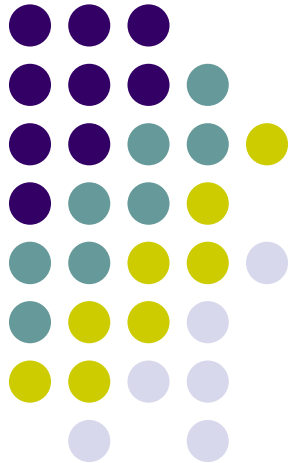
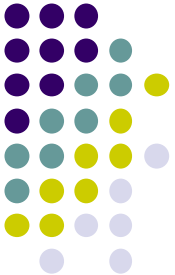
C語言補充
格式化輸入

學習 cin / scanf() 的輸入方法

認識輸入格式碼

學習字元的輸入函數





輸入函數scanf() (1/6)

- scanf() 函數可用來輸入字元、數字或字串
- scanf() 函數的格式如下：

scanf() 函數的格式

```
scanf("格式字串", &變數1, &變數2, ...);
```

只有scanf有，printf沒有

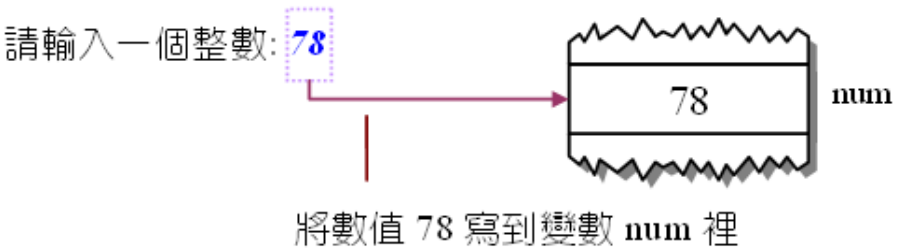
取變數的記憶體位置 (房子的地址)

輸入函數scanf() (2/6)

- 由鍵盤輸入一個整數的範例：

```
01  /* prog4_9, 使用 scanf()函數 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06     int num;
07
08     printf("請輸入一個整數:");
09     scanf("%d",&num);          /* 由鍵盤輸入整數，並指定給 num 存放 */
10     printf("num=%d\n",num);    /* 印出 num 的內容 */
11
12     system("pause");
13     return 0;
14  }
```

```
scanf("%d",&num);
```

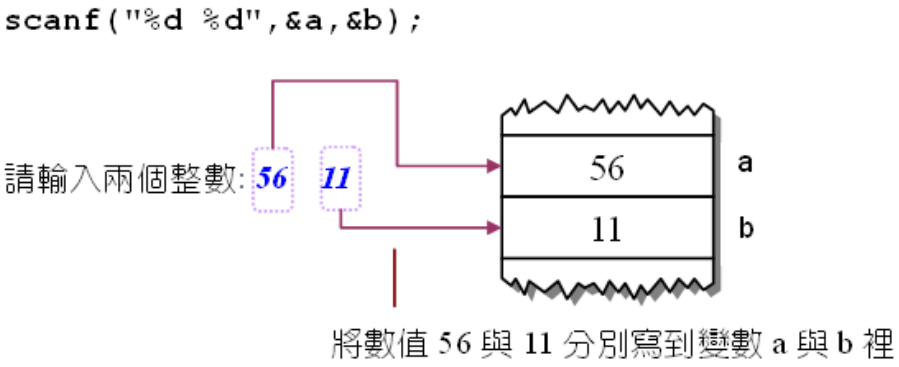



```
/* prog4_9 OUTPUT---
請輸入一個整數: 78
num=78
----------------------*/
```

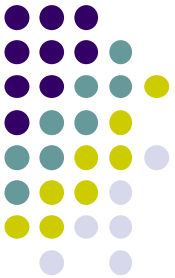
輸入函數scanf() (3/6)

- 由鍵盤上輸入兩個整數的範例：

```
01  /* prog4_10, 使用 scanf()函數
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06     int a,b;
07
08     printf("請輸入兩個整數: ");
09     scanf("%d %d",&a,&b);          /* 由鍵盤輸入二個數並設定給變數 a、b */
10     printf("%d+%d=%d\n",a,b,a+b);    /* 計算總和並印出內容 */
11
12     system("pause");
13     return 0;
14  }
```




```
/* prog4_10 OUTPUT--

請輸入兩個整數: 56 11
56+11=67
----------------------*/
```



輸入函數scanf() (4/6)

- 使用逗號區隔輸入：

```
01  /* prog4_11, 使用逗號區隔輸入格式 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06     int a,b;
07
08     printf("請輸入兩個整數，請用逗號隔開數值: ");
09     scanf("%d,%d",&a,&b);          /* 以「,」隔開兩個輸入格式碼 */
10     printf("%d+%d=%d\n",a,b,a+b);    /* 計算總和並印出內容 */
11
12     system("pause");
13     return 0;
14  }
```

控制輸入格式. CH05-1, practice 1: ("(%f, %f)", &a, &b)

```
/* prog4_11 OUTPUT--------------
請輸入兩個整數，請用逗號隔開數值: 14,36
14+36=50
--------------------------------*/
```

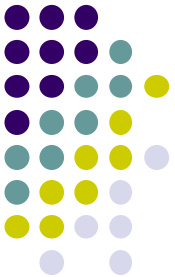
輸入函數scanf() (5/6)

