GLOBAL EDITION

Computer Programming and Practice (Ⅰ)
計算機程式設計與實習(一)
- Ch02 -

110年上學期
國立臺南大學 電機工程系
梁家銘

2 C程式設計入門

2.1 簡介

- C語言方便讓程式設計者以結構化且有條理的方法來設計電腦程式。
- 以下將簡介C程式設計，透過幾個例子來說明C語言的重要特性。

2.2 一個簡單的C程式：列印一行文字

- 我們從一個簡單的C程式開始，第一個例子：
  - 列印一行文字 "Welcome to C!"

圖2.1 第一個C程式

1. 註解

- 此程式非常簡單，其包含了C程式幾個重要特性。

```
// Fig. 2.1: fig02_01.c
// A first program in C
```

- 以 // 開頭的敘述，表示是**單行註解 (comment)**
  - 增加程式可讀性
- 也可使用 /* ... */ 進行**多行註解(multi-line comments)**
  - 從 /* 開始並以 */ 結尾之間的所有內容都被視為註解，編譯器不會進行編譯，

```
/*.....多行註解.....
.....*/
```

2. #include 前置處理器指令

```
#include <stdio.h>          //引入常用函式庫
```

- 是**C程式前置處理器 (C preprocessor) 指令**。
- 在程式編譯之前，前置處理器會處理以 **#** 開頭的每一行。
  - 此例是告知前置處理器將**標準輸入/輸出標頭檔 (standard input/output header) (<stdio.h>)** 引入程式裡面，以便使用程式中的輸出函式(printf)。

3. 空白列與空白格

- 第4行是一列空白列
  - 空白行、空白字元或跳格字元(例如：tab)，目的是讓程式更容易閱讀 (*重要：程式也需排版)。
  - 這些字元統稱為**空白 (white space)**，在編譯時會自動被忽略。

4. main 函式

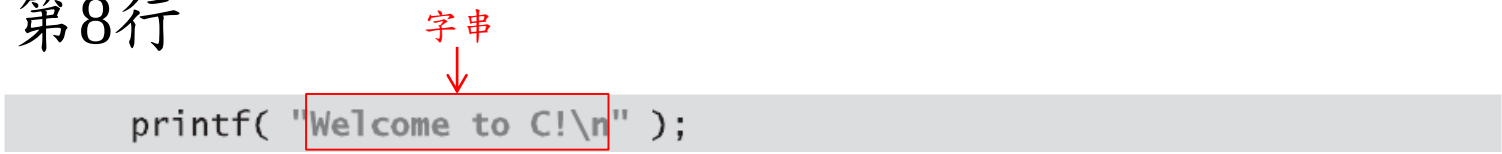
- 主程式 - 入口函式

  - 每個C程式都有的主要函式 (進入點)
  - 在**main**之後有一對小括號 (...)，表示**main**是程式的一個建構區塊，稱為**函式 (function)**
  - 函式允許傳回一些結果，以main為例，左方關鍵字**int**，表示**main**函式會「回傳(return)」一個整數值(之後會詳細說明)
  - **main**的小括號內**void**，表示**main**沒有餵入任何資訊

- 每個函式**本體(body), 如：{…本體…}**
  - 以左大括號 (left brace) { 表示函式開始位置
  - 以右大括號 (right brace) } 表示函式結束位置。
  - 這兩個括號以及之間的程式，稱為一個區塊 (block)

5. 輸出敘述式(Output statement)

- 第8行



- 命令電腦執行一個**動作(action)**
  - 將雙引號(”…”)內之**字串(string)** 顯示於螢幕上。
  - 字串有時稱做**字元字串(character string)**、**訊息(message)**，或是**字面常數(literal)**。

- 其中 **printf** 為**輸出函式**
  - 字母**f**表示「formatted (格式化的)」
  - 括號(...)、括號裡的引數/參數(argument)及分號(semicolon「 ; 」)等，稱為一項**敘述式 (statement)**。
- 每行敘述，須以分號 (也叫敘述結束符號，**statement terminator)** 做結束。

