Ch 3. 流程控制

迴圈結構 : for, while, do while

for 迴圈（for loop）

- for 迴圈 可以讓一個或多個敘述，重複執行一個被指定次數的迴圈，而這個次數是由一個判斷式的成立與否決定的。
- 格式

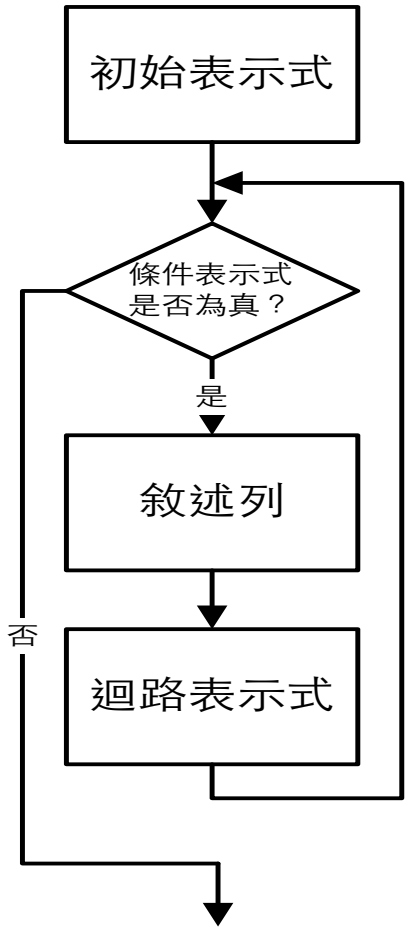
```
for (〔初始表示式〕;〔條件表示式〕;〔迴路表示式〕)
        {
              敘述列;
        }
```

for 迴圈（for loop）　　(cont'd)

- 初始表示式
  - 用來初始化迴圈開始的狀況，也就是迴圈開始之前，會先執行此表示式
- 條件表示式
  - 用來判斷迴圈結束的條件
- 迴路表示式
  - 用來改變條件表示式的條件值，使條件值有變為假（false）的機會，使迴圈停止
- 敘述列
  - for迴圈執行的主體，也就是條件表示式成立時所要執行的敘述

for 迴圈的流程

```
for (〔初始表示式〕;〔條件表示式〕;〔迴路表示式〕) { 敘述列; }
```



Example 3-1　for Loop

```
// [ Code ]: ex3-1.c
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    int i,j;
    for (i=0;i<=9;i++) /* 利用巢狀迴圈
                          產生輸出結果 */
      {
         for (j=0;j<=9;j++)
          {
              // 每一行數字輸出
             printf("%d",(j+i)%10);
          }
         printf("\n");   // 跳下一行
      }
    system("PAUSE");
   return 0;
}
```

[輸出結果]：

```
0123456789
1234567890
2345678901
3456789012
4567890123
5678901234
6789012345
7890123456
8901234567
9012345678
```

Example 3-2 數字跑馬燈

```
// [ Code ]: ex2-9.c  --- 數字跑馬燈程式

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

   int main()
   {
     int i, j, k, jc;

        printf("\n *** << 數字跑馬燈程式 >> *** \n\n");
   //     printf("12345678901234567890123456789O\n\n");
```

Example 3-2 數字跑馬燈 (cont'd)

```
for(i = 0; i <=30; i++)
{
      printf("\r");
      for (k = (30-i); k >= 1; k--)
       {
        printf(" ");
       }

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

      jc = 1;
      while(1)
      {
            if(jc > 100000000) break;
               jc++;    // delay the printout ...
      }
}
```

Example 3-2 數字跑馬燈 (cont'd)

```
printf("\n\n ****** 程式完成! \n\n");
    system ("PAUSE");
    return 0;
}
```

EXERCISE 3.1

(1) 參考 數字跑馬燈程式，設計一個倒數計時器，
在 console 上顯示，從 99 開始倒數至 0。

(2) 參考 數字跑馬燈程式，設計一個數位時鐘，
在 console 上持續顯示正確時間 (hh : mm : ss)。

(3) 利用迴圈，計算下列算式，並輸出至螢幕：
1+2+3+…+100 = ? 以及 10！= ?

