

# 程式設計 (**Programming**)

真理大學 資訊工程系 吳汶涓老師

CH03

結構化程式的開發



# 本章綱要

3-1 簡介

3-2 演算法

3-3 虛擬程式碼

3-4 控制結構

3-5 if 選擇敘述式

3-6 if...else 選擇敘述式

3-7 重複敘述式

3-8 案例研究1

3-9 案例研究 2

3-10 案例研究3

3-11 指定運算子

3-12 遞增和遞減運算子

## 3.10 while 案例研究3：巢狀控制

### ■ 問題

- 某所學校有一份十位學生的測驗成績（1=及格，2=不及格）
- 寫個程式來分析考試結果
  - 若超過8位學生通過考試的話，印出"Raise tuition"訊息

### ■ 我們可以注意到

- 此程式必須處理10個考試結果 → 可用一個計數器來控制迴圈個數
- 計算多少個及格，多少個不及格 → 使用兩個計數器
- 每個考試結果不是輸入1便是輸入2 → 作判斷動作

## ■ 總敘述式

*Analyze exam results and decide if tuition should be raised*

## ■ 第一次改進

*Initialize variables*

*Input the ten quiz grades and count passes and failures*

*Print a summary of the exam results and decide if tuition should be raised*

## ■ 將Initialize variables 改進為

*Initialize passes to zero*

*Initialize failures to zero*

*Initialize student counter to one*

- Input the ten quiz grades and count passes and failures 改進為

*While student counter is less than or equal to ten*  
*Input the next exam result*  
*If the student passed*  
*Add one to passes*  
*else*  
*Add one to failures*  
*Add one to student counter*

- Print a summary of the exam results and decide if tuition should be raised 改進為

*Print the number of passes*  
*Print the number of failures*  
*If more than eight students passed*  
*Print "Raise tuition"*

---

```
1  Initialize passes to zero
2  Initialize failures to zero
3  Initialize student to one
4
5  While student counter is less than or equal to ten
6      Input the next exam result
7
8      If the student passed
9          Add one to passes
10     else
11         Add one to failures
12
13     Add one to student counter
14
15 Print the number of passes
16 Print the number of failures
17 If more than eight students passed
18     Print "Bonus to instructor!"
```

圖 3.9 有關考試結果問題的虛擬碼

```

1  /* Fig. 3.10: fig03_10.c */
2
3  #include <stdio.h>
4
5
6  int main( void )
7  {
8      /* initialize variables */
9      int passes = 0;
10     int failures = 0;
11     int student = 1;
12     int result;
13
14     /* process 10 students using counter-controlled loop */
15     while ( student <= 10 ) {
16
17
18         printf( "Enter result ( 1=pass,2=fail ): " );
19         scanf( "%d", &result );
20
21         /* if result 1, increment passes */
22         if ( result == 1 ) {
23             passes = passes + 1;
24         }
25         else {
26             failures = failures + 1;
27         }
28
29         student = student + 1;
30     }

```

設定初值的動作是在編譯時進行

迴圈會不斷重複，直到10位學生分數處理完畢為止

If...else...敘述式以巢狀方式放在while回圈內部

圖 3.10 有關考試結果問題之 C 程式及執行範例

```

32
33     printf( "Passed %d\n", passes );
34     printf( "Failed %d\n", failures );
35
36
37     if ( passes > 8 ) {
38         printf( "Bonus to instructor!\n" );
39     }
40
41     return 0;
42 } /* end function main */

```

圖 3.10 有關考試結果問題之 C 程式及執行範例

```

Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 2
Enter Result (1=pass,2=fail): 2
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 2
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 2
Passed 6
Failed 4

```

```

Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 2
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Passed 9
Failed 1
Bonus to instructor!

```





### 增進效能的小技巧 3.1

---

在宣告變數時便為他們設定初值，可減少程式的執行時間。



### 軟體工程的觀點 3.6

---

根據經驗顯示，利用電腦解決問題時，最困難的部分便是解決方法之演算法的開發。一旦有了正確的演算法，接下來便很容易寫出可以執行的 C 程式。

# 練習

- 撰寫一個程式來計算薪資所得，使用者輸入員工的工作時數以及每小時的工資，若工作在**40**小時內，則以正常工資計算；超過**40**小時的部分，以正常工資的**1.5**倍計算，並輸出每位員工的所得
  - **Example:** 工作**35** 小時，每小時**100**元，應得 **3500**元  
工作**45** 小時，每小時 **90**元，應得 **4275**元
  - 可由使用者不斷輸入，直到輸入 **-1**為止



