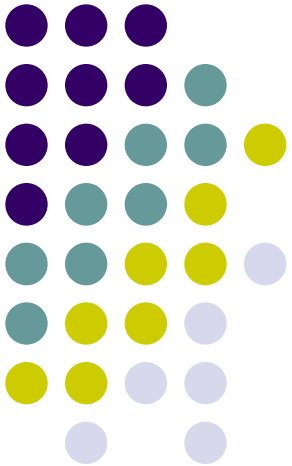
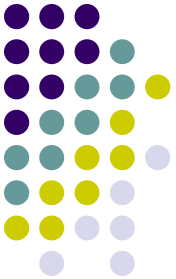
# 第五章
# 選擇性敘述與迴圈

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



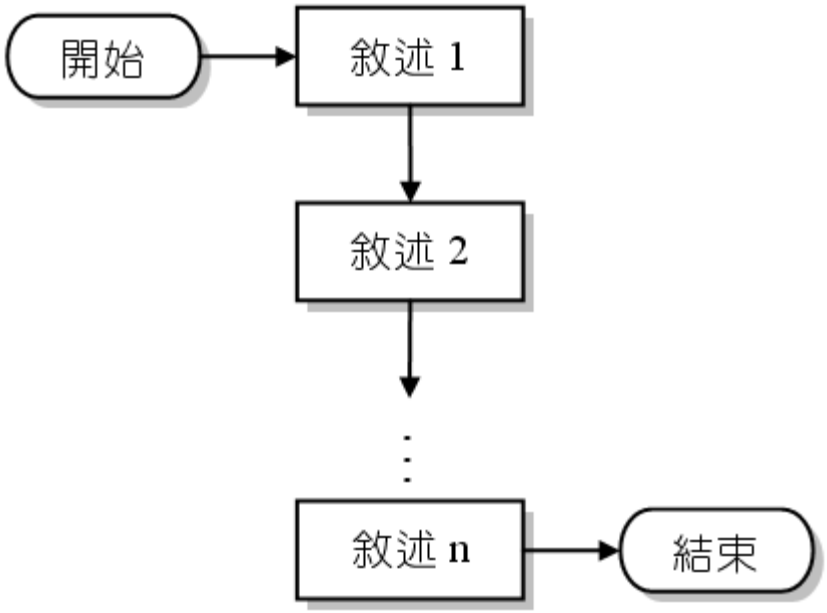


# 程式的結構

- 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）

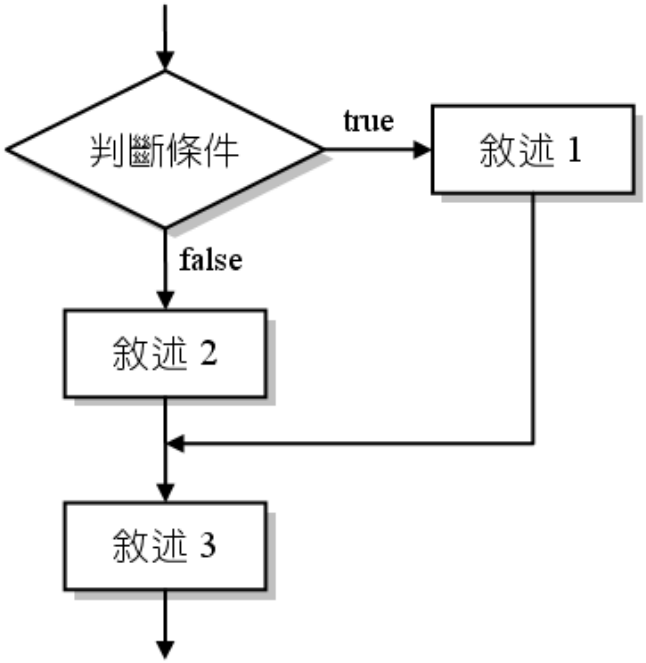
# 循序性結構

- 循序性結構是採上至下（top to down）的敘述方式



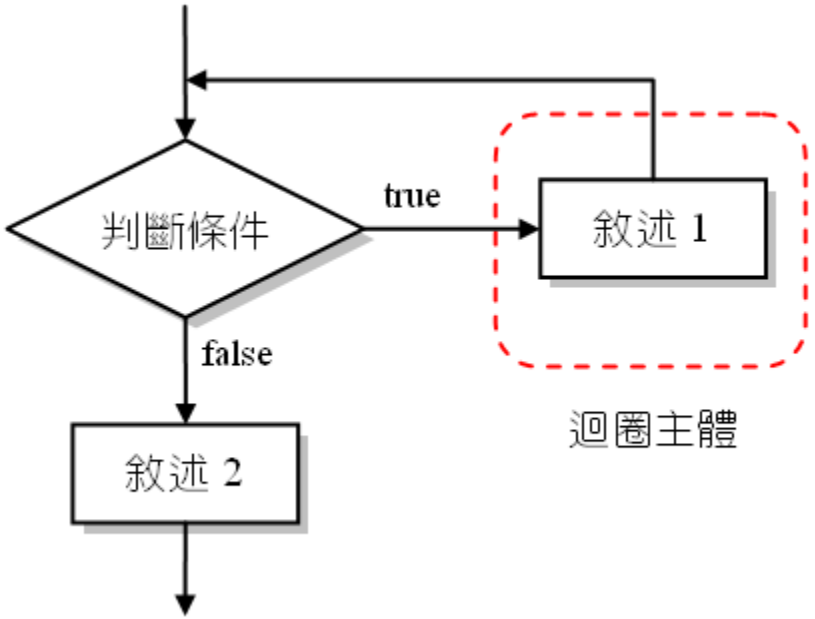
# 選擇性結構

- 選擇性結構根據條件的成立與否，再決定要執行哪些敘述