容易漏掉

```
printf( "Welcome to C!\n" );
```

7. 跳脫序列(Escape sequence)

- 特別注意，字元 **\n** 不會顯示在螢幕上
- 反斜線符號 (\) 為首的元素，通稱為**跳脫字元(escape character)**

```
printf( "Welcome to C!\n" );
```

- 表示**printf**將印出特殊字元。
  - 跳脫序列 **\n**表示**換行 (newline)**, 表示換到下一行最前端。

跳脫字元	說明
\n	• 換行：將游標移到下一行開始處
\t	• 水平tab：將游標移到下一個tab定位點
\a	• 警告：讓系統發出聲音或是顯示警告，但不改變游標位置
\\	• 反斜線符號：在字串中插入一個反斜線字元
\”	• 雙引號：在字串中插入一個雙引號

- 圖2.2　常見之跳脫字元

8. 連結器(linker)與可執行(executable)程式

- 標準函式庫中的函式 (如：**printf**或**scanf**) 並非 C 語言的一部分。
  - 因此，編譯器無法發現他們是否有拼字錯誤。

- 當編譯器在編譯**printf**敘述時，它只是在目的碼中空出一塊空間，用來「呼叫」此函式庫函式。
  - 編譯器不知道函式庫函式在哪裡，只有連結器(linker)知道
  - 直到連結器進行連結時，才會找出函式庫函式的位置，並在目的碼(object program)中插入對函式庫函式的正確呼叫。
- 最後，目的碼便完整且可執行
  - 已完成連結之程式，通常稱為**可執行(executable)**程式

9. 使用多個 printf

- **printf**函式能以多種不同的方式印出"**Welcome to C!**"
  - 舉例來說，下方程式可產生與前例相同之結果
  - 由於每個**printf**會從上一個**printf**結束列印的地方繼續列印。
  - 第一個**printf**(第8行) 印出**Welcome**及一個空格
  - 第二個**printf**(第9行) 接續從空格之後開始列印

fig02_03.c

```
1  // Fig. 2.3: fig02_03.c
2  // Printing on one line with two printf statements
3  #include <stdio.h>
4
5  // function main begins program execution
6  int main( void )
7  {
8     printf( "Welcome " );  //可以使用多個printf來呈現
9     printf( "to C!\n" );
10 } // end function main
11
```

程式碼

```
Welcome to C!
```

- 同一個**printf**，可插入**newline**字元，以印出多行文字 (如下圖)。
- 每碰到 **\n** (newline)跳脫序列時，**printf**便移至下一行的起頭位置進行輸出文字

程式碼

fig02_04.c

```
1  // Fig. 2.4: fig02_04.c
2  // Printing multiple lines with a single printf
3  #include <stdio.h>
4
5  // function main begins program execution
6  int main( void )
7  {
8     printf( "Welcome\nto\nC!\n" );          //可以有不同的做法
9  } // end function main
10
```

//換行

//換行

```
Welcome
to
C!
```

圖2.4　以一個printf列印數行文字

2.3 第二個簡單的C程式：兩個整數相加

- 使用標準函式庫之輸入函式(**scanf**)，來讀取使用者鍵盤輸入的兩個整數，並計算兩數值之和(加總)，再以**printf**將結果輸出螢幕
  - 程式如下：

fig02_05.c

```
1  // Fig. 2.5: fig02_05.c
2  // Addition program
3  #include <stdio.h>
4
5  // function main begins program execution
6  int main( void )
7  {
8     int integer1; // first number to be entered by user
9     int integer2; // second number to be entered by user
10
11    printf( "Enter first integer\n" ); // prompt
12    scanf( "%d", &integer1 ); // read an integer
13
14    printf( "Enter second integer\n" ); // prompt
15    scanf( "%d", &integer2 ); // read an integer
16
17    int sum; // variable in which sum will be stored
18    sum = integer1 + integer2; // assign total to sum
19
20    printf( "Sum is %d\n", sum ); // print sum
21 } // end function main
```

