第五章

流程控制

前言

- 最直覺的程式執行方式當然是一行一行的往下執行，這種結構稱之爲循序結構。
- 非循序結構的流程控制是由決策與跳躍組成，而決策與跳躍也可以組合成迴圈。
- 在一般的結構化程式語言中，都不允許（或不建議）使用者自行定義跳躍，而把跳躍移到決策與迴圈之內。
- C語言的流程控制結構有以下三種：
  - 1. 循序結構(Sequence Structure)
  - 2. 選擇結構(Selection Structure)
  - 3. 迴圈結構(Loop Structure)

前言

- 選擇結構可以讓程式設計師依據不同的狀況，選擇不同的對應策略，例如：今天下雨，則開車上班；若沒下雨，則騎機車上班。
- 迴圈結構則可以重複不停的做某些動作直到某個條件成立時，動作才會停止。
- 選擇結構與迴圈結構提供了條件判斷的能力，這使得程式設計師可以依照自己的意願來控制程式的執行流程，也就是所謂的『流程控制』。
- 在C語言中，我們可以將流程控制再細分爲
  - 條件判斷控制
  - 迴圈控制
  - 強制跳躍

大綱

- 5.1 結構化程式語言與C程式設計
  - 5.1.1 『循序』結構
  - 5.1.2 『選擇』結構
  - 5.1.3 『迴圈』結構
  - 5.1.4 『複合敘述』解析
- 5.2 『循序』結構
- 5.3 『選擇』敘述
  - 5.3.1 單一選擇敘述（if敘述）
  - 5.3.2 雙向選擇敘述（if-else敘述）
  - 5.3.3 e1？e2：e3特殊選擇運算式

大綱

- 5.3.4 巢狀式選擇敘述
- 5.3.5 else-if格式
- 5.3.6 if與else的配對
- 5.3.7 多向選擇敘述（switch-case敘述）

- 5.4『迴圈』敘述
  - 5.4.1 計數迴圈（for迴圈敘述）
  - 5.4.2 前測式條件迴圈（while迴圈敘述）
  - 5.4.3 後測式條件迴圈（do-while迴圈敘述）
  - 5.4.4 迴圈的適用狀況
- 5.5 強制跳躍（goto敘述）
- 5.6 巢狀與縮排
- 5.7 本章回顧

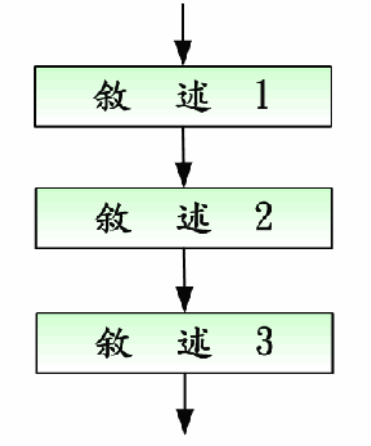
5.1 結構化程式語言與C程式設計

- C語言即爲一種結構化程式語言，結構化程式語言最大的特色之一爲不允許使用『無條件式跳躍指令』（Goto）。除此之外，更詳細地來說，一個程式語言足以稱爲結構化程式語言，必須具有下列特性：
  - 1. 只允許使用三種基本的邏輯結構：循序、選擇和迴圈。（不允許或不建議使用Goto）
  - 2. 使用由上而下（Top-down）的程式設計技巧。
  - 3. 具模組獨立性。
- 我們將首先簡介結構化程式設計的三項基本流程控制結構：循序、選擇和迴圈。
  - 【Goto敘述】：
    - Goto＝＝Go Home。

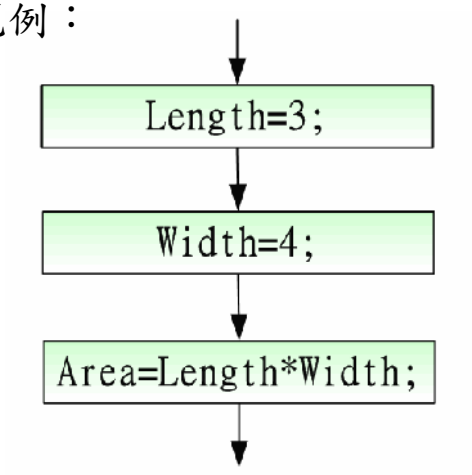
5.1.1 『循序』結構

- 『循序』結構非常直覺而簡單，也就是『程式碼被執行的順序爲由上而下，一個敘述接著一個敘述依序執行』。

結構：



範例：



5.1.2 『選擇』結構

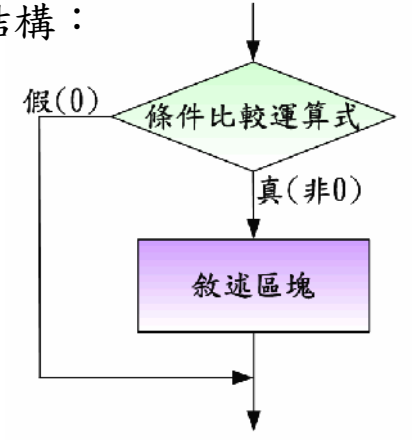
- 『選擇』結構則是代表程式在執行時，會依據條件（運算式的結果）適當地改變程式執行的順序。當滿足條件時，會執行某一敘述區塊（通常是接續的敘述區塊），若條件不滿足時，則執行另一敘述區塊。

  - 一般來說，選擇結構可以分爲三種：
    - 單一選擇
    - 雙向選擇
    - 多向選擇

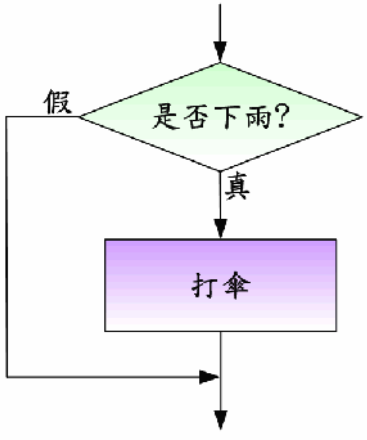
5.1.2 『選擇』結構

- 單一選擇結構：
  - 單一選擇結構只能註明條件成立時，要執行的敘述區塊。當條件不成立時將不會執行該區塊內的敘述，並且也不會執行任何敘述而逕自前往執行選擇結構之後的敘述。而當條件成立時，會先執行區塊內的敘述然後才前往執行選擇結構之後的敘述。

單一選擇結構：



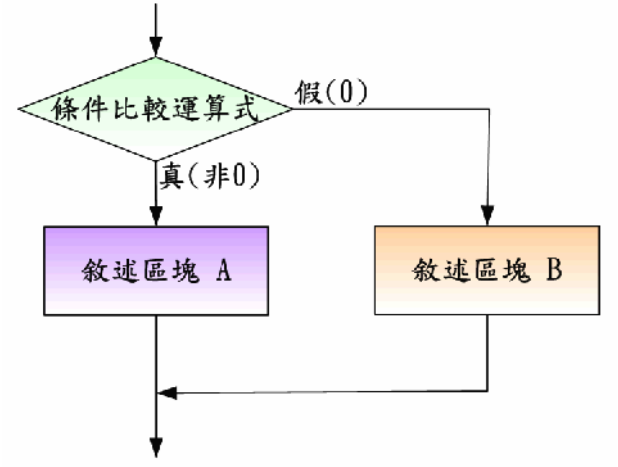
範例：



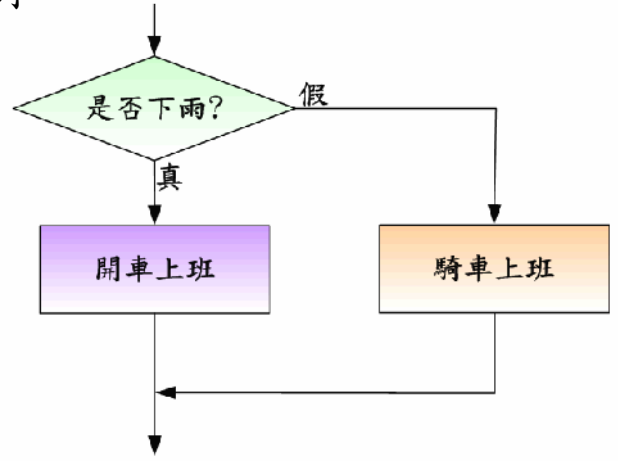
5.1.2 『選擇』結構

- 雙向選擇結構：
  - 雙向選擇結構則能夠指定條件成立時要執行的敘述區塊，也能指定條件不成立時要執行的敘述區塊。敘述區塊執行完畢將會繼續執行選擇結構之後的敘述。