# 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

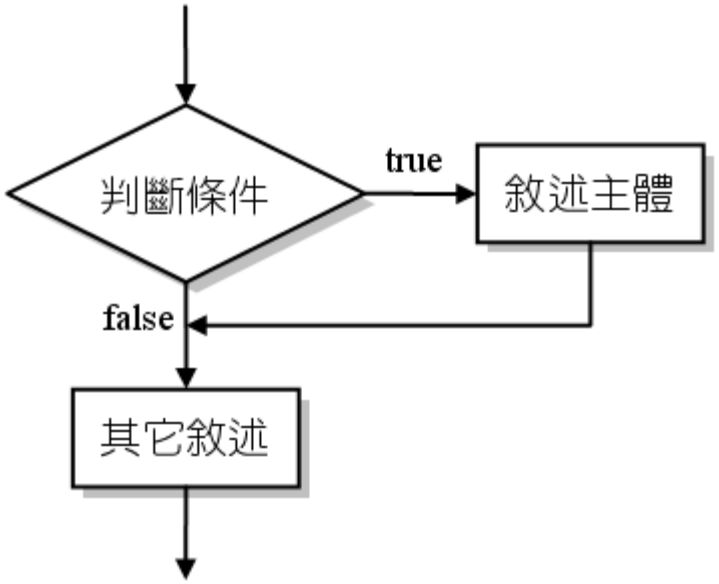


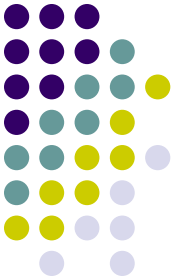

- 重複性結構有for、while及do while三種迴圈

# 選擇性結構與if敘述

- 選擇性結構包括if、if-else及switch敘述
- if敘述的格式如下

```
if(判斷條件)
{
    敘述主體;
}
```





# 關係運算子

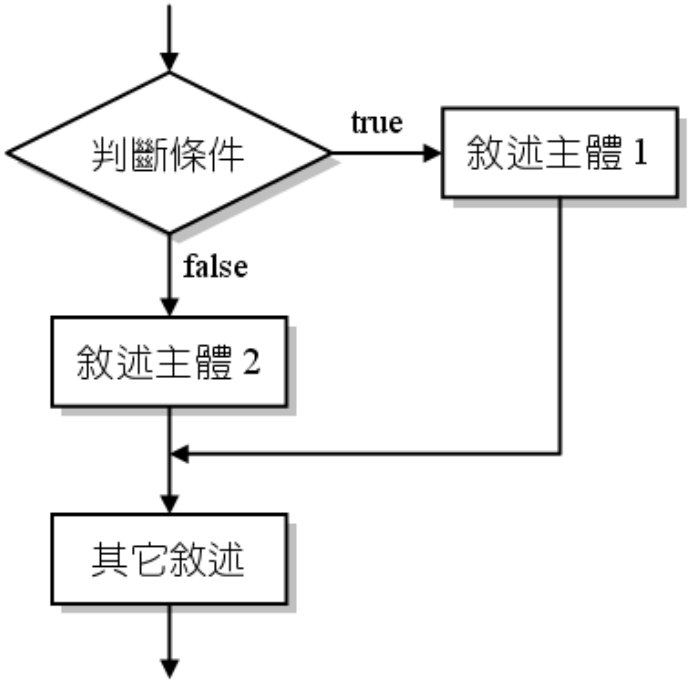
- 「關係運算子」(Relational Operator)是用來比較關係運算子左右兩邊的運算式
- 關係運算子將比較的結果傳回，若比較的結果為真，傳回值為1；比較結果不成立時傳回值為0(代表假)。
- 在C語言中的關係運算子是透過大於、小於或等於運算子組合成下表中的六種狀態，供您在設計程式時使用。

