+1=1$ (商)

(2) $a=5$; $b=3$

$a \geq b$ 成立，所以執行迴圈指令

$a=a-b=2$

$q=q+1=2$ (商)

(3) $a=2$; $b=3$

$a \geq b$ 不成立，所以離開迴圈

$q=2$ (商)

$a=2$ (餘數)

程式列印

1. 本例必須重複很多次，所以適用迴圈，程式才能精簡，迴圈有 for 與 while，本例設計階段不知重複幾次，而是一面減、一面判斷，所以適用 while 迴圈。
2. 本程式僅能使用前測試 while 迴圈，而不能使用後測試迴圈，因為有可能一開始被除數就小於除數。

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int a=8,b=3,q=0;
    //printf("input a dividend: ");          // 輸入被除數
    //scanf("%d",&a); //&
    //printf("\ninput a divisor: ");          // 輸入除數
    //scanf("%d",&b);
    while(a>=b) {
        a=a-b;
        q++;
    }
    printf("quotient = %d\n", q);              // 商數
    printf("remainder= %d\n", a);              // 餘數
    return 0;
}
```

 自我練習

1. 同範例，但可以求到小數點 1 位的實數除法。提示：將被除數先乘以 10，再運算，最後將商再除以 10。

範例8-2b

若有一級數 $s=3+6+9+\cdots$ ，請問加到第 n 項，其和剛好超過 100。

 程式列印

1. 本例於設計階段並不知道迴圈執行次數，所以不適用 for。其次，題目給的條件是累加超過 100，所以可以使用 while (sum <100) 則繼續執行迴圈，繼續累加。

```
#include <stdio.h>
#include <stdlib.h>
int main(){
    int i=3,sum=0,n=0;
    while(sum<100){
        printf("%d\n",i);
        sum=sum+i;
        printf("sum=%d\n",sum);
        n=n+1;//次數
        i=i+3;
    }
    printf("\n");
    printf("%d\n",i-3);
    printf("%d\n",n);
    return 0;
}
```

2. 本例若使用 for，則程式如下：

```
#include <stdio.h>
#include <stdlib.h>
int main(){
    int i=3,sum=0,n=0;
    for (i=3;sum<100;i=i+3){
        printf("%d\n",i);
    }
```

```
        sum=sum+i;
        printf("sum=%d\n",sum);
        n=n+1;//次數
    }
    printf("\n");
    printf("%d\n",i-3);
    printf("%d\n",n);
    return 0;
}
```

自我練習

1. 請問 $3+3+3\cdots$ 共加給次，其和剛好超過 100。
2. 請寫一程式，可以輸入任意個數的整數，當輸入 -1 時結束，計算輸入數字的平均。
3. 請寫一程式，可以輸入一個整數，並將其從個數數逐一輸出，且計算其數字和。例如，輸入 25，輸出 5 2 7。
4. 心算練習。請連續產生兩個 1 位數，由使用者輸入相加結果，電腦回應對或錯，直到錯了才停止，並統計正確題數。
5. 音感練習。請連續發出 10 個 Do 到高音 Do，由使用者輸入 1~8，電腦回應對或錯，直到錯了才停止，並統計正確題數。
6. 密碼。假設密碼為「1234」的整數，每次程式執行時，至多可以錯 2 次，才可以使用以上範例，密碼錯誤 3 次就離開此程式。
7. 請判斷所輸入任意數是否全偶數。例如，2682 為全偶數，但 26814 就不是。
8. 請判斷所輸入任意數是否全遞增。例如，1268 為全遞增，但 12681 就不是。
9. 請寫一個程式，可以連續輸入阿拉伯數字，每次 1 個，當輸入「e」時結束，且組合成一個數字。例如：輸入 4,3,8,e 則輸出 438。

8-3 巢狀迴圈

迴圈中又有迴圈，稱為巢狀迴圈。巢狀迴圈在程式設計的領域非常重要，這種運算思維是電腦的強項，也是初學者最感頭疼的單元，本節將使用若干範例引領學生征服此運算思維。

範例8-3a

請寫一程式，印出如下的九九乘法表。

執行結果

```
1 * 1 = 1 1 * 2 = 2 1 * 3 = 3 1 * 4 = 4 1 * 5 = 5 1 * 6 = 6 1 * 7 = 7 1 * 8 = 8 1 * 9 = 9
2 * 1 = 2 2 * 2 = 4 2 * 3 = 6 2 * 4 = 8 2 * 5 = 10 2 * 6 = 12 2 * 7 = 14 2 * 8 = 16 2 * 9 = 18
3 * 1 = 3 3 * 2 = 6 3 * 3 = 9 3 * 4 = 12 3 * 5 = 15 3 * 6 = 18 3 * 7 = 21 3 * 8 = 24 3 * 9 = 27
4 * 1 = 4 4 * 2 = 8 4 * 3 = 12 4 * 4 = 16 4 * 5 = 20 4 * 6 = 24 4 * 7 = 28 4 * 8 = 32 4 * 9 = 36
5 * 1 = 5 5 * 2 = 10 5 * 3 = 15 5 * 4 = 20 5 * 5 = 25 5 * 6 = 30 5 * 7 = 35 5 * 8 = 40 5 * 9 = 45
6 * 1 = 6 6 * 2 = 12 6 * 3 = 18 6 * 4 = 24 6 * 5 = 30 6 * 6 = 36 6 * 7 = 42 6 * 8 = 48 6 * 9 = 54
7 * 1 = 7 7 * 2 = 14 7 * 3 = 21 7 * 4 = 28 7 * 5 = 35 7 * 6 = 42 7 * 7 = 49 7 * 8 = 56 7 * 9 = 63
8 * 1 = 8 8 * 2 = 16 8 * 3 = 24 8 * 4 = 32 8 * 5 = 40 8 * 6 = 48 8 * 7 = 56 8 * 8 = 64 8 * 9 = 72
9 * 1 = 9 9 * 2 = 18 9 * 3 = 27 9 * 4 = 36 9 * 5 = 45 9 * 6 = 54 9 * 7 = 63 9 * 8 = 72 9 * 9 = 81
```

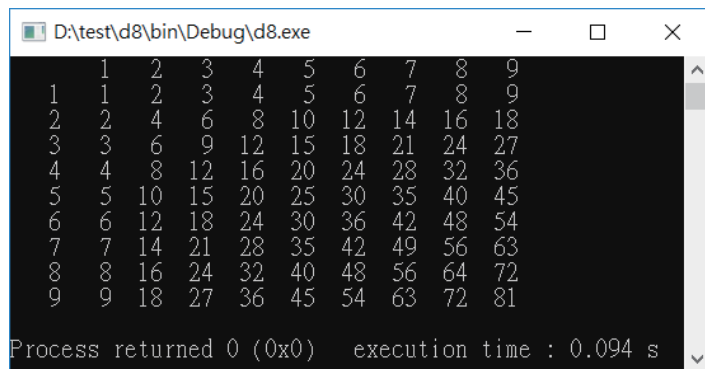
程式列印

前面單元已經介紹如何輸出 1 個數字的九九乘法表，現在只是此一數字也要以迴圈表示，所以雙迴圈程式如下：

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    for (int i=1;i<=9;i++){
        for (int j=1;j<=9;j++){
            printf("%d * %d = %2d ",i,j,i*j);
        }
        printf("\n");
    }
    return 0;
}
```

自我練習

1. 請寫一程式，嘗試使用兩層迴圈印出如下的九九乘法表。

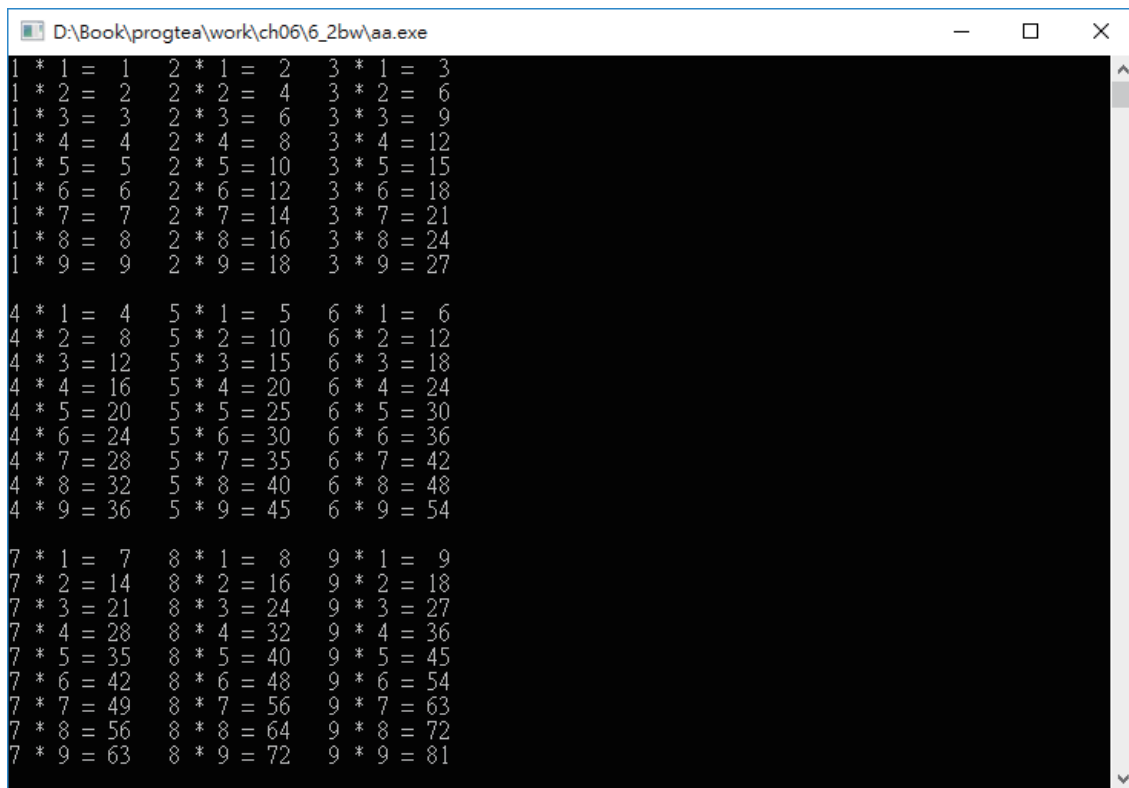


```

D:\test\d8\bin\Debug\d8.exe
1 2 3 4 5 6 7 8 9
1 1 2 3 4 5 6 7 8 9
2 2 4 6 8 10 12 14 16 18
3 3 6 9 12 15 18 21 24 27
4 4 8 12 16 20 24 28 32 36
5 5 10 15 20 25 30 35 40 45
6 6 12 18 24 30 36 42 48 54
7 7 14 21 28 35 42 49 56 63
8 8 16 24 32 40 48 56 64 72
9 9 18 27 36 45 54 63 72 81
Process returned 0 (0x0) execution time : 0.094 s

```

2. 請寫一程式，嘗試使用三層迴圈印出如下的九九乘法表。



```

D:\Book\progtea\work\ch06\6_2bw\aa.exe
1 * 1 = 1 2 * 1 = 2 3 * 1 = 3
1 * 2 = 2 2 * 2 = 4 3 * 2 = 6
1 * 3 = 3 2 * 3 = 6 3 * 3 = 9
1 * 4 = 4 2 * 4 = 8 3 * 4 = 12
1 * 5 = 5 2 * 5 = 10 3 * 5 = 15
1 * 6 = 6 2 * 6 = 12 3 * 6 = 18
1 * 7 = 7 2 * 7 = 14 3 * 7 = 21
1 * 8 = 8 2 * 8 = 16 3 * 8 = 24
1 * 9 = 9 2 * 9 = 18 3 * 9 = 27

4 * 1 = 4 5 * 1 = 5 6 * 1 = 6
4 * 2 = 8 5 * 2 = 10 6 * 2 = 12
4 * 3 = 12 5 * 3 = 15 6 * 3 = 18
4 * 4 = 16 5 * 4 = 20 6 * 4 = 24
4 * 5 = 20 5 * 5 = 25 6 * 5 = 30
4 * 6 = 24 5 * 6 = 30 6 * 6 = 36
4 * 7 = 28 5 * 7 = 35 6 * 7 = 42
4 * 8 = 32 5 * 8 = 40 6 * 8 = 48
4 * 9 = 36 5 * 9 = 45 6 * 9 = 54

7 * 1 = 7 8 * 1 = 8 9 * 1 = 9
7 * 2 = 14 8 * 2 = 16 9 * 2 = 18
7 * 3 = 21 8 * 3 = 24 9 * 3 = 27
7 * 4 = 28 8 * 4 = 32 9 * 4 = 36
7 * 5 = 35 8 * 5 = 40 9 * 5 = 45
7 * 6 = 42 8 * 6 = 48 9 * 6 = 54
7 * 7 = 49 8 * 7 = 56 9 * 7 = 63
7 * 8 = 56 8 * 8 = 64 9 * 8 = 72
7 * 9 = 63 8 * 9 = 72 9 * 9 = 81
Process returned 0 (0x0) execution time : 0.094 s

```

範例8-3b

請寫一程式，印出 2 至 1000 的所有質數。

 運算思維

前面我們已經說明如何判別一個整數是否質數，現在要找 2 至 1000 的所有質數，所以只要再使用一個迴圈就好，參考程式如下：

 程式列印

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    _Bool prime=1; //布林型態，有些編譯器不行
    for(int a=2 ;a<=1000;a++){
        prime=1; //每一次都要預設初值
        for(int i=2;i<=a-1;i++){
            if (a%i==0) { // 留意==, 不是=
                prime=0;
                break; //找到1個就不是了
            }
        }
        if(prime)
            printf("%5d",a); //每個數字佔5格，靠右對齊
    }
    return 0;
}
```

 自我練習

1. 同上範例，但請加上讓每列僅輸出 10 個數字。
2. 請寫一程式，可以計算 $1+(1+2)+(1+2+3)+(1+2+3+4)+\dots+(1+2+\dots+n)$ ， n 可指派為 1 到 100 的自然數。
- ※ 3. 質因數連乘積。請寫一個程式，輸入一正整數，將其質因數分解後印出其式子，例如：

```
輸入：319
輸出：319=11*29
輸入：19
```

輸出：19 =質數

輸入：521752

輸出：521752 = $2^3 * 7^2 * 11^3$

範例8-3c

試寫一程式，找出三位數“阿姆斯壯數”。所謂阿姆斯壯數是指一數等於各個位數的立方和。例如， $153=1^3+5^3+3^3$ 。

👉 程式列印

1. 本例可以使用三個迴圈，程式如下：

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h> //使用pow()
int main() {
    int i, j, k;
    int n=0;          /* 個數 */
    int sum1, sum2;
    for(i=1; i<=9; i++){
        for(j=0; j<=9; j++){
            for(k=0; k<=9; k++) {
                sum1=100*i+10*j+k;
                sum2=(int)(pow(i,3)+pow(j,3)+pow(k,3));
                //pow傳回double，所以要轉型
                if(sum1==sum2){
                    n++;
                    printf("%d : %4d\n",n,sum1);
                }
            }
        }
    }
    return 0;
}
```

2. 也可以使用 1 個迴圈，然後分解數字，程式如下：

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int main() {
```

```

for (int i=100;i<=999;i++){
    int a=i/100;
    int b=(i-a*100)/10;
    int c=i%10;
    int d=pow(a,3)+pow(b,3)+pow(c,3);
    if (i==d )
        printf("%d\n",i);
}
return 0;
}

```

自我練習

1. 試寫一程式，找出四位數“阿姆斯壯數” $abcd=a^4+b^4+c^4+d^4$ 。提示：同範例，但使用四層迴圈。

範例8-3d

請嘗試使用雙迴圈，寫一程式輸出如下：

```

*
**
***
****
*****
*****

```

程式列印

1. 本例的列數、每一列的星星個數，列表解析如下：

列編號	星星個數
1	1
2	2
3	3
4	4
5	5
6	6

2. 由上表可知，共 6 列，所以 i 從 1 到 6。

3. 共有兩個變數，我們就是要找關係，這樣才能簡化問題，找出解答，本例內迴圈 j 與 i 的關係為 $i=j$ ，所以 j 從 1 到 i 。(此一變數找關係的運算思維，往後會持續用到)

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int i, j;
    for(i=1; i<=6; i++) {
        for(j=1; j<=i; j++) {
            printf("*");
        }
        printf("\n");//跳列且歸位
    }
    return 0;
}
```

自我練習

題號	題目	題號	題目
1	<pre> * ** *** **** ***** </pre>	2	<pre> ***** *** *** ** * </pre>
3	<pre> * *** ***** ***** ***** ***** (105APCS試題) </pre>	4	<pre> ***** ***** ** (105APCS試題) </pre>
5	<pre> 1 1 1 1 1 2 2 2 2 3 3 3 4 4 5 </pre>	6	<pre> 5 5 4 5 4 3 5 4 3 2 5 4 3 2 1 </pre>
7	<pre> 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 </pre>	8	<pre> A B B C C C D D D D E E E E E </pre>

8-4 迴圈應用實例

循序法

循序法故名思義就是將所有的值一一列出或一一嘗試，程式語言裡的 for 即可實現此演算法。請看以下範例。

範例8-4a

電腦的乘法運算。

運算思維

1. 人類計算乘法大致如下：

$$\begin{array}{r}
 82 \\
 \times 36 \\
 \hline
 492 \\
 246 \\
 \hline
 2952
 \end{array}$$

2. 但是電腦可不用如此麻煩，不用任何記憶，不用任何學習，電腦的強項是計算能力強，所以就用 for 迴圈實現累加如下：（電腦內部只有累加器與比較器）

```

#include <stdio.h>
#include <stdlib.h>
int main()
{
    int a=82,b=36,sum=0;
    for(int i=1; i<=b; i++)
        sum=sum+a; //累加a
    printf("sum=%d", sum);
    return 0;
}

```

範例 8-4b

等差數列的求和。例如：寫一個程式計算 $1+2+3+\cdots+100$ 的和，或計算 $3+6+9+\cdots+93$ 的和。

 運算思維

本例人類通常使用等差級數求和的公式（首項 + 末項）* 項數 / 2，但電腦可不用如此麻煩、不用額外學習，電腦就直接一個一個逐一累加，程式如下：

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int s=0;
    for (int i=1;i<=100;i++) {
        printf("%d\n",i);
        s=s+i;//逐一將i累加
    }
    printf("\n");
    printf("%d",s);
    return 0;
}
```

 自我練習

1. 請寫一個程式計算 $5+11+17+\cdots+53$ 的和。
2. 請寫一個程式計算 $1+1/2+1/3+\cdots+1/20$ 的和。

 循序搜尋法（Sequential search）

所謂循序搜尋法，就是將所有可能的解一一循序代入，又稱為暴力猜值法。例如，您要求任一數的開平方，因為開平方的結果一定在 0 與此數之間，所以我們就從 0 開始，每次遞增 1、0.1、0.01 或 0.001…，至於是要遞增多少，那就依您要的精密度了。例如，要求整數解，那就遞增 1，要求到小數 1 位，那就遞增 0.1，要求到小數兩位，就遞增 0.01 等等等，請看以下範例說明。

範例8-4c

請用循序猜值法求任意數的平方根。

運算思維

1. 人類求開根號的方法，大致如下：

		3 7 2
1	$9*9=81(>13 \text{ 不行})$ $8*8=64(>13 \text{ 不行})$ $7*7=47(>13 \text{ 不行})$ $6*6=36(>13 \text{ 不行})$ $5*5=25(>13 \text{ 不行})$ $4*4=16(>13 \text{ 不行})$ $3*3=9(\text{沒有 } >13, \text{ 可以})$	1 3 8 3 8 4 9
2	9,8 太大 $3 \times 20 + 7 = 67$ $67 \times 7 = 469$	4 8 3 4 6 9
3	9,8,7,6,5,4,3 太大 $37 \times 20 + 2 = 742$ $742 \times 2 = 1484$	1 4 8 4 1 4 8 4
4		0

以上演算法則在第 9 章習題 4(p9-63) 有詳細說明，可以減少計算，但是電腦計算能力強，所以可使用循序搜尋法或二分搜尋法求解，此即為電腦的運算思維。(二分搜尋法請看範例 8-4e)

程式列印

1. 以下程式每次遞增 1。

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int i, y;
    int a=9;
    for ( i=0;i<=a;i=i+1){
        printf("%d\n",i);
    }
}
```

```

        y=i*i;
        if (y>=a){
            break;
        }
    }
    printf("%d",i);
    return 0;
}

```

2. 以下程式每次遞增 0.1。

```

#include <stdio.h>
#include <stdlib.h>
int main()
{
    double i, y;
    double a=10;
    double d=0.1;
    for ( i=0;i<=a;i=i+d){
        printf("%lf\n",i);
        y=i*i;
        if (y>=a){
            break;
        }
    }
    printf("%lf",i);
    return 0;
}

```

自我練習

1. 請用循序法求解某一正數的立方根。例如，輸入 27 可得到 3.0。本例小數點取 1 位。
2. 假設有一函式 $y=f(x)=x^2-4x-5$ 請分別印出 x 從 -10 到 10 的值。
3. 同上題，請用循序猜值找出其整數解。提示： x 從 -10 到 10 一一代入，找出使函數為零的值，此即為暴力猜值法解題。
4. 同上題，請找出極小值。

5. 函數極值。請寫一程式，可以輸入一個一元二次函式，並求其極大或極小值。例如，輸入 $y=f(x)=x^2-2x+2$ 有極小值 1，輸入 $y=f(x)=-x^2-2x+2$ 有極大值 -1，
6. 假設一個一元多次方程式含有實數解，請寫一程式，可求其解。例如， $y=f(x)=x^2+x-0.75=0$ 的解是 0.5 和 -1.5。提示：浮點運算時，若無法得到 0，此時要使用接近 0 的判斷。例如， $\text{abs}(y)<0.0001$ ，即可視為成立，0.0001 即是其精密度，請換成自己想要的精密度 0.1、0.01 或 0.001。
7. 請用循序法求解任意整數的所有因數。
8. 請寫一個程式，找出 1 至 200 的整數中，找出含有 2 的數字，且統計共含有多少個 2。例如，122 有兩個 2。

8.8 輾轉相除法

以輾轉相除法，求兩數的最大公因數的演算法如下：

1. 輸入 a、b 兩數。
2. $r = \text{餘數}$ 。
3. 將 a 除以 b，得餘數 r，假如餘數不為 0，則以原除數為被除數，原餘數為除數，重複執行 a 除以 b，直到餘數為 0，促使餘數為 0 的除數，即為最大公因數。
4. 本例以 $a=21$ ， $b=9$ ，實際演練如下：
 - (1) $a=21$ ； $b=9$ ；

$$r=a\%b=3$$

$$a=b=9$$

$$b=r=3$$
 $r>0$ 成立，所以繼續執行迴圈。
 - (2) $a=9$ ； $b=3$ ；

$$r=a\%b=0$$

$$a=b=3$$

$$b=r=0$$
 $r>0$ 不成立，所以離開迴圈，最大公因數為 $a=3$ 。

範例8-4d

以輾轉相除法，求兩數的最大公因數。

 程式列印

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int a=9,b=21,t,r;
    do {
        r=a%b;          // 餘數
        a=b;             // 以原除數為被除數
        b=r;             // 以原餘數為除數
    }
    while (r>0);
    //離開迴圈時，最大公因數已經轉交給a
    printf(" The highest common factor:%d",a); // 輸出最大公因數
    return 0;
}
```

 程式說明

1. 輾轉相除法的執行之初並無法預估執行次數，所以較適用 while，且本例迴圈至少執行一次，所以用後測試迴圈。
2. 後測試迴圈的 while 之後必須加分號(;)，以免產生錯誤的訊息。
3. 此題也可寫成前測試迴圈，程式如下，此為 APCS105 試題。

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int a=76,b=48,r;
    while ((a% b) !=0){
        r=a%b;
        a=b;
        b=r;
    }
    printf("%d",b);
    return 0;
}
```

4. 因為是前測試，所以離開迴圈時，當時的 b ，就是餘數，就是最大因數。

自我練習

1. 請寫一個程式，滿足以下條件：
 - (1) 可以產生 0 至 5 的亂數。
 - (2) 累加以上亂數。
 - (3) 輸出此亂數與統計其和。
 - (4) 若亂數不為 0，則重複 (1) ~ (3)，否則輸出其和並結束程式。
2. 請寫一個程式，電腦一直產生 1 到 6 的亂數，直到有連續兩個相等為止，並輸出其和。例如，1,3,4,4，答案是 12。
3. 請寫一個程式，電腦一直產生 1 到 6 的亂數，直到有連續三個亂數相同為止，並輸出其和。例如：1,5,3,2,2,2，答案是 9；又例如：4,5,2,2,3,3,3 答案是 13。
4. 請寫一個程式，滿足以下條件：(擲骰子遊戲)
 - (1) 可以產生兩個 1 至 6 的亂數。(提示： $1 + \text{rand}() \% 6$)
 - (2) 累加以上亂數。
 - (3) 輸出此亂數與其和。
 - (4) 若亂數和大於 8，則重複 (1) ~ (3)，直到亂數和小於等於 8，則程式結束。
5. 請寫一程式，可以連續產生兩個 1 ~ 6 的亂數，並輸出此兩個亂數，直到後面的亂數大於前面的亂數。
6. 請寫一程式，可以連續產生 3 個 1 ~ 6 的亂數，並輸出此 3 個亂數，直到有其中兩個亂數相等為止。
7. 請寫一程式，可以連續產生 4 個 1 ~ 6 的亂數，並輸出此 4 個亂數，直到有其中兩個亂數相等為止，並輸出此不相等的數字與和。例如，產生 6,4,5,1 則繼續產生亂數，若亂數為 6,2,1,6 則其和為 3。
- ※ 8. 請寫一猜拳遊戲程式，可以讓人與電腦猜拳，並輸出結果，直到任一方贏 3 次為止。

二分搜尋法

前面的循序法是循序一個一個搜尋，若沒搜尋中，每次僅減少一筆資料，這裡要介紹一個較有效率的搜尋法，稱為二分搜尋法。二分搜尋法是每次搜尋其可能範圍的中間值，若搜尋的太大，則調整搜尋上限為此中間值，表示後半部都刪除；若搜尋的太小，則調整搜尋下限為此中間值，表示前半部都刪除，每次都縮小範圍為一半，所以可提升搜尋效率，請看一下範例。

範例8-4e

猜數字。請您先默想一個 1 ~ 100 的數字，讓電腦猜，但每次要回應太大 '3' 或猜中 '2' 或太小 '1'。

執行結果

以下是我默想 40，逐次回答電腦，電腦搜尋的過程。

```
調整 x1=1,x2=100
I guess 50,please enter 1(太小),2(正確) or 3(太大):3
a=3 ,太大 調整 x1=1,x2=49
I guess 25,please enter 1(太小),2(正確) or 3(太大):1
a=1 ,太小 調整 x1=26,x2=49
I guess 37,please enter 1(太小),2(正確) or 3(太大):1
a=1 ,太小 調整 x1=38,x2=49
I guess 43,please enter 1(太小),2(正確) or 3(太大):3
a=3 ,太大 調整 x1=38,x2=42
I guess 40,please enter 1(太小),2(正確) or 3(太大):
```

演算法

1. 讓電腦搜尋，電腦可從 1,2,3 開始搜尋，此稱為循序搜尋法，若數字是 1，那運氣好，1 次就搜尋到；若數字是 100，那就要搜尋 100 次，所以平均是 50 次才可搜尋到。這就由讀者練習。
2. 本範例要介紹二分搜尋法，那就是每次搜尋其一半。例如，本例下限 $x1=1$ ，上限 $x2=100$ ，每次就搜尋一半 $x=(x1+x2)/2=50$ ，然後您會告知搜尋中、太大或太小。

3. 若是太大，調整上限 $x2=x-1=49$; 若是太小，則調整下限 $x1=x+1=51$ ，然後重複步驟 2 與 3。每次範圍縮小一半，很快就會搜尋到，程式如下：

程式列印

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *argv[]) {
    int x1=1,x2=100,x;
    int a;
    _Bool right=0;//舊版編譯器不行，請改為int right=0;
    do{
        printf(" 調整  x1=%d,x2=%d\n",x1,x2);
        x=(x1+x2)/2;
        printf("\nI guess %d,please enter 1(太小),2(正確) or 3(太大):",x);
        scanf("%d",&a);
        printf("a=%d ",a);
        if (a==2){
            right=1;
            break;
        }
        else {
            if (a==3){
                printf("太大");
                x2=x-1;//調整上限為x-1
            }
            else{//a=1
                printf("太小");
                x1=x+1;//調整下限
            }
        }
    }while( right==0);
    printf(" Bingo, The number is %d",x);
    return 0;
}
```

範例8-4f

請以二分搜法求解一正數的平分根。本例要精密度到小數點以下第二位。

執行結果

```
4.500000
2.250000
3.375000
2.812500
3.093750
2.953125
3.023438
2.988281
3.005859
2.997070
The sqrt of 9.000000 is 2.997070
```

演算法

1. 本例以求 9 的平分根為例。
2. 求解 9 的平方根，其解必在 0 到 9 之間，所以設定下界為 0，上界為 9。首先，先搜尋 4.5，如下圖步驟 1，但 4.5 的平方為 20.25，大於 9，表示搜尋的太大，那就縮小範圍，將上界調為 4.5，持續在 0 與 4.5 之間搜尋值，如下圖步驟 2。第二次就搜尋 2.25，但是 2.25 的平方為 5.025，小於 9，表示搜尋的太小，那就調整下界為 2.25，且持續在 2.25 與 4.5 之間搜尋，如下圖步驟 3，那要搜尋到何時呢？答案是設定一個精密度，例如，您要小數一位，那就是下界與上界之間的距離小於 0.1，若是要小數兩位，那就是下界與上界之間的距離小於 0.01，此時下界或上界的值，就是所要求的答案了。

搜尋步驟	搜尋值內容		
1	x1 0	x 4.5 (太大)	x2 9
2	x1 0	x 2.25 (太小)	x2 4.5
3	x1	x 3.374 (太大)	x2

3. 以上敘述的演算法如下：

- (1) 設求解的正數為 y ，則其平方根必在 $x1=0$ （下界）與 $x2=y$ （上界）之間，搜尋範圍為 $[x1, x2]$ 。
- (2) 首先猜 $x1+x2$ 之和的一半 x 。
- (3) 若所搜尋 x 的平方小於 y ，表示搜尋的太小，並縮小搜尋範圍為 $[x, x2]$ 。
- (4) 若所搜尋的 x 的平方大於 y ，表示搜尋的太大，並縮小搜尋範圍為 $[x1, x]$ 。
- (5) 重覆步驟 (2)、(3)、(4)，直到 $[x1, x2]$ 的範圍小於所要求的精密度（小數兩位 0.01 或小數三位 0.001），此時的 $x1$ 或 $x2$ 即為平方根。

程式列印

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h> //使用fabs()
int main() {
    double y, x, x1, x2, t;
    y=9; x1=0; x2=y;
    do {
        x=(x1+x2)/2;
        t=x*x;
        if (t<y)
            x1=x; //搜尋的太小，調整下限為x
        else
            x2=x; //搜尋的太大，調整上限為x
    }
    while (fabs(x2-x1) > 0.01) ; //只要x2-x1大於0.01就要繼續搜尋
    //fabs() 絕對值
    printf("The sqrt of %f is %f ", y, x);
    return 0;
}
```

