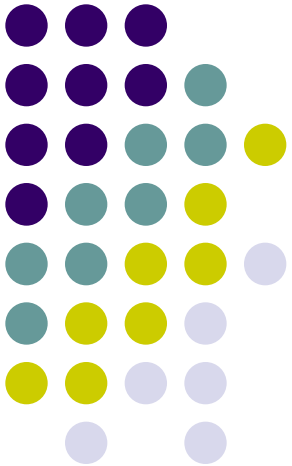
類別的進階認識

認識建構元及其引數的使用

傳遞物件到函數與使用物件陣列

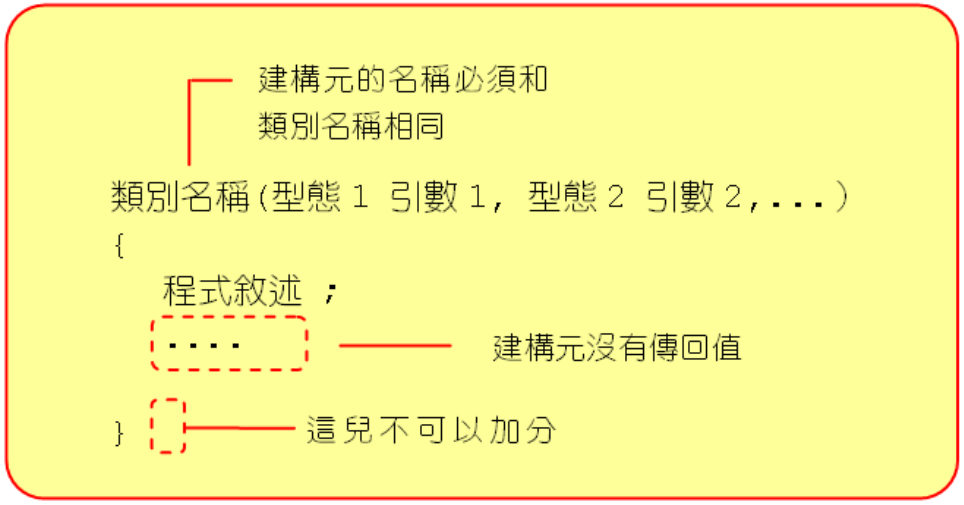
使用類別裡的靜態成員

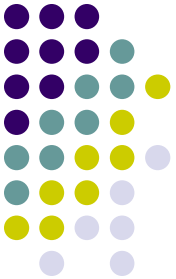
學習使用指向物件的指標與參照



建構元的基本認識

- 建構元的主要目的，是用來設定物件的初值。
- 建構元的定義格式如下



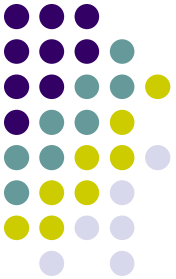


建構元的使用範例 (1/3)

- 下面的例子說明建構元的使用方式

```
01  // prog13_1, 建構元的使用
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  class CWin
06  {
07     private:
08       char id;
09       int width, height;
10
11     public:
12       CWin(char i,int w,int h)        // CWin()建構元,可接收三個引數
13       {
14          id=i;
15          width=w;        } 設定資料成員的初值
16          height=h;
17          cout << "CWin 建構元被呼叫了..." << endl;
18       }
```

```
/* prog13_1 OUTPUT--------------

CWin 建構元被呼叫了...
CWin 建構元被呼叫了...
Window A: width=50, height=40
Window B: width=60, height=70
--------------------------------*/
```



建構元的使用範例 (2/3)

```
19        void show member(void)          // 成員函數，用來顯示資料成員的值
20        {
21           cout << "Window " << id << ": ";
22           cout << "width=" << width << ", height=" << height << endl;
23        }
24    };
25
26    int main(void)
27    {
28       CWin win1('A',50,40);            // 宣告 win1 物件,並設定初值
29       CWin win2('B',60,70);            // 宣告 win2 物件,並設定初值
30
31       win1.show member();
32       win2.show member();
33
34       system("pause");
35       return 0;
36    }
```