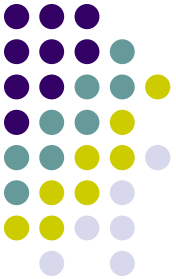
| 運算子 | 說明 | 使用例 | 結果 |
|---|---|---|---|
| ==<br>(相等) | 判斷此運算子左右兩邊運算式的值是否相等。 | 18 == 18 | 1(真) |
| | | 18 == 35 | 0(假) |
| | | 3+2 == 1+4 | 1(真) |
| !=<br>(不相等) | 判斷此運算子左右兩邊運算式的值是否不相等。 | 17 != 18 | 1(真) |
| | | 56 != 56 | 0(假) |
| | | 12*3 != 3*12 | 0(假) |
| <<br>(小於) | 判斷此運算子左邊運算式的值是否小於右邊運算式的值。 | 17 < 18 | 1(真) |
| | | 42 < 30 | 0(假) |
| | | 2 < 10-7 | 1(真) |
| >(大於) | 判斷此運算子左邊運算式的值是否大於右邊運算式的值。 | 19 > 18 | 1(真) |
| | | 26 > 36 | 0(假) |
| | | 12*3 > 12*2 | 1(真) |
| <=<br>(小於等於) | 判斷此運算子左邊運算式的值是否小於等於右邊運算式的值。 | 17 <= 18 | 1(真) |
| | | 18 <= 18 | 1(真) |
| | | 10+3 <= 12 | 0(假) |
| >=<br>(大於等於) | 判斷此運算子左邊運算式的值是否大於等於右邊運算式的值。 | 17 >= 18 | 0(假) |
| | | 18 >= 18 | 1(真) |
| | | 12*3 >= 35 | 1(真) |

# if-else敘述 (1/2)

- **if-else敘述**
  - 當條件成立，即執行if敘述主體；如果不成立，則執行else後面的敘述主體
- **if-else敘述的格式如下**

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





# if-else敘述 (2/2)

- 下面是if-else的範例

```
01  // prog5_1, if-else 敘述
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  int main(void)
06  {
07      int num=42;
08      if(num%3==0 && num%7==0)
09          cout << num << "可以被 3 與 7 整除" << endl;
10      else
11          cout << num << "不能被 3 與 7 整除" << endl;
12      system("pause");
13      return 0;
14  }
```

```
/* prog5_1 OUTPUT---
42 可以被 3 與 7 整除
------------------*/
```

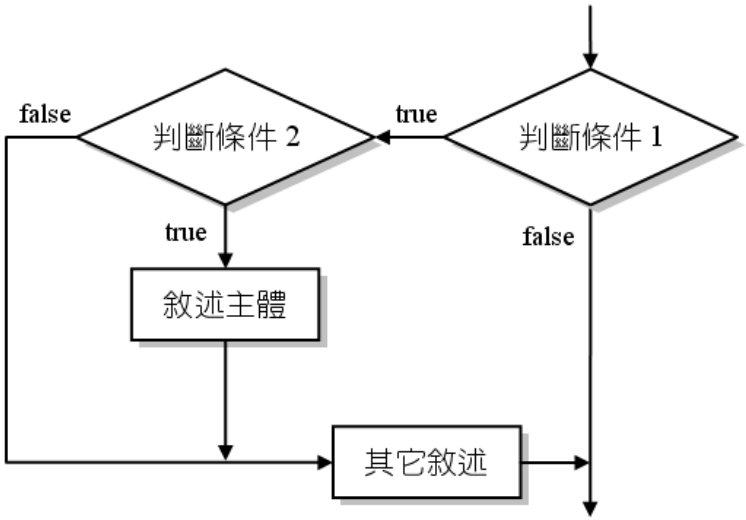
# 更多的選擇—巢狀if敘述

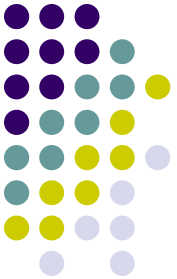
- 巢狀if：當if敘述中又包含其它if敘述

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

若判斷條件 2 成立，則執行這個部份

若判斷條件 1 成立，則執行這個部份





# 條件運算子 (1/2)

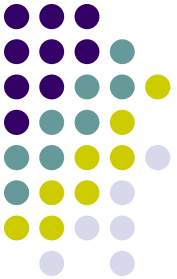
- 條件運算子（conditional operator）可代替if-else敘述

