第六章 迴圈指令

6-1 for

6_2 巢狀迴圈

6-3 while

6-4 實例探討

6_5 APCS 觀念題

6_6 APCS 實作題

前面第四、五兩章，我們已經介紹如何輸入一個人的成績，及判斷一個人的成績是否及格，也介紹如何輸入多人成績、多人成績的極大值及排序等問題。我們發現，當人數只要一多，寫起程式來真是洋洋灑灑，程式設計的領域果真如此磨人嗎？所幸，答案是否定的。因為本章要介紹一個高效率的指令，此稱為迴圈指令。C 語言常用的迴圈指令分別是 for 與 while。請鍵入以下程式，並觀察執行結果。

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *argv[]) {
    int i;
    for( i=1;i<=10;i++) {
        printf("%d",i);
    }
    return 0;
}
```

又例如，假如沒有乘法運算子，以下是乘法運算的程式。

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *argv[]) {
    int s=0;
    int a=26,b=42;
    for (int i=1 ;i<=b;i++)
        s=s+a;
    printf("%d",s);
    return 0;
}
```

以上是程式設計階段就知道要重複幾次，就用 for 迴圈，for 迴圈請看 6_1 節。但是有些情況，程式設計階段就是不知道要重複幾次，以除法為例，商就是被除數能減除數的次數，此時我們程式設計階段就是不知道要重複減幾次，所以使用 while 迴圈如下，while 迴圈請看 6_3 節。

```
#include <stdio.h>
#include <stdlib.h>
```

```
int main(int argc, char *argv[]) {
    int a=8;//被除數
    int b=3;//除數
    int q=0;//商
    while(a>=b) {
        a=a-b;
        q++;
    }
    printf("quotient = %d\n", q);        /* 商數 */
    printf("remainder= %d\n", a);        /* 餘數 */
    return 0;
}
```

6-1 for

若於程式設計階段已知迴圈的執行次數，則可使用 for 指令，for 指令的語法如下：

```
for([計數變數＝起始值] ；[迴圈運算式]；[計數變數的變量])
{
    [指令區塊 1；]
    [break；]
    [continue；]
    [goto (標籤名稱)；]
    [指令區塊 2；]
}
```

以上語法說明如下：

1. 只要“迴圈運算式”結果為 1(tuue)，則繼續執行迴圈內的指令區塊。
2. 計數變數可為正或負的整數或浮點數，正整數請看範例 6-1a，浮點數請看範例 6-1b。
3. 程式若執行到 break，則會提早離開 for 迴圈，請看範例 6-1c。
4. 程式若執行到 continue，則會略過 continue 下面的指令區塊 2，繼續執行下一個計數變量，請看範例 6-1c
5. 以下程式片段可印出 1 至 10 的整數。

```
int i;
for( i=1;i<=10;i++) {
    printf("%d",i);
}
```

6. for 指令若只有一個指令，則大括號可省略。例如，以上指令同義於：

```
int i;
 for( i=1;i<=10;i++)
    printf("%d",i);
```

但以下指令的 j++並不包含於迴圈，它只會被執行一次。

```
int i,j;
```

```
for( i=1;i<=10;i++)
    printf("%d",i);
    j++;
```

7. 上面也不能寫成

```
int i ;
for ( i=1;i=10;i++)
printf("%d",i);
```

因為 C 關係運算子沒有=，這樣是語法錯誤。其次，若寫成

```
for ( i=1;i==10;i++)
printf("%d",i);
```

則是語法沒錯，但結果錯誤，因為一開始 1==10 這件事就 false，所以什麼都沒有作就離開迴圈。

8.以下程式，每次改變量是 2。

```
for ( i=1;i<=10;i+=2)
    printf("%d",i);//13579
```

9.前面都是由小而大，都是使用『<=』，若是由大到小，則要用『>=』，這樣第一筆資料才會得到『true』，這樣才有輸出結果。

```
for( i=10;i>=1;i--) {
    printf("%d",i);//10987654321
}
```

10.以下程式每次遞減 3。

```
for( i=10;i>=1;i-=3) {
    printf("%d",i);//10741
}
```

11.舊版 C 語言編譯器不能將變數宣告寫到 for()區塊內，如下圖左，只能寫成下圖右。

```
for( int i=1;i<=10;i++) {
    printf("%d",i);
}
```

```
int i;
for( i=1;i<=10;i++) {
    printf("%d",i);
}
```

12. 指令區塊內可以放置任何合法的指令，當然也可含 for。for 內有 for，稱為巢狀迴圈。例如，以下指令可印出 1 至 10 五次，請看 6_2 節。

```
for ( i=1;i<=5;i++){
    for (j=1;j<=10;j++)
        printf("%d",j);
}
```