- scanf() 函數常用的輸入格式碼：

表 4.2.1　scanf() 函數常用的輸入格式

輸入格式	輸入敘述	輸入格式	輸入敘述
%c	字元	%s	字串
%d	十進位整數	%o	八進位整數
%f	浮點數	%x	十六進位整數
%lf	倍精度浮點數（注意%lf 裡的 l 是英文小寫字母 l）		

```
scanf("格式字串", &變數1, &變數2, …);
```



輸入函數scanf() (6/6)

- 下面的範例可輸入一個十六進位的數值：

```
01  /* prog4_12, 輸入十六進位數值，再印出它的十進位 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06     int num;
07
08     printf("請輸入十六進位的整數：");
09     scanf("%x",&num);          /* 輸入十六進位數值，並指定給變數 num */
10     printf("%x 的十進位為%d\n",num,num); /* 將十六進位數值以十進位印出 */
11
12     system("pause");
13     return 0;
14  }
```

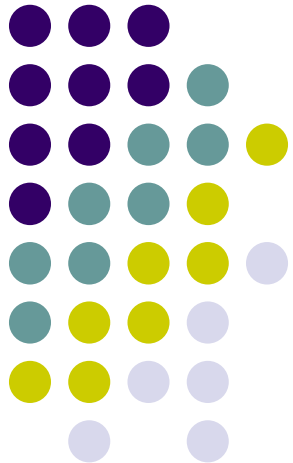
```
/* prog4_12 OUTPUT----

請輸入十六進位的整數：12ab
12ab 的十進位為 4779
-------------------------*/
```

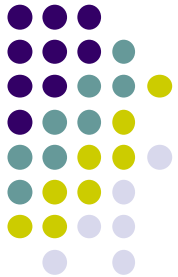
第四章
運算子、運算式與敘述

遞增遞減運算子

簡潔運算子

條件運算子





運算子優先順序的排列

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

遞增與遞減運算子 (1/3)

- 遞增與遞減運算子的成員

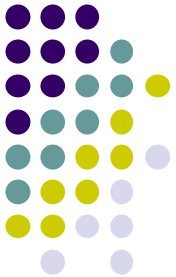
遞增與遞減運算子	意義
++	遞增，變數值加 1
--	遞減，變數值減 1

- 想讓變數i的值加上1，有下列兩種寫法

```
i=i+1;        // i 加 1 後再設定給 i 存放
i++;          // i 加 1 後再設定給 i 存放，i++為簡潔寫法
```

- i++與++i的區別
  - i++ 會先執行整個敘述後再將i的值加1
  - ++i 則先把i的值加1後，再執行整個敘述

遞增與遞減運算子 (2/3)

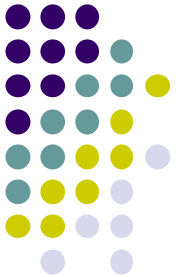


- 下面的程式可以觀察遞增運算子的使用

```
01  // prog4_6, 遞增運算子「++」在運算元之後
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  int main(void)
06  {
07     int a=10;
08     cout << "a=" << a << endl;          // 印出 a
09     cout << "a++*2=" << (a++*2) << endl;   // 印出 a++*2
10     cout << "a=" << a << endl;          // 印出 a
11     system("pause");
12     return 0;
13  }
```

```
/* prog4_6 OUTPUT---

a=10
a++*2=20
a=11

--------------------*/
```

遞增與遞減運算子 (3/3)



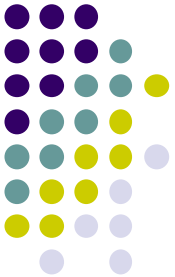
- 請比較一下遞增運算子放在運算元前後的差別

```
01  // prog4_7, 遞增運算子「++」在運算元之前
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  int main(void)
06  {
07     int a=10;
08     cout << "a=" << a << endl;                    // 印出 a
09     cout << "++a*2=" << (++a*2) << endl;          // 印出++a*2
10     cout << "a=" << a << endl;                    // 印出 a
11     system("pause");
12     return 0;
13  }
```

(--a*2) : a=10
--a*2=18
a=9

先a+1，再*2

```
/* prog4_7 OUTPUT---
a=10
++a*2=22
a=11
--------------------*/
```

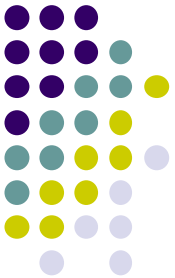


算數與設定運算子的結合 (1/3)

- 下面幾個運算式，皆是簡潔的寫法

```
a++;          // 相當於 a=a+1
b-=3;         // 相當於 b=b-3
b%=c;         // 相當於 b=b%c
```

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



算數與設定運算子的結合 (2/3)

- 下面的範例實際練習一下簡潔運算式的寫法

```
01  // prog4_8, 簡潔運算式
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  int main(void)
06  {
07     int a=100,b=15;
08     cout << "a=" << a << ", b=" << b << endl;
09     a-=b;                              // 計算 a=a-b 的值
10     cout << "after a-=b, a=" << a << ", b=" << b << endl;
11     system("pause");
12     return 0;
13  }
```

```
/* prog4_8 OUTPUT---------
a=100, b=15
after a-=b, a=85, b=15
-------------------------*/
```