| 條件運算子 | 意義 |
|---|---|
| ?: | 根據條件的成立與否，來決定是哪一個運算式會被執行 |

- 條件運算子的格式如下

```
傳回值 = 判斷條件 ? 運算式1 : 運算式2 ;
```

第一個運算元：判斷條件
第二個運算元：運算式1
第三個運算元：運算式2



# 條件運算子 (2/2)

- 條件運算子的使用範例

```
01  // prog5_2, 條件運算子?:的使用
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  int main(void)
06  {
07      int a=5,b=12,min;
08      min=(a<b)?a:b;          // 利用條件運算子判斷 a 與 b 何者為小
09      cout << "a=" << a << ", b=" << b << endl;
10      cout << min << "是較小的數" << endl;
11
12      system("pause");
13      return 0;
14  }
```

```
/* prog5_2 OUTPUT----

a=5, b=12
5 是較小的數
--------------------*/
```

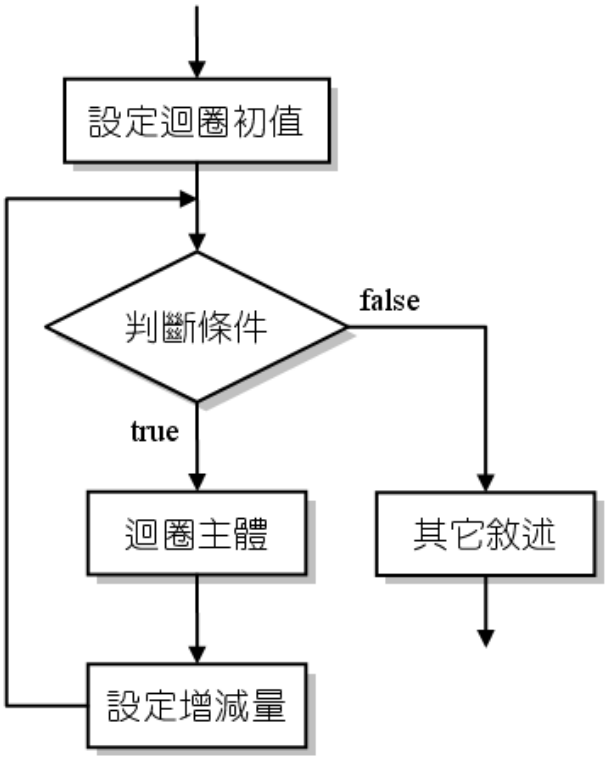
# for迴圈 (1/3)

- for迴圈敘述格式如下

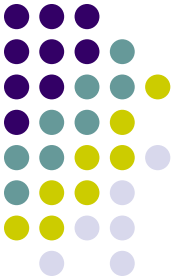
```
for(設定迴圈初值; 判斷條件; 設定增減量)
{
    迴圈主體;
}
```

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

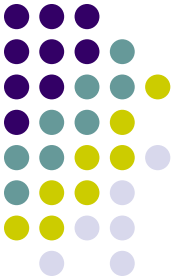


# for迴圈 (2/3)

- 下面的程式可計算由1累加至15的運算結果

```
01  // prog5_3, for 迴圈
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  int main(void)
06  {
07      int i,sum=0;
08      for(i=1;i<=15;i++)
09         sum+=i;
10      cout << "1+2+...+15=" << sum << endl;
11
12      system("pause");
13      return 0;
14  }
```

```
/* prog5_3 OUTPUT---

1+2+...+15=120
------------------*/
```



