1.3 設計程式的步驟

電腦是協助人類解決問題的工具，因此程式設計者必須先充分瞭解問題，才能撰寫出合乎需要的應用程式。一般設計程式的步驟可分為以下五大階段：

一. 定義問題

對問題一定要有充分的研究與分析，以瞭解該問題是否適合電腦來處理，如此才能明確地定義出要解決問題的方法。描述問題時避免使用含糊不清的語句，以方便程式設計人員對問題的敘述有深入的了解。

二. 問題分析

先認清問題的癥結，對症下藥才能解決問題，對現有的資訊加以整理，再根據輸出格式的需求找出需要輸入哪些資料，並明確定出各種輸出入的限制。下圖是設計一個 C 語言程式步驟的流程圖。

三. 設計演算法

根據問題的輸出入需求，詳細寫下解決問題的步驟。在這個階段，不需要考慮使用何種程式語言來撰寫。若是簡單的問題繪製流程圖即可，至於較複雜的問題則採用虛擬碼方式來描述。若能想出其他演算法將它寫下來，試著比較選出一個較佳的演算法。一個好的演算法應具備下列五大要件：

① 有限性：演算法必須在有限的步驟內解決問題。

② 明確性：演算法中的每一個步驟都必須很清楚地表達出來。

③ 有效性：必須在有限的時間內完成。

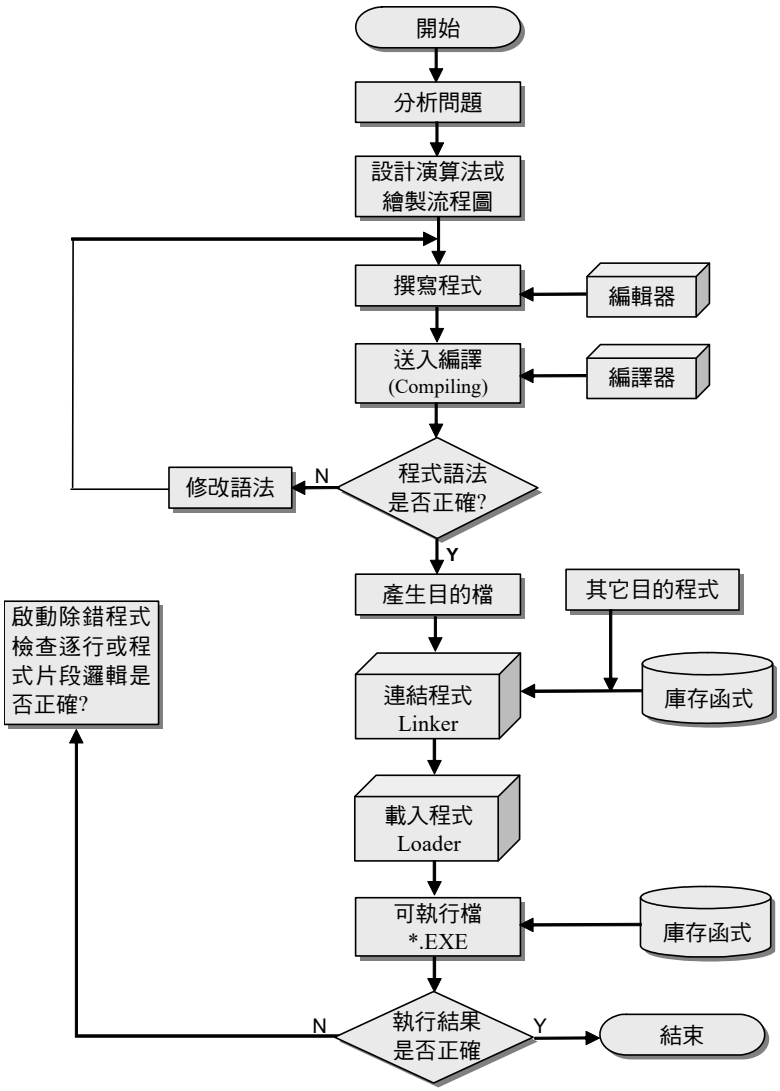
④ 輸入資料：包含零個或一個以上的輸入資料。

⑤ 輸出資料：至少產生一個輸出結果。

四. 撰寫程式

根據選定的演算法，選擇合適的程式語言，再依照演算法步驟來撰寫程式，同時儘量以模組與物件方式來編寫程式；在寫程式的過程中，不要忘了在程式中加上註解。因為一個複雜程式，若沒有註解不但別人看不懂，日子久了就算是自己，也可能會忘了當初為何如此構思。程式的註解必須在編寫程式時

一併寫入、不宜在寫完後許久才補上。若遇到過於複雜的演算法，需詳細說明時，就必須另外編寫程式的說明手冊。



五. 程式的測試與維護

此步驟包含程式驗證、測試、除錯與維護四大部份。在執行程式時應試著去驗證是否正確？每一個條件都要測試，而且要使每一條件成立與不成立都要執行，以驗證程式是否正確無誤。若程式無法正確的執行，表示程式可能有語法(Syntax)或邏輯上錯誤，必須找出錯誤的地方加以修改，也有可能是演算法錯誤，這時可能要回到階段二、三、四做重複的處理，直到程式驗證無誤為止。當程式正確無誤後，便需要對程式寫文件說明，以供日後方便維護和閱讀。

編輯器(Editor)是用來編輯原始程式檔案的工具程式，一般編輯程式都由提供 C 語言的廠商提供。當使用 C 語言編輯完畢的程式存檔後即稱為原始程式檔(Source Program)。由於 C 語言所編寫出的程式是屬於高階語言(High-Level Language)，因此必須透過「編譯器」(Compiler)，將它變成「目的碼」(Object Code)。在編譯的過程中會將原始程式做字彙分析(Lexical Analysis)、語法分析(Parsing)、語意分析(Semantic Analysis)、產生中間碼(Intermediate Code Generation)、程式碼最佳化(Optimization)、產生組合語言程式碼(Code Generation) 六大步驟。前三步驟若有發生錯誤，編譯器會停止編譯，此時必須將發生錯誤的地方更正，再重新編譯直到無錯誤為止，此時編譯器會進行第 4~6 步驟，將程式中所有敘述透過「產生中間碼」轉成更低階的「組合語言」。至於字彙分析是將程式中所有敘述拆成有意義的字串，我們將這些獨立字串稱為「Token」。譬如：

```
perimeter = 2 * PI * radius;
```