while 迴圈

- 當迴圈次數無法量計時，可以使用 while 迴圈
- 格式

```
while ( 條件表示式 )
        {
                敘述列；
        }
```

for vs. while

Q：計算階乘 **10!**

[for loop] :

```
int a=1;
for (i=1;i<=10;i++) a*=i;
```

[while loop] :

```
int a=1, i=1;
while (i<=10)
 { a*=i;
   i++; }
```

Example 3-3　while Loop

Q : 請寫一 C/C++程式,由鍵盤輸入並讀取兩個正整數 a 和 b (其中,a < b);

同時,程式須完成 下列需求:

(A) 求兩個數的最大公因數(greatest common divisor, GCD) ,並輸出其結果於螢幕。

(B) 求兩個數的最小公倍數(least common multiple, LCM) ,並輸出其結果於螢幕。

Example 3-3 while Loop (cont'd)

```
// [ Code ]: ex3-3.c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int main() {
   int x,y,z,i,a,b;
   printf("請輸入兩個數(大數在前,小數在後,二數以逗點隔開):");
   scanf("%d, %d", &x, &y);
   a = y;
   b = x;
   while (y!=0)
    {
        z = x % y;        //將餘數存放在變數z
        x = y;            //除數變被除數
        y = z;            //餘數變除數
    }
   printf("\n最大公因數是:%d", x);
   printf("\n最小公倍數是:%d", a*b/x);
   system("PAUSE");
   return 0;
}
```

do…while

- do…while和while的作用相同，只是判斷的先後不同
- while是在執行迴圈前就先判斷是否要執行
- do…while是先做迴圈內的敘述列，再判斷條件是否符合，決定是否要繼續。

```
do {
        敘述列；
}while ( 條件表示式 );
```

for vs. do…while

Q：計算階乘 **10!**

[for loop] :

```
int a=1;
for (i=1;i<=10;i++) a*=i;
```

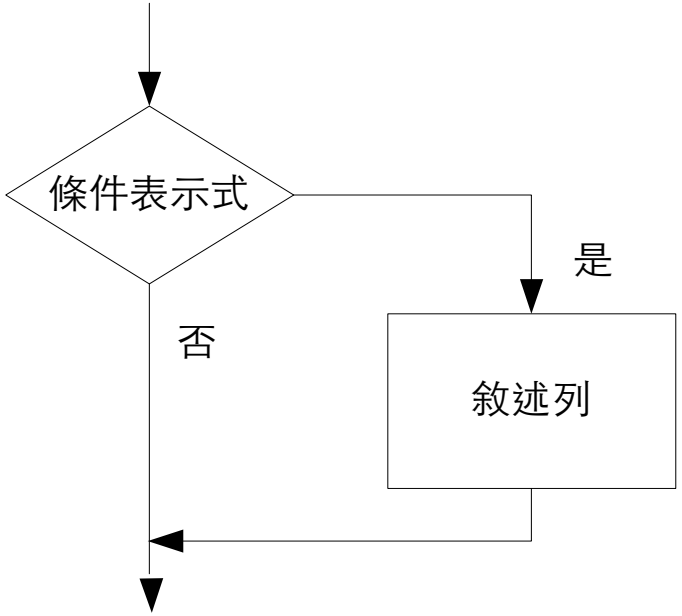
[while loop] :

```
int a=1, i=1;
do{
      a*=i;
      i++;
}while (i<=10);
```

條件流程控制：if, if…else, if…else if

if 條件流程控制

```
/*  當條件表示式成立時，
    執行敘述列內的程式，
    否則繼續執行。  */

if ( 條件表示式 )
   {
       敘述列;
   }
```