# 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  }
```

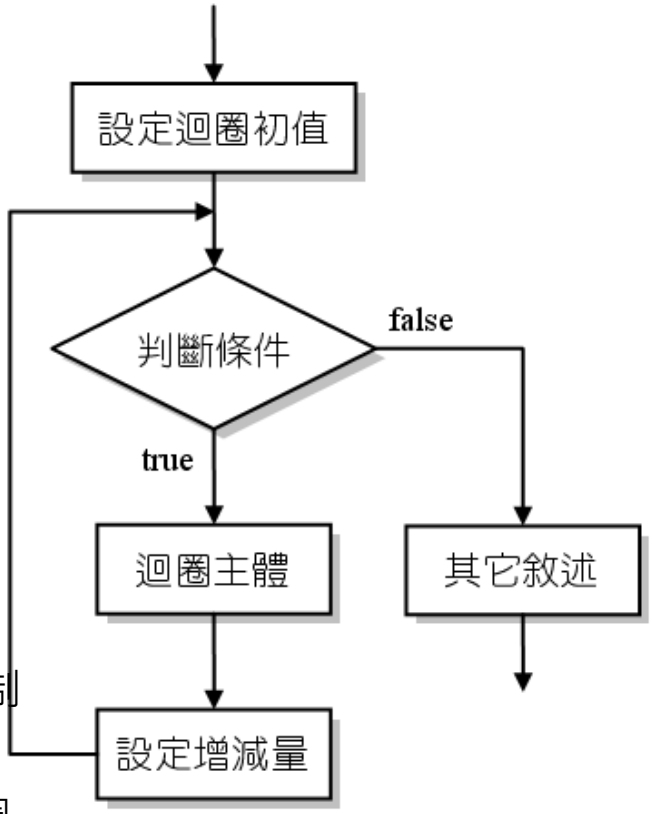
變數 i 的有效範圍

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

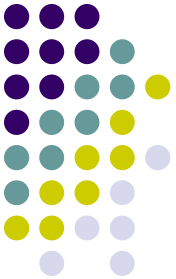
# while迴圈 (1/2)

- 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
  - 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 (2/2)

- 下面的例子是利用while迴圈計算1累加到15

```
01  // prog5_5, while迴圈
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  int main(void)
06  {
07      int num,i=1,sum=0;
08      cout << "請輸入整數值:";
09      cin >> num;
10      while(i<=num)
11      {
12         sum+=i;
13         i++;
14      }
15      cout << "1+2+...+" << num << "=" << sum << endl;
16      system("pause");
17      return 0;
18  }
```

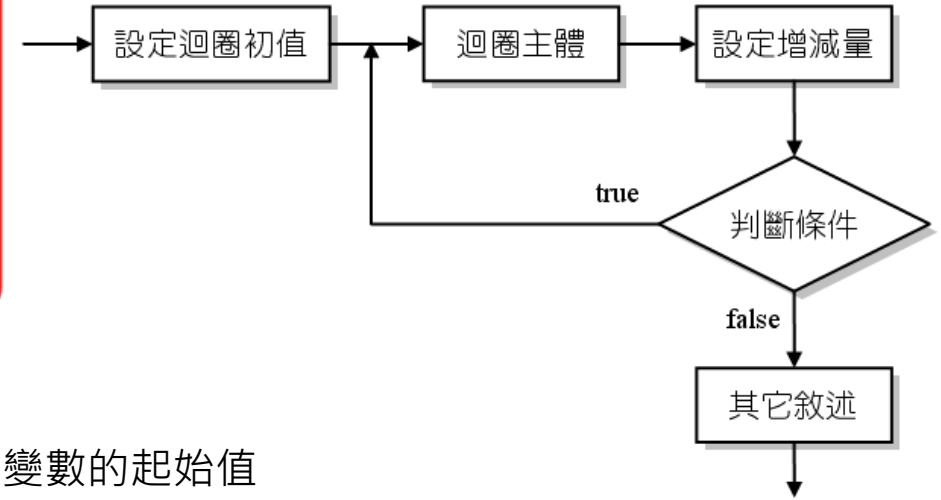
```
/* prog5_5 OUTPUT---
請輸入整數值:10
1+2+...+10=55
------------------*/
```

# do while迴圈 (1/2)

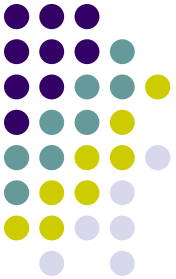
- do while迴圈的格式如下

