Contents

01 C 語言簡介 3

1-1 程式語言簡介 3

1-2 Dev C++ 的操作 4

02 基本資料型態與運算 17

2-1 變數 17

2-2 基本運算式 21

03 選擇結構 27

3-1 單向與雙向選擇結構 27

3-2 巢狀選擇結構 31

3-3 Switch 結構 33

04 重複結構 37

4-1 迴圈結構 37

4-2 前測式重複結構 39

4-3 後測式重複結構 40

4-4 巢狀重複結構 43

Contents

05 陣列

5-1 一維陣列 46
5-2 實例練習—計算總分與平均數 48
5-3 二維陣列 49

06 問題解決實例練習

6-1 求陣列所有元素之最大值 55
6-2 求 N 階乘 56
6-3 求質數問題 57

07 自訂函式

7-1 求加總兩數 60
7-2 求質數 62

Chapter 01

C 語言簡介

1-1 程式語言簡介

程式語言概說

電腦的資料儲存與指令運算都是使用二進位表示，所謂二進位是指資料與指令皆由 0 與 1 所組成。最早的程式語言是由 0 與 1 所組成，程式設計師必須查詢指令的二進位代碼，將所需的資料轉成 0 與 1，這樣的語言稱作**機器語言** (Machine Language)，機器語言對不同處理器的電腦，就需要改寫程式，欠缺彈性且不易撰寫複雜的程式。

因機器語言的撰寫困難，後來演進成**組合語言** (Assembly Language)，組合語言提供文字的指令與變數，透過**組譯器**將組合語言轉成機器語言，組合語言提供文字的指令與變數讓撰寫程式更容易，但組合語言的指令與處理器相關，不同處理器就要重寫程式。

為了能讓程式對不同處理器都不需改寫，產生了**高階語言** (High-Level Programming Language)，透過**編譯器**將高階語言轉成機器語言，不同的處理器只要透過編譯器的支援，將程式轉成不同的機器語言，如此可以達成程式設計與機器無關，C 語言就屬於高階語言。

C 語言的起源

在 1970 年後，Dennis Ritchie 和 Ken Thompson 創造了 C 語言，以 B 語言為基礎改進而來，因此命名為「C」語言。在 1973 年，Dennis Ritchie 和 Ken Thompson 利用 C 語言創造了 Unix 作業系統，在此之前都是利用組合語言撰寫作業系統。在 1980 年代，為了解決各個版本 C 語言的些許

不同造成程式撰寫的困擾，美國國家標準局 (American National Standard Institution) 為 C 語言制定了標準語法，符合這標準的 C 語言稱作 ANSI C，到目前為止的 C 語言編輯器一般都支援 ANSI C 的標準。

C 語言的特性

C 語言廣泛用於系統與應用軟體的開發，具有高階語言的特性，將接近口語語法所撰寫的高階程式，經過編譯器轉成可執行檔，於作業系統中執行執行檔，獲得我們想要的結果。有了 C 語言編譯器的發明，程式設計者可以利用高階語言容易撰寫的特性，輕鬆撰寫程式。

幾乎所有作業系統皆有支援 C 語言的編輯器，因此相同程式碼就可以到不同作業系統重新編譯就可以執行，C 語言就擁有跨平台的能力。

高階 C 語言可以結合低階的組合語言與作業系統，做出驅動程式作為電腦與周邊設備連結的軟體。Unix-Like 作業系統大部分是由 C 語言所寫出，在這類作業系統甚至可以直接呼叫系統所提供的 C 語言函式，寫出的程式直接影響作業系統的運作。

1-2 Dev C++ 的操作

下載與安裝

Dev C++ 是一套免費的 C++ 整合開發環境 (Integrated Development Environment，簡寫為 IDE) 與函式庫，整合開發環境將編輯、編譯、執行與除錯等功能結合在一起，該軟體為 Sourceforge 的開發專案，Sourceforge 為線上開放程式碼專案管理網站，提供有志之士撰寫與開發開放程式碼 (Open Source) 程式的整合與溝通平台，目前有許多實用的程式皆來自此網站。

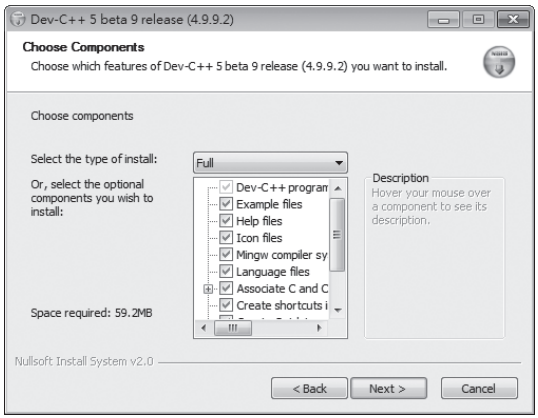
軟體下載

Dev C++ 下載網址為：http://sourceforge.net/projects/dev-cpp，以下就來介紹 Dev C++ 的下載及安裝方式。

Step 1　點選網頁旁的「View All Files」按鈕，尋找「All Files → Binaries → Dev-C++ 4.9.9.2 → devcpp-4.9.9.2_setup.exe」檔案並下載，檔案大小約 9.3MB，下載後執行安裝。

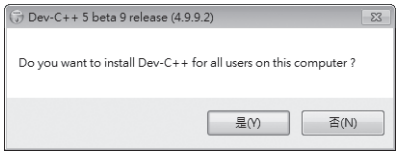
Step 2　選擇安裝過程的語言，在安裝過程中沒有繁體中文的選項，所以選擇「English」，再點選「OK」，接著會出現版權宣告，點選「I Agree」同意版權宣告後，才可以安裝。

Step 3　出現選擇安裝元件視窗，在這邊我們先依預設值設定，可以不用修改，點選「Next」繼續。



Step 4　選擇安裝資料夾，我們可以直接使用預設值的安裝位置，或是點選「Browse」選擇安裝資料夾，再點選「Install」。安裝時會解壓縮與安裝檔案進入系統。

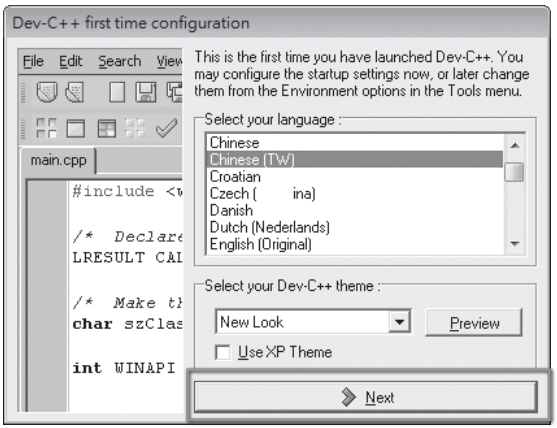
安裝過程中，會詢問是否要為電腦上的所有使用者安裝，按下「是」即可。



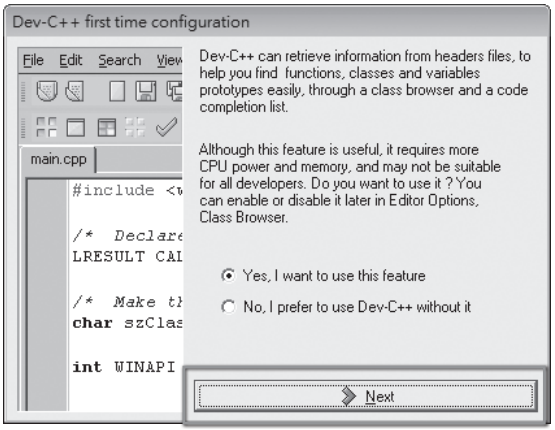
Step 5　安裝完成後，點選「Finish」，即可完成安裝 Dev C++。

第一次執行時的設定

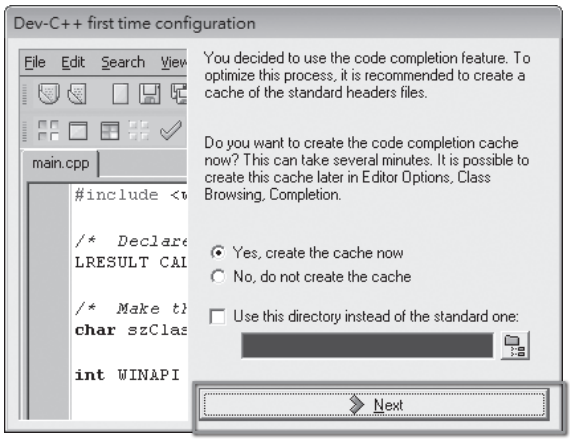
第一次開啓 Dev C++ 時，會出現選擇操作介面語言的視窗，在此選擇「Chinese(TW)」，表示使用繁體中文，再點選「Next」。



接著會詢問是否允許使用類別瀏覽器 (Class Browser)，點選「Yes , I want to use this feature」後 ，點選「Next」。



為了增加執行的效能，製作表頭檔 (Header) 的快取 (Cache)，點選「Yes , create the cache now」後 ，點選「Next」。



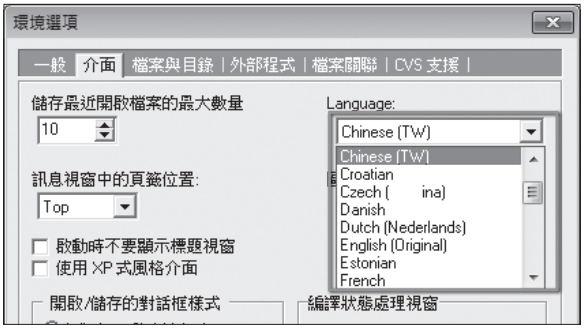
進行快取的製作需要一段時間，快取製作完成後，點選「 OK 」，Dev C++ 的安裝就完成了。

介面語系修改

視需要，我們可以將介面的語系再次修改成其它語系，點選「 工具→環境選項 」即可設定。



點選「介面」頁籤，修改 Language，選擇自己想要更改的語言後，點選「確定」即可更改介面的語言。

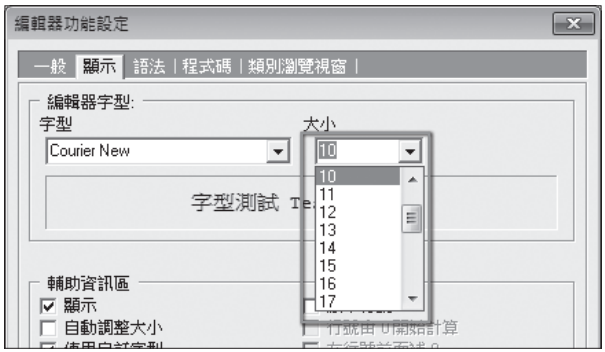


設定程式碼字體大小

一般而言預設字體大小為 10，如果覺得太小，我們可以更改編輯器字體。點選「工具→編輯器選項」。



點選「顯示」頁籤，選擇自己想要的大小後，點選「確定」即可。

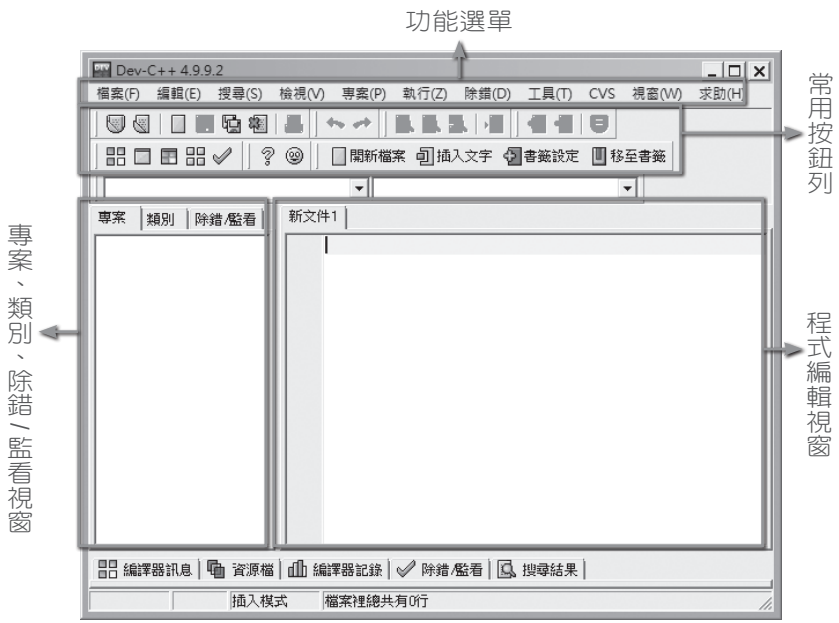


工作環境

完成了基礎的 Dev C++ 環境設定後，再來看 Dev C++ 的操作介面介紹。新增 C++ 檔案，點選「新增→開新檔案→原始碼」。



就會出現新增 C++ 檔案的畫面，以下簡單介紹介面：



- **功能選單**：Dev-C++ 所有功能皆置於此區塊，分類為檔案、編輯 ... 等選單，點選後出現下拉選單，將相關功能操作放在一起。
- **常用按鈕列**：將常用功能轉換成按鈕，點選按鈕執行對應的功能，如開新檔案、儲存檔案、編譯、執行、找尋與取代。

