C 程式語言 2

關係運算子與 if 敘述

- if 敘述與關係運算子

if 敘述的格式

```
if(判斷條件)
    敘述主體;
```

關係運算子	意義	範例	說明
>	大於	a>b	判別 a 是否大於 b
<	小於	a<b	判別 a 是否小於 b
>=	大於等於	a>=b	判別 a 是否大於等於 b
<=	小於等於	a<=b	判別 a 是否小於等於 b
==	等於	a==b	判別 a 是否等於 b
!=	不等於	a!=b	判別 a 是否不等於 b

遞增與遞減運算子

- 遞增運算子(++)與遞減運算子(--)
  - ✓來自變數內容遞增 1 與遞減 1 的精簡版寫法

- 簡單的說
  - ✓a = a + 1 可寫成 a++
    - 讓變數內容加 1
  - ✓a = a - 1 可寫成 a--
    - 讓變數內容減 1

遞增與遞減運算子

- 遞增與遞減運算子的成員：

遞增與遞減運算子	意義	範例	說明
++	遞增，變數值加 1	a++	a 加 1 後再設定給 a 存放
--	遞減，變數值減 1	a--	a 減 1 後再設定給 a 存放

- a++ 會先執行整個敘述後，再將 a 的值加 1
- ++a 則是先把 a 的值加 1 後，再執行整個敘述

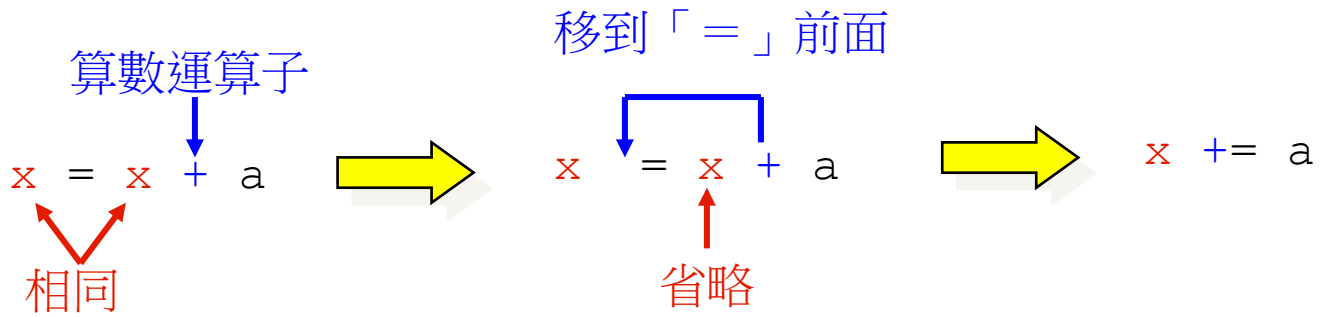
遞增與遞減運算子

- 下面的程式是使用遞增運算子的範例：

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
  int a=3, b=3;
  printf("a=%d",a);
  printf(", a++ return value%d",a++);     /* 計算a++，並印出其傳回值 */
  printf(", a=%d\n",a);
  printf("b=%d",b);
  printf(“, ++b return value%d",++b);     /* 計算++b，並印出其傳回值 */
  printf(", b=%d\n",b);

