程式設計　第三章


www.parallax.com
PING)))
HTC
10
08
MaxStream
XBee
ZX-BLUETOOTH
UNO

洪麗玲

除了指定運算子 = 之外，其他所有運算子的結合性都是由左至右。
指定運算子 (=) 是從右至左結合

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

- 小叮嚀

  = 與 ==的差異

  “==“, “!=“, “>=“, “<=“ 中間不能空格

  “{“ 與 “}” 請一同寫下

– 避免單一參數的printf函式

- 指導原則的其中一項，就是避免在使用**printf**時，只用一個字串當作參數。如果你要顯示一個以換行(newline) 結束的字串，請使用**puts函式(puts function)**，它將參數裡的字串後加上換行字元並顯示。如圖2.1的第8行。

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

- 應該寫成:

```
puts( "Welcome to C!" );
```

- 我們字串後面沒有**\n**，因為puts會幫自動加上。
- 如果你要顯示一個字串且不使用換行符號結尾，請在使用**printf**時帶入兩個參數。**轉換指定詞 (conversion specifier) %s** 可用於顯示一個字串，例如圖2.3第8行。

```
    printf( "Welcome " );
```

- 應該寫成：

```
    printf( "%s", "Welcome " );
```

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
```

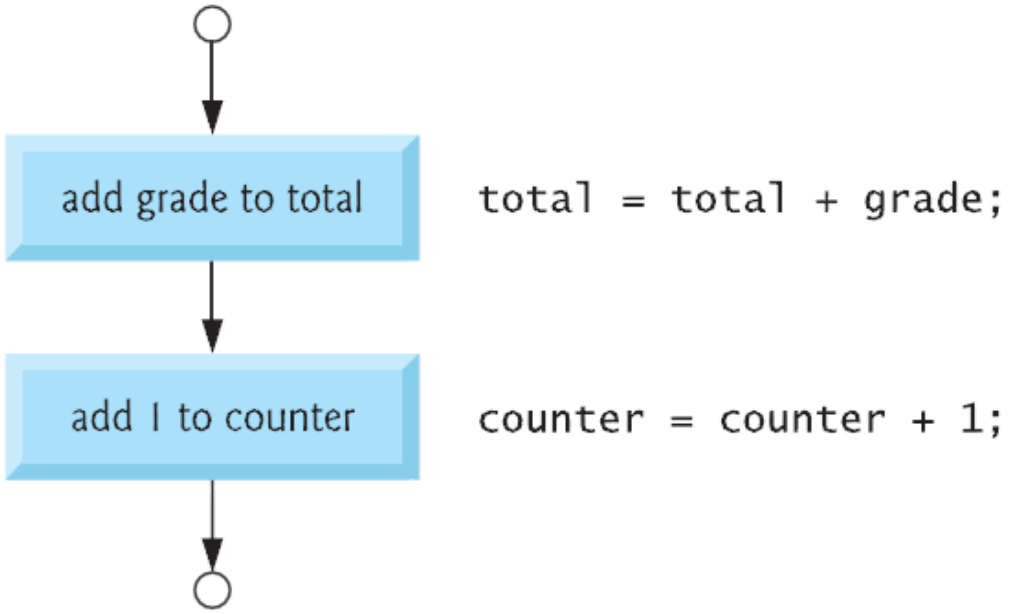
3.4 控制結構

- 一般說來，程式中的敘述式是以他們在程式中的順序一個接一個地被執行。這叫做**循序式的執行 (sequential execution)**。
- 有些C敘述式能夠讓你指定下一個執行的敘述式 (非循序式的)。這叫做**控制權的移轉 (transfer of control)**。

- **控制結構 (control structure)**
  - **循序結構 (sequence structure)**
  - **選擇結構 (selection structure)**
  - **重複結構 (repetition structure)**
  - 其中循序結構是C內建的特性，除非改變程式執行的流程，否則電腦會自動地按照你所寫的C敘述式的順序，一行一行地執行。圖3.1所示的**流程圖 (flowchart)** 片段，說明了C的循序結構。

– 流程圖 (flowchart)

- 流程圖是整個演算法或是演算法的一部分的圖形表示法。流程圖使用具有特殊涵義的標誌來繪製，像是矩形、菱形、圓角矩形以及圓形等等；這些標誌用稱為**流向(flowline)**的箭頭連接起來。