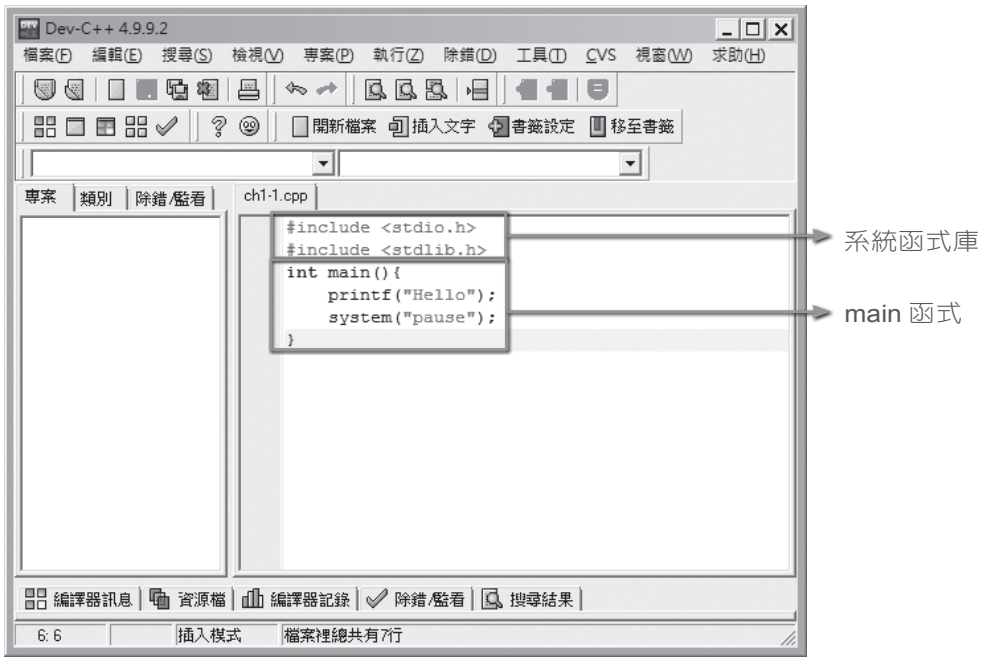
- **專案、類別與除錯 / 監看視窗**：專案視窗將同一專案內檔案資訊顯示於此；類別視窗顯示程式中的類別；除錯與監看視窗當除錯時顯示相關變數的值。
- **程式編輯視窗**：此視窗用於編寫程式碼。

編譯與執行

第一個程式

於「程式編輯視窗」輸入以下程式碼，這是一個最簡單的程式碼，在螢幕印出「Hello」，執行中斷並等待按任意鍵繼續，若程式不中斷，程式執行完後就結束，無法得知執行的結果，所以需要中斷程式。

C 語言在撰寫時有一定的格式，例如以下的程式碼：



系統函式庫

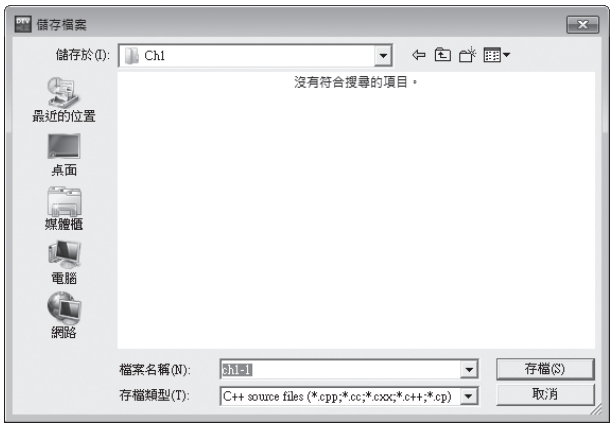
C 語言一定會包含系統函式庫，這樣程式就不用從頭寫到尾，利用系統現有的函式庫直接引用，如在螢幕顯示「Hello」使用 printf 函式，該函式由 `stdio.h` 提供，所以第 1 行就將 `stdio.h` 包含進來，程式執行要中斷使用 `system` 函式，該函式由 `stdlib.h` 提供。

main 函式

C 語言定義程式第一個開始執行的函式為 `main` 函式，所以需先寫一個 `main` 函式，函式名稱 `main` 後接小括號，表示 `main` 為函式，函式需要指定開始與結束的位置，C 語言定義使用大括號刮起來，左大括號 `{` 表示函式開始，右大括號 `}` 表示函式結束。

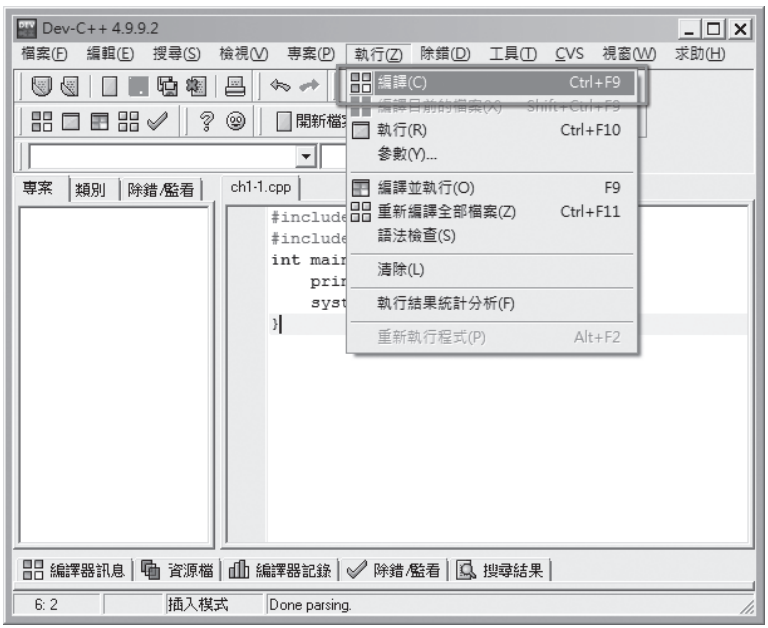
本程式範例 `main` 函式只有兩行，第 4 行「`printf("Hello")`」印出「Hello」，第 5 行 `system("pause")` 中斷程式執行，程式為循序執行，循序執行為第 1 行執行完後換第 2 行執行，如此一直執行到最後一行為止。

儲存程式，瀏覽儲存的目錄，輸入檔名如「ch1-1.cpp」後，點選「儲存」。

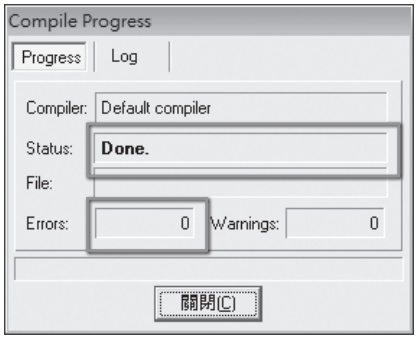


程式編譯

點選「執行→編譯」。



Status 出現「Done」表示，Errors 為 0，表示編譯已完成且沒有錯誤，點選「關閉」以關掉這個視窗。

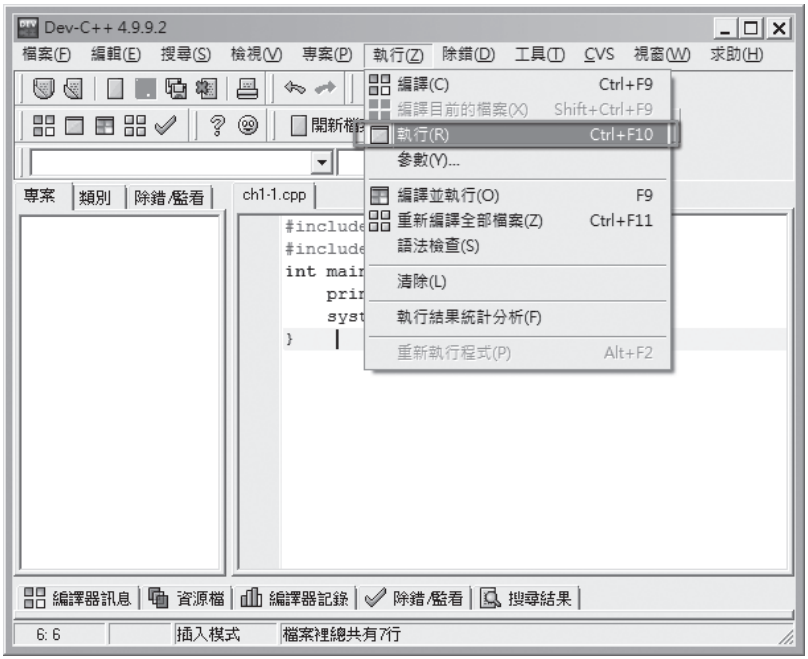


編譯完成後，原始碼資料夾會多出一個執行檔 ch1-1.exe，表示 ch1-1.cpp 已經編譯成功了。



程式執行

點選「 執行→執行 」，就可以執行剛剛這段程式碼。

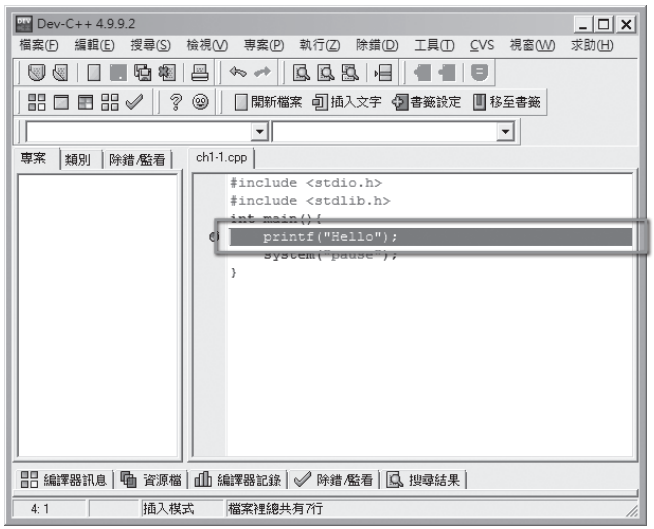


執行結果如下，螢幕出現「Hello」後，等待按下任意鍵繼續。



偵錯與除錯

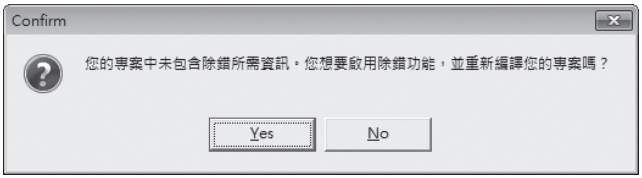
點選程式碼編輯視窗左側空白處，游標所在的整行程式碼都會變成紅色，此程式執行時到此行之前就會中斷，等待使用者輸入的除錯命令，如下一步、追蹤入函式 ... 等。



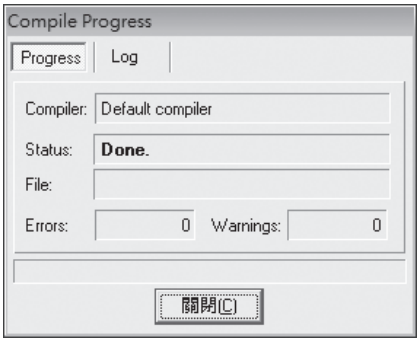
進行除錯，會先重新編譯程式，讓程式有除錯的功能，點選「除錯→除錯」。



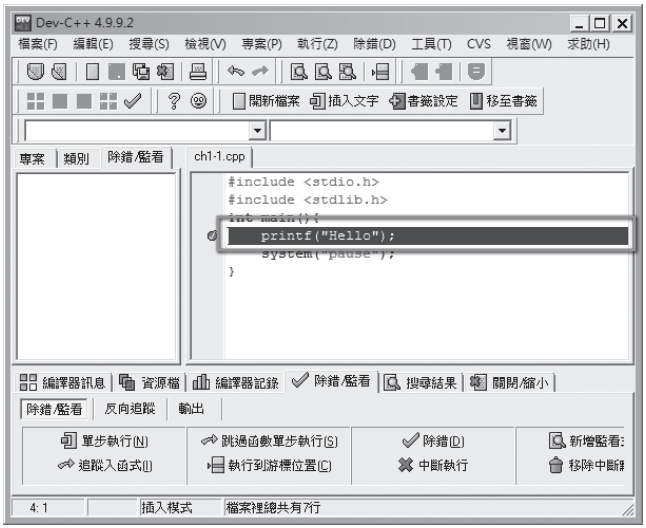
會出現需重新編譯的訊息，點選「Yes」。



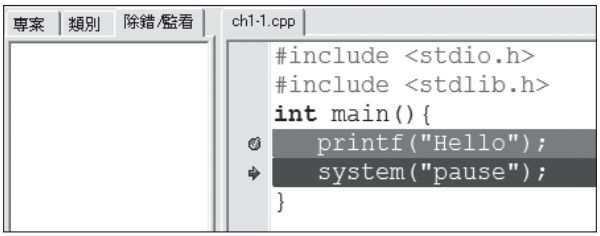
編譯後出現此訊息視窗，Status 出現「Done」編譯成功。點選「關閉」。



除錯後會出現右圖畫面，該行由紅色變成藍色表示程式執行到此中斷，等待使用者除錯，點選「單步執行」。



螢幕輸出「Hello」後，跳到下一行。



隨堂練習

( ) 1. C 語言由哪一個語言演進而來？

(A) A 語言 (B) B 語言 (C) D 語言 (D) E 語言。

( ) 2. 以下何種語言可以讓相同程式不需修改可以在不同處理器執行？

(A) 組合語言 (B) 機器語言 (C) 高階語言 (D) 自然語言。

( ) 3. C 語言透過什麼軟體轉成可執行程式？

(A) 直譯器 (B) 編譯器 (C) 組譯器 (D) 翻譯器。

( ) 4. Dev-C++ 所提供的功能不包含下列何者？

(A) 編輯 (B) 編譯 (C) 執行 (D) 自動修正錯誤

Chapter 02

基本資料型態與運算

2-1 變數

變數是程式放資料的空間，佔有對應的記憶體空間，程式在運算過程中，將資料進行處理與運算，就是對變數與對應的記憶體進行處理與運算。

國中數學中學到的一元二次方程式，如 x+y=12，x 與 y 是未知數，跟程式的變數有相同的概念，都是代表某個資料。程式中命名變數的方式，也與數學相同，通常以英文字母命名，如 a，b，xyz 等，再對變數進行運算，例如 xyz=a+b，就是將 a 加 b 的結果儲存到 xyz。