雙向選擇結構：

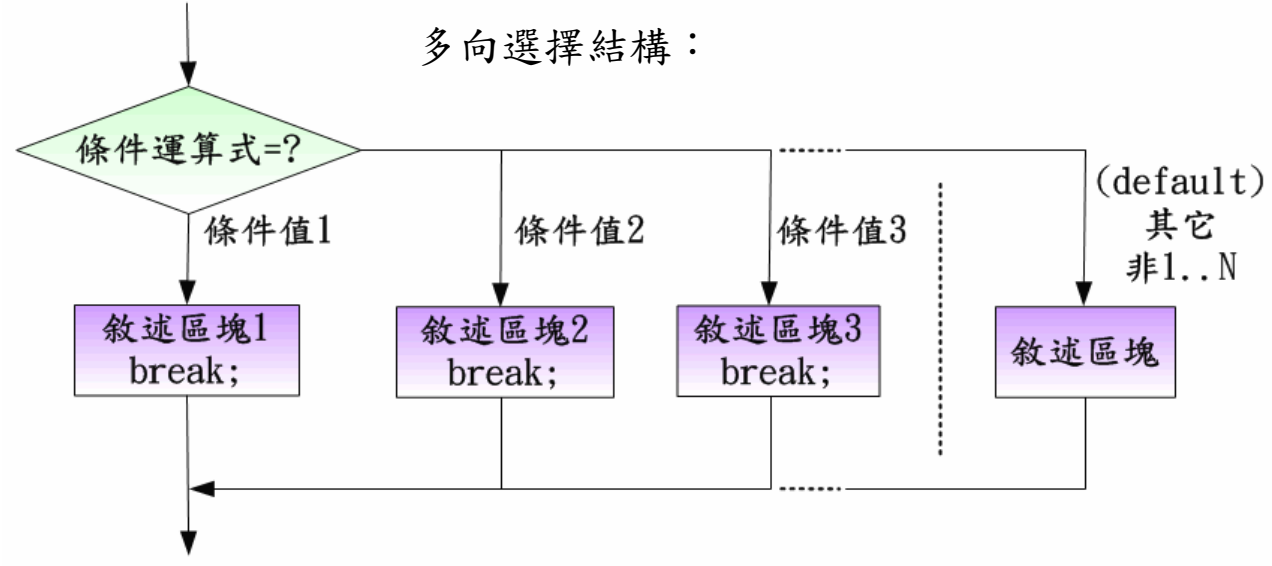


範例：



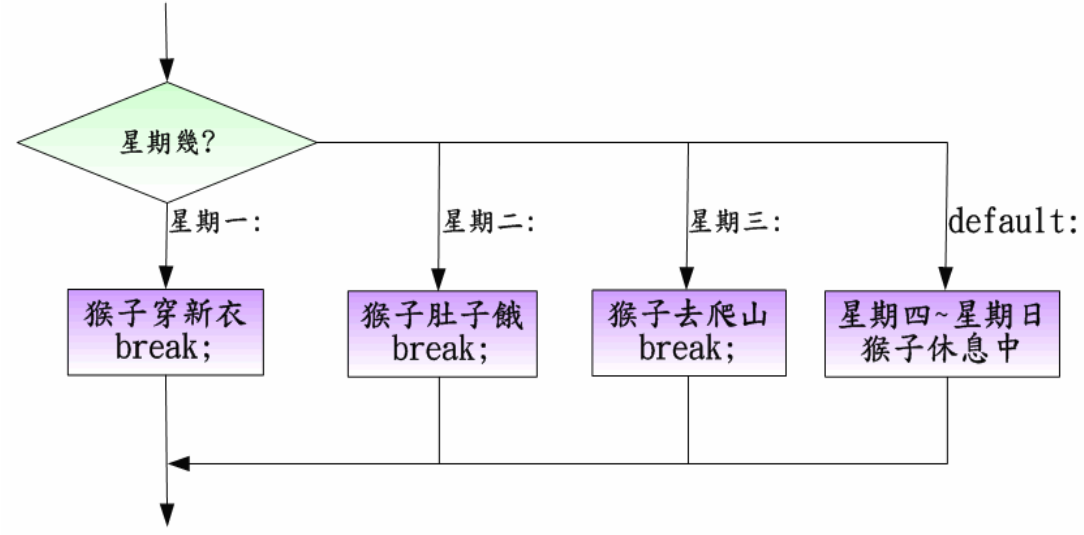
5.1.2 『選擇』結構

- 多向選擇結構：
  - 多向選擇結構可以設定條件值爲各類狀況時要執行的敘述區塊，甚至還可以指定不符合所列之各類狀況時要執行的區塊。敘述區塊執行完畢將會繼續執行選擇結構之後的敘述。



5.1.2 『選擇』結構

範例：


星期幾?
星期一:
星期二:
星期三:
default:
猴子穿新衣
break;
猴子肚子餓
break;
猴子去爬山
break;
星期四~星期日
猴子休息中

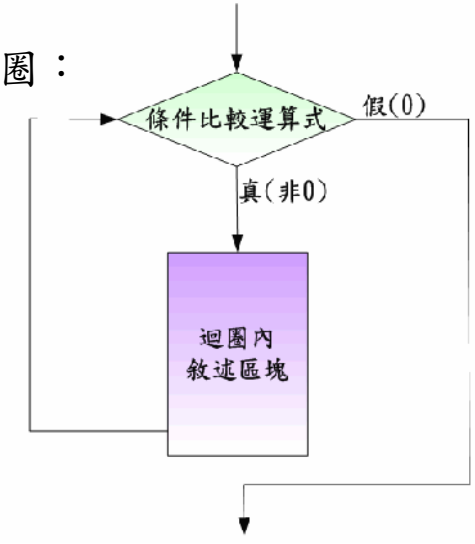
5.1.3 『迴圈』結構

- 『迴圈』結構代表電腦會重複執行某一段敘述區塊，直到某個條件成立或不成立時，重覆動作才會停止。
- 迴圈結構可以分爲：
  - 計數迴圈
  - 前測式迴圈
  - 後測式迴圈三種
- 計數迴圈：
  - 計數迴圈具有『自動執行指定的迴圈初始運算式』、『自動檢測迴圈終止條件運算式』以及『自動執行迴圈增量變化運算式』的能力，在C語言中，for迴圈即爲計數迴圈。

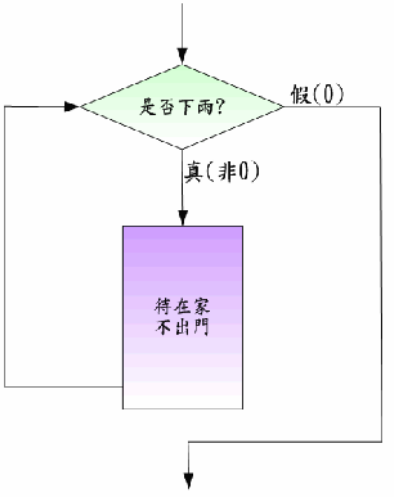
5.1.3 『迴圈』結構

- 前測式迴圈：
  - 先測試條件，若條件爲真，才執行敘述區塊，當敘述區塊執行完畢後，會再回到測試條件重新測試條件，若條件仍舊成立，則再一次執行敘述區塊。如此反覆，直到條件不成立時才會離開迴圈。

前測式迴圈：



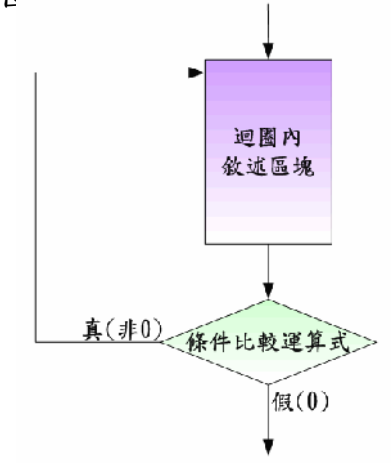
範例：



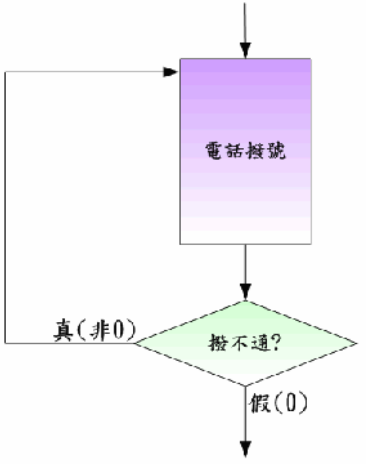
5.1.3 『迴圈』結構

- 後測式迴圈：
  - 先執行迴圈內的敘述區塊一次，然後再測試條件，因此，迴圈內的敘述區塊至少會被執行一次。至於實際被執行多少次，則必須視條件值而定。

後測式迴圈：



範例：



5.1.4 『複合敘述』解析

- 我們在2.5節中，曾經提及過以下六種C語言的常見敘述。
  - 算式敘述 (expression-statement)
  - 複合敘述 (compound-statement)
  - 選擇敘述 (selection-statement)
  - 迴圈敘述 (iteration-statement)
  - 標籤敘述 (labeled-statement)
  - 跳離敘述 (jump-statement)
- 複合敘述 (compound-statement)代表一個敘述區塊它可以將多個定義敘述（例如：int a;）及運算式敘述（例如：a=a*2;）合併爲一個敘述，使用的符號爲{}。

5.2『循序』結構

- 循序結構非常簡單，會依序執行各個敘述，我們以範例來做說明。
- 範例5-1：ch5_01.c。

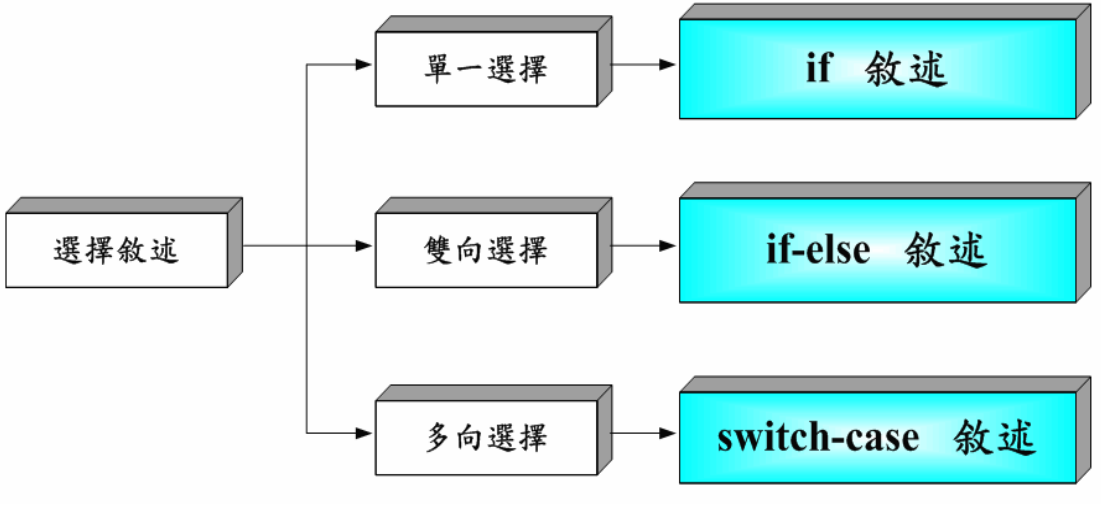
```
1  /*****************************
2      檔名:ch5_01.c
3      功能:循序結構
4   *****************************/
5  #include <stdio.h>
6  #include <stdlib.h>
7   void main(void)
8  {
9    int x;
10   x=1;
11   printf("%d\n",x);
12   x=x+1;
13   printf("%d\n",x);
14   /*  system("pause");  */
15 }
```

- 執行結果：

```
1
2
```

5.3 『選擇』敘述

- C語言的選擇敘述分爲
  - 單一選擇
  - 雙向選擇
  - 多向選擇



C語言的選擇敘述種類

5.3.1 單一選擇敘述（if敘述）

- if敘述很簡單，就是當某個條件運算式成立時，就去做某件事（或某些事），當條件運算式不成立時，就不會做這些事，下面是一個生活實例。

```
if(下雨)
        打傘;
```

  - 在C語言中，寫上述的實例程式也是同樣的道理，只不過將『下雨』使用條件運算式來表達。把『打傘』用敘述來表達。例如：if(A>1)　B=100，就是當A>1時，將B的變數值指定爲100。