(練習Trace code)

儲存到變數之位址

整數格式

```
Enter first integer
45
Enter second integer
72
Sum is 117
```

1. 變數(Variable)與變數定義(Variable definition)

- 第8-9 行為變數**定義 (definitions)**

```
int integer1; // first number to be entered by user
int integer2; // second number to be entered by user
```

註解

  - **integer1**、**integer2**和**sum**是**變數 (variables)** 的名稱
  - 變數是電腦記憶體中的特定位置，用以存放數值供程式使用
  - **int**表示整數型態(Integer)，只能存放整數值，如：7、–11、0、31914 等整數。
- 變數定義也可以合併成為單一敘述式，如下：

```
int integer1, integer2;    //方便,但也不好註解
```

2. 使用變數前，必須先宣告變數定義

- 變數使用前，均需在程式前宣告名稱和資料型別。
  - C語言允許在main函式內任何地方宣告變數定義
  - 前題是在變數第一次使用前，就先完成宣告 (但有些舊編譯器不允許任意位置變數宣告)

3. 識別字(Identifier)和大小寫區別(Case sensitivity)

- C語言當中，變數名稱可以使用任意合法的**識別字(identifier)**。
  - 識別字是由字母、數字和底線（ _ ）組成的一連串字元，但第一個字元不可以是數字。
    - 例如：aAbc_0187 (可以), 0xc7a (錯誤)
  - 識別字的長度沒有特別限制
  - 識別字有嚴格**區分大小寫 (case sensitive)**：
    - **a1**與**A1**會被視為不同的識別字

4. 提示訊息(Prompting messages) //對使用者更友善

- 第11行

```
printf( "Enter first integer\n" ); // prompt
```

- 按照字面，在螢幕上輸出"**Enter first integer**"，並將游標移至下一行開頭
  - 此訊息稱為**提示(prompt)**，指示使用者以便進行某特定動作

5. scanf函式與格式化輸入(Formatted input)

- 第12行


```
scanf( "%d", &integer1 ); // read an integer
```

- **scanf函式**是從使用者處讀取一個數值
  - **scanf**函式由標準輸入(通常是鍵盤)讀取輸入值。
  - %為一個跳脫字元，作為轉換指定詞的開端，%d表示十進位數值(decimal digit)。
  - 接收參數**integer1**需以&(ampersand)開頭，&為位址運算子(address operator)，&加上變數名稱，可告知**scanf**的參數**integer1**存放之記憶體位址，以便將值儲存於該位址。

6. 指定敘述句

- 第18行的**指配敘述句 (assignment statement)**

```
sum = integer1 + integer2; // assign total to sum
```

- 計算變數**integer1**和**integer2**的總和，並使用指定運算子“=“ (assignment operator) 將結果設指配給變數**sum**。
- 大部分計算都是用指配敘述式來執行。
  - 運算符號 “**=**” 和 “**+**” ，稱為二元運算子 (binary operator)，由於需由二個**運算元 (operands)**作結合
  - 例如：A + B, C = 5 等

7. 使用格式控制字串列印

- 第20行

```
printf( "Sum is %d\n", sum ); // print sum
```

//印出變數

- 執行函式**printf**，在螢幕上印出字面常數”**Sum is**”及變數**sum**之值。
  - 若printf內含參數，則表示有資料將要被輸出。

8. 在printf敘述中進行計算

- 可以將計算放置於**printf**敘述裡，合併成

```
printf( "Sum is %d\n", integer1 + integer2 );
```

//更簡潔(不用再多加一個變數)

- 最後，第21行的右大括號(}) 表示main函式結束

9. 編譯錯誤(Compile Error)及編譯時期錯誤

- 當編譯器無法辨識特定敘述時，便會產生語法錯誤(syntax error)
- 通常會發出錯誤訊息(error message)，以協助找出錯誤敘述
- 語法錯誤也稱為編譯錯誤 (compile errors)或編譯時期錯誤(compile-time errors)

(以後會遇到：執行時錯誤(Run Time Error!))