補充說明 變數與記憶體

電腦運算的暫存空間爲記憶體，程式的執行爲電腦將資料與程式由硬碟中讀取到記憶體，電腦執行記憶體中的程式並處理資料，過程中所需暫存資料的空間也在記憶體，記憶體存取的方式，是以位址的方式存取資料，記憶體位址的編列，由最開頭處第 1 個 Byte 位址爲 0，第二個 Byte 位址爲 1，依此類推。程式中使用變數命名某個記憶體，亦即某個變數代表某個記憶體，程式中變數初始化相當於將值存入所對應的記憶體位置，使用者只需瞭解變數的使用，記憶體與變數的對應由 C 語言協助處理。

變數的命名與宣告

變數的命名

在 C 語言中，變數需要符合以下規則：

1. 開頭不能為數字，例如 `2abc` 是不符規則。
2. 只能使用大小寫英文字母、數字與底線 _ 所組成，例如 `a!b` 與 `a?b` 都是不符規則。

3. 不能使用保留字，所謂保留字就是對 C 語言有意義的字，如 `int` 等。

4. 變數大小寫有差別，例如 `aa`、`Aa`、`aA` 與 `AA` 視為不一樣的變數。

在命名變數時，通常變數使用易懂的方式命名，如計算成績可以宣告由變數 `score` 來儲存，好的變數命名會讓他人更易閱讀程式。

變數的宣告

宣告變數

```
資料型別 變數;
```

例如：`int x;`，宣告 `x` 為整數，C 語言中 `int` 表示整數。

在宣告變數的程式碼中，最後之所以要使用分號，是因為 C 語言規定一句話結束時要使用 `;`，編譯器才知道本句結束了。

宣告變數並初始化

```
資料型別 變數 = 初始值;
```

例如：`int x=100;`，宣告 `x` 為整數，並初始化設定成 `100`。

資料型別

C 語言的資料型別分成整數、浮點數與字元等。整數與浮點數的差異在於數值是否含有小數點，有小數點的數值就是浮點數，未含有小數點的數值就是整數，程式中數值變數都有其空間限制，所表達的數值有其範圍限制，超過範圍後數值就無法表示，運算結果會不正確；任何英文字母、數字與半形標點符號都是字元，而字串可由多個字元所組成。

整數分為 `short`、`int` 與 `long`，其中 `int` 與 `long` 兩資料型別差異不大，要看 C 語言的版本，有的版本視為相同都佔 4 個位元組，有的版本 `long` 佔 4 位元組，`int` 只有 2 個位元組。

資料型別	所佔記憶體空間（位元組）	數值範圍	格式化輸出與輸入代碼
short	2	-32768 ~ 32767	%hi
int	2 或 4	32768 ~ 32767 或 -2147483648 ~ 2147483647	%d
long	4	-2147483648 ~ 2147483647	%ld

浮點數分為單精度與倍精度浮點數，倍精度是單精度儲存空間的兩倍，所以倍精度浮點數較單精度浮點數可以表達的小數點以下位數更多，比較下表的數值範圍就可窺知。

資料型別	所佔記憶體空間（位元）	數值範圍	格式化輸出與輸入代碼
float	4	3.4x10-38 到 3.4x10+38- 3.4x10+38 到 -3.4x10-38 註：準確到小數點以下第 7 位	%f
double	8	1.7x10-308 到 1.7x10+308- 1.7x10+308 到 -1.7x10-308 註：準確到小數點以下第 15 位	%lf

所有的大小寫英文字母、數字、英文標點符號與特殊字元都是字元。

資料型別	所佔記憶體空間（位元）	數值範圍	格式化輸出與輸入代碼
char	1	-128~127	%c

補充說明 字串的格式化輸出與輸入爲 %s，字串爲多個字元所組成，如「abcd」由四個字元所組成就是字串，C 語言中沒有字串的資料型別，需由字元陣列表示。

輸入與輸出函數

C 語言中最常用的輸入與輸出函式為 scanf 與 printf，使用這些函式先事先包含 stdio.h，因 stdio.h 中寫明 scanf 與 printf 的函式表頭，編譯時才找得到這些系統函式，使用方式如下表。

函式與參數	說明	範例

printf
(控制字串 , 參數)
	輸出字串到螢幕	printf("Hello");
		輸出結果：Hello
		printf("Hello,%s","John");
		輸出結果：Hello,John
		註：控制字串除了格式化參數 %s 外，會直接顯示於螢幕，%s 表示這裡保留給字串顯示，第二個參數就會顯示在此。
		printf("%d+%d=%d",2,3,5);
		輸出結果：2+3=5
		註：控制字串除了格式化參數 (%d) 外，會直接顯示於螢幕，三個 %d 表示這裡保留給三個整數顯示，由後面參數依序放入前面格式化參數，如上面字串。

scanf (控制字串 , 參數)	由鍵盤輸入到變數	scanf("%d" , &num); 輸出結果：螢幕會等待使用者輸入數值，數值會存入 num 中。 註：&num 前的 & 在 C 語言中表示取址符號，負責將 num 變數轉成 num 的記憶體位址，使用 scanf 函式的參數，需使用變數的位址。

範例 01 問年齡

寫一程式，螢幕輸出「How old are you?」，等待使用者輸入，假設輸入「25」，螢幕輸出「I am 25 years old」。

提示

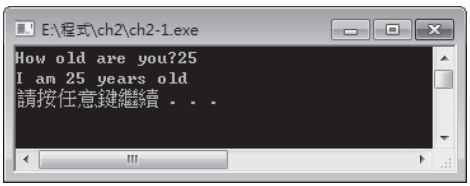
這個程式需要使用 printf 與 scanf 兩函式。

行號	程式碼	說明
1	#include <stdio.h>	第 1 行：爲了 printf 與 scanf 函式
2	#include <stdlib.h>	第 2 行：爲了 system 函式
3	int main(){	第 3~9 行：main 主程式

行號	程式碼	說明
4	`int num;`	第 4 行：宣告 num 爲整數變數
5	`printf("How old are you?");`	第 5 行：螢幕上輸出「How old are you?」
6	`scanf("%d",&num);`	第 6 行：等待輸入，輸入的數值放入變數 num。
7	`printf("I am %d years old",num);`	第 7 行：螢幕上輸出「I am XX years old」，XX 表示鍵盤輸入的數值。
8	`system("pause");`	第 8 行：等待按下任意鍵繼續。
9	`}`	

程式執行結果

輸入 25，結果如右。



補充說明

C 語言中特殊控制字元，如換行、按下 tab 鍵… 等，表示方式如下表。

表示方式	說明
\n	換行字元
\t	相當於按「tab」鍵，用於定位
\0	通常接在字串的最後，表示字串的結束
\a	會使系統發出嗶一聲

2-2 基本運算式

C 語言中有許多種運算子，運算子是用於定義變數與變數間的關係，產生程式設計者預想的結果，運算子依功能可分成指定、算術、關係與邏輯運算子，以下分別說明之。

指定運算子

等號 = 為指定運算子，用於將等號右邊計算結果儲存到等號左邊。例如：

程式	說明
`num=25;`	將 25 儲存入變數 num。
`a=b`	將變數 b 儲存入變數 a。

算術運算子

算術運算子負責數值的運算，如加、減、乘、除與求餘數，分別說明如下表。

運算子	說明	範例
+	相加	B=6,C=3; A=B+C; 計算結果 A=9
–	相減	B=6,C=3 ; A=B-C; 計算結果 A=3
*	相乘	B=6,C=3 ; A=B*C; 計算結果 A=18
/	相除	B=6,C=3 ; A=B/C; 計算結果 A=2
%	求餘數	B=6,C=3 ; A=B%C; 計算結果 A=0

範例 02 溫度單位的轉換

國際上有兩種溫度的表示方式，分別為攝氏與華氏，其間轉換公式如下：

```
F=C*9/5+32
```

請寫出一程式，將攝氏轉成華氏。

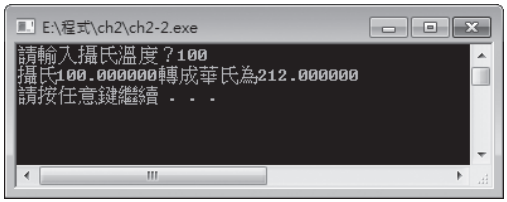
提示

這個程式需要兩個浮點數變數，攝氏命名為 C，華氏命名為 F，利用本節算術運算子，將輸入的攝氏溫度轉成華氏溫度輸出。

行號	程式碼	說明
1	#include <stdio.h>	第 1 行：爲了 printf 與 scanf 函式
2	#include <stdlib.h>	第 2 行：爲了 system 函式
3	int main(){	第 3~10 行：main 主程式
4	float C,F;	第 4 行：宣告 C 與 F 爲浮點數變數
5	printf(" 請輸入攝氏溫度？");	第 5 行：螢幕上輸出「請輸入攝氏溫度？」
6	scanf("%f",&C);	第 6 行:等待輸入，輸入的數值放入變數 C。
7	F=C*9/5+32;	第 7 行：計算華氏溫度。
8	printf(" 攝氏 %f 轉成華氏爲 %f\n",C,F);	第 8 行：螢幕上輸出「攝氏 100.00000 轉成華氏爲 212.00000」。
9	system("pause");	第 9 行：等待按下任意鍵繼續。
10	}	

執行結果

輸入 100，程式的執行結果如下圖。



關係運算子

關係運算子判斷兩變數之間大於小於與等於的關係，以下是關係運算子的說明。

運算子	說明	範例
<	判斷是否小於，若是回傳 true，否則回傳 false	int A=3,B=2 ; A<B， 計算結果爲 false，回傳 0。 註：C 語言中 true 爲 1，false 爲 0。
>	判斷是否大於，若是回傳 true，否則回傳 false	int A=3,B=2 ; A>B， 計算結果爲 true，回傳 1。 註：C 語言中 true 爲 1，false 爲 0。
<=	判斷是否小於等於，若是回傳 true，否則回傳 false	int A=3,B=2 ; A<=B， 計算結果爲 false，回傳 0。
>=	判斷是否大於等於，若是回傳 true，否則回傳 false	int A=3,B=2 ; A>=B， 計算結果爲 true，回傳 1。
==	判斷是否等於，若是回傳 true，否則回傳 false	int A=3,B=2 ; A==B， 計算結果爲 false，回傳 0。
!=	判斷是否不等於，若是回傳 true，否則回傳 false	int A=3,B=2 ; A!=B， 計算結果爲 true，回傳 1。

邏輯運算子

最基本的邏輯運算子有三個且 (&&)、或 (||)、非 (!)，以下就其所對應的運算子分別敘述如下。

X && Y

當 X 與 Y 都為 True，結果為 True，當其中一個為 False，結果為 False。以下表格表示且 (&&) 連結兩個變數 X 與 Y，運算後結果至於表格中。