自我練習

1. 請將 `fabs()` 改為 `abs()`，並觀察執行結果。
2. 請以二分搜尋法，求解兩數相除的結果。（本例假設除數大於 1 的整數）

3. 請以二分搜尋法，求解一正數的立方根。
4. 請寫一程式，由電腦產生一個 1 到 100 的亂數，由使用者用二分搜尋法猜，電腦應逐次回答太大、太小、或猜中，且回應幾次猜中。
- ※ 5. 請將 10 進位數轉為 N 進位數（本例暫討論 $N \leq 9$ ，其餘 $N \geq 11$ 時，待串列介紹之後再討論）。提示：演算法如下：
 - (1) 若要將十進位的 a 轉為 2 進位，則以數學式表示如下：

$$(a)_{10} = a_0 + a_1 2^1 + a_2 2^2 + a_3 2^3 + a_4 2^4$$

$$= a_0 + 2(a_1 + a_2 2^1 + a_3 2^2 + a_4 2^3) \quad \#a_1 \text{ 以後，提出 } 2$$
 - (2) 上式的 a_0 為 a 除以 2 的餘數， $a_1 + a_2 2^1 + a_3 2^2 + a_4 2^3 \cdots$ 則為 a 除以 2 的整數商，重複上式，直至整數商等於零為止。
 - (3) 最先出爐的餘數應放在 2 進位的最右邊。 $(a)_{10} = (a_n \cdots a_3 a_2 a_1 a_0)_2$

課後習題

一、填充題

（請先追蹤程式寫出執行結果，然後鍵入以下程式，核對自己答案。`#include <stdio.h> #include <stdlib.h>` 請自己補上）

題號	題目	輸出結果
1	<pre>int i, total = 0; for(i = 1; i < 8; i += 2) total += i; printf("%d",total);</pre>	
2	<pre>int y, r, a = 30, b = 42; r = a%b; while(r != 0) { a = b; b = r; r = a%b; } y = b; printf("%d",y);</pre>	

3	<pre> int main() { int k=4; int m=1; for (int i=1;i<=5;i++){ //cout<<k<<" "<<m; for (int j=1;j<=k;j++){ printf(" "); } for (int j=1;j<=m;j++){ printf("*"); } printf("\n"); k=k-1; //m=m+1; m=2*i+1;//修改處 } return 0; } </pre>	
4	<pre> void main () { //APCS int a=0; int b=1; int i, temp, N=12; for (i=2; i<=N; i=i+1) { temp = b; b=a+b ; a = temp; printf ("%d\n", (b)); } } </pre>	
5	<pre> void main () { //APCS int i = 76; int j = 48,k; while ((i % j) != 0) { k=i%j; i=j; j=k; } printf ("%d\n", j); } </pre>	

6	<pre> void main() { //APCS for (int i=0; i<=10; i=i+1) { printf ("%d ", i); i = i + 1; } printf ("\n"); } </pre>	
7	<pre> void main() { //APCS int a=0, n=1; for (int i=1; i<=n; i=i+1) for (int j=i; j<=n; j=j+1) for (int k=1; k<=n; k=k+1) a = a + 1; printf("%d",a); } //請分別寫出n=1~6,10,100的輸出值。 </pre>	
8	<pre> void main() { //APCS105/10 for (int i=0; i<=3; i=i+1) { for (int j=0; j<i; j=j+1) printf(" "); for (int k=6-2*i; k>0 ; k=k-1) printf("*"); printf("\n"); } } </pre>	
9	<pre> void main() { //APCS10510 int i=2, x=3; int N=65536; while (i <= N) { i = i * i * i; x = x + 1; } printf ("%d %d \n", i, x); } </pre>	

10	以下程式片段無法正確列印 20 次的 "Hi!"，請問下列哪一個修正方式仍無法正確列印 20 次的 "Hi!" ？ // APCS10603 <pre>for (int i=0; i<=100; i=i+5) { printf ("%s\n", "Hi!"); }</pre> (A) 需要將 $i \leq 100$ 和 $i = i + 5$ 分別修正為 $i < 20$ 和 $i = i + 1$ (B) 需要將 $i = 0$ 修正為 $i = 5$ (C) 需要將 $i \leq 100$ 修正為 $i < 100$; (D) 需要將 $i = 0$ 和 $i \leq 100$ 分別修正為 $i = 5$ 和 $i < 100$	
11	<pre>int main() { //APCS10603 int x = 0, n = 5; for (int i=1; i<=n; i=i+1) for (int j=1; j<=n; j=j+1) { if ((i+j)==2) x = x + 2; if ((i+j)==3) x = x + 3; if ((i+j)==4) x = x + 4; } printf ("%d\n", x); return 0; }</pre>	

二、程式設計

1. 秘密差（106/10 APCS 試題）

將一個十進位正整數的奇數位數的和稱為 A，偶數位數的和稱為 B，則 A 與 B 的絕對差值 $|A - B|$ 稱為這個正整數的秘密差。例如：263541 的奇數位數的和 $A = 6 + 5 + 1 = 12$ ，偶數位數的和 $B = 2 + 3 + 4 = 9$ ，所以 263541 的秘密差是 $|12 - 9| = 3$ 。給定一個十進位正整數 X，請找出 X 的秘密差。

2. 完全奇數（107/10 APCS 試題）

13311, 13199 稱為完全奇數，13021 就不是。請寫一程式，可以輸入一個整數 N，並求出比此數大的完全奇數 M 與比此數小的完全奇數 P，並求其差的最小值，也就是 $|N - M|$ 與 $|N - P|$ 的最小值。例如，輸入 $N = 13256$ ，則 $M = 13311$ ， $P = 13199$ ，而 $|13256 - 13311| = 55$ ， $|13256 - 13199| = 57$ ，所以輸出 55。

※ 3. 遞增數。APCS 107 試題

345 等稱為遞增數，53、55 等就不是，請寫一個程式，可以求出 1 到指定輸入數字的遞增數。(APCS107 試題)

※ 4. 請用迴圈解 $\int_0^{10} x dx$ ，dx 分別取 1、0.1 與 0.01。※ 5. 請使用迴圈求解 $\sin(x)$ 正半週面積。 $\sin(x)$ 正半週面積 = $\int_0^{\pi} \sin x dx = 2$ 。(sinx 請先查詢 11-1 公用函式，此為高職基本電學一年級內容)

6. e=2.718 的由來

工程與科學的指數與對數計算，通常不是以 2 或 10 為底，而是以 e 為底，我簡單闡述如下：

- (1) 假設借款金額為一元，年利率為 100%，每年複利一次，則一年後本利和為 2 元。

$$1 \times (1+1)^1 = 2$$

- (2) 假設借款金額為一元，年利率為 100%，每月複利一次，則一年後本利和為 2.613。

$$1 \times \left(1 + \frac{1}{12}\right)^{12} = 2.613$$

- (3) 假設借款金額為一元，年利率為 100%，每日複利一次，則一年後本利和為 2.714。

$$1 \times \left(1 + \frac{1}{365}\right)^{365} = 2.714$$

- (4) 假設借款金額為一元，年利率為 100%，每秒複利一次，則一年後本利和為 2.718。

$$1 \times \left(1 + \frac{1}{365 \times 24 \times 60 \times 60}\right)^{365 \times 24 \times 60 \times 60} = 2.718$$

- (5) 細菌的繁殖、電容的充放電等，這些都是數量非常龐大的成長現象，都要用 e 來解釋才能理解。也就是數量非常龐大的物件，若其一年「平均」成長 2 倍，則一年後總數量會成長為 2.718 倍，而不是 2 倍。

- (6) 請根據以上演算法，求出 e 值。

陣列

chapter

09

學習綱要

- ☞ 9-1 一維陣列
- ☞ 9-2 二維陣列
- ☞ 9-3 字元陣列
- ☞ 9-4 結構
- ☞ 9-5 陣列應用實例

學習目標

- ☞ 1. 認識陣列的種類與性質。
- ☞ 2. 能以一維陣列處理資料。
- ☞ 3. 能以二維陣列處理資料。
- ☞ 4. 能以字元陣列處理字串。
- ☞ 5. 能以結構整合不同性質的資料型態。
- ☞ 6. 能實作陣列、字串、結構等應用實例。

前面處理少量的資料，可以個別設定一些單一變數。但是如果資料龐大，例如，若有資料如下：

```
3,8,4,7,2,9
```

要求極大、極小、平均等運算，若還是使用 6 個變數儲存，則程式將會很冗長。本章將會介紹另一種資料結構『陣列』，然後配合上一章的 `for` 或 `while` 迴圈，就可以以非常精簡的程式，處理以上大批資料。其次，陣列還有維度之分，分別是一維、二維、三維等等，要如何選擇維度就要看您資料運算的方式，若只有同批資料的關係，例如，上例就是一維資料陣列就可以。但是，在一個班級的國、英、數、自然、社會等 5 科成績裡，我們不僅要求縱向的各科平均，還要求橫向的每人平均，此時就需要使用二維陣列，請看本章以下說明。

9-1 一維陣列

處理少量的資料，可以為每一筆資料設定一個變數，但若資料數量很多，例如，若有資料如下：

```
3,8,4,7,2,9
```

要求其和、極大、極小呢？是不是指派 6 個變數，當然也可以，但是程式會非常冗長，請看範例 7-3a。為了改善此一問題，就要使用本節的陣列型態了。因為陣列型態，可以使用『陣列索引』配合 `for` 與 `while` 迴圈而簡化程式，此即為本章重點。例如：以上資料就可宣告陣列如下：

```
int a[7];
```

`int` 是陣列資料型態，陣列型態可為 3-2 節任一型態，`a` 是陣列變數名稱，變數名稱也應該遵循 4-1 節變數命名規則。然後就可依序指派這些資料到陣列，程式如下：

```
a[1]=3;a[2]=8;a[3]=4;a[4]=7;a[5]=2;a[6]=9;
```


中括號內的 1,2,3 稱為『陣列索引』，簡稱索引。人類索引編號通常從 1 號開始使用，但電腦卻是從 0 號開始，所以宣告

```
int a[7]
```

其實是使用 0,1,2,3,4,5,6 等 7 個位置。索引 0 號可用可不用，陣列索引從 0 開始，有其獨到用法，若配合迴圈與取餘，可以執行一些無限循環的功能。例如：以下程式，可以讓陣列索引在 0 到 6 之間無限循環。

```
void loop() {
    PORTA=a[i];
    delay(1000);
    i=(i+1) %7;//保障i在0,1,2,3,4,5,6循環
}
```

♂ 陣列的存取

陣列的存取都要靠索引，例如：以下程式可重新指派 a 陣列索引 3 的值。

```
a[3]=1;
```

以下程式可將 a 陣列索引 3 的值指派給變數 b，但請留意兩者資料型態要相符。

```
int b=a[3];
int b;b=a[3];
```

♂ 陣列宣告與初值設定

單一變數可宣告變數的同時就指派初值，陣列也可以。例如：前面陣列的宣告與初值指派，程式可以簡化如下：

```
int a[7]={0,3,8,4,7,2,9};
```

本例索引 0 不用，也要給予 1 個 0，以上資料 3 才會從索引 1 開始放。事實上，陣列宣告時，您可以不用指派陣列的個數，所以以下程式就可以。

```
int a[]={0,3,8,4,7,2,9};
```

但是陣列的大小給的太小也不行。例如，以下程式就不行：

```
int a[3]={0,3,8,4,7,2,9};
```

其次，給大一點當然可以，以下程式，資料僅給索引 0~6，索引 7,8,9 沒有給初值，電腦會自動指派為 0。

```
int a[10]={0,3,8,4,7,2,9};
```

陣列的資料結構的最大優點是，陣列結構可配合迴圈（前面單一變數不行），例如，以下程式配合陣列的索引，即可以以精簡的程式，處理大批數據。

```
int a[]={0,3,8,4,7,2,9};
int sum=0;
for(int i=1; i<=6; i++) {
    a[i]=a[i]*2; //每一個都乘以2
    sum+=a[i]; //相加每一元素
}
```

範例9-1a

設有資料如下

77, 66, 99, 44, 55

請寫一個程式，可以計算平均、最高分及最低分。

程式列印

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int a[] = { 77, 66, 99, 44, 55 }; //資料用陣列儲存,才能用迴圈
    int i, max, min, sum;
    max = a[0]; //先假設第0號為最高分播台主
    min = a[0];
    sum = a[0];
    for (i = 1; i <= 4; i++) { //使用迴圈
        sum += a[i];
    }
```

```

        if (a[i]>max)//每個人逐一與擂台主比較,若大於則取代
            max = a[i];
        if (a[i]<min)
            min = a[i];
    }
    printf( "Average:%d\n", sum / 5);
    printf("max      :%d\n" , max );
    printf( "min      :%d\n" , min);
    return 0;
}

```

自我練習

1. 寫出以下程式執行結果。

```

int main( )//統測111
{
    int Sum = 0, Array[4] = {1, 2, 3}, i;
    for( i = 1; i < 4; i++ )
        Sum += Array[i];
    printf( "%d", Sum );
    return 0;
}

```

2. 設有資料如下：

-1, 0, 3, 4, -5, 8

- (1) 統計正數、0、負數的個數。
- (2) 請問您的資料結構為何？

3. 設有資料如下：

1,3,2,6,4,2,1,3,4,5

- (1) 統計共有多少個 3。
- (2) 統計 1 到 6 各幾個。

4. 設有資料如下：

11,33,22,66,44,22,11,33,44,55,88,99,77,66

統計與輸出 0~9,10~19,20~29,30~39,40~49,50~59,60~69,70~79,80~89,90~100 等區間的人數。

範例9-1b

請用讀入 5 位學生的國文成績，並可由使用者輸入座號來查詢成績。

執行結果

```
input the grade of number 1:22
input the grade of number 2:33
input the grade of number 3:55
input the grade of number 4:88
input the grade of number 5:99
a[1]=22a[2]=33a[3]=55a[4]=88a[5]=99
input a number:3
a[3]=55
input a number:
```

資料結構

本例也是可使用單獨變數讀入 5 位學生的國文成績，但是程式卻非常冗長，也是選用一維陣列，才可配合迴圈，簡化程式的撰寫。

程式列印

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int a[7];
    int c, i;
    /* 成績輸入 */
    for(i=1; i<=5; i++) {
        printf("input the grade of number %d:",i);
        scanf("%d",&a[i]); //留意&
    }
    //輸出索引與值
    for(i=1; i<=5; i++)
        printf("a[%d]=%d",i,a[i]);
    printf("\n");
    printf("input a number:");
    scanf("%d",&c); //留意&
```

```

while (c!=0) {
    printf("a[%d]=%d\n",c,a[c]);
    printf("input a number:");
    scanf("%d",&c); //留意&
}
return 0;
}

```

👉 自我練習

1. 假設班上共有 12 人，請寫一程式，可以由使用者自行輸入座號，電腦並隨時輸出共有哪些人未到。(提示：可用陣列的值表示到與未到，例如陣列值 1 代表到，0 代表未到)

🔗 查表

寫程式有時候會用到查表問題，例如，十進位轉為十六進位時，當除以 16 時，其餘數的對應值如下表：

餘數	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
對應值	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

此時即可使用陣列。

範例9-1c

請寫一程式，可將任意十進位數轉為 N 進位數。(2≤N≤16)

👉 執行結果

```
the 16 base of 255 value is :      FF
```

👉 演算法

這個題目的演算法在範例 8-2a 的自我練習，當時就討論到，其餘數出現的位置應該是從 0,1,2,3 開始放，所以本例就以陣列的索引 0,1,2,3 開始放置。

👉 程式列印

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    char c[]={'0','1','2','3','4','5','6','7','8','9','A','B',
             'C','D','E','F'};
    char d[20];
    int a, b, r, n=0;
    a=255;b=16 ;
    int i;
    for ( i=20 ;i>=0 ;i--)
        d[i]=' '; //協助清空記憶體
    n=0;//索引從0開始
    int a1=a;
    while(a>0) {
        r=a%b;
        d[n]=c[r]; //將餘數轉為字元，且從索引0開始放
        n++;
        a=a/b;
    }
    printf("the %d base of %d value is : ",b,a1);
    for (i=20 ;i>=0 ;i--)
        printf("%c",d[i]);
    return 0;
}
```

👉 自我練習

1. 假設有一個大富翁遊戲，共有 10 個位置，每個位置的獎罰如下表，遊戲一開始假設有 10000 元，請寫程式讓使用者玩。本例假設使用 1 顆骰子決定每次前進的步數。

0	1	2	3	4	5	6	7	8	9
中發票 得 200 元	闖紅燈 罰 200 元	中樂透 得 10000 元	任意停 車，罰 1000 元	繳汽車 稅 2000 元	繳房屋 稅 3000 元	停玩一 次	得到股 息 1000 元	抽獎得 到 3000 元	繳停車 費 30 元

2. 假設 2 月永遠是 28 天，請寫一程式，可以輸入一個日期，且轉為今年第幾天。例如，輸入 2,1 代表 2 月 1 日，那就是第 32 天，輸出 32。
3. 同上題，可以輸入兩個日期，計算其相差的天數。例如，輸入

```
2018 9 30
2018 10 1
```

則輸出 1。（此為國內高中生程式解題系統 (<https://zerojudge.tw/>) 基礎題。

提示：這題要先界定年份的最小值當成一個參考起點，本例就設為 2018 1 1，然後大家都先求到起點的日期，再相減就可以。

4. 同上題，但可比較兩個日期的先後。例如，應該還書日期是 2018 9 30，但是 2018 10 1 來還，表示逾期。

提示：這題要先界定年份的最小值當成一個參考起點，本例就設為 2018 1 1，然後大家都先求到起點的日期，再比較大小就可以。

5. 前面範例 7-2b 使用 switch case 製作電子琴，請使用陣列重做電子琴。
6. 英文打字練習。假設有單字如下：

```
look、at、one、two、tree、and、day、book、from、go
```

請寫一個程式，可以出現 10 題英文單字，讓使用者可鍵入此單字，並評判正確與錯誤題數。

範例 9-1d

可以輸入天數，但轉為日期輸出，例如輸入 32，能得到 2 月 1 日。（本例先簡化題目，假設 2 月通通 28 天）

執行結果

```
d=365 is m=12,d=31
```

演算法

1. 將每個月的天數以陣列儲存。
2. 從 1 月開始，只要天數大於每月天數，就扣掉，所以程式如下：

 程式列印

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int a[]={0,31,28,31,30,31,30,31,31,30,31,30,31};
    int d=365;
    int m=1;
    int d1=d;
    while (d>a[m]){
        d=d-a[m];
        m++;
    }
    printf("d=%d is m=%d,d=%d",d1,m,d);
    return 0;
}
```

 自我練習

1. 假設 2020/1/1 是星期三，請寫一程式，可以輸入任一日期，而得到星期幾。
2. 假設有兩筆資料如下：

```
3,4,7,9,12
1,5,6,8,15,
```

請分別以陣列儲存，再合併放到第 3 個陣列。

- ※ 3. 假設有兩筆資料如下：（每筆資料都已經是由小而大排列）

```
3,4,7,9,12
1,5,6,8,15,16,17,18
```

請分別以陣列儲存，再合併放到第 3 個陣列，且合併後還是由小而大。

- ※ 4. 假如有兩筆資料如下

```
3,1,4,5,6,7
5,1,7,4,3
```

- (1) 請以兩個陣列儲存。

(2) 請比較與分析有哪些資料重複，將重複的資料放到第三個陣列。(交集)

(3) 將兩筆資料合併，放到第四個陣列，但資料不可重複。(聯集)

※ 5. 假如有資料如下

```
3, 1, 4, 5, 6, 7, 1, 1, 3
```

請將重複的資料去掉，放到另一陣列。

9-2 多維陣列

含有兩個或兩個以上索引值的陣列稱為多維陣列，一般最常應用的是二維陣列，下面我們就以二維陣列為例，來介紹如何使用多維陣列。

⚠ 多維陣列變數宣告

資料型態 陣列變數名稱 [陣列的大小] [陣列的大小] . . . ;

以下敘述宣告一個二維的整數型態陣列變數。

```
int    a[3][5];
```

你可以把二維陣列想像成一個表格，所以上式 **a** 這個陣列就像一個有 3 列 5 行可以存放 15 個元素的表格。和一維陣列一樣，不管是列或行，其索引值皆由 0 開始。為了讓讀者更清楚二維陣列的資料結構，我們將 **a** 陣列的所有元素以表格表示如下：

	第0行	第1行	第2行	第3行	第4行
第 0 列	a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]
第 1 列	a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]
第 2 列	a[2][0]	a[2][1]	a[2][2]	a[2][3]	a[2][4]

🔗 多維陣列變數值的存取

以下敘述可以將第 0 列第 1 行這個陣列元素的值設定為 1。

```
a[0][1]=1;
```

以下敘述可以將第 0 列第 1 行這個陣列元素的值指定給變數 b。

```
b= a[0][1];
```

🔗 多維陣列變數值的初始化

假設有表格資料如下：

10	11	12	13	14
20	21	22	23	24
30	31	32	33	34

若以二維陣列儲存，則其敘述如下：

```
int a[3][5]={ {10, 11, 12,13, 14},
               {20, 21, 22, 23, 24},
               {30, 31, 32, 33, 34} };
```

以上敘述示範了二維陣列變數值的初始化，我們以下表顯示每個陣列元素的初始值。

	第0行	第1行	第2行	第3行	第4行
第 0 列	10	11	12	13	14
第 1 列	20	21	22	23	24
第 2 列	30	31	32	33	34

範例9-2a

假設有兩個矩陣如下：

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \quad \begin{pmatrix} 1 & 2 & 0 \\ 2 & 1 & 3 \end{pmatrix}$$

(1) 請求其相加的結果如下：

$$\begin{pmatrix} 2 & 4 & 3 \\ 6 & 6 & 9 \end{pmatrix}$$

(2) 請問要使用何種資料結構較省事。

 輸出結果

```
Matrix A
1 2 3
4 5 6
Matrix B
1 2 0
2 1 3
Matrix C
2 4 3
6 6 9
```

 資料結構

1. 本例可用 18 個變數儲存，但程式卻很冗長。
2. 本例也可使用一維陣列，但程式還是很冗長，若使用二維陣列，且配合雙層迴圈，則程式就簡潔了。

 程式列印

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int i, j, k;
    const int m=2;
    const int n=3;
    int a[][3]={1,2,3},{4,5,6};
    int b[][3]={1,2,0},{2,1,3};
    int c[2][3];
    printf("Matrix A\n");
    for(i=0; i<m; i++) {
```

```

        for(j=0; j<n; j++){
            printf("%4d",a[i][j]);
        }
        printf("\n");
    }
    printf("Matrix B\n");
    for(i=0; i<m; i++) {
        for(j=0; j<n; j++){
            printf("%4d",b[i][j]);
        }
        printf("\n");
    }
    /* A + B */
    for(i=0; i<m; i++)
        for(j=0; j<n; j++)
            c[i][j]=a[i][j]+b[i][j];
    printf("Matrix C\n");
    for(i=0; i<m; i++) {
        for(j=0; j<n; j++){
            printf("%4d",c[i][j]);
        }
        printf("\n");
    }
    return 0;
}

```

自我練習

1. 假設有矩陣如下：

$$\begin{pmatrix} 1 & 2 & 0 \\ 0 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 0 \end{pmatrix}$$

- (1) 請檢查幾個 0。
- (2) 求所有元素和。
- (3) 再將每一元素乘以 2 後輸出。
- (4) 請問以何種資料結構才會省事。

範例9-2b

假設有學生成績如下：

座號	國文	英文	數學	平均	不及格科數
1	50	60	70		
2	30	40	50		
3	70	80	90		
4	66	77	88		
5	22	33	44		
平均					

1. 請選用適當資料結構儲存以上資料。
2. 計算每人平均。
3. 統計每人不及格科數。
4. 統計各科平均。

執行結果

```
No chin eng math avg fail
1 50 60 70 60 1
2 30 40 50 40 3
3 70 80 90 80 0
4 66 77 88 77 0
5 22 33 44 33 3
0 47 58 68 58 2
```

程式設計步驟資料結構

1. 本例雖可使用 5 個一維陣列儲存以上資料，這樣只簡化縱向的各科成績計算，但橫向每一位學生的平均計算還是無法使用迴圈。若使用二維陣列，那就可以直向、縱向通通配合迴圈，使程式非常精簡。

```
int a[7][7]={0,0,0,0,0,0},
           {1,50,60,70,0,0},
           {2,30,40,50,0,0},
           {3,70,80,90,0,0},
           {4,66,77,88,0,0},
           {5,22,33,44,0,0},
           {0,0,0,0,0,0}};
const int stdnum=5;
```

2. 以上資料的索引如下表：

0	0	0	0	0	0
1 (1,0)	50 (1,1)	60 (1,2)	70 (1,3)	0 (1,4)	0 (1,5)
2 (2,0)	30 (2,1)	40 (2,2)	50 (2,3)	0 (2,4)	0 (2,5)
3 (3,0)	70 (3,1)	80 (3,2)	90 (3,3)	0 (3,4)	0 (3,5)
4 (4,0)	66 (4,1)	77 (4,2)	88 (4,3)	0 (4,4)	0 (4,5)
5 (5,0)	22 (5,1)	33 (5,2)	44 (5,3)	0 (5,4)	0 (5,5)
0 (6,0)	0 (6,1)	0 (6,2)	0 (6,3)	0 (6,4)	0 (6,5)

3. 計算每個人總分的索引變化如下：

```
a[1][4]+=a[1][1] a[1][4]+=a[1][2] a[1][4]+=a[1][3]
a[2][4]+=a[2][1] a[2][4]+=a[2][2] a[2][4]+=a[2][3]
a[3][4]+=a[3][1] a[3][4]+=a[3][2] a[3][4]+=a[3][3]
a[4][4]+=a[4][1] a[4][4]+=a[4][2] a[4][4]+=a[4][3]
a[5][4]+=a[5][1] a[5][4]+=a[5][2] a[5][4]+=a[5][3]
```

外迴圈是人數，內迴圈是科目，陣列索引是 `a[ 人員 ][ 科目 ]`，所以程式如下：

```
for(j=1; j<=stdnum; j++) {
    for(k=1; k<=3; k++) {
        a[j][4]=a[j][4]+a[j][k];
    }
    a[j][4]=a[j][4]/3;
```

4. 計算各科的索引變化如下：

```
a[6][1]+=a[1][1] a[6][1]+=a[2][1] a[6][1]+=a[3][1] a[6][1]+=a[4][1] a[6][1]+=a[5][1]
a[6][2]+=a[1][2] a[6][2]+=a[2][2] a[6][2]+=a[3][2] a[6][2]+=a[4][2] a[6][2]+=a[5][2]
a[6][3]+=a[1][3] a[6][3]+=a[2][3] a[6][3]+=a[3][3] a[6][3]+=a[4][3] a[6][3]+=a[5][3]
a[6][4]+=a[1][4] a[6][4]+=a[2][4] a[6][4]+=a[3][4] a[6][4]+=a[4][4] a[6][4]+=a[5][4]
```

外迴圈是科目，內迴圈是人數，陣列索引是 `a[ 人員 ][ 科目 ]`，所以程式設計如下：

```
for(j=1; j<=4; j++) {
    for(k=1; k<=stdnum; k++)
        a[stdnum+1][j]=a[stdnum+1][j]+a[k][j];
    a[stdnum+1][j]/=stdnum;
}
```

程式列印

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int a[7][7]={0,0,0,0,0,0,0},
        {1,50,60,70,0,0,0},
        {2,30,40,50,0,0,0},
        {3,70,80,90,0,0,0},
        {4,66,77,88,0,0,0},
        {5,22,33,44,0,0,0},
        {0,0,0,0,0,0,0}};

    int i, j, k;
    const int stdnum=5;
    //每人平均
    for(j=1; j<=stdnum; j++) {
        for(k=1; k<=3; k++) {
            a[j][4]=a[j][4]+a[j][k];
        }
        a[j][4]=a[j][4]/3;
    }
    //各科平均
    for(j=1; j<=4; j++) {
        for(k=1; k<=stdnum; k++)
            a[stdnum+1][j]=a[stdnum+1][j]+a[k][j];
        a[stdnum+1][j]/=stdnum;
    }
    //不及格科數
    for(j=1; j<=stdnum+1; j++)
        for(k=1; k<=3; k++)
            if(a[j][k]<60)
```

```

        a[j][5]++;

//輸出
printf("    No chin    eng math    avg fail\n");
for(i=1; i<=stdnum+1; i++) {
    for(j=0; j<6; j++){
        printf("%5d",a[i][j]);
    }
    printf("\n");
}
return 0;
}

```

自我練習

1. 假設某一公司有三個業務員，其每季的業績如下：

編號	第一季	第二季	第三季	第四季	和	零的個數
1	0	2	4	6		
2	8	0	2	1		
3	7	0	0	9		
和						
零的個數						

- (1) 計算每一個人的業績總和及零的個數。
 - (2) 計算每一季的業績總和及零的個數。
 - (3) 請問資料結構為何？
2. 大部分程式語言都是以列為主的方式儲存陣列。在一個 8x4 的陣列 (array) A 裡，若每個元素需要兩單位的記憶體大小，且若 A[0][0] 的記憶體位址為 108 (十進制表示)，則 A[1][2] 的記憶體位址為何？ (APCS10503 試題)

9-3 字元陣列

在 C 語言中，基本資料型態並無字串資料型態（C++ 才有字串資料型態）。例如：無法使用

```
string a;
```

若要處理字串，則要使用字元陣列或指標，指標請參看 10-3 節。在 C 語言中，字串是利用簡單的一維字元陣列加上一個字串結束字元 '\0'（NULL）來表示。例如：以下列敘述是宣告並初始化變數 a 的初值為字串 "abc"。其次，請注意字串使用雙引號（" "），字元使用單引號（' '），如果誤用則會造成錯誤。

```
char a[]="abc";
```

上式的效果等於

```
char b[]={'a', 'b', 'c', '\0'}; // 將字元單獨放入字元陣列中
```

雖然第一種字串宣告方式中的字串 "abc" 的長度為 3，但是編譯器編譯時會自動在字串末端加上一個結束字元 ('\0') 表示字串的結束。因此，所宣告的 a 字元陣列在記憶體中實際上是占了 4 個字元組。請鍵入以下程式，並觀察執行結果。

```
int main() {
    char a[]="abc";//編譯器會自動放\0
    char b[]={'a', 'b', 'c', '\0'};//編譯器不會自動放'\0'
    char c[]={'a', 'b', 'c' };//編譯器不會自動放'\0'
    printf("%s\n",a);
    printf("%s\n",b);
    printf("%s\n",c);
    return 0;
}
```

☞ 字元與字串

字串既然是字元陣列，所以字元的處理也同陣列索引的處理。例如：

```
int main() {
    char a[]="abc";//編譯器會自動放\0
    printf("%s\n",a);
    printf("%c\n",a[0]);
    printf("%c\n",a[1]);
    printf("%c\n",a[2]);
    printf("%c\n",a[3]);
    char b=a[0];
    printf("%c\n",b);
    return 0;
}
```