2.4 記憶體觀念

- 在前例中，比如：**integer1**、**integer2**和**sum**等變數名稱，都會對應到電腦中的記憶體位置(locations)
  - 每個變數都具有：一個**名稱(name)**、一個**型別 (type)**、及一個**數值 (value)**。
  - 舉例：圖2.5的加法程式(第18行)

```
sum = integer1 + integer2; // assign total to sum
```

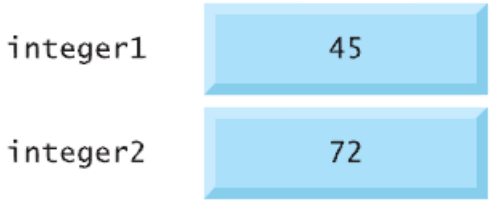
- 執行時，使用者鍵入的數值會被放置在**integer1**所被指定的記憶體位置。
  - 舉例來說：使用者輸入數字**45**作為變數**integer1**的值。

圖2.6　變數名稱及數值的記憶體位置

- 回到加法程式，當下列敘述式 (第15行)

```
scanf( "%d", &integer2 ); // read an integer
```

- 執行時，假設使用者輸入**72**，此數值將存到**integer2**的位置，其記憶體配置如下圖所示。



- 當程式讀取**integer1**和**integer2**的數值後，會將此兩值相加 (sum = interger1+ interger2)，然後將其和放到變數**sum**，敘述式如下：

```
sum = integer1 + integer2; // assign total to sum
```

- 以取代目前sum變數中的數值。

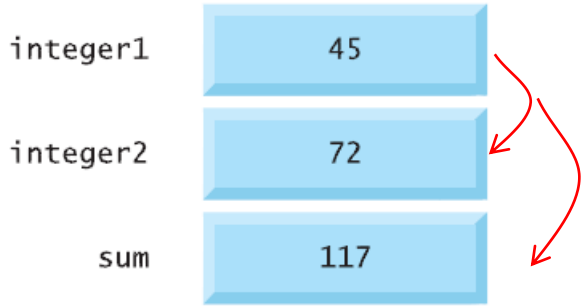


圖2.8　在執行計算後的記憶體內容

2.5 C語言的算術運算

C的運算	算術運算子 (arithmetic operator)	代數運算式	C運算式
加法	+	$f+7$	f + 7
減法	−	$p-c$	p - c
乘法	*	bm	b * m
除法	/	x/y or $\frac{x}{y}$ or $x \div y$	x / y
模數除法	% //取餘數	$r \bmod s$	r % s

- 圖2.9 算術運算子(arithmetic operator)

1. 整數除法與模數運算子

- **整數除法 (Integer division)** 得到的結果會是個整數 (僅取整數部分)
  - 例如：`7/4`會等於`1`，而`17/5`會等於`3`。
- **模數運算子 % (remainder operator)** 是整數相除後的餘數。
  - 模數運算子是個整數型態運算子，其運算元必須是整數。
  - 運算式：`x%y`會計算`x`除以`y`的餘數。
  - 例如：`7%4`等於`3`，而`17%5`等於`2`。

//Lab練習題

2. 橫行形式的算術運算式

- C語言的算術運算式須以**橫行形式 (straight-line form)** 輸入，以便將程式輸入電腦。
  - 因此，像是「a除以b」的運算式必須寫成a/b
  - 如此，運算子和運算元皆會排成橫行(橫式)

- 請注意，以下的代數符號：

$$\frac{a}{b}$$

  - 無法為編譯器接受 (但未來某些特殊用途的軟體將可以提供更多的支援)

3. 小括號用於子運算式

- 在C程式運算中，小括號(parentheses) (...)用法跟代數運算的用法類似
  - 例如：將a乘以b + c的結果算式，可寫成 a * (b + c)