X && Y	Y=True	Y=False
X=True	True	False
X=False	False	False

X || Y

當X與Y其中一個為 True，結果為 True，當都為 False，結果為 False。

以下表格表示或 (||) 連結兩個變數 X 與 Y，運算後結果至於表格中。

X || Y	Y=True	Y=False
X=True	True	True
X=False	True	False

!X

當X為 True，結果為 False，當 X 為 False，結果為 True。

以下表格表示非 (!) 連結變數 X，運算後結果至於表格中。

	!X
X=True	False
X=False	True

範例 03 判斷體溫使否介於 37 到 38 度

通常溫度介於 37 到 38 度代表體溫正常，輸入一溫度判斷溫度是否介於 37 到 38，若介於 37 到 38 回傳 1，否則回傳 0。

提示

這個程式需要一個浮點數變數，命名為 deg，由鍵盤輸入的溫度儲存到此，需利用關係運算子與邏輯運算子運算輸入的溫度是否介於 37 到 38 度。

行號	程式碼	說明
1	#include <stdio.h>	第 1 行：爲了 printf 與 scanf 函式
2	#include <stdlib.h>	第 2 行：爲了 system 函式
3	int main(){	第 3~10 行：main 主程式
4	float C,F;	第 4 行：宣告 C 與 F 爲浮點數變數
5	printf(" 請輸入攝氏溫度？");	第 5 行：螢幕上輸出「請輸入攝氏溫度？」
6	scanf("%f",&C);	第 6 行:等待輸入，輸入的數值放入變數 C。
7	F=C*9/5+32;	第 7 行：計算華氏溫度。
8	printf(" 攝氏 %f 轉成華氏爲 %f\n",C,F);	第 8 行：螢幕上輸出「攝氏 100.00000 轉成華氏爲 212.00000」。
9	system("pause");	第 9 行：等待按下任意鍵繼續。
10	}	

執行結果

輸入 37.5，結果如下。



隨堂練習

請寫出一程式，能將 a, b 兩數的數值交換。例如程式執行前 a = 1, b = 6，執行後變成 a = 6, b = 1。

隨堂練習

( ) 1. 下列何者為 C 語言正確的變數宣告？
(A) 1A (B) A? (C) A-1 (D)A_1

( ) 2. C 語言中整數的 long int 型別所佔空間為
(A) 1 (B) 2 (C) 4 (D) 8 位元組

( ) 3. C 語言中浮點數的 double 型別所佔空間為
(A) 1 (B) 2 (C) 4 (D) 8 位元組

( ) 4. C 語言中宣告變數 A 為整數，使用「_ A」語法，請問底線處應填入
(A) float (B) double (C) char (D) int

( ) 5. C 語言中浮點數的格式化輸出與輸入為何？
(A) %c (B) %d (C) %f (D) %hi

( ) 6. C 語言中下列為何表示換行？
(A) \n (B) /n (C) \t (D) /t

( ) 7. C 語言中算術運算子下列為求餘數？
(A) + (B) - (C) * (D) %

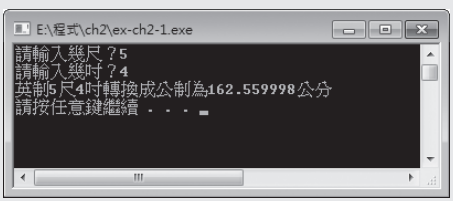
( ) 8. C 語言中關係運算子下列為判斷「不相等」？
(A) <> (B) >= (C) <= (D) !=

英吋與公分轉換

英制的身高以幾尺幾吋表示，而公制以公分表示，現在寫一程式將英制轉換成公制，由鍵盤輸入幾英尺幾英吋，轉換成公分輸出。轉換公式如下。

假設：A 英尺 B 英吋轉成 C 公分，公式為 C=(A*12+B)*2.54

執行範例：輸入「5」尺「4」吋，轉換成「162.559998」公分



詳解請參閱光碟中的範例檔「ch2\ex-ch2-1.cpp」。

Chapter 03

選擇結構

3-1 單向與雙向選擇結構

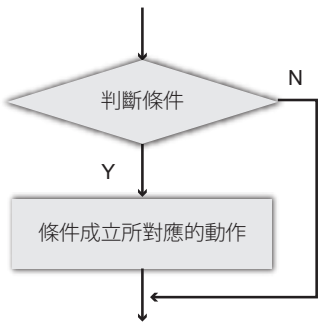
日常生活中就有許多選擇結構的例子，例如：如果下雨，出門就要帶傘；如果國文考 60 分，國文就及格，否則國文不及格等，都是選擇結構。C 語言也提供選擇結構，若條件成立，則做條件成立的動作，否則做條件不成立的動作。

單向結構

單向選擇是較簡單的選擇結構，只有若條件成立，則做對應的動作。單向結構的語法表示如下：

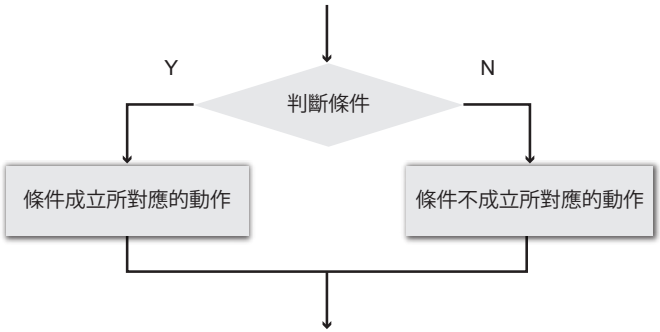
```
if (條件) {
  條件成立所對應的動作;
}
```

單向結構的流程圖表示如下圖。



雙向選擇結構

雙向選擇是單向選擇的延伸，若條件成立，則做條件成立時對應的動作，否則做條件不成立時對應的動作，流程圖及語法的表示如下：




```
if (條件) {
   條件成立所對應的動作;
} else {
   條件不成立所對應的動作;
}
```

範例 01 判斷成績及格

由使用者輸入一個成績，程式判斷成績是否及格，及格則輸出「很棒，繼續保持」，否則輸出「要加油」

想法

成績輸入到一個整數變數，在使用雙向選擇結構判斷成績是否及格，及格與否做對應的動作。

行號	程式碼	說明
1	#include <stdio.h>	第 1 行：爲了 printf 與 scanf 函式
2	#include <stdlib.h>	第 2 行：爲了 system 函式
3	int main(){	第 3~13 行：main 主程式
4	int score;	第 4 行：宣告 score 爲整數變數
5	printf(" 請輸入成績 ");	第 5 行：螢幕上輸出「請輸入成績」
6	scanf("%d",&score);	第 6 行：等待輸入，輸入的數值放入變數 score。
7	if (score >= 60){	第 7~11 行：雙向選擇結構，程式判斷成績是否及格（第 7 行），及格則輸出「很棒，繼續保持」（第 8 行），否則輸出「要加油」（第 10 行）。

行號	程式碼	說明
8	printf(" 很棒，繼續保持 \n");	
9	} else {	
10	printf(" 要加油 \n");	
11	}	
12	system("pause");	第 12 行：等待按下任意鍵繼續。
13	}	

程式執行結果

輸入 60，結果如下圖。



輸入 59，結果如下圖。



範例 02　取數值的絕對值

由使用者輸入一個可正可負的整數數值，程式將負整數取絕對值，正整數則不變。

想法

輸入值儲存到一個整數變數，在使用雙向選擇結構判斷輸入值是否大於零，輸入值大於零則不變，輸入值小於零則取負數。

行號	程式碼	說明
1	`#include <stdio.h>`	第 1 行：爲了 printf 與 scanf 函式
2	`#include <stdlib.h>`	第 2 行：爲了 system 函式
3	`int main(){`	第 3~13 行：main 主程式
4	`int num;`	第 4 行：宣告 num 爲整數變數
5	`printf(" 請輸入整數值？");`	第 5 行：螢幕上輸出「請輸入整數值？」
6	`scanf("%d",&num);`	第 6 行：等待輸入，輸入的數值放入變數 num。
7	`if (num >= 0){`	第 7~11 行：雙向選擇結構，程式判斷輸入值 num 是否大於等於 0（第 7 行），若是則直接輸出（第 8 行），否則輸入值爲負值，取負號變正值再輸出（第 10 行）。
8	`printf(" 輸入值取絕對值後爲 %d\n",num);`	
9	`} else {`	
10	`printf(" 輸入值取絕對值後爲 %d\n",-num);`	
11	`}`	
12	`system("pause");`	第 12 行：等待按下任意鍵繼續。
13	`}`	

程式執行結果

輸入 **1**，結果如下圖。



輸入 **-1**，結果如下圖。



3-2 巢狀選擇結構

在很多時候選擇結構不止雙向，可能分類有好幾種，這時就要使用巢狀選擇結構。體溫與發燒狀態為例，體溫與對應的狀態如下表，就需使用巢狀選擇結構。

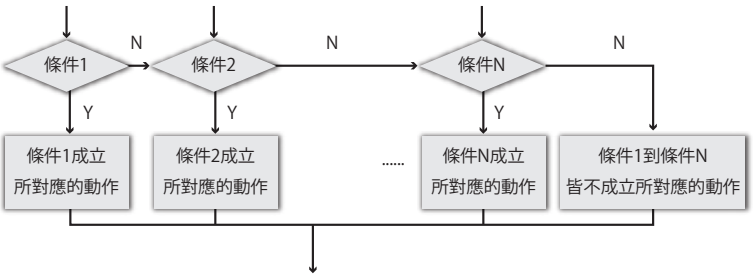
體溫	發燒狀態
體溫 >=39	發高燒
39> 體溫 >=38	有點燒
38> 體溫 >=36	體溫正常
36> 體溫	體溫過低

巢狀選擇結構的語法如下。右邊為將體溫與發燒狀態代入結構的演算法。

巢狀選擇結構語法	將體溫與發燒狀態代入結構

if (條件 1) {
　條件 1 成立所對應的動作；
} else if (條件 2){
　條件 2 成立所對應的動作；
}else if(條件 3){
　條件 3 成立所對應的動作；
}else {
　條件 1、條件 2 與條件 3 皆不成立所對應的動作；
}
	if (體溫 >=39) {
	顯示「發高燒」
	} else if (39> 體溫 >=38){
	顯示「有點燒」
	}else if(38> 體溫 >=36){
	顯示「體溫正常」
	}else {
	顯示「體溫過低」
	}

巢狀選擇結構的流程圖如下。



以下的實例練習，就將體溫與發燒狀態判斷轉成程式。

範例 03 體溫與發燒判斷

說明

體溫與發燒狀態的對應如下表。

體溫	發燒狀態
體溫 >=39	發高燒
39> 體溫 >=38	有點燒
38> 體溫 >=36	體溫正常
36> 體溫	體溫過低

說明

發燒狀態有四種，使用單向與雙向選擇結構都不適用，使用巢狀選擇結構較適合本範例，使用巢狀選擇結構撰寫程式如下。

行號	程式碼	說明
1	`#include <stdio.h>`	第 1 行：爲了 printf 與 scanf 函式
2	`#include <stdlib.h>`	第 2 行：爲了 system 函式
3	`int main(){`	第 3~17 行：main 主程式
4	`  int deg;`	第 4 行：宣告 deg 爲整數變數
5	`  printf(" 請輸入體溫？");`	第 5 行：螢幕上輸出「請輸入體溫？」
6	`  scanf("%d",&deg);`	第 6 行：等待輸入，輸入的數值放入變數 deg。
7	`  if (deg >= 39){`	第 7~15 行：巢狀選擇結構，程式判斷輸入值 deg 是否大於等於 39（第 7 行），若是則輸出「發高燒」（第 8 行），否則判斷 deg 是否小於 39 且大於等於 38（第 9 行），若是則輸出「有點燒」（第 10 行），否則判斷 deg 是否小於 38 且大於等於 36（第 11 行），若是則輸出「體溫正常」（第 12 行），否則輸出「體溫過低」（第 14 行）。
8	`    printf(" 發高燒 \n");`	
9	`  } else if ((deg<39) &&(deg>=38)){`	
10	`    printf(" 有點燒 ");`	
11	`  } else if  ((deg<38) &&(deg>=36)){`	
12	`    printf(" 體溫正常 ");`	
13	`  } else {`	
14	`    printf(" 體溫過低 ");`	
15	`  }`	
16	`  system("pause");`	第 16 行：等待按下任意鍵繼續。
17	`}`	

補充說明 第 9 行的 `deg<39` 可以刪除，因爲到第 9 行的 `else` 時，需先經過第 7 行判斷是否大於等於 39，條件不成立才會到第 9 行，因此隱含 `deg` 小於 39 的意思。同理第 11 行的 `deg<38` 也可刪除。

程式執行結果

輸入 39，結果如右。



輸入 37，結果如右。



3-3 Switch 結構

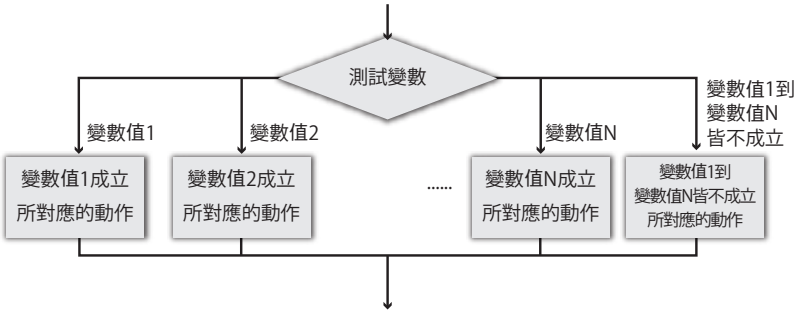
前一節巢狀選擇結構程式撰寫太過複雜且容易出錯，因此 C 語言提供 Switch 結構簡化巢狀選擇結構，Switch 結構的語法及流程圖如下。