add grade to total
total = total + grade;
add 1 to counter
counter = counter + 1;

- 流程圖中最重要的也許是**菱形符號 (diamond symbol)**，它也被稱為**判斷符號(decision symbol)**。這種符號表示某項判斷將在此進行。

– 選擇結構

- C語言以敘述式的形式提供了三種選擇結構。
- **if**選擇結構 (3.5節) 在條件式為真時選擇 (執行) 某項動作，而當條件為偽時則跳過這項動作。
- **if...else**選擇敘述式 (3.6節) 在條件為真時執行某項動作，而當條件為偽時則執行另一項動作。
- **switch**選擇敘述式 (在第4章當中討論) 會依運算式的不同，選擇執行許多動作中的一項。

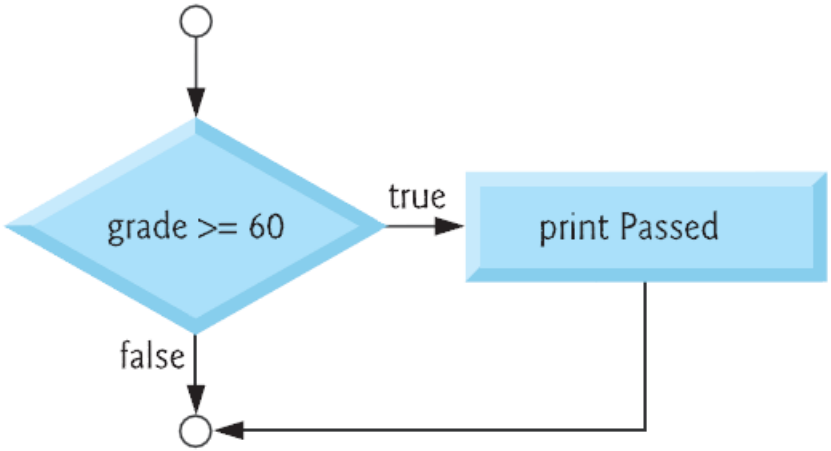
– 重複敘述

- C以敘述式的形式提供了三種重複結構
- **while** (3.7節)
- **do...while** (第4章討論)
- **for** (第4章討論)。
- 以上所介紹的便是C的所有控制結構。C只有七種控制敘述式：循序、三種選擇和三種重複。每個C程式都是依據其演算法的需要，組合這七種結構所形成的。

3.5 if選擇敘述

- 選擇敘述可用來選取不同功能的動作。例如，假設考試中及格的成績為60分，以下的虛擬碼敘述式

```
If students grade is greater than or equal to 60
    Print Passed
```



- 圖3.2 單一選擇**if**敘述式的流程圖

IF 的寫法

```
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  {
9     int num1; // first number to be read from user
10    int num2; // second number to be read from user
11
12    printf( "Enter two integers, and I will tell you\n" );
13    printf( "the relationships they satisfy: " );
14
15    scanf( "%d%d", &num1, &num2 ); // read two integers
16
17    if ( num1 == num2 ) {
18       printf( "%d is equal to %d\n", num1, num2 );
19    } // end if
20
```

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