4. 運算子之優先順序規則

- 規則：同一對小括號中的運算子優先進行計算
  - 小括號具有「最高優先權」
  - 若為**巢狀 (nested)** 或稱為**嵌入 (embedded)** 之小括號等多層括號，如：

```
( ( a + b ) + c )
```

  - 則最內層小括號之運算，將最優先計算。

- 接著，將會處理乘法、除法和模數運算。
  - 當運算式有多個乘法、除法和模數運算，則依序從左到右進行計算。
  - 規則：乘法、除法和模數運算，具有相同優先權層級。
- 然後，進行加、減之運算。
  - 當運算式有多個加、減法運算，則依序從左到右進行計算。
  - 規則：加、減的優先順序等級是相同的，但是次優先於乘法、除法和模數運算。
- 最後，會處理配值(=)運算子。

	運算子	運算	計算的順序（優先順序）
最高優先	( )	小括號	優先計算。如果小括號是巢狀的，則會先計算最內層的。若有數個此類型的運算，則會從左到右運算。
次高優先	*	乘法	第二個計算。若有數個此類型的運算，則會從左到右運算。
	/	除法	
	%	模數除法	
較低優先	+	加法	第三個計算。若有數個此類型的運算，則會從左到右運算。
	-	減法	
	=	指定	最後計算。

圖2.10 算術運算子的優先順序

5. 代數與C運算式

- 舉例幾個運算式的優先順序
  - 每例含：代數運算式和對應C運算式。
  - Ex. 1: 計算5個數(a, b, c, d, e)的算術平均數：

Algebra: $m = \frac{a+b+c+d+e}{5}$

```
C:      m = ( a + b + c + d + e ) / 5;
```

  - 利用小括號將加法圍起來
    - 若未加上小括號，將會誤算成 **a + b + c + d + e/5**，如下：

$$a+b+c+d+\frac{e}{5}$$

- Ex 2: 下式為直線方程式：

(代數)Algebra: $y = mx + b$

C:
```
y = m * x + b;
```

  - 此例不需要小括號 (因為先乘除後加減：乘法優先權高於加法優先權)

- Ex 3: 下式包含模數運算 (%)、乘法、除法、加法、減法和賦值等運算：

//先乘除後加減

Algebra: $z = pr\%q + w/x - y$

C:
```
z = p * r % q + w / x - y;
```

6 1 2 4 3 5 //數字為運算順序

  - 運算式下方數字為執行順序。
    - 首先，由於沒有小括號，乘法、模數運算和除法會先依據從左到右之順序計算 (因為優先權高於加減法)
    - 接著，再計算加法、減法 (從左到右)
    - 最後，將計算結果賦值於變數z。

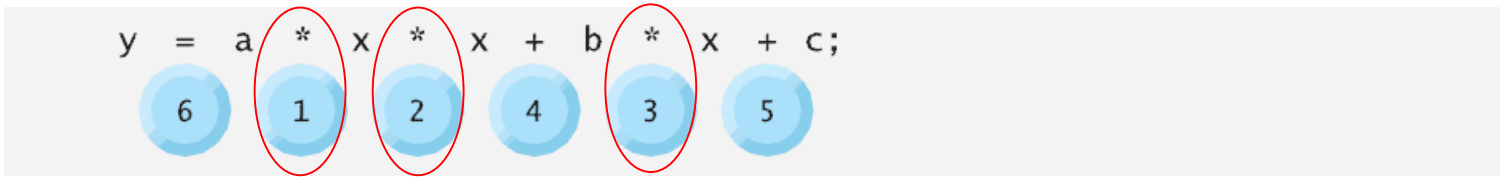
- Ex 4: 下方運算式為單層括號 (兩對括號屬於「同層級的」)

```
a * ( b + c ) + c * ( d + e )
```