自我練習

1、請於 C 完成以上程式，並觀察執行結果。

範例 6-1a

請寫一個程式，印出 1 至 10，並求其和。

演算法

這題大家都會想到，這是等差級數，等差級數和的公式是：『(上底+下底)*(項數)/2』，以上式人腦的運算思維，這樣可以減少計算。但是電腦就不用如此麻煩，電腦就是計算能力強，所以可以用以下國小方法。

方法一：一個一個作，此為國小的方法。

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *argv[]) {
    int s=0;
    printf ("%d",1);
    s=s+1;
    printf ("%d",2);
    s=s+2;
    printf ("%d",3);
    s=s+3;
    printf ("%d",4);
    s=s+4;
    printf ("%d",5);
    s=s+5;
    printf("\n")
    printf("%d",s);
    return 0;
}
```

方法二：以上一個一個逐一累加，卻是電腦的強項，因為電腦的迴圈，就是用來完成這些有規律性的重複動作。

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *argv[]) {
    int i;s=0;
    for (i=1;i<=10;i++) {
        printf("%d",i);
        s=s+i;
    }
    printf("\n");
    printf("%d",s);
    return 0;
}
```

【程式說明】

1、C 語言並不會自動將所宣告變數的初值設為 0，所以若要執行累加的變數，務必自己親自設定其初值，程式如下：

```
    int s=0;
```

2、本章又是另一種運算思維的改變，電腦的強項就是計算，慢慢要改成以電腦的方式去思考問題。

補充自我練習

1、請寫一程式，可以使用迴圈輸出『ABCDEFG』。
2、請寫一程式，可以使用迴圈輸出『GFEDCBA』。
3、請寫一程式，可以使用迴圈輸出『cdefgh』。
4、請寫一程式，可以使用迴圈輸出『hgfedc』。
5、請寫一程式，可以輸出『ABCDE』五次。

自我練習

1、請寫一程式，計算 1＋3＋5＋7＋9 之和。
2、請寫一程式，計算 46＋36＋26＋16＋6 之和。
3、乘法運算。假如沒有乘法運算子，請用連加法，求兩數相乘的結果。例如，6 乘以 4，就是 6＋6＋6＋6＝24。

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *argv[]) {
    int s=0;
    int a=6,b=4;
    for (int i=1 ;i<=b;i++)
        s=s+a;
    printf("%d",s);
    return 0;
}
```

範例 6_1b 請寫一程式，可以解任意數的開根號。

演算法

解開根號人類通常使用$(a+b)^2=a^2+2ab+b^2=a^2+(2a+b)*b$，所以其直式解法如下：

		3 7 2
1	法則 3	1 3 8 3 8 4 9
2	3×20+7=67 67×7=469	3 8 4 4 6 9
3	37×20+2=742 742×2=1484	1 4 8 4 1 4 8 4
4		0

但是電腦就不用如此麻煩，電腦有循序猜值法與二分猜值法，這通通可快速求解。所謂循序猜值法，就是將所有可能的解一一代入，又稱為暴力猜值法。例如，您要求任一數的開根號，

因為開根號的結果一定在 0 與此數之間，那我就將從 0 開始，每次遞增 1、0.1、0.01 或 0.001…，至於是 1、0.1、0.01 或 0.001…，那就依您要的精密度了。此即為電腦的運算思維，二分猜值法，請看 6_4 節。

【程式列印】

1、若要求整數解，則可用 for 實現以上演算，程式如下：

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *argv[]) {
    int i;
    int y;
    int ans;
    for (i=0;i<=9;i=i+1){
        printf("%d\n",i);
        y=i*i;
        if (y==9){
            ans=i;
            break;
        }
    }
    printf("%d",ans);
}
```

2、若要求實數解，則程式如下：

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *argv[]) {
    int a=9;
    double i;
    for (i=0;i<=9;i=i+0.1){
        if (i*i==a)//實數運算不會相等
            break;
    }
    printf("%f",i);
    return 0;
}
```

3、程式修正如下：

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *argv[]) {
 int a=9;
```

```
    double i;
    for (i=0;i<=9;i=i+0.1){
        if (i*i>=a){
            break;
        }
    }
    printf("%f",i);
    return 0;
}
```

補充說明

1、程式設計的進階課程，有一門是『演算法』，此循序法是演算法裡，最基本的演算法，後面還有效率較高的演算法，例如，二分猜值法。

自我練習

1、請用暴力法求解某一正數的立方根。例如，輸入 27 可得到 3.0。本例小數點取 1 位。

2、請用暴力法求解兩數相除的結果。例如，輸入 27 與 9 可得到 3.0。本例小數點取 1 位。

3、假設有一函式 $y=f(x)=x^2-4x-5$ 請分別印出 x 從-10 到 10 的值。

4、同上題，請用暴力猜值找出其整數解。x 從-10 到 10 一一帶入，找出使函數為零的值，此即為暴力猜值法解題。

5、同上題，請找出極小值。

6、函數極值。請寫一程式，可以輸入一個一元二次函式，並求其極大或極小值。例如，輸入 $y=f(x)=x^2-2x+2$ 有極小值 1，輸入 $y=f(x)=-x^2-2x+2$ 有極大值-1，