☞ 字串複製

字串的複製，並不能使用

```
c=a;//C++才可以
```

一定要使用字元陣列複製函式 strcpy，請參考 11-3 節。

```
strcpy(c,a);
```

👉 自我練習

1. 請鍵入以下程式，寫出執行結果。

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>//使用strcmp,strcpy
int main()
{
    char a[]="Horng";
    char b[10];
    strcpy(b,a);
    printf("%s\n",b);
    return 0;
}
```

2. 若宣告一個字元陣列 `char str[20] = "Hello world!"`; 該陣列 `str[12]` 值為何？(A) 未宣告 (B) `\0` (C) `!` (D) `\n` (APCS10510)

♂ 字串輸入與輸出

字串的輸出入和其他資料型態一樣都可以使用 `printf()`、`scanf()`，唯須注意下列事項：

1. 列印格式需要置換為 `%s`。
2. 使用 `scanf()` 時必須使用取址符號 (`&`)，但輸出時就不用再取址，程式如下：

```
char a[4];
scanf("%s", &a);
printf("%s", a);
```

3. 若所輸入的字串中包含空白字元，則必須以讀入字元 (`getchar()`) 的方式將字串輸入到陣列中，並在陣列末端加入一結束字元 `'\0'`。

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    char c, d[10]; int i=0;
    while ((c=getchar()) != EOF && c!='\n'){
        // 本例假設輸入d="abc xyz"
        d[i]=c;
        i++;
    }
    d[i]='\0';
    printf("%s\n", d);
    return 0;
}
```

4. 請鍵入以下程式，寫出執行結果。

```
int main()
{
    char c, d[10];int i=0;
    scanf("%s", &d); // 本例假設輸入"abc xyz"
    printf("%s\n", d);
    return 0;
}
```

⚡ 字串陣列

若有連續字串如下：

Hand、Mary、洪煥維、陳惠敏、王一平

請問該如何儲存呢？答案是使用字元二維陣列，請鍵入以下程式，並觀察執行結果。

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    char a[][10]={"Hand","Mary","洪煥維","陳惠敏","王一平"};
    printf("%c\n",a[0][0]); //H
    printf("%s\n",a[0]); //Hand
    printf("%c%c\n",a[2][0],a[2][1]); //洪, 中文佔兩個字元
    printf("%s\n",a[2]); //洪煥維
    printf("\n");
    int i,j;
    int n=5;
    // 輸出全部
    for (i=0;i<=n-1;i++){
        printf("%10s",a[i]);
    }
    printf("\n");
}
```

👉 程式說明

1. 字串宣告時，僅可省略最高維度的大小。本例預設字串長度最多為 10，程式如下：

```
char a[][10]={"Hand","Mary","洪煥維","陳惠敏","王一平"};
```

👉 實作

1. 請寫一個程式，可以輸入若干個人名，輸入 0 時，結束輸入，輸出這些人名，且由大而小輸出這些人名。

🔗 二維字串陣列

若有二維資料如下：

姓名	數學	APCS	平均
Tom	90	80	
John	60	90	
Mary	80	60	

請問如何儲存，如何計算平均。以下是 C 語言程式。請鍵入以下程式，並寫出執行結果。

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    char a[][4][10]={{ "Tom", "90", "80", ""},
                     { "John", "60", "90", ""},
                     { "Mary", "80", "60", ""}};
    printf("%c\n",a[0][0][0]); //T
    printf("%c\n",a[0][0][1]); //o
    printf("%s\n",a[0][0]); //Tom
    printf("%s\n",a[0][1]); //90
    printf("%s\n",a[1][0]); //John
    int i,j;
    //輸出
```

```
    for (i=0;i<=2;i++){
        for (j=0;j<=3;j++){
            printf("%4s",a[i][j]);
        }
        printf("\n");
    }
    //計算總分、平均
    int sum=0,avg;
    for (i=0;i<=2;i++){
        sum=0;
        for (j=1;j<=2;j++){
            sum=sum+atoi(a[i][j]); //轉數值，請看11-3
        }
        avg=sum/2;
        itoa(avg,a[i][3],10); //轉字串，請看11-3
    }
    printf("\n");
    //輸出
    for (i=0;i<=2;i++){
        for (j=0;j<=3;j++){
            printf("%4s",a[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

自我練習

1. 請寫一程式，可以輸入以上資料。
2. 同上題，可計算每人不及格科數、每科平均、每科不及格人數、依照平均排序。
3. 同上題，可計算每個人的名次，輸出按照原來的順序排列。
4. 同上題，可輸入姓名，而查得其平均。
5. 鍵入以下程式，並寫出執行結果。

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main()
{
    char a[]="ABCD";
    char b[][10]={"AA1", "BB2", "CC3", "DD4"};
    char c[][2][10]={{ "AA", "90"}, {"BB", "80"}, {"CC", "70"} };
    char d[10];
    strcpy(d,a);
    printf("%c\n",a[0]); //A
    printf("%s\n",a); //ABCD
    printf("%s\n",d); //ABCD
    printf("%c\n",b[0][0]); //A
    printf("%s\n",b[0]); //AA1
    strcpy(d,b[1]);
    printf("%s\n",d); //BB2
    printf("%c\n",c[0][0][0]); //A
    printf("%s\n",c[0][0]); //AA
    strcpy(d,c[0][1]);
    printf("%s\n",d); //90
    return 0;
}

```

9-4 結構

範例9-4a

前面的二維陣列僅可以所有欄位的資料型態通通相同，但是若資料如下，請留意姓名是字串，成績是數值。

座號	姓名	國文	英文	數學	平均	備註
1	aa	80	90	70		
2	bb	20	10	80		
3	cc	90	80	70		
平均						

1. 請用二維陣列儲存以上資料。

2. 計算每人平均。
3. 統計每人不及格科數。
4. 統計各科平均。

執行結果

aa	80	90	70	80
bb	20	10	80	36
cc	90	80	70	80

演算說明

以上資料有數值也有字串，因為陣列的所有欄位的資料型態要求一致，也就是僅能全部是數值，或全部是字串，又因為字串無法以數值表示，所以只能通通以字串儲存。

```
char a[][5][3]={{"aa","80","90","70",""},
                 {"bb","20","10","80",""},
                 {"cc","90","80","70",""}}; //均視為字串
```

但是成績的計算就要先將字串的成績轉為數值，等計算完再轉字串，最後再放回，全部程式如下：

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h> //use strcpy()
#include <stdlib.h> //use atoi()
int main() {
    char a[][5][3]={{"aa","80","90","70",""},
                    {"bb","20","10","80",""},
                    {"cc","90","80","70",""}}; //均視為字串

    //計算總分、平均
    int sum=0, avg;
    int i, j;
    for (i=0; i<=2; i++){
        sum=0;
        for (j=1; j<=3; j++){
            sum=sum+atoi(a[i][j]); //轉數值再計算
        }
    }
}
```



```

        avg=sum/3;
        itoa(avg,a[i][4],10); //計算完再轉字串
    }
    //輸出
    for (i=0;i<=2;i++){
        for (j=0;j<=4;j++){
            printf("%8s",a[i][j]);
            //佔8格，且靠右
        }
        printf("\n");
    }
    return 0;
}

```

♂ 結構(struct)

前一範例通通以字串表示，運算時要轉來轉去，真的很麻煩。另一種變通的辦法，就是使用本節介紹的資料結構 `struct`，本書翻譯為結構。因為一個結構容許包含不同型態的資料，也就是容許包含字串與數值，這樣就解決上例使用陣列的不便，請看以下說明。

♂ 結構型態的宣告

結構型態宣告的語法如下：

```

struct 結構型態名稱 {
    資料型態    欄位名稱1;
    資料型態    欄位名稱2;
    :
    資料型態    欄位名稱N;
};

```

以上語法說明如下：

1. 當宣告結構型態時，必須使用 `struct` 關鍵字。
2. 結構型態名稱的命名規則同識別字命名規則。
3. 在大括弧中的項目是結構的成員，也稱為欄位。欄位的宣告方式同一般變數的宣告方式，必須先宣告資料型態，然後緊跟著欄位名稱。

4. 不同的欄位之間必須以分號（；）隔開。
5. 欄位的資料型態可為前面基本資料型態的任一資料型態。例如，char、int、long、double、陣列等型態。
6. 欄位名稱的命名規則同識別字命名規則。

以下以一個學生的成績記錄為例，來說明以上結構型態的宣告。例如：以下的程式片段可以宣告一個含有姓名、國文、數學及英文成績等欄位的結構型態，而結構型態的名稱為 stuGrade。

```
struct stuGrade
{
    char name[5]; // C 語言字串
    int chi;
    int math;
    int eng;
}; //請留意要分號
```

♂ 結構型態變數的宣告

結構型態一經宣告後，即為一種使用者自訂的資料型態，其用法同基本資料型態。例如：以下敘述宣告變數 student 為使用者自訂的 stuGrade 結構型態。

```
struct stuGrade student;
```

♂ 結構欄位的存取

結構欄位存取的方法是在結構變數與欄位之間加上點（.）運算子。例如：以下的敘述可設定變數 student 的欄位內容。

```
strcpy(student.name, "洪峻維"); // student.name="洪峻維"; 不行
student.chi=90;
student.math=100;
student.eng=80;
```

🔗 結構欄位值的初始化

如同陣列一般，也可以在宣告結構變數時，同時指派結構初始值。例如：以下敘述可設定變數 `student` 的初始值。

```
struct stuGrade student={"aa", // name 欄位
                        90, // chi 欄位
                        100, // math 欄位
                        80 /* eng 欄位 */
}; //請留意，要分號
```

以上說明的全部程式如下：

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct stuGrade
{
    char name[7]; // C 語言字串
    int chi;
    int math;
    int eng;
};
int main()
{
    //struct stuGrade student;
    struct stuGrade student={"aa", // name 欄位
                              80, // chi 欄位
                              90, // math 欄位
                              70 /* eng 欄位 */
    }; //請留意，要分號

    printf("%5s", student.name);
    printf("%4d", student.chi);
    printf("%4d", student.math);
    printf("%4d", student.eng);
    printf("\n");
    strcpy(student.name, "bb");
    student.chi=90;
    student.math=100;
    student.eng=80;
```

```

printf("%5s", student.name);
printf("%4d", student.chi);
printf("%4d", student.math);
printf("%4d", student.eng);
return 0;
}

```

以上程式執行結果如下：

```

aa 80 90 70
bb 90 100 80

```

🔗 陣列型態與結構型態綜合使用

於前面陣列，曾提到陣列型態可為任一資料型態，當然也包含了使用者自訂的結構型態。例如：以下的敘述即宣告陣列變數 a 為 stuGrade 的結構型態。

```
struct stuGrade a[10];
```

當完成以上宣告，變數 a 即擁有 10 筆 stuGrade 型態的陣列資料。如此，便可以設定一號或二號同學的成績紀錄如下：

```

strcpy(a[0].name, " 洪煥維"); // 一號的姓名
a[0].chi=90; // 一號的國文成績
strcpy(a[1].name, " 陳惠敏"); // 二號的姓名
a[1].chi=80; // 二號的國文成績

```

範例9-4b

請以結構重做範例 9-4a。

👉 執行結果

```

aa 70 80 90 80
bb 90 80 70 80
cc 40 50 60 50

```

👉 程式列印

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct stuGrade
{
    char name[7]; // C 語言字串
    int chi;
    int math;
    int eng;
    int avg;
};
int main()
{
    //struct stuGrade a;
    struct stuGrade a[3]={{"aa", 70,80,90,0},
                           {"bb",90,80,70,0},
                           {"cc",40,50,60,0}};
    for (int i=0;i<=2;i++ ){
        a[i].avg=(a[i].chi+a[i].eng+a[i].math)/3;
    }
    for (int i=0;i<=2;i++){
        printf("%5s",a[i].name);
        printf("%4d",a[i].chi);
        printf("%4d",a[i].math);
        printf("%4d",a[i].eng);
        printf("%4d\n",a[i].avg);
    }
    return 0;
}
```

🔗 欄位陣列

前面所介紹的是將陣列變數宣告為結構型態，那結構欄位的資料型態是否可為陣列呢？答案是肯定的。例如：先前 stuGrade 結構型態的 chi、math 及 eng 欄位均為 int，所以 stuGrade 亦可重新定義如下：

```
struct stuGrade
{
    char name[7];
```

```
int score[4];
};
```

其次，宣告陣列變數 a 為 stuGrade 的結構型態如下：

```
struct stuGrade a[10];
```

則此型態的第一個人的成績，可表示如下：

```
strcpy(a[0].name, " 洪竣維"); // 一號的姓名
a[0].score[0]=90; // 一號的第一科成績
a[0].score[1]=70; // 一號的第二科成績
a[0].score[2]=80; // 一號的第三科成績
```

範例9-4c

請以欄位陣列重作範例 9-4b。

執行結果

```
aa 70 80 90 80
bb 90 80 70 80
cc 40 50 60 50
```

程式列印

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct stuGrade
{
    char name[7]; // C 語言字串
    int score[4];
    char memo[5];
};
int main()
{
    //struct stuGrade a;
    struct stuGrade a[3]={{"aa", 70,80,90,0,""},
                           {"bb", 90,80,70,0,""},
                           {"cc", 40,50,60,0,""}};
```

```
for (int i=0;i<=2;i++ ){
    for (int j=0;j<=2;j++){
        a[i].score[3]=a[i].score[3]+a[i].score[j];
    }
    a[i].score[3]=a[i].score[3]/3;
}
for (int i=0;i<=2;i++){
    printf("%5s",a[i].name);
    for (int j=0;j<=3;j++)
        printf("%4d",a[i].score[j]);
    printf("\n");
}
return 0;
}
```

自我練習

1. 同上範例，若平均不及格，則備註欄加上“Fail”。
2. 同上範例，但如何計算每一個人不及格科數。
3. 同上範例，但如何計算每科不及格人數。
4. 關於 C 語言中結構（structures）資料型態的敘述，下列何者錯誤？（統測 111）
 - (A) 結構是一種自行定義的資料型態
 - (B) 宣告結構資料型態的變數時，只能使用已定義的結構名稱宣告
 - (C) 定義結構只能使用不同資料型態的變數（陣列僅能使用相同資料型態，結構可以）
 - (D) 定義結構只能使用 struct 關鍵字

9-5 陣列應用實例

☞ 排序 (Sorting)

資料由大而小或由小而大稱為排序，前面第七章我們已經介紹 3 ~ 5 筆資料的排序，當時我們是使用氣泡排序法。資料量少時，我們還可逐一寫程式完成，倘若有 1000 筆，甚至 10000 筆呢？這時候執行速度就很重要了，所以專家就提出很多演算法改善執行效率，本書僅提供 2 種簡單演算法，分別是氣泡排序法與計數排序法，分別說明如下：

☞ 氣泡排序法 (Bubble Sort)

在一數列當中，從第一筆資料逐一往後兩兩給予比較，若前者大於後者，則兩者交換（本例為由小而大排序，若由大而小排序，則當前者小於後者時，兩者交換），每次的比較與交換均可得該數列的最大值於數列最右邊（末端）。所以若有 N 筆資料進行排序，則應作 N-1 階次的逐一比較，且每階次的逐一比較範圍均逐漸減一個，此即為氣泡排序法 (Bubble Sort)。若有 8、9、7、1、2 五筆資料，則其排序過程如下：

1. 將資料由左而右排列如下：

8 9 7 1 2

2. 共有五筆資料 N=5，共須進行四個階次的逐一比較，每一階次都能將該階次的最大值移至最右邊，四個階次的逐一比較細節說明如下：

- (1) 從第一筆到第五筆兩兩比較，若前者大於後者則兩者交換如下，得最大值 9 於最右邊。9 已是最大值，待會兒不必參與比較，所以下一次只要從第一筆比較至第四筆即可。

i=1	比較前	8 9 7 1 2	
j=1, 2, 3, 4	比較次數	↓ ↓ ↓ ↓	比較四次
	比較後	8 7 1 2 9	

- (2) 從第一筆到第四筆兩兩比較，若前者大於後者，則兩者交換如下，得次大值 8 於右邊。8 與 9 已達定位，待會兒不用參與比較。

i=2	比較前	8 7 1 2 9	
j=1, 2, 3	比較次數	↓ ↓ ↓	比較三次
	比較後	7 1 2 8 9	

- (3) 從第一筆資料到第三筆資料兩兩比較，若前者大於後者則兩者交換如下，得第三大值 7 於右邊。

i=3	比較前	7 1 2 8 9	
j=1, 2	比較次數	↓ ↓	比較二次
	比較後	1 2 7 8 9	

- (4) 從第一筆資料到第二筆資料兩兩比較，若前者大於後者則兩者交換如下，排序完成。

i=4	比較前	1 2 7 8 9	
j=1	比較次數	↓	比較一次
	比較後	1 2 7 8 9	

- 若有 N 筆資料，則應作 $N-1$ 階次的比較與交換，每階次的比較範圍均逐漸縮小 $[N-1, N-2, \dots, 1]$ ，其比較次數總計大約為 $[N*N]/2$ ，資料交換次數大約為 $[N*N]/4$ 。
- 以上每次均找到一個最大值於右邊，若將該次搜尋所得最大值與右邊位置交換，則可大大減少資料交換次數，且其比較次數不變仍為 $[N*N]/2$ ，但其資料交換次數總計約為 N ，請讀者自行練習。

以上氣泡排序法，在第七章已經寫程式完成，當時資料限於 5 筆內，逐一比較與交換尚可克服，但若資料超過 5 筆，程式可就落落長，現在使用迴圈的演算法如下：

- 設有 N 筆資料進行排序。
- 應進行 $N-1$ 階次尋找最大值，並將最大值移至最右邊。

```
for (i=1; i<=N-1; i++)
```

3. 每一階次均應逐次尋找最大值，並移至右邊，且每階次的比較次數 j 與外迴圈 i 的關係為 $i+j=N$ ，所以內迴圈的比較次數 j 為 $N-i$ 。

```
for(j=1; j<=N-i; j++)
```

4. 兩兩逐一比較，若前者大於後者則兩者交換。

```
if(a[j]>a[j+1]) { t=a[j]; a[j]=a[j+1]; a[j+1]=t; }
```

範例9-5a

示範氣泡排序法。

執行結果

```
Before sort:
 99  88  77  66  55
After sort:
 55  66  77  88  99
```

程式列印

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int a[]={0, 99, 88, 77, 66, 55};
    //所有排序的測資，要用條件最差的情況，才能評估是否正確

    int i, j, t;
    const int N=5;
    printf("Before sort:\n");
    for(i=1; i<=N; i++)
        printf("%5d", a[i]);
    //排序
    for(i=1; i<=N-1; i++)
        for(j=1; j<=N-i; j++)
            if(a[j]>a[j+1]){
                t=a[j]; //交換
                a[j]=a[j+1];
                a[j+1]=t;
            }

    //輸出
    printf("\nAfter sort:  \n");
```

```

    for(i=1; i<=N; i++)
        printf("%5d" ,a[i]);
    return 0;
}

```

👉 自我練習

1. 以上範例，沒用索引 0。若資料從索引 0 開始存放，請自己撰寫程式。
2. 同上範例，但每階次均尋找一個最大值，且逐次與第 $n, n-1, n-2$ 筆資料交換即可。

範例9-5b

假設有資料如下：

座號	國文	數學	總分
1	10	10	20
2	9	9	18
3	9	10	19
4	10	9	19
5	9	8	17

- (1) 請以適當資料結構儲存。
- (2) 以總分為依據，從小而大輸出。

👉 執行結果

```

5  9  8 17
2  9  9 18
3  9 10 19
4 10  9 19
1 10 10 20

```

👉 程式列印

本例選用二維資料陣列 `a[][4]` 如下，這樣才能配合迴圈，簡化程式的撰寫。

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int a[][4]={0,0,0,0},
              {1,10,10,20},
              {2,9,9,18},
              {3,9,10,19},
              {4,10,9,19},
              {5,9,8,17}};

    int i,j,k,t;
    const int N=5;
    for(i=1;i<=N-1;i++){
        for (j=1;j<=N-i;j++){
            if(a[j][3]>a[j+1][3]){
                //座號，成績
                //每一個都要換
                for (k=0;k<=3;k++){
                    t=a[j][k];
                    a[j][k]=a[j+1][k];
                    a[j+1][k]=t;
                }
            }
        }
    }
    for (i=1;i<=N;i++){
        for(j=0;j<=3;j++)
            printf("%4d",a[i][j]);
        printf("\n");
    }
    return 0;
}
```

自我練習

1. 同範例，但總分相同時，請以數學為排序依據。
提示：將（總分 * 10 + 數學）當作排序依據就可以。

2. 假設有資料如下：

座號	姓名	國文	數學	總分
1	aa	8	10	18
2	bb	9	9	18
3	cc	7	10	17
4	dd	10	9	19
5	ee	9	8	17

- (1) 請以適當資料結構儲存。
- (2) 以總分爲依據，從大而小輸出。

♂ 計數排序法 (Counting Sort)

前面的氣泡排序法是採用兩者比較並交換，達到由小而大或由大而小排列的效果，不過本節的計數排序法則是不移動資料，但直接給予名次。爲了充份理解計數排序法的原理，你先假想一個 N 位學生的教室裡，每人均舉著自己的分數，每一學生如何知道自己是第幾名呢？有一個辦法就是每個人都數一數分數比自己高的人數，再將此數量加一，就是自己的名次，但其比較次數是 $N*N$ 。假若令他們排成一排，由第一個逐一往右比較，且分數低的，令其名次加 1，則第二個人就不必與第一個比較。同理，第三個也是往右比較即可，不必與第一、二個人比較，所以其總比較次數大約爲 $[N*N]/2$ ，此即爲計數排序法，若有數列資料 8、2、9、7、1 則其演算過程如下：

1. 先假設每人均爲第一名。

分數	8	2	9	7	1
名次	*	*	*	*	*

2. 共有五筆資料，分別以前四筆資料爲基準往右比較：

- (1) 以第一筆資料 8 為基準，往右（第二、三、四、五筆）比較，分數低的，將其名次加一，則其名次如下，第一筆資料 8 確定為第二名，下次不用再參與比較。

分數	8	2	9	7	1
名次	**	**	*	**	**

- (2) 以第二筆資料 2 為基準，往右（第三、四、五筆）比較，分數低的，將其名次加一，則其名次如下，第二筆資料 2 確定為第四名，下次不用再參與比較。

分數	8	2	9	7	1
名次	**	****	*	**	***

- (3) 以第三筆資料 9 為基準，往右（第四、五筆）比較，分數低的，將其名次加一，則其名次如下，第三筆資料 9 確定為第一名，下次不用參與比較。

分數	8	2	9	7	1
名次	**	****	*	***	*****

- (4) 以第四筆資料 7 為基準，往右（第五筆）比較，分數低的，將其名次加一，則其名次如下，全部名次確定，排序完成。

分數	8	2	9	7	1
名次	**	****	*	***	*****

3. 若有 N 筆資料，則須作 $N-1$ 階次的比較，每階次的比較次數均逐漸縮小，分別為 $N-1$ ， $N-2$ ，.....， 1 ，故其總比較次數約為 $[N*N]/2$ 。

範例9-5c

示範計數排序法。

執行結果

```
grade:
 77  66  99  44  55
Rank:
  2   3   1   5   4
```

程式列印

本例以一維陣列 `a[]` 儲存所有資料，這樣才能配合迴圈，簡化程式的撰寫。

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int a[]={0, 77, 66, 99, 44, 55};
    int b[6];
    int t;
    const int N=5;
    printf("grade: \n");
    for(int i=1; i<=N; i++) {
        printf("%5d",a[i]);
        b[i]=1;//設定每個都是第一名
    }
    /* sort */
    for(int i=1; i<=N-1; i++){
        for(int j=i+1; j<=N; j++){
            if(a[i]>a[j]){
                b[j]++;
            }
            else{
                b[i]++;
            }
        }
    }
    printf("\nRank:  \n");
    for(int i=1; i<=N; i++){
        printf("%5d",b[i]);
    }
    return 0;
}
```

範例9-5d

假設有資料如下：

座號	姓名	國文	數學	總分	名次
1	aa	10	10	20	
2	bb	9	9	18	
3	cc	7	10	17	
4	dd	10	9	19	
5	ee	9	7	16	

- (1) 請以適當資料結構儲存。
- (2) 以總分為依據，計算其名次，輸出以座號小者先輸出。

 輸出結果

```
1 aa 10 10 20 1
2 bb 9 9 18 3
3 cc 7 10 17 4
4 dd 10 9 19 2
5 ee 9 7 16 5
```

 程式列印

1. 本例每筆資料有字串也有數值，所以應該選用「結構」儲存資料。

```
#include <stdio.h>
#include <stdlib.h>
struct stuGrade{
    int no;
    char name[3]; // C 語言字串
    int chi;
    int math;
    int total;
    int order;
};
int main() {
    const int N=5;
    struct stuGrade a[6]={0,"",0,0,0,0},
                        {1,"aa", 10,10,20,0},
                        {2,"bb",9,9,18,0},
```



```

                                {3,"cc",7,10,17,0},
                                {4,"dd",10,9,19,0},
                                {5,"ee",9,7,16,0}};

for (int i=1;i<=N;i++)
    a[i].order=1;//設定大家都是第1名
for(int i=1; i<=N-1; i++){
    for(int j=i+1; j<=N; j++){
        if(a[i].total>a[j].total){
            a[j].order++;//分數少的名次加1
        }
        else{
            a[i].order++;
        }
    }
}
for(int i=1; i<=N; i++){
    printf("%3d %4s %3d %3d %3d %3d",a[i].no,a[i].name,a[i].chi,
        a[i].math,a[i].total,a[i].order);
    printf("\n");
}
return 0;
}

```

自我練習

1. 同範例 9-5d，但是總分相同時，以數學為排序依據。

搜尋 (Search)

我們日常生活總是不斷的搜尋，在家裡找東西、到圖書館找書、到超商買東西，都是在搜尋。本單元將介紹若干資料搜尋的方法，分別是循序搜尋與二分搜尋法，分別說明如下：

循序搜尋法 (Sequential Search)

於序列中，從頭到尾逐一比對搜尋的方式，稱為循序搜尋法，其次，若是資料沒有按照某一規則存放，那就只能靠循序搜尋法，請看以下範例。

範例9-5e

假設有員工電話如下，請以適當資料結構儲存，並能輸入姓名而查得電話號碼。

編號	姓名	電話號碼
1	aa	1111168
2	hh	2222168
3	cc	3333168
4	gg	4444168
5	ff	5555168
6	ii	6666168
7	ee	7777168
8	bb	8888168
9	jj	9999168
10	dd	1688168
11	kk	2688268
12	ll	3688368

 執行結果

1. 輸入“aa”傳回 aa 的電話號碼。

```
input a name: aa
the phone number is 1111168
```

 程式列印

1. 本例要能配合迴圈，那就要將資料以二維陣列儲存，但是 C 沒有字串型態，所以要用三維的字元陣列 `char a[13][2][8]` 儲存以上資料。
2. 前面範例 8_4c 已經介紹循序法解開根號，本例也是於陣列逐一搜尋，所以程式如下：

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h> //use strcmp()
int main() {
```

```

char a[13][2][8]={{"0", "0"},
                  {"aa", "1111168"},
                  {"hh", "2222168"},
                  {"cc", "3333168"},
                  {"gg", "4444168"},
                  {"ff", "5555168"},
                  {"ii", "6666168"},
                  {"ee", "7777168"},
                  {"bb", "8888168"},
                  {"jj", "9999168"},
                  {"dd", "1688168"},
                  {"kk", "2688168"},
                  {"ll", "3688168"}};
//3688168共7個字元，加上字串結束字元，共8個才夠
int i, found=0;
char name[3];
printf("input a name: ");
scanf("%s", &name);//&
for(i=1; i<=12; i++)    {
    if(strcmp(a[i][0],name)==0)    {
        found=1;
        break;
    }
}
if(found==1)
    printf(" the phone number is %s", a[i][1]);
else
    printf("no data found");
return 0;
}

```

3. 資料結構選「結構」程式的可讀性會提高，程式如下：

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>//use strcmp()
struct customer {
    char name[3];
    char tel[8];
};

```

```

int main() {
    struct customer  a[13]={{"0", "0"},
                             {"aa", "1111168"},
                             {"hh", "2222168"},
                             {"cc", "3333168"},
                             {"gg", "4444168"},
                             {"ff", "5555168"},
                             {"ii", "6666168"},
                             {"ee", "7777168"},
                             {"bb", "8888168"},
                             {"jj", "9999168"},
                             {"dd", "1688168"},
                             {"kk", "2688168"},
                             {"ll", "3688168"}};
    //3688168共7個字元，加上字串結束字元，共8個才夠
    int i, found=0;
    char  name[3];
    printf("input a name: ");
    scanf("%s", &name);//&
    for(i=1; i<=12; i++)    {
        if(strcmp(a[i].name,name)==0) {
            found=1;
            break;
        }
    }
    if(found==1)
        printf(" the phone number is %s", a[i].tel);
    else
        printf("no data found");
    return 0;
}

```

補充說明

1. 字串比較要使用 strcmp() 函式，請看第 11 章。

自我練習

1. 同上題，但可重複查詢。
2. 假如有帳號密碼如下表，只有輸入這些帳號與密碼才能使用範例 2-6a 的面積計算，請寫程式完成此功能。

編號	帳號	密碼
1	aa	1111168
2	hh	2222168
3	cc	3333168
4	gg	4444168
5	ff	5555168
6	ii	6666168
7	ee	7777168
8	bb	8888168
9	jj	9999168
10	dd	1688168
11	kk	2688268
12	ll	3688368

🔗 二分搜尋法 (Binary Search)

前面的循序搜尋法，是採用逐一訪視的方式搜尋所要的資料，但當資料量龐大時，就顯得毫無效率。舉例而言，資料數量若有 1000 筆，則採用循序搜尋法平均搜尋次數為 500 次，但若採用本節所討論的二分搜尋法，則至多不會超過 10 次 ($2^{10}=1024$)。使用二分搜尋法前應先將資料排序。在一個已排序的序列中，首先與序列的中間值比較，若待搜尋資料比中間值大，則待搜尋資料必在此中間值的後面，反之則在前面，所以二分搜尋法每次可將搜尋範圍減半，如此對於資料量龐大時，可提高搜尋效率，請看以下範例：

範例9-5f

同範例 9-5e，但使用二分搜尋法。

👉 執行結果

1. 將陣列以氣泡排序法排序如下。
2. 以下為輸入“dd”的執行結果。