- 如果要做的事不只一件的時候，我們又該如何撰寫呢？以下是一個生活實例。

```
if(感冒了)
{
    多喝水;
    多休息;
}
```

5.3.1 單一選擇敘述（if敘述）

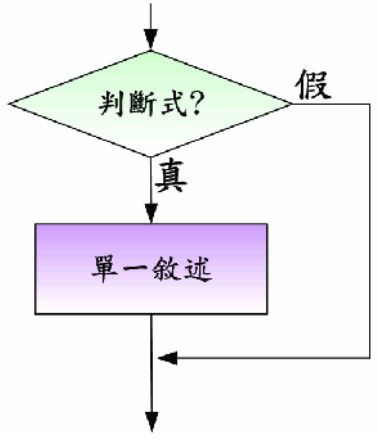
- if敘述標準語法

```
if (expression) statement
```

  - 上述語法我們已經在前一節中已經說明過了，爲了讓初學程式語言的讀者能夠更快上手，我們將if敘述之語法改寫如下。

- if敘述語法一：

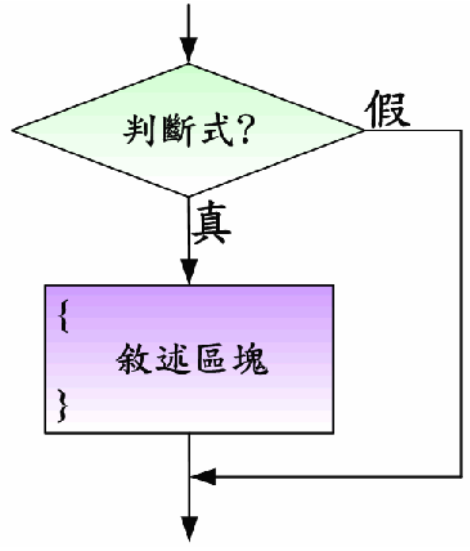
```
if(條件運算式)
    要執行的單一敘述
```



5.3.1　單一選擇敘述（if敘述）

- if敘述語法二：

```
if(條件運算式)
{
   .....敘述區塊.....
}
```



  - 【註】：爲了與一般的運算式區別，有時我們也會將條件運算式稱爲判斷式。

5.3.1　單一選擇敘述（if敘述）

- 範例5-2：ch5_02.c

```
1  /*****************************
2      檔名:ch5_02.c
3      功能:if敘述的練習
4   *****************************/
5  #include <stdio.h>
6  #include <stdlib.h>
7  void main(void)
8  {
9   int x;
10  char *str1;
11  str1="您輸入的是正數或0";
12  printf("請輸入一個整數:");
13  scanf("%d",&x);
14  if(x<0)
15        str1="您輸入的是負數";
16  printf("%s\n",str1);
17  /*  system("pause");  */
18 }
```

5.3.1　單一選擇敘述（if敘述）

- 執行結果01：

```
請輸入一個整數:10
您輸入的是正數或0
```

- 執行結果02：

```
請輸入一個整數:-50
您輸入的是負數
```

5.3.1　單一選擇敘述（if敘述）

- 自我練習：
  - 設計是否有超過「標準體重」的程式：
  - 輸入「身高」和「目前體重 」
  - 公式
    - 男性：（身高cm－80）x70% ＝標準體重
      女性：（身高cm－70）x60% ＝標準體重

自我練習：

```
1   /*****************************
2   自我練習
3   是否有超過「標準體重」的程式
4   *****************************/
5   #include <stdio.h>
6   #include <stdlib.h>
7   void main(void)
8   {
9       int x;
10      char *str1;
11      float fltHeight=0.0f,fltWeight=0.0f;
12      float fltstandard=0.0f;
13      int intsex=0;
14      //*************************************
15      printf("請輸入性別 (男生:1 女生:0) ");
```

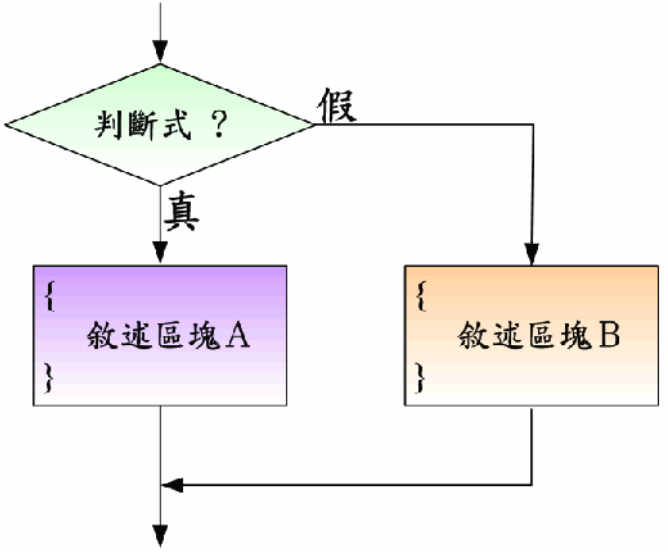
自我練習：

```
16      scanf("%d",&intsex);
17      printf("請輸入個人身高:");
18      scanf("%f",&fltHeight);
19      printf("請輸入個人體重:");
20      scanf("%f",&fltWeight);
21      //**********************************
22      fltstandard=(fltHeight-70)*0.6;
23      if(intsex==1)
24              fltstandard=(fltHeight-80)*0.7;
25      //**********************************
26      str1="你的體重已經超標";
27      if(fltstandard >= fltWeight)
28              str1="恭喜你的體重還在範圍內";
29      //**********************************
30      printf("%s\n",str1);
31      system("pause");
32  }
```

5.3.2 雙向選擇敘述（if-else敘述）

- 使用if敘述無法在『當條件運算式不成立』時，指定所執行的動作。而if-else敘述就可以在『條件運算式不成立』的狀況下，執行某些指定的程式碼，其語法如下。
- if-else敘述語法：

```
if(條件運算式)
{
   條件成立時要執行的敘述區塊A
}
else
{
   條件不成立時所執行的敘述區塊B
}
```



5.3.2 雙向選擇敘述（if-else敘述）

- 【實用範例5-5】：根據購買入場卷的數量是否大於10張，決定是否打折優待，ch5_05.c。

```
1   /*****************************
2       檔名:ch5_05.c
3       功能:if-else敘述的練習
4    *****************************/
5
6   #include <stdio.h>
7   #include <stdlib.h>
8
9   void main(void)
10  {
11   int OnePrice,Qty;
12   float TotalPrice;
13  OnePrice=200;
```

5.3.2 雙向選擇敘述（if-else敘述）

```
14    printf("每張入場卷價格爲%d元\n",OnePrice);
15    printf("請輸入您要購買的張數:");
16    scanf("%d",&Qty);
17    printf("=========================\n");
18    if(Qty>=10)
19    {
20        TotalPrice=OnePrice*Qty*0.9;
21        printf("購買10張以上打九折\n");
22    }
23    else
24    {
25        TotalPrice=OnePrice*Qty;
26        printf("您爲購買10張以上的入場券,恕不打折\n");
27    }
28    printf("總價爲%.0f元\n",TotalPrice);
29    /*  system("pause");  */
30  }
```

5.3.2 雙向選擇敘述（if-else敘述）

- 執行結果：

```
每張入場卷價格爲200元
請輸入您要購買的張數:15
============================
購買10張以上打九折
總價爲2700元
```

```
每張入場卷價格爲200元
請輸入您要購買的張數:5
============================
您爲購買10張以上的入場券,恕不打折
總價爲1000元
```

- 範例說明：
  - 在執行結果中，當使用者輸入『15』時，判斷式『Qty>=10』成立，因此第20~21行會被執行（第25~26行不會被執行）。當使用者輸入『5』時，判斷式『Qty>=10』不成立，因此第25~26行會被執行（第20~21行不會被執行）。而第28行是一定會被執行的，因爲它並非if-else敘述的一部份。

5.3.3 e1？e2：e3特殊選擇運算式

- 在前面章節介紹運算子時，我們曾介紹過條件運算符號「?:」，這個運算符號可以用來替代簡單的if-else敘述，其語法如下：

語法：(條件運算式1) ？(運算式2)：(運算式3)；
功能：依照條件運算式1的成立與否，分別執行運算式2或運算式3。

  - 【語法說明】：
    - (1)條件運算式1爲真時，執行運算式2。條件運算式1爲假時，執行運算式3。
    - (2)條件運算式1、運算式2、運算式3皆不含分號結尾。但整個敘述的最後必須含分號結尾。
- 【實用範例5-6】：設計一個猜數字遊戲，由使用者輸入的猜測數字，予與回覆是否爲正確

5.3.3 e1？e2：e3特殊選擇運算式

```
1  /*************************
2      檔名:ch5_06.c
3      功能:?條件運算子的練習
4   *************************/
5
6  #include <stdio.h>
7  #include <stdlib.h>
8
9  void main(void)
10 {
11  int Ans=38; /* 答案爲38 */
12  int Guess;
13  printf("請猜一個1~99的號碼:");
14  scanf("%d",&Guess);
15  if(Guess==Ans)
16      printf("恭喜您猜到了,獎品是一包乖乖.\n");
17  else
18      (Guess>Ans) ? printf("您猜得太大了\n") : printf("您猜得太小了\n");
19  /*  system("pause");  */
20 }
```

5.3.3 e1？e2：e3特殊選擇運算式

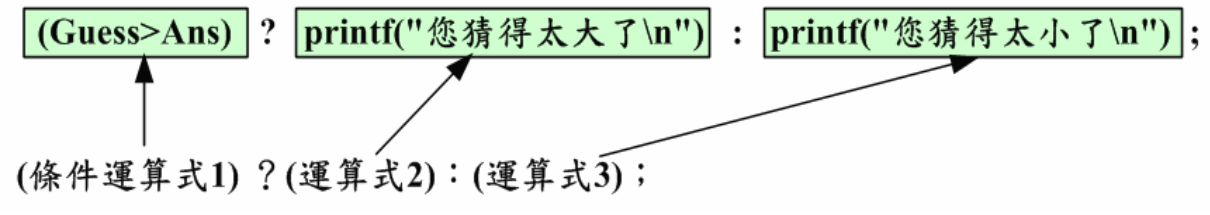
- 執行結果：

```
請猜一個1~99的號碼:38
恭喜您猜到了,獎品是一包乖乖.
```

```
請猜一個1~99的號碼:50
您猜得太大了
```