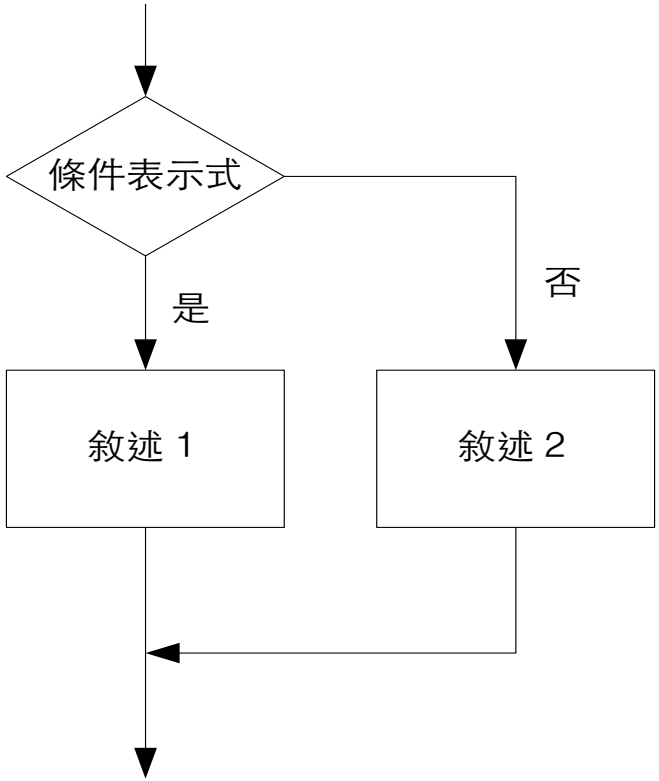
Example 3-4　if condition

找出 1 到 100 之間,不包含 5 或 7 的倍數的整數及其個數,並輸出結果。
(提示: 也就是找出 1, 2, 3, 4, 6, 8, 9, 11, ... , 94, 96, 97, 99)

```
// [ Code ] : ex3-4.c

#include <stdio.h>

#include <stdlib.h>

int main() {

   int s=0,i;
   for (i=1;i<100;i++){
    if (i%5==0 || i%7==0) continue;
    s++;
    if (s%20 == 0) printf(" %d \n", i);

    printf(" %d", i);}

    printf("\n\n 1 至 100 之間,不是 5 和 7 的倍數共有 %d 個\n\n",s);

    system("PAUSE");
    return 0;

}
```

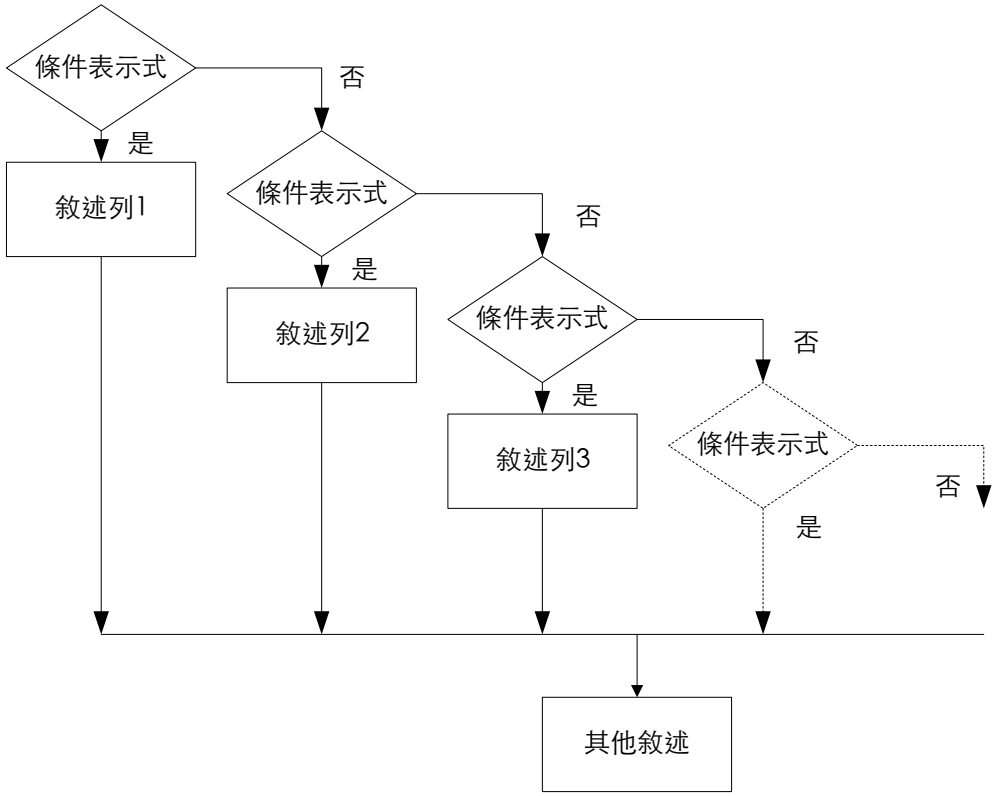
if...else 條件流程控制

```
/* 當條件表示式成立時，
   執行敘述1 程式區段，
   否則執行敘述2 程式區段。 */

if ( 條件表示式 )
{
        敘述1;
}
else
{
        敘述2;
}
```