```
aa 1111168
bb 8888168
cc 3333168
dd 1688168
ee 7777168
ff 5555168
gg 4444168
hh 2222168
ii 6666168
jj 9999168
kk 2688168
ll 3688168
input a name: dd
the phone number is 1688168
```

程式列印

1. 本例使用字元陣列表示字串。
2. 資料要先排序，才能使用二分搜尋法。
3. 前面範例 8_4f 已經介紹二分法解開根號，本例也是以陣列索引兩端當上下邊界，逐一縮小範圍，程式如下：
4. 資料若以陣列儲存，程式如下：

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main() {
    char a[13][2][8]={{"0", "0"},
                      {"aa", "1111168"},
                      {"hh", "2222168"},
                      {"cc", "3333168"},
                      {"gg", "4444168"},
                      {"ff", "5555168"},
                      {"ii", "6666168"},
                      {"ee", "7777168"},
                      {"bb", "8888168"},
                      {"jj", "9999168"},
                      {"dd", "1688168"},
                      {"kk", "2688168"},
                      {"ll", "3688168"}};

    int i,j, found=0;
    char t[9];
    const int N=12;
    for(i=1; i<=N-1; i++) {
```

```

        for(j=1; j<=N-i; j++)        {
            if(strcmp(a[j][0], a[j+1][0])>0)        {
                strcpy(t,a[j][0]);
                strcpy(a[j][0],a[j+1][0]);
                strcpy(a[j+1][0],t);
                strcpy(t,a[j][1]);
                strcpy(a[j][1],a[j+1][1]);
                strcpy(a[j+1][1],t);
            }
        }
    }
    for(i=1; i<=12; i++)        {
        for(j=0; j<=1; j++){
            printf("%9s",a[i][j]);
        }
        printf("\n");
    }
    char  name[3];
    printf("input a name: ");
    scanf("%s", &name);//&
    int l=1;//下邊界
    int u=12;//上邊界
    int m;
    while((l<=u) & (found==0))    {
        m=(int)((l+u)/2);
        if(strcmp(name, a[m][0])==0){//#include<string.h>
            found=1;
            break;
        }
        else {
            if(strcmp(name, a[m][0])>0)
                l=m+1;//調整下邊界
            else
                u=m-1;//調整上邊界
        }
    }
    if(found==1)
        printf(" the phone number is %s", a[m][1]);
    else
        printf("no data found");
    return 0;
}

```

5. 資料若以「結構」儲存，程式結構化才會好，程式如下：

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct customer {
    char name[3];
    char tel[8];
};
int main() {
    struct customer a[13]={{"0", "0"},
                            {"aa", "1111168"},
                            {"hh", "2222168"},
                            {"cc", "3333168"},
                            {"gg", "4444168"},
                            {"ff", "5555168"},
                            {"ii", "6666168"},
                            {"ee", "7777168"},
                            {"bb", "8888168"},
                            {"jj", "9999168"},
                            {"dd", "1688168"},
                            {"kk", "2688168"},
                            {"ll", "3688168"}};

    int i,j, found=0;
    struct customer t;
    const int N=12;
    for(i=1; i<=N-1; i++) {
        for(j=1; j<=N-i; j++) {
            if(strcmp(a[j].name, a[j+1].name)>0) {
                //a[j] a[j+1] 交換
                t=a[j];
                a[j]=a[j+1];
                a[j+1]=t;
            }
        }
    }
    for(i=1; i<=12; i++){
        printf("%4s %9s",a[i].name,a[i].tel);
        printf("\n");
    }
}
```



```

char name[3];
printf("input a name: ");
scanf("%s", &name); //&
int l=1; //下邊界
int u=12; //上邊界
int m;
while((l<=u) & (found==0)) {
    m=(int)((l+u)/2);
    if(strcmp(name, a[m].name)==0){ // #include <string.h>
        found=1;
        break;
    }
    else {
        if(strcmp(name, a[m].name)>0)
            l=m+1; //調整下邊界
        else
            u=m-1; //調整上邊界
    }
}
if(found==1)
    printf(" the phone number is %s", a[m].tel);
else
    printf("no data found");
return 0;
}

```

範例9-5g

KTV 音樂伴奏與字幕。我們已經於範例 5_4b 作出音樂伴奏，但卻是一個音一個指令，程式非常冗長，本範例資料結構改用一維陣列，分別儲存音階與音長，然後使用迴圈輸出以上資料，就可以簡化程式。(本例同樣以小毛驢為例)

| 1 1 1 3 | 5 5 5 5 | 6 6 6 1 | 5 - |
 我 有 一 隻 小 毛 驢 我 從 來 也 不 騎

 資料結構

本例一分鐘 80 拍，所以 1 拍的時間是 $60000\text{ms}/80$ ，以一個四分音符為 1 拍，所以上圖一個八分音符所佔的時間是 $60000\text{ms}/(2*80)$ ，每小節是 2 拍。其次，上圖，所有音符的最小時間是八分音符，所以我就取八分音符的時間為 1 個單位，且以陣列兩個一組儲存每一音符的音高與時間如下，以下僅作 4 小節，『我有一隻小毛驢我從來也不騎』：第 1 音『我』半拍就數位化為 1,1；最後 1 個『騎』是 2 拍，就數位化為 5,4。

```
const int a[]={1,1,1,1,1,1,3,1,5,1,5,1,5,1,5,1,6,1,6,1,6,1,8,1,5,4};
```

然後每一個音階就可由查表得知，如以下陣列。

```
const int b[]={0,523,587,659,698,784,880,988,1046};
```

例如，1 就對應 523，2 就對應 587，餘此類推。

 程式列印

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    const int b[]={0,523,587,659,698,784,880,988,1046};
    //小毛驢
    const int s=80;
    const int a[]={1,1,1,1,1,1,3,1,5,1,5,1,5,1,5,1,6,1,6,1,6,1,8,1,5,4};
    const int l=26;
    //蝴蝶
    //const int a[]={1,2,1,2,2,1,3,2,3,2,2,1,1,1,2,1,3,1,1,2,3,
    //                2,3,1,4,1,5,2,5,2,4,1,3,1,4,1,5,1,3,2};
    //const int l=40;
    while(1){
        for (int i=0;i<l/2;i=i+1){
            _beep(b[a[2*i]],a[2*i+1]*60000/(s*2)); //音階,延遲時間
        }
    }
}
```

↑
↑

音階
延遲時間

 自我練習

1. 以上範例使用一維陣列，請以二維陣列改寫。

- ### 範例9-5h

編號	英文	詞性	中文
1	delighted	adj	高興的
2	gear	n	服裝
3	behave	v	表現
4	attract	v	吸引
5	criticism	n	批評
6	concerned	adj	擔心的

👉 執行結果

 程式列印

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h> //use strcmp()
int main() {
    char b[100];
    int c=0,d=0;
    //將題目用陣列儲存，等待學習檔案後，可將這些資料放在檔案
    char a[14][3][100]={{"","",""},
        {"delighted", "adj", "高興的"},
        {"gear", "n", "服裝"},
        {"behave", "v", "表現"},

```

```
    {"attract", "v", "吸引"},
    {"criticism", "n", "批評"},
    {"concerned", "adj", "擔心的"},
    {"There are four people in my family.", " ", "我們家有四個人"},
    {"good", "adj", "好的"},
    {"apple", "n", "蘋果"},
    {"look", "v", "看見"},
    {"book", "n", "書"},
    {"television", "n", "電視"},
    {"cellphone", "n", "手機"};
int i, j=0;
char e;
for ( i=1; i<=10; i++){
    printf("%s,%s :", a[i][1], a[i][2]); //輸出詞性, 中文
    j=0;
    while ((e=getchar( )) !='\n'){ //可輸入含空白字串
        b[j]=e;
        j++;
    }
    b[j]='\0'; //記得自己要放字串結尾
    if (strcmp(a[i][0], b)==0){ //與題目比對
        printf("Right\n");
        c++;
    }
    else{
        printf("Wrong\n");
        d++;
    }
}
printf("The Right number is :%d\n", c);
printf("The wrong number is :%d\n", d);
return 0;
}
```

 自我練習

1. 線上選擇題測驗。若有英文單字如下：請寫一程式，可以讓使用者使用選擇題測驗。

編號	單字	類別	答案	選項1	選項2	選項3	選項4
1	book	n	2	學校	書	老師	校長
2	delighted	adj	1	高興的	悲傷的	生氣的	憤怒的
3	gear	n	3	背包	褲子	服裝	鞋子
4	behave	v	2	生氣	表現難過	看到	
5	attract	v	4	打架	斥責	排斥	吸引

課後習題

一、填充題

題號	題目	輸出結果
1	<pre> int main() { //APCS10503 int a[100], b[100]; for (int i=1; i<=100; i=i+1) { b[i] = i; } a[0] = 0; for (int i=1; i<=100; i=i+1) { a[i] = b[i] + a[i-1]; } printf ("%d\n", a[50]-a[30]); return 0; } </pre>	
2	<pre> int main() { //APCS10503 int A[5], B[5], i, c; for (i=1; i<=4; i=i+1) { A[i] = 2 + i*4; B[i] = i*5; } c = 0; for (i=1; i<=4; i=i+1) { if (B[i] > A[i]) { c = c + (B[i] % A[i]); } else { c = 1; } } printf ("%d\n", c); return 0; } </pre>	

3	定義 $a[n]$ 為一陣列 (array)，陣列元素的指標為 0 至 $n-1$。若要將陣列中 $a[0]$ 的元素移到 $a[n-1]$，底下程式片段空白處該填入何運算式？(A) $n+1$ (B) n (C) $n-1$ (D) $n-2$ <pre> int i, hold, n; for (i=0; i<= ; i=i+1) { hold = a[i]; a[i] = a[i+1]; a[i+1] = hold; } </pre>	
4	<pre> #define TRUE 1 #define FALSE 0 int main() { int val, allBig; //int d[]={0,11,12,13,14,15};// //val=3; int d[]={0,11,12,13,14,25};// val=20; //int d[]={0,23,15,18,20,11};// //val=12; //int d[]={0,18,17,19,24,15};// //val=16; //for (int i=1; i<=5; i=i+1) { // scanf ("%d", &d[i]); //} //scanf ("%d", &val); allBig = TRUE; for (int i=1; i<=5; i=i+1) { if (d[i]>val) { allBig=TRUE;//繼續找，全部大，才是AllBig } else { allBig=FALSE; break;//加這行，找到一個就離開 } } if (allBig==TRUE) { printf("%d is the smallest.\n", val); } else { printf ("%d is not the smallest.\n", val); } return 0; } </pre>	

5	以下程式片段主要功能為：輸入六個整數，檢測並印出最後一個 數字是否為六個數字中最小的值。然而，這個程式是錯誤的。請問以下哪一組測試資料可以測試出程式有誤？(A) 11 12 13 14 15 3 (B) 11 12 13 14 25 20 (C) 23 15 18 20 11 12 (D) 18 17 19 24 15 16 <pre> #define TRUE 1 #define FALSE 0 int d[6], val, allBig; ... for (int i=1; i<=5; i=i+1) { scanf ("%d", &d[i]); } scanf ("%d", &val); allBig = TRUE; for (int i=1; i<=5; i=i+1) { if (d[i] > val) { allBig = TRUE; } else { allBig = FALSE; } } if (allBig == TRUE) { printf ("%d is the smallest.\n", val); } else { printf ("%d is not the smallest.\n", val); } </pre>	
6	<pre> int main() { //APCS10510 int a[9] = {1, 3, 5, 7, 9, 8, 6, 4, 2}; int n=9, tmp; for (int i=0; i<n; i=i+1) { tmp = a[i]; a[i] = a[n-i-1]; a[n-i-1] = tmp; } for (int i=0; i<=n/2; i=i+1) printf ("%d %d ", a[i], a[n-i-1]); return 0; } </pre>	

7	下面哪組資料若依序存入陣列中，將無法直接使用二分搜尋法搜尋資料？ (A) a, e, i, o, u (B) 3, 1, 4, 5, 9 (C) 10000, 0, -10000 (D) 1, 10, 10, 10, 100	
8	<pre> int main() { int maze[5][5]= {{1, 1, 1, 1, 1}, {1, 0, 1, 0, 1}, {1, 1, 0, 0, 1}, {1, 0, 0, 1, 1}, {1, 1, 1, 1, 1} }; int count=0; for (int i=1; i<=3; i=i+1) { for (int j=1; j<=3; j=j+1) { int dir[4][2] = {{-1,0}, {0,1}, {1,0}, {0,-1}}; for (int d=0; d<4; d=d+1) { if (maze[i+dir[d][0]][j+dir[d] [1]]==1) { count = count + 1; } } } } printf("%d",count); return 0; } </pre>	
9	<pre> int main() { //APCS10510 int i, sum, arr[10]; for (int i=0; i<10; i=i+1) arr[i] = i; sum = 0; for (int i=1; i<9; i=i+1) sum = sum - arr[i-1] + arr[i] + arr[i+1]; printf ("%d", sum); return 0; } </pre>	

10	<pre> int main(){//APCS10603 //int A[]={9,6,7,3,1,2}; int A[]={9,9,9,9,9,9}; int n=6; int p,q; p=q=A[0]; for (int i=1; i<n; i=i+1) { if (A[i]>p) p=A[i]; if (A[i]<q) q=A[i]; } printf("p=%d,q=%d",p,q);//9,9 return 0; } </pre>	
11	若 A 是一個可儲存 n 筆整數的陣列，且資料儲存於 A[0]~A[n-1]。經過底下程式碼運算後，以下何者敘述不一定正確？ (A) p 是 A 陣列資料中的最大值 (B) q 是 A 陣列資料中的最小值 (C) $q < p$ (D) $A[0] \leq p$ <pre> int A[n]={ ... }; int p = q = A[0]; for (int i=1; i<n; i=i+1) { if (A[i] > p) p = A[i]; if (A[i] < q) q = A[i]; } </pre>	

12	<pre> int main() { int M=2,N=3; int A[2][3]={0,1,2},{3,4,5}}; for (int i=0; i<M; i=i+1){ int rowsum =0; for (int j=0; j<N; j=j+1) { rowsum = rowsum + A[i][j]; } printf("The sum of row %d is %d\n", i,rowsum); } return 0; } </pre>	
13	若 A[][] 是一個 MxN 的整數陣列，底下程式片段用以計算 A 陣列每一列的總和，以下敘述何者正確？ (A) 第一列總和是正確，但其他列總和不一定正確 (B) 程式片段在執行時會產生錯誤 (run-time error) (C) 程式片段中有語法上的錯誤 (D) 程式片段會完成執行並正確印出每一列的總和 <pre> void main () { int rowsum 0; for (int i=0; i<M; i=i+1) for (int j=0; j<N; j=j+1) { rowsum = rowsum + A[i][j]; printf(The sum of row %d is %d. n "", i, } </pre>	
14	以下程式若輸入依序為整數 0, 1, 2, 3, 4, 5, 6, 7, 8, 9，請問 X[] 陣列的元素值依順序為何？ //APCS10603 <pre> int main() { int X[10] = {0}; for (int i=0 ; i<10; i=i+1) scanf("%d" ,&X[(i+2)%10]); for (int i=0 ; i<10; i=i+1) printf("%d,%d\n",i,X[i]); return 0; } </pre>	

15	若 A[1]、A[2]，和 A[3] 分別為陣列 A[] 的三個元素 (element)，下列那個程式片段可以將 A[1] 和 A[2] 的內容交換？ (A) A[1] = A[2]; A[2] = A[1]; (B) A[3] = A[1]; A[1] = A[2]; A[2] = A[3]; (C) A[2] = A[1]; A[3] = A[2]; A[1] = A[3]; (D) 以上皆可	
16	<pre>int main() { int M = 1, N = 101, s = 3; int A [] = {80, 90, 100}; for (int i=0; i<s; i=i+ 1) { if (A [i]>M) M = A [i]; if (A[i]<N) N = A [i]; } printf("Max = %d, Min= %d \n", M, N); return 0; }</pre>	
17	<pre>int main() { int M = 1, N = 101, s = 3; int A [] = {80, 90, 100}; for (int i=0; i<s; i=i+ 1) { if (A [i]>M) M = A [i]; else if (A[i]<N) N = A [i]; } printf("Max = %d, Min= %d \n", M, N); return 0; }</pre>	

二、程式設計

1. 假設有資料如下：

5,7,8,4,5,6,2,5,9

求平均、中位數、眾數。

2. T 分數的轉換公式如下：

(1) 求全班平均 $d = (\sum_{i=1}^n A_i) / n$ 。(Ai 為每人成績，n 為全班人數)

(2) 求全班標準差 $SD = \sqrt{\frac{\sum_{i=1}^n (A_i - d)^2}{n}}$ (d 為平均)

(3) 求每人 Z 分數 $Z_i = \frac{A_i - d}{SD}$ (Zi 為每人的 Z 分數)

(4) 求每人 T 分數 $T_i = 500 + 100 \times Z_i$ 。(Ti 為每人的 T 分數)

請產生 40 個 1 至 100 的亂數，其平均分數為何？請依以上演算法將每人成績轉為 T 分數，輸出每人 T 分數，並觀察名次 20 的人平均為何？又 T 分數與名次的關係為何？

- ※3. 長整數加法（假設可 100 位數）。長整數加法演算法如下，請依此演算法完成兩數的相加。

$$\begin{array}{r} 1567 \\ + 835 \\ \hline 2402 \end{array}$$

- ※4. 開根號運算。開根號運算原理如下：

$$(10a+b)^2 = 100a^2 + 20ab + b^2 = 100a^2 + b(20a+b)$$

以 138384 為例，開根號運算過程如下：

(1) 由以上 $100a^2 + b(20a+b)$ ，表示我們由左到右，每兩個一組，先找 a，再找 b。

(2) 找出 a。從 9,8,7 到 1 的平方依序找出小於等於最左邊那一組的數字（本例是 3），因為 $81(9*9)$, $64(8*8)$, 49 , 36 , 25 , 16 都太大，所以找到 3，並扣掉 3 的平方 9，剩下 4，請看下圖運算步驟 1。

(3) 使用迴圈，從第二組數字開始逐一找 $b_1 b_2 b_3 \dots$ 。

(a) 計算每一次的餘數。餘數 $d = \text{前面餘數} \times 100 + \text{這一組數字}$ ，本例 d 是 483。

(b) 用迴圈找 b 。 b 從 9,8,7 到 1，找 $b(20a + b)$ 小於餘數 d 。例如，本例

a 是 3， d 是 483

$9 \times (20 \times 3 + 9) = 621$ 太大，不行

$8 \times (20 \times 3 + 8) = 544$ 太大，不行

$7 \times (20 \times 3 + 7) = 469$ 已經小於 483，所以可以，請看下圖步驟 2。

(c) 餘數 d 扣掉 $b(20a + b)$ 。 $d = d - b(20a + b)$ 。

(4) 輸出 $ab_1b_2b_3\cdots$ ，即為所求。

步驟		3	7	2
1		1	3 8 3 8	4
			9	
2	$3 \times 20 + 7 = 67$ $67 \times 7 = 469$	4	8 3	
		4	6 9	
3	$37 \times 20 + 2 = 742$ $742 \times 2 = 1484$		1 4 8 4	
			1 4 8 4	
4				0

指標

Chapter

10

學習綱要

- 📖 10-1 指標與變數
- 📖 10-2 指標與陣列
- 📖 10-3 指標與字串

學習目標

- 📖 1. 能以指標處理單一變數。
- 📖 2. 能以指標處理陣列。
- 📖 3. 能以指標處理字串。
- 📖 4. 能以指標實作單一變數、陣列與字串實例。

指標型態是 C 語言獨有的資料型態，它的優點是指標型態直接使用記憶體位址存取資料，如此可以提升程式的執行效率。指標型態的應用非常廣，可以協助處理陣列與字串，以提高程式的執行效率。

10-1 指標與單一變數

請鍵入以下程式，寫出執行結果。

```
int main() {  
    int  *ptr;  
    int  a=5;  
    ptr = &a;  
    printf("%d\n", *ptr); //5  
    printf("%d\n", ptr);  
    return 0;  
}
```

指標型態的宣告

指標型態的宣告比較特殊，它必須藉助一種指標運算子（*），其宣告語法如下：

資料型態 *指標變數；

資料型態可以是 3-2 節的基本資料型態，指標變數可以是一個合法的變數名稱。例如，以下式子即宣告 ptr 是一個整數的指標變數。

```
int  *ptr;
```

經過以上宣告之後，ptr 即可指向任一整數變數位址，例如：

```
int  a=5;  
ptr = &a;
```

則 ptr 指向變數 a 的位址，&是取位址運算符號。往後即可使用 *ptr 代表變數 a。例如，以下敘述可印出 *ptr 的值。


```
printf("%d\n", *ptr); //5
```

但是，若硬要印出 `ptr` 的值，如以下敘述：

```
printf("%d\n", ptr);
```

則沒有意義了。因為 `ptr` 的值是記憶體的字址，此字址會隨著不同的機器與不同的執行次數而有不同的值。所以，在指標型態中，我們並不在意 `ptr` 所在字址，在意的是 `*ptr` 的值。

⚙ 運算

指標指向之值也可以用來運算，但運算結果僅能放在變數。例如：

```
int main() {
    int a=5, b=4, c;
    int *p, *q, *r;
    p=&a;
    q=&b;
    printf("%d\n", *p); //5

    c=*p+*q;
    printf("%d\n", c); //9
    r=&c;
    printf("%d\n", *r); //9
    return 0;
}
```

若將運算結果也放在指標，那就不行。例如，以下就不行。

```
*r=*p+*q;
```

👉 自我練習

1. 請用指標計算長方形的面積與周長。

10-2 指標與陣列

接著我們要談如何使用指標型態存取陣列的值。例如，請鍵入以下程式，寫出執行結果。

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int a[5]={3, 8, 6, 2, 5};
    int *ptr;    //務必和所指的陣列相同型態
    ptr=a;      // 或
    //ptr = &a[0];
    printf("%d\n",*ptr);//3
    printf("%d\n",ptr[0]);//3
    printf("%d\n",ptr[1]);//8
    printf("%d\n",*(ptr+1));//8
    ptr++;      // ptr=ptr+1
    printf("%d\n",*ptr);//8
    printf("%d\n",ptr[0]);//8
    printf("%d\n",ptr[1]);//6
    printf("%d\n", *(ptr+1));//6
    printf("%d\n", ptr);
    return 0;
}
```

⚡ 指標與一維陣列

```
int a[5]={3, 8, 6, 2, 5};
int *ptr;    //務必和所指的陣列相同型態
```

宣告 ptr 是一種指標型態，ptr 本身是一個位址，*ptr 才是一個值，而且 *ptr 的型態務必和所指的陣列相同型態。將指標指向一維陣列的程式如下：

```
ptr=a;
```

或

```
ptr = &a[0];
```

將 `ptr` 指向 `a[0]` 的位址，`&` 符號是取址的意思。所以若經以上設定，以下程式都可以取陣列值。

```
printf("%d\n", *ptr); //3
printf("%d\n", ptr[0]); //3
printf("%d\n", ptr[1]); //8
printf("%d\n", *(ptr+1)); //8
```

☞ 指標的移動

若要得下一個陣列值，則應將指標向後移一個位置，例如：

```
ptr++; // ptr=ptr+1
printf("%d\n", *ptr); //8
printf("%d\n", ptr[0]); //8
printf("%d\n", ptr[1]); //6
printf("%d\n", *(ptr+1)); //6
```

若您強制要印出 `ptr`，例如：

```
printf("%d\n", ptr);
```

則是得到一個位址，而且這個結果對使用者來說是沒有意義，因為這個位址每次執行均會不同，而且不同的電腦系統也會有不同的值（位址）。重點是我們要得到指標的值 `*ptr`，而不是指標本身 `ptr`。

☞ 指標與資料型態的變化

指標是有智慧的，它會隨著指標值所佔記憶體的大小做出正確的移動，例如：

```
double b[5]={3.3, 9.9, 2.2, 4.4, 6.6};
double *p1; //務必和所指的陣列相同型態
p1=b;
p1++;
printf("%d\n", *p1);
```

亦可得到陣列的第 1 個元素 9.9，但若：

```
int a[5]={3, 8, 6, 2, 5};
double *p1;
p1=a;
```

則會得到錯誤結果，因為 *p1 是指向 8 個 byte 的 double 型態，而 *ptr 是指向 2 個 byte 的 int 型態，所以同樣是指向下一個元素 ptr++ 或 p1++，但移動的記憶體數不同。

範例10-2a

同範例 9-1a，但改用指標型態處理陣列的值。

```
Average: 68
max: 99
min: 44
```

程式列印

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int a[]={77, 66, 99, 44, 55}; // 陣列宣告與初始化
    int *ptr=a;                    // 宣告ptr為指標型態，並指向a陣列
    int i;
    int max=*ptr;                  // 先設1號為最高分的播臺主
    int min=*ptr;                  // 先設1號為最低分的播臺主
    int sum=*ptr;                  // 先設1號為總分
    ptr++;
    for(i=1; i<=4; i++) {
        sum+=*ptr;
        if(*ptr>max)                // 只要大於播臺主便取代播臺主
            max=*ptr;
        if(*ptr<min)
            min=*ptr;
        ptr++;                      // 指向下一個陣列值
    }
    // 結果輸出
```

```

printf("Average: %d\n", sum/5);
printf("max:      %d\n", max);
printf("min:      %d\n", min);
return 0;
}

```

👉 自我練習

- 設有資料如下：
 - 1, 0, 3, 4, -5, 8
 - (1) 請以陣列儲存以上資料。
 - (2) 請以指標的方式，統計正數、0、負數的個數。

範例10-2b

請以指標型態重做氣泡排序法。

```

Before sort:
99 88 77 66 55
After sort:
55 66 77 88 99

```

👉 程式列印

```

#include <stdio.h>
#include <stdlib.h>
int main(){
    int a[]={99, 88, 77, 66, 55};
    int *ptr=a;
    int i, j, t;
    const int N=5;
    //cout<<"Before sort:\n";
    printf("Before sort:\n");
    for(i=1; i<=N; i++)    {
        //cout<< *ptr<<",";
        printf("%4d ",*ptr);
        ptr++;
    }
    /* sort */
    for(i=1; i<=(N-1); i++)    {
        ptr=a;

```

```

        for(j=1; j<=(N-i); j++) {
            if(*ptr>*(ptr+1)){
                t=*ptr;
                *ptr=*(ptr+1);
                *(ptr+1)=t;
            }
            ptr++;
        }
    }
    //cout<<"\nAfter sort:\n";
    printf("\nAfter sort:\n");
    ptr=a;
    for(i=1; i<=N; i++) {
        //cout<< *ptr<<" ";
        printf("%4d ",*ptr);
        ptr++;
    }
    return 0;
}

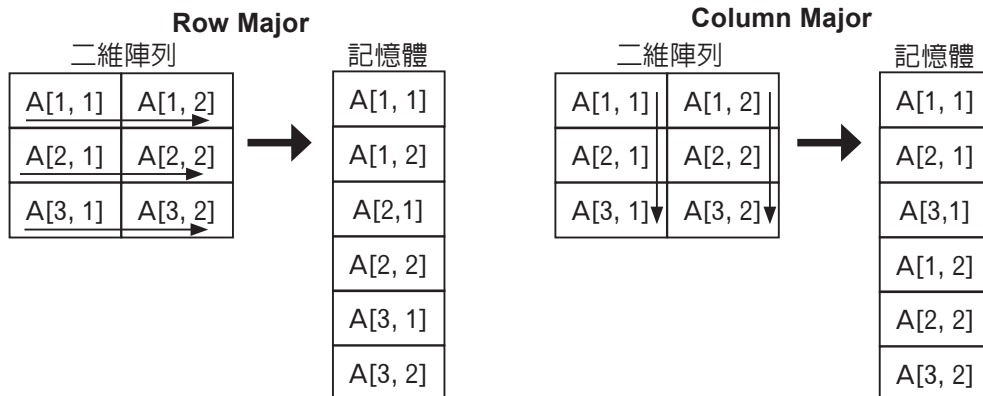
```

自我練習

1. 請以指標型態，重作計數排序法。

指標與二維陣列

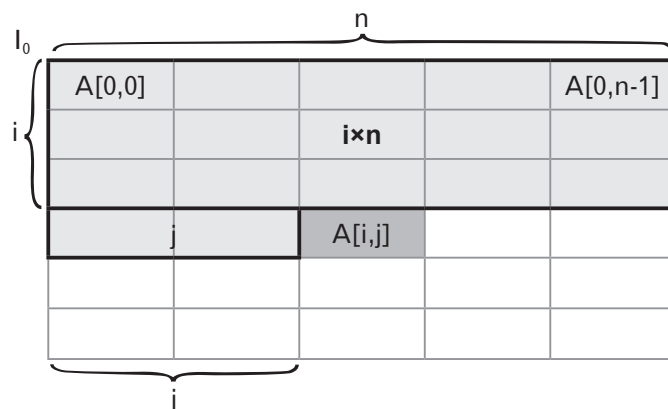
二維陣列在人們眼中通常用於處理二維的行列表格，但電腦內部的記憶體中，是以一維的方式排列，二維轉一維的轉換方式有兩種，分別是 Row-major 與 Column-major，如下表所示：



指標則是採用 Row-major。若第一個元素為 $A[0, 0]$ 位址是 I_0 ，則 $a[i, j]$ 位址公式為：

$$A[i, j] = I_0 + (i \times n + j) \times d$$

(n 是每列元素個數，d 是每個元素所佔記憶體的数量)



例如，若有資料如下：

0	1	2	3
4	5	6	7
8	9	10	11

先以二維陣列儲存，程式如下：

```
int a[3][4]={0,1,2,3},{4,5,6,7},{8,9,10,11}};
```

二維陣列以指標處理的步驟是先指定第一元素給指標變數，如以下敘述：

```
int *ptr;
ptr = &a[0][0];          /* 不能寫成 ptr=a */
```

若要存取其值，只要使用陣列名稱與適當的位移即可。例如：以下敘述可設定第 i 列第 j 行元素等於 5。

```
*( *(a+i) + j ) =5;      /* 不能寫成 * ( *ptr+i ) + j = 5 */
```

上式若要以 `ptr` 表示，則應使用以下敘述將二維排列轉為一維的直線排列，其中 3 是每一維的欄數。

```
*( ptr + 3*i + j ) =5;  /* 不能寫成 * ( a + 3*i + j ) = 5 */
```

以上敘述將二維的列與行轉為一維的直線排列，如下表所示，括號內為二維（列、行），括號外的為一維直線的位移量。

(0,0)0	(0,1)1	(0,2)2	(0,3)3
(1,0)4	(1,1)5	(1,2)6	(1,3)7
(2,0)8	(2,1)9	(2,2)10	(2,3)11

自我練習

1. 請鍵入以下程式，寫出執行結果。

題號	程式	執行結果
1	<pre>#include <stdio.h> #include <stdlib.h> int main(){ int a[3][4]={0,1,2,3},{4,5,6,7},{8,9,10,11}}; int *p=&a[0][0]; printf("%d\n",*p); p++; printf("%d\n",*p); p=&a[0][0]; printf("%d\n",*p); printf("%d\n",*(*(a+1) + 2)); printf("%d\n",*(p + 4*1 + 2)); }</pre>	
2	程式同第 1 題，以 <code>a</code> 撰寫程式，將資料 2 改為 12，並輸出。	
3	程式同第 1 題，以 <code>p</code> 撰寫程式，將資料 5 改為 15，並輸出。	

※範例10-2c

請以指標型態重做範例 9-2b 的學生成績統計。

no	chin	eng	math	avg	rank	fail
1	50	60	70	60	3	1
2	30	40	50	40	4	3
3	70	80	90	80	1	0
4	66	77	88	77	2	0
5	22	33	44	33	5	3
0	47	58	68	58	0	0

 程式列印