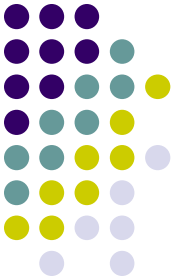
```
/* prog13_1 OUTPUT--------------

CWin 建構元被呼叫了...
CWin 建構元被呼叫了...
Window A: width=50, height=40
Window B: width=60, height=70
-------------------------------*/
```

建構元的使用範例 (3/3)

- 建構元在建立物件時便會自動執行

```
class CWin
{  ...
   CWin(char i,int w,int h) // CWin()建構元
   {
       id=i;
       width=w;
       height=h;
       cout << "CWin 建構元被呼叫了..." <<endl;
   }
}
int main(void)
{
   CWin win1('A',50,40);
   CWin win2('B',60,70);
      ...
}
```

在建立 win1 與 win2 物件時，CWin()建構元便會自動呼叫，並傳遞相關的引數

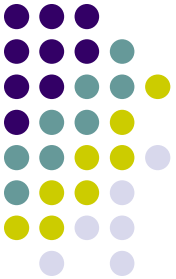


建構元的位置 (1/2)

- 下面的範例把建構元的定義移到CWin類別的外面

```
01  // prog13_2, 將建構元的定義移到類別外面
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  class CWin
06  {
07     private:
08       char id;
09       int width, height;
10
11     public:
12       CWin(char,int,int);              // CWin 建構元的原型
13
14       void show member(void)           // 成員函數，用來顯示資料成員的值
15       {
16          cout<< "Window " << id << ": ";
17          cout<< "width=" << width << ", height=" << height << endl;
18       }
19  };
20
```

```
/* prog13_2 OUTPUT--------------

CWin 建構元被呼叫了...
CWin 建構元被呼叫了...
Window A: width=50, height=40
Window B: width=60, height=70
--------------------------------*/
```



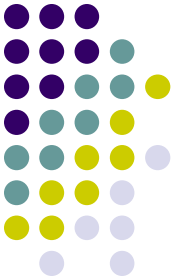
建構元的位置 (2/2)

```
21  CWin::CWin(char i,int w,int h)     // CWin 建構元的定義
22  {
23     id=i;
24     width=w;
25     height=h;
26     cout << "CWin 建構元被呼叫了..." <<endl;
27  }
28
29  int main(void)
30  {
31     CWin win1('A',50,40);
32     CWin win2('B',60,70);
33
34     win1.show member();
35     win2.show member();
36
37     system("pause");
38     return 0;
39  }
```

```
/* prog13_2 OUTPUT-------------
CWin 建構元被呼叫了...
CWin 建構元被呼叫了...
Window A: width=50, height=40
Window B: width=60, height=70
--------------------------------*/
```

範疇解析運算子，用來表示 CWin()建構元是屬於 CWin 類別

```
CWin::CWin(char i,int w,int h)
{
    ...
}
```

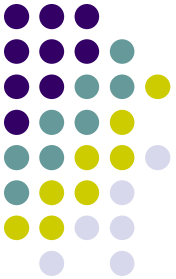


建構元的多載 (1/2)

- 建構元可多載（overloading），也就是它可依據引數數目的不同呼叫正確的建構元
- 下面的程式將建構元CWin() 多載成兩個版本

```
01  // prog13_3, 建構元的多載
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  class CWin
06  {
07     private:
08       char id;
09       int width, height;
10
11     public:
12       CWin(char i,int w,int h)        // 有三個引數的建構元
13       {
14          id=i;
15          width=w;
16          height=h;
17          cout << "CWin(char,int,int) 建構元被呼叫了..." << endl;
18       }
```

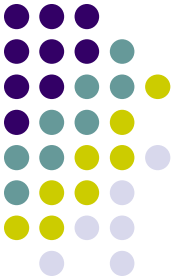
```
/* prog13_3 OUTPUT-----------------
CWin(char,int,int) 建構元被呼叫了...
CWin(int,int) 建構元被呼叫了...
Window A: width=50, height=40
Window Z: width=80, height=120
-----------------------------------*/
```