算數與設定運算子的結合 (3/3)

- 下表列出簡潔寫法的運算子及其範例說明

運算子	範例	執行前		說明	執行後	
		a	b		a	b
+=	a+=b	12	4	a+b 的值存放到 a 中 (同 a=a+b)	16	4
-=	a-=b	12	4	a-b 的值存放到 a 中 (同 a=a-b)	8	4
*=	a*=b	12	4	a*b 的值存放到 a 中 (同 a=a*b)	48	4
/=	a/=b	12	4	a/b 的值存放到 a 中 (同 a=a/b)	3	4
%=	a%=b	12	4	a%b 的值存放到 a 中 (同 a=a%b)	0	4
b++	a*=b++	12	4	a*b 的值存放到 a 後，b 加 1 (同 a=a*b; b++)	48	5
++b	a*=++b	12	4	b 加 1 後，再將 a*b 的值存放到 a (同 b++; a=a*b)	60	5
b--	a*=b--	12	4	a*b 的值存放到 a 後，b 減 1 (同 a=a*b; b--)	48	3
--b	a*=--b	12	4	b 減 1 後，再將 a*b 的值存放到 a (同 b--; a=a*b)	36	3

練習下列之運算

- 課本4-28頁,第10題

int a=5,b=3,num=0;

	num	a	b	
a)	8	6	3	num=(a++)+b;
b)	9	6	3	num=(++a)+b;
c)	8	6	4	num=(a++)+(b++);
d)	10	6	4	num=(++a)+(++b);
e)	x	~~14~~ 13	4	a+=a+(b++);
f)	x	14	4	a+=a+(++b)

num=a+++b 無法編譯
num=a++空格+b 可編譯

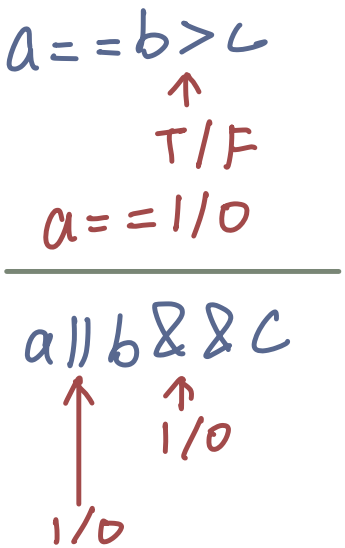
- 額外練習

int a=12, b=6;

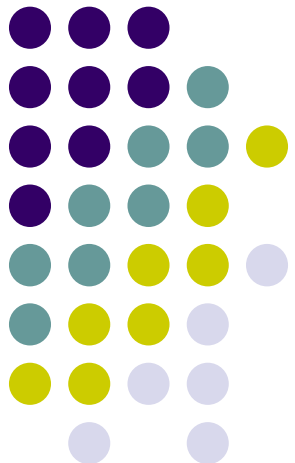
	a	b	
a)	2	6	a/=b;
b)	72	7	a*=b++;
c)	84	7	a*=++b;
d)	72	5	a*=b--;
e)	0	6	a%=b;

運算子優先順序的排列

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



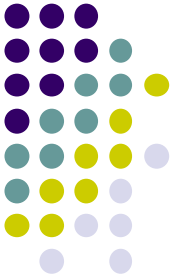
第五章
選擇性敘述與迴圈



認識程式的結構設計

學習選擇性敘述與各種迴圈的用法

學習多重選擇敘述的使用

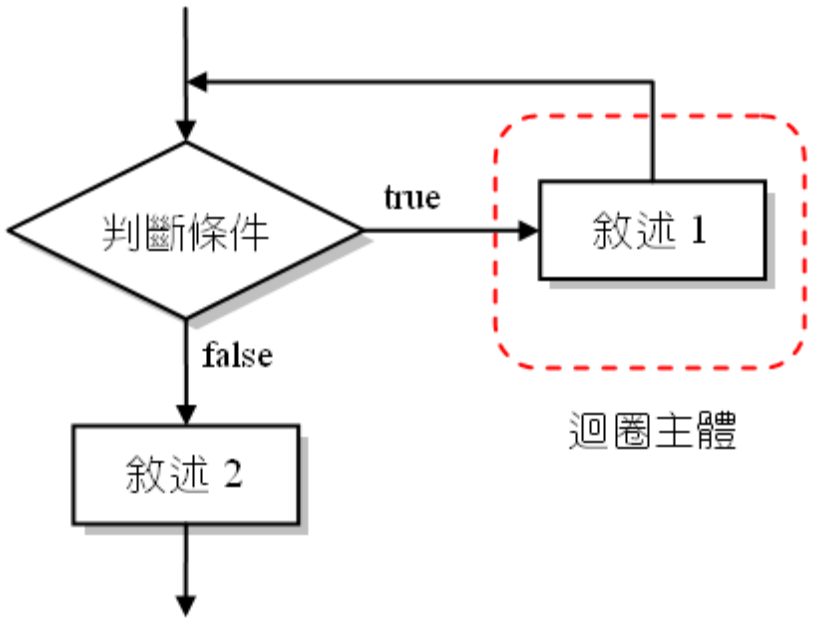


程式的結構

- 程式的結構包含有下面三種：
  - 循序性結構（sequence structure）
  - 選擇性結構（selection structure）
  - 重複性結構（iteration structure）