```
#include <stdio.h>
#include <stdlib.h>
int main(){
    int a[7][7]={0,0,0,0,0,0,0},
                {1,50,60,70,0,0,0},
                {2,30,40,50,0,0,0},
                {3,70,80,90,0,0,0},
                {4,66,77,88,0,0,0},
                {5,22,33,44,0,0,0},
                {0,0,0,0,0,0,0}};

    int *ptr=&a[0][0];          // ptr指向陣列
    int i, j, k;
    const int stdnum=5;
    for(j=1; j<=stdnum; j++){   // 計算每人各科成績
        for(k=1; k<=3; k++){
            (*(a+j)+4) += (*(a+j)+k);    // a[j][4]+=a[j][k]
        }
        (*(a+j)+4) /= 3;           // a[j][4]/=3
    }
    /* 計數排序 */
    for(j=1; j<=stdnum; j++)
        *(ptr+7*j+5)=1;           // a[j][5]=1 每人名次先填第一名
        /*(*(a+j)+5)=1;
    for(j=1; j<=stdnum; j++)
        for(k=j+1; k<=stdnum; k++)
            if( (*(a+j)+4) < (*(a+k)+4) )
                (*(a+j)+5) = (*(a+j)+5)+1; // a[j][5]++
            else
```

```

        (*(a+k)+5) = (*(a+k)+5)+1; // a[k][5]++

/* 計算每人不及格科數 */
for(j=1; j<=stdnum; j++)
    for(k=1; k<=3; k++)
        if( (*(a+j)+k) < 60) // a[j][k] < 60
            (*(a+j)+6) = (*(a+j)+6)+1; // a[j][6]++

/* 統計各科平均 */
for(j=1; j<=4; j++) {
    for(k=1; k<=stdnum; k++)
        (*(a+stdnum+1)+j) += (*(a+k)+j);
        // a[stdnum+1][j]=a[stdnum+1][j]+a[k][j]

    (*(a+stdnum+1)+j) /= stdnum; // a[stdnum+1][j] /= stdnum
}
//output
//cout<<" no chin eng math avg rank fail "<<endl;
printf(" no chin eng math avg rank fail \n");
for(i=1; i<=stdnum+1; i++) {
    for(j=0; j<7; j++){
        //cout.width(5);
        //cout<< (*(a+i)+j); // a[i][j] */
        printf("%5d", (*(a+i)+j));
    }
    //cout<<endl;
    printf("\n");
}
return 0;
}

```

自我練習

1. 請以指標型態，重作兩個矩陣的相加。例如：

$$\begin{pmatrix} 2 & 1 & 3 \\ 0 & 1 & 0 \end{pmatrix} + \begin{pmatrix} 1 & 2 & 3 \\ 1 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 3 & 3 & 6 \\ 1 & 2 & 0 \end{pmatrix}$$

10-3 指標與字串

範例10-3a

請鍵入以下程式，並觀察執行結果。

```
#include <stdio.h>
#include <stdlib.h>
int main( ){
    char s[] = "abcde";
    char *ptr = s;
    printf( "%c\n", *ptr );//指向一個字元 a
    printf( "%s\n", ptr );//代表整個字串 abcde
    ++ptr;//指標前進
    printf( "%c\n", *ptr ); //指向一個字元b
    printf( "%s\n", ptr );//bcde
    ptr--;
    printf( "%c\n", *ptr ); //指向一個字元a
}
```

前面我們說明了如何使用字元陣列來處理字串，如以下程式：

```
char s[10]="abcde";
```

現在我們將示範如何使用指標來處理字串。首先須將字串變數宣告為指標，如下：

```
char *ptr=s;
```

正如我們上節所提過的，陣列名稱 *s* 代表的即為 *s* 陣列的基位址 (&*s*[0])，因此在上列程式宣告中，都是以傳位址的方式進行。指標變數 *ptr* 所儲存的值即為 *s* 陣列的基位址，而非陣列中的資料項，接下來我們便可以利用指標的移動來存取字串。例如：

```
printf( "%c\n", *ptr );//指向一個字元 a
```

**ptr* 的結果是 *a*。

```
printf( "%s\n", ptr );//代表整個字串 abcde
```

ptr 的結果為 abcde。

```
++ptr;
printf( "%c\n", *ptr ); //b
```

的結果為 b。

```
printf( "%s\n", ptr );//bcde
```

ptr 的結果為 bcde，代表基址已經右移一個位置。

```
ptr--;
printf( "%c\n", *ptr ); //a
```

又得到 a。

自我練習

1. 請鍵入以下程式，並觀察執行結果。

題號	程式	結果
1	<pre>#include < stdio.h > //統測111 int main() { char Str[] = "abcde"; char *ptr = Str; printf("%s\n", ++ptr); printf("%s\n",ptr); printf("%c\n",ptr); printf("%c\n",*ptr); printf("%s\n",*ptr); return 0; }</pre>	

```

2  #include <stdio.h>
   #include <stdlib.h>
   int main( ){
       char *str="I am a good teacher.";
       char s[20];
       int a=0,b=0;
       while(*str != '\0') {
           printf("%c",*str);
           s[a]=*str;
           a++;
           if(*str==' ')
               b++;
           str++;
       }
       printf("\n");
       printf("%s\n",str);
       printf("%c\n",*str);
       printf("%s\n",s);
       printf("%d\n",a);
       printf("%d\n",b);
       return 0;
   }

```

範例 10-3b

寫一程式，要求如下，

1. 自行輸入一字串。
2. 輸入字串中，如有空格即換新行。
3. 輸入字串中，如有 's' 者，改為 'S'。

```

input: This is a book.
This is a book.
ThiS
iS
a
book.

```

 程式列印

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#define MAX 100
int main() {
    char in[MAX], *ptr;
    int c, i=0;
    printf("input: ");
    while( (c=getchar()) !=EOF && c != '\n') {
        /* 將每個字元單獨讀入陣列中 */
        in[i]=c;
        i++;
    }
    in[i]='\0';
    ptr=in;
    printf("\n%s", ptr);
    while(*ptr != '\0') {
        if(*ptr=='s') { /* 如果輸入的字串中有's'，則代換為'S' */
            *ptr = toupper(*ptr);
        }
        if(*ptr==' ') { /* 如果輸入的字串中有空白，則代換為換行 */
            *ptr = '\n';
        }
        ptr++;
    }
    ptr=in;
    printf("\n%s", ptr);
    return 0;
}
```

 自我練習

1. 請寫一程式，滿足以下要求：
 - (1) 可輸入一含有空白與大小寫字元的字串。
 - (2) 統計阿拉伯數字的個數。
 - (3) 統計大寫字元的個數。
 - (4) 統計小寫字元的個數。
 - (5) 統計空白個數。

公用函式及自訂函式

學習綱要

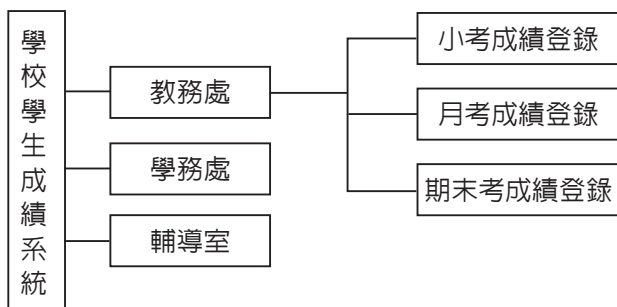
- ☞ 11-1 數值函式
- ☞ 11-2 字元處理函式
- ☞ 11-3 字串函式
- ☞ 11-4 時間函式
- ☞ 11-5 自訂函式
- ☞ 11-6 實例探討

學習目標

- ☞ 1. 認識 C 語言的函式種類。
- ☞ 2. 認識 C 語言數值函式。
- ☞ 3. 認識 C 語言字元處理函式。
- ☞ 4. 認識 C 語言字串函式。
- ☞ 5. 認識 C 語言時間函式。
- ☞ 6. 能夠自訂函式。
- ☞ 7. 能實作功用函式及自訂函式應用實例。

概念

現今工業化的產品通常以模組開發，例如，一部電腦分成主機、螢幕、硬碟、鍵盤、滑鼠等模組，產品的設計與維修都是模組化，彼此獨立，也就是鍵盤壞了，就換鍵盤，滑鼠壞了就換滑鼠。軟體工業是發展較晚的新興產業，也是遵循此道理，希望將問題模組化，每個模組也和硬體系統一樣，是可組合、可分解和可更換的單元。模組化程式設計是指解決一個複雜問題時由上而下逐層把軟體系統劃分成若干模組。每個模組完成一個特定的子功能，所有的模組的功能與模組之間的介面要先定義，並可按某種方法組裝起來，完成整個系統所要求的功能。例如，要撰寫學校學生成績管理系統，就可先分成三個模組，分別是教務處的考試成績模組、學務處的操行系統模組、輔導處的輔導評量系統模組，這些子系統都可完成特定功能，且資料交換介面也已經定義且公開。教務處模組則再分為小考成績登錄模組、月考成績登錄模組、期末考成績登錄模組等系統，如下圖所示：



公用函式(Common functions)

以上每一軟體系統也應遵循模組化程式設計的原則，將使用者經常使用的一些功能撰寫成函式 (Functions)，這些由軟體系統廠商所開發的函式，稱為公用函式，公用函式於本書 11-1~11-4 節介紹。同樣的道理，使用者也可以將常用功能寫成函式，此稱為自訂函式 (Self-defined function)，自訂函式於本書 11-5 節介紹。

11-1 數值函式(Mathematical Functions)

C 語言中，有關數值運算的內儲函式大部分是被集中宣告在 `math.h` 標頭檔中，程式設計者要使用這些函式，要先在程式開頭以 `#include <math.h>` 載入這些函式庫，茲將一些常用的數值函式介紹如下：

1. **abs**

語法：`int abs (int x) ;`

`double fabs (double x) ;`

說明：傳回 x 的絕對值。

例如：1. `abs(-3)= 3`

2. `abs(- 4.6)= 4` // 不會 4.6，而是 4，這很容易錯，請小心。

若要傳回浮點數，請用 `fabs`。例如：

1. `fabs(-4.6)=4.6`

2. **pow**

語法：`double pow (double x , double y) ;`

說明：傳回 x^y 值。

例如：`pow(2, 3)= 8.0`

3. **sqrt**

語法：`double sqrt (double x) ;`

說明：傳回 $\sqrt{x}$ 的值。

例如：`sqrt(4)= 2.0`

4. **cbrt**

語法：`double cbrt( double x );`

說明：傳回開立方值。

例如：`cbrt(27)=3.0`

5. **trunc**

語法：`double trunc( double x );`

說明：去掉小數部分。

例如：`trunc(4.8)=4.0;trunc(-4.8)=-4.0;`

6. rint

語法：double rint(double x);

說明：取最近的整數，0.5(含) 以下捨去，0.5 以上則無條件進 1。

例如：rint(0.51)=1.0;rint(0.5)=0.0; rint(1.8)=2.0;
rint(-0.51)=-1.0;rint(-0.5)=0.0;rint(-0.49)=0.0;

7. round

語法：double round(double x);

說明：四捨五入，0.49(含) 以下捨去，0.5(含) 以上則無條件進 1。

例如：round(4.4)=4.0;round(4.5)=5.0; round(-4.5)=-5.0;

8. ceil

語法：double ceil(double x);

說明：傳回大於等於 x 的最小整數。

例如：1. ceil(2.5)= 3.0
2. ceil(- 2.5)= -2.0

9. floor

語法：double floor(double x);

說明：傳回小於等於 x 的最大整數。

例如：1. floor(2.5)= 2.0
2. floor(-2.5)= -3.0

※ 10. sin 、 cos 、 tan

語法：double sin(double x) ;

double cos(double x) ;

double tan(double x) ;

說明：1. 傳回三角函數值。

2. x 的單位為弧度 (弧度)，弧度與度度的關係為

$$\text{弧度} = \text{度} \times \frac{\pi}{180} \quad (\pi \cong 3.14159)$$

$$30^\circ = \frac{\pi}{6} \cong 0.5235987$$

3. 例如：sin(M_PI/6)= 0.5 // M_PI 為 math.h 的常數 π

※ 11. asin 、 acos 、 atan

語法：double asin(double x)；

double acos(double x)；

double atan(double x)；

說明：1. 傳回反三角函數值。

2. asin 與 acos 的 x 範圍為 $-1 \leq x \leq 1$ ，若超出此範圍，將出現錯誤訊息。例如：asin(0.5)/M_PI*180 的值為 30（度）。

3. atan 將傳回 $-\pi/2$ 與 $\pi/2$

※ 12. atan2

語法：double atan2 (double y, double x)；

說明：將直角座標 (x,y) 化為極座標，傳回極座標 (r, θ) 的 θ ，其中 $\theta =$ ，其值在 $-\pi$ 與 π 之間。

例如：atan2(3, 4)*180/M_PI

結果：36.9（度），0 度在正 x 軸。

又例如，atan2(-3, 4)*180/M_PI 是 -36.9。

※ 13. exp

語法：double exp (double x)；

說明：傳回 e^x 值。

例如：exp(0)= 1.0，請自行求 exp(-1),exp(-2),exp(-3),exp(-4),exp(-5)，高職基本電學用得到。

※ 14. log

語法：double log(double x)；

說明：傳回 $\ln(x)$ 值（base e ）。

例如：log(10)= 2.3

自我練習

1. 請自行執行以上函式。例如：

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int main( ){
    printf("%d\n",abs(-3.4));
    printf("%lf\n",fabs(-3.4));
}
```

2. 請寫一程式，可以滿足以下函式，並分別將 x 值從 -10 到 10 輸出其對應 y 值。

$$f(x) = 2|x-2| + 1$$
範例11-1a

試寫一程式，使用雙迴圈印出如下結果：

```
* * * * *
* * * *
* * *
* *
*
* *
* * *
* * * *
* * * * *
```

演算分析

1. 由前一自我練習可知，一個 `abs()` 的輸出圖形會有一個折點，所以若圖形有折點，請務必聯想到使用 `abs()` 函式。
2. 利用 `abs()` 在 0 產生折點的特性，且本例共印出 9 列，所以假設 i 從 4 到 -4 如下。

i	星星個數	j	輸出結果
4	5	01234	*****
3	4	0123	****
2	3	012	***
1	2	01	**
0	1	0	*
-1	2	01	**
-2	3	012	***
-3	4	0123	****
-4	5	01234	*****

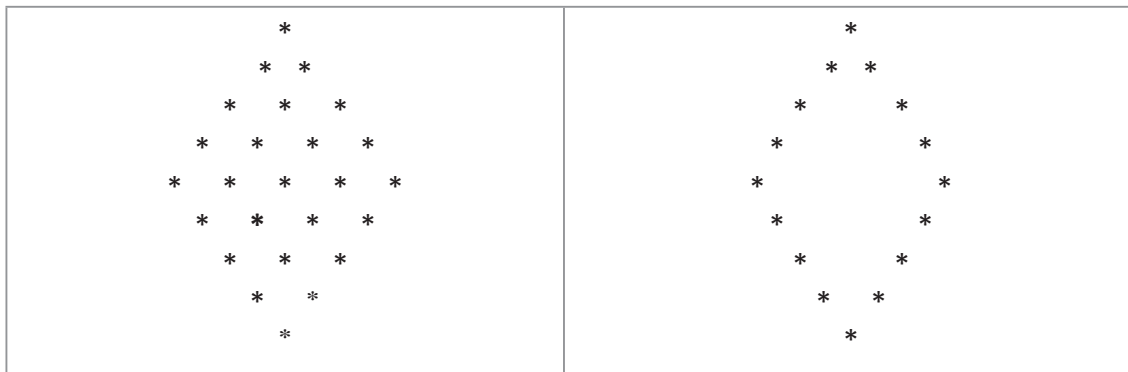
3. 內迴圈 j 為星星個數，其與外迴圈 i 的關係為 $j=abs(i)+1$ ，但因從 0 算起，所以應減 1。

程式列印

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int main()
{
    for(int i=4; i>=-4; i--) {
        for(int j=0; j<=abs(i); j++)
            printf("* ");
        printf("\n");
    }
    return 0;
}
```

自我練習

1. 請寫一程式，使用雙迴圈輸出以下結果。(這就是一種電腦程式設計的運算思維，自己練習才有樂趣)



※ 範例11-1b

請寫一程式，可以將輸入的直角座標轉為極座標。

輸出結果

```
3.14159
5.0∠36.9
```

程式列印

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int main()
{
    printf("%8.5f\n",M_PI);
    double x=4,y=3;
    double r,s;
    r=sqrt(x*x+y*y);
    s=atan2(y,x)*180/M_PI;
    printf("%4.1f∠%4.1f",r,s);
    return 0;
}
```

自我練習

- ※ 1. 若電腦有繪直線指令，給兩點就可繪出此直線，那要繪出三角形，只要任意給三點就可繪出此三角形，若任意給一個凸四邊形的四個點，除非規定使用者要按序輸入此四個點，不然無法繪出此四邊形，請寫一程式，可以將任一輸入的凸四邊形的四個點按照順時針順序排序。
- 提示：
- (1) 求此四個點的中心點。
 - (2) 將此中心點當作極座標原點，求此四點的極座標。
 - (3) 將此四個極座標的 θ 排序，就可重排其順序。

stdlib.h 函式庫

前面 5_4 節已經簡單介紹亂數的產生，現在詳細整理如下。關於亂數的產生，stdlib.h 中有亂數產生等函式，分別說明如下：

1. rand

語法：`int rand(void);`

說明：產生大於等於 0 且小於 32767 隨機亂數。

2. srand

語法：`srand(time(0));` //include<time.h>

說明：以系統時間為亂數種子，請回顧 2_3 節。

3. RAND_MAX

語法：`RAND_MAX`

說明：取得亂數最大值 32767。

範例11-1c

請寫出一個擲兩顆骰子的程式。

執行結果

The Random Number is 4,6

 程式列印

```
#include <stdio.h>
#include <stdlib.h>
//步驟1，引入標頭檔
#include <time.h> //本例使用Time() 函式
int main() {
    int a,b;
    //步驟2，以當時的系統時間，當作亂數表的起點
    srand(time(0));
    //步驟3，取亂數
    a=rand()%6+1;
    b=rand()%6+1;
    printf("The Random Number is %d,%d .\n", a,b);
    return 0;
}
```

 自我練習

1. 請將 `srand(time(0));` 去掉，並重複執行程式，觀察執行結果。
2. 請練習產生 3 到 8 之間的亂數。

範例11-1d

撲克牌洗牌。撲克牌剛買來時都是按序排好，請寫一程式，可以協助洗牌，並發給四個人，每人 13 支。

 執行結果

(C 代表黑梅，D 代表紅磚，H 代表紅心，S 代表黑桃)

1C	2C	3C	4C	5C	6C	7C	8C	9C	10C	11C	12C	13C
1D	2D	3D	4D	5D	6D	7D	8D	9D	10D	11D	12D	13D
1H	2H	3H	4H	5H	6H	7H	8H	9H	10H	11H	12H	13H
1S	2S	3S	4S	5S	6S	7S	8S	9S	10S	11S	12S	13S
1: 10H	11C	10D	6S	9S	12H	4D	5C	5D	8D	4S	7D	5S
2: 9S	12H	4D	5C	5D	8D	4S	7D	5S	7S	3H	3S	4C
3: 5D	8D	4S	7D	5S	7S	3H	3S	4C	2S	8S	10C	13H
4: 5S	7S	3H	3S	4C	2S	8S	10C	13H	10S	2C	2H	7C

👉 資料結構與演算法則

1. 使用 0 到 51 分別代表撲克牌的 1C,2C,3C,4C,5C,6C,7C,8C,9C,10C,11C,12C,13C,1D,2D...，並以 a[] 儲存。
2. 所以牌面值為 $a[i]\%13+1$ ，花色是 $a[i]/13$ 。

👉 程式列印

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int main()
{
    int x1,x2;
    time_t t;    //需要宣告#include <time.h>
    srand(time(0)); //隨機選取亂數起點
    int a[52];
    int b[13][4];
    char c[]="CDHS";
    int i,j, r, tt;
    //設定每一張牌
    for(i=0; i<=51; i++)
        a[i]=i;
    //輸出
    for (i=0;i<=3;i++){
        for (j=0;j<=12;j++){
            printf("%2d%c ",a[i*13+j]%13+1,c[a[i*13+j]/13]); //大小/花色
            printf("\n");
        }
        printf("\n");
    }
    /* 洗牌，逐一將每張牌與產生的亂數交換 */
    for(i=0; i<=51; i++) {
        r=rand()%52;
        tt=a[i]; a[i]=a[r]; a[r]=tt;
    }
    //每人輪流發一張，陣列索引按照b00b10b20b30b10b11b12b13 拿牌
    for(i=0; i<=51 ; i++)
        b[i %4][i/4]=a[i];
    //輸出每一人的牌
    for (i=0;i<=3;i++){
        printf("%d:",i+1);
```

```

        for (j=0;j<=12;j++){
            printf("%2d%c ",b[i][j]%13+1,c[b[i][j]/13]);
        }
        printf("\n");;
    }
    return 0;
}

```

自我練習

- 請寫一程式，滿足以下條件，
 - 產生一付撲克牌，分四種花色由小而大輸出。
 - 洗牌，然後分給四個人。
 - 將每個人的牌先按花色，再按大小輸出。
- 請寫一個樂透開獎程式。(假設 1 到 42 號球，依序隨機選出七個，第七球是特別號。)
- 收音機電台通常都有來信者贈獎活動，請寫一程式可以輸入來信人數與贈品數量，就可自動公平抽出幸運中獎編號。提示：與上一題相同，來信數就是上一題的樂透球數，贈品數量就是隨機選出的數量。

11-2 字元函式(Character Function)

在 C 語言中，字元的輸出入，除了可使用 `scanf()` 與 `printf()` 外，亦可使用 `getchar()` 與 `putchar()`，此二函式都是包含於 `stdio.h` 標頭檔。

1. `getchar()`

`getchar()` 的作用是用來讀取鍵盤輸入的一個字元，如以下敘述，

```

int b;
b = getchar();

```

可從鍵盤讀取一個字元，其與 `scanf("%c",&b)` 最大的不同是，`getchar()` 可輸入含有空白的字串。其次，與 `getchar()` 接近的函式還有 `getche()` 與 `getch()`，其差別是 `getche` 不用按『Enter』，且與 `getchar()` 相同，有輸出使用者所按鍵字元；`getch()` 不用按『Enter』，也沒有輸出按鍵內容。

例如，請個別執行以下敘述，並比較其差別。

```
int b= getchar( );//有輸出，要Enter，include <stdio.h>
int b=getche( );//有輸出，不要Enter，include <conio.h>
int b=getch();//無輸出，不要Enter，include <conio.h>
```

2. putchar ()

putchar() 是將一個字元寫到 (輸出) 螢幕上。假設在上例中，我們透過鍵盤輸入一字元 'B'，'B' 將會被儲存在字元變數 b 中，這時便可以使用 putchar() 函式將其輸出至螢幕上，如下：

```
putchar(b );
```

結果會得到 'B'。

範例11-2a

請寫一個程式，可以輸入含有空白的字串。

執行結果

```
Gwosheng Horng
Hello Gwosheng Horng
```

程式列印

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    char c, d[20];
    int i=0;
    while ((c=getchar( )) !='\n'){
        d[i]=c;
        i++;
    }
    d[i]='\0';//要自己放字串結束字元
    printf("Hello %s\n", d);
    return 0;
}
```

 自我練習

1. 同上範例，但將每一字元的 ASCII 碼的數值加 1 後輸出。

 ctype.h

ctype.h 存有一些字元處理函式，分別說明如下：

1. isalpha(c)

判斷字元 c 是否為字母。若為真，則傳回非零值；否則傳回 "0"，例如：

```
printf("%d", isalpha(97)); // 大寫A~Z的ASCII是65~90, 小寫97~122
```

其結果為真，所以傳回非零值。又例如：

```
int a;  
a = isalpha(96);  
printf("%d", a);
```

其結果為 0。

2. isupper(c)

判斷字元 c 是否為大寫字母。若為真，則傳回非零值；否則傳回 "0"，例如：

```
printf("%d", isupper(65));
```

其結果為真，所以傳回非零值。又例如：

```
int a;  
a = isupper(96);  
printf("%d", a);
```

其結果為 0。

3. islower(c)

判斷字元 c 是否為小寫字母。若為真，則傳回非零值；否則傳回 "0"，例如：

```
printf("%d", islower(97));
```

其結果為真，所以傳回非零值。又例如：

```
int a;  
a = islower(65);  
printf("%d", a);
```

其結果為 0。

4. isdigit(c)

判斷字元 c 是否為數字字元（ASCII 48 ~57）。若為真，則傳回非零值；否則傳回 "0"，例如：

```
printf("%d", isdigit(48));
```

其結果為真，所以傳回非零值。又例如：

```
int a;  
a = isdigit(47);  
printf("%d", a);
```

其結果為 0。

5. isspace(c)

判斷字元 c 是否為空白字元（null）。若為真，則傳回非零值；否則傳回 "0"，例如：

```
printf("%d", isspace(32));
```

其結果為真，所以傳回非零值。又例如：

```
int a;  
a = isspace(33);  
printf("%d", a);
```

其結果為 0。

6. tolower(c)

傳回字元 c 所對應的小寫字母。例如：

```
printf("%c", tolower('A'));
```

其結果為 'a'。又例如：

```
int a;  
a = tolower(65);  
printf("%d", a);
```

其結果為 97。

7. toupper(c)

傳回 c 所對應的大寫字母。例如：

```
printf("%c", toupper('a'));
```

其結果為 'A'。又例如：

```
int a;  
a = toupper(97);  
printf("%d", a);
```

其結果為 65。

8. toascii(c)

傳回 c 所對應的 ASCII 碼。例如：

```
printf("%d", toascii('a'));
```

其結果為 97。

自我練習

請自行執行以上函式，並觀察執行結果。

範例11-2b

請寫一個程式，完成以下要求。

- (1) 計算字母個數。
- (2) 計算大寫字母個數。
- (3) 計算數字個數。
- (4) 計算空白個數。

執行結果

```
This is a book.  
alphabet:11  
upper:1  
digit:0  
space:3
```

程式列印

```
#include <stdio.h>  
#include <stdlib.h>  
#include <ctype.h>  
int main()  
{  
    char c, s[50];  
    int n=0, i, alph=0, upper=0, digit=0, space=0;  
    while((c=getchar())!='\n') {  
        s[n]=c;  
        n++;  
    }  
    for(i=0; i<n; i++) {  
        if(isalpha(s[i]))  
            alph++;  
        if(isupper(s[i]))  
            upper++;  
        if(isdigit(s[i]))  
            digit++;  
        if(isspace(s[i]))  
            space++;  
    }  
    printf("alphabet:%d\n", alph);  
    printf("upper:%d\n", upper);  
    printf("digit:%d\n", digit);  
    printf("space:%d\n", space);  
    return 0;  
}
```

自我練習

1. 請寫一個程式，完成以下要求。
 - (1) 將小寫字母的 ASCII 碼都加 5。
 - (2) 將大寫字母的 ASCII 碼都減 6。
 - (3) 將數字的 ASCII 碼都減 2。
 - (4) 輸出以上密文。

11-3 字串函式(String Function)

C 的字串處理函式都放在 `string.h`，使用前應該先使用 `# include<string.h>`，本單元介紹常用字串函式如下：

1. strlen

語法：`size_t strlen( const char *str );`

說明：計算 `s` 字串之長度，不含 `'\0'`。

例如：

```
char s1[]="ABCD";
printf("%d",strlen(s1)); //4
```

2. strcmp

語法：`int strcmp( const char *s1, const char *s2 );`

說明：比較 `s1` 與 `s2` 後，傳回一整數值。依字串順序，從第 0,1,2,3 個字元逐一比較，若 `s1` 小於 `s2`，傳回值為負數；若 `s1` 等於 `s2`，傳回值為 0；若 `s1` 大於 `s2`，傳回值為正數。（比較的依據是 ASCII 碼，「A」是 65，「B」是 66，「a」是 97，「b」是 98，順序較後面的英文字母其 ASCII 碼所對應數值較大，且小寫英文字母的 ASCII 碼數值大於大寫英文字母的 ASCII 碼數值）

例如：

```
char s1[]="ABCD";
char s2[]="ABCD";
char s3[]="FF";
char s4[]="AA";
printf("%d\n",strcmp(s1,s2)); //0，所有字元要完全相同才會0
printf("%d\n",strcmp(s1,s3)); //-1，s1-s3<0，本例第0個字元就分勝負了
printf("%d\n",strcmp(s1,s4)); //1，s1-s4>0，本例第1個字元才分勝負
```

3. strncmp

語法：`int strncmp( const char *lhs, const char *rhs, size_t count );`

說明：依照指定 `count` 的個數，比較 `lhs` 與 `rhs` 字串，傳回 `lhs-rhs` 的差額。字串完全相同傳回 0，不完全相同時，從前面開始比較，只要有字元不同，就傳回其字元的差額，後面字元就不管了。

例如：

```
char s1[]="ABCD";
char s2[]="ABED";
char s3[]="FF";
char s4[]="AA";
printf("%d\n",strncmp(s1,s2,2));//0,只取前面2個,所以相同
printf("%d\n",strncmp(s1,s3,2));//-1,第0個字元就不同,傳回其差
printf("%d\n",strncmp(s1,s4,2));//1,第1個字元才不同,傳回其差
```

4. strcpy

語法：char *strcpy(char *dest, const char *src);

說明：將字串 src 的內容複製到字串 dest。原字串 dest 的內容已被字串 src 覆蓋。

例如：

```
char s1[]="ABCD";
char s2[]="1234";
strcpy(s1,s2);//s1=s2
printf("%s\n",s1);//1234
```

5. strncpy

語法：char *strncpy(char *dest, const char *src, size_t count);

說明：將字串 src 的內容指定長度 count 複製到字串 dest。原字串 dest 的內容相對長度被字串 src 覆蓋。

例如：

```
char s1[]="ABCD";
char s2[]="1234";
strncpy(s1,s2,2);
printf("%s\n",s1);//12CD
```

例如：

```
char s1[]="ABCD";
char s2[]="1234";
strncpy(s1+1,s2,2);//可以指定從第幾個開始被取代
printf("%s\n",s1);//A12D
```

6. strcat

語法：`char *strcat( char *dest, const char *src );`

說明：將字串 `src` 的內容連接到字串 `dest` 之後，傳回 `dest` 的內容。

例如：

```
char s1[]="ABCD";
char s2[]="1234";
strcat(s1,s2);
printf("%s\n",s1);//ABCD1234
```

7. strncat

語法：`char *strncat( char *dest, const char *src, size_t count );`

說明：將字串 `src` 的內容取指定長度 `count` 連接到字串 `dest` 之後，傳回 `dest` 的內容。

例如：

```
char s1[]="ABCD";
char s2[]="1234";
strncat(s1,s2,2);
printf("%s\n",s1);//ABCD12
```

※ 8. strchr

語法：`char *strchr( const char *str, int ch );`

說明：根據指定字元，將指標指到第一個出現在字串 `str` 中的相同字元，並傳回指標值，也就是字元搜尋。

例如：