建構元的多載 (2/2)

```
19      CWin(int w,int h)                  // 只有兩個引數的建構元
20      {
21         id='Z';
22         width=w;
23         height=h;
24         cout << "CWin(int,int) 建構元被呼叫了..." << endl;
25      }
26      void show member(void)             // 成員函數，用來顯示資料成員的值
27      {
28         cout<< "Window " << id << ": ";
29         cout<< "width=" << width << ", height=" << height << endl;
30      }
31  };
32
33  int main(void)
34  {
35     CWin win1('A',50,40);          // 建立 win1 物件，並呼叫三個引數的建構元
36     CWin win2(80,120);             // 建立 win2 物件，並呼叫二個引數的建構元
37
38     win1.show member();
39     win2.show member();
40
41     system("pause");
42     return 0;
43  }
```

```
/* prog13_3 OUTPUT------------------
CWin(char,int,int) 建構元被呼叫了...
CWin(int,int) 建構元被呼叫了...
Window A: width=50, height=40
Window Z: width=80, height=120
-------------------------------------*/
```



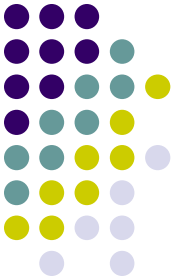
預設建構元 (1/5)

- 在prog13_3中，如果把main() 改寫成如下的程式碼

```
01  int main(void)
02  {
03     CWin win1('A',50,40);
04     CWin win2(80,120);
05     CWin win3;                    // 只建立物件，但未呼叫特定的建構元
06        ...
07  }
```

編譯會錯誤，因為編譯器找不到 "沒有引數" 的建構元

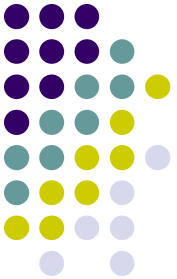
- 如果沒有撰寫建構元，C++會提供一個沒有引數的預設建構元
- 如果程式裡有撰寫建構元，C++不會再提供沒有引數的預設建構元



預設建構元 (2/5)

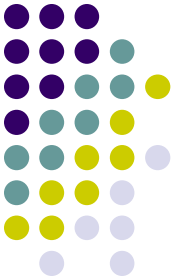
- prog13_4是加入一個預設建構元的完整範例

```
01  // prog13_4, 預設的建構元
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  class CWin                              // 定義視窗類別 CWin
06  {
07     private:
08       char id;
09       int width, height;
10
11     public:
12       CWin(char i,int w,int h)           // CWin 建構元
13       {
14          id=i;
15          width=w;
16          height=h;
17          cout << "CWin(char,int,int) 建構元被呼叫了..." << endl;
18       }
```



預設建構元 (3/5)

```
19        CWin(int w,int h)                    // CWin 建構元
20        {
21           id='Z';
22           width=w;
23           height=h;
24           cout << "CWin(int,int) 建構元被呼叫了..." << endl;
25        }
26        CWin()                               // 沒有引數的(預設)建構元
27        {
28           id='D';
29           width=100;
30           height=100;
31           cout << "預設建構元被呼叫了..." <<endl;
32        }
33        void show member(void)               // 成員函數，用來顯示資料成員的值
34        {
35           cout << "Window " << id << ": ";
36           cout << "width=" << width << ", height=" << height << endl;
37        }
38    };
39
```



預設建構元 (4/5)

```
40  int main(void)
41  {
42     CWin win1('A',50,40);
43     CWin win2(80,120);
44     CWin win3;                          // 此行會呼叫預設建構元
45
46     win1.show member();
47     win2.show member();
48     win3.show member();
49
50     system("pause");
51     return 0;
52  }
```