分成 "perimeter"、"="、"2"、"*"、"PI"、"*"、"radius"、";" 共 8 個 Tokens。語法分析就是檢查這些 Token 是否符合 C 語言的文法規則，譬如：是否漏打符號、括號、或括號不成對等錯誤發生。至於語意分析是檢查是否有拼錯字、變數是否未宣告等錯誤發生。

所謂「函式庫」或稱「庫存函式」是指一些事先已經編譯好而且具有能執行某特定功能的集合。一般程式語言都將這些功能直接建立在程式語言定義上，變成敘述來使用。但 C 語言卻將這些經過編譯過的特定功能採庫存函式處理，這些庫存函式依性質放在不同的庫存函式檔，只要經由「連結程式」(Linker)

連結，便可自動將程式中使用到的庫存函式連結到可執行檔中，至於沒用到的庫存函式是不會連結到程式中。連結程式是將剛編譯過的程式、事先已經編譯過的目的檔以及程式中將用到的庫存函式連結起來成為一個可執行檔，再透「載入程式」(Loader)載入到記憶體中去執行。

除錯程式(Debugger)是用來協助偵測程式發生錯誤地方的程式，程式經過編譯時若語法沒有發生錯誤，但是程式執行的結果卻發生不符合預期的結果，就表示程式邏輯上有錯誤，此時便需透過除錯程式來逐行偵測或執行一個程式片段後，檢查是否符合預期結果，便可找出發生錯誤的地方。

一. 一個設計良好的程式所具備的條件

1. 程式具可讀性，且程式中重要部分都有詳細的註解說明。
2. 程式的執行結果符合預期且正確。
3. 程式具模組化或結構化，以利程式修改或更新時更便捷。
4. 程式的架構有完整的說明。
5. 程式的執行效率和相容性要高，不會因更換設備而造成錯誤或執行速度變慢。

二. 選擇程式語言時應考慮的因素

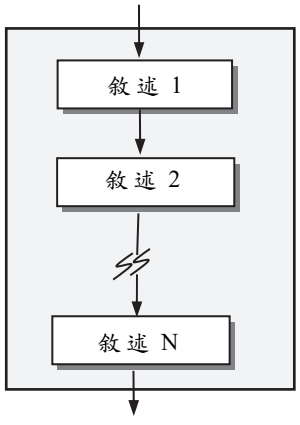
1. 根據該程式是應用在科學、商業、Web 應用程式、智慧型行動裝置應用程式、物聯網、數據分析與人工智慧應用…等環境，來選擇合適且自己熟悉的程式語言。
2. 該程式語言最好和其他高階語言的語法相容性高，讓日後更換程式語言時更加容易。
3. 該程式語言提供的開發環境(包括編輯器、編譯器、除錯程式)是否親和力高，以及功能強大。
4. 該程式語言是否提供完整的參考手冊。
5. 該程式語言是否普及，價位是否合理，廠商研發能力是否堅強。
6. 未來更換機種，是否會影響該程式語言所設計出來的程式。

CHAPTER

5 選擇結構

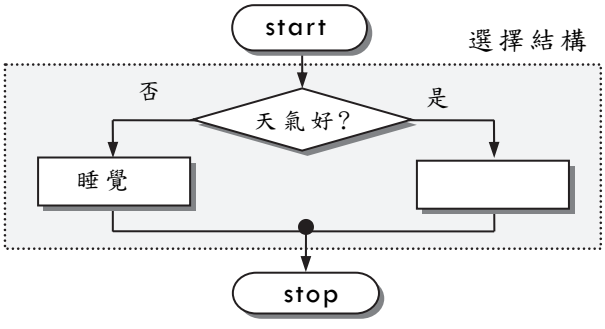
5.1 選擇結構簡介

在前一章所撰寫的程式其流程都如右圖是一行接一行由上而下逐行執行，即使再執行一次其流程仍不改變，我們將此種程式架構稱為「循序結構」。



循序結構

在撰寫程式的過程中，可能因需要而改變不同的執行流程，此時就必須使用「選擇結構」來達成。舉一個日常生活的例子：如果今天天氣好就去 "血拼" (Shopping)，天氣不好就待在家裡睡大頭覺。其流程圖表示如下：



由上可知，天氣好不好是一個條件，因條件不同而選擇不同的流程，不管條件與否最後都會回到同一終點，繼續往下執行，這就是「選擇結構」。在程

程式碼 FileName : elseif.c

```
01 #include <stdio.h>
02 #include <stdlib.h>
03
04 int main(int argc, char *argv[]) {
05     int score;
06     printf("請輸入您的分數：");
07     scanf("%d", &score);
08     if(score>=80 && score<=100 )
09         printf("\n Excellent! 等級為 A .");
10     else if(score>=70 && score<80 )
11         printf("\n Good! 等級為 B.");
12     else if(score>=60 && score<70 )
13         printf("\n Fair! 等級為 C.");
14     else if(score>=0 && score<60 )
15         printf("\n Bad! 等級為 D.");
16     else
17         printf("\n 輸入成績錯誤!  分數限 0~100 ... ");
18
19     printf("\n\n");
20     system("PAUSE");
21     return 0;
22 }
```

說明

1. 第 5~7 行：宣告 score 整數變數用來存放使用者所輸入的分數。
2. 第 8~9 行：若 80≤score≤100 成立，則顯示 "Excellent! 等級為 A."。
3. 第 10~11 行：若 70≤score<80 成立，則顯示 "Good! 等級為 B."。
4. 第 12~13 行：若 60≤score<70 成立，則顯示 "Fair! 等級為 C."。
5. 第 14~15 行：若 0≤score<60 成立，則顯示 "Bad! 等級為 D."。
6. 第 16~17 行：若 score 沒有介於 0~100 之間，則顯示 "輸入成績有誤! 分數限 0~100 …"。