Switch結構語法

```
switch(變數){
 case  變數值1:
   符合變數值1所對應的動作;
   break;
 case  變數值2:
   符合變數值2所對應的動作;
   break;
…
 case  變數值K:
   符合變數值K所對應的動作;
   break;
 default:
   預設所對應的動作;
   break;
}
```

將體溫與發燒狀態代入結構

```
switch(deg){
 case 39 ... 50:
   printf("發高燒\n");
   break;
 case 38:
   printf("有點燒\n");
   break;
 case 37:
 case 36:
   printf("體溫正常\n");
   break;
 case 30 ... 35:
   printf("體溫過低\n");
   break;
}
```



範例 04 體溫與發燒判斷以 switch-case 表示

說明

體溫與發燒狀態的對應如下表。

體溫	發燒狀態
體溫 >=39	發高燒
39> 體溫 >=38	有點燒
38> 體溫 >=36	體溫正常
36> 體溫	體溫過低

想法

發燒狀態有四種，使用單向與雙向選擇結構都不適用，使用巢狀選擇結構較適合本範例，除了巢狀選擇結構外，也可使用 switch 結構撰寫，寫起來更簡潔，以下使用 switch 結構撰寫程式。

行號	程式碼	說明
1	`#include <stdio.h>`	第 1 行：爲了 printf 與 scanf 函式
2	`#include <stdlib.h>`	第 2 行：爲了 system 函式
3	`int main(){`	第 3~23 行：main 主程式
4	`  int deg;`	第 4 行：宣告 deg 爲整數變數
5	`  printf(" 請輸入整數體溫？");`	第 5 行:螢幕上輸出「請輸入整數體溫？」
6	`  scanf("%d",&deg);`	第 6 行：等待輸入，輸入的數值放入變數 deg。
7	`  switch(deg){`	第 7~23 行：switch-case 結構，程
8	`  case 39 ... 50:`	式判斷輸入值 deg 是否介於 39 到 50

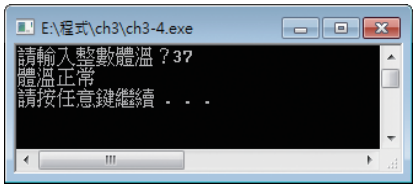
行號	程式碼	說明
9	printf(" 發高燒 \n");	(第 8 行) , 若是則輸出「發高燒」(第 9 行) , 否則判斷 deg 是否等於 38 (第 11 行) , 若是則輸出「有點燒」(第 12 行) , 否則判斷 deg 是否介於 36 到 37 (第 14~15 行) , 若是則輸出「體溫正常」(第 16 行) , 否則判斷 deg 是否介於 30 到 35 (第 18 行) , 若是則輸出「體溫過低」(第 19 行) 。
10	break;	
11	case 38:	
12	printf(" 有點燒 \n");	
13	break;	
14	case 37:	
15	case 36:	
16	printf(" 體溫正常 \n");	
17	break;	
18	case 30 ... 35:	
19	printf(" 體溫過低 \n");	
20	break;	
21	}	
22	system("pause");	第 22 行 : 等待按下任意鍵繼續。
23	}	

程式執行結果

輸入 39，結果如右圖。



輸入 37，結果如右圖。



隨堂練習

1. 學生成績五等第畫分的方式如下：

   A：90 分以上、B：80 分以上未滿 90 分、C：70 分以上未滿 80 分、D：60 分以上未滿 70 分、E：未滿 60 分

   寫一個程式，能讓使用者輸入成績，並根據成績的高低，輸出對應的等第。

隨堂練習

2. 依油品分類為 92 無鉛、95 無鉛與 98 無鉛，計算加油金額，輸入油品分類與加油公升數，加油金額為油品價格乘以加油公升數，油品價格如下。

油品	每公升價格
92 無鉛（以 92 代表）	29.1
95 無鉛（以 95 代表）	29.7
98 無鉛（以 98 代表）	30.2

詳解請參閱範例檔中的「ch3/ex-ch3-1.cpp」，執行結果如下圖。



3. 設計一個電腦程序，讓使用者輸入身高和體重後，可依下表自動判定身體質量的等級。

分級	身體質量指數
體重過輕	BMI<18.5
正常範圍	18.5≤BMI<24
體重過重	24≤BMI<27
輕度肥胖	27≤BMI<30
中度肥胖	30≤BMI<35
重度肥胖	30≤BMI

其中，$BMI=\frac{體重\ (kg)}{身高^2\ (m^2)}$

(解題方法可參考高中資訊科技概論課本第 301 頁)

4. 寫一個程式，可以輸入西元年分，並能自動判定該年是否為閏年。
(解題流程可參考高中資訊科技概論課本第 331 頁)

Chapter 04

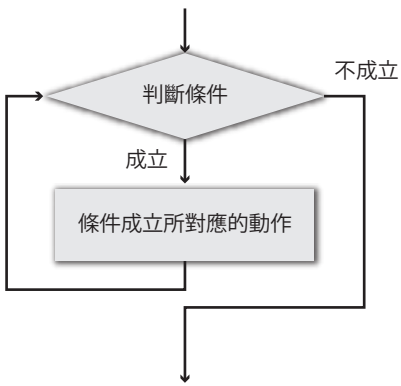
重複結構

C 語言程式的三大架構為**循序**、**選擇**與**重複**，循序架構為程式是由前往後一步一步執行，這樣的特性大家已經很熟悉，在說明程式的運作，我們也是從第 1 行開始依序一行一行說明，與循序架構呼應；選擇架構為前一章所介紹，若條件成立，則做條件成立所對應的動作，例如：若成績大於 60 分，則成績及格等；本章要介紹重複結構，重複結構是讓電腦重複做類似的動作，電腦之所以功能強大在於可以快速重複做相同的動作，人腦無法在如此短的時間內，重複作指令運算那麼多次，藉由電腦強大的計算能力不斷重複做。。

4-1 迴圈結構

利用迴圈結構達成重複做相同或類似的工作多次，而迴圈結構最基本的為 `for` 迴圈，`for` 迴圈需指定初始值、測試條件、與遞增減值所組成，迴圈結構如下：

```
for(迴圈變數=初始值;迴圈變數測試條件是否
    (成立;迴圈變數遞增或遞減){
  重複的動作;
}
```



範例 01 印出 1 到 1000 的所有數到螢幕

說明：

使用迴圈結構，讓螢幕顯示出 1、2、3… 到 1000 的數字。

想法

使用迴圈結構較容易達成，迴圈初始值 1，測試條件需小於等於 1000，每次加 1。

行號	程式碼	說明
1	`#include <stdio.h>`	第 1 行：爲了 printf 函式
2	`#include <stdlib.h>`	第 2 行：爲了 system 函式
3	`int main(){`	第 3~8 行：main 主程式
4	`  for(int i=1;i<1001;i++){`	第 4~6 行：迴圈結構，i 由 1 變化到 1000
5	`    printf("%d\n",i);`	第 5 行:螢幕上依序輸出「1 到 1000 的數字」
6	`}`	
7	`   system("pause");`	第 7 行：等待按下任意鍵繼續。
8	`}`	

程式執行結果

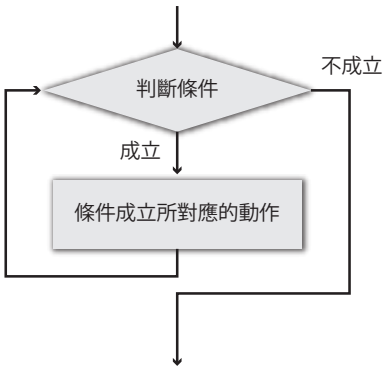
此範例的執行結果如下圖。



4-2 前測式重複結構

迴圈除了 `for` 迴圈外，還有 `while` 迴圈，兩者功能差不多，其語法結構及流程圖如下。

```
迴圈變數初始化；
while(迴圈測試條件){
    迴圈內動作；
    迴圈變數遞增減；
}
```



範例 02 印出 1 到 1000 的所有數到螢幕

說明

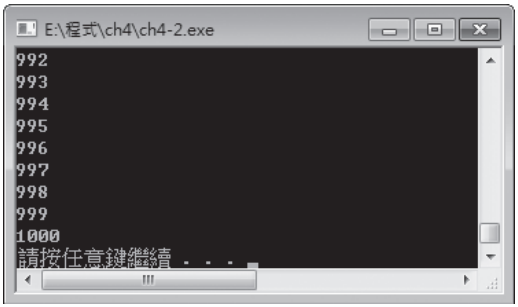
使用前測試重複結構，讓螢幕顯示出 1、2、3… 到 1000 的數字。

想法

迴圈初始值 1，測試條件需小於等於 1000，每次加 1。

行號	程式碼	說明
1	`#include <stdio.h>`	第 1 行：爲了 printf 函式
2	`#include <stdlib.h>`	第 2 行：爲了 system 函式
3	`int main(){`	第 3~10 行：main 主程式
4	`  int i=1;`	第 4 行：迴圈變數 i 初始化爲 1
5	`  while(i<1001){`	第 5~8 行：迴圈測試變數 i 是否小於 1001（第 5 行），印出數值到螢幕（第 6 行），迴圈變數 i 每次加 1（第 7 行），經由迴圈結構變數 i 由 1 每次遞增 1 到 1000 爲止。
6	`    printf("%d\n",i);`	
7	`    i=i+1;`	
8	`  }`	
9	`  system("pause");`	第 9 行：等待按下任意鍵繼續。
10	`}`	

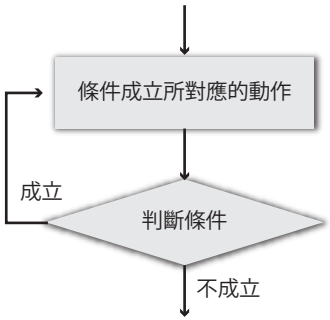
程式執行結果



4-3 後測式重複結構

前側與後側重複結構兩者功能差不多，只是後側重複結構至少做一次，其流程圖及語法結構如下。

```
   迴圈變數初始化；
do{
   迴圈內動作；
   迴圈變數遞增減；
}while(迴圈測試條件)
```



範例 03 印出所有大寫英文字母

對電腦而言，英文、數字與其他控制字元等儲存到硬碟是由 0 與 1 所組成，ASCII 碼負責文字與編碼的對應，ASCII 碼中大寫 A 對應的二進位編碼為 01000001，大寫 A 儲存在硬碟中就變成 01000001，轉成十進位為 65，大寫 B 對應的二進位編碼為 01000010，大寫 B 儲存在硬碟中就變成 01000010，轉成十進位為 66，依此類推，大寫 C 的十進位編碼為 67。我們可以利用此特性，利用迴圈與 ASCII 編碼印出所有大寫英文字母。

想法

使用迴圈結構，迴圈初始值 65，測試條件需小於等於 90，每次加 1，利用 `printf` 函式印出數字所對應的字元。

行號	程式碼	說明
1	#include <stdio.h>	第 1 行：爲了 printf 函式
2	#include <stdlib.h>	第 2 行：爲了 system 函式
3	int main(){	第 3~10 行：main 主程式
4	int i=65;	第 4 行：迴圈變數 i 初始化爲 65
5	do{	第 5~8 行：印出數值與字元，相同數值使用 %d 顯示數值，%c 顯示字元，會將數值轉換成對應的字元（第 6 行），變數 i 遞增 1（第 7 行），迴圈測試變數 i 是否小於等於 90（第 8 行），在後測式重複結構，測式條件最後才測試，所以迴圈至少要執行一次，測試條件若不成立才結束迴圈。
6	printf("數值 %d 所對應的字元爲 %c\n",i,i);	
7	i=i+1;	
8	} while (i<=90);	
9	system("pause");	第 9 行：等待按下任意鍵繼續。
10	}	

程式執行結果

此範例的執行結果如右圖。



範例 04 計算 1+2+3+4+...+50 之加總

想法

使用迴圈結構，迴圈初始值 1，測試條件需小於等於 50，每次加 1，迴圈內變數值由 1、2、3… 依序變化到 50，依序加總迴圈內的變數值。

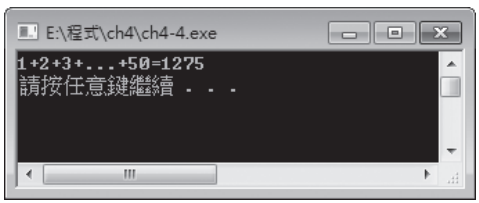
行數	程式碼	說明
1	`#include <stdio.h>`	第 1 行：爲了 printf 函式
2	`#include <stdlib.h>`	第 2 行：爲了 system 函式
3	`int main(){`	第 3~10 行：main 主程式
4	`  int i=1,sum=0;`	第 4 行：迴圈變數 i 初始化爲 1，sum 初始化爲 0
5	`  do{`	第 5~8 行：變數 sum 暫存加總值（第 6 行），變數 i 遞增 1（第 7 行），迴圈測試變數 i 是否小於等於 50（第 8 行），在後測式重複結構，測式條件最後測試，所以迴圈至少要執行一次，測試條件若不成立才結束迴圈。
6	`    sum=sum+i;`	
7	`    i=i+1;`	
8	`  } while (i<=50);`	
9	`  printf("1+2+3+...+50=%d\n",sum);`	第 9 行：印出加總結果。
10	`  system("pause");`	第 10 行：等待按下任意鍵繼續。
11	`}`	