7、假設一個一元多次方程式含有實數解，請寫一程式，可求其解。例如，$y=f(x)= x^2+x-0.75=0$ 的解是 0.5 和-1.5。提示：浮點運算時，無法得到 0，此時要使用接近 0 的判斷。例如，fabs(y)<0.0001，即可視為成立，0.0001 即是其精密度，請換成自己想要的精密度 0.1、0.01 或 0.001。

8、勘根定理。若 f(x)*f(x+1)<0，則表示 f 函數於 x 與 x+1 有一實數根。請寫一程式，可輸入一函數，並求其有多少個實根，並求其實根範圍。例如，$y=f(x)= x^2+x-0.75=0$，以上已經求出其解是 0.5 與—1.5,則實根範圍在-2 與-1，還有另一個在 1 與 2 之間，請用勘根定理驗證。

6_2 巢狀迴圈

迴圈中又有迴圈，稱為巢狀迴圈。巢狀迴圈在程式設計的領域非常重要，這種運算思維是電腦的強項，也是初學者（應該說是人類）最感頭疼的單元，本節將使用若干範例引領學生征服此運算思維。

範例 6-2a

請寫一程式，印出如下的九九乘法表。

【執行結果】

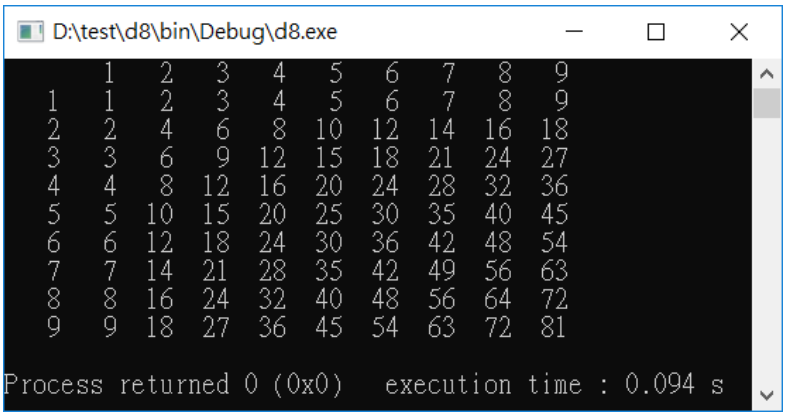
G606

【程式列印】

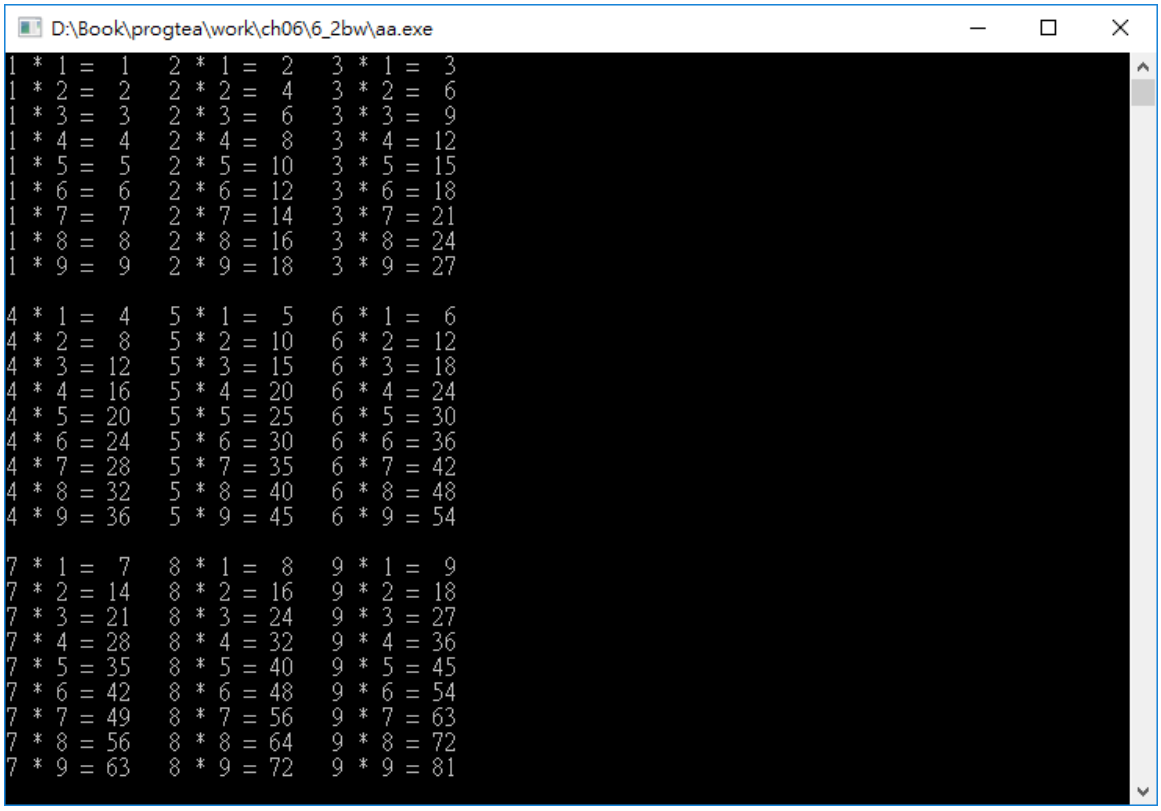
```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *argv[]) {
    int i,j;
    for (i=1;i<=9;i++){
        for (j=1;j<=9;j++){
            printf("%d * %d = %2d ",i,j,i*j);
        }
        printf("\n");
    }
    return 0;
}
```