```
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

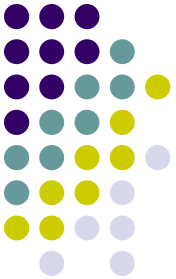


# do while迴圈 (2/2)

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

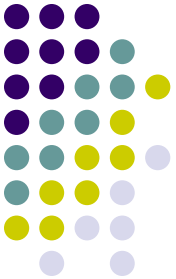


# 巢狀迴圈

- 下面的程式以列印九九乘法表為例，來練習巢狀迴圈

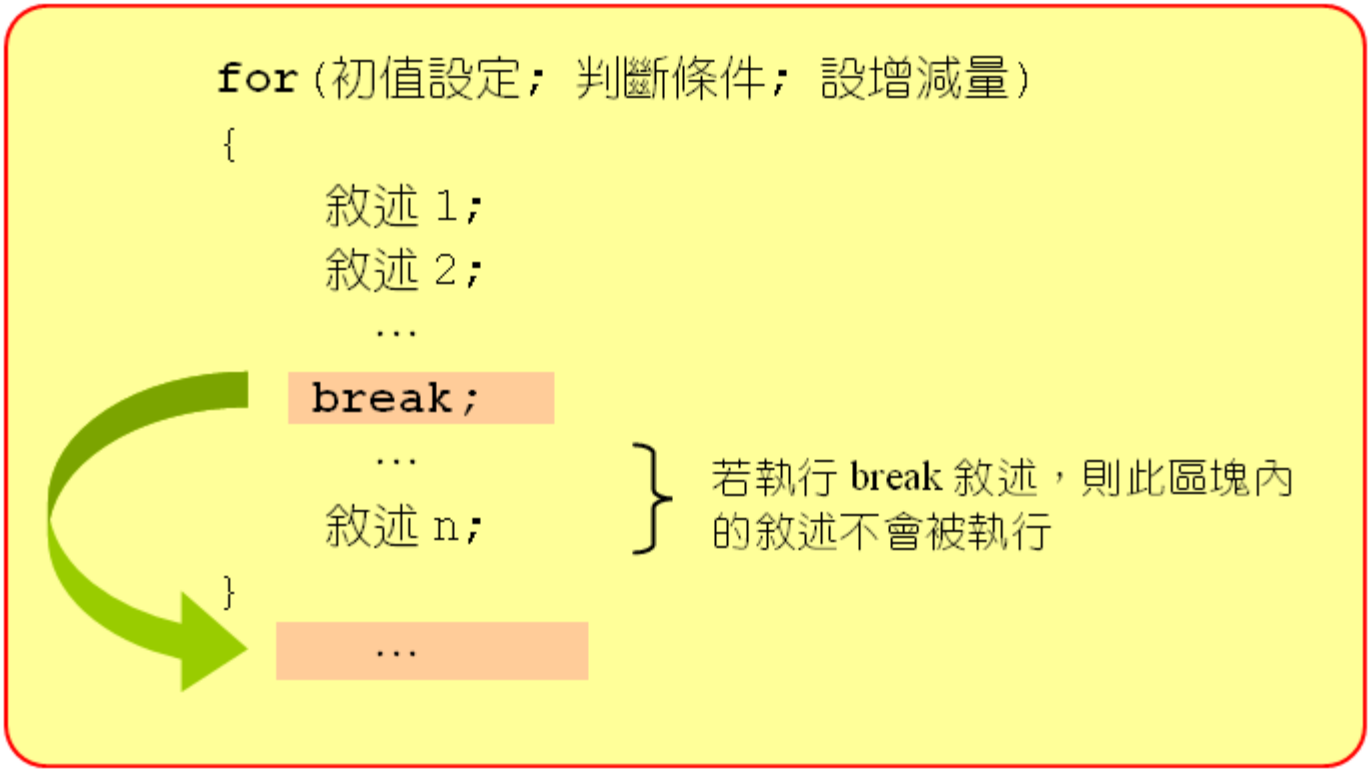
```
01  // prog5_7, 巢狀 for 迴圈求 9*9 乘法表
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  int main(void)
06  {
07      int i,j;
08
09      for(i=1;i<=3;i++)       // 外層迴圈
10      {
11         for(j=1;j<=3;j++)    // 內層迴圈
12            cout << i << "*" << j << "=" << (i*j) << "\t";
13         cout << endl;
14      }
15
16      system("pause");
17      return 0;
18  }
```

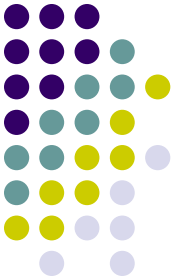
```
/* prog5_7 OUTPUT-----
1*1=1   1*2=2   1*3=3
2*1=2   2*2=4   2*3=6
3*1=3   3*2=6   3*3=9
--------------------*/
```