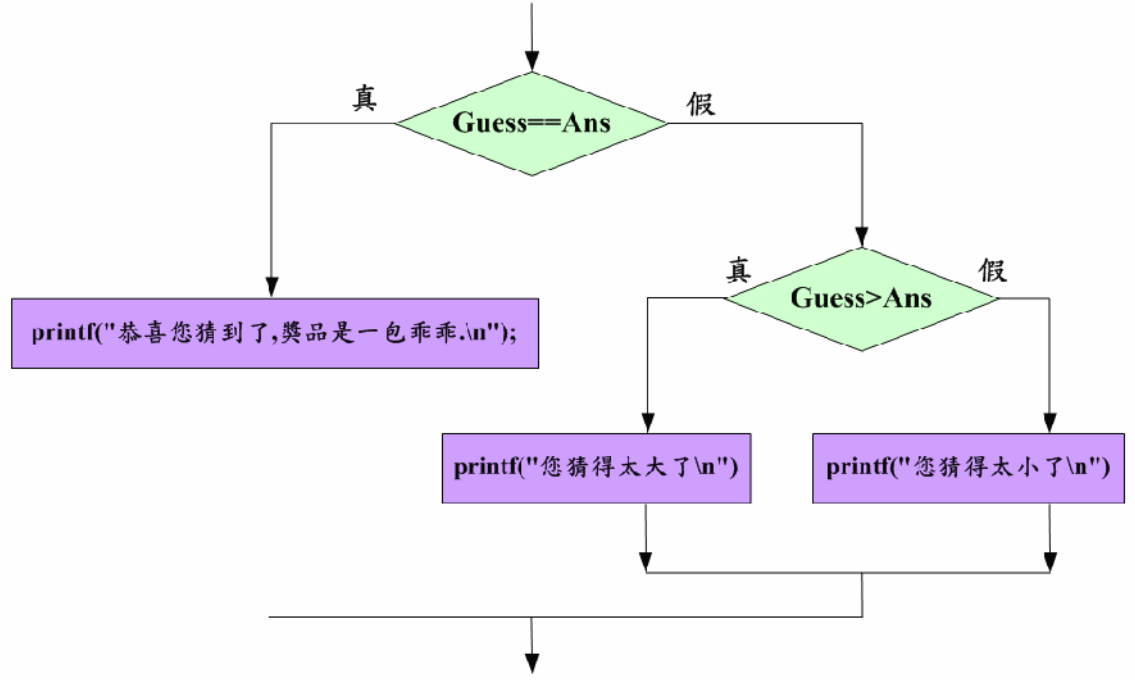
```
請猜一個1~99的號碼:12
您猜得太小了
```

- 範例說明：
  - (1)在執行結果中，當使用者輸入『38』時，第16 行會被執行。在執行結果中，當使用者輸入的數字不是『38』時，第18 行會被執行。
  - (2)第18行對應語法如下：



5.3.3 e1？e2：e3特殊選擇運算式

- (3)第18行會依照(Guess>Ans)是否成立，決定執行『printf("您猜得太大了\n")』（例如輸入50），還是執行『printf("您猜得太小了\n")』（例如輸入12）。所以整個流程如下。



5.3.4 巢狀式選擇敘述

- 選擇敘述的敘述區塊中可以包含任何敘述，因此，我們可以在該區塊中再放入另一個選擇敘述，如此一來就形成了所謂的兩層式『巢狀式選擇』。
  - 巢狀式選擇的應用一般常見於程式判斷需要『兩個以上的選擇條件』時。
  - 巢狀式選擇敘述並未規定外層的選擇敘述必須使用哪一種，換句話說，我們可以把內層的選擇式敘述放在單一選擇（if敘述）的敘述區塊，也可以放在雙向選擇（if-else）的敘述區塊A或else敘述區塊B中。以下是一個巢狀式選擇的格式範例：

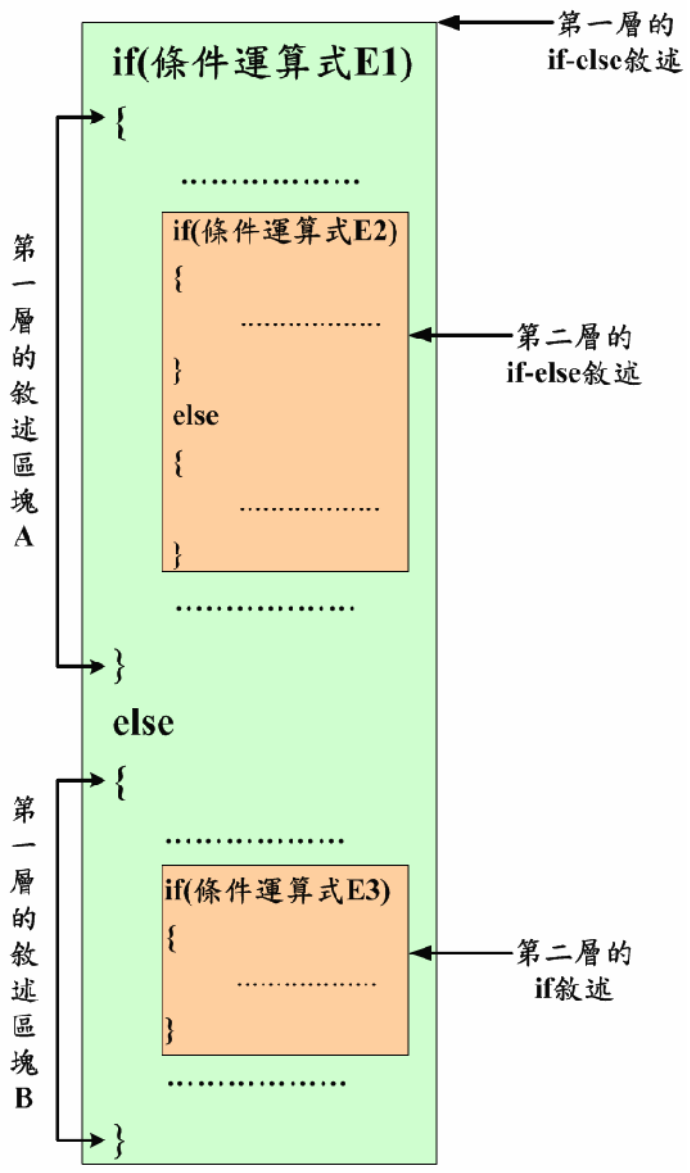


圖5-2 巢狀式選擇格式範例

上述的巢狀式選擇敘述所對應的流程圖如下：

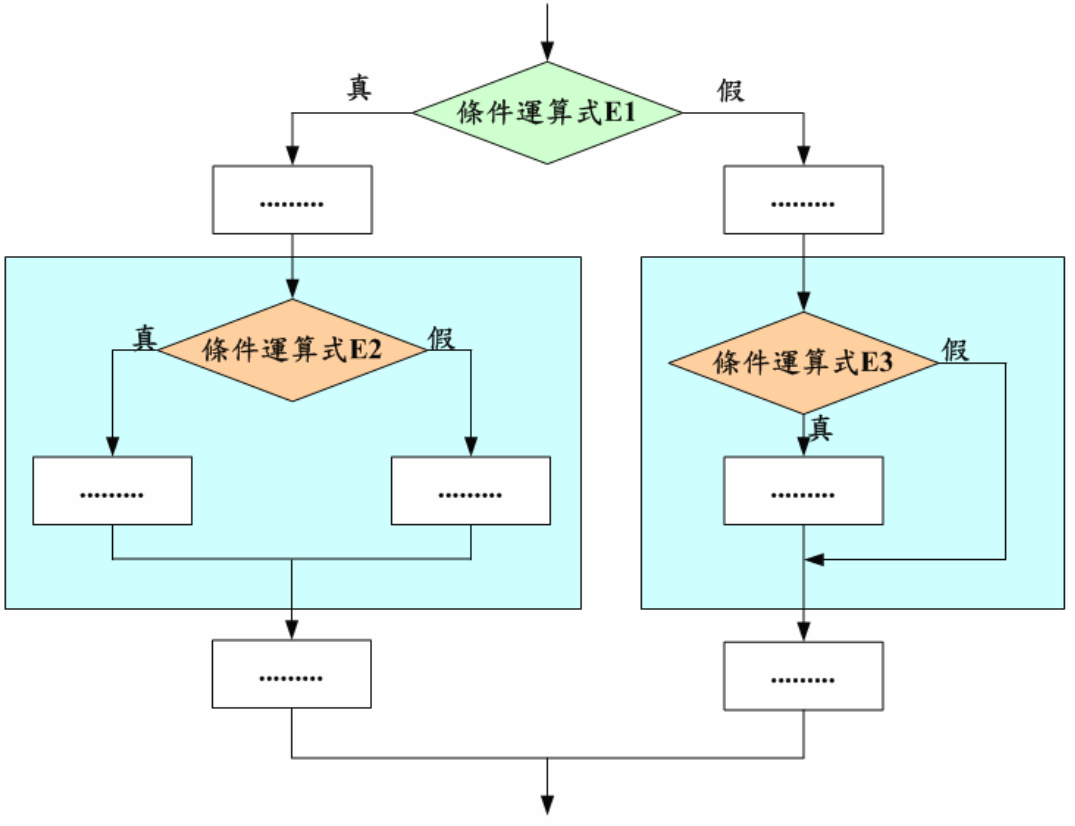


圖5-3 巢狀式選擇範例的對照流程圖

5.3.4 巢狀式選擇敘述

- 【實用範例5-7】：根據輸入的繳款記錄、持卡年份評斷預借現金額度。其公式如下：
  - 繳款記錄：不正常 ==========>無法預借現金
  - 繳款記錄：正常
    - 持卡未滿半年 ==========>無法預借現金
    - 持卡滿半年未滿1年 ==========>預借現金額度爲信用額度之一半
    - 持卡滿1年 ==========>預借現金額度爲全額
- 範例5-7：ch5_07.c

5.3.4 巢狀式選擇敘述

```
/*****************************
    檔名:ch5_07.c
    功能:巢狀選擇的練習
 *****************************/

#include <stdio.h>
#include <stdlib.h>
void main(void)
{
 long Credit;     /*  信用額度 */
 int Status;      /*  繳款狀態 */
 float Year;      /*  持卡年份 */
 printf("請輸入信用額度:");
 scanf("%d",&Credit);
 printf("繳款是否正常(1:正常,0:不正常):");
 scanf("%d",&Status);
```

```
18  if(Status)
19   {
20       printf("請輸入持卡年份:");
21       scanf("%f",&Year);
22       if(Year>=0.5)
23       {
24            if(Year<1)
25            {
26                 printf("預借現金金額爲%.0f元\n",Credit*0.5);
27            }
28            else
29            {
30                 printf("預借現金金額爲%d元\n",Credit);
31            }
32       }
33       else
34       {
35            printf("預借現金金額爲0元\n");
36       }
37   }
38   else
39   {
40       printf("預借現金金額爲0元\n");
41   }
42   /*  system("pause");  */
43  }
```