自我練習

1、請寫一程式，嘗試使用兩層迴圈印出如下的九九乘法表。



2、請寫一程式，嘗試使用三層迴圈印出如下的九九乘法表。



6-3 while

上一節的 for 是用於程式設計階段已知迴圈次數，但有些情況，我們於程式設計階段並不知迴圈的執行次數，此時即可使用 while 指令，且有些迴圈可能一次都不執行，所以 while 指令又分為前測試迴圈與後測試迴圈。while 的前測試迴圈語法如下圖左，後測試迴圈如下圖右。

while(運算式)
{
　　指令區塊；
}
	do
	{
	　指令區塊；
	}while (運算式)；

以上語法說明如下：

1. 不論是前測試或後測試迴圈，均是運算式值為 1(真)時，繼續執行迴圈，運算式為 0(偽)時，離開迴圈。
2. 前測試與後測試迴圈的差別為，前測試迴圈有可能一次均不執行迴圈，但後測試迴圈至少執行一次。
3. 後測試迴圈的 while(運算式)後面要加分號(；)，而前測試迴圈的 while 不用加分號。
4. while 指令區塊內亦適用 break 與 continue，前者為強迫提早離開迴圈，後者可略過部份指令，提早進入條件運算式。

5. 以下程式片段可用前測試迴圈解除法。(前面乘法一開始就知道加幾次，但除法就是不曉得)

```
int a=8,b=3,q=0;
while(a>=b) {
        a-=b;
        q++;
}
printf("quotient = %d\n", q); /* 商數 */
printf("remainder= %d\n", a); /* 餘數 */
```

6. **以下程式片段可用後測試迴圈解除法。**

```
a=8;b=3;q=0;
do {
        a-=b;
        q++;
}while(a>=b);
printf("quotient = %d\n", q); /* 商數 */
printf("remainder= %d\n", a); /* 餘數 */
```

範例 6-3a

若有一級數 s=3+6+9+...，請問加到第 n 項，其和剛好超過 1000。

【執行結果】

G612

【程式列印】

1.本例於設計階段並不知道迴圈執行次數，所以不適用 for。其次，題目給的條件是累加超過 1000，所以可以使用 while (sum <1000) 則繼續執行迴圈，繼續累加。

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *argv[]) {
    int sum=0, n=0;
    while(sum<100) {
        i=i+3;
        n++;
        sum+=i;
    }
    printf("s=3+6+... will over 100, when n= %d i=%d,the sum=%d",n, i,sum);
    return 0;
}
```

自我練習

1、請寫一程式，可以輸入任意個數的整數，當輸入-1 時結束，計算輸入數字的平均。

2、請寫一程式，可以輸入任意個數的整數，並將其從個數數逐一輸出。例如，輸入 25，輸出 5 2。

3、請寫一程式，可以輸入任意個數的整數，並計算其幾位數。例如，輸入 25，輸出 2 位數。

4、於範例 4_3a 中，請增加一個功能，要使用此程式前，要先輸入密碼，密碼為數字 1234，但至多可輸入 3 次，第三次錯了，就結束程式。

以下程式，

範例 6_3b

請寫一個程式，滿足以下條件：

(1) 可以產生 0 至 5 的亂數。

(2) 累加以上亂數。

(3) 輸出此亂數與統計其和。

(4) 若亂數不為 0，則重複（1）～（3），否則輸出其和並結束程式。

執行結果

s615