- Ex 5: 若括號層級內還有其他括號，則屬於「多層級」
  - a * (b + c * (d + e))
    //內層優先計算

6. 二次多項式計算

- 為了熟悉運算子優先順序，以下來看C語言如何計算二次多項式:



  - 運算式下方數字，表示C語言執行順序。

- 觀念：如同代數運算，運算式加入非必要的小括號，可讓運算式的計算順序更清楚(易讀)。
  - 這些括號稱為**多餘括號 (redundant parentheses)**
  - 例如，將前述敘述加上以下小括號，可以讓程式更易讀：

```
y = ( a * x * x ) + ( b * x ) + c;   //內層優先計算
```

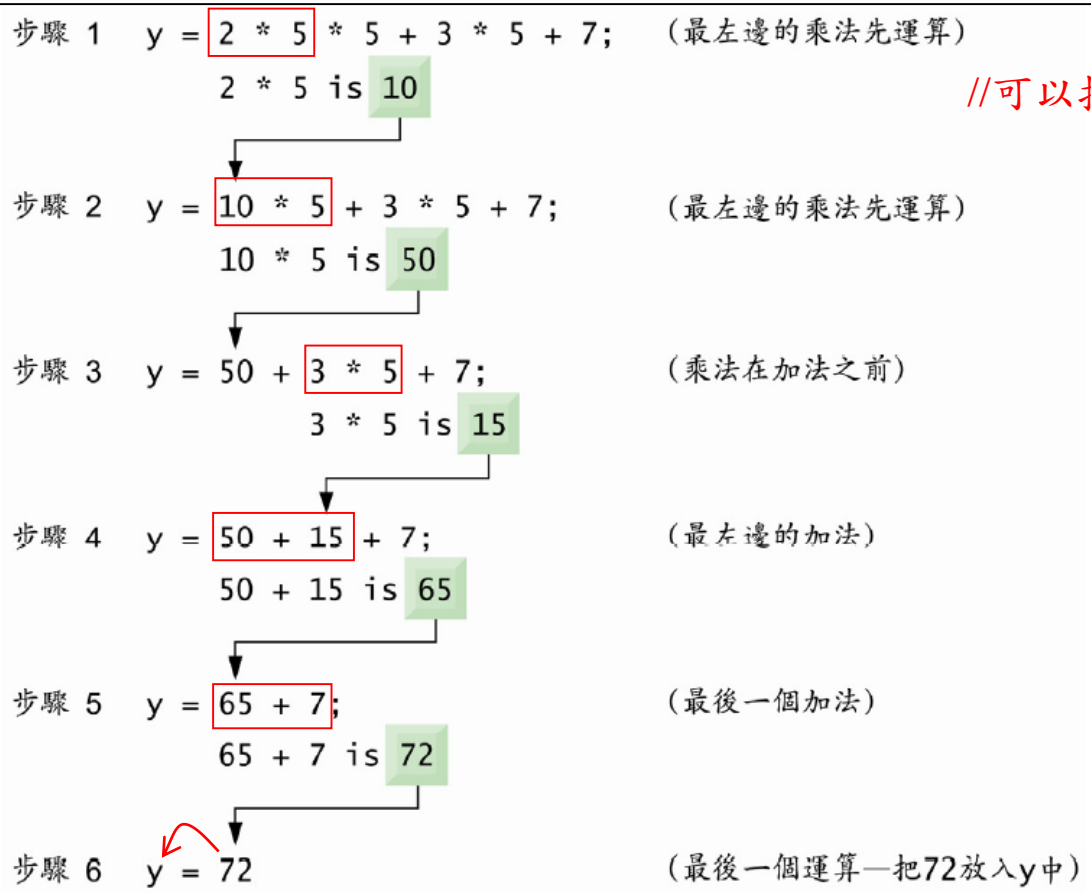


圖2.11　二次多項式的計算順序

2.6 決策判斷：等號運算子和關係運算子

- C語言的**if**敘述，用來來進行判斷真(True)或偽(False)
  - 若條件為真(true)，則程式會執行**if**本體{...}中的敘述
  - 若條件是偽(false)，則程式會執行**else**本體{...}敘述