式語言中的條件就是透過運算式來設定，C 語言中能產生條件的運算式有「關係運算式」和「邏輯運算式」，C 語言將這些運算式的結果進行判斷，若結果不為零，視為真；若結果為零值，視為假。本章在 5.2 節將介紹關係運算子，5.3 節介紹邏輯運算子，然後再以簡單的範例來說明不同的選擇結構。

5.2 關係運算子

「關係運算子」(Relational Operator)是用來比較關係運算子左右兩邊的運算式，並將比較的結果傳回，若比較的結果為真，傳回值為 1；比較結果不成立時傳回值為 0(代表假)。在 C 語言中的關係運算子是透過大於、小於或等於運算子組合成下表中的六種狀態，供您在設計程式時使用。

運算子	說明	使用例	結果
== (相等)	判斷此運算子左右兩邊運算式的值是否相等。	15 == 15	1(真)
		15 == 25	0(假)
		3+2 == 1+4	1(真)
!= (不相等)	判斷此運算子左右兩邊運算式的值是否不相等。	17 != 18	1(真)
		56 != 56	0(假)
		12*3 != 3*12	0(假)
< (小於)	判斷此運算子左邊運算式的值是否小於右邊運算式的值。	10 < 20	1(真)
		40 < 30	0(假)
		2 < 10-7	1(真)
> (大於)	判斷此運算子左邊運算式的值是否大於右邊運算式的值。	20 > 10	1(真)
		20 > 30	0(假)
		12*3 > 12*2	1(真)
<= (小於等於)	判斷此運算子左邊運算式的值是否小於等於右邊運算式的值。	10 <= 20	1(真)
		10 <= 10	1(真)
		10+3 <= 12	0(假)
>= (大於等於)	判斷此運算子左邊運算式的值是否大於等於右邊運算式的值。	10 >= 20	0(假)
		10 >= 10	1(真)
		12*3 >= 35	1(真)

5.3 邏輯運算子

當一個條件中有兩個以上關係運算式需要一起做判斷時，就必須使用到邏輯運算子來連接，也就是說邏輯運算子是用來判斷兩個以上關係運算式之間的關係，這在程式設計的流程中是很常用的，至於邏輯運算式的表示語法如下：

語法

```
結果 = [ 運算式 1 ]  邏輯運算子  運算式 2 ;
```

下表列出常用的邏輯運算子說明：

<table>
<tr><th>邏輯運算子</th><th>說明</th><th>真值表</th></tr>
<tr><td>&&
(AND,且)</td><td>此運算子左右兩邊的運算式結果皆不為零值，結果為1(真)；否則為零值(假)。</td><td>
<table>
<tr><th>運算式 1</th><th>運算式 2</th><th>結果</th></tr>
<tr><td>非 0</td><td>非 0</td><td>1</td></tr>
<tr><td>非 0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>非 0</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
</table>
</td></tr>
<tr><td>||
(OR,或)</td><td>此運算子左右兩邊的運算式結果只要其中有一個不為零值，結果就是 1；兩個都為零結果才是零值。</td><td>
<table>
<tr><th>運算式 1</th><th>運算式 2</th><th>結果</th></tr>
<tr><td>非 0</td><td>非 0</td><td>1</td></tr>
<tr><td>非 0</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>非 0</td><td>1</td></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
</table>
</td></tr>
<tr><td>!
(NOT)</td><td>此運算子是單一的運算，主要是將敘述結果相反，即1⇨0，0⇨1。</td><td>
<table>
<tr><th>運算式</th><th>結果</th></tr>
<tr><td>非 0</td><td>0</td></tr>
<tr><td>0</td><td>1</td></tr>
</table>
</td></tr>
</table>

[例 1]　$10 \le x < 20$，　　條件式為：(x >= 10 && x <20)

[例 2]　$x \le 10$ 或 $x > 20$　條件式為：(x <= 10 || x >20)

5.4 選擇敘述

5.4.1 if-else 敘述

程式中的選擇結構有如口語中的「如果.....就.....否則....」，在 C 語言中是使用 if-else 敘述來達成，如下面語法，若 <條件式> 成立時，則執行接在 if 後面的 [敘述區段 1]，否則(條件不成立)執行接在 else 後面的 [敘述區段 2]。

語法1 單一敘述

```
if (條件式)
    敘述區段 1;
[else
    敘述區段 2; ]
```

語法2 多行敘述

```
if (條件式) {
   ⋮
   敘述區段 1
}
[else {
   ⋮
   敘述區段 2
} ]
```

說明

1. 如果 if-else 的選擇敘述所要執行的敘述只有一行時，可以使用語法 1，省略左大括號{ 及 右大括號}。
2. 若敘述超過一行以上時，就必須使用語法 2。
3. 上面語法中的 [...] 中括號內的敘述是當不滿足條件且不執行任何敘述時，此部份可省略。
4. 譬如:由分數 score 來判斷是否 Pass(及格)或 Down(不及格)？若 score≥60 顯示 "Pass"；如果 score<60 顯示 "Down"，有下列兩種撰寫方式：

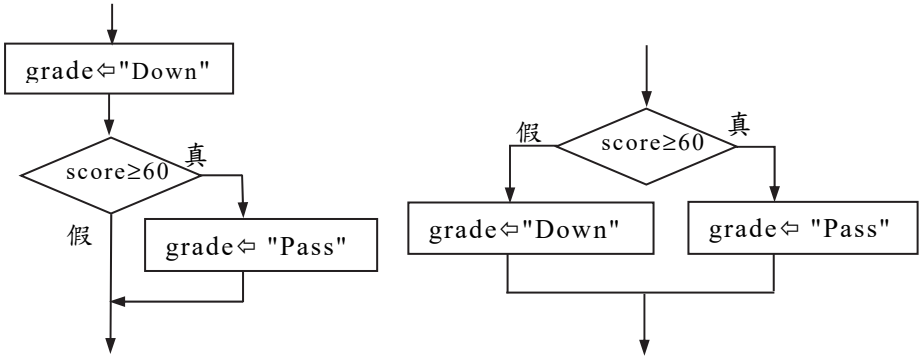
使用單一選擇(省略 else 敘述)

```
grade = "Down" ;
if (score >= 60)
   grade = "Pass" ;
```