迴圈中 `sum=sum+i` 的運作原理，迴圈中 `i` 值變化與 `sum` 值的對應，如下表。

i 值	sum 值
i=1	sum=sum+i=0+1=1
i=2	sum=sum+i=1+2=3=1+2
i=3	sum=sum+i=3+3=6=1+2+3
i=4	sum=sum+i=6+4=10=1+2+3+4
…	…
i=49	sum=sum+i=1176+49=1+2+3+4+…+49
i=50	sum=sum+i=1225+50=1275=1+2+3+4+…+49+50

程式執行結果

此範例的執行結果如下圖。



4-4 巢狀重複結構

巢狀重複結構不用想得太複雜，其實巢狀迴圈為迴圈作用範圍內又有一個迴圈，外層迴圈稱外迴圈，而內層迴圈稱內迴圈，巢狀迴圈可以有好幾層，其運作原理與單層迴圈相同，由外層迴圈來看，內層迴圈只是外層迴圈重複的動作，也就是外層迴圈執行一次，內層迴圈要執行好幾次直到完成，我們將以九九乘法表為範例，當外層迴圈執行一次，內層迴圈要執行九次，當外層迴圈執行九次，內層迴圈總共執行 81 次，如此產生九九乘法表。

範例 05 列印九九乘法表

請寫出一程式印出九九乘法表，使其執行結果如下：

```
1*1=1  1*2=2  …  1*9=9

2*1=2  2*2=4  …  2*9=18

…

9*1=9  9*2=18  …  9*9=81
```

想法：

使用巢狀迴圈結構，外層迴圈初始值 1，測試條件需小於等於 9，每次加 1，內層迴圈初始值 1，測試條件需小於等於 9，每次加 1，依序印出內外迴圈變數值與相成結果。

行號	程式碼	說明
1	`#include <stdio.h>`	第 1 行：爲了 printf 函式
2	`#include <stdlib.h>`	第 2 行：爲了 system 函式
3	`int main(){`	第 3~12 行：main 主程式
4	`  int i,j;`	第 4 行：宣告變數 i 與 j 爲整數
5	`  for(i=1;i<=9;i++){`	第 5~10 行：印出九九乘法表，外層迴圈的變數 i 變化由 1、2、3 …到 9（第 5 行），內層迴圈的變數 i 變化由 1、2、3 …到 9（第 6 行），印出每一個 i、j 與 i 乘 j 的結果（第 7 行），內層迴圈結束後印出換行字元（第 9 行）

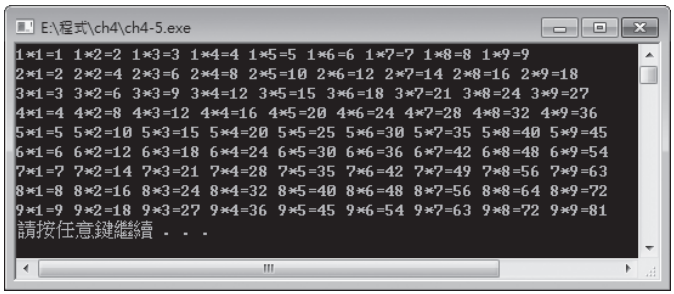
行號	程式碼	說明
6	for(j=1;j<=9;j++){	
7	printf("%d*%d=%d ",i,j,i*j);	
8	}	
9	printf("\n");	
10	}	
11	system("pause");	第 11 行：等待按下任意鍵繼續。
12	}	

巢狀迴圈中 `printf("%d*%d=%d ",i,j,i*j);` 的運作原理，外層迴圈中 `i` 值變化、內層迴圈中 `j` 值的變化與印出資料，如下表。

外層 i 值	內層 j 值	printf("%d*%d=%d ",i,j,i*j) 結果
i=1	j=1,2,3,4,5,6,7,8,9	1*1=1 1*2=2 … 1*9=9
i=2	j=1,2,3,4,5,6,7,8,9	2*1=2 2*2=4 … 2*9=18
i=3	j=1,2,3,4,5,6,7,8,9	3*1=3 3*2=6 … 3*9=27
…	…	…
i=9	j=1,2,3,4,5,6,7,8,9	9*1=9 9*2=18 … 9*9=81

程式執行結果

此範例的執行結果如下圖。



隨堂練習

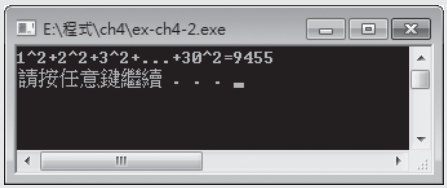
1. 使用程式計算 1+3+5+7+…+999 的值，使其執行結果如右圖。

   詳解請參閱範例檔中的「ch4/ex-ch4-1.cpp」。



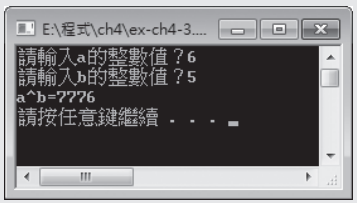
2. 使用程式計算 1^2+2^2+3^2+...+30^2 的值，使其執行結果如右圖。

   詳解請參閱範例檔中的「ch4/ex-ch4-2.cpp」。



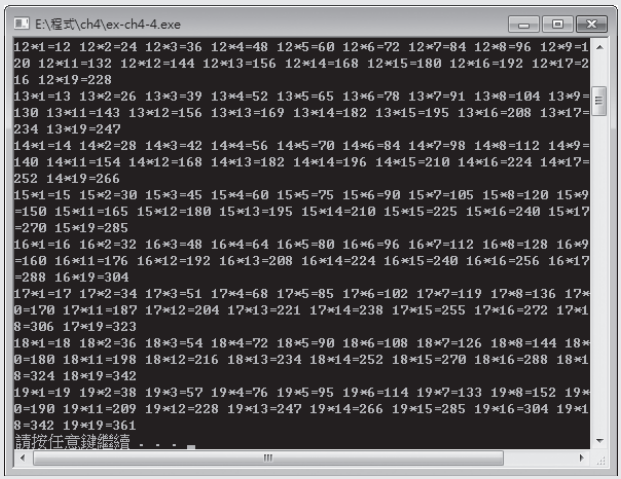
3. 使用程式計算 a^b 的值，使其執行結果如右圖。

   詳解請參閱範例檔中的「ch4/ex-ch4-3.cpp」



4. 印度是記 19 乘 19 乘法表，請寫一程式印出 19 乘 19 乘法表，使其執行結果如右圖。

   詳解請參閱範例「ch4/ex-ch4-4.cpp」。



5. 寫一個程式，讓使用者可以輸入任意兩個大於零整數數，該程式可以找出此兩數的最大公因數及最小公倍數。

   (解題方法可參考高中資訊科技概論課本第 336 頁)

Chapter

05 陣列

陣列是多個相同資料型別的變數綁在一起，佔有連續的記憶體，可以利用索引值 (index) 存取陣列元素，陣列第一個元素其索引值為 0，第二個元素其索引值為 1，依此類推，每個索引值對應唯一的陣列元素，n 個元素的陣列，存取陣列最後一個元素其索引值為 n-1，我們可以利用迴圈與陣列結合寫出複雜的程式。

5-1 一維陣列

假設我們要計算第一次段考國文全班總分，不使用陣列改用變數宣告，sco1 表示 1 號同學的國文成績，sco2 表示 2 號同學的國文成績，若全班 20 位同學，依此類推第 20 同學的國文成績以 sco20 表示，這時計算加總時無法使用重複結構需要寫成如下格式：

```
sum=sco1+sco2+soc3+sco4+sco5+sco6+sco7+soc8+sco9+sco10+sco11+sco12+soc13
+sco14+sco15 +sco16+sco17+soc18+sco19+sco20;
```

但是若要計算全校國文總分時，全校有 500 位學生，用這種方式撰寫程式就顯得很麻煩。

可以宣告一個 `sco` 成績陣列，有 20 個元素，陣列是以索引方式存取，也就是 `sco[index]` 代表為存取 `sco` 陣列中索引值為 `index` 的元素，索引值是陣列從頭到尾由零開始編號，也就是陣列中第一個元素 (索引值為 0) 存放 1 號同學的國文成績以 `sco[0]` 表示、陣列中第二個元素 (索引值為 1) 存放 2 號同學的國文成績以 `sco[1]` 表示、… 、依此類推陣列中第二十個元

素 (索引值為 19) 存放 20 號同學的國文成績以 `sco[19]` 表示，陣列儲存於電腦的示意圖如下。



使用陣列計算總分時，使用重複結構與陣列索引存取的特性，可以寫出簡潔的程式。

```
for( int  i; i<20;i++){
  sum=sum+sco[i];
}
```

這就是使用陣列將相同資料型別資料放在一起，使用重複結構與陣列索引存取特性就可以寫出存取陣列中元素的程式。

陣列的宣告

```
資料型別  陣列名稱 [ 元素個數 ];
```

例如：

```
int sco[20]
```

宣告一個整數陣列，名稱為 sco，內有 20 個元素。

陣列的宣告與初始化

```
資料型別  陣列名稱 [ 元素個數 ]={ 陣列第 1 個元素值 , 陣列第 2 個元素
值 , … , 陣列最後一個元素值 };
```

例如

```
int sco[20]={80,81,82,83,84,85,86,87,88,89,80,81,82,83,84,85,86,87,88,89}
```

宣告一個整數陣列，名稱為 sco，內有 20 個元素，`sco[0]` 其值為 80，`sco[1]` 其值為 81，依此類推 `sco[19]` 其值為 89。

5-2 實例練習—計算總分與平均數

請寫出一程式計算第一次段考全班 20 位同學國文成績之總分與平均數。

想法

使用陣列儲存全班成績，利用迴圈結構，迴圈初始值 1，測試條件需小於 20，每次加 1，迴圈內變數值由 0、1、2、3… 依序變化到 19，利用此為陣列索引值依序存取與加總陣列元素的值，再除以 20 即可得平均數。

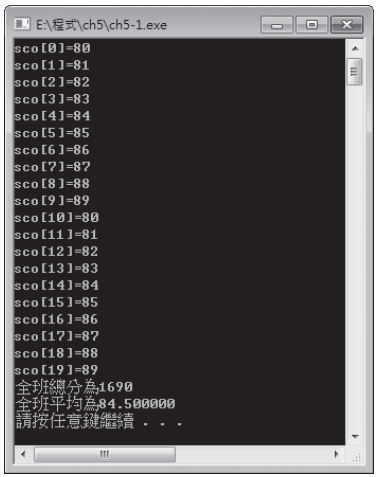
行號	程式碼	說明
1	`#include <stdio.h>`	第 1 行：爲了 printf 函式
2	`#include <stdlib.h>`	第 2 行：爲了 system 函式
3	`int main(){`	第 3~16 行：main 主程式
4	`  int sco[20]={80,81,82,83,84,85,86,87,88,89,80,81,82,83,84,85,86,87,88,89};`	第 4 行：宣告 sco 成績陣列並初始化。
5	`  int i=0,sum=0;`	第 5 行：迴圈變數 i 初始化爲 0，sum 初始化爲 0
6	`  float avg;`	第 6 行：宣告 avg 爲浮點數變數
7	`  do{`	第 7~11 行：變數 sum 暫存加總值（第 8 行），印出陣列中索引值爲 i 的值（第 9 行），變數 i 遞增 1（第 10 行），迴圈測試變數 i 是否小於 20（第 11 行）。
8	`    sum=sum+sco[i];`	
9	`    printf("sco[%d]=%d\n",i,sco[i]);`	
10	`    i=i+1;`	
11	`  } while (i<20);`	
12	`  avg=(float)sum/20;`	第 12 行：計算平均，(float)sum，強迫將 sum 由整數轉換成浮點數資料型別。
13	`  printf(" 全班總分爲 %d\n",sum);`	第 13 行：印出全班總分。
14	`  printf(" 全班平均爲 %f\n",avg);`	第 14 行：印出全班平均。
15	`  system("pause");`	第 15 行：等待按下任意鍵繼續。
16	`}`	

迴圈中 `sum=sum+sco[i]` 的運作原理，迴圈中 i 值變化與 `sum` 值的對應，如下表。等號右邊的結果 `sum+sco[i` 先運算，結果儲存入等號的左邊 `sum`。

i 值	sum 值
i=0	sum=sum+sco[0]=sco[0]
i=1	sum=sum+sco[1]=sco[0]+sco[1]
i=2	sum=sum+sco[2]=sco[0]+sco[1]+sco[2]
i=3	sum=sum+sco[3]=sco[0]+sco[1]+sco[2]+sco[3]
i=4	sum=sum+sco[4]=sco[0]+sco[1]+sco[2]+sco[3]+sco[4]
…	…
i=18	sum=sum+sco[18]=sco[0]+sco[1]+sco[2]+sco[3]+sco[4]+…+sco[18]
i=19	sum=sum+sco[19]=sco[0]+sco[1]+sco[2]+sco[3]+sco[4] +…+sco[18]+sco[19]