```
char s1[]="ABCD";
printf("%d\n",strchr(s1,'B'));//傳回整個記憶體的字址，這沒意義
//要減去相對s1的字址
printf("%d\n",strchr(s1,'B')-s1);//1
printf("%s\n",strchr(s1,'B'));//BCD
```

※ 9. strstr

語法：char *strstr(const char* str, const char* substr);

說明：前面 strchr 是字元搜尋，strstr 則是子字串搜尋，傳回 substr 在 str 的起始位置。

例如：char s1[]="ABCD";

```
char s2[]="BC";
printf("%d\n",strstr(s1,s2)); //傳回整個記憶體位址，沒意義
printf("%d\n",strstr(s1,s2)-s1); //1
```

10. strrev

說明：字串反轉。

例如：

```
char s1[]="ABCD";
strrev(s1);
printf("%s\n",s1); //DCBA
```

※ 11. strtok

說明：以某一個字元分割字串為若干子字串函式。例如，翻譯軟體總要先將句子分為單字。

語法：char *strtok(char *str, const char *delim);

例如：

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main()
{
    char a[]="How many people in your family? These are four
           people in my family.";
    char c[]=" .?"; //分割的字元有三種
    char *d; //指標型態
    d=strtok(a,c); //分割完，剩餘放NULL
    printf("%s\n",d);
    do{
        d=strtok(NULL,c); //下次從NULL繼續分割
    }
```

```
        printf("%s\n",d);  
    }while (d!=NULL);  
    return 0;  
}
```

執行結果如下：

```
How  
many  
people  
in  
your  
family  
These  
are  
four  
people  
in  
my  
family
```

範例11-3a

請寫一程式，滿足以下要求。

- (1) 預設一個 20 個字元的字串。
- (2) 可輸入任一子字串。
- (3) 可搜尋此子字串是否包含於原字串。

執行結果

```
input a sub-string: am  
sub-string am found in I am a super teacher.
```

程式列印

```
#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
int main()  
{  
    char s1[]="I am a super teacher."  
    char s2[30];
```

```

printf("input a sub-string: ");
scanf("%s",&s2);
if(strstr(s1, s2)!=NULL)
    printf("sub-string %s found in %s",s2 ,s1);
else
    printf("can not find sub-string %s found in %s",s2, s1);
return 0;
}

```

自我練習

1. 寫一程式，可以於指定字串中，刪除指定子字串。
2. 請寫一程式，可輸入兩個子字串 1 與子字串 2，若原字串包含子字串 1，則用子字串 2 取代子字串 1。
3. 大部分的登入系統，爲了防止使用者寫程式強迫一直登入，所以會出現亂數要求使用者輸入。請寫一程式，可以產生 5 個字元，此字元僅含小寫英文字元，然後由使用者輸入此亂數，若輸入正確，才可進入系統。本例假設登入後，才可使用範例 2_5a 的面積計算。

※範例11-3b

大部分的程式語言都有類似 Left(a,n)(從左邊取 n 個字元)、Mid(a,m,n)(從索引編號 m 開始，取長度 n 個字元)與 Right(a,n)(取最右邊 n 個字元)，本例示範 Left(a,n) 與 Mid(a,m,n)，Right(a,n) 請自我練習。(索引編號從 0 開始)

輸出結果

```

AB
CDEF

```

程式列印

```

#include <stdio.h>
#include <stdlib.h>
int main()
{
    char s1[]="ABCDEFGH";

```

```

char s2[10];
//left
int n=2;//從左邊取長度2個
int i;
for (i=0;i<n;i++){
    s2[i]=s1[i];
}
s2[n]='\0';//要幫忙放字串結束字元
printf("%s\n",s2);
//mid
int m=2;n=4;//從索引m開始取4個
for (i=m;i<m+n;i++){
    s2[i-2]=s1[i];
}
s2[m+n-2]='\0';//要幫忙放字串結束字元
printf("%s\n",s2);
return 0;
}

```

※ 自我練習

1. 請寫程式完成 Right(a,n) 功能 (取最右邊 n 個字元)。
2. 請寫一個程式，可檢查四星彩的中獎情形。例如，開獎號碼若為 8642，若輸入 8642 則為四星正彩，8641 為前 3 彩，8601 為前 2 彩，3242 為後 2 彩，3642 為後 3 彩。
3. 請寫一程式，可將同一發票中獎號碼全部儲存，然後使用者一一輸入自己的號碼，即可自動對獎，並告知中獎號碼。

範例11-3c

假設有資料如下：

Mary、Hand、洪煥維、陳惠敏、王一平

請寫一個程式，可以將以上資料排序輸出。排序的方式為由小而大，英文先輸出，英文按字母，中文按筆畫的多寡，都是由小而大。

執行結果

Before Sort:	Mary	Hand	洪煥維	陳惠敏	王一平
After Sort:	Hand	Mary	王一平	洪煥維	陳惠敏

程式列印

1. C 語言沒有 `string` 型態，字串使用字元陣列，如以下程式的 `char a[10]`。
2. C 語言沒有字串比較運算子 (`>`, `>=`, `<`, `<=`, `==`, `!=`)，而是要用本單元 `strcmp()` 函式。
3. C 語言字串不能用指派運算子 (`=`)，而是要用本單元 `strcpy()` 函式。

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main()
{
    char a[][10]={"Mary", "Hand", "洪峻維", "陳惠敏", "王一平"};
    int i, j;
    int n=5;
    // 輸出全部
    printf("Before Sort:\n");
    for (i=0; i<=n-1; i++){
        printf("%10s", a[i]);
    }
    printf("\n");
    //排序, 由小而大, 英文按字母, 中文按筆畫的多寡
    char t[10];
    for(i=0; i<=n-1-1; i++){
        for (j=0; j<=n-i-1-1; j++){
            if (strcmp(a[j], a[j+1])>0){ //字串比較, 請看11_3
                strcpy(t, a[j]); //字串複製, 請看11_3
                strcpy(a[j], a[j+1]);
                strcpy(a[j+1], t);
            }
        }
    }
    printf("After Sort:\n");
    for (i=0; i<=n-1; i++){
        printf("%10s", a[i]);
    }
    return 0;
}
```

字串與數值轉換

字串與數值轉換都放在 `<stdlib.h>`，使用前必須在程式前加上

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

1. `atoi`: 將字串轉整數。例如：

```
char a[10]="12345";  
int b=atoi(a);  
b++;  
printf("%d\n",b);//12346
```

結果是 12346。

2. `atof`: 將字串轉為浮點數。例如：

```
char c[10]="123.45";  
float d=atof(c);  
d=d+5;  
printf("%6.2f\n",d);//128.45
```

結果是 128.45。

3. `itoa`: 將數值轉為指定進位的字串。例如：

```
int num=127;  
char f[10] ;  
itoa (num,f,10);//轉為十進位字串  
printf("%s\n",f);//127  
printf("%d\n",strlen(f));//3
```

又例如：

```
itoa (num,f,2);//轉二進位字串  
printf("%s\n",f);//01111111
```

結果是 1111111。又例如：

```
itoa (num,f,16);
```



```
printf("%s\n",f);//7F
```

結果是 7F。

自我練習

1. 請鍵入以下程式，寫出執行結果。

```
int main( ){
    int num=127;
    char f[10] ;
    itoa (num,f,10);//轉為十進位字串
    printf("%s\n",f);//127
    printf("%d\n",strlen(f));//3
    char *ptr;
    itoa (num,ptr,10);//轉為十進位字串
    printf("%s\n",ptr);//127
    printf("%d\n",strlen(ptr));//3
    printf("%c\n",*ptr);//1
    ptr++;
    printf("%c\n",*ptr);//2
}
```

11-4 日期與時間函式(Data Function/Time Function)

日期與時間函式都放在 time.h，常用函式說明如下：

1. _strdate

取得系統日期。語法是

```
char * _strdate(char *b)
```

例如：

```
char d[9];
_strdate(d);
printf("%s\n",d);//01/06/19
```

結果是系統現在的時間 01/06/19。

2. `_strtime`

取得系統時間。語法是

```
char * _strtime(char *b)
```

例如：

```
char t[9];  
_strtime(t);  
printf("%s\n",t);//08:32:03
```

結果是系統現在的時間。

3. `clock()`

傳回程式執行後，所經過的滴答數，每 1000 個滴答等於 1 秒。語法是：

```
clock_t clock(void);
```

`clock_t` 型態定義在 `time.h`，是長整數。例如：

```
clock_t ts,te;  
ts=clock();//單位是滴答，1000滴答=1秒  
_sleep(1000);  
te=clock();  
printf("%d\n", (te-ts)/1000);//1
```

又例如：

```
ts=clock();//單位是滴答，1000滴答=1秒  
_beep(523,500);  
_beep(587,500);  
_beep(659,500);  
_beep(698,500);  
te=clock();  
printf("%d\n", (te-ts)/1000);//2
```

4. time()

取得自 1970/1/1 00:00:00 到目前所經過的秒數，型態是長整數。語法是：

```
time_t time(time_t *t)
```

time_t 定義在 time.h，是長整數。例如：

```
time_t t1,t2;//型態是long
t1=time(NULL);//單位是秒
_beep(523,500);
_beep(587,500);
_beep(659,500);
_beep(698,500);
t2=time(NULL);
printf("%d\n",t2-t1);//2
```

範例11-4a

停車費計算。假如只有一個停車位，請寫一程式，可以輸入字元『a』開始計時，輸入字元『b』可計算其停車時間。

輸出結果

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <conio.h>
int main()
{
    char a;
    time_t ts,te;
    int t;
    char b[9],c[9];
    while(1){
        a=getche();
        if (a=='a'){
            ts=time(NULL);
            _strtime(b);
            printf("Start: %s\n",b);
```

```

    }
    if (a=='b'){
        te=time(NULL);
        t=(te-ts)/60;
        _strtime(c);
        printf("Start at %s,End at %s,t=%d minute\n",b,c,t);
    }
}
return 0;
}

```

👉 自我練習

1. 打字速度測試。請寫一程式，可以一直出現小寫字元，讓使用者輸入，直到正確輸入 20 個，然後計算所需時間。
2. 打字速度測試。請寫一程式，可以一直出現小寫字元，讓使用者輸入，直到 1 分鐘結束，輸出使用者正確輸入的字元數。

範例11-4b

假設時間格式是 "hh:mm:ss"，請寫一程式，可以計算兩個時間的時間間隔長度。

👉 執行結果

```

49
18
30

```

👉 運算思維

這是國小數學問題，當『分』不夠時，要向『時』借位，但電腦可不是這樣，電腦計算很強，通通化為秒數或分數，相減，再轉為時間格式就好。

👉 程式列印

```

#include <stdio.h>
#include <stdlib.h>
int main()
{
    char t1[]="18:50:00";

```

```

char t2[]="19:20:00";
long a1,a2;
printf("%d\n",t1[0]); //49 輸出ASCII
a1=(t1[0]-48)*10+(t1[1]-48);
printf("%d\n",a1); //18
a1=((t1[0]-48)*10+(t1[1]-48))*60+(t1[3]-48)*10+(t1[4]-48); //18*60+50
a2=((t2[0]-48)*10+(t2[1]-48))*60+(t2[3]-48)*10+(t2[4]-48); //19*60+20
printf("%d\n",a2-a1); //30
return 0;
}

```

自我練習

1. 同上範例，停車費每小時 30 元，每日最高 150 元，不足 1 小時以 1 小時計，請計算其停車費。
2. 假如日期格式是 "mm/dd/yy"，請寫一程式可以判斷兩個日期的前後（或是否逾期）。例如，d1="01/22/19", d2="01/28/19"，表示逾期。
3. 假如日期格式是 "mm/dd/yy"，請寫一程式可以計算兩個日期的差額。例如，d1="01/22/19", d2="01/28/19"，表示相差 6 天。
- ※ 4. 假如銀行存款是以每日餘額乘以日利率，張三存款簿記錄如下，請計算其 06/20 的利息。請問您的資料結構如何？假設年利率是 2%。

日期	餘額
01/20	30000
03/10	40000
05/08	20000

11-5 自訂函式(Self-defining function)

前面已經介紹一些公用函式，那在程式設計時，常會遇到某些程式片段需要在同一個程式或不同一個程式重複出現多次，如果這些程式片段都分別在每個地方寫一次，那是一件非常浪費時間的事，且會使程式變得冗長而不易閱讀；更糟的是，萬一要調整此程式片段的部分功能，更要至不同的地方修改，造成程式維護困難，此時可透過自訂函式解決以上困難。

自訂函式的使用方式是將此常用的程式片段賦予一個名稱，當程式設計者需要使用此程式片段時，只要使用此名稱即可呼叫此程式片段，此程式片段即稱為自訂函式，又稱為副程式（Subroutine）。其次，自訂函式完成之後可交由不同的程式呼叫使用，以節省程式設計者的時間。例如：已在前面章節所介紹過的數值函式、字元函式及字串函式等等，這些稱為公用內儲函式，若公用內儲函式無程式設計者要求的功能，此時就可使用本章的自訂函式。其次，自訂函式同樣可讓別的使用者使用。

♂ 函式之原型宣告與呼叫

在本章之前已經介紹了基本資料型態、陣列型態，本節將介紹函式型態。同理，使用函式前一樣要宣告函式型態，但函式的宣告較前面幾種複雜，它包含有函式的傳回型態、函式名稱、參數型態與參數名稱，所以另稱為函式原型（Prototype），而不稱為函式宣告。（雖然若說函式宣告，別人也知道指的是函式原型宣告，但較少人使用函式宣告）。函式的使用步驟是，步驟一：函式原型宣告，步驟二：實作函式本體，步驟三：呼叫函式。分別說明如下：

步驟一：函式原型宣告

函式原型宣告之簡要語法如下：

```
傳回值型態  函式名稱 ([ 參數型態1 參數名稱1, ...]....);
```

以上語法說明如下：

1. 傳回值型態為函式所要傳回的資料型態，可以是任一基本資料型態（int、char 及 double 等）、結構及指標等，若無傳回值，則用 void 表示。
2. 函式名稱的命名同 4-1 節的識別字命名規則。
3. 參數型態與參數名稱必須成對出現。參數為主程式與函式傳遞資料或回傳執行結果的窗口，若無資料傳遞則括號內空白即可。
4. 函式原型宣告的位置，亦影響其有效範圍，其有效範圍同一般變數的宣告。

5. 以下程式片段為函式原型宣告，其意義是函式傳回一個整數型態，函數名稱為 `add`，且傳遞兩個整數型態給函式本體。

```
int add(int a, int b);
```

其次，函式原型宣告的變數名稱可省略，例如：以上敘述亦可簡寫如下：

```
int add(int , int );
```

步驟二：實作函式本體

實作函式本體的簡要語法如下：

```
傳回值型態  函式名稱  ( [ 參數 1, 參數 2... ] ){
    [ 敘述區塊; ]
    [ return ( 運算式); ]
    [ 敘述區塊; ]
}
```

以上語法說明如下：

1. 傳回值型態與函式名稱須與函式原型宣告相同。
2. 每個函式可放在 `main` 函式前面或後面均可，若放在 `main()` 前面則函式原型宣告還可省略。其次，每個函式的地位均相同，所以不可以在函式中再定義其它函式。
3. 函式原則上從左大括號『{』執行到右大括號『}』，但若中途有特殊原因要離開函式，可用 `return` 提早離開。
4. 參數 (Parameter) 有些書亦稱為引數 (Argument)。函式呼叫的參數名稱與實作函式本體的參數名稱可不同，但其資料具有傳遞性，也就是函式呼叫的參數將會傳遞給函式本體，至於函式本體的運算結果是否回傳至主程式，那就得依參數的傳遞方式了。
5. 以下程式片段，可將所傳來的兩個參數相加並傳回。

```
int add(int a, int b){
    int c;
    c=a+b;
    return (c);
}
```

步驟三：呼叫函式

呼叫函式的簡要語法如下：

```
[ 傳回值 =]  函式名稱([ 參數 1, 參數 2, ...]);
```

以上語法說明如下：

1. 函式若無傳回值（void），則函式名稱前的傳回值可省略。
2. 主程式呼叫函式與函式原型宣告的參數稱為實際參數（Actual Parameter），函式本體的參數稱為形式參數（Formal Parameter），且兩者的名稱可以不同。實際參數會將參數值傳給形式參數，而形式參數是否將值傳回，則要看參數傳遞方式，參數的傳遞方式有傳值、傳址及傳參考。
3. 以下程式片段可呼叫 add 函式，傳回值皆為 8：

```
c=add(6,2);
```

或

```
m=6; n=2;  
c=add(m,n);
```

以上三個步驟，全部程式如下：

```
#include <stdio.h>  
#include <stdlib.h>  
//1.函數原型宣告  
int add(int a, int b);  
int main()  
{  
    int m=6,n=2;  
    int c;  
    //3.呼叫函式  
    c=add(m,n);  
    printf("%d\n",c);  
    c=add(6,2);  
    printf("%d\n",c);  
}
```



```
        return 0;
    }
    //2.實作函式實體
    int add(int a, int b){ //a,b分別按照順序接收m,n
        int c;
        c=a+b;
        return (c);
    }
```

若將函數實體放在 main() 前面，可省略函數原型宣告，程式如下：

```
#include <stdio.h>
#include <stdlib.h>
//函式放在前面，可省略函式原型宣告
//1.實作函式實體
int add(int a, int b){
    int c;
    c=a+b;
    return (c);
}
int main()
{
    int m=6,n=2;
    int c;
    //2.呼叫函式
    c=add(m,n);
    printf("%d\n",c);
    c=add(6,2);
    printf("%d\n",c);
    return 0;
}
```

範例11-5a

試寫一程式，計算 C_n^m 的值。

提示： $C_n^m = \frac{m!}{n!(m-n)!}$ 。

 程式列印

1. 以下程式未使用函式，階乘共需重複寫三次，若要修改函式內容，則要到處修改，若沒有修改到，則造成程式的不一致。

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int m=5, n=3, m1, m2, m3, m4, i;
    m1=1;
    for(i=1; i<=m; i++)
        m1=m1*i;
    m2=1;
    for(i=1; i<=n; i++)
        m2=m2*i;
    m3=1;
    for(i=1; i<=m-n; i++)
        m3=m3*i;
    m4=m1/(m2*m3);
    printf("%d",m4);
    return 0;
}
```

2. 以下以函式改寫如下，萬一要修改函式，也是在同一個地方修改。

```
#include <stdio.h>
#include <stdlib.h>
// 1. 實作函式實體
int sum(int m){
    int m1=1;
    int i;
    for(i=1; i<=m; i++)
        m1=m1*i;
```

```

        return(m1);
    }
    int main()
    {
        int m=5, n=3, m1, m2, m3, m4;
        m1=sum(m); // 1. 函式呼叫
        m2=sum(n); // 1. 函式呼叫
        m3=sum(m-n); // 1. 函式呼叫
        m4=m1/(m2*m3);
        printf("%d",m4);
        return 0;
    }
}

```

自我練習

1. 試寫一程式，使用函式完成 $1! + 2! + 3! + 4! + \dots + n!$ (n 可由使用者輸入)

參數的傳遞

C 語言參數的傳遞有兩種情形，第一種是傳值 (Call by value)、第二種是傳址 (Call by address)，分別說明如下：

► 傳值呼叫

參數的傳遞若是傳值，則是一種單向的傳遞方式，參數僅能由主程式傳至函式本體，函式本體並不能藉由參數傳回值。也就是使用傳值參數至多能用函式名稱傳回一個值，若要傳回兩個以上的值，就要使用傳址或傳參考。

以傳值方式傳遞參數時，被傳遞的變數值通常只是被複製到其對應的參數，在函式執行過程，主程式中的變數之值並不會受到影響，其優點是保護主程式的值不會隨便被更動。例如：範例 11-5b 的 x, y 雖在函式有更動其內容，但並不回傳主程式。

範例11-5b

示範傳值呼叫。

```
void swap(int, int); //函式原型宣告
int main() {
    int x=3, y=4;
    swap(x, y);
    printf("3:x=%d,y=%d\n", x, y); //3,4
    return 0;
}
void swap(int p, int q) { //p接收x, q接收y
    int t;
    printf("1:p=%d,q=%d\n", p, q); //3,4
    t=p; p=q; q=t; //交換
    printf("2:p=%d,q=%d\n", p, q); //4,3
}
```

以上傳址呼叫，函式原型宣告、函式呼叫、函式原型如下表：

函式原型宣告	void swap(int, int);
函式呼叫	swap(x, y);
函式原型	void swap(int p, int q)

► 傳址呼叫

傳址呼叫參數是一種雙向的參數傳遞，因為傳址是共用記憶體位址，主程式不僅可以將參數傳至函式本體，函式本體內若參數的內容有所變動，亦可將變動的結果傳回主程式。

範例 11-5c 是傳址呼叫的寫法，x,y 在函式的更動，都會回傳主程式。

範例11-5c

示範傳址呼叫。

```
void swap(int *, int *); //函式原型宣告
int main() {
    int x=3, y=4;
    swap(&x, &y);
}
```

```

    printf("3:x=%d,y=%d\n",x,y); //4,3
    return 0;
}
void swap(int *p, int *q){ //p接收x,q接收y
    int t;
    printf("1:p=%d,q=%d\n", *p, *q); //3,4
    t=*p; *p=*q; *q=t; //交換
    printf("2:p=%d,q=%d\n", *p, *q); //4,3
}

```

以上傳址呼叫，函式原型宣告、函式呼叫、函式原型如下表：

函式原型宣告	void swap(int *, int *);
函式呼叫	swap(&x, &y);
函式原型	void swap(int *p, int *q)

♂ 陣列的傳遞

陣列的名稱是指向陣列的開始位址，所以呼叫函式時，只要將陣列名稱傳給函式即可，且是一種傳址呼叫，陣列的執行結果將可回傳主程式。若為確保陣列在函式中不被修改，則可在函式本體與函式原型宣告的陣列型態前加 `const`。呼叫函式的語法如下：

函式名稱（陣列名稱）；

以下式子是傳遞 a 陣列至函式本體。

```

int a[5], n=5;
getarray(a, n); //請留意不要中括號

```

陣列函式本體的語法如下：

```

傳回值型態 函式名稱（陣列型態 陣列名稱[ ]）{
    敘述區塊；
}

```

以下敘述是函式本體的實例。

```
void getarray (int a[], int n){
}
```

函式原型的宣告同函式本體的實作，但同一般變數，可省略陣列名稱如下：

```
傳回值型態 函式名稱 ([ 陣列型態] );
```

以下敘述是傳遞陣列的函式原型宣告實例。

```
void getarray (int [], int);
```

範例11-5b

請完成以下陣列處理函式。

```
Before sort:
 99  88  77  66  55
After sort:
 55  66  77  88  99
```

程式列印

```
#include <stdio.h>
#include <stdlib.h>
void print(const int a[],int n){
    int i;
    for (i=1;i<=n;i++){
        printf("%5d",a[i]);
    }
    printf("\n");
}
void swap(int *p, int *q){//傳址呼叫，p,q值才能傳回
    int t;
    t=*p; *p=*q; *q=t;
}
void sortarray(int a[],int N){
    int i,j;
```

```

        for(i=1; i<=N-1; i++)
            for(j=1; j<=N-i; j++)
                if(a[j]>a[j+1])
                    swap(&a[j], &a[j+1]);
    }
int main()
{
    int a[]={0, 99, 88, 77, 66, 55};
    int i, j;
    const int N=5;
    printf("Before sort:\n");
    print(a,N);
    //Sort
    sortarray(a,N);
    //output
    printf("After sort:\n");
    print (a,N);
}

```

自我練習

- 若要利用 C 語言寫一個 BMI 函式，此一函式接收傳入兩個整數資料，經計算後回傳的數值必須有小數點後至少兩位數精確度，BMI 函式的原型宣告應為下列何者？（111 年統測）

(A) double BMI ();	(B) void BMI (int, int);
(C) int BMI (int, int);	(D) float BMI (int, int);
- 假設有資料如下：

33,99,88,66,55,22,88

請完成以下函式。

 - 列印陣列。
 - 求極大值。
 - 將全部元素乘以 2。
 - 刪除重複值。

♂ 變數與常數的有效範圍

在一個大型程式裏，一個程式是由一或數個函式組合而成。這些函式通常是由許多人合力完成，爲了避免彼此變數互相干擾，例如張三設 `stunum=15`，李四又設 `stunum=20`，則必然造成無法預期的錯誤，所以爲了防止變數互相干擾，才有變數的有效範圍（又稱爲變數生命週期），茲將變數的有效範圍敘述如下：

任一變數的宣告，若無特殊聲明，均屬於區域變數 (Local Variable)，其有效範圍僅止於該變數所在的程式區塊。例如，若宣告於函式中，則它的有效範圍僅止於該函式，其它的函式均無法取得該區域變數的值；若宣告在 `main()` 的前面，才是全域變數 (Global Variable)。例如，以下程式，全域的 `a` 有效範圍是 `main()`、`setup()`，`c` 的有效範圍則是 `main()`。其次，`main()` 與 `setup()` 內雖然都有變數 `b`，但此 `b` 在不同函式只是同名而已，它們各自佔有不同的記憶體，且其值不會互相干擾。就如同不同家庭會有相同的人名，但他們一定可配到不同身份證號碼，兩者不會互相干擾。

```
#include <stdio.h>
#include <stdlib.h>
int a=6; //全域變數
void setup(){
    int b=1; //區域變數
}
int main(int argc, char *argv[]) {
    int b=2; //區域變數
    int c=3; //區域變數
    printf("%d\n",a); //使用全域
    return 0;
}
```


自我練習

請鍵入以下程式，並觀察執行結果。

程式	執行結果
<pre> #include <stdio.h> #include <stdlib.h> int a=6; void setup(){ int b=1; a=3; printf("b=%d\n",b); } int main(int argc, char *argv[]) { int b=2; int c=3; printf("a=%d\n",a); //使用全域 printf("b=%d\n",b); setup(); printf("b=%d\n",b); printf("a=%d\n",a); return 0; } </pre>	

以下程式，又於 setup() 內重複宣告 int a；那也合理，那此 a 的生命，僅止於 setup()，不會影響全域變數。

```

#include <stdio.h>
#include <stdlib.h>
int a=6;
void setup(){
    int a=3;
    printf("b=%d\n",a); //使用區域
}
int main(int argc, char *argv[]) {
    printf("a=%d\n",a); //使用全域
    setup();
    printf("a=%d\n",a);    //使用全域
    return 0;
}

```

11-6 函式應用實例

猜數字

有一種遊戲稱為幾 A 幾 B 的猜數字遊戲，兩個人對玩，互相猜對方預先寫下的四位數（四位數中的阿拉伯數字不可重覆），若所猜的數字與對方位置相同者為 A，數字相同，位置不對，則稱為 B。例如對方預寫的數字為 6713，若猜 6731 則應回應 2A2B，若猜 7652 則應回應 0A2B。試寫一程式，電腦自動產生四位數的亂數，再讓使用者猜的一種遊戲程式。（電腦應逐一回應使用者已猜的狀況）

範例11-6a

示範以上猜數字。

執行結果

```
The rand is:3024
input your 1 guess:1234
1 A 2 B input your 2 guess:5678
0 A 0 B input your 3 guess:9012
1 A 1 B input your 4 guess:4321
1 A 2 B input your 5 guess:4203
0 A 4 B input your 6 guess:2034
2 A 2 B input your 7 guess:3024
4 A 0 B Bingo
Total guess is :7
```

程式列印

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int main()
{
    char c[]={'0','1','2','3','4','5','6','7','8','9'};
    char d[5], e[5];
    int n;
    int i, j;
    int a,b;
    int r,temp;
```

```
srand(time(0)); //隨機選取亂數起點
//洗牌
for(i=0; i<=9; i++) {
    r=rand()%10;
    temp=c[i]; c[i]=c[r]; c[r]=temp;
}
//取前四個為亂數
for(i=0; i<=3; i++)
    d[i]=c[i];
d[4]='\0'; //記得放結束字元
printf("The rand is:%s\n",d); //測試完成再去掉
n=0; //
do
{
    n++; //猜測次數
    printf("input your %d guess:",n);
    scanf("%s",e);
    a=0;b=0;
    for (i=0;i<=3;i++){
        if (d[i]==e[i])
            a++;
        for (j=0;j<=3;j++){
            if (d[i]==e[j])
                b++;
        }
    }
    b=b-a;
    printf("%d A %d B ",a,b);
}
while (a!=4);
printf("Bingo\n");
printf("Total guess is :%d",n);
return 0;
}
```

 自我練習

- ※ 1. 刪除法。先將所有可能的答案通通列出來，在逐步將所有不可能的答案一一刪掉，最後剩下的值即為所求，此稱為刪除法。例如，同上題，剛剛是人腦猜，人腦有人腦的運算思維，但是電腦有電腦的運算思維，既然是電腦程式設計，那就要利用電腦的強項，才能事半功倍，以下示範電腦用刪除法來猜，使用者必須逐一回應電腦已猜的狀況。電腦猜數字若使用以上刪除法，則演算法如下：
- (1) 使用陣列列出 1000 至 9999 的四位數。
 - (2) 逐一刪除阿拉伯數字重複者，例如 1001 或 3343 等。
 - (3) 於剩下的可能數字中，挑最小的當作猜值，本例為 1023。(使用者應回應幾 A 幾 B，本例假設使用者的數字是 7203，所以回應 1A2B)
 - (4) 於剩下的可能值中，使用臆測值，本例為 1023 逐一比對，將不是 1A2B 者都全部刪除。(為什麼呢？因為答案 7203 也會是 1A2B，此即為刪除法，逐一將不可能的答案，逐一刪除。)
 - (5) 重複 (3)、(4) 兩個步驟，直到剩下最後一個，就是答案了，或使用者回應 4A0B 為止。這題演算法已經很詳細，很有趣，也很有成就感，留給大家作。
- ※ 2. 簡單計算機。請寫一個程式，可以當作簡單計算機，可以計算輸入兩個數的 +, -, *, /，輸入格式可為「1023+23」、「321-2」、「23*420」、「96/4」等，僅 4 種運算子，但數字長度不限。

課後習題

本章的另一重點是變數的有效範圍，請鍵入以下程式，並觀察執行結果。

題號	題目	執行結果
1	<pre> const double PI=3.14159;//全域常數 int a;//全域a int main(int argc, char** argv) { int b, c; a=5;//全域a c=5;//main 內 c for (int a =2 ;a<=3;a++){//好習慣 //for 內a,僅for內有效,不會改變全域的a //反過來說,若未宣告a,那就慘了,會影響全域a } if (c>2){ int a; a=1;//if 內a,僅if內有效 } return 0; } void aa(){ int c, d;//c,d 僅aa內有效 c=3;//aa內的c,僅aa內有效 } void bb(){ a=8;//使用全域a } </pre>	
2	<pre> #include <stdio.h> #include <stdlib.h> int a=1; void aa(){ printf("1.a=%d\n",a); int a=2; printf("2.a=%d\n",a); } void bb(){ printf("4.a=%d\n",a); a=3; } </pre>	