else if

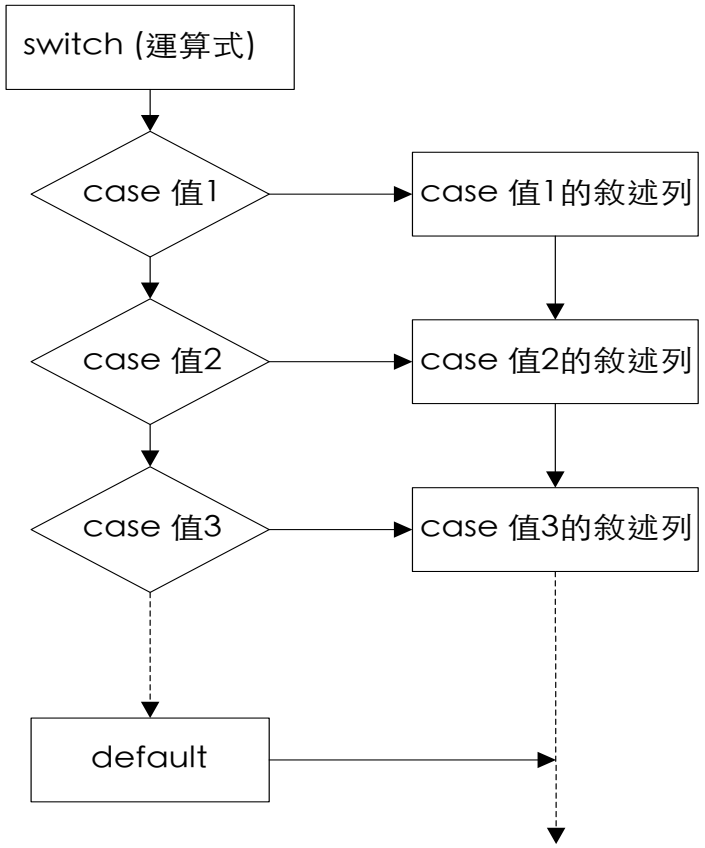
- **else if 敘述其實就是 if…else… 的巢狀判斷敘述，亦即在else之後再接上至少一個 if 敘述 或 if…else… 敘述。**



多重分支結構（switch…case）

- switch…case可以根據一個運算式的計算結果，讓程式去執行不同的程式碼

```
switch(運算式)
{
 case 值1:      敘述列;
                ..............
                break;
 case 值2:      敘述列;
                ..............
                break;
 case 值3:      敘述列;
                ..............
                break;
 default:       敘述列;
                ..............
}
```



中斷分支 break

- 跳離迴圈或 switch…case 敘述，而且把控制權交由迴圈或 switch 敘述之外的下一列程式時，可以使用 break 敘述來完成

迴圈接續continue

- 和break敘述相反動作的敘述是 continue，break 可以強迫將控制權跳離迴圈，而 continue 則將控制權轉移到迴圈開頭的敘述繼續執行

Q： 下列程式執行後，

(A) 該迴圈會執行幾次?