```
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
```

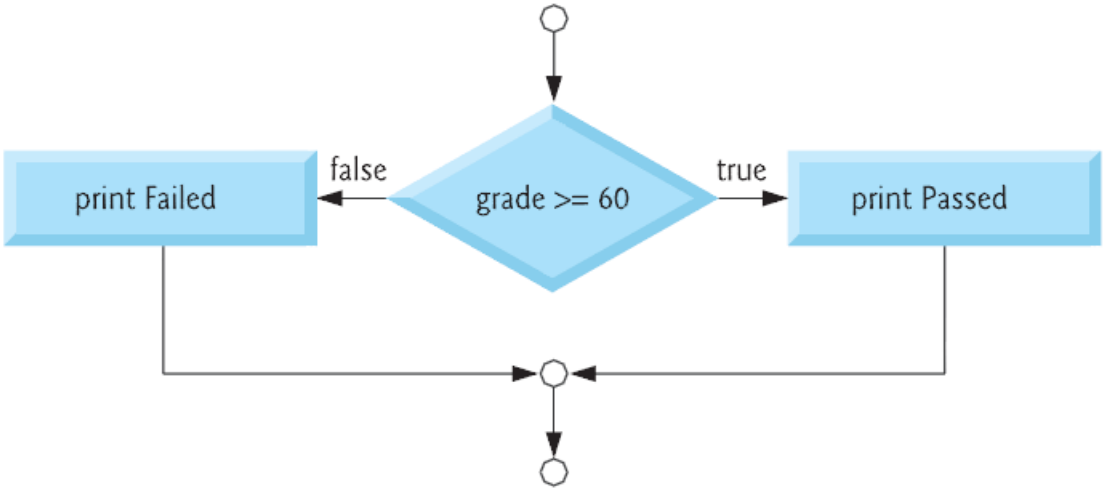
```
Enter two integers, and I will tell you
the relationships they satisfy: 22 12
22 is not equal to 12
22 is greater than 12
22 is greater than or equal to 12
```

```
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
```

3.6 if...else選擇敘述式

- **if...else**選擇敘述式則讓你可依據條件的真偽，來執行不同的動作。舉例來說，下列的虛擬碼敘述式

```
If students grade is greater than or equal to 60
    Print Passed
else
    Print Failed
```



- 圖3.3 C之雙重選擇**if...else**敘述式的流程圖

– 巢狀的if...else敘述式

- 我們可將**if...else**敘述式放到另一個**if...else**敘述式裡，構成**巢狀的if...else敘述式 (nested if...else statements)**，用以檢測多重的狀況。

```
If students grade is greater than or equal to 90
    Print A
else
    If students grade is greater than or equal to 80
        Print B
    else
        If students grade is greater than or equal to 70
            Print C
        else
            If students grade is greater than or equal to 60
                Print D
            else
                Print F
```

```
if ( grade >= 90 ) {
   puts( "A" );
} // end if
else {
   if ( grade >= 80 ) {
      puts("B");
   } // end if
   else {
      if ( grade >= 70 ) {
         puts("C");
      } // end if
      else {
         if ( grade >= 60 ) {
            puts( "D" );
         } // end if
         else {
            puts( "F" );
         } // end else
      } // end else
```

```
if ( grade >= 90 ) {
   puts( "A" );
} // end if
else {
   if ( grade >= 80 ) {
      puts("B");
   } // end if
   else {
      if ( grade >= 70 ) {
         puts("C");
      } // end if
      else {
         if ( grade >= 60 ) {
            puts( "D" );
         } // end if
         else {
            puts( "F" );
         } // end else
      } // end else
   } // end else
} // end else
```

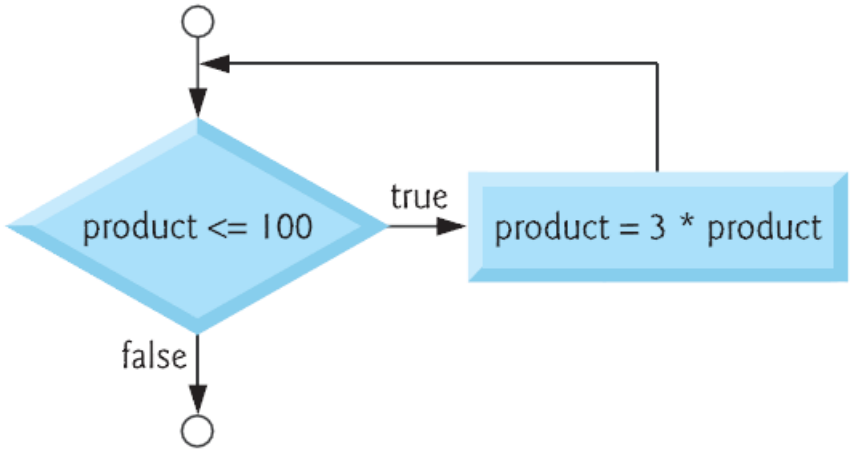
```
if ( grade >= 90 ) {
   puts( "A" );
} // end if
else if ( grade >= 80 ) {
   puts( "B" );
} // end else if
else if ( grade >= 70 ) {
   puts( "C" );
} // end else if
else if ( grade >= 60 ) {
   puts( "D" );
} // end else if
else {
   puts( "F" );
} // end else
```

- 這兩種方式對C編譯器來說是相同的。
- 後者較受到歡迎的原因是它可避免因過深的縮排導致程式向右傾斜。

3.7 while重複敘述式

- **重複敘述式 (repetition statement) ，或也稱為循環敘述(iteration statement)**可讓你指定在某種條件持續為真時，重複執行同一項動作。下面這個虛擬碼敘述式

While there are more items on my shopping list
Purchase next item and cross it off my list



需要重複執行時？　迴圈

A