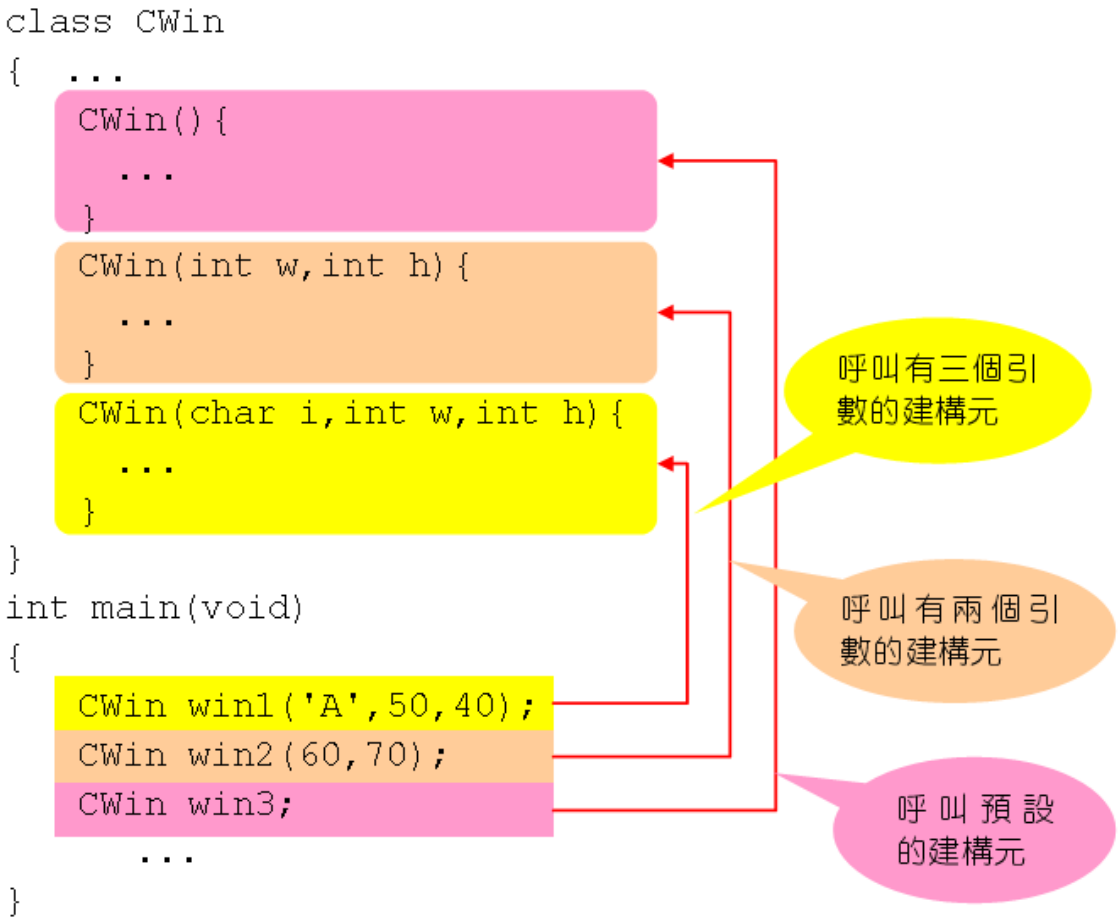
```
/* prog13_4 OUTPUT------------------
CWin(char,int,int) 建構元被呼叫了...
CWin(int,int) 建構元被呼叫了...
預設建構元被呼叫了...
Window A: width=50, height=40
Window Z: width=80, height=120
Window D: width=100, height=100
-------------------------------------*/
```

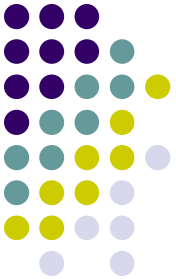
注意不要在 win3 後面加上空的括號，如下面的敘述：

```
CWin win3();      // 錯誤， win3 後面不能加上空的括號
```

預設建構元 (5/5)

- 下圖說明於prog13_4中，建構元呼叫的情形





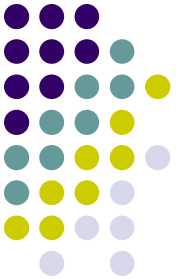
預設值的設定 (1/3)

- 若是把CWin() 建構元的定義寫成

```
CWin(char i='D',int w=100,int h=100)
{
    ...
}
```

- 下面是幾種可能的情況

```
CWin win1('A',20);      // 省略 h，因此以其預設值 100 來設定
CWin win1('A');         // 省略引數 h 與 w，則以其預設值來設定
CWin win1;              // 所有的引數均用預設值
CWin win1();            // 錯誤，不能加上括號
```