5.3.4 巢狀式選擇敘述

- 執行結果：

請輸入信用額度:**80000**
繳款是否正常(1:正常,0:不正常):**0**
預借現金金額爲0元

請輸入信用額度:**80000**
繳款是否正常(1:正常,0:不正常):**1**
請輸入持卡年份:**0.3**
預借現金金額爲0元

請輸入信用額度:**80000**
繳款是否正常(1:正常,0:不正常):**1**
請輸入持卡年份:**0.8**
預借現金金額爲40000元

請輸入信用額度:**80000**
繳款是否正常(1:正常,0:不正常):**1**
請輸入持卡年份:**2**
預借現金金額爲80000元

- 範例說明：
  - (1)第18~41行：最外層if-else敘述，若條件(Status)成立（非0值），則執行第19~37行敘述區塊。若不成立則執行第39~41行。也就是符合題目之繳款不正常。
  - (2)第22~36行：第二層if-else敘述，若條件(Year>=0.5)成立，則執行第23~32行敘述區塊。若不成立則執行第34~36行。也就是符合題目之持卡未滿半年。
  - (3)第24~31行：第三層if-else敘述，若條件(Year<1)成立，則執行第25~27行敘述，也就是符合題目之持卡滿半年但未滿1年。若不成立則執行第29~31行。也就是符合題目之持卡滿1年。

5.3.4 巢狀式選擇敘述

- 【實用範例5-8】：使用單一層的if或if-else敘述設計具有範例5-7效果的程式。
- 範例5-8：ch5_08.c。

```
1  /*****************************
2      檔名:ch5_08.c
3      功能:條件運算式的練習
4   *****************************/
5
6  #include <stdio.h>
7  #include <stdlib.h>
8
9  void main(void)
10 {
11  long Credit;     /*  信用額度 */
12  int Status;      /*  繳款狀態 */
13  float Year;      /*  持卡年份 */
14  printf("請輸入信用額度:");
15  scanf("%d",&Credit);
16  printf("繳款是否正常(1:正常,0:不正常):");
17  scanf("%d",&Status);
```

5.3.4 巢狀式選擇敘述

```
18   if(!(Status))
19   {
20        printf("預借現金金額爲0元\n");
21   }
22   else
23   {
24        printf("請輸入持卡年份:");
25        scanf("%f",&Year);
26   }
27   if((Status) && (Year>=0.5) && (Year<1))
28        printf("預借現金金額爲%.0f元\n",Credit*0.5);
29   if((Status) && (Year>=1))
30        printf("預借現金金額爲%d元\n",Credit);
31   if((Status) && (Year<0.5))
32        printf("預借現金金額爲0元\n");
33   /*  system("pause");  */
34 }
```

- 執行結果：（同範例5-7）

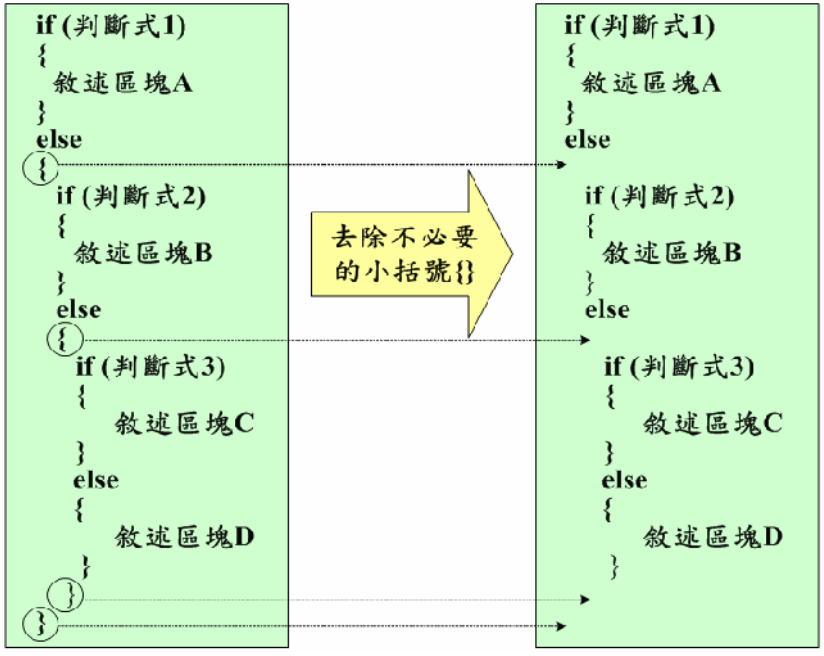
5.3.4 巢狀式選擇敘述

- 範例說明：
  - 由本程式可以看出，不使用巢狀式選擇敘述也可以解決相同的問題，不過您可能必須思考更多關於條件運算式涵蓋的範圍，才不至於出錯。

- 【程式設計原則】：
  - 範例5-7、5-8解決了相同的問題，讀者可以選擇任何一種方法或者自行設計程式，但最好掌握下列原則：
    - 1. 正確並符合題意的執行結果。（這是絕對需要遵守的條件）
    - 2. 容易維護的設計方法（例如：易懂、適合除錯、具有擴充性）。
    - 3. 縮短程式發展時程。
    - 4. 考慮程式所需要的記憶體空間及執行效率。

5.3.5 else-if格式

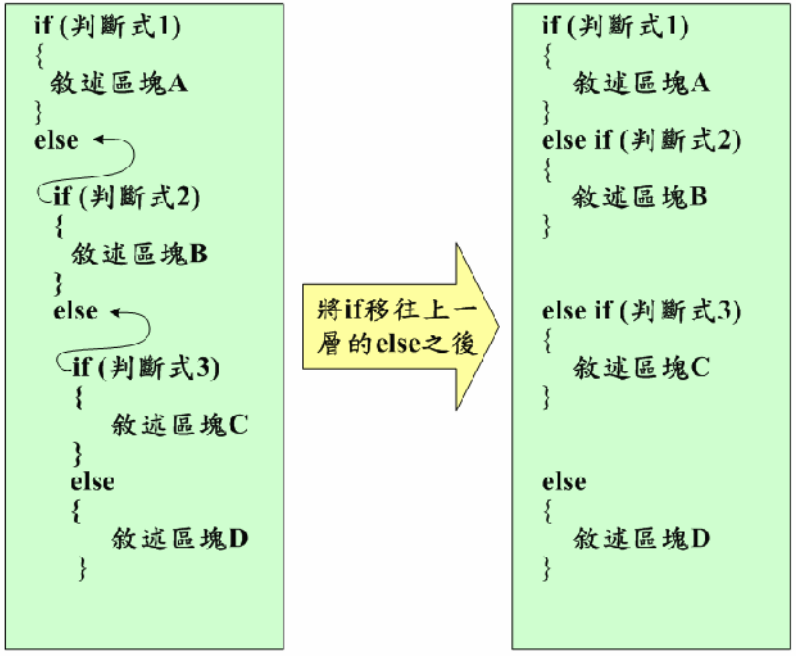
- else-if並非C語言的敘述，它只不過是將if-else敘述重新排列而已，如下圖所示。



if-else轉換為else-if格式之步驟一

5.3.5 else-if格式

- 當我們進一步改變格式後，將變成else-if格式如下圖。



if-else轉換為else-if格式之步驟二

5.3.5 else-if格式

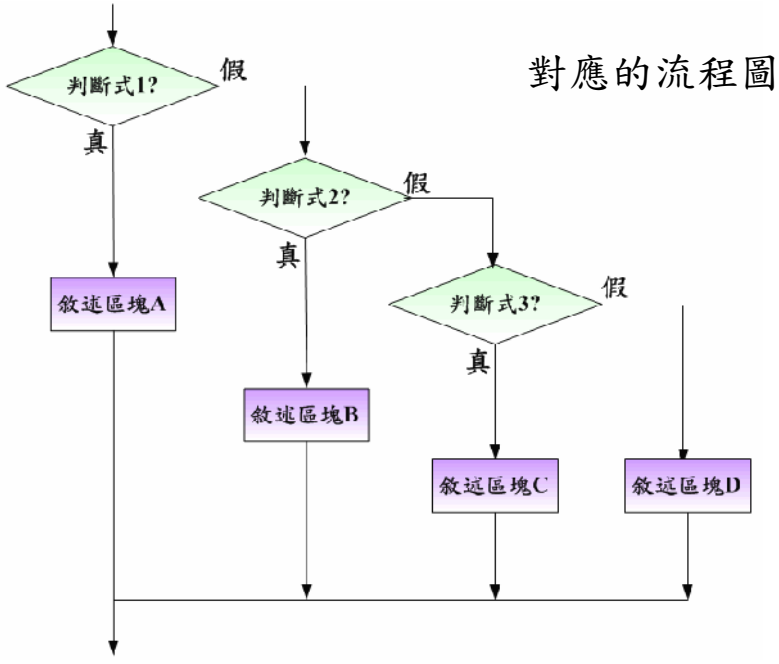
else-if格式一：

```
if(條件運算式1)
{
   敘述區塊A
}
else if(條件運算式2)
{
   敘述區塊B
}
else if(條件運算式3)
{
   敘述區塊C
}
else
{
   敘述區塊D
}
```

else-if格式二：

```
if(條件運算式1){ 敘述區塊A }
else if(條件運算式2){ 敘述區塊B }
else if(條件運算式3){ 敘述區塊C }
else { 敘述區塊D }
```

對應的流程圖：



5.3.5 else-if格式

- 【實用範例5-9】：使用else-if格式設計一個評定分數等級的程式。
- 範例5-9

```
1  /*****************************
2      檔名:ch5_09.c
3      功能:else-if格式的練習
4   *****************************/
5  #include <stdio.h>
6  #include <stdlib.h>
7  void main(void)
8  {
9   unsigned int Score;
10  printf("請輸入計概成績:");
11  scanf("%d",&Score);
12  if(Score<60) { printf("分數等級爲丁等\n"); }
13  else if(Score<=69) { printf("分數等級爲丙等\n"); }
14  else if(Score<=79) { printf("分數等級爲乙等\n"); }
15  else if(Score<=89) { printf("分數等級爲甲等\n"); }
16  else if(Score<=99) { printf("分數等級爲優等\n"); }
17  else if(Score==100) { printf("完美分數\n"); }
18  else { printf("您輸入了不合法的分數\n"); }
19  /*  system("pause");  */
20 }
```

5.3.5 else-if格式

- 執行結果：

```
請輸入計概成績:99
分數等級爲優等
```

- 範例說明：
  - else-if格式的特點是，每一個敘述區塊的執行條件彼此是互斥的（因爲它是巢狀if-else的else部分）。除了使用else-if來製作上述範例之外，馬上我們也將使用switch-case敘述的多向選擇來製作同樣功能的程式。

5.3.6 if與else的配對

- if與if-else兩種選擇敘述，在一個程式一起出現if關鍵字個數與else關鍵字個數不相對稱的現象時，我們要如何得知哪一個else屬於哪一個if所有呢？其實在C語言中規定，**任何一個else都將與最接近的if配對**。我們以下列範例來加以說明。