重複性結構

- 重複性結構根據判斷條件的成立與否，決定程式執行的次數



- 重複性結構有for、while及do while三種迴圈

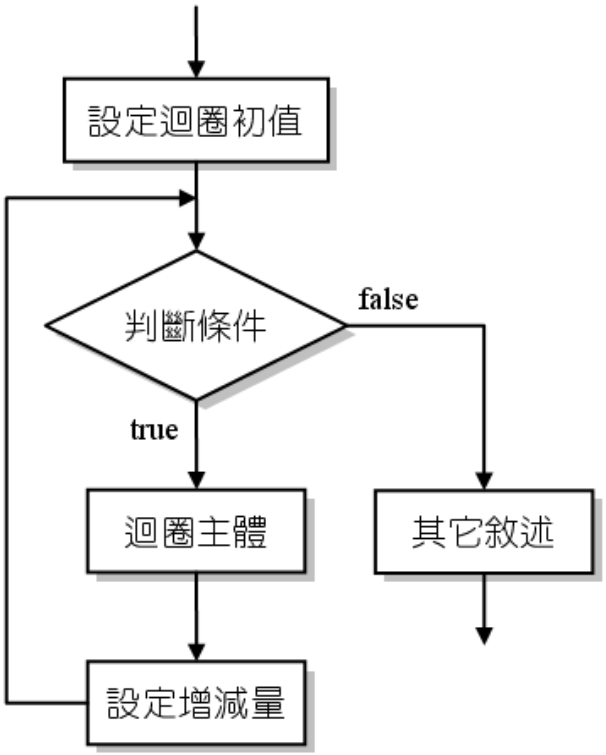
for迴圈 (1/3)

- for迴圈敘述格式如下

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

這兒不可以加分號

這兒不可以加分號

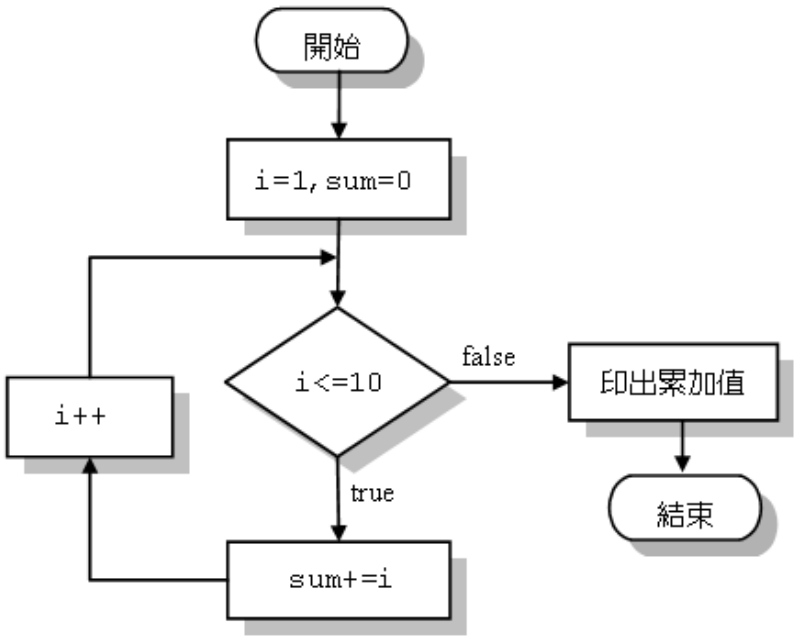


- for迴圈執行的流程
  - 第一次進入for迴圈時，設定迴圈控制變數的起始值
  - 根據判斷條件的內容，檢查是否要繼續執行迴圈
  - 迴圈控制變數會根據增減量的設定，更改迴圈控制變數的值，再回到上一個步驟重新判斷是否繼續執行迴圈

for 迴圈範例一 (1/2)

- 利用for迴圈計算1加到10的總和

```
01  /* prog7_1, for 迴圈的使用 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06     int i,sum=0;
07     for(i=1;i<=10;i++){
08        sum+=i;}
09     printf("1+2+3+...+10=%d\n",sum);
10
11     system("pause");
12     return 0;
13  }

/* prog7_1 OUTPUT--

1+2+3+...+10=55
----------------------*/
```



for 迴圈範例一 (1/2)

- 迴圈內，變數變化的情形

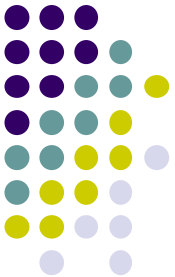
```
06      int i,sum=0;
07      for(i=1;i<=10;i++)
08         sum+=i;
```

表 7.2.1 for 迴圈內，i 與 sum 值變化的情形

i 的值	sum 的值	計算 sum+=i 之後，sum 的值
1	0	sum= 0+1= 1
2	1	sum= 1+2= 3
3	3	sum= 3+3= 6
4	6	sum= 6+4= 10
5	10	sum=10+5= 15
6	15	sum=15+6= 21
7	21	sum=21+7= 28
8	28	sum=28+8= 36
9	36	sum=36+9= 45
10	45	sum=45+10=55

sum = sum + i

執行完 for 迴圈之後，sum 的值



for迴圈 (3/3)