  system("pause");
  return 0;
}
```

運算式與簡潔運算子

- 算數運算子與指定運算子「=」的簡便寫法
  - 例如：**x = x + a 可以寫成 x += a**



運算式與簡潔運算子

運算式 為運算子與運算元組成

- 簡潔運算子可簡化運算式

運算子	範例用法	說明	意義
+=	a+=b	a+b 的值存放到 a 中	a=a+b
-=	a-=b	a-b 的值存放到 a 中	a=a-b
*=	a*=b	a*b 的值存放到 a 中	a=a*b
/=	a/=b	a/b 的值存放到 a 中	a=a/b
%=	a%=b	a%b 的值存放到 a 中	a=a%b

括號運算子

- 括號運算子「()」用來提高運算式的優先順序

括號運算子	意義
()	提高括號中運算式的優先順序

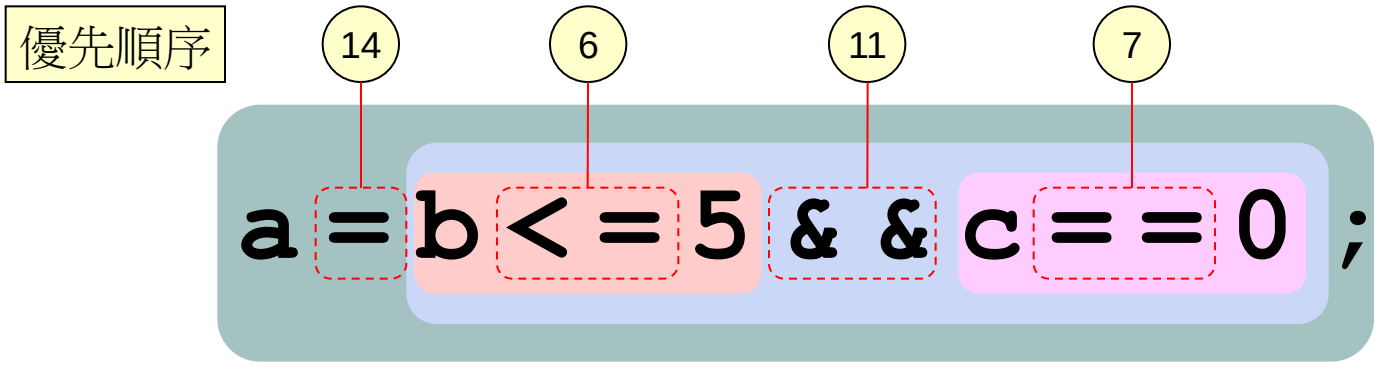
- `3+5*4*6-7;` /* 未加括號的運算式 */
- `(3+5*4)*(6-7);` /* 加上括號的運算式 */

運算子的優先順序

優先順序	運算子	類別	結合性
1	()	括號運算子	由左至右
1	[]	方括號運算子	由左至右
2	!、+ （正號）、- （負號）	一元運算子	由右至左
2	~	位元邏輯運算子	由右至左
2	++、--	遞增與遞減運算子	由右至左
3	*、/、%	算數運算子	由左至右
4	+、-	算數運算子	由左至右
5	<<、>>	位元左移、右移運算子	由左至右
6	>、>=、<、<=	關係運算子	由左至右
7	==、!=	關係運算子	由左至右
8	& （位元運算的 AND）	位元邏輯運算子	由左至右
9	^ （位元運算的 XOR）	位元邏輯運算子	由左至右
10	| （位元運算的 OR）	位元邏輯運算子	由左至右
11	&&	邏輯運算子	由左至右
12	||	邏輯運算子	由左至右
13	?:	條件運算子	由右至左
14	=	設定運算子	由右至左

運算子的優先順序

- 運算子優先順序的範例：



1. 先計算 b<=5 (<=的優先順序為6)
2. 再計算 c==0 (==的優先順序為7)
3. 然後進行&&運算 (&&的優先順序為11)
4. 最後再把運算結果設給變數 a 存放 (= 的優先順序為14)

條件運算子

- ?: 是一個非常有趣的運算子
- 語法如下：

條件判斷 ？ 運算式1 ： 運算式2

- 規則
  - 當「條件判斷　」的計算結果為 true 時
    - 傳回「運算式1」的計算結果
  - 當「條件判斷　」的計算結果為 false 時
    - 傳回「運算式2」的計算結果。

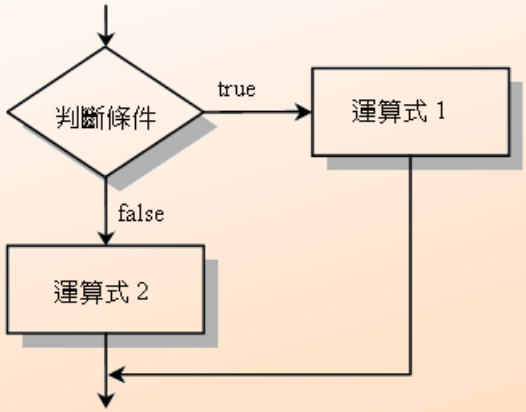
條件運算子

- 條件運算子可代替簡單的 if-else 敘述

條件運算子	意義
?:	根據條件的成立與否，來決定結果為?或:後的運算式

條件運算子的語法

條件判斷 ? 運算式1 : 運算式2



條件運算子的範例

- 利用條件運算子判斷兩個數中較大的數：

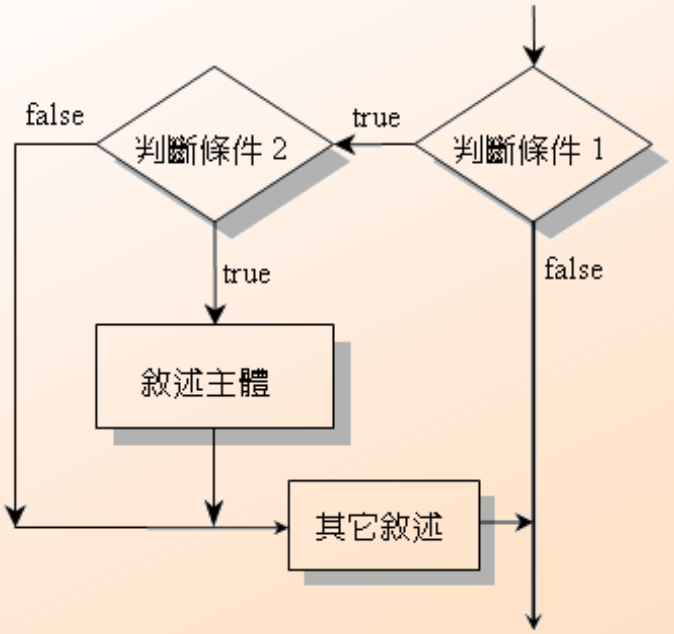
```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
  int num1,num2,larger;
  printf("Please input two integers: :");
  scanf("%d %d",&num1,&num2);