	<pre> int main(int argc, char** argv) { aa(); printf("3.a=%d\n",a); bb(); printf("5.a=%d\n",a); return 0; } </pre>	
3	<pre> #include <stdio.h> #include <stdlib.h> const double PI=3.14159;//全域常數 int a;//全域a void aa() { int c, d; c=3;//aa內的c } void bb() { a=8;//全域a } int main(int argc, char** argv) { int b, c; a=5;//全域a c=5;//main 內 c aa(); bb(); for (int a =2 ;a<=3;a++) printf("4.for 內的 a=%d\n",a); if (c>2){ int a; a=1; printf("8.if 內a=%d\n",a); } return 0; } </pre>	

```
4  #include <stdio.h>
    #include <stdlib.h>
    const double PI=3.14159;//全域常數
    int a;//全域a
    void aa()
    {
        int c, d;
        c=3;//aa內的c
        printf("1.aa內的c=%d\n",c);
    }
    void bb()
    {
        a=8;//全域a
    }
    int main(int argc, char** argv) {
        void aa();
        void bb();
        int b, c;
        a=5;//全域a
        c=5;//main 內 c
        aa();
        printf("2.main 的c=%d\n",c);
        bb();
        printf("3.全域a=%d\n",a);
        for (int a =2 ;a<=3;a++)
            printf("4.for 內的 a=%d\n",a);
        printf("5.全域a=%d\n",a);
        for ( a =2 ;a<=3;a++)//誤用
            printf("6.for 內的 a=%d\n",a);
        printf("7.全域a=%d\n",a);
        if (c>2){
            int a;
            a=1;
            printf("8.if 內a=%d\n",a);
        }
        printf("9.全域a=%d\n",a);
        return 0;
    }
```

5	<pre>double Func(int w,int h){//111年統測 double A; A=w*h/2.0; return A; } void main () { int a=5,b=3; double result=0; result=Func(a,b); printf("%lf",result); }</pre>	
6	<pre>int f() { //APCS10503 int p = 2; while (p < 2000) { p = 2 * p; } return p; } void main () { int a=f(); printf("%d",a); }</pre>	
7	<pre>int f(int n){ //APCS10503 int p=0; int i=n; while(i>=1){ p=10-2*i; printf("%d",p); i=i-1; } } void main() { f(4); }</pre>	

8	給定函式 F()，執行 F() 時哪一行程式碼可能永遠不會被執行到？ //APCS10603 (A) a = a + 5; (B) a = a + 2; (C) a = 5; (D) 每一行都執行得到 <pre> void F (int a) { while (a < 10) a = a + 5; printf("%d\n",a); if (a < 12) a = a + 2; printf("%d\n",a); if (a <= 11) a = 5; printf("%d\n",a); } int main() { F(1); return 0; } </pre>	
9	<pre> int main() { int n=100; int x=0; for (int i=1;i<=100;i++){ x=0; for (int k=1;k<=i;k=k*2) x=x+1; printf("i=%d,x=%d\n",i,x); } return 0; } </pre>	
10	<pre> int main() { //APCS10603 int n=5; int i,j; int s=0; </pre>	

	<pre> for (i=1;i<=n;i++){ for(j=i;j<=n;j++){ s=s+1; } } printf("s=%d",s); return 0; } </pre>	
11	<pre> int F (int n) { //APCS10603 int x = 0; for (int i=1; i<=n; i=i+1) for (int j=i; j<=n; j=j+1) for (int k=1; k<=n; k=k*2) x = x + 1; return x; } int main() { int n=1; int k=F(n); printf("%d",k); return 0; } //寫出n=1~32的結果 </pre>	
12	<pre> int f1(int a[],int value){ //APCS10503 int r_value=-1; int i=0; int j=0; //執行次數 while(i<100){ j++; if (a[i]==value){ r_value=i; break; } i=i+1; } printf("j=%d\n",j); return r_value; } int f2(int a[],int value){ int r_value=-1; int low=0,high=99; int mid; int j=0; //執行次數 </pre>	

	<pre> while (low<=high){ j++; mid=(low+high)/2; printf("j=%d,low=%d,mid=%d,high=%d\n",j, low,mid,high); if (a[mid]==value){ r_value=mid; break; } else if (a[mid]<value) { low=mid+1; } else{ high=mid-1; } } printf("j=%d\n",j); return r_value; } int main() { int a[100]; int i; for (i=0;i<=99;i++){ a[i]=3*i+1; } printf("r_value=%d\n",f1(a,100)); printf("r_value=%d\n",f2(a,100)); return 0; } </pre>	
13	<pre> void F(void) { char t, item[]={'2', '8', '3','1', '9'}; int a, b, c, count=5; for (a=0; a<count-1; a=a+1) { c=a; t=item[a]; printf("c=%d,t=%c",c,t) ; for (b=a+1; b<count; b=b+1) { //cout<<a<<","<<b<<","<<c<<","<<t<<endl; if (item[b]<t) { </pre>	

	<pre> c=b; t=item[b]; } if ((a==2) && (b==3)) { printf ("%c %d\n", t, c); } } printf("c=%d,t=%c\n",c,t) ; } } int main(int argc, char** argv) { F(); return 0; } </pre>	
14	<pre> int A[8]={0, 2, 4, 6, 8, 10,12, 14}; //APCS10603 int Search (int x) { int high=7; int low=0; if (x>A[high]) return A[high]; while (high>low) { int mid=(high+low)/2; if (A[mid]<=x) { low=mid+1; } else { high=mid; } } return A[high]; } int main(int argc, char** argv) { printf("%d\n",Search(-1)); //0 printf("%d\n",Search(0)); //2 printf("%d\n",Search(10)); //12 printf("%d\n",Search(16)); //14 return 0; } </pre>	

15	給定一個 1x8 的陣列 A，A = {0, 2, 4, 6, 8, 10, 12, 14}。函式 Search(x) 真正目的是找到 A 之中大於 x 的最小值。然而，這個函式有誤。請問下列哪個函式呼叫可測出函式有誤？ (A) Search(-1) (B) Search(0) (C) Search(10) (D) Search(16) <pre> int A[8]={0, 2, 4, 6, 8, 10,12, 14}; //APCS10603 int Search (int x) { int high=7; int low=0; while (high>low) { int mid=(high+low)/2; if (A[mid]<=x) { low=mid+1; } else { high=mid; } } return A[high]; } </pre>	
16	<pre> int f (int a[], int n) { int index = 0; for (int i=1; i<=n-1; i=i+1) { if (a[i] >= a[index]) { index = i; } } return index; } int main() { int a[10]={1,3,9,2,5,8,4,9,6,7}; int k=f(a,10); printf("%d\n",k); // return 0; } </pre>	

17	<pre> int s=1; // 全域變數 void add (int a) { int s=6; //區域變數，其有效範圍僅止於此 //不會回傳影響全域變數 printf("2:a=%d",a); //我增加的 for(; a>=0; a=a-1) { //a用上面 printf("%d,,\n", s); s++; printf("%d,,, \n", s); } } int main () { printf("1:%d, \n", s); add(s); printf("3:%d, \n", s); s=9; printf("4:%d, \n", s); return 0; } </pre>	
18	<pre> #include <iostream> //APCS using namespace std; int g1=30, g2=20; //全域變數 int f1(int v) { int g1=10; //不影響全域變數 return g1+v; } int f2(int v) { int c=g2; cout<<"v="<<v<<endl; cout<<"c="<<c<<endl; cout<<"g1="<<g1<<endl; v=v+c+g1; g1=10; //會影響全域變數 c=40; return v; } int main(int argc, char** argv) { g2=0; g2=f1(g2); //g1=30, g2=10 cout<<"g1="<<g1<<endl; //30 </pre>	

	<pre> cout<<"g2="<<g2<<endl;//10 int a=f2(g2); cout<<"g1="<<g1<<endl;//10 cout<<"g2="<<g2<<endl;//10 cout<<"a="<<a<<endl;//50=10+10+30 cout<<f2(a)<<endl;//70=50+10+10 //printf("%d", f2(f2(g2))); return 0; } </pre>	
19	<pre> int main() { double x1=4.51,x2=-4.51; printf("%d\n",abs(x1)); printf("%lf\n",fabs(x2)); printf("%lf\n",ceil(x1)); printf("%lf\n",ceil(x2)); printf("%lf\n",floor(x1)); printf("%lf\n",floor(x2)); printf("%lf\n",trunc(x1)); printf("%lf\n",trunc(x2)); printf("%lf\n",rint(x1)); printf("%lf\n",rint(x2)); printf("%lf\n",round(x1)); printf("%lf\n",round(x2)); return 0; } </pre>	
20	<pre> int g1=30, g2=20;//全域變數APCS10603 int f1(int v) { int g1=10;//不影響全域變數 return g1+v; } int f2(int v) { int c=g2; printf("v=%d\n",v); printf("c=%d\n",c); printf("g1=%d\n",g1); v=v+c+g1; g1=10;//會影響全域變數 c=40; } </pre>	

```
        return v;
    }
    int main() {
        g2=0;
        g2=f1(g2); //g1=30,g2=10
        printf("g1=%d\n",g1);
        printf("g2=%d\n",g2);
        int a=f2(g2);
        printf("g1=%d\n",g1);
        printf("g2=%d\n",g2);
        printf("a=%d\n",a);
        printf("f2(a)=%d", f2(a));
        return 0;
    }
```


物件導向程式設計

學習綱要

- 📖 12-1 C++ 程式設計
- 📖 12-2 類別
- 📖 12-3 繼承
- 📖 12-4 應用實例

學習目標

- 📖 1. 瞭解 C++ 語言與 C 語言的差異。
- 📖 2. 能以類別寫出程式。
- 📖 3. 能以繼承寫出程式。
- 📖 4. 能應用類別、繼承、多型等實作物件導向應用實例。

人類之所以會是萬物之靈，其中一個主要原因是人類可以在錯誤中成長。物件導向的程式設計（Object Oriented Programming, OOP）也是人們在程式語言中逐漸累積的成果，這個觀念在 1970 年代就已提出，只是當時未受到任何程式語言支持。現在，OOP 則已是所有程式語言的標準配備，爲了說明 OOP 大行其道的原因，筆者將程式語言的發展分爲 3 個時期，分別爲非程序導向 (Non-procedure-oriented language)、程序導向 (Procedure-oriented language) 及物件導向 (Object-oriented programming)，三者分別說明如下。

非程序導向

早期的程式語言，並沒有內儲副程式（又稱程式庫）。當開發新的應用程式時，如果某一功能與之前寫過的程式相近，則會將此段已完成的程式整段複製，並稍加修改即可重新加以利用。但是，這些程式的分身包括本尊，自從複製出來以後就開始以各自的方式發展，結果造成各版本的差異越來越大，這些程式很難分辨誰複製誰，彼此之間也難再共用某些程式碼，當遇到錯誤或欲新增功能時，更是很難逐一修改所有程式。

程序導向

爲了解決以上程式共用問題，各編譯器廠商便開始提供一些常用的函式，較有規模的軟體設計公司亦會將常用的函式集中在一個函式庫 (Function library)，旗下的軟體產品一律呼叫這些標準的函式庫，而不是從函式庫複製出來修改，此即爲程序導向的程式設計。程序導向與非程序導向相比，的確解決了程式共用的問題，但是，人們並不以此爲自滿，有些問題還是不夠順暢。例如：有些函式庫會隨著人們需求的增加而有不同版本，當某些函式功能增加時，只好重新取函式庫的函式加以修改，並賦予新的名稱。如此日積月累，我們的函式庫已有許多函式。這些函式有的功能相近、有的是前後版本不同、有的函式裡的變數來龍去脈不明，造成使用者的混亂。爲了突破以上瓶頸，於是有物件導向的發展，以解決以上程序導向的不足。

物件導向

程序導向中的函式，存有許多解決問題的函式（函式在目前的物件導向中稱為方法），它偏重在方法的解決。但是，人類的生活方式不僅是單純的行為描述，更存在著屬性 (Attributes) 的記載。例如：當在描述一個人時，通常描述如下：「他的名字是洪煥維，身高 172、體重 70，具有滾進、游泳及跑步的能力」；又例如：描述一輛車子時，其描述如：「它的名字是 SENTRA，排氣量是 1600c.c.，耗油量是每公里 0.1 公升，且具有每小時 120 公里的移動能力」。以上人與車即稱為「物件」(Object)，名字、身高、排氣量則稱為「屬性」，而滾進、游泳、跑步、移動則稱為「方法」(Method)。既然真實世界是以物件描述物種，程式設計亦不應侷限在狹隘的函式，僅偏重解決問題的方法，而是應以物件的宏觀角度撰寫程式。所以，基於物件導向的新觀念，程式開發工具即制訂一種新的型態，此型態稱為類別 (Class)。每一個類別都有屬於自己的屬性與方法，也就是我們已將眾多的函式依照類別存放，如此可解決目前日益龐大的函式命名與函式取用的困擾。例如：程序導向的時代，關於開門的函式即有電梯開門、汽車開門、房子開門等數種開門的方法，如此徒增命名與取用的困擾。但在物件導向的領域中，開門這個方法是附在相對應的類別。例如：於電梯類別有電梯的開門，汽車類別有汽車的開門，房子類別有房子的開門方法，各類別的方法名稱都叫「開門」，撰寫程式時，也是「電梯.開門」，「汽車.開門」，或是「房子.開門」（物件與方法、屬性之間以點 (.) 運算子連結），如此既可簡化程式的撰寫，亦可減少程式出錯的機會。但是，在程序導向的領域中，所有的函式都集中，就有可能用錯方法。例如：用開電梯門的方法去開汽車的門，結果當然是錯的。

物件導向的程式設計裡，先定義類別，其次，類別的變數即稱為物件，物件亦稱為類別的實現或類別的樣例化 (Instance)。也就是說，類別就像是一個酒瓶或機器人的模型，在這些模型裡面已設計好他所具備的屬性與方法，當您需要酒瓶或機器人時，只要依照這些模型樣例一個或多個酒瓶或機器人即可。其次，物件導向亦提出三個理念，分別是物件的封裝 (Encapsulation)、繼承 (Inheritance) 及多型 (Polymorphism)，以解決程序導向的不足。以上觀念，將在本章說明。

12-1 C++ 程式設計

由於物件導向理念的需求與衝擊，在 1980 年代有物件導向 C++ 語言的誕生，由於 C 語言的執行速度很好且使用率很高，所以 C++ 就繼承 C，且加入物件導向的類別等元素。既然是繼承，表示原來 C 的指令都可以使用。例如：請在 Code::Blocks 新增 C++ 程式，C++ 程式架構如下：

```
#include <iostream>
using namespace std;
int main(){
    cout << "Hello world!" << endl;
    return 0;
}
```

先執行以上程式，若有輸出「Hello world!」，請將「cout << "Hello world!" << endl;」刪除，接著，鍵入 2-4a 程式如下，並執行。

```
#include <iostream>
using namespace std;
int main(){
    int a=15,b=8,c;
    c=a*b;
    printf("面積=%d\n",c);
    return 0;
}
```

也可以在 C++ 直接開啓或鍵入範例 2-4a，程式如下，所有 C 語言標頭檔、輸出入與指令都可在 C++ 環境執行，此稱為「繼承」。

```
#include <stdio.h>
#include <stdlib.h>
int main(){
    int a=15,b=8,c;
    c=a*b;
    printf("面積=%d\n",c);
    return 0;
}
```

以下程式則以 C++ 指令，重做範例 2-4a 的面積計算。

```
#include <iostream>//載入輸出入標頭檔
using namespace std;//指派使用std命名空間
int main()
{
    int a,b,c;
    cin>>a>>b;//輸入a b
    cout<<a<<endl;//輸出a
    cout<<b<<endl;//輸出b
    c=a*b;
    cout<<c;
    return 0;
}
```

♂ 標頭檔

本例因為使用 cout 物件，所以要預先載入其函式庫

```
#include <iostream>
```

C++ 新增的輸出入物件為 cout 與 cin，其函數庫為 iostream 則用來取代

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

函式庫的 scanf() 與 printf() 函式。C 語言的函式庫均為 .h，例如：數學函式庫為 math.h。為了方便區分，C++ 新增函式的函式庫，則均沒有 .h，例如：C++ 新增 cmath 用來包含與取代 math.h。

♂ 命名空間

using namespace std 為使用命名空間 std，本例若不指定命名空間，也就是不寫 using namespace std; 則每次鍵入 cout、cin 時，都要指定命名空間 std::，程式如下：

```
#include <iostream>
int main() {
    int a;
    std::cout<<"input a:";
    std::cin>>a;
    std::cout<<a;
    return 0;
}
```

cout

cout 物件可輸出指定任何文數字與變數內容，而且不用去管變數的資料型態。其語法如下：

```
cout <<字串或變數 [<<字串或變數] [<<endl] ;
```

語法中的中括號 "[]" 表示可省略，例如：

```
int a=5;
cout<<a;//5
```

又例如：

```
int a=5;
cout <<"a="<<a<<endl;
```

可輸出 a=5。其中第一個用雙引號表示字串內容，將會以字串方式輸出，第二個 a，則是變數，將會輸出對應變數的內容。endl(endline) 代表此列的輸出結束，跳離此列，往後的輸出移至下一列的開頭。例如：

```
cout <<"Good"<<endl;
cout <<"Good";
```

可輸出：

```
Good
Good
```

但是，

```
cout <<"Good";
cout <<"Good";
```

則輸出：

```
GoodGood
```

範例12-1a

示範輸出以上結果。

程式列印

```
#include <iostream>
using namespace std;
int main() {
    cout <<"Good"<<endl;
    int a=5;
    cout <<"a="<<a<<endl;
    return 0;
}
```

自我練習

1. 請於 C++ 環境完成以上程式。
2. 請嘗試將「using namespace std;」去掉，請問程式要如何修改。

cin

cin 物件可輸入文數字至指定的變數。其語法如下：

```
cin>>變數 [>>變數] ;
```

例如：

```
char a;
cin >>a;
cout <<"a="<<a<<endl;
```

可輸入字元至變數 a。又例如：

```
int b;  
cin >>b;  
cout<<"b="<<b<<endl;
```

可輸入整數至變數 b。又例如：

```
char c[5];  
cin >>c;  
cout<<"c="<<c<<endl;
```

可輸入字串至變數 c。以下程式則可連續輸入 3 個變數，輸入時，變數之間請以空白隔開，例如，輸入 a 5 Horng，則三個變數變數 a,b,c 分別得到 a, 5, Horng。

```
cin >>a>>b>>c;
```

例如，以下是使用 C++ 的 cout、cin 重做範例 7-1c 的程式。

```
#include <iostream>  
#include <time.h> // 使用time()  
#include <stdlib.h> // 使用rand()  
using namespace std;  
int main()  
{  
    srand(time(NULL));  
    int a=97+rand()%26;  
    cout<<a<<endl;//輸出a  
    cout<<(char)a;  
    char b;  
    cin>>b;//輸入b  
    if ((int)b==a)//資料型態相同才能比較  
        cout<<"right" //輸出  
    else  
        cout<<"wrong";  
    return 0;  
}
```


 自我練習

1. 請鍵入以下程式，寫出執行結果。

```
#include <iostream>
using namespace std;
int main() {
    char a;
    cin >>a;
    cout <<"a="<<a<<endl;
    int b;
    cin >>b;
    cout<<"b="<<b<<endl;
    char c[5];
    cin >>c;
    cout<<"c="<<c<<endl;
    cin >>a>>b>>c;
    cout <<"a="<<a<<endl;
    cout <<"b="<<b<<endl;
    cout <<"c="<<c<<endl;
    return 0;
}
```

2. 請寫一個 C++ 程式，可以輸入人名、國文成績、數學成績，然後輸出人名與平均分數。

 string 型態

C 語言資料型態並沒有字串，僅使用字元陣列當作字串；C 語言也不能用比較運算子，而是要用比較函式 `strcmp()`。C++ 則新增 `string` 型態，且可以使用「=」當作指派運算子、使用「+」當作字串合併運算子、使用「>,>=,<=,<,<=,<=,!=」當作比較運算子。此即為物件導向的「多型」理念——使用相同的運算子，編譯器在編譯階段，再載入對應的函式，這樣可以減輕開發者的負擔。

 自我練習

1. 請鍵入以下程式，並寫出執行結果。

```
#include <iostream>
using namespace std;
int main() {
    string a="aa",b="bb",c;
    cout<<a<<endl;
    cout<<a[0]<<endl;
    c=a+b;//合併、指派運算子
    cout<<c<<endl;
    cout<<(a>b)<<endl;//比較運算子
    return 0;
}
```

2. 請鍵入以下程式，並寫出執行結果。

```
#include <iostream>
using namespace std;
int main() {
    string a[]={"ab","cd","ef"};
    cout<<a[0]<<endl;
    cout<<a[0][0]<<endl;
    cout<<a[1]<<endl;
    return 0;
}
```

3. 以下是字串以一維陣列儲存，然後排序、輸出的操作。C++ 的程式裡，字串可用 string 型態，可用比較運算子 (>)，可用字串指派運算子 (=)，如以下程式。

```
#include <iostream>
using namespace std;
int main() {
    string a[]={"Hand","Mary","洪峻維","陳惠敏","王一平"};;
    cout<<a[0][0]<<endl;//H
    cout<<a[0][1]<<endl;//a
    cout<<a[1]<<endl;//Mary
    int n=5;
```

```

string t;
int i,j;
for(i=0;i<=n-1-1;i++){
    for (j=0;j<=n-2-i;j++){
        if (a[j]>a[j+1]){//字串比較
            t=a[j];//字串複製
            a[j]=a[j+1];
            a[j+1]=t;
        }
    }
}
for (i=0;i<=n-1;i++)
    cout<<a[i]<<endl;
return 0;
}

```

4. 請寫一程式，可以輸入指定數量 N，然後輸入 N 筆姓名，且排序後輸出。
5. 鍵入以下程式，並寫出執行結果。

```

#include <iostream>//C++
using namespace std;
int main()
{
    string a="ABCD";
    string b[]={"AA1","BB2","CC3","DD4"};
    string c[][2]={{ "AA","90"}, {"BB","80"}, {"CC","70"} };
    string d;
    d=a;
    cout <<a[0]<<endl;//A
    cout <<a<<endl;//ABCD
    cout <<d<<endl;//ABCD
    cout <<b[0][0]<<endl;//A
    cout <<b[0]<<endl;//AA1
    d=b[1];
    cout<<d<<endl;//BB1
    cout <<c[0][0][0]<<endl;//A
    cout<<c[0][0]<<endl;//AA
    d=c[0][1];
}

```

```
    cout<<d<<endl;//90
    return 0;
}
```

6. 以下程式是字串以二維陣列儲存，然後計算總分、計算平均、輸出的操作。

```
#include <iostream>
#include <cstring>//使用C++字串函式
using namespace std;
int main()
{
    string a[][4]={{"Tom","90","80",""},
                   {"John","60","90",""},
                   {"Mary","80","60",""}};

    cout<<a[0][0][0]<<endl;//T
    cout<<a[0][0][1]<<endl;//o
    cout<<a[1][1][0]<<endl;//6
    cout<<a[0][0]<<endl;//Tom
    cout<<a[0][1]<<endl;//90
    cout<<a[1][0]<<endl;//John
    int i,j;
    //輸出
    for (i=0;i<=2;i++){
        for (j=0;j<=3;j++){
            cout<<a[i][j]<<" ";
        }
        cout<<endl;
    }
    //計算總分、平均
    int sum=0,avg;
    for (i=0;i<=2;i++){
        sum=0;
        for (j=1;j<=2;j++){
            sum=sum+stoi(a[i][j]); //C++轉數值
            //sum=sum+atoi(a[i][j]); //C style轉數值，請看11_3
        }
        avg=sum/2;
        a[i][3]=to_string(avg); //C++轉字串
    }
}
```

```

        //itoa(avg,a[i][3],10); //C style轉字串，請看11_3
    }
    cout<<endl;
    //輸出
    for (i=0;i<=2;i++){
        for (j=0;j<=3;j++){
            cout<<a[i][j]<<" ";
        }
        cout<<endl;
    }
    return 0;
}

```

7. 續第 6 點，請寫一程式，可計算每人不及格科數、每科平均、每科不及格人數、輸出依照平均排序。
8. 續第 6 點，請寫一程式，可計算每個人的名次，輸出按照原來的順序排列，但加上名次欄位。
9. 續第 6 點，第 6 點是用陣列，請改用結構重做。
10. 續第 6 點，請寫一程式，可輸入姓名，而查得其平均。
11. 續第 6 點，第 6 點是採用陣列直接指派字串初值，請寫一程式，可以程式一執行，讓使用者輸入以上資料，然後以陣列儲存。

☞ 函式傳參考呼叫

C 語言有傳值與傳址，請看本書範例 11-5b、11-5c，但是當傳值想改為傳址或傳址想改為傳值，函式裡的變數必須到處修改，造成程式開發者的負擔，C++ 爲了改善此一不便，所以有傳參考。

範例12-1b

示範傳參考呼叫。以傳參考重做範例 11-5b 與 11-5c。

👉 程式列印

現在希望參數是雙向傳遞，若使用傳址，則函式內容必須到處修改變數，但若使用傳參考，則只要改變函式宣告就行。例如，以下程式，分別

以傳值、傳址，傳參考的方式重做兩數交換函式，請讀者仔細比較函式原型宣告、主程式與函式寫法的差異。

```
#include <iostream>//C++
using namespace std;
void swap1(int , int );//傳值
void swap2(int *, int *);//傳址
void swap3(int &, int &);//傳參考
int main() {
    int x=3, y=4;
    swap1(x, y); //傳值
    cout<<"1:x="<<x<<" ,y="<<y<<endl;//3,4
    x=3;y=4;
    swap2(&x,&y); //傳址
    cout<<"2:x="<<x<<" ,y="<<y<<endl;//4,3
    x=3;y=4;
    swap3(x,y); //傳參考，謹函式宣告更動
    //請留意傳參考的主程式也一樣
    cout<<"3:x="<<x<<" ,y="<<y<<endl;//4,3
    return 0;
}
void swap1(int p, int q){ //傳值
    int t;
    t=p; p=q; q=t;//交換
}
void swap2(int *p, int *q){ //傳址
    //請留意，函式內變數要更動
    int t;
    t=*p; *p=*q; *q=t;//交換
}
void swap3(int &p, int &q){ //傳參考，謹函式宣告更動
    //請留意，函式內變數不用更動
    int t;
    t=p; p=q; q=t;//交換
}
```

以上傳參考呼叫，函式原型宣告、函式呼叫、函式原型如下表：

函式原型宣告	void swap(int &, int &);
函式呼叫	swap(x, y);
函式原型	void swap(int &p, int &q)

☞ 函式多載 (Function overloading)

於函式的設計裡，有時會有兩種以上的函式功能接近，例如只是參數型態或參數個數不同，是否可以使用相同的函式名稱，以減少程式設計者命名的困擾？答案是肯定的。因為物件導向的程式設計均允許程式設計者在參數型態與參數個數不同時，使用相同的函式名稱，這些相同的函式即稱為函式多載。以下我們分別實作 3 個 add 函式，其參數型態與數目均不同，但都共用一個函式名稱，待執行階段即可依參數的型態與數目執行對應的函式。

1. 兩個整數

```
int add(int a1,int b1) {  
    int sum=a1+b1;  
    return(sum);  
}
```

2. 三個整數

```
int add(int a1,int b1,int c1) {  
    int sum=a1+b1+c1;  
    return(sum);  
}
```

3. 兩個整數，一個實數

```
double add(int a1,int b1,double c1) {  
    double sum;  
    sum=a1+b1+c1;  
    return(sum);  
}
```

其次，於主程式撰寫如下：

```
#include <iostream> //載入輸入輸出標頭檔  
using namespace std; //指派使用std命名空間  
int add(int,int);  
int add(int,int,int);
```

```
double add(int,int,double);
int main() {
    int a=3,b=4,e;
    double c=3.4,d;
    e=add(a,b);
    cout<<e<<endl;
    e=add(a,b,b);
    cout<<e<<endl;
    d=add(a,b,c);
    cout<<d<<endl;
    return 0;
}
int add(int a1,int b1) {
    int sum=a1+b1;
    return(sum);
}
int add(int a1,int b1,int c1) {
    int sum=a1+b1+c1;
    return(sum);
}
double add(int a1,int b1,double c1) {
    double sum;
    sum=a1+b1+c1;
    return(sum);
}
```

👉 補充說明

C 語言就無函式多載性質。例如：傳回整數的絕對值函數「int abs()」與傳回浮點數的絕對值函數「double fabs()」就不同，請複習 11_1 節。但是 C++ 的 abs() 函式就無此問題了，因為 $\text{abs}(-3)=-3$ ， $\text{abs}(-3.4)=3.4$ 。

👉 自我練習

1. 於二維空間中，直線方程式是 $ax+by+c=0$ ，點 (x_0, y_0) 到直線的距離是 $d = \frac{|ax_0 + by_0 + c|}{\sqrt{a^2 + b^2}}$ ，三維空間中，平面方程式是 $ax+by+cz+d=0$ ，點 (x_0, y_0, z_0) 到直線的距離是 $d = \frac{|ax_0 + by_0 + cz_0 + d|}{\sqrt{a^2 + b^2 + c^2}}$ ，請寫一程式，使用函式多載的方式，當給 5 個參數時，即呼叫點與直線的距離，當給 7 個參數時，即呼叫點到平面的距離。

12-2 類別

真實世界是以物件描述物種，程式設計亦不應侷限在狹隘解決問題的函式，而是應以物件的宏觀角度撰寫程式。所以，基於物件導向的新觀念，程式開發工具即製訂一種新的型態，此型態稱為類別。每一類別都有屬於自己的屬性、函式。其次，下一節還要介紹繼承，讓程式設計者能繼承舊有類別開發新類別。物件導向的程式設計步驟分別是設計類別、樣例類別（類別的樣例就是物件）、使用物件。分別說明如下：

步驟一：設計類別

類別成員有屬性、函式。類別的簡易語法如下：

```
class 類別名稱 {
    private:
        屬性成員;
        函式成員;
    protected:
        屬性成員;
        函式成員;
    public :
        屬性成員;
        函式成員;
};
```

基於資料封裝等級，類別成員分為三級，分別是 `private`、`protected` 及 `public`。成員若是 `private`，則該成員僅能在類別內存取；`public` 則是公開的，類別與類別物件皆可存取。`protected` 則是折衷的，可供類別、類別物件與衍生類別與衍生類別物件存取。例如：以下一個簡易的類別實作，可判別成績是否及格，屬性成員有兩個，分別是 `score`、`result`，函式成員有一個 `Check()`，其封裝等級都是公開的，也就是類別內，與類別物件都可存取。

```
class Pass {
    public :
        int score;
        char result ='F';
        void Check() {
            result='F';
        }
};
```

```

        if (score >= 60)
            result = 'P';
    }
};

```

步驟二：樣例類別

類別的變數即稱為物件，物件亦稱為類別的實現或類別的樣例化 (Instance)。也就是說，類別就像是一個酒瓶或機器人的模型，在這些模型裏面已設計好它所具備的屬性與函式，當您需要酒瓶或機器人時，只要依照這些模型樣例一個或多個酒瓶或機器人即可。類別樣例的簡易語法如下：

```
類別名稱 物件名稱；
```