# 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



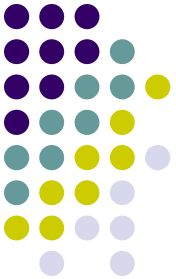


# break敘述 (2/2)

- 下面的程式說明如何使用break敘述跳離迴圈

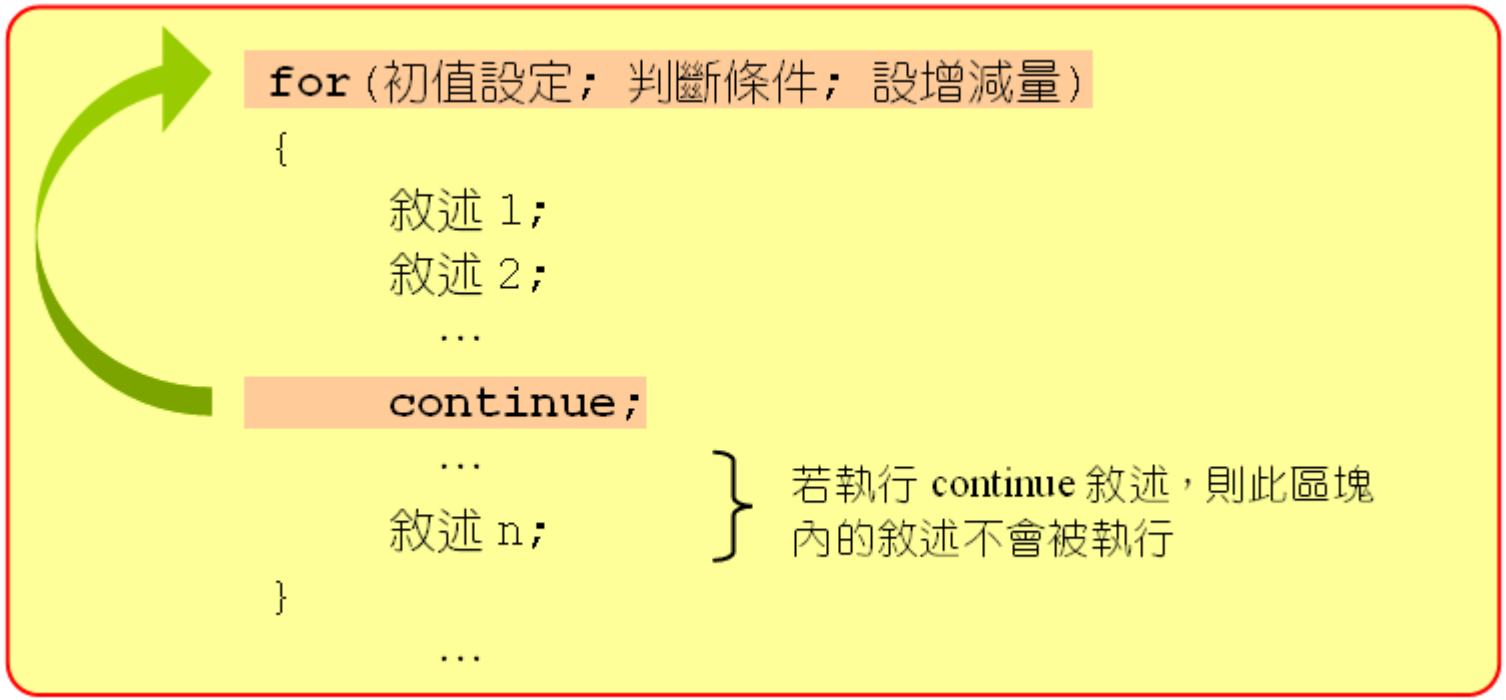
```
01  // prog5_8, break 的使用
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  int main(void)
06  {
07      int i;
08
09      for(i=1;i<=10;i++)
10      {
11         if(i%4==0)
12            break;              // i%4 為 0 時即跳出迴圈
13         cout << "i=" << i << endl;
14    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

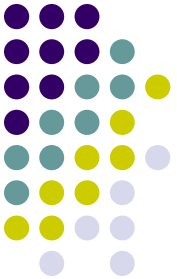
```
/* prog5_8 OUTPUT---------