使用 if...else 敘述：

```
if (score >= 60)
     grade = "Pass" ;
else
     grade = "Down";
```

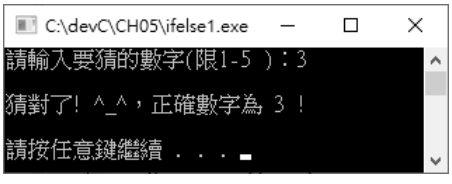
流程圖



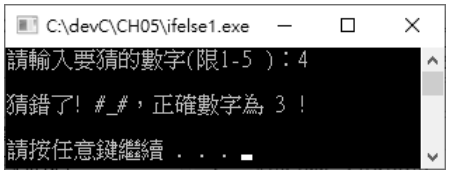
範例：ifelse1.c

製作簡易的猜數字遊戲。程式執行時電腦自動產生 1~5 之間的亂數，接著等待您由鍵盤鍵入您猜的數字。若猜對，則顯示「猜對了」相關訊息；若猜錯，則顯示「猜錯了」相關訊息。

執行結果



猜對畫面



猜錯畫面

問題分析

1. 若在程式中直接設定被猜數字，導致每次執行所猜的數字都一樣，此種作法程式不具彈性。C 語言提供 srand()函式，配合 rand()函式可產生介於 0~32,767 間的亂數。若要使每次執行時，所產生被猜的數字會不一樣，可用時間當做亂數產生器的種子。寫法如下：

```
srand((unsigned)time(NULL));
```

上面得到的時間亂數當種子所產生 0~32,767 的整數，若要存入 guess 整數變數，寫法為：

```
guess = rand();
```

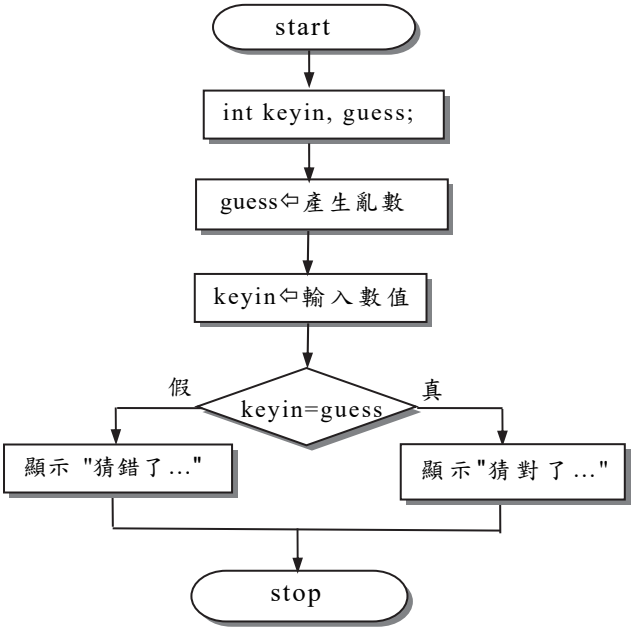
2. 將產生介於 0~32,767 的亂數除以 5 取其餘數，結果所產生的餘數會介於 0~4 之間。寫法：rand() % 5

3. 產生介於 1~5 之間的整數亂數，並指定給 guess 整數變數，寫法如下：

   guess = rand() % 5 + 1;

   由於 srand()和 rand()兩個函式都宣告在 stdlib.h 標頭檔中，time()函式宣告在 time.h 標頭檔，因此，在程式開頭要記得含入 #include <stdlib.h> 和 include <time.h> 兩個敘述。

流程圖



程式碼 FileName : ifelse1.c

```
01 #include <stdio.h>
02 #include <stdlib.h>
03 #include <time.h>
04
05 int main(int argc, char *argv[]) {
06     int keyin, guess;
07     srand((unsigned)time(NULL));
08     guess=rand()%5+1;
09     printf("請輸入要猜的數字(限 1-5 )：");
```

```
10      scanf("%d", &keyin);
11      if(keyin==guess)
12          printf("\n 猜對了! ^_^，正確數字為 %d !\n", guess);
13      else
14          printf("\n 猜錯了! #_#，正確數字為 %d !\n", guess);
15
16      printf("\n");
17      system("PAUSE");
18      return 0;
19  }
```

說明

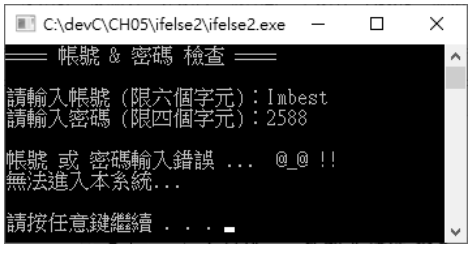
1. 第 6 行：宣告 keyin 及 guess 整數變數。keyin 表示由鍵盤所輸入的數字，guess 用來存放程式所產生要猜的整數亂數。
2. 第 7,8 行：產生介於 1~5 之間的亂數並指定給 guess 整數變數。
3. 第 10 行：輸入一個整數，並指定給 keyin 整數變數。
4. 第 11~14 行：判斷 keyin 是否等於 guess，若相等則執行第 12 行顯示猜對相關訊息；否則執行第 14 行顯示猜錯相關訊息。

範例：ifelse2.c

試寫一個帳號與密碼檢查程式。若輸入正確的帳號 "mebest" 及密碼 "1688"，出現左下圖畫面；若帳號或密碼輸入錯誤出現右下圖畫面。

執行結果

帳號、密碼正確



帳號或密碼錯誤

問題分析

1. 本例所輸入的帳號與密碼都需要連續的字元(字串)，因此必須宣告字元陣列用來存放連續字元。字元陣列的宣告方式如下：(關於字元陣列與字串的宣告方式請參考第十二章)

```
char usr_id[7] ;    /* 宣告一個字元陣列 usr_id 用來存放 6 個字元的帳號 */
char pwd[5] ;       /* 宣告一個字元陣列 pwd 用來存放 4 個字元的密碼 */
```