程式執行結果

執行結果如右圖。



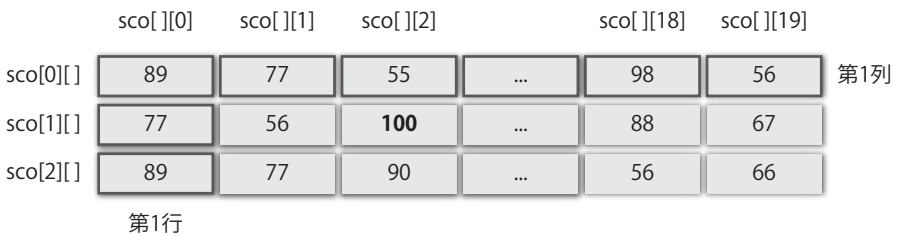
5-3　二維陣列

我們已經介紹過了一維陣列，舉例計算全班二十位同學國文科成績總分。若計算全班二十位同學的國英數三科各科總分，可以使用三個一維陣列，每個陣列各計算一次，就可獲得各科總分。

除此之外，我們也可以使用二維陣列 (橫的為列，直的為行)，將國英數三科成績以列區隔、國文成績儲存在第一列、英文成績儲存在第二列、數學成績儲存在第三列；將同一科目二十個同學成績儲存在同一列的各行，第

一位同學儲存在第 1 行，第二位同學儲存在第 2 行，第二十位同學儲存在第 20 行，這樣需要 3 列 20 行的二維陣列 (假設陣列名稱為 sco)，如下圖。

陣列 sco 的第一個中括號表示第幾列，第二個中括號表示第幾行，陣列中括號內數字表示陣列索引值，索引值由 0 開始，所以英文科的第三位同學成績為 100 分，儲存在陣列 sco 的第 2 列第 3 行 (以 sco[1][2] 表示)。



使用二維陣列計算各科總分時，使用巢狀重複結構與陣列索引存取的特性，可以寫出簡潔的程式，如以下程式範例。

```
for( int  i=0; i<3;i++){
  for( int  j=0; j<20;j++){
    sum[i]=sum[i]+sco[i][j];
  }
}
```

這就是使用陣列將相同資料型別資料放在一起，使用巢狀重複結構與陣列索引存取，就可以寫出存取陣列中每一個元素的程式。

二維陣列的宣告

```
資料型別  陣列名稱 [ 列數 ][ 行數 ];
```

例如：`int sco[3][20]`，宣告一個二維整數陣列，名稱為 sco，內含 3 列 20 行共 60 個元素。

陣列的宣告與初始化

```
資料型別  陣列名稱 [ 列數 ][ 行數 ]={{ 陣列第 1 列第 1 個元素值 , 陣列第 1 列第 2 個元素值 , … , 陣列第 1 列最後一個元素值 },{ 陣列第 2 列第 1 個元素值 , 陣列第 2 列第 2 個元素值 , … , 陣列第 2 列最後一個元素值 },…,{ 陣列最後一列第 1 個元素值 , 陣列最後一列第 2 個元素值 , … , 陣列最後一列最後一個元素值 }};
```

例如：int

```
sco[3][20]={{80,81,82,83,84,85,86,87,88,89,80,81,82,83,84,85,86,87,88,89
},{99,90,91,92,83,84,85,86,87,88,89,80,81,82,83,84,85,86,87,100},{80,81,
82,83,84,85,86,87,88,89,80,81,82,83,84,85,86,87,88,89}}
```

宣告一個整數陣列，名稱為 sco，內有 3 列 20 行共 60 個元素，`sco[0][0]` 其值為 80，`sco[0][1]` 其值為 81，依此類推 `sco[0][19]` 其值為 89，`sco[1][0]` 其值為 99，`sco[1][1]` 其值為 90，依此類推 `sco[1][19]` 其值為 100，`sco[2][0]` 其值為 80，`sco[2][1]` 其值為 81，依此類推 `sco[2][19]` 其值為 89。

範例 01　計算各科總分與平均數

請寫出一程式計算第一次段考全班 20 位同學國文、英文與數學成績之總分與平均數。

想法

使用二維陣列儲存全班成績，利用巢狀迴圈結構，外層迴圈迴初始值 0，測試條件需小於 3，每次加 1，迴圈內變數值由 0、1、2，內層迴圈初始值 0，測試條件需小於 20，每次加 1，迴圈內變數值由 0、1、2、3… 依序變化到 19，利用此為陣列索引值依序存取與加總陣列中各列元素的值，再除以 20 即可得各科平均。

行號	程式碼	說明
1	`#include <stdio.h>`	第 1 行：爲了 printf 函式
2	`#include <stdlib.h>`	第 2 行：爲了 system 函式
3	`int main(){`	第 3~19 行：main 主程式
4	`int sco[3][20]={{80,81,82,83,84,85,86,87,88,89,80,81,82,83,84,85,86,87,88,89},{99,90,91,92,83,84,85,86,87,88,89,80,81,82,83,84,85,86,87,100},{80,81,82,83,84,85,86,87,88,89,80,81,82,83,84,85,86,87,88,89}};`	第 4 行：宣告 sco 二維成績陣列並初始化。
5	`int i,j,sum[3]={0,0,0};`	第 5 行：宣告迴圈變數 i 與 j，一維整數陣列 sum，初始化所有元素皆爲 0

行號	程式碼	說明
6	float avg;	第 6 行：宣告 avg 爲浮點數變數
7	for(i=0;i<3;i++){	第 7~12 行：使用巢狀迴圈存取二維成績陣列 sco 的每一個元素，外層迴圈 i 值由 0、1、2 依序變化（第 7 行），內層迴圈 j 值由 0、1、2、… 到 19（第 8 行），整數陣列 sum[i] 暫存陣列 sco 第 i 列的加總值（第 9 行），印出陣列中索引值爲第 i 列第 j 行的值（第 10 行）。
8	for(j=0;j<20;j++){	
9	sum[i]=sum[i]+sco[i][j];	
10	printf("sco[%d][%d]= %d\n",i,j,sco[i][j]);	
11	}	
12	}	
13	for(i=0;i<3;i++){	第 13~17 行：計算國英數三科平均，迴圈 i 值由 0、1、2 依序變化（第 13 行）存取陣列 sum[i] 的值，sum[i] 前使用 (float)，強迫將 sum[i] 由整數轉換成浮點數資料型別，依序印出全班總分（第 15 行）與印出全班平均（第 16 行）。
14	avg=(float)sum[i]/20;	
15	printf(" 全班總分爲 %d\n",sum[i]);	
16	printf(" 全班平均爲 %f\n",avg);	
17	}	
18	system("pause");	第 18 行：等待按下任意鍵繼續。
19	}	

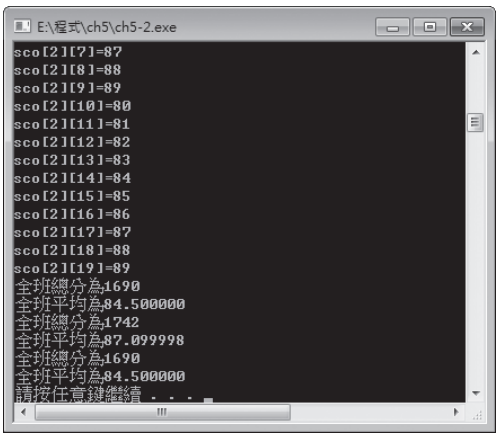
迴圈中 `sum[i]=sum[i]+sco[i][j]` 的運作原理，迴圈中 `i` 與 `j` 值變化與 `sum[i]` 值的對應，如下表。等號右邊的結果 `sum[i]+sco[i][j]` 先運算，結果儲存入等號的左邊 `sum[i]`。

i 值與 j 值	sum 值
i=0,j=0	sum[0]=sum[0]+sco[0][0]=sco[0][0]
i=0,j=1	sum[0]=sum[0]+sco[0][1]=sco[0][0]+sco[0][1]
i=0,j=2	sum[0]=sum[0]+sco[0][2]=sco[0][0]+sco[0][1]+sco[0][2]
i=0,j=3	sum[0]=sum[0]+sco[0][3]=sco[0][0]+sco[0][1]+sco[0][2] +sco[0][3]
i=0,j=4	sum[0]=sum[0]+sco[0][4]=sco[0][0]+sco[0][1]+sco[0][2] +sco[0][3]+ sco[0][4]
…	…
i=0,j=18	sum[0]=sum[0]+sco[0][18]=sco[0][0]+sco[0][1]+sco[0][2] +sco[0][3]+ sco[0][4]+…+sco[0][18]

i 值與 j 值	sum 值
i=0,j=19	sum[0]=sum[0]+sco[0][19]=sco[0][0]+sco[0][1]+sco[0][2] +sco[0][3]+ sco[0][4]+⋯+sco[0][18]+sco[0][19]
i=1,j=0	sum[1]=sum[1]+sco[1][0]=sco[1][0]
i=1,j=1	sum[1]=sum[1]+sco[1][1]=sco[1][0]+sco[1][1]
i=1,j=2	sum[1]=sum[1]+sco[1][2]=sco[1][0]+sco[1][1]+sco[1][2]
i=1,j=3	sum[1]=sum[1]+sco[1][3]=sco[1][0]+sco[1][1]+sco[1][2] +sco[1][3]
i=1,j=4	sum[1]=sum[1]+sco[1][4]=sco[1][0]+sco[1][1]+sco[1][2] +sco[1][3]+ sco[1][4]
…	…
i=1,j=18	sum[1]=sum[1]+sco[1][18]=sco[1][0]+sco[1][1]+sco[1][2] +sco[1][3]+ sco[1][4]+⋯+sco[1][18]
i=1,j=19	sum[1]=sum[1]+sco[1][19]=sco[1][0]+sco[1][1]+sco[1][2] +sco[1][3]+ sco[1][4]+⋯+sco[1][18]+sco[1][19]
i=2,j=0,1,2,⋯,18,19 依此類推。	

程式執行結果

此程式的執行結果如下圖。



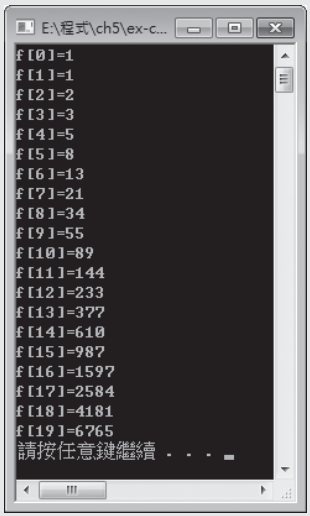
隨堂練習

1. 請列出費氏數列前 20 項

費氏數列是將第 1 項與第 2 項相加等於第 3 項，第 2 項與第 3 項相加等於第 4 項，依此類推，陣列有索引值存取的特性，非常適合解決費氏數列問題。

費氏數列的定義為 f[0]=1，f[1]=1，f[n]=f[n-1]+f[[n-2]，當 n>=2。

詳解請參閱範例檔「ch5/ex-ch5-1.cpp」。



2. 矩陣相加

假設兩矩陣 A 與矩陣 B 相加，結果儲存到矩陣 C，陣列 A、陣列 B 與陣列 C 有以下關係，`C[i][j]=A[i][j]+B[i][j]`。假設陣列 A、陣列 B 與陣列 C 皆為 2 列 3 行的矩陣，陣列初始化如下，寫一程式計算陣列加總結果。

```
int A[2][3]={{1,2,3},{1,2,3}};
int B[2][3]={{4,5,6},{4,5,6}};
```

此程式的執行結果如下圖，詳解請參閱「ch5/ex-ch5-2.cpp」。



Chapter 06

問題解決實例練習

6-1 求陣列所有元素之最大值

想法

使用陣列儲存二十個元素，利用迴圈結構，迴圈初始值 1，測試條件需小於 20，每次加 1，迴圈內變數值由 0、1、2、3… 依序變化到 19，利用此為陣列索引值依序存取每個元素，有一個變數儲存「到目前為止最大值」，若比「到目前為止最大值」來得大，將較大的值儲存入「到目前為止最大值」的變數。如此所有元素都存取過後就可找出最大值