- 下面的範例說明如何在for迴圈內使用區域變數

```
01  // prog5_4, 區域變數
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  int main(void)
06  {
07      int sum=0;
08      for(int i=1;i<=5;i++)
09      {
10         sum+=i;
11         cout << "i=" << i << ", sum=" << sum << endl;
12      }
13
14      system("pause");
15      return 0;
16  }
```

宣告在內，i隨迴圈存在或消失

新版C++可宣告在迴圈內，C語言不可

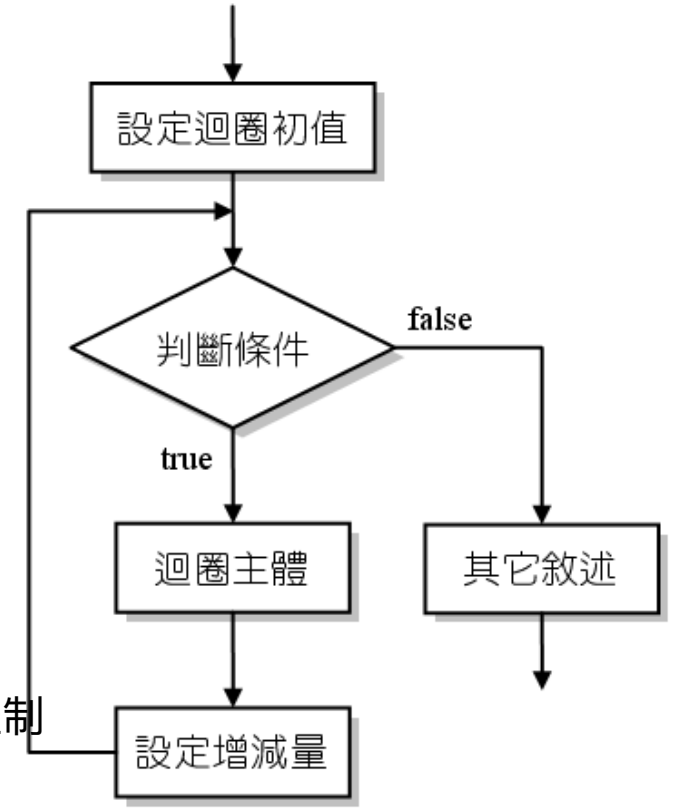
變數i的有效範圍

```
/* prog5_4 OUTPUT---
i=1, sum=1
i=2, sum=3
i=3, sum=6
i=4, sum=10
i=5, sum=15
------------------*/
```

while迴圈 (1/2)

- while迴圈的格式如下

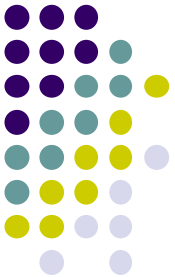
```
設定迴圈初值;
while(判斷條件)          → 這兒不可以加分號
{
    迴圈主體;
    設定增減量;
}                        → 這兒不可以加分號
```



下面列出while迴圈執行的流程

不一定要

- 第一次進入while迴圈前，就必須先設定迴圈控制變數的起始值
- 根據判斷條件的內容，檢查是否要繼續執行迴圈
- 執行完迴圈主體內的敘述後，重新設定（增加或減少）迴圈控制變數的值，再回到步驟2



while迴圈的範例

- 將while用在迴圈執行次數為未知：

```
01  /* prog7_3, while 迴圈的使用 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06     int i=1,sum=0;      /* 設定迴圈初值 */
07     while(sum<=100)     /* while 迴圈，當 sum 小於 100 則繼續累加 */
08     {
09        sum+=i;
10        printf("從 1 累加到%2d=%2d\n",i,sum);
11        i++;
12     }
13     printf("必須累加到%d\n",i-1);
14     system("pause");
15     return 0;
16  }
```

```
/* prog7_3 OUTPUT---
從 1 累加到  1=  1
從 1 累加到  2=  3
  ...
從 1 累加到 14=105
必須累加到 14
--------------------*/
```

```
for(i=1; sum<=100; i++)
{
 sum+=i;
 printf("從1累加到%2d=%2d\n",i,sum);
}
```