   宣告字元陣列 usr_id(字串)可存放 7 個字元，但是字串本身後面必須放置 '\0' 空字元當結束字元才能成為字串，因此字元陣列 usr_id 實際所能容納的字元個數為 6 個；pwd[5] 字元陣列可容納 4 個字元。

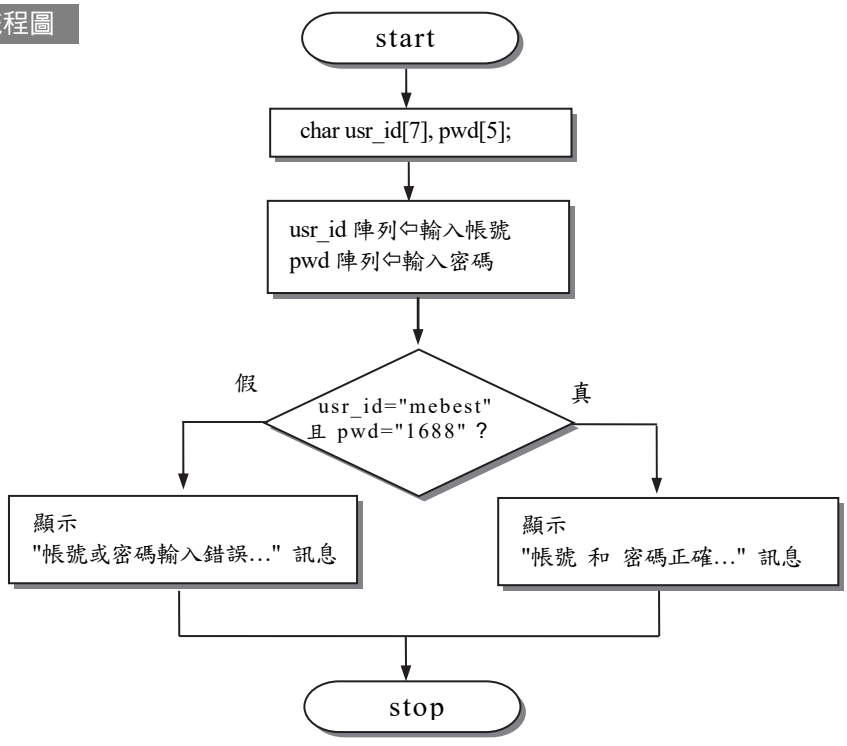
2. 由於程式中需要使用 strcmp()函式來比較兩個字串是否相等？由於此函式原型宣告在 string.h，因此必須在程式的最開頭撰寫 #include <string.h> 敘述。在呼叫 strcmp()函式時，會將比較的結果以零、負值或正值傳回，此處以 s1="Tax " ; s2="Tax" ; s3="Tom" ; s4="Tank" 為例：

   ① strcmp(s1, s2);　　/*由於 s1 字串與 s2 字串相等，傳回值為 0*/
   ② strcmp(s1, s3);　　/*由於 s1 字串小於 s3 字串，傳回值小於 0*/
   ③ strcmp(s1, s4);　　/*由於 s1 字串大於 s4 字串，傳回值大於 0*/

3. 輸入帳號時，必須使用 scanf("%s", usr_id);，將由鍵盤鍵入的字串存到 usr_id 字元陣列中。其中使用%s 表示輸入的資料是字串，usr_id 是陣列名稱本身就代表陣列在記憶體中的起始位址，所以在 scanf()函式內 usr_id 前面不用再加上&位址符號，至於一般變數名稱欲代表該記憶體位址時，必須在變數名稱前面加上&位址符號來代表該變數的記憶體位址。同樣方式應用到輸入密碼上面 scanf("%s", pwd);。

【註】字串是以 ASCII 內碼來比較大小，a 的 ASCII 碼是 97，比 b 的 ASCII 碼 98 小，所以 b 比 a 大。字串中若第一個字元的 ASCII 碼相同，接著比第二個字元的 ASCII 碼大小，依此類推。

流程圖



程式碼 FileName : ifelse2.c

```
01 #include <stdio.h>
02 #include <stdlib.h>
03 #include <string.h>
04
05 int main(int argc, char *argv[]) {
06     char usr_id[7];     /* 宣告一個字元陣列 usr_id 用來存放 6 個字元的帳號 */
07     char pwd[5];        /* 宣告一個字元陣列 pwd 用來存放 4 個字元的密碼 */
08     printf("==== 帳號 & 密碼 檢查 ====\n\n");
09     printf("請輸入帳號 (限六個字元) : ");
10     scanf("%s", usr_id);
11     printf("請輸入密碼 (限四個字元) : ");
12     scanf("%s", pwd);
13     printf("\n");
14     if(strcmp(usr_id, "mebest")==0 && strcmp(pwd,"1688")==0) {
15         printf("帳號 和 密碼正確 ...  ^_^ !!\n");
16         printf("歡迎進入本系統...\n\n");
```

```
17      }
18      else {
19          printf("帳號 或 密碼輸入錯誤 ...  @_@ !!\n");
20          printf("無法進入本系統...\n\n");
21      }
22
23      system("PAUSE");
24      return 0;
25  }
```

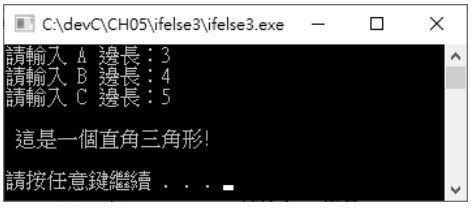
說明

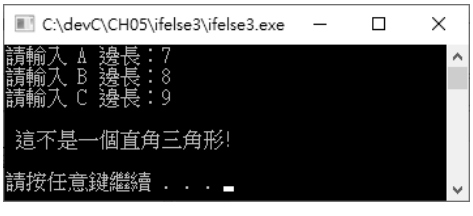
1. 第 6~7 行:宣告可存放 6 個字元的 usr_id 及 4 個字元的 pwd 字元陣列。usr_id 字元陣列用來存放帳號，pwd 字元陣列用來存放密碼。
2. 第 9,11 行：提示輸入帳號和密碼分別存入 usr_id 和 pwd 字元陣列中。若您輸入的字元超過設定的長度，會發生不可預期的錯誤情形。
3. 第 14 行：帳號和密碼必須都正確，條件才成立，否則條件不成立。因此兩個條件式中間必須使用 &&(AND、且) 邏輯運算子來連接。
4. 第 14~21 行：strcmp()函式可比較兩個字串是否相等，若兩個字串相等則會傳回 0。在第 14 行判斷輸入的 usr_id(帳號)是否等於 "mebest" 字串，且 pwd(密碼)是否等於 "1688" 字串，若成立會執行第 15~16 行印出 "帳號和密碼正確..." 訊息；否則執行第 19~20 行印出 "帳號或密碼錯誤..." 訊息。