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

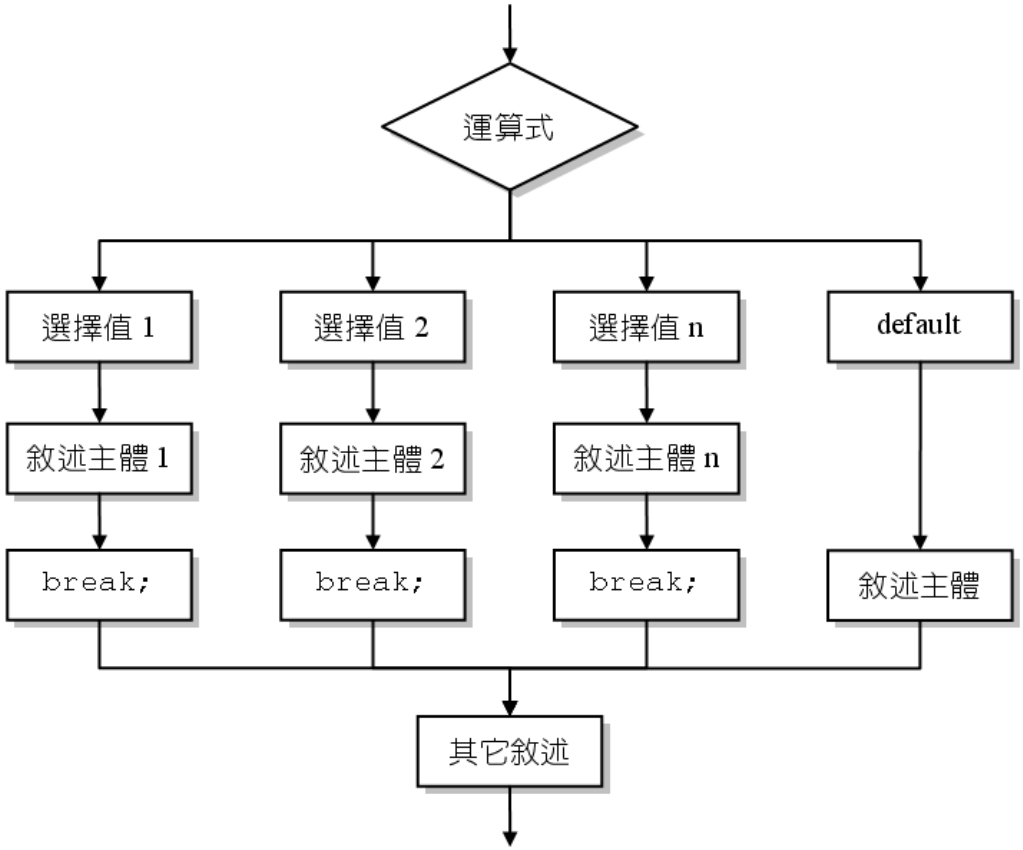
```
01  // prog5_9, continue 的使用
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  int main(void)
06  {
07      int i;
08
09      for(i=1;i<=10;i++)
10      {
11          if(i%4==0)
12              continue;          // i%4 為 0 時由迴圈起始處繼續執行
13          cout << "i=" << i << endl;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

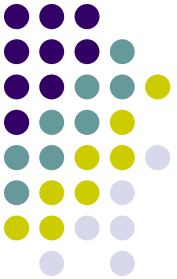
```
/* prog5_9 OUTPUT-----------

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

# switch敘述 (1/3)

- switch敘述的格式如下

```
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;
              …
    case 選擇值 n:
            敘述主體 n;
            break;
    default:
            敘述主體;
}
```



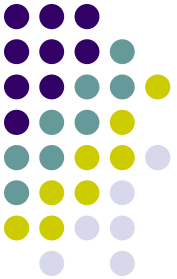


# switch敘述 (2/3)

- witch敘述執行的流程
  - switch敘述先計算括號中運算式的運算結果
  - 如果某個case的選擇值符合運算式的結果，就會執行該case所包含的敘述，直到執行至break敘述後才跳離整個switch敘述
  - 若是所有case的選擇值皆不適合，則執行default之後所包含的敘述，執行完畢即離開switch敘述
  - 如果沒有定義default的敘述，則直接跳離switch敘述

# switch敘述 (3/3)





- 下面的程式是利用switch敘述撰寫而成的

```
01  // prog5_10, switch 敘述
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  int main(void)
06  {
07      int month=11;
08
09      cout << month << "月是";
10      switch(month)
11      {
12         case 3:
13         case 4:
14         case 5: cout << "春天" << endl;
15                 break;
16         case 6:
17         case 7:
18         case 8: cout << "夏天" << endl;
19                 break;
20         case 9:
21         case 10:
22         case 11:cout << "秋天" << endl;
23                 break;
24         case 12:
25         case 1:
26         case 2:cout << "冬天" << endl;
27                break;
28         default:
29                cout << "不存在!" << endl;
30      }
31      system("pause");
32      return 0;
33  }
```

```
/* prog5_10 OUTPUT---
5 月是春天
----------------------*/
```


-The End-