  num1>num2 ? (larger=num1) : (larger=num2);  /* 條件運算子 */
  printf("%d greater value \n",larger);

  system("pause");
  return 0;
}
```

巢狀 if 敘述

- if 裡面還有其它的if 敘述，則稱為巢狀 if 敘述

巢狀 if 敘述的語法

```
if(判斷條件1)
{
    if(判斷條件2)
    {
        敘述主體;
    }
    ...
    其它敘述;
}
```

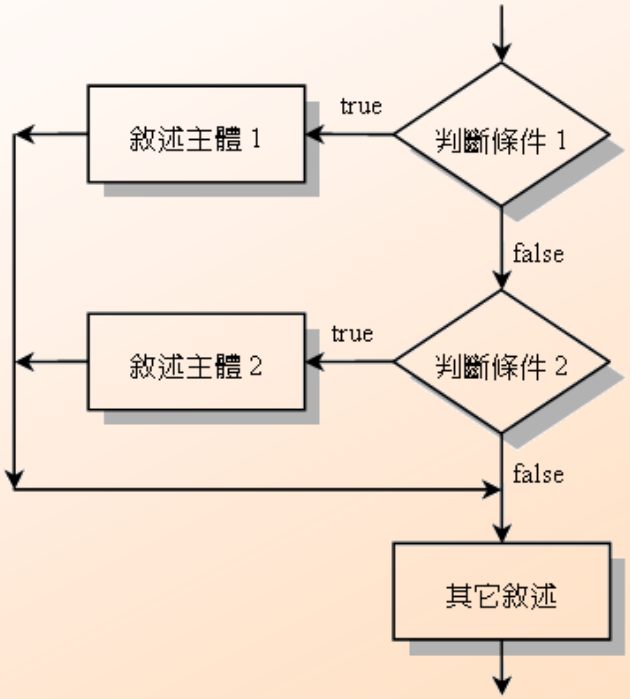


使用 if-else-if 敘述

- if-else-if：當 if 判斷不成立，必須進行其它判斷時

if-else-if 敘述的語法