範例：ifelse3.c

輸入一個三角形的三個邊長 A、B、C，若任兩邊長的平方和等於第三邊長的平方即是直角三角形，否則不是直角三角形。

執行結果





問題分析

假設 A、B、C 為三角形三邊的邊長，

$A^2 + B^2 = C^2$ 或

$A^2 + C^2 = B^2$ 或

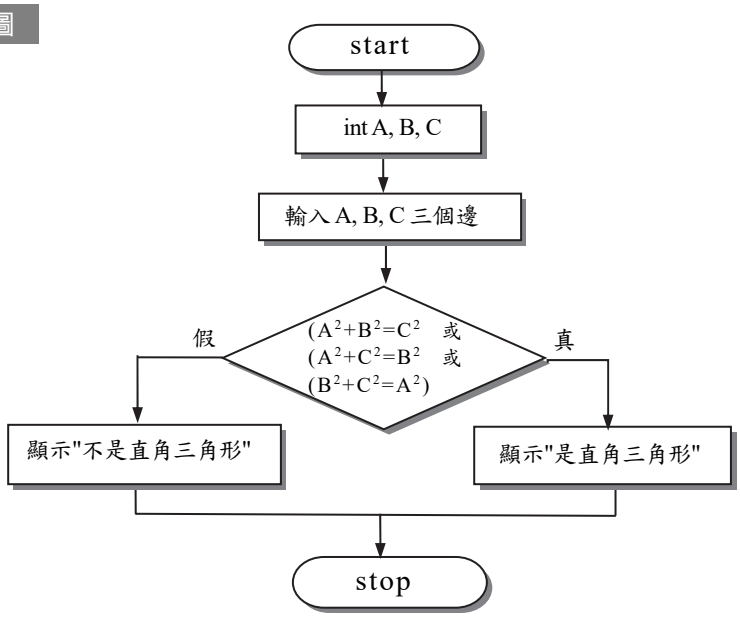
$B^2 + C^2 = A^2$

只要上面三個條件式其中之一成立即為真時，三個條件式中間必須使用 邏輯 OR 運算子(||) 來連接。寫法如下：

```
if((pow(A,2)+pow(B,2)) == pow(C,2) || (pow(A,2)+pow(C,2)) == pow(B,2)
                || (pow(B,2)+pow(C,2) == pow(A,2))) {
        ⋮
}
else {
        ⋮
}
```

上面條件式中，pow(a,b)函式可傳回 a^b 值(a 的 b 次方)，此函式宣告在 math.h 標頭檔內，因此在程式最開頭要插入 #include <math.h> 標頭檔。

流程圖



程式碼 FileName : ifelse3.c

```
01 #include <stdio.h>
02 #include <stdlib.h>
03 #include <math.h>
04
05 int main(int argc, char *argv[]) {
06     float a, b, c;
07     printf("請輸入 A 邊長：");
08     scanf("%f", &a);
09     printf("請輸入 B 邊長：");
10     scanf("%f", &b);
11     printf("請輸入 C 邊長：");
12     scanf("%f", &c);
13     if((pow(a,2)+pow(b,2))==pow(c,2)||(pow(a,2)+pow(c,2))==pow(b,2)
                ||(pow(b,2)+pow(c,2))==pow(a,2))
14        printf("\n 這是一個直角三角形!" );
15     else
16        printf("\n 這不是一個直角三角形!");
17     printf("\n\n");
18     system("PAUSE");
19     return 0;
20 }
```

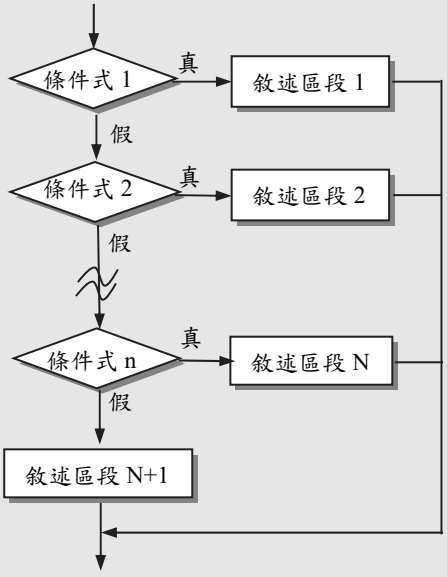
說明

1. 第 6~12 行：宣告 A, B, C 整數變數為三角形的三邊，由鍵盤輸入三個整數依序存入 A, B, C 三個整數變數中。
2. 第 13~16 行：判斷是否為直角三角形？只要 $A^2+B^2= C^2$、$A^2+C^2= B^2$ 或 $B^2+C^2= A^2$ 其中之一成立，則執行第 14 行，否則執行第 16 行。
3. pow(a, b)函式可用來取得 a^b 值。要注意 a 與 b 引數必須宣告為 float 或 double 資料型別。

5.4.2 if-else if 敘述

如果判斷資料的條件超過兩組以上可供選擇，就可以加上 else if 來做其他條件判斷，除了在第一個條件使用 if，其他條件都使用 else if 來描述，最後再以 else 來處理都不滿足以上條件(即剩下的可能性)。其語法如下：

```
if (條件式 1) {
      敘述區段 1
}
else if (條件式 2) {
      敘述區段 2
}
      ⋮
else if (條件式 N) {
      敘述區段 N
}
else {
      敘述區段 N+1
}
```