例如，以下程式 pa 樣例 Pass：

```
Pass pa;
```

與變數的宣告相同，其實就是類別變數，只是類別的變數稱為物件了。

► 步驟三：使用物件。

使用物件的簡易語法是

```
物件.成員；
```

但僅限於封裝等級是 `protected` 與 `public`，因為 `private` 僅限類別內存取。例如，以下程式可設定 `score` 成員。

```
pa.score=66;
```

以下程式可評判其及格與否。

```
pa.Check();
```

以下程式可取的評判結果。

```
char a=pa.result;
```

範例12-2a

示範以上程式。

 執行結果

```
88 is P
33 is F
```

 程式列印

```
#include <iostream>
using namespace std;
class Pass {
    public :
        int score;
        char result = 'F';
        void Check() {
            result='F';
            if (score>= 60)
                result='P';
        }
};
int main() {
    Pass pa;
    pa.score=88;
    pa.Check();
    cout <<pa.score <<" is "<<pa.result<<endl;
    pa.score=33;
    pa.Check();
    cout <<pa.score <<" is "<<pa.result<<endl;
    return 0;
}
```

 自我練習

1. 請設計一個類別，此類別有四個方法，分別是兩個整數相加、相減、相乘、相除。請問您至少要有哪些資料成員？
2. 請設計一個類別，此類別有一個方法，可處理自來水水費計算的計算。假設自來水費率如下：

- (1) 100 度以下，每度 3 元。
- (2) 超過 100 度的部分，每度 5 元。

請問您至少要有哪些資料成員？

※ 資料封裝

軟體科技的開發與其它的工業，如汽車、電視、收音機等機械、電子產品相較，可說是起步較晚的領域，在汽車、電視及收音機等產品上，我們發現這些產品很重視物件的封裝。例如，以電視機而言，我們發現內部有許多零件與開關，對於電視製造商而言，它們使用機殼將這些零件與開關 " 封裝 " 起來，才能避免使用者錯誤使用，只留下部份必要的按鈕（如開關鈕或選台鈕）與螢幕讓使用者欣賞節目。軟體程式的開發何嘗不應如此呢？所以對於類別的規劃，我們亦應重視所有函式、欄位及屬性封裝，使得這些成員有不同的封裝等級，以避免主程式與類別庫之間的干擾。就如同電視機或汽車的一些開關與旋鈕，有些是開放給一般的使用者調整，有的是讓維修工程師調整，有些則是出廠調校後，永遠不讓任何人調整。

基於資料的安全性，有些必須先檢查其是否安全與有效，再放入，此時就要透過函式的存取，先給予檢驗再放入。例如，上例的 `score`，我們必須先檢查其成績範圍是否正確，才能進行下一步的評判，所以我們先宣告其資料封裝等級為 `private`，並且使用 `setScore()` 函數輸入資料，且先檢查其範圍是否正確，若正確再放入 `score`，請看以下程式：

```
class Pass {
    private :
        int score;
        char result ='F';
    public:
        void setScore(int a){
            if (a>=0 && a<=100)
                score=a;
            else
                cout <<a <<"is error";
        }
        int getScore() {
```

```

        return(score);
    }
    char getResult(){
        return (result);
    }
    void Check()    {
        result='F';
        if (score>= 60)
            result='P';
    }
};

```

其次，既然 score 與 result 已經封裝為 private，那就無法用物件直接存取，也就是您無法使用 `pa.score=88` 設定成績，而必須使用 public 的 `getScore()` 與 `getResult()` 取得 private 的 score 與 result 資料。

範例12-2b

示範以上程式。

👉 執行結果

```

88 is P
33 is F
133 is error
33 is F

```

👉 程式列印

```

#include <iostream>
using namespace std;
class Pass {
    private :
        int score;
        char result ='F';
    public:
        void setScore(int a){
            if (a>=0 && a<=100)
                score=a;
            else

```

```

        cout <<a <<"is error"<<endl;
    }
    int getScore(){
        return(score);
    }
    char getResult(){
        return (result);
    }
    void Check() {
        result='F';
        if (score>= 60)
            result='P';
    }
};

int main() {
    Pass pa;
    pa.setScore(88);
    pa.Check();
    cout <<pa.getScore() <<" is " <<pa.getResult() <<endl;
    pa.setScore(33);
    pa.Check();
    cout <<pa.getScore() <<" is " <<pa.getResult() <<endl;
    pa.setScore(133);
    pa.Check();
    cout <<pa.getScore() <<" is " <<pa.getResult() <<endl;
    return 0;
}

```

自我練習

1. 請將上一範自我練習 1 的兩數相除改為私有成員，且當除數不為 0 時，才可相除。
2. 請將上一範例的實習的用水度數改為私有成員，當使用者輸入使用度數時，先檢查是否為正數或小於 200，若為正數且小於 200，才可放入類別的私有成員，並計算其水費。

※ 建構子 (Constructor)

建立變數的同時可以設定初值，例如，`int a=2;` 那建立物件的同時是否可以設定初值呢？答案是肯定的，因為類別有一函式成員非常特殊，此一函式成員名稱與類別名稱相同，此稱為建構子。C++ 對於建構子的規定如下：

1. 建構子是特殊的函式，函數名稱必須與類別名稱相同，但不能有傳回值。
2. 上一節提到函數可以有多載 (Overloading)，那建構子可以嗎？答案也是肯定的，因為建構子也是函式。例如，我們於上例 `Pass` 類別，新增兩個 `Pass` 函式如下：

```
class Pass {
    private :
        int score;
        char result ='F';
    public:
        Pass(int a){ //函式名稱與類別名稱相同
            if (a>=0 && a<=100)
                score=a;
            else
                cout <<a <<"is error";
        }
        Pass(){
            score=0;
        }
        void setScore(int a){
            if (a>=0 && a<=100)
                score=a;
            else
                cout <<a <<"is error"<<endl;
        }
        int getScore(){
            return(score);
        }
        char getResult(){
            return (result);
        }
}
```

```
    }  
    void Check() {  
        result='F';  
        if (score>= 60)  
            result='P';  
    }  
};
```

主程式撰寫如下：

```
int main() {  
    Pass pa1(33); //建立函式一併設定初值，且使用Pass(int)建構子  
    pa1.Check();  
    cout <<pa1.getScore() <<" is " <<pa1.getResult() <<endl;  
  
    Pass pa2; //建立函式未設定初值，則使用Pass()建構子，預設初值為0分  
    pa2.Check();  
    cout <<pa2.getScore() <<" is " <<pa2.getResult() <<endl;  
  
    Pass pa3; //建立函式未設定初值，則使用Pass()建構子，預設初值為0分  
    pa3.setScore(88);  
    pa3.Check();  
    cout <<pa3.getScore() <<" is " <<pa3.getResult() <<endl;  
  
    return 0;  
}
```

範例12-2c

示範以上程式。

👉 執行結果

```
33 is F  
0 is F  
88 is P
```

👉 自我練習

1. 請將上一範例的自我練習加上建構子。

※ 解構子 (Destructor)

C++ 爲了提高記憶體使用效率，還有解構子的設計，可以將建構子所佔用記憶體刪除。此解構子的名稱亦與類別名稱相同，但是在名稱前加上波浪符號（~）。例如；~Pass(){}，則可於物件使用完畢，先行歸還建構子所佔用記憶體。C++ 對於解構子的規定如下：

1. 解構子的名稱亦與類別名稱相同，但是在名稱前加上波浪符號（~）。
2. 解構子不可以有傳回值。
3. 解構子不得有參數。例如：上面 Pass 類別新增解構子如下：

```
~Pass() {} //解構子, 解構子不需有任何程式，系統會自動刪除已使用的記憶體
```

則於物件使用完畢，可執行此解構子如下：

```
pa2.~Pass();
```

則系統將會自動刪除已使用的記憶體。

範例12-2d

示範以上程式。

```
#include <iostream>
using namespace std;
class Pass {
    private :
        int score;
        char result ='F';
    public:
        Pass(int a){ //函式名稱與類別名稱相同
            if (a>=0 && a<=100)
                score=a;
            else
                cout <<a <<"is error";
        }
        Pass(){
            score=0;
        }
}
```

```
    ~Pass(){};
    void setScore(int a){
        if (a>=0 && a<=100)
            score=a;
        else
            cout <<a <<"is error";
    }
    int getScore(){
        return(score);
    }
    char getResult(){
        return (result);
    }
    void Check() {
        result='F';
        if (score>= 60)
            result='P';
    }
};

int main() {
    Pass pa1(33); //建立函式一併設定初值，且使用Pass(int)建構子
    pa1.Check();
    cout <<pa1.getScore() <<" is " <<pa1.getResult() <<endl;

    Pass pa2; //建立函式未設定初值，則使用Pass()建構子，預設初值為0分
    pa2.Check();
    cout <<pa2.getScore() <<" is " <<pa2.getResult() <<endl;

    Pass pa3; //建立函式未設定初值，則使用Pass()建構子，預設初值為0分
    pa3.setScore(88);
    pa3.Check();
    cout <<pa3.getScore() <<" is " <<pa3.getResult() <<endl;
    pa2.~Pass();
    return 0;
}
```

自我練習

1. 請將上一範例的自我練習加上建構子與解構子。

※ 靜態（static）成員

類別成員前加上 `static`，則稱此成員為靜態成員，靜態成員可不用樣例而直接使用。例如，於前面 `Pass` 類別新增資料靜態成員 `d`，與函式成員 `Checkgrade()`，此兩成員前均加上 `static` 如下，則此兩成員均是靜態成員。

```
class Pass {
public:
    static const int d=60;
    static char Checkgrade(int c){
        if (c>=d)
            return 'P';
        else
            return 'F';
    }
};
```

其次，靜態成員於時前，不用事先樣例，直接使用範圍運算子（`::`）就可直接使用。例如：

```
int main() {
    char a;
    int b;
    b=Pass::d;    //直接使用類別資料成員d
    cout <<b<<endl;

    a=Pass::Checkgrade(88); //直接使用類別函式成員
    cout <<a<<endl;

    a=Pass::Checkgrade(22);    //直接使用類別函式成員
    cout <<a<<endl;
    return 0;
}
```

範例12-2e

示範以上程式。

執行結果

```
60
P
F
```

程式列印

```
#include <iostream>
using namespace std;
class Pass {
    public:
        static const int d=60;
        static char Checkgrade(int c){
            if (c>=d)
                return 'P';
            else
                return 'F';
        }
};
int main() {
    char a;
    int b;
    b=Pass::d;    //直接使用類別資料成員d
    cout <<b<<endl;

    a=Pass::Checkgrade(88); //直接使用類別函式成員
    cout <<a<<endl;

    a=Pass::Checkgrade(22);    //直接使用類別函式成員
    cout <<a<<endl;
    return 0;
}
```

自我練習

1. 請設計一個計算面積的類別，此類別有常數 π ，及正方形、長方形、圓面積計算函式，且均可不用樣例此類別，即可使用該常數或該函式。

※ 12-3 繼承

繼承(Inherit)

任何新產品的開發，均不是無中生有，而是從舊有的產品中繼承某些特性，再加入新的零件或修改部份零件而成一項新的產品。例如，SENTRA180 正是繼承 NEW SENTRA 而來，只是排氣量提高了、內裝豪華了，但是輪胎、方向盤及座椅還是用原來 SENTRA 的東西，這就是繼承的道理，使得新產品的開發得以縮短時程。軟體的開發何嘗不應如此？繼承的另一優點是同一函式得以讓數個新舊版本同時存在。因為當你有新產品時，你不可能同時讓你的新舊客戶同時更新，所以您必須讓這些不同的版本同時存在，以滿足不同的產品需求，就像是您修車更換零件時，師傅總會強調這是哪一版本的汽車，以便決定訂購哪一版本的零件。C++ 的類別繼承簡要語法如下：

```
class 衍生類別 : 繼承方式 基礎類別{;
```

繼承方式有 public、protected 及 private，此三種繼承對基礎類別的影響如下表：

基礎類別成員封裝方式	繼承方法	衍生類別成員封裝方式
private	public	無法存取
protected	public	protected
public	public	public
private	protected	無法存取
protected	protected	protected
public	protected	protected
private	private	無法存取
protected	private	private
public	private	private

例如，以下程式，衍生類別 Pass2 繼承基礎類別 Pass，繼承的方式為 public，原來基礎類別的資料封裝方式 private 改為無法存取，protected 及 public 不變。

```
class Pass2:public Pass{  
};
```

所以主程式樣例 Pass2 物件之後，即可使用基礎類別的所有成員。例如：

```
Pass2 pa;  
pa.setScore(66);  
pa.Check();  
cout <<pa.getScore() <<" is "<<pa.getResult()<<endl;
```

範例12-3a

示範以上類別繼承。

👉 執行結果

```
66 is P  
33 is F
```

👉 程式列印

```
#include <iostream>  
using namespace std;  
class Pass {  
    protected :  
        int score;  
        char result ='F';  
    public:  
        Pass(int a){ //函式名稱與類別名稱相同  
            if (a>=0 && a<=100)  
                score=a;  
            else  
                cout <<a <<"is error";  
        }  
        Pass() {
```

```

        score=0;
    }
    void setScore(int a){
        if (a>=0 && a<=100)
            score=a;
        else
            cout <<a <<"is error";
    }
    int getScore(){
        return(score);
    }
    char getResult(){
        return (result);
    }
    void Check() {
        result='F';
        if (score>= 60)
            result='P';
    }
};

class Pass2:public Pass{
};

int main() {
    Pass2 pa;
    pa.setScore(66);
    pa.Check();
    cout <<pa.getScore() <<" is " <<pa.getResult() <<endl;
    pa.setScore(33);
    pa.Check();
    cout <<pa.getScore() <<" is " <<pa.getResult() <<endl;
    return 0;
}

```

自我練習

1. 同範例 12-2a 自我練習 1，請設計一個類別繼承此類別，且新增一個計算兩數相除的餘數的方法。

2. 請設計一個類別，此類別繼承上一單元的水費計算，但新增計算電費的方法。電費假設未超過 50 度，基本費 100 元，超過時每度 3 元。

※ 多型

多型 (Polymorphism) 有些書譯成同名異式，它的原文是希臘文，意思是說，一個名稱有多種表現的方式，目的是滿足資料的抽象化 (Data Abstraction)。例如，你有一個僕人專門幫你開門，那麼不論這門是內推，外拉或向旁邊推，你都是下同一指令「開門」，然後你的僕人即會依照門的結構而完成開門的動作。以上的重點是，使用者不用去描述細節，編譯器會依照實際的狀態，解決問題。物件導向的程式設計亦發揚此理念，讓程式設計師於程式設計階段使用相同的指令，而編譯器能於執行階段依據不同的需求，執行不同的程式片段，此即為多型。C++ 物件導向的多型表現，常用的分別是多載 (Overloading) 與改寫 (Overriding)，函式的多載已經於 12-1 節介紹，函式改寫說明如下：

► 函式改寫

函式改寫 (Function overriding) 有些書翻譯成函式覆載，也就是說衍生類別可繼續使用原基礎類別的函式名稱，但可以修改或賦予新的功能，因為名稱相同，這樣才不會影響使用者的程式撰寫習慣。例如，原來 Pass 類別的及格標準是 60 分，現在及格標準是 70 分了，所以衍生類別可以改寫 Check 方法如下：

```
class Pass2:public Pass{
    public :
        void Check()    {
            result='F';
            if (score>= 70)
                result='P';
        }
};
```

其次，新舊客戶都不影響其程式撰寫習慣，新客戶就樣例衍生類別 Pass2，其及格標準是 70 分；舊客戶繼續樣例 Pass，其及格標準是 60。請看以下範例。

範例12-3b

示範以上函式改寫。

 執行結果

```
66 is F
72 is P
66 is P
```

 程式列印

```
#include <iostream>
using namespace std;
class Pass {
    //同上一範例12-3a
};

class Pass2:public Pass{
public :
    void Check() {
        result='F';
        if (score>= 70)
            result='P';
    }
};

int main() {
    Pass2 pa;
    pa.setScore(66);
    pa.Check();
    cout <<pa.getScore() <<" is "<<pa.getResult()<<endl;
    pa.setScore(72);
    pa.Check();
    cout <<pa.getScore() <<" is "<<pa.getResult()<<endl;

    Pass pb;
    pb.setScore(66);
    pb.Check();
    cout <<pb.getScore() <<" is "<<pb.getResult()<<endl;
    return 0;
}
```

👉 自我練習

1. 同上一範例自我練習 1，但加法改寫為 $c=a+2b$ 。
2. 同上一範例，但是水費計價方式請調為
 - (1) 100 度以下，每度 4 元。
 - (2) 超過 100 度的部分，每度 6 元。

※👤 建構子與繼承

基礎類別的所有資料與函式成員皆可被繼承，但是建構子與解構子並沒有被繼承，所以衍生類別的建構子必須自行撰寫。以下範例示範撰寫衍生類別 Pass2 的建構子。

範例12-3c

示範撰寫衍生類別 Pass2 的建構子。

👉 執行結果

```
66 is F
72 is P
66 is P
```

👉 程式列印

```
#include <iostream>
using namespace std;
class Pass {
    protected :
        int score;
        char result = 'F';
    public:
        Pass(int a){ //函式名稱與類別名稱相同
            if (a>=0 && a<=100)
                score=a;
            else
                cout <<a <<"is error";
        }
        Pass(){
            score=0;
        }
};
```

```
    }

    void setScore(int a){
        if (a>=0 && a<=100)
            score=a;
        else
            cout <<a <<"is error";
    }
    int getScore(){
        return(score);
    }
    char getResult(){
        return (result);
    }
    void Check() {
        result='F';
        if (score>= 60)
            result='P';
    }
};

class Pass2:public Pass{
public :
    Pass2(int a){ //函式名稱與類別名稱相同
        if (a>=0 && a<=100)
            score=a;
        else
            cout <<a <<"is error";
    }
    Pass2(){
        score=0;
    }
    void Check() {
        result='F';
        if (score>= 70)
            result='P';
    }
};

int main() {
```

```

    Pass2 pa(66);
    pa.Check();
    cout <<pa.getScore() <<" is "<<pa.getResult()<<endl;

    Pass2 pb(72);
    pb.Check();
    cout <<pb.getScore() <<" is "<<pb.getResult()<<endl;

    Pass pc(66);
    pc.Check();
    cout <<pc.getScore() <<" is "<<pc.getResult()<<endl;
    return 0;
}

```

自我練習

1. 同上一範例自我練習，但請增加建構子。

※ 多重繼承 (Multiple inheritance)

如果爸爸有錢，媽媽很漂亮，通常生出既漂亮又有錢的兒女，那程式設計是否可以如此多重繼承呢？答案是亦是肯定的。多重繼承的語法如下：

```
class 衍生類別:繼承方式,基礎類別1,[ 繼承方式,基礎類別2,][...]
```

繼承方式同單一繼承的 private、protected、public，繼承效力也同單一繼承。例如，類別 Fish 如下，有游泳能力。

```

class Fish{
    public:
        void swimming(){
            cout<<"I can swimming"<<endl;
        }
};

```

類別 Bird 如下，有飛翔能力。

```

class Bird{
    public:

```

```

        void flying() {
            cout<<"I can flying"<<endl;
        }
    };

```

使用多重繼承如下：

```

class Super:public Fish,public Bird{};

```

衍生類別 Super 即同時擁有游泳與飛翔能力，例如，主程式不用吹灰之力，不用撰寫任何程式，即可同時使用 Fish 類別的 swimming() 函式與 Bird 類別的 Flying() 函式，請看以下範例。

```

Super su;
su.swimming();
su.flying() ;

```

範例12-3d

示範多重繼承。

執行結果

```

I can swimming
I can flying

```

程式列印

```

#include <iostream>
using namespace std;
class Fish{
    public:
        void swimming(){
            cout<<"I can swimming"<<endl;
        }
};
class Bird{
    public:
        void flying(){
            cout<<"I can flying"<<endl;
        }
};

```

```
};
class Super:public Fish,public Bird{
};
int main() {
    Super su;
    su.swimming();
    su.flying() ;
    return 0;
}
```

自我練習

1. 請分別寫一個可以解二元一次方程式與解一元二次方程式的類別（本例均假設方程式有解），再寫一類別，同時繼承以上兩個類別，並新增一個可以解三元一次方程式的函式。
2. 關於物件導向程式語言的敘述，下列何者錯誤？（111 年統測）
 - (A) 物件導向程式語言具有「多型」、「繼承」與「封裝」的特性
 - (B) 物件導向程式語言的「多型」可以達到資訊隱藏的目的（資料的抽象化）
 - (C) 「繼承」的特性讓物件導向程式語言具有避免重複撰寫相同程式碼的優點
 - (D) 物件導向程式語言可以簡化大型程式的開發流程

12-4 應用實例

複數運算

複數的標準式為 $a+bi$ ，若有兩以標準式表示的複數 $P=a+bi$ 與 $Q=c+di$ ，則兩複數的加減乘除運算如下：

- (1) $P+Q=(a+c)+(b+d)i$
- (2) $P-Q=(a-c)+(b-d)i$
- (3) $P*Q=(ac-bd)+(ad+bc)i$
- (4) $P/Q=(ac+bd)+(bc-ad)i/(a^2+d^2)$

範例12-4a

請開發一個類別，可以執行以上運算，並輸出結果。

執行結果

```
1+2i  
2+3i  
3+5i
```

程式列印

```
#include <iostream>
using namespace std;
class Complex{
    private:
        int real,img;
    public:
        Complex(){}
        Complex(int m,int n){
            real=m;
            img=n;
        }
        void Add( Complex a, Complex b){
            real=a.real+b.real;
            img=a.img+b.img;
        }
        void Data(){
            cout <<real<<"+"<<img<<"i"<<endl;
        }
};

int main() {
    Complex a(1,2);
    a.Data();
    Complex b(2,3);
    b.Data();
    Complex c;
    c.Add(a,b);
    c.Data();
    return 0;
}
```

 自行練習

1. 請自行完成以上範例的減法、乘除法。

※ 運算子改寫

以上的加法運算是採用函式完成，例如

```
c.Add(a,b);
```

那是否可如同我們平常的撰寫習慣，寫成

```
c=a+b;
```

這樣的話，就是讓使用者不用進行任何調整，即可使用舊有書寫習慣。現在要讓程式語言有智慧，去判斷此加法(+)是實數還是複數的加法運算，實數就採用實數的加法規則，複數就採用複數的加法規則。反正使用者只要下達加法指令就好了，電腦就會依照當時的情況執行相加，此即為物件導向的多型表現，就如同本章開頭說的，您只要下達開門就好，電腦就會依照門的構造，執行外推，內推或外拉等開門動作。運算子改寫簡要語法如下：

```
friend 傳回型態 operator運算子(參數列) {
    return 傳回資料;
}
```

以上的傳回型態、參數列、傳回資料均與函式相同。但是類別的私有資料成員僅有類別的函式成員可存取，所以要將此運算子成員宣告為friend(意思為伙伴或朋友)。請看以下範例說明。

範例12-4b

示範運算子改寫。

 執行結果

```
1+2i
2+3i
3+5i
11
```


程式列印

1. 改寫加法運算子如下：

```
friend Complex operator+( Complex a, Complex b){
    Complex c;
    c.real=a.real+b.real;
    c.img=a.img+b.img;
    return c;
}
```

則主程式即可使用：

```
Complex a(1,2);
Complex b(2,3);
Complex c;
c=a+b;
c.Data(); // 3+5i
```

其次，實數的加法亦使用相同運算子如下，電腦會自動依照其型態執行正確的運算。

```
int d=3,e=8;
int f=d+e;
cout<<f;    //11
```

2. 全部程式列印如下：

```
#include <iostream>
using namespace std;
//class Complex;
class Complex{
private:
    int real,img;
public:
    Complex(){}
    Complex(int m,int n){
        real=m;
        img=n;
    }
    void Add( Complex a, Complex b){
        real=a.real+b.real;
```

```

        img=a.img+b.img;
    }
    friend Complex operator+( Complex a, Complex b){
        Complex c;
        c.real=a.real+b.real;
        c.img=a.img+b.img;
        return c;
    }
    void Data(){
        cout <<real<<"+"<<img<<"i"<<endl;
    }
};

int main() {
    Complex a(1,2);
    a.Data();
    Complex b(2,3);
    b.Data();
    Complex c;
    c=a+b;//3+5i
    c.Data();
    int d=3,e=8;
    int f=d+e;//11
    cout<<f<<endl;
    return 0;
}

```

自行練習

1. 請補足以上的減法，乘除法運算。

課後習題

1. 請設計一個類別，可以計算兩個以極式 ($5 < 30$) 表示的複數的加、減、乘、除等運算。

假設 $P=r_1 \angle \theta_1$ $Q=r_2 \angle \theta_2$

(1) $P+Q$ 與 $P-Q$ 請先將極式轉為標準式，再相加減，最後再轉為極式。

(2) $P*Q=(r_1*r_2) \angle (\theta_1+\theta_2)$

(3) $P/Q=(r_1/r_2) \angle (\theta_1-\theta_2)$

※2. 請以改寫運算子的方式，重作上一題。

附錄1：關鍵字索引

茲將本書一些 C 語言關鍵字第 1 次出現的頁數整理如下，當讀者遇到關鍵字時，可於此找到對應的說明。

編號	關鍵字	出處
1	C 專案架構	2-4
2	C 程式架構 <pre>#include <stdio.h> #include <stdlib.h> int main(){ printf("Hello world!\n"); return 0; }</pre>	2-5
3	C 語言架構	2-7
4	演算法	2-8
5	流程圖符號	2-9
6	Code::Blocks 整合開發環境	2-11
7	資料種類：字元、整數、浮點數、字元、字串	3-2
8	資料型態：short,int,long,unsigned,float,double,char	3-4
9	跳脫字元序列	3-7
10	識別字	4-2
11	保留字	4-3
12	變數的宣告： <pre>int a;//整數 long b;//長整數 double c;//浮點數 char d;//字元 char e[10];字串</pre>	4-4

編號	關鍵字	出處
13	常數的宣告 : <code>const</code> , <code>define</code>	4-6
14	資料型態的轉換 (隱含 / 強制轉換)	4-8
15	<code>printf()</code>	5-2
16	網路 C 語言手冊	5-5
17	<code>scanf()</code>	5-8
18	亂數的取得 : <code>rand()</code> , <code>srand()</code>	5-12
19	聲音的產生 : <code>_beep()</code>	5-15
20	算術運算子 : <code>=</code> , <code>++</code> , <code>--</code> , <code>+</code> , <code>-</code> , <code>*</code> , <code>/</code> , <code>%</code>	6-2
21	複合指派運算子 : <code>+=</code> , <code>-=</code> , <code>*=</code> , <code>/=</code> , <code>%=</code>	6-7
22	比較運算子 : <code><</code> , <code>></code> , <code><=</code> , <code>>=</code> , <code>==</code> , <code>!=</code>	6-8
23	邏輯運算子 : <code>!</code> , <code>&&</code> , <code> </code>	6-9
24	位元運算子 : <code>~</code> , <code><<</code> , <code>>></code> , <code>&</code> , <code>^</code> , <code> </code>	6-12
25	運算元優先順序 :	6-18
26	運算子結合律 :	6-20
27	運算式 : <code>x>3</code>	6-22
28	敘述 : <code>x=y+3</code>	6-23
29	註解 : <code>//</code> , <code>/* */</code>	6-23
30	專案除錯	6-33
31	<code>if</code> ... <code>else</code>	7-2
32	<code>switch</code> ... <code>case</code>	7-11
33	<code>for</code>	8-2
34	<code>while</code>	8-10
35	一維陣列 : <code>int a[6];</code>	9-2
36	二維陣列 : <code>int a[6][3];</code>	9-11
37	字元陣列 : <code>char a[]="ABC";</code>	9-19
38	結構 : <code>struct aa {};</code>	9-25
39	氣泡排序法	9-34
40	計數排序法	9-39
41	循序搜尋法	9-43
42	二分搜尋法	9-47

編號	關鍵字	出處
43	指標： <code>*ptr</code>	10-2
44	數值函式： <code><math.h></code> , <code>abs()</code> , <code>fabs()</code> , <code>pow()</code> , <code>sqrt()</code>	11-3
45	字元處理函式： <code><stdlib.h></code> , <code>getchar()</code> , <code>putchar()</code>	11-12
46	字串函式： <code><string.h></code> , <code>strcpy()</code> , <code>strcmp()</code> , <code>strlen()</code>	11-19
47	字串與數值互轉： <code><stdlib.h></code> , <code>atoi()</code> , <code>atof()</code> , <code>itoa()</code>	11-26
48	日期時間函式： <code><time.h></code> , <code>_strdate()</code> , <code>strtime()</code> , <code>clock()</code> , <code>time()</code> , <code>time_t</code>	11-27
49	自訂函式： <code>void</code>	11-31
50	傳值與傳址	11-37
51	變數與常數的有效範圍	11-42
52	<code><iostream></code>	12-5
53	<code>using namespace std;</code>	12-5
54	<code>cout</code>	12-6
55	<code>cin</code>	12-7
56	<code>endl</code>	12-6
57	<code>string</code>	12-9
58	函式多載	12-15
59	類別： <code>class</code> , <code>private</code> , <code>protected</code> , <code>public</code>	12-17
60	資料封裝：	12-20
61	建構子 / 解構子： <code>~</code>	12-23~25
62	靜態成員： <code>static</code>	12-27
63	類別繼承：	12-29
64	函式改寫：	12-32
65	多重繼承：	12-36
66	運算子改寫：	12-40

國家圖書館出版品預行編目(CIP)資料

程式設計實習 / 洪國勝, 孫銘宏, 蔡懷文, 鍾享旭, 蔡懷介, 陳蘊慈編著. -- 初版. -- 高雄市: 泉勝出版有限公司, 民 112.04

面; 公分

技術型高級中等學校電機與電子群

ISBN 978-986-99632-7-5(平裝)

1.CST: 資訊教育 2.CST: 電腦程式設計 3.CST: 技職教育
528.831 112003361

十二年國民基本教育

技術型高級中等學校電機與電子群

程式設計實習(全一冊)

審定字號: 技審字第 112007 號

書號: SE001

編著者: 洪國勝、孫銘宏、蔡懷文、鍾享旭、蔡懷介、陳蘊慈

責任編輯: 洪國勝

排版封面設計: 陳彥慈(芒現)

發行所: 泉勝出版有限公司

地址: 高雄市左營區華夏路 708 號 17 樓

電話: 0900757159 LINE ID: 5abook

網址: www.goodbooks.com.tw

Email: aa163677@yahoo.com.tw

訂購方式: 電話: 0900757159 LINE ID: 5abook Email: aa163677@yahoo.com.tw

初版日期: 112 年 4 月

版次: 初版第一刷

定價: 500

ISBN: 978-986-99632-7-5

◎書內程式、圖片、資料的來源已盡查明之責且標注出處,若有疏漏,在此致歉,並請通知我們,我們將於再版時修訂。

◎若發現書有缺頁、倒裝、嚴重污損,請直接盡快與本公司聯繫,我們會盡快更換。

◎版權所有,請勿翻印。