```
while ( x <= 10 ) { // loop while x is less than or equal to 10
    sum =  sum + x; // add x to sum
    x = x + 1; // increment x
} // end while
```

B

```
while ( grade != -1 ) {
   total = total + grade; // add grade to total
   counter = counter + 1; // increment counter
   // get next grade from user
   printf( "%s", "Enter grade, -1 to end: " ); // prompt for
input
   scanf("%d", &grade); // read next grade
} // end while
```

3.8 建構演算法案例研究1：計數器控制的重複結構

- 為了說明如何發展一個演算法，我們利用了數種方法來解決計算全班平均成績的問題。請看下列的問題描述：
- 10個學生的一個班級進行一次測驗。你手上已有這次測驗的成績 (從0到100的整數)。請求出此班的平均成績為何。

- 全班平均即等於所有成績的總和除以學生的人數。在電腦上解決此問題的演算法必須輸入每個學生的成績，執行求平均值的計算，然後再將結果印出來。
- 我們利用**計數器控制的重複結構 (counter-controlled repetition)**，一次一個地輸入這些成績。在這項技巧裡，我們用了一個稱為**counter (計數器)** 的變數，來指定某一組陳述句應被執行的次數。在本例中，當counter超過10時，重複動作便告結束。本節我們將只列出虛擬碼演算法 (圖3.5) 及其相對應的C程式 (圖3.6)。

```
1    Set total to zero
2    Set grade counter to one
3
4    While grade counter is less than or equal to ten
5        Input the next grade
6        Add the grade into the total
7        Add one to the grade counter
8
9    Set the class average to the total divided by ten
10   Print the class average
```

圖3.5　利用計數器控制的重複結構來解決全班平均問題的虛擬碼 演算法

```c
// Fig. 3.6: fig03_06.c
// Class average program with counter-controlled repetition.
#include <stdio.h>

// function main begins program execution
int main( void )
{
   unsigned int counter; // number of grade to be entered next
   int grade; // grade value
   int total; // sum of grades entered by user
   int average; // average of grades

   // initialization phase
   total = 0; // initialize total
   counter = 1; // initialize loop counter

   // processing phase
   while ( counter <= 10 ) { // loop 10 times
      printf( "%s", "Enter grade: " ); // prompt for input
      scanf( "%d", &grade ); // read grade from user
      total = total + grade; // add grade to total
      counter = counter + 1; // increment counter
   } // end while

   // termination phase
   average = total / 10; // integer division

   printf( "Class average is %d\n", average ); // display result
} // end function main
```