下面範例，依使用者輸入的分數來做不同程度的分類，先判斷分數是否為 80~100，再判斷是否為 70~79，再判斷是否為 60~69…，以此類推。

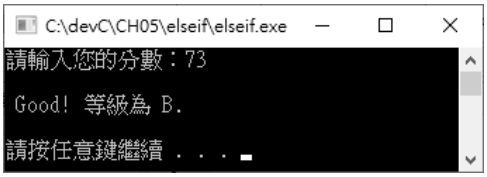
由於程式執行是由上而下，當符合了某一個條件，則執行緊接在其後的敘述區段，執行完畢馬上離開選擇結構敘述，不會再執行其他的條件判斷。因此我們在撰寫條件式時，必須依條件的大小順序撰寫，使得輸入的分數只允許落在一種可能的條件上，而只執行該條件相對應的敘述。

範例 ：elseif.c

輸入分數，依不同的分數給予不同的等級。條件與對應的等級如下：

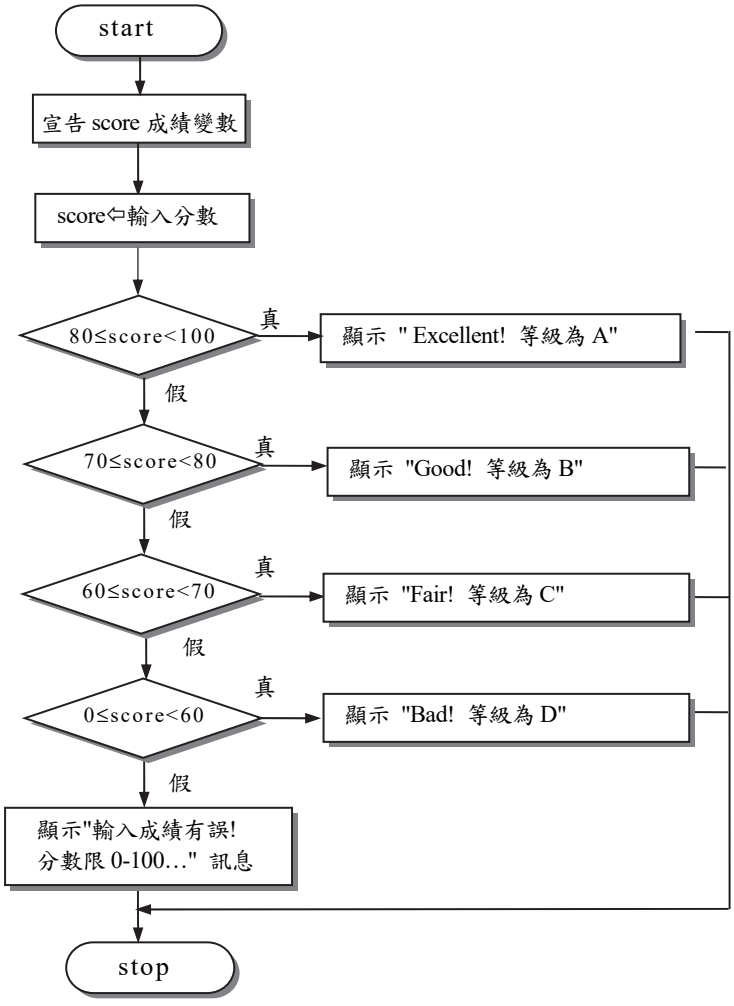
① 80~100：Excellent, 等級為 A
② 70~79：Good, 等級為 B
③ 60~69：Fair, 等級為 C
④ 0~59：Bad, 等級為 D
⑤ 其他：輸入成績有誤! 分數限 0~100 …

執行結果


C:\devC\CH05\elseif\elseif.exe
請輸入您的分數：73
Good! 等級為 B.
請按任意鍵繼續 . . .

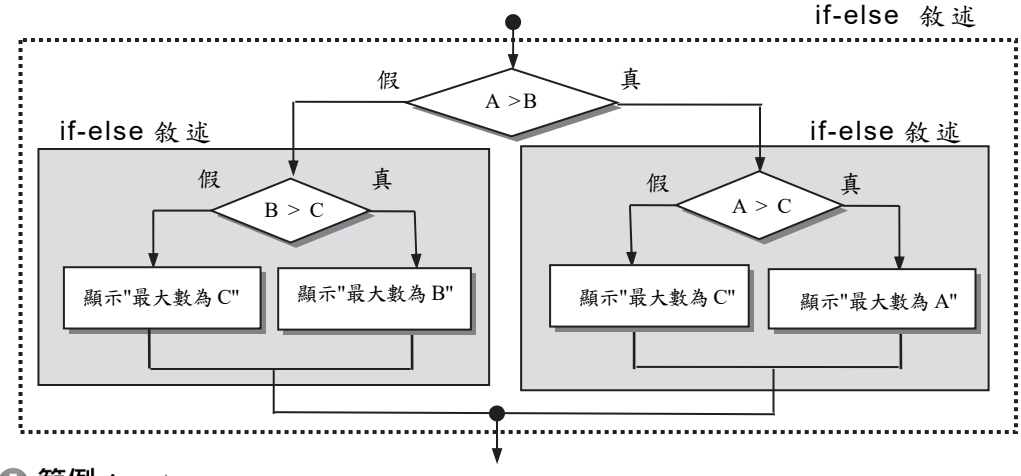

C:\devC\CH05\elseif\elseif.exe
請輸入您的分數：120
輸入成績錯誤! 分數限 0~100 ...
請按任意鍵繼續 . . .