```
if(判斷條件1)
{
    敘述主體1;
}
else if(判斷條件2)
{
    敘述主體2;
}
```

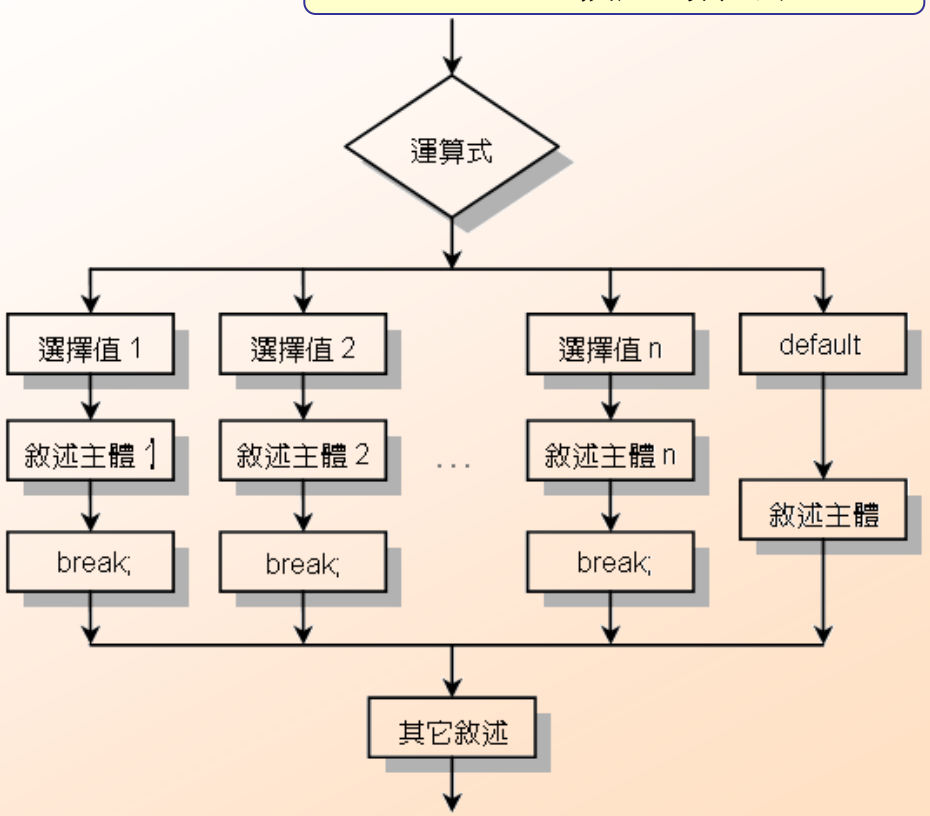


switch 敘述

- switch 敘述可用來進行多重選擇

switch 敘述的語法

```
switch(運算式)
{
    case 選擇值1:
        敘述主體1;
        break;
    case 選擇值2:
        敘述主體2;
        break;
         ...
    default:
        敘述主體;
}
```



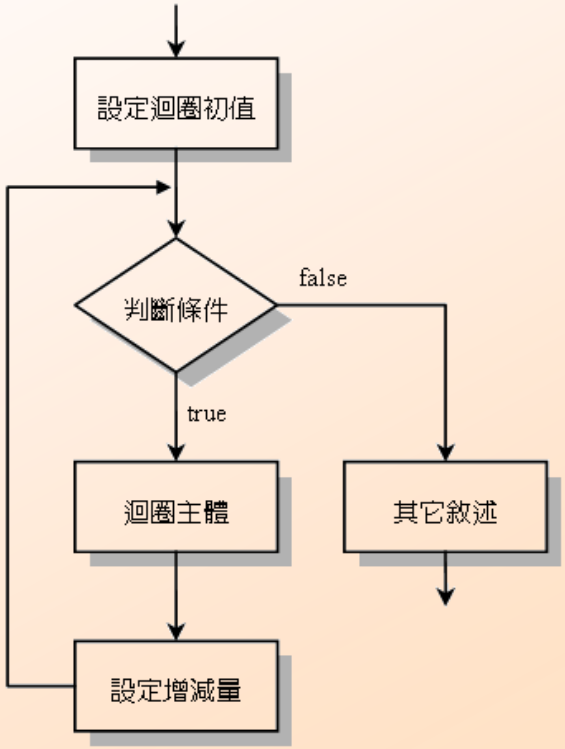
使用while 迴圈

- while 適合用在迴圈執行次數為未知時

while 迴圈的語法