for與while迴圈的比較

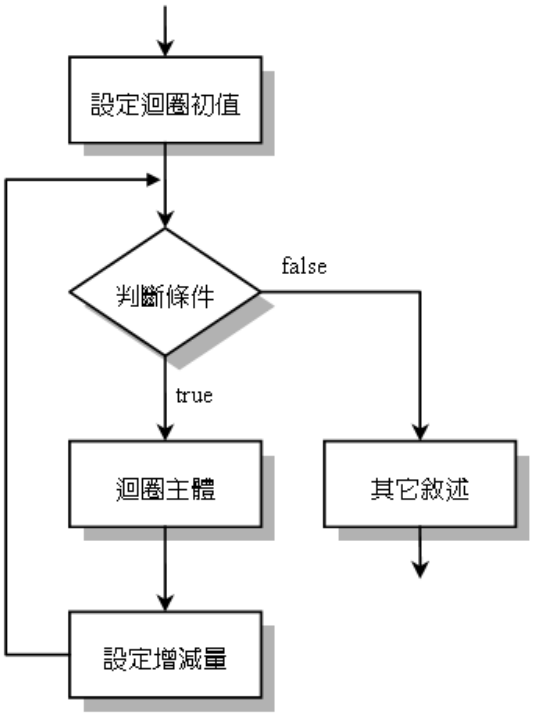
- for 迴圈與 while 迴圈的對等敘述：

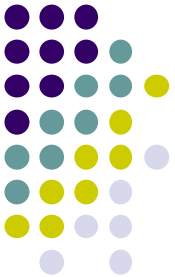
表 7.3.1　for 迴圈與 while 迴圈的敘述比較

for 迴圈	while 迴圈

for(設定初值;判斷條件;設定增減量)
{
敘述 1;
敘述 2;
⋮
敘述 n;
}
可留空白
ex: for(;判斷條件;)
易判斷迴圈重行次數 ⟷ 不易
	設定初值;
	while(判斷條件)
	{
	敘述 1;
	敘述 2;
	⋮
	敘述 n;
	設定增減量
	}

for 100%可改成 while
while 100%可改成 for





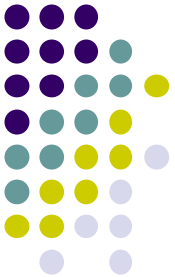
無窮迴圈的範例一

- 當迴圈沒有出口時，即稱為無窮迴圈

```
01  /* prog7_4, 無窮迴圈的說明 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06     int i=1;
07
08     while (i > 0)       /* 當 i>0 時執行 while 迴圈的主體 */
09        printf("i=%d\n",i++);
10
11     system("pause");
12     return 0;
13  }
```

```
/* prog7_4 OUTPUT---
i=1
i=2
i=3
... （無窮迴圈的輸出）
--------------------*/
```

到2147483647後再+1
會到-2147483648
迴圈執行2147483648次



無窮迴圈的範例二

- 利用無窮迴圈持續輸入字元：

```
01  /* prog7_5, 無窮迴圈的應用 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06     char ch;未初始化有ch一開始就ㄎㄧ的風險
07     while(ch!=17)              /* 當按下的鍵不是 Ctrl+q 時 */
08     {
09        ch=getch();             /* 從鍵盤取得字元 */
10        printf("ASCII of ch=%d\n",ch); /* 印出取得字元的 ASCII 碼 */
11     }
12     printf("您已按了 Ctrl+q...\n");
13
14     system("pause");
15     return 0;
16  }
```

```
/* prog7_5 OUTPUT---

ASCII of ch=117
ASCII of ch=104
ASCII of ch=13
ASCII of ch=17
您已按了 Ctrl+q...
------------------*/
```

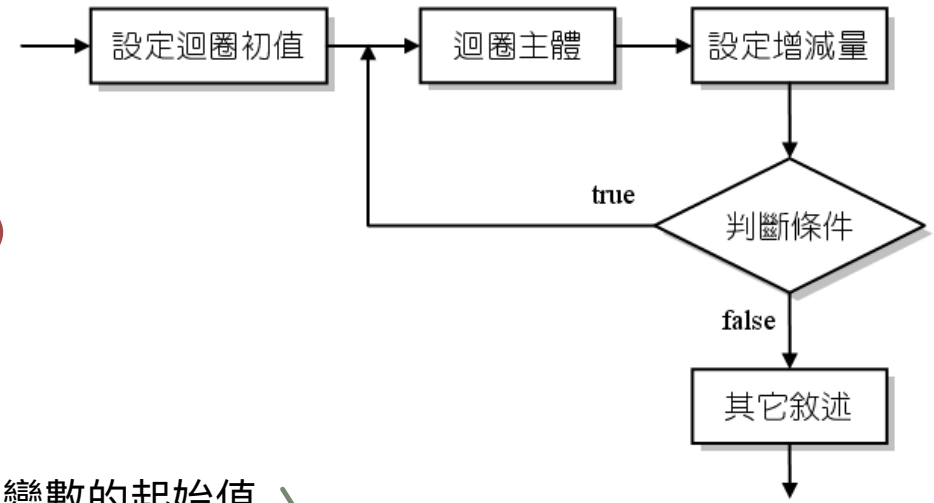
do while迴圈 (1/2)

後驗迴圈

- do while迴圈的格式如下

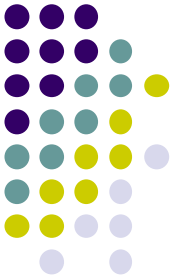
```
設定迴圈初值;
do
{
    迴圈主體;
    設定增減量;
} while(判斷條件);
```

要加分號



- 下面列出do while迴圈執行的流程：
  - 進入do while迴圈前，先設定迴圈控制變數的起始值
  - 迴圈主體執行完畢，才開始根據判斷條件的內容，檢查是否繼續執行迴圈
  - 執行完迴圈主體內的敘述後，重新設定（增加或減少）迴圈控制變數的值

不一定要



do while迴圈 (2/2)

```
01  // prog5_6, do while 迴圈
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  int main(void)
06  {
07      int n,i=1,sum=0;
08      do{
09         cout << "請輸入欲累加的最大奇數值:";
10         cin >> n;
11      }while(n<1 || n%2==0);
12
13      do{
14         sum+=i;
15         i+=2;
16      }while(i<=n);
17      cout << "1+3+...+" << n << "=" << sum << endl;
18
19      system("pause");
20      return 0;
21  }
```