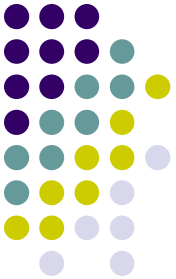


預設值的設定 (2/3)

- 下面是建構元引數預設值的使用範例

```
01  // prog13_5,建構元引數的預設值
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  class CWin                          // 定義視窗類別 CWin
06  {
07     private:
08       char id;
09       int width, height;
10
11     public:
12       CWin(char i='D',int w=100,int h=100)        // 引數的預設值
13       {
14          id=i;
15          width=w;
16          height=h;
17       }
```

```
/* prog13_5 OUTPUT----------------

Window A: width=50, height=40
Window B: width=80, height=100
Window D: width=100, height=100
-----------------------------------*/
```



預設值的設定 (3/3)

```
18        void show member(void)    // 成員函數，用來顯示資料成員的值
19        {
20           cout << "Window " << id << ": ";
21           cout << "width=" << width << ", height=" << height << endl;
22        }
23    };
24
25    int main(void)
26    {
27       CWin win1('A',50,40);          // 自行設定所有的資料成員
28       CWin win2('B',80);             // 只有 height 成員使用預設值
29       CWin win3;                     // 所有的值都使用預設值
30
31       win1.show member();
32       win2.show member();
33       win3.show member();
34
35       system("pause");
36       return 0;
37    }
```

```
/* prog13_5 OUTPUT----------------

Window A: width=50, height=40
Window B: width=80, height=100
Window D: width=100, height=100
-----------------------------------*/
```

於建構元裡初始化成員的技巧

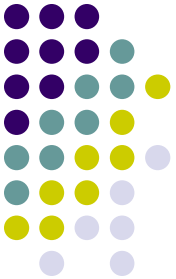
- 在CWin() 建構元內進行成員的初始化

```
CWin(char i='D',int w=100,int h=100): id(i),width(w),height(h)
{
   // 撰寫在建構元的程式碼
}
```

這種設定法相當於設定

```
id=i;
width=w;
height=h;
```

- 用虛線框起來的部分稱為「初始化串列」（initialization list）
- 初始化串列和CWin() 建構元的引數必須以冒號隔開來

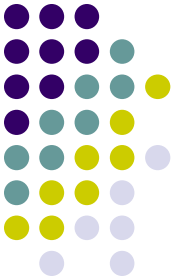


初始化串列的設值 (1/2)

- 下面的範例是使用「初始化串列」的技巧來初始化成員

```
01  // prog13_6, 使用初始化串列來設定初值
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  class CWin
06  {
07     private:
08       char id;
09       int width, height;
10
11     public:
12       CWin(char i='D',int w=100,int h=100):id(i),width(w),height(h)
13       {
14          cout << "成員已被初始化了" << endl;
15       }
16       void show member(void)          // 成員函數，用來顯示資料成員的值
17       {
18          cout << "Window " << id << ": ";
19          cout << "width=" << width << ", height=" << height << endl;
20       }
21  };
```

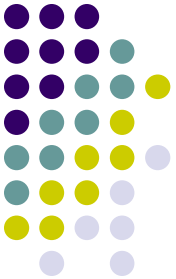
```
/* prog13_6 OUTPUT-----------------
成員已被初始化了
成員已被初始化了
Window A: width=80, height=100
Window D: width=100, height=100
-----------------------------------*/
```



初始化串列的設值 (2/2)

```
22
23  int main(void)
24  {
25     CWin win1('A',80);                    // 建立 win1 物件
26     CWin win2;                            // 建立 win2 物件
27
28     win1.show member();
29     win2.show member();
30
31     system("pause");
32     return 0;
33  }
```

```
/* prog13_6 OUTPUT-----------------

成員已被初始化了
成員已被初始化了
Window A: width=80, height=100
Window D: width=100, height=100
-----------------------------------*/
```