```
設定迴圈初值;
while(判斷條件)
{
    迴圈主體;
    設定增減量;
}
```

這兒不可以加分號



清除緩衝區的資料

- fflush可用來清除緩衝區的資料

fflush() 函數的用法

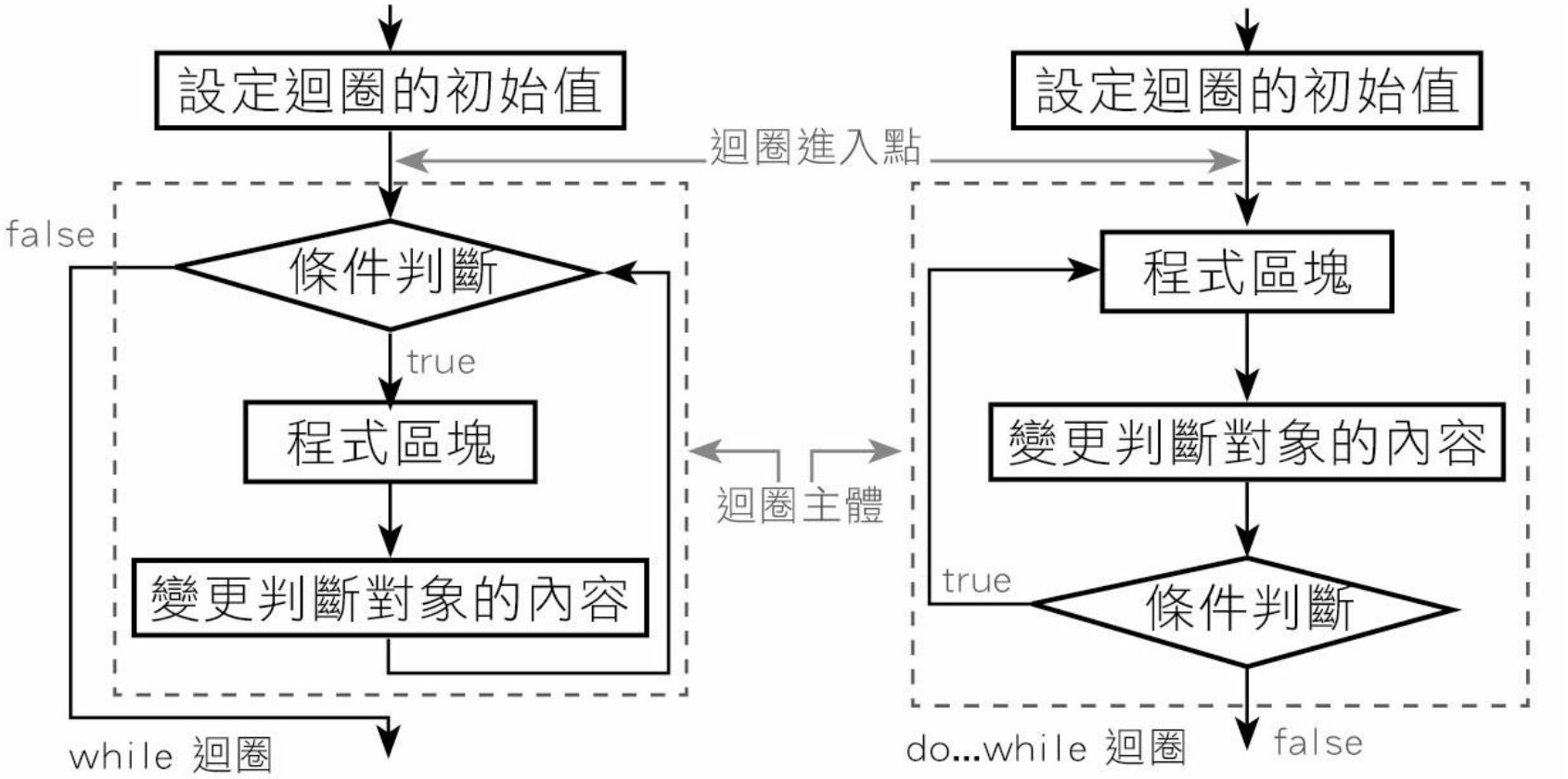
```
fflush(stdin);                    /* 清除緩衝區內的資料 */
```

while與do while

後測式do/while和前測式while迴圈的主要差異是在迴圈結尾檢查條件，因為先執行程式區塊的程式碼後才測試條件，所以do/while迴圈的程式區塊至少會執行「1」次

迴圈的概念與基本架構

- 先判斷再執行
- 先執行再判斷

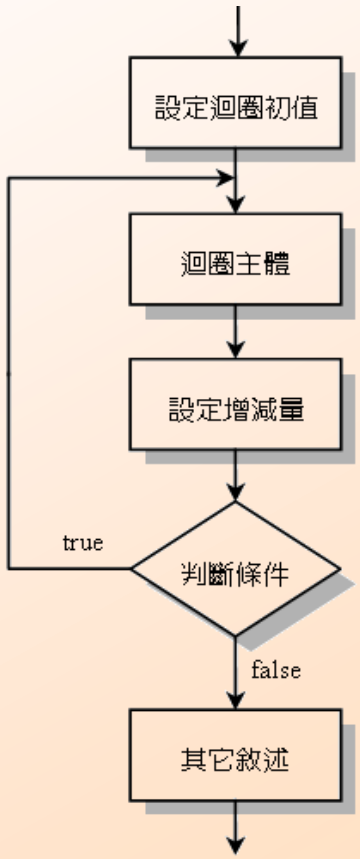


使用do while迴圈