5.3.6 if與else的配對

- 【觀念範例5-10】：if與else的配對。
- 範例5-10：ch5_10.c

```
1  /*****************************
2      檔名:ch5_10.c
3      功能:if else的配對練習
4   *****************************/
5  #include <stdio.h>
6  #include <stdlib.h>
7   void main(void)
8  {
9   int Score=75;
10  if(Score>60)
11     if (Score > 80)
12        printf("成績真不錯\n");
13  else
14     printf("成績差強人意\n");
15  /*  system("pause");  */
16 }
```

5.3.6 if與else的配對

- 執行結果：

```
成績差強人意
```

- 範例說明：
  - (1) 第15行的else看起來好像是和第12行的if配對，但實際上並非如此，它是與最接近的if（第13行的if）配對，因此執行結果爲『成績差強人意』。如果我們將第11行的Score設定爲50，則執行結果將不會印出任何字串。
  - (2) 本範例說明了兩件事：(1)C語言是一種自由格式的程式語言，因此縮排與否並不會影響執行結果。(2)C語言的else將與最接近的if配對，因此本範例最好將之改寫如下縮排方式，以免產生錯覺。

5.3.6 if與else的配對

```
11  int Score=75;
12  if(Score>60)
13      if (Score > 80)
14          printf("成績真不錯\n");
15      else
16          printf("成績差強人意\n");
```

- (3) 如果讀者希望第16行的『printf("成績差強人意\n");』改爲在Score<=60的狀況下印出，則應該配合{}，改寫成下列格式。

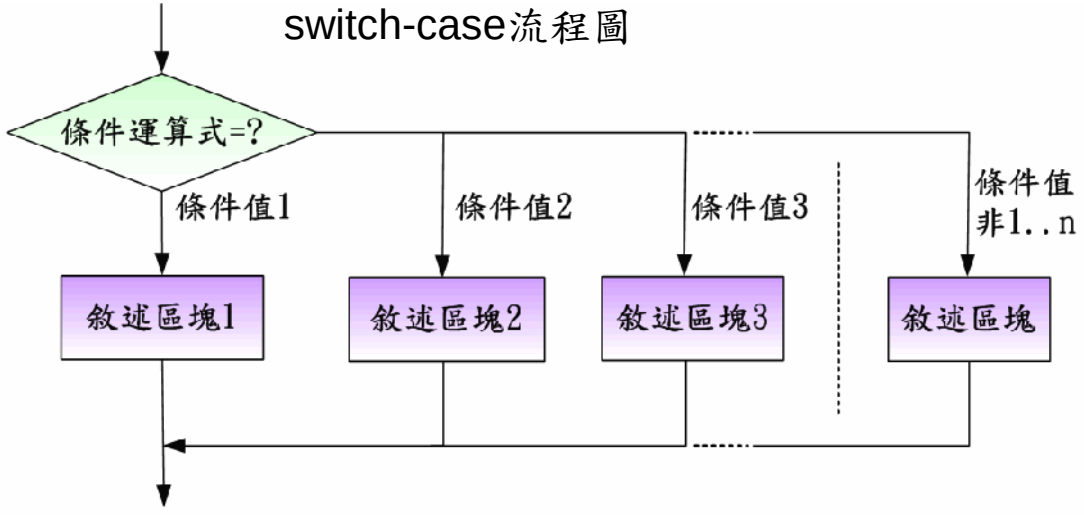
```
11  int Score=75;
12  if(Score>60)
13    {
14      if (Score > 80)
15         printf("成績真不錯\n");
16    }
17  else
18    printf("成績差強人意\n");
```

5.3.7 多向選擇敘述（switch-case敘述）

- 除了單一選擇與雙向選擇之外，C語言還提供了switch-case多向選擇敘述。
- 當狀況不只一個的時候，我們還可以使用switch-case多向選擇敘述，使用switch-case來做選擇決策，其語法如下所示。

```
switch(條件運算式)
{
case條件值1：
        …敘述區塊1…
break;
case條件值2：
        …敘述區塊2…
break;
        :
        :
case 條件值n：
        …敘述區塊n…
break;
default：
        ….敘述區塊…
        break;
}
```

switch-case流程圖



switch-case語法：

5.3.7 多向選擇敘述（switch-case敘述）

- 【語法說明】：
  - switch的每一個case跟隨著一個值，當中的敘述區塊代表當條件運算式=條件值時，所要執行的敘述區塊。
  - 而default之後不跟隨條件值，其中的敘述區塊則是代表當條件運算式不等於任何一個條件值時所要執行的敘述區塊。
  - default可有可無，若沒有default，則當條件運算式不等於任何一個條件值的時候，整個switch-case敘述就不會執行任何的程式碼敘述。
  - case的敘述區塊可以是空敘述。

5.3.7 多向選擇敘述（switch-case敘述）

- 條件值：條件值必須是資料常數（不可以是變數），例如整數或字元。並且每一個case的條件值必須不同。
  - 【範例】：case 1: ：測試值是否爲1。
  - 【範例】：case 'X':：測試值是否爲大寫字元『X』。
- break敘述：如果要符合上述的流程圖，則break敘述不可省略。break的功能是作爲跳離內部迴圈的跳耀敘述（jump statement）。
- default的break敘述：事實上，switch-case敘述的case與default順序並不重要，您可以隨意更動case的順序，甚至是將default放在case之前或之間也不會影響其正確性default內的break敘述則不可省略。

5.3.7 多向選擇敘述（switch-case敘述）

- 【實用範例5-11】：使用switch-case敘述設計一個評定分數等級的程式。

5.3.7 多向選擇敘述（switch-case敘述）

- 範例5-11：ch5_11.c

```
1  /*****************************
2      檔名:ch5_11.c
3      功能:switch-case的練習
4   *****************************/
5
6  #include <stdio.h>
7  #include <stdlib.h>
8
9  void main(void)
10 {
11  int Score;
12  printf("請輸入計概成績:");
13  scanf("%d",&Score);
14  if((Score>=0) && (Score<=100))
```

5.3.7 多向選擇敘述（switch-case敘述）

```
15  switch(Score / 10)
16    {
17     case 10:
18          printf("完美分數\n");
19          break;
20     case 9:
21          printf("分數等級爲優等\n");
22          break;
23     case 8:
24          printf("分數等級爲甲等\n");
25          break;
26     case 7:
27          printf("分數等級爲乙等\n");
28          break;
29     case 6:
30          printf("分數等級爲丙等\n");
31          break;
32     default:
33          printf("分數等級爲丁等\n");
34          break;
35    }
36   /*  system("pause");  */
37  }
```

5.3.7 多向選擇敘述（switch-case敘述）

- 執行結果：

```
請輸入計概成績:99
分數等級爲優等
```

```
請輸入計概成績:75
分數等級爲乙等
```

- 範例說明：
  - (1)第14行的if敘述是用來確保輸入的數字在規定的範圍0~100之內，當符合規定範圍後switch-case敘述。
  - (2)因(Score/10)的結果也將是整數（小數部分將被捨棄）。
  - (3)根據（Score/10）的結果，決定要執行哪一個case敘述區塊，若結果非10、9、8、7、6，則執行default敘述區塊。
  - (4)我們在每一個case敘述區塊末端加入了break敘述，所以眾多的case敘述區塊只會被執行其中之一（不會發生falling through現象）。至於本範例default敘述區塊的末端break敘述（第34行）則可以省略，因爲它已經是位於switch-case的最後敘述。

5.3.7 多向選擇敘述（switch-case敘述）

- 【觀念範例5-12】：falling through現象。
- 範例5-12：ch5_12.c

```
1  /*******************************
2      檔名:ch5_12.c
3      功能:falling through的示範
4   *******************************/
5
6  #include <stdio.h>
7  #include <stdlib.h>
8
9  void main(void)
10 {
11  int Score;
12  printf("請輸入計概成績:");
13  scanf("%d",&Score);
14  if((Score>=0) && (Score<=100))
```

5.3.7 多向選擇敘述（switch-case敘述）

```
15   switch(Score / 10)
16    {
17     case 10:
18         printf("完美分數\n");
19     case 9:
20         printf("分數等級爲優等\n");
21     case 8:
22         printf("分數等級爲甲等\n");
23     case 7:
24         printf("分數等級爲乙等\n");
25     case 6:
26         printf("分數等級爲丙等\n");
27     default:
28         printf("分數等級爲丁等\n");
29    }
30   /*  system("pause");  */
31  }
```

5.3.7 多向選擇敘述（switch-case敘述）

- 執行結果：

```
請輸入計概成績:99
分數等級爲優等
分數等級爲甲等
分數等級爲乙等
分數等級爲丙等
分數等級爲丁等
```

```
請輸入計概成績:75
分數等級爲乙等
分數等級爲丙等
分數等級爲丁等
```

- 範例說明：
  - 本範例將break敘述去除，因此產生了falling through現象。從執行結果中，讀者可以發現，當某一個case被滿足後，除了該敘述區塊會被執行之外，其後所有的case與default敘述區塊也會被執行（不論是否符合該case的條件值）。

5.4『迴圈』敘述

- 高階語言另一項重要的發展就是迴圈結構，迴圈結構事實上是結合了低階語言的決策與跳躍，使得程式中可以有某部分敘述區塊能夠被重複執行多次。

- C語言的迴圈結構分爲
  - 『計數迴圈』
  - 『條件式迴圈』
    - 『前測式』
    - 『後測式』

5.4.1 計數迴圈（for迴圈敘述）

- 迴圈是結構化程式語言另一項重要的設計，它可以重複不停的做某些動作直到某個條件成立時，動作才會停止。在C語言中提供了多種的迴圈，我們首先介紹for計數迴圈。

```
Sum=0;
for(count=1;count<=10;count++)
        Sum=Sum+count;
```

  - 在上面的小範例中（Sum與count都需要在事前先宣告），迴圈一共會被執行10次，因此『Sum= Sum+count』也總共會被執行10次。最開始count 值爲1、Sum值爲0，每次重複執行迴圈時，變數count值都會加1，所以當迴圈執行完畢，Sum值就會是1~10的總和55。