(B) 該程式執行的輸出結果為何?

```
#include <stdio.h>
int main()
{
 int i;
 for (i=0;i<=100;i++)
 {
       if (i>=4 && i<=25)
       if (i>=30 && i<=35) break;
       if (i>=27 && i<=40) continue;
       printf("\nThe index is %d", i);
}
return 0; }
```

條件運算子（ ？ ： ）

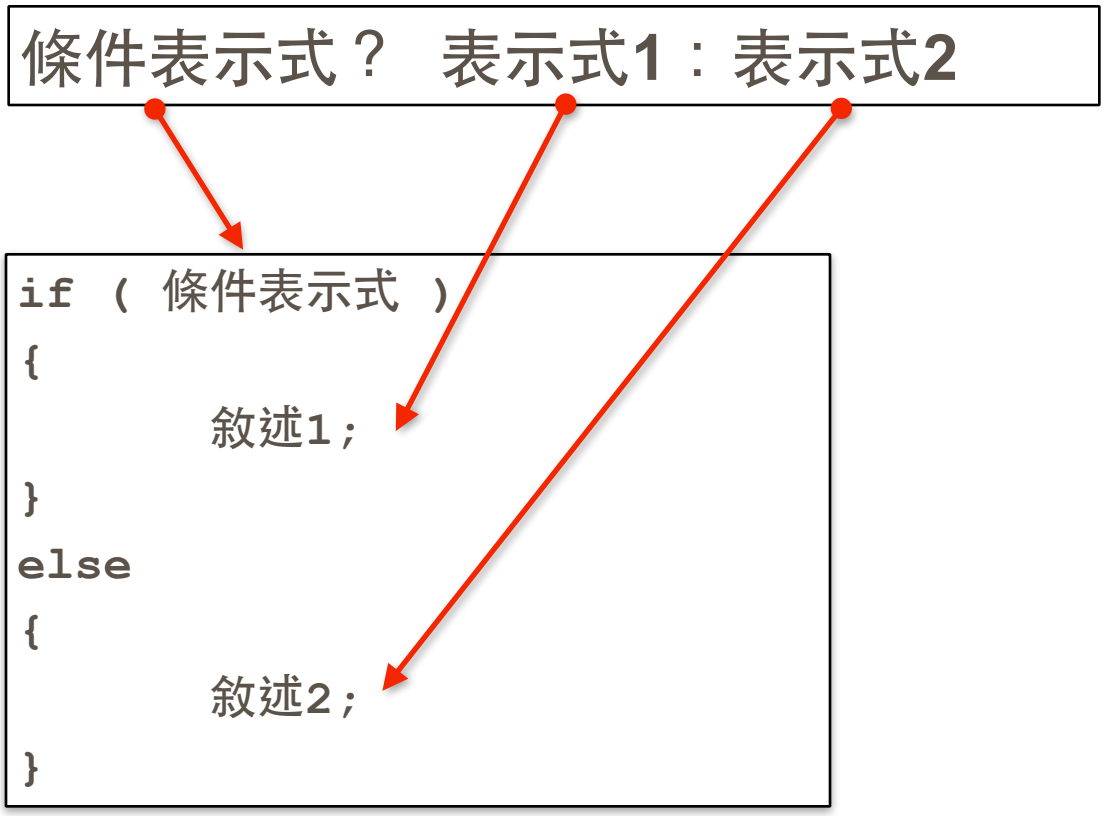
- 「條件表示式」用來判斷要輸出哪一個表示式，當條件表示式運算結果為真truth時，則整個表示式輸出「表示式1」，否則輸出「表示式2」。

條件表示式？ 表示式1：表示式2

例：

```
cout << (grade > 60 ? “pass” : “fail”);
```

(? :) vs. if…else…



跳躍敘述（goto）

- 將程式控制權跳至某一行敘述
- 使用goto敘述須配合設定的標籤，此標籤是一個識別字加上冒號”:”形成識別字標籤，goto敘述便依據此標籤的位置決定跳躍的目的地

```
void main(){
        … … …
 START:
        … … …
        … … …
        … … …
        goto START;
        … … …
}
```

Example 3-5 (switch … case) & (? :)

```
// [ Code ] : ex3-5.c (switch…case),( ? : ), goto

#include <stdio.h>
#include <stdlib.h>
int main() {

   int grade;
 START:
   printf("輸入一個成績介於 0~100:");
   scanf("%d", &grade);

   switch(grade >= 60? 1:0){ // 判斷成績是否及格
    case 0:
        printf("\n 成績不及格！\n\n");
        break;

    case 1:
        printf("\n 成績及格！恭喜！\n\n");
        break;
    }
    goto START; // 回到 START
    system("PAUSE");
    return 0;
}
```