- prog5_6是利用do while迴圈設計的程式

```
/* prog5_6 OUTPUT-------
請輸入欲累加的最大奇數值:-6
請輸入欲累加的最大奇數值:7
1+3+...+7=16
---------------------------*/
```

共轉4圈

空迴圈

- 迴圈主體內沒有任何的敘述，稱為空迴圈

空迴圈的說明

```
for(設定初值;判斷條件;設定增減量)
{ }
```

或是

```
for(設定初值;判斷條件;設定增減量);
```

這兒要加分號

空迴圈的範例

- 下面的範例是一個不做任何事的空迴圈：

```
01  /* prog7_8, 空迴圈的誤用 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06     int i;
07     for(i=1;i<=10000;i++);   /* 空迴圈 */
08        printf("i=%d\n",i);
09
10     system("pause");
11     return 0;
12  }
```

「有;」
∴printf不屬於for迴圈，
不應縮排

```
/* prog7_8 OUTPUT--

i=10001
----------------------*/
```

使用哪一種迴圈？

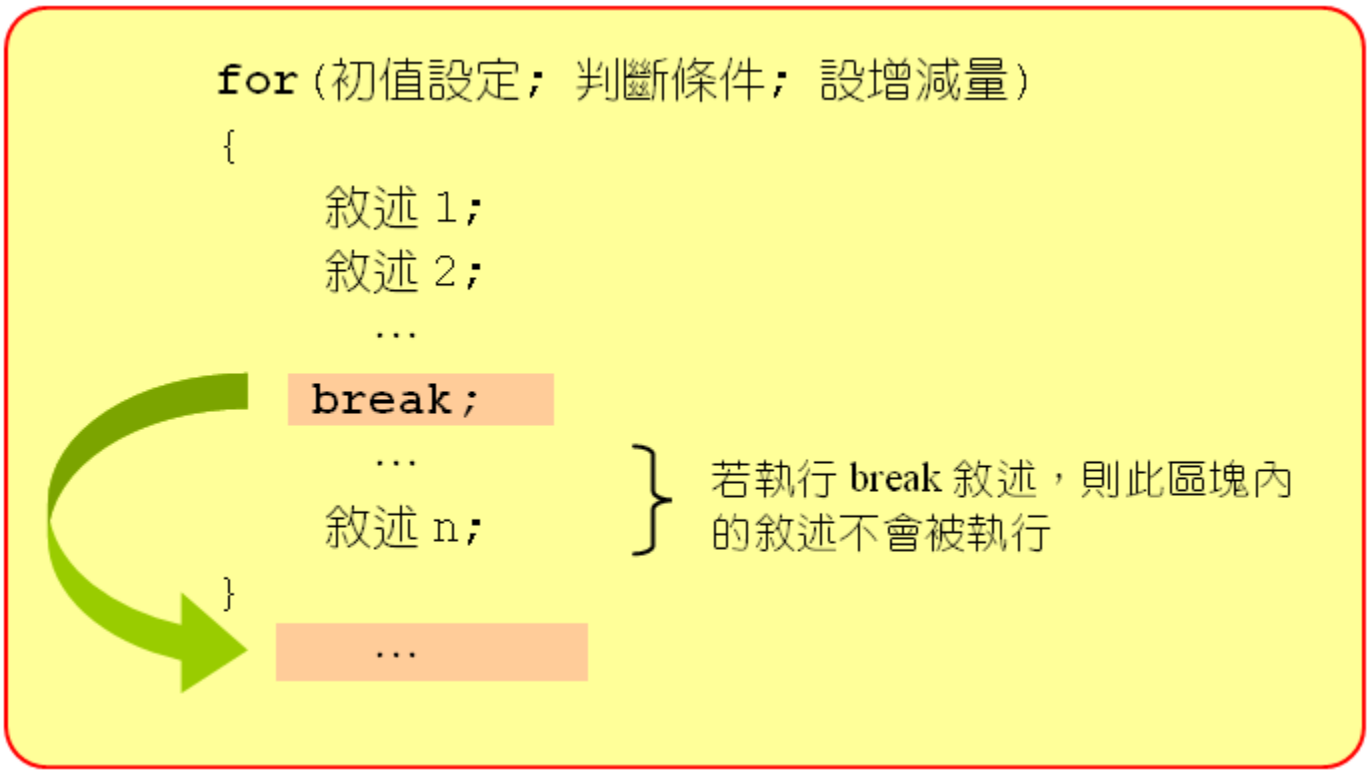
- 下表列出了每一種迴圈的特性比較：

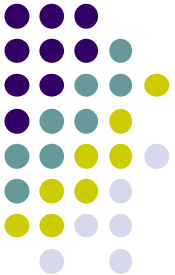
表 7.6.1　for、while 與 do while 迴圈的比較

迴圈特性	迴圈種類		
	for	while	do while
前端測試判斷條件	是	是	否
後端測試判斷條件	否	否	是
於迴圈主體中需要更改控制變數的值	否	是	是
迴圈控制變數會自動變更	是	否	否
迴圈重複的次數	已知	未知	未知
至少執行迴圈主體的次數	0 次	0 次	1 次
何時重複執行迴圈	條件成立	條件成立	條件成立

break敘述 (1/2)

- 以for迴圈為例，程式執行到break，即會離開迴圈主體，到迴圈外層的敘述繼續執行





break敘述 (2/2)