5.4.1 計數迴圈（for迴圈敘述）

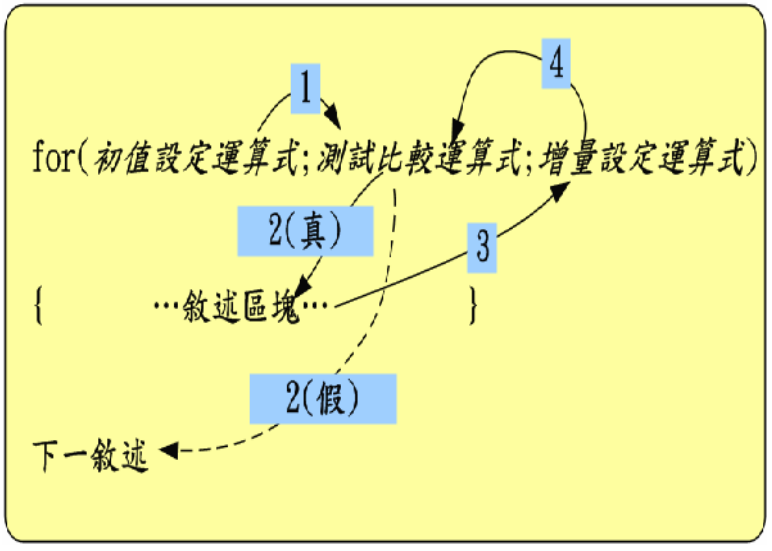
- for迴圈語法

```
for(初值設定運算式;測試比較運算式;增量設定運算式)
{
… 敘述區塊 …
}
```

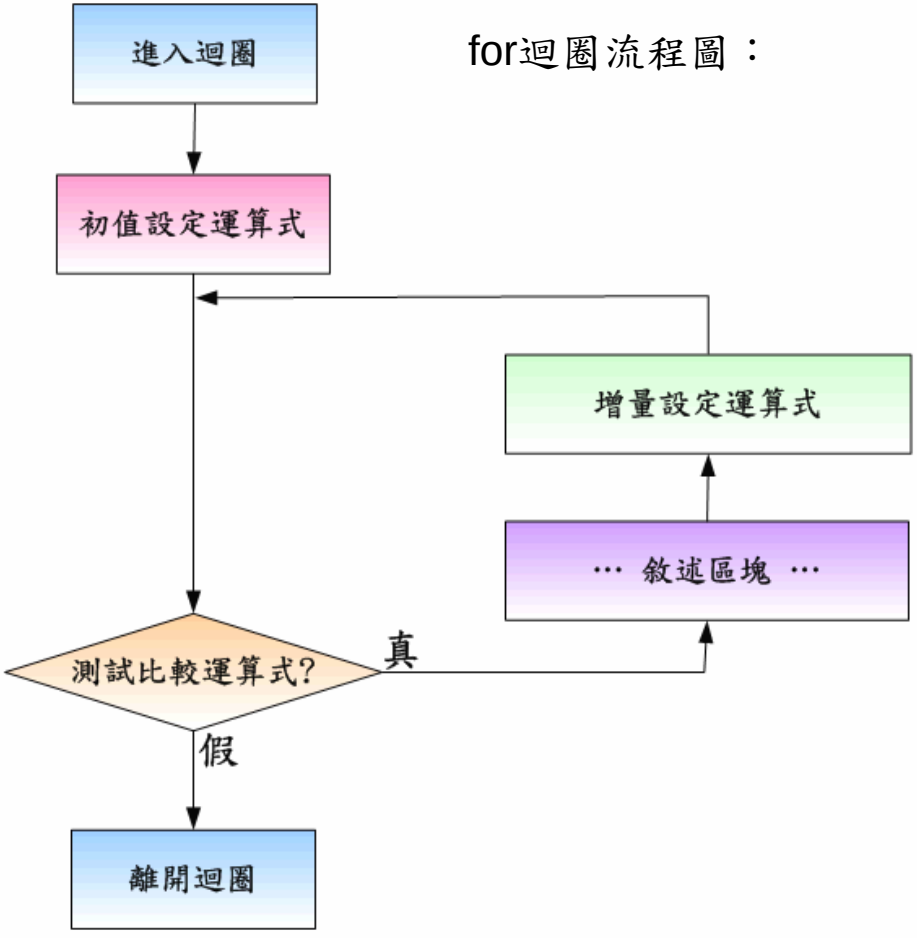
```
Sum=0;
for(count=1;count<=10;count++)
{
        Sum=Sum+count;
}
```

5.4.1計數迴圈（for迴圈敘述）

for迴圈執行流程：

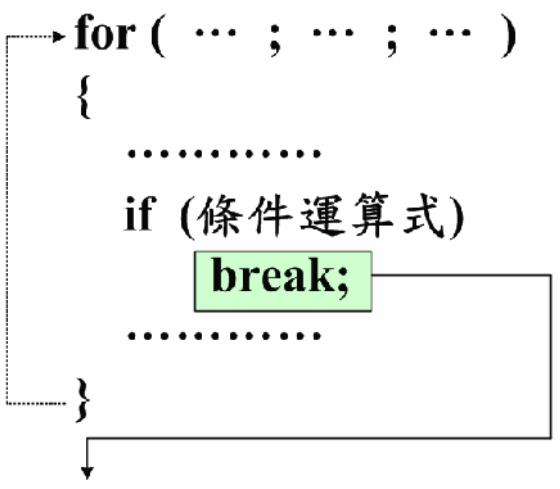


for迴圈流程圖：



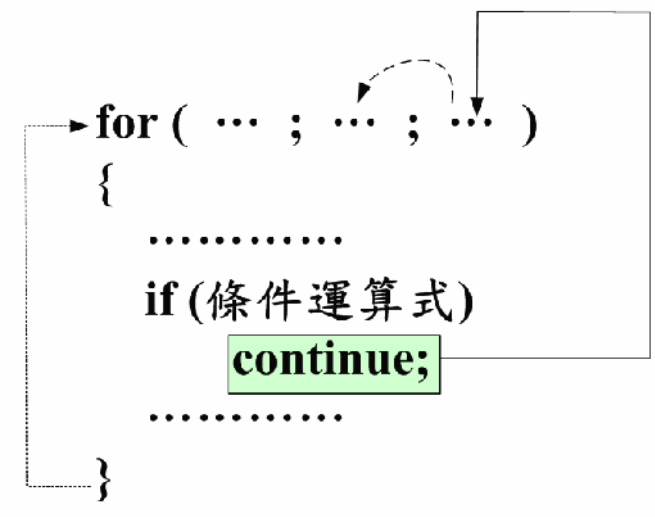
5.4.1 計數迴圈（for迴圈敘述）

- 用來強制中途跳出迴圈-break敘述，如下圖示意：



5.4.1 計數迴圈（for迴圈敘述）

- 則是用來強制中途略過本次迴圈剩餘尚未執行的步驟，-continue敘述，如下圖示意：



5.4.1 計數迴圈（for迴圈敘述）

- 【實用範例5-13】：使用for迴圈，計算1+3+ ···· +N（N爲奇數時）或1+3+····+N-1（N爲偶數時）的總和。

```
1  /*****************************
2      檔名:ch5_13.c
3      功能:for迴圈的示範
4   *****************************/
5  #include <stdio.h>
6  #include <stdlib.h>
7   void main(void)
8  {
9   int Sum,n,i;
10  printf("請輸入N值:");
11  scanf("%d",&n);
12  for(Sum=0,i=1;i<=n;i=i+2)
13       Sum=Sum+i;
14  if((n%2)==1)  printf("1+3+...+N=%d\n",Sum);
15  else
16       printf("1+3+...+N-1=%d\n",Sum);
17  /*  system("pause");  */
18 }
21
```

5.4.1 計數迴圈（for迴圈敘述）

- 【實用範例5-15】： 使用2層for迴圈設計九九乘法表。
- 範例5-15：ch5_15.c

```
1  /*****************************
2      檔名:ch5_15.c
3      功能:多層for迴圈的示範
4   *****************************/
5
6  #include <stdio.h>
7  #include <stdlib.h>
8
9  void main(void)
10 {
11  int i,j;
12  for(i=1;i<=9;i++)
13  {
14     for(j=1;j<=9;j++)
15        printf("%d*%d=%d\t",i,j,i*j);
16     printf("\n");
17  }
18  /*  system("pause");  */
19 }
```

5.4.1 計數迴圈（for迴圈敘述）

- 執行結果：

```
1*1=1   1*2=2   1*3=3   1*4=4   1*5=5   1*6=6   1*7=7   1*8=8   1*9=9
2*1=2   2*2=4   2*3=6   2*4=8   2*5=10  2*6=12  2*7=14  2*8=16  2*9=18
3*1=3   3*2=6   3*3=9   3*4=12  3*5=15  3*6=18  3*7=21  3*8=24  3*9=27
4*1=4   4*2=8   4*3=12  4*4=16  4*5=20  4*6=24  4*7=28  4*8=32  4*9=36
5*1=5   5*2=10  5*3=15  5*4=20  5*5=25  5*6=30  5*7=35  5*8=40  5*9=45
6*1=6   6*2=12  6*3=18  6*4=24  6*5=30  6*6=36  6*7=42  6*8=48  6*9=54
7*1=7   7*2=14  7*3=21  7*4=28  7*5=35  7*6=42  7*7=49  7*8=56  7*9=63
8*1=8   8*2=16  8*3=24  8*4=32  8*5=40  8*6=48  8*7=56  8*8=64  8*9=72
9*1=9   9*2=18  9*3=27  9*4=36  9*5=45  9*6=54  9*7=63  9*8=72  9*9=81
```

5.4.1計數迴圈（for迴圈敘述）

- 【觀念範例5-16】：使用continue敘述，計算2+4+6+8+10的總和。
- 範例5-16：ch5_16.c

```
1  /*************************
2      檔名:ch5_16.c
3      功能:continue敘述的示範
4   *************************/
5  #include <stdio.h>
6  #include <stdlib.h>
7   void main(void)
8  {
9   int Sum=0,i,j;
10  for(i=1;i<=10;i++)
11  {
12      if ((i%2)==1)
13          continue;
14      Sum=Sum+i;
15  }
16  printf("2+4+6+8+10=%d\n",Sum);
17  /*  system("pause");  */
18 }
```

5.4.1計數迴圈（for迴圈敘述）

- 【觀念範例5-17】：使用break敘述強迫中途跳出迴圈。
- 範例5-17：ch5_17.c

```
1  /*****************************
2      檔名:ch5_17.c
3      功能:break敘述的示範
4   *****************************/
5  #include <stdio.h>
6  #include <stdlib.h>
7  void main(void)
8  {
9   short int Sum,n,i;
10  printf("求1~N的總和,請輸入N值:");
11  scanf("%d",&n);
12  Sum=0;
13  for(i=1;i<=n;i++)
14  {
15     if(Sum>30000)
16        break;
17     Sum=Sum+i;
18  }
19  printf("1~%d的總和爲%d\n",i-1,Sum);
20  /*  system("pause");  */
21 }
```