## 3.11 指定運算子

- 指定運算子使運算式可以縮寫

`c = c + 3;`

可利用加法指定運算子縮寫為 `c += 3;`

- 下面格式的敘述式

*variable = variable **operator** expression;*

可以寫成

*variable **operator=** expression;*

- 指定運算子的其他範例：

`d -= 4`      `(d = d - 4)`

`e *= 5`      `(e = e * 5)`

`f /= 3`      `(f = f / 3)`

`g %= 9`      `(g = g % 9)`

指定運算子	範例運算式	展開式	指定值
假設： <code>int c = 3, d = 5, e = 4, f = 6, g = 12;</code>			
<code>+=</code>	<code>c += 7</code>	<code>c = c + 7</code>	10 到 c
<code>-=</code>	<code>d -= 4</code>	<code>d = d - 4</code>	1 到 d
<code>*=</code>	<code>e *= 5</code>	<code>e = e * 5</code>	20 到 e
<code>/=</code>	<code>f /= 3</code>	<code>f = f / 3</code>	2 到 f
<code>%=</code>	<code>g %= 9</code>	<code>g = g % 9</code>	3 到 g

圖 3.11 算術指定運算子

## 3.12 遞增和遞減運算子

- 遞增運算子 ( $++$ )
  - 可用來代替  $c+=1$
- 遞減運算子 ( $--$ )
  - 可用來代替  $c-=1$
- 前置遞增
  - 運算子放在變數之前 ( $++c$  或  $--c$ )
  - 在計算運算式之前，先改變變數
- 後置遞增
  - 運算子放在變數之後 ( $c++$  或  $c--$ )
  - 在變數改變之前，先計算運算式

## ■ 假如 **c** 等於 **5**，則


- `printf("%d", ++c);` → 會印出 6
- `printf("%d", c++);` → 會印出 5
- 在這兩個範例中，執行完最後**c**的值皆為 6

## ■ 當變數不在運算式中時

- 前置遞增與後置遞增的效果一樣

```
++c;  
printf( "%d", c );
```

```
c++;  
printf( "%d", c );
```



```
++ c;  
c++;  
c = c + 1;  
c += 1;
```



### 良好的程式設計習慣 3.7

單元運算子應與其運算元緊密地寫在一起，中間不要留有空白。

運算子	範例運算式	說明
++	++a	先將 a 遞增 1，再以 a 的新值進行運算
++	a++	以 a 目前的值進行運算，再將 a 遞增 1
--	--b	先將 b 遞減 1 再以 b 的新值進行運算
--	b--	以 b 目前的值進行運算，再將 b 遞減 1

圖 3.12 遞增和遞減運算子

```

1  /* Fig. 3.13: fig03_13.c
2     Preincrementing and postincrementing */
3  #include <stdio.h>
4
5
6  int main( void )
7  {
8     int c;
9
10
11     c = 5;
12     printf( "%d\n", c );
13     printf( "%d\n", c++ );
14     printf( "%d\n\n", c );
15
16
17     c = 5;
18     printf( "%d\n", c );
19     printf( "%d\n", ++c );
20     printf( "%d\n", c );
21     return 0;
22 }

```

先輸出，再遞增

先遞增，再輸出

5  
5  
6  
  
5  
6  
6

圖 3.13 示範前置遞增和後置遞增的差異



# 練習

- 請問下列運算式執行完的各變數結果為何？  
設，每個變數初始值皆為 5

1. `product *= x++;`      **product =25 , x = 6**

2. `z += x++ +y;`      **x =6 , y = 5 , z = 15**

3. `total -= --x;`

4. `total += x--;`

5. `total += x-- + --x;`

# 練習

- 撰寫一個程式找出使用者輸入的數字中最大值與最小值，直到使用者輸入負值為止
  - 警示訊息控制：輸入負值，則程式結束
  - 判斷：由目前最大的數值與使用者輸入的數字相比較
- 續上題，若改輸出最大的兩個數。



# 練習

- 撰寫一個程式，使用**while**迴圈指令讓使用者不斷輸入整數值，並計算出該整數值的階乘值，直到使用者輸入負值為止
- 迴文判斷：判斷使用者輸入的數是否為迴文？
  - 12321, 555, 45554, 161 以上皆為迴文