流程圖


start
宣告 score 成績變數
score⇦輸入分數
80≤score<100
真
顯示 " Excellent! 等級為 A"
假
70≤score<80
真
顯示 "Good! 等級為 B"
假
60≤score<70
真
顯示 "Fair! 等級為 C"
假
0≤score<60
真
顯示 "Bad! 等級為 D"
假
顯示"輸入成績有誤!
分數限 0-100..." 訊息
stop

5.4.3 巢狀選擇結構

如果在 if 的敘述區段或 else 敘述區段內還有另一組的 if-else 敘述，就構成一個「巢狀選擇結構」。譬如下面流程圖是找出三數中的最大值就是使用巢狀選擇結構所構成。



範例：getmax.c

試寫一個找出三數中最大值的程式。首先由鍵盤連續輸入三個整數，然後使用上面流程圖的巢狀選擇結構來找出三個整數中的最大數。

執行結果

程式碼 FileName : getmax.c

```
01 #include <stdio.h>
02 #include <stdlib.h>
03
04 int main(int argc, char *argv[]) {
05     int a, b, c, max;
06     printf("請連續輸入三個不同的整數（資料間使用「,」逗號隔開): ");
```

```
07     scanf("%d,%d,%d", &a, &b, &c);
08     if(a>b) {              /* 判斷 a 是否大於 b*/
09         if(a>c)            /* 判斷 a 是否大於 c*/
10             max = a;
11         else
12             max = c;
13     }
14     else {
15         if(b>c)            /* 判斷 b 是否大於 c*/
16             max = b;
17         else
18             max = c;
19     }
20     printf("\n比較結果 : %d, %d, %d 三數中最大數為 %d \n\n", a,b,c,max);
21     system("PAUSE");
22     return 0;
23 }
```

說明

1. 第 5~7 行：連續輸入三個整數，每個輸入值之間使用「,」號隔開，然後依序逐一指定給 a, b, c 三個整數變數。
2. 第 8~13 行：若 a>b 且 a>c，則執行第 10 行，將 a 值置入 max 變數中；若 a>b 且 a<c，則執行第 12 行，將 b 值置入 max 變數中。兩者執行完畢跳到第 20 行。
3. 第 14~19 行：若 a<b 且 b>c，執行第 16 行，將 b 值置入 max 變數中；若 a<b 且 b<c，則執行第 18 行，將 c 值置入 max 變數中。兩者執行完畢跳到第 20 行。

5.4.4 條件運算子

C 語言提供「條件運算子」或稱「三元運算子」，讓您能在程式中，經由條件式運算的結果(真或假)來決定傳回哪個指定值，該敘述只寫一行即可，不像 if-else 要寫成多行。其語法如下：

語法

```
變數 = 條件式 ? 運算式1 : 運算式2 ;
```

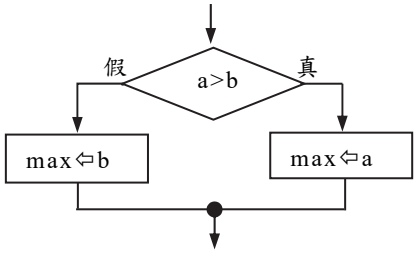
說明

上述語法若 <條件式> 為真(不為零)時，會將 <運算式 1> 的結果指定給等號(=)左邊的變數；若 <條件式> 結果為假(零)時，則會將 <運算式 2> 的結果指定給等號左邊的變數，其傳回值可以是數值、字元、字串等資料型別，等號左邊的變數必須和傳回值同一資料型別。要注意條件運算子 ?... : … 的 <條件式>、<運算式 1>、<運算式 2> 三者都不可以省略。

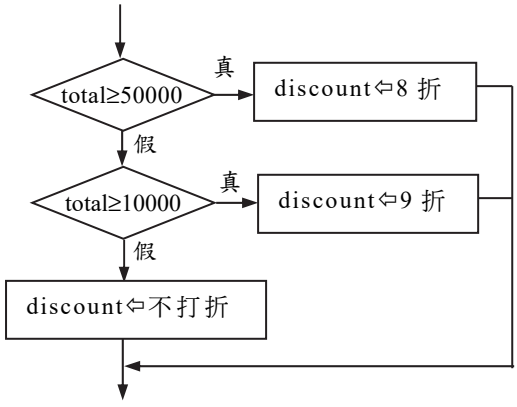
【例 1】求 a、b 的最大值

```
max = a > b ? a : b ;
```

用來取得 a, b 兩數的最大數。若 a 大於 b，會將 a 指定給 max，否則將 b 指定給 max。



【例 2】**? ... : ...** 條件運算子也可以是巢狀的。巢狀條件運算子運算方式是由左而右運算。譬如：金額(total)超過 5 萬元打八折，超過 1 萬元打九折，低於一萬元不打折，就需要使用到巢狀的條件運算子得到折扣(discount)，其寫法如下：




```
discount = ( total >= 50000 ? 0.8 : (total >= 10000 ? 0.9 : 1 ) );
```

範例：ternary.c

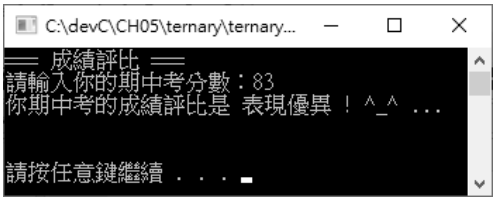
請使用巢狀條件運算子，先由鍵盤鍵入你的期中考分數，程式會自動告知你期中考的成績評比：

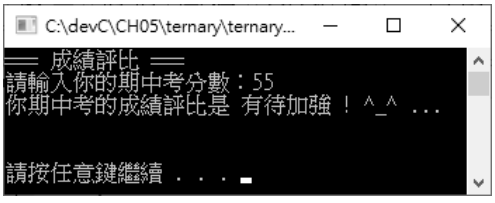
① 期中考分數大於等於 80：表現優異

② 期中考分數大於等於 60 且小於 80：差強人意

③ 期中考分數小於 60：有待加強"

執行結果





程式碼 FileName：ternary.c

```
01 #include <stdio.h>
02 #include <stdlib.h>
03
04 int main(int argc, char *argv[]) {
05     int score;
06     printf("=== 成績評比 ===\n");
07     printf("請輸入你的期中考分數：");
08     scanf("%d", &score);
09     printf("你期中考的成績評比是 %s ！ ^_^ ... \n", score>=80? "表現優異":
                    (60<=score && score<80 ? "差強人意" : "有待加強"));
10     printf("\n\n");
11     system("PAUSE");
12     return 0;
13 }
```

說明

1. 第 5~8 行：宣告 score 整數變數，使用者所輸入的分數會指定給 score。
2. 第 9 行：若 score≥80 會印出「表現優異」；若 60≤score<80 會印出「差強人意」；若都不滿足上述條件(即 score<60)會印出「有待加強」。