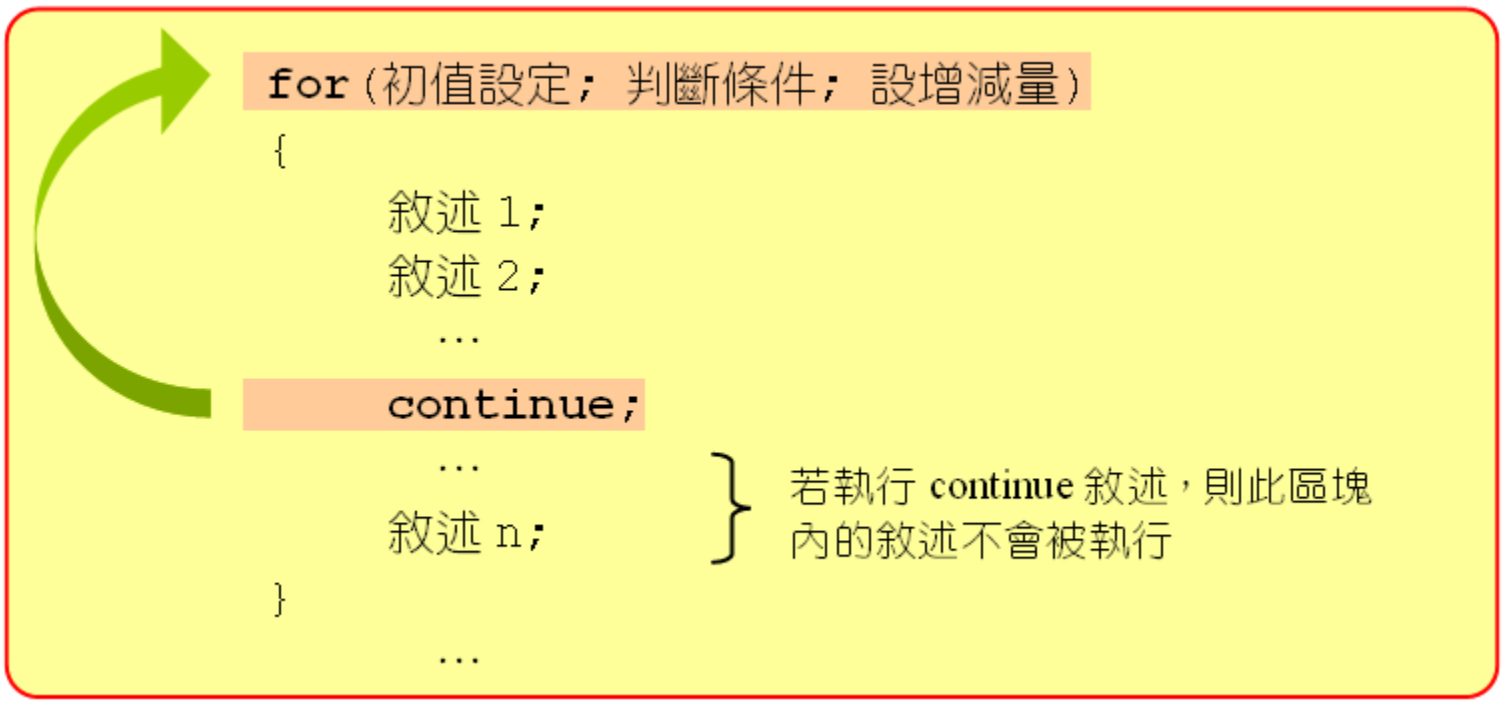
- 下面的程式說明如何使用break敘述跳離迴圈

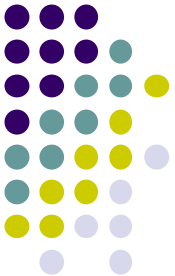
```
01  // prog5_8, break 的使用
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  int main(void)
06  {
07      int i;
08
09      for(i=1;i<=10;i++)
10      {
11         if(i%4==0)
12            break;              // i%4 為 0 時即跳出迴圈
13         cout << "i=" << i << endl;
14      }
15      cout << "when loop interruped,i=" << i << endl;
16      system("pause");
17      return 0;
18  }
```

```
/* prog5_8 OUTPUT---------

i=1
i=2
i=3
when loop interruped,i=4
-------------------------*/
```

continue敘述 (1/2)

- 程式執行到continue，即會回到迴圈的起點，繼續執行迴圈主體的部分敘述：



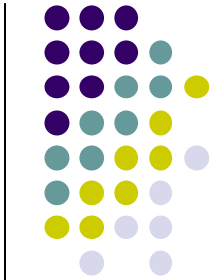


continue敘述 (2/2)

- 下面的範例，可觀察break與continue敘述的不同

```
01  // prog5_9, continue 的使用
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  int main(void)
06  {
07      int i;
08
09      for(i=1;i<=10;i++)
10      {
11          if(i%4==0)
12              continue;          // i%4 為 0 時由迴圈起始處繼續執行
13          cout << "i=" << i << endl;
14      }
15      cout << "when loop interruped,i=" << i << endl;
16      system("pause");
```

```
/* prog5_9 OUTPUT-----------

i=1
i=2
i=3
i=5
i=6
i=7
i=9
i=10
when loop interruped,i=11
--------------------------*/
```



-The End-