行號	程式碼	說明
1	`#include <stdio.h>`	第 1 行：爲了 printf 函式
2	`#include <stdlib.h>`	第 2 行：爲了 system 函式
3	`int main(){`	第 3~16 行：main 主程式
4	`  int sco[20]={80,81,82,83,84,85,86,87,88,100,80,81,82,83,84,85,86,99,88,89};`	第 4 行：宣告 sco 成績陣列並初始化。
5	`  int i,max=sco[0];`	第 5 行：迴圈變數 i，變數 max 初始化爲 sco[0]，陣列 sco 的第一元素
6	`  for(i=1;i<20;i++){`	第 6~10 行：迴圈變數 i 爲 1，迴圈測試變數 i 是否小於 20，變數 i 遞增 1（第 6 行），迴圈變數值由 1 依序變化到 19 存取陣列 sco 每個元素，若 max 小於 sco[i]（第 7 行），max 等於 sco[i] 第 8 行）。
7	`    if (max < sco[i]){`	
8	`      max=sco[i];`	
9	`    }`	
10	`  }`	
11	`  printf(" 最大值爲 %d\n",max);`	第 11 行：印出最大值。
12	`  system("pause");`	第 12 行：等待按下任意鍵繼續。
13	`}`	

程式執行結果

此範例的執行結果如下圖。



6-2 求 N 階乘

寫一程式輸入 n，計算 n!=1*2*3....*(n-1)*n，n<=10

想法

使用迴圈其中迴圈變數 i，其值由 1 每次遞增 1 到 n，迴圈中計算到目前為止的 1*2*3...*(i-1)*i 結果，最後印出結果。

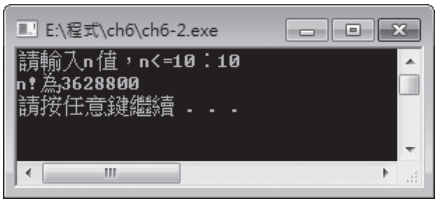
行號	程式碼	說明
1	`#include <stdio.h>`	第 1 行：爲了 printf 函式
2	`#include <stdlib.h>`	第 2 行：爲了 system 函式
3	`int main(){`	第 3~11 行：main 主程式
4	`  int i,n,result=1;`	第 4 行：宣告整數變數 i、n 與 result，並初始化 result 爲 1。
5	`  printf(" 請輸入 n 值，n<=10：");`	第 5 行：螢幕印出「請輸入 n 值，n<=10：」
6	`  scanf("%d",&n);`	第 6 行：等待鍵盤輸入儲存入變數 n
7	`  for(i=1;i<=n;i++){`	第 7~9 行：迴圈變數 i 爲 1，迴圈測試變數 i 是否小於等於 n，變數 i 遞增 1（第 7 行），迴圈變數值由 1 依序變化到 n。result*i 先計算結果儲存入 result。
8	`    result=result*i;`	
9	`  }`	
10	`  printf("n! 爲 %d\n",result);`	第 10 行：印出 n! 的結果。
11	`  system("pause");`	第 11 行：等待按下任意鍵繼續。
12	`}`	

迴圈中 `result=result*i;` 的運作原理，迴圈中 i 值變化與 result 值的對應，如下表。等號右邊的結果 `result=result*i;` 先運算，結果儲存入等號的左邊 `sum`。

i 值	result 值
i=1	result=result*i=1*1=1
i=2	result=result*i=1*2=2=1*2
i=3	result=result*i=2*3=6=1*2*3
i=4	result=result*i=6*4=24=1*2*3*4
…	...
i=n-1	result=result*i=1*2*3*4*...*(n-1)
i=n	result=result*i=1*2*3*4*...*(n-1)*n

程式執行結果

此範例的執行結果如下圖。



6-3　求質數問題

寫一程式輸入 n，計算 n 是否為質數，質數為 1 和 n 之外沒有其他因數。

想法

由鍵盤輸入 n 值，使用重複結構判斷 n 的因數，若只有 1 和 n 沒有其他因數，則回傳 n 為質數，否則 n 為因數。其執行程式碼及解說如下頁。

行號	程式碼	說明
1	`#include <stdio.h>`	第 1 行：爲了 printf 函式
2	`#include <stdlib.h>`	第 2 行：爲了 system 函式
3	`int main(){`	第 3~20 行：main 主程式
4	`  int i=2,n,isPrime=1;`	第 4 行：宣告整數變數 i 並初始化爲 2 與整數變數 n、isPrime，並初始化 isPrime 爲 1。isPrime 爲自訂整數變數，預設爲 1，表示預設爲質數，若 isPrime 爲 0 表示已經找到 n 的因數，所以 n 就不爲質數，isPrime 是我們自訂的整數變數，其意義經由程式賦予。
5	`  printf(" 請輸入 n 值：");`	第 5 行：螢幕印出「請輸入 n 值：」
6	`  scanf("%d",&n);`	第 6 行：等待鍵盤輸入儲存入變數 n
7	`  while(i<n){`	第 7~13 行：迴圈測試變數 i 是否小於等於 n（第 7 行），變數 i 遞增 1（第 12 行），迴圈變數 i 值由 2 依序變化到 n。計算 (n%i) 爲 n 除以 i 的餘數，若 (n%i) 等於 0 表示 i 爲 n 的因數，所以 n 就不是質數，則 isPrime 設定爲 0，遇到 break 跳出 while 迴圈。
8	`    if ((n%i) == 0){`	
9	`      isPrime=0;`	
10	`      break;`	
11	`    }`	
12	`    i=i+1;`	
13	`  }`	
14	`  if (isPrime){`	第 14~18 行：若 isPrime 爲 1，條件測式 isPrime 變數，只要非 0 對 C 語言中代表 true，若 isPrime 爲 0，0 對 C 語言中代表 false。若 isPrime 爲 1 則顯示「輸入的 n」是質數，否則顯示「輸入的 n」不是質數。
15	`    printf("%d 是質數 \n",n);`	
16	`  }else{`	
17	`    printf("%d 不是質數 \n",n);`	
18	`  }`	
19	`system("pause");`	第 19 行：等待按下任意鍵繼續。
20	`}`	

程式執行結果

此範例的執行結果如右圖。



隨堂練習

1. 請設計一個程式，讓使用者可以輸入一個大於零的數值 N，程式可以找到小於 N 的所有質數。

   (解題方法可參考高中資訊科技概論課本第 346-348 頁)

2. 有一陣列之數值為 15, 35, 45, 30, 50, 20, 40, 55, 25。寫一個循序搜尋及二分搜尋法的程式，找出 20 及 60 在陣列中的位置。

   (解題方法可參考高中資訊科技概論課本 第 360-362 頁)

3. 使用循序搜尋找出成績陣列中是否包含成績為 59 分的學生，假設成績陣列為整數陣列，且第一位學生成績為 60 分，第二位學生成績為 90 分，第三位學生成績為 59 分，第四位學生為 98 分，第 5 位學生成績為 50 分，若找到 59 分的學生則輸出「找到 59 分的學生」，否則輸出「找不到 59 分的學生」。

   執行結果如下圖，詳解請參閱範例檔「ch6/ex-ch6-1.cpp」。



Chapter

07

自訂函式

C 語言中允許自訂函式，可以將程式中特定功能獨立出來寫成函式，需要此函式就呼叫使用。呼叫自訂函式只需要注意輸入與輸出，輸入經由函式運算產生想要的輸出，如此一來，若有很多地方需要此函式時就呼叫使用。當需要修改函式功能時，可以只修改函式，所有呼叫此函式的程式都同步修改，若不使用函式方式撰寫程式，程式中有多處需要此功能程式碼就需要寫幾次，當功能修改時，分散於各處的程式都要一個個修改。

自訂函式需要三個部分，函式要**事先宣告**、**之後定義**、最後程式中**呼叫使用**等，**函式宣告**用於讓主程式知道有什麼函式可以使用，函式定義用於確定函式的功能，而使用時以呼叫方式讓程式的執行轉給該函式，以下以自訂加總兩數為範例。

7-1 求加總兩數

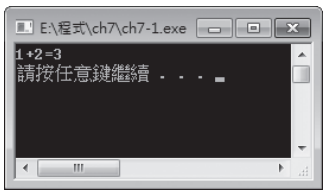
若程式中有很多地方都需要加總兩數的功能，這部分就可以獨立成函式。下表是此範例的主要程式碼與程式說明。

行數	程式	說明
1	`#include <stdio.h>`	第 1 行：內含 printf 函式
2	`#include <stdlib.h>`	第 2 行：內含 system 函式
3	`int sum(int,int);  // 函式宣告`	第 3 行：宣告自訂函式 sum
4	`int main(){`	第 4~9 行：程式的 main 函式
5	`  int i=1,j=2,k;`	第 5 行：宣告整數變數 i 並初始化爲 1，整數變數 j 並初始化爲 2，整數變數 k。
6	`  k=sum(i,j);  // 函式呼叫`	第 6 行：呼叫自訂函式 sum，輸入 i 與 j，回傳 k，變數 i 的值傳入函式 sum 的第一個參數，變數 j 的值傳入函式 sum 的第二個參數，函式 sum 需回傳一個值給 k。
7	`printf("%d+%d=%d\n",i,j,k);`	第 7 行：印出兩數與相加的結果。

行數	程式	說明
8	system("pause");	第 8 行：程式執行到此中斷，按任意鍵後繼續。
9	}	
10	int sum(int x,int y){ // 函式定義	第 10~12 行：自訂函式 sum 的定義。輸入兩個變數 x 與 y，回傳 x 與 y 相加的結果。當呼叫函式 sum 時第一個輸入參數傳入 x，第二個輸入參數傳入 y，程式指令 return 表示將值回傳給呼叫他的程式，本範例中會回傳給呼叫他的 main 函式，並存入整數變數 k。
11	return x+y;	
12	}	

程式執行結果

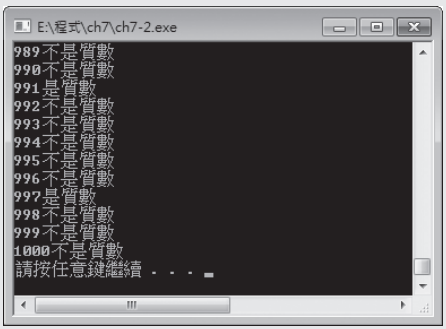
此程式的執行結果如右圖。



隨堂練習

修改第 7-1 節的「求加總兩數」程式為「求相相乘兩數」，自訂函式 mul，輸入兩個整數，輸出相乘的結果。

程式結構可以參考範例程式的「ch7/ex7-1.cpp」，程式執行結果如下圖。



7-2 求質數

此節中，我們要利用函數來求 2 到 1000 的所有質數。求質數問題在第 6-3 節中已有敘述，在此我們將第 6-3 節中判斷質數的程式轉換成函式，自訂函式名稱為 prime，輸入為被判斷的數 n，回傳為 1 表示該數為質數，回傳為 0 表示該數為非質數。

主要程式碼與程式說明如下。

行數	程式	說明
1	`#include <stdio.h>`	第 1 行：內含 printf 函式
2	`#include <stdlib.h>`	第 2 行：內含 system 函式
3	`int prime(int);  // 函式宣告`	第 3 行：宣告自訂函式 prime
4	`int main(){`	第 4~14 行：程式的 main 函式
5	`  int i;`	第 5 行：宣告整數變數 i。
6	`  for(i=2;i<1001;i++){`	第 6~12 行：使用 for 迴圈讓變數 i 由 2、3、4 變化到 1000（第 6 行），呼叫自訂函式 prime 將變數 i 的值當作輸入（第 7 行），用於判斷變數 i 是否為質數，若是函式 prime 回傳 1，進入 if 測試條件測試結果為真，印出「i 是質數」（第 8 行），否則函式 prime 回傳 0，進入 if 測試條件測試結果為假，印出「i 不是質數」（第 9 行）。
7	`    if (prime(i)){  // 函式呼叫`	
8	`      printf("%d 是質數 \n",i);`	
9	`    }else{`	
10	`      printf("%d 不是質數 \n",i);`	
11	`    }`	
12	`  }`	
13	`  system("pause");`	第 13 行：程式執行到此中斷，按任意鍵後繼續。
14	`}`	
15	`int prime(int x){  // 函式定義`	第 15~24 行：自訂函式 prime 的定義。輸入變數 x，回傳判斷 x 是否為質數的結果。當呼叫函式 prime 時輸入參數傳入 x，變數 i 由 2 開始到 x-1（第 16、17 與 21 行），依序將 x 除以變數 i，若可以整除則回傳 0，表示非質數（第 19 行），若都無法整除則回傳 1，表示是質數（第 23 行），程式指令 return 表示將值回傳給呼叫他的程式，本範例中會回傳給呼叫他的 main 函式。
16	`  int i=2;`	
17	`  while(i<x){`	
18	`    if ((x%i) == 0){`	
19	`      return 0;`	

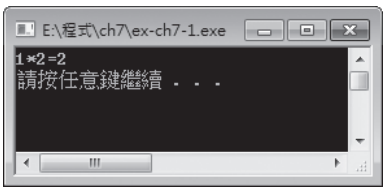
行數	程式	說明

20	}
	i=i+1;
	}
	return 1;
	}	

21		
22		
23		
24		

程式執行結果

此程式的執行結果如下圖。



隨堂練習

請修改第 7-2 節的範例，求印出 1001 到 9999 所有質數，並請簡述使用函式的好處。程式結構可以參考範例檔的「ch7/ex7-2.cpp」程式，程式執行結果如下圖。