5.4.2前測式條件迴圈（while迴圈敘述）

- for迴圈一般用在已經有明確初值及增量運算時使用。

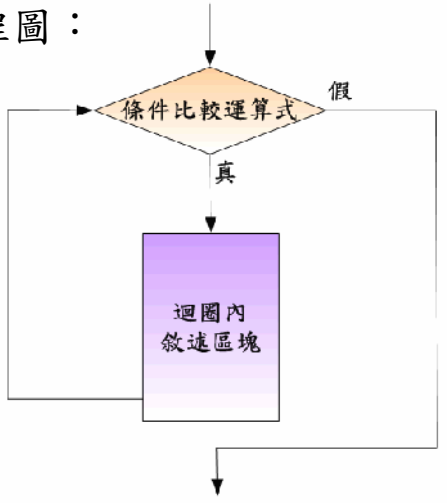
- 若無明確初值及增量運算時，C語言提供的條件式迴圈有兩種：前測式迴圈（while迴圈）及後測式迴圈（do-while迴圈）。

5.4.2前測式條件迴圈（while迴圈敘述）

- while迴圈語法：

```
while(條件比較運算式)
{
   ……敘述區塊……
}
```

while迴圈流程圖：



- 【功能】：
  - 執行迴圈前先檢查是否滿足條件式，若滿足則進入迴圈，否則離開迴圈。
- 【語法說明】：
  - 1. 若『條件比較運算式』成立，則進入迴圈內執行敘述區塊；否則不進入迴圈。
  - 2. 迴圈內敘述區塊執行完畢，將重新測試條件比較運算式，直到條件式不成立時，才跳離迴圈。
  - 3. 若迴圈內的敘述區塊僅包含單一敘述，則可以省略{ }。

5.4.2前測式條件迴圈（while迴圈敘述）

- 【實用範例5-18】：使用前測式while迴圈，撰寫一個根據輾轉相除法求兩數最大公因數的程式。輾轉相除法範例如右：

	792	102	
7	714	78	1
	78	24	
3	72	24	4
	6	0	

最大公因數

```
/**************************
    檔名:ch5_18.c
    功能:while迴圈的示範
 **************************/
#include <stdio.h>
#include <stdlib.h>
 void main(void)
{
 int x,y,gcd,temp;
 printf("輸入x:");  scanf("%d",&x);
 printf("輸入y:");  scanf("%d",&y);
 printf("(%d,%d)=",x,y);
if (x<y)        /* 將較大的數值放在x,較小的放在y */
 {
     temp = x; x = y; y = temp;  /* x,y數值對調 */
 }
 while(x!=0)
 {
     x=x%y;
     if(x!=0)
     {
       temp = x; x = y; y = temp; /* x,y數值對調 */
     }
 }
 gcd=y;
 printf("%d\n",gcd);
 /*  system("pause");  */
}
```

5.4.2前測式條件迴圈（while迴圈敘述）

- 執行結果：

```
輸入x:792
輸入y:102
(792,102)=6
```

```
輸入x:117
輸入y:663
(117,663)=39
```

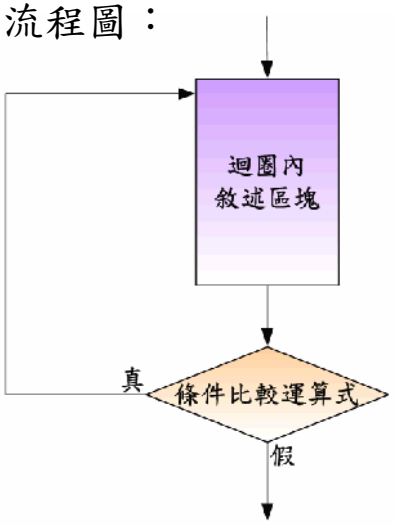
- 範例說明：
  - (1) 第17行共有3個敘述『temp = x;』、『x = y;』、『y = temp;』目的是爲了將x與y的值對調。
  - (2) 第19~26行：此處出現了一個while迴圈，迴圈執行的次數完全依照x變數值的變化來決定，除非x變數值爲0，否則迴圈將一直重複執行。

5.4.3後測式條件迴圈（do-while迴圈敘述）

- 另一種條件迴圈稱之爲後測式條件迴圈，也就是do-while迴圈。此類迴圈最大的特色是，迴圈內的敘述區塊至少會被執行一次。
- do-while迴圈語法：

```
do
{
   …..敘述區塊……
}while(條件比較運算式);
```

do-while迴圈流程圖：



- 【功能】：
  - 先進入迴圈，執行敘述區塊一次後，再判斷是否要繼續重覆執行迴圈。

5.4.3後測式條件迴圈（do-while迴圈敘述）

- 【實用與觀念範例5-19】：使用後測式do-while迴圈，撰寫一個模擬遊戲結束時的『Play Again?』詢問選項。
- 範例5-19：ch5_19.c

```
1  /*****************************
2      檔名:ch5_19.c
3      功能:do-while迴圈的示範
4   *****************************/
5  #include <stdio.h>
6  #include <stdlib.h>
7  void main(void)
8  {
9   char InputChar;
10  /* ..遊戲程式撰寫處.. */
11  printf("Game Over...\n");
12  do
13  {
14      printf("Play Again?(y/n)");
15      InputChar=getchar();
16      getchar();
17  }while((InputChar!='y') && (InputChar!='n'));}
18
```

5.4.4 迴圈的適用狀況

- 在5.4節中，我們介紹了3種迴圈，您可以在不同狀況下，選擇最方便的迴圈來實作程式需要重複執行的片段，以下是我們的建議。

迴圈	適用時機
for	有初始設定與增量運算時。
while	無初始設定與增量運算時。
do-while	至少執行迴圈內的敘述區塊一次。

5.4.4 迴圈的適用狀況

- 至於break與continue敘述的區別則整理如下

迴圈內的特殊敘述	敘述類型	功能
break	跳躍敘述（jump-statement）	立刻強制中途跳出迴圈。
continue	跳躍敘述（jump-statement）	強制略過該次迴圈重複（iteration）尚未被執行的敘述，直接跳到下一個重複（iteration）條件比較判斷式（while迴圈、do-while迴圈）或增量設定運算式（for迴圈）。

5.5 強制跳躍（goto敘述）

- goto敘述是一種跳躍敘述（jump-statement），用來強制引導程式的執行順序到指定的位置，雖然很多程式語言都提供這個敘述。
- 但是結構化的程式語言（例如C語言）大多不建議使用這個敘述，原因則是因爲使用goto敘述會破壞程式結構，使得程式難以維護，而且所有的程式都可以在完全不使用goto敘述的情況下來完成，因此建議讀者最好不要使用goto敘述。
- goto敘述語法：

語法：goto label
功能：直接跳躍到label位置。

5.5 強制跳躍（goto敘述）

- 【範例5-20】：使用goto敘述跳出多層迴圈。計算1!+2!+...+m!,(0<m<10)。
- 範例5-20：ch5_20.c

```
/*****************************
    檔名:ch5_20.c
    功能:goto敘述的示範
 *****************************/

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

void main(void)
{
 int Sum,FactNum,x,y,m;
 printf("計算
1!+2!+...+m!,(0<m<10)\n");
 printf("請輸入m:");
 scanf("%d",&m);
 for(Sum=0,x=1;x<10;x++)
 {
    for(FactNum=1,y=1;y<=x;y++)
    {
        FactNum=FactNum*y;
        if(y==m+1)
            goto ProgEnd;
    }
    Sum=Sum+FactNum;
  }
ProgEnd:
printf("1!+2!+...+m!=%d(m=%d)\n",Sum,m);
 /*  system("pause");  */
}
```

5.5 強制跳躍（goto敘述）

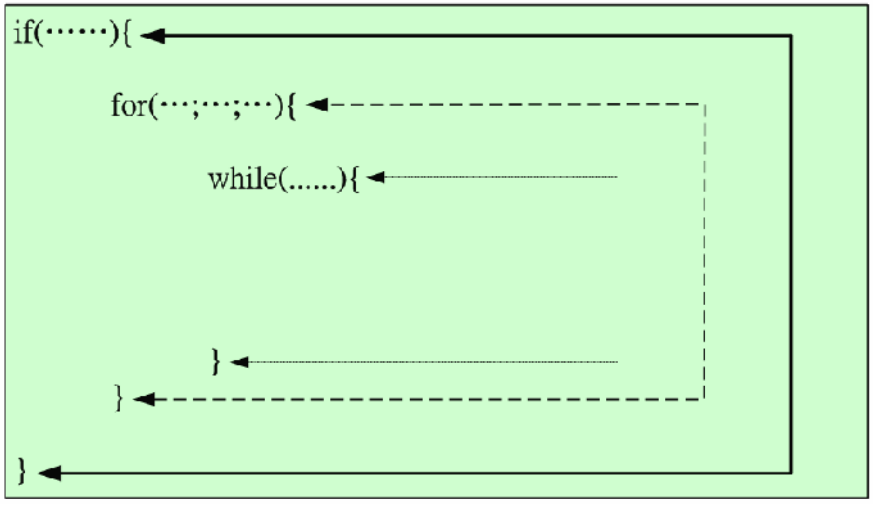
- 執行結果：

```
計算1!+2!+...+m!,(0<m<10)
請輸入m:6
1!+2!+...+m!=873(m=6)
```

- 範例說明：
  - 當第22行被執行時，就會強制將程式流程跳躍到第26行的ProgEnd標籤處（離開了兩個迴圈）。

5.6 巢狀與縮排

- 『巢狀』和『縮排』原本是兩件不相干的事。
- 『巢狀』可以使用在條件敘述，製作巢狀條件敘述；也可以使用在迴圈，製作多重迴圈敘述；或者將條件敘述與迴圈混合使用。因此，對於編譯器而言，『巢狀』敘述是有意義的。
- 『縮排』只不過是爲了讓程式設計師更容易看出程式的邏輯結構所做的自發性動作，對於編譯器而言，一個程式是否縮排並不影響程式的正確性。



5.7 本章回顧

- 在本章中，我們認識了結構化程式語言的3種基本程式流程分別是『循序』結構、『選擇』結構、『迴圈』結構。相對於C語言來說，也同樣提供了此三種基本結構。
  - 『循序』結構是最基本的流程，由上而下一一執行敘述，而C的選擇敘述則包含if、if-else、switch-case等三種，以及變形的else-if格式。除此之外，e1？e2：e3特殊選擇運算式則可以用來作爲簡單的選擇替換敘述。
  - 在迴圈敘述方面，C語言提供了for計數迴圈、while前測式迴圈以及do-while後測式迴圈。
  - 同時我們也釐清了break與continue在迴圈應用時的不同處。
  - goto敘述則不建議讀者採用，因爲它將會破壞程式的結構化，導致日後程式難以維護。

本章習題