```
設定迴圈初值;
do
{
    迴圈主體;
    設定增減量;
} while(判斷條件);
```

這兒要加分號

do-while 迴圈的語法



迴圈的跳離

雖然迴圈敘述可以在開頭或結尾測試結束條件，但是有時我們需要在迴圈之中測試結束條件，我們可以使用break關鍵字來跳出迴圈，如同switch條件敘述使用break關鍵字跳出程式區塊

迴圈的跳離

- break 敘述：
  - 略過迴圈主體的其餘部分，執行迴圈之後的敘述

break 敘述的語法

```
for(初值設定; 判斷條件; 設定增減量)
{
    敘述 1;
    敘述 2;
      …
    break;
      …
    敘述 n;
}
  ...
```

若執行break敘述，則此區塊內的敘述不會被執行

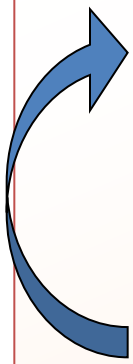
continue 敘述

在迴圈執行過程中，相對於上一個使用break關鍵字跳出迴圈，continue關鍵字可以馬上繼續下一次迴圈的執行，而不執行程式區塊位在continue關鍵字之後的程式碼

continue 敘述

- continue 敘述：
  - 略過迴圈主體的其餘部分，直接開始下一個迴圈循環

continue 敘述的語法



```
for(初值設定; 判斷條件; 設定增減量)
{
    敘述 1;
    敘述 2;
      …
    continue;
      …
    敘述 n;
}
  ...
```

若執行continue敘述，則此區塊內的敘述不會被執行

-The End-