```
Enter grade: 98
```

ineering

3.9 以從上而下逐步改進的方式建構演算法 案例研究2：警示訊號控制的重複結構

- 現在讓我們將全班平均問題一般化。請看下面所述的問題：
- 發展一個求全班平均的程式，它可以處理任何數量的成績。
- 在第一個全班平均的例子裡，成績的個數(10個) 是事先知道的。而在本例中則不知會輸入多少個成績。此程式必須能夠處理任何數目的成績。

– 從上而下逐步改進的技術

- 我們用一種**從上而下逐步改進的技術 (top-down, stepwise refinement)**，來發展上述的全班平均程式。我們先從代表總敘述式 (**top**) 的虛擬碼開始：

Determine the class average for the quiz

- 總敘述式是一個單一敘述式，它涵蓋了程式所有的功能。我們先將總敘述式分割成數項較小的工作，並依它們執行的順序列出來。於是我們得到了下列**初步的改進 (first refinement)**

> *Initialize variables*
> *Input, sum, and count the quiz grades*
> *Calculate and print the class average*

- 到目前為止我們只使用了循序結構——以上所列的各步驟是一個接一個執行。

```
1   Initialize total to zero
2   Initialize counter to zero
3
4   Input the first grade
5   While the user has not as yet entered the sentinel
6       Add this grade into the running total
7       Add one to the grade counter
8       Input the next grade (possibly the sentinel)
9
10  If the counter is not equal to zero
11      Set the average to the total divided by the counter
12      Print the average
13  else
14      Print No grades were entered
```

```
1  // Fig. 3.8: fig03_08.c
2  // Class-average program with sentinel-controlled repetition.
3  #include <stdio.h>
4
5  // function main begins program execution
6  int main( void )
7  {
8     unsigned int counter; // number of grades entered
9     int grade; // grade value
10    int total; // sum of grades
11
12    float average; // number with decimal point for average
13
14    // initialization phase
15    total = 0; // initialize total
16    counter = 0; // initialize loop counter
17
18    // processing phase
19    // get first grade from user
20    printf( "%s", "Enter grade, -1 to end: " ); // prompt for input
21    scanf( "%d", &grade ); // read grade from user
22
```

圖3.8　以警示值控制重複結構來解決全班平均問題(1/2)

```
23      // loop while sentinel value not yet read from user
24      while ( grade != -1 ) {
25         total = total + grade; // add grade to total
26         counter = counter + 1; // increment counter
27
28         // get next grade from user
29         printf( "%s", "Enter grade, -1 to end: " ); // prompt for input
30         scanf("%d", &grade); // read next grade
31      } // end while
32
33      // termination phase
34      // if user entered at least one grade
35      if ( counter != 0 ) {
36
37         // calculate average of all grades entered
38         average = ( float ) total / counter; // avoid truncation
39
40         // display average with two digits of precision
41         printf( "Class average is %.2f\n", average );
42      } // end if
43      else { // if no grades were entered, output message
44         puts( "No grades were entered" );
45      } // end else
46   } // end function main
```

圖3.8　以警示值控制重複結構來解決全班平均問題(2/2)

```
Enter grade, -1 to end: 75
Enter grade, -1 to end: 94
Enter grade, -1 to end: 97
Enter grade, -1 to end: 88
Enter grade, -1 to end: 70
Enter grade, -1 to end: 64
Enter grade, -1 to end: 83
Enter grade, -1 to end: 89
Enter grade, -1 to end: -1
Class average is 82.50
```

```
Enter grade, -1 to end: -1
No grades were entered
```

圖3.8　以警示值控制重複結構來解決全班平均問題

- 明確地類型與隱含地類型轉換
- 第38行

```
average = ( float ) total / counter;
```

- 含有 (**float**) 這個強制型別轉換運算子，它會為它的運算元**total**產生一個暫時的浮點數拷貝。而存放在**total**的值仍然是個整數。以這種方式來使用強制型別轉換運算子稱為**明確地轉換 (explicit conversion)**。

- 作業改成可以運算很多次
  - 限定次數
  - 由使用者決定要幾次

今日作業

- 請寫一支幫忙計算BMI值的程式
  - 先說明程式要做什麼事，並適當指引要做什麼輸入
  - 因為BMI的標準與男女有別－所以也要輸入性別
  - 收到輸入後要再輸出收到的資料以進行確認
  - BMI計算完後輸出計算結果並告知其值是否在標準範圍內

- 其他要求參照上週課程要求”程式註解”