標準代數的等號或關係運算子	C 的等號或關係運算子	C 的條件式範例	C 條件式的意義
等號運算子			
=	==（容易錯）	x == y	x 等於 y
≠	!=	x != y	x 不等於 y
關係運算子			
>	>	x > y	x 大於 y
<	<	x < y	x 小於 y
≥	>=	x >= y	x 大於或等於 y
≤	<=	x <= y	x 小於或等於 y

圖2.12　等號運算子和關係運算子

- 下例，共用了6個**if**敘述來比較輸入的兩個數。
- 若任一個**if**敘述條件滿足，則對應**printf**敘述便會執行。
- 此例包含三個示範輸出情形。

fig02_13.c

```
1  // Fig. 2.13: fig02_13.c
2  // Using if statements, relational
3  // operators, and equality operators
4  #include <stdio.h>
5
6  // function main begins program execution
7  int main( void )
8  {                                    //提示文字
9     printf( "Enter two integers, and I will tell you\n" );
10    printf( "the relationships they satisfy: " );
11
12    int num1; // first number to be read from user
13    int num2; // second number to be read from user
14                                       //讀取兩數字
15    scanf( "%d %d", &num1, &num2 ); // read two integers
16
17    if ( num1 == num2 ) {//若相同
18       printf( "%d is equal to %d\n", num1, num2 );
19    } // end if
20
```

fig02_13.c

```
21    if ( num1 != num2 ) {    //若不同
22       printf( "%d is not equal to %d\n", num1, num2 );
23    } // end if
24
25    if ( num1 < num2 ) {     //若小於
26       printf( "%d is less than %d\n", num1, num2 );
27    } // end if
28
29    if ( num1 > num2 ) {     //若大於
30       printf( "%d is greater than %d\n", num1, num2 );
31    } // end if
32
33    if ( num1 <= num2 ) {    //若小於等於
34       printf( "%d is less than or equal to %d\n", num1, num2 );
35    } // end if
36
37    if ( num1 >= num2 ) {    //若大於等於
38       printf( "%d is greater than or equal to %d\n", num1, num2 );
39    } // end if
40 } // end function main
```

圖2.13　使用if敘述式、關係運算和等號運算

```
Enter two integers, and I will tell you
the relationships they satisfy: 3 7
3 is not equal to 7
3 is less than 7
3 is less than or equal to 7
```

//輸入兩數字

```
Enter two integers, and I will tell you
the relationships they satisfy: 22 12
22 is not equal to 12
22 is greater than 12
22 is greater than or equal to 12
```

//輸入兩數字

```
Enter two integers, and I will tell you
the relationships they satisfy: 7 7
7 is equal to 7
7 is less than or equal to 7
7 is greater than or equal to 7
```

//輸入兩數字

7. 數值比較

- 其中，**if**敘述式(17-19行)

```
if ( num1 == num2 ) { //條件本體
   printf( "%d is equal to %d\n", num1, num2 );
}
```

  - 進行比對兩變數**num1**和**num2** 之數值是否相等。

- 下表列出運算子之運算優先順序 (由高至低)
  - 注意：等號(==)也是運算子。
  - 除了賦值運算子 (=) 之外，其他運算子結合性(association)都是由左至右。
  - 賦值運算子 (=) 是由右至左結合

高 → 優先順序 → 低

運算子	結合性
() //括弧最高	從左到右
* / %	從左到右
+ -	從左到右
< <= > >=	從左到右
== !=	從左到右
=	從右到左

圖2.14 目前提到的運算子優先順序和結合性

- C語言中，特定文字屬於**關鍵字(keywords)** 或**保留字(reserved words)**。
  - 下圖列出C語言常見的關鍵字(特殊用途)。
  - 關鍵字具有特殊用途，不可作為變數名稱。

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

底線開頭(內建變數)

圖2.15　C語言常見之關鍵字