設定引數預設值的注意事項

- 不能同時定義引數皆設有預設值的建構元，以及不需引數的建構元，否則編譯時將發生錯誤

```
CWin(char i='D',int w=100,int h=100)    // 引數皆設有預設值的建構元
{
    cout << "建構元被呼叫了" << endl;
}

CWin()                                  // 不需引數的建構元
{
    cout << "建構元被呼叫了" << endl;
}
```

- 若在主程式裡有這麼一行敘述，則編譯時將發生錯誤

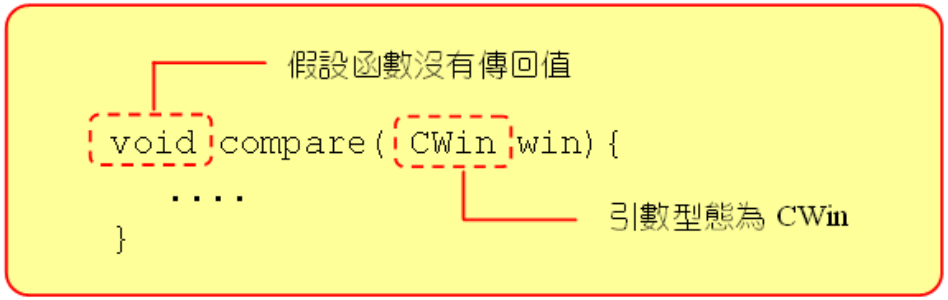
```
CWin win1;    // 錯誤，編譯器不知要呼叫哪一個建構元
```

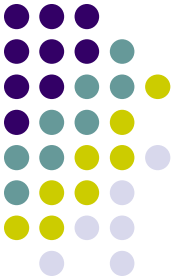
傳遞物件到函數裡 (1/3)

- 假設compare() 函數是用來比較呼叫它的物件win1與compare() 裡的引數win2的面積，可用如下的語法

```
win1.compare(win2);        // 比較物件 win1 與 win2 的面積
```

- compare() 函數的定義必須以如下的語法來撰寫



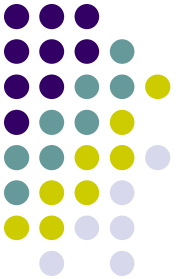


傳遞物件到函數裡 (2/3)

- 下面的範例說明如何傳遞引數類別型態的變數

```
01  // prog13_7, 傳遞物件到函數裡
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  class CWin                              // 定義視窗類別 CWin
06  {
07    private:
08      char id;
09      int width, height;
10
11    public:
12      CWin(char i,int w,int h):id(i),width(w),height(h)   // 建構元
13      {}
14
15      void compare(CWin win)
16      {
17        if(this->area() > win.area())
18          cout << "Window " << this->id << " is larger" << endl;
19        else
20          cout << "Window " << win.id << " is larger" << endl;
21      }
```

```
/* prog13_7 OUTPUT---
Window A is larger
--------------------*/
```



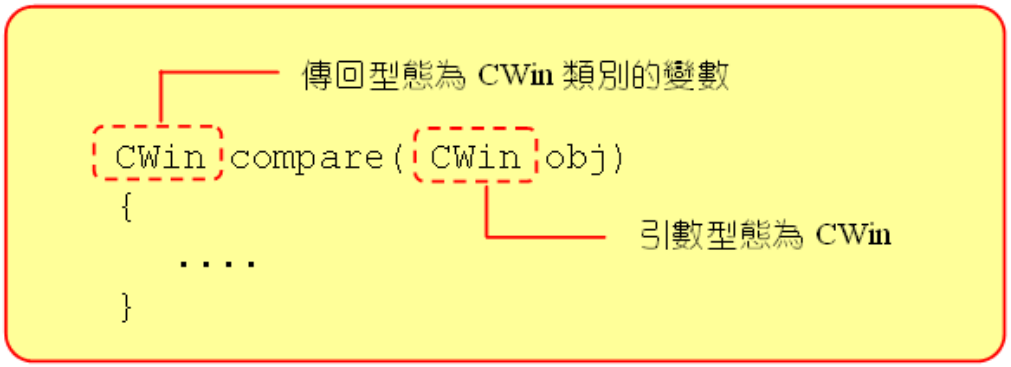
傳遞物件到函數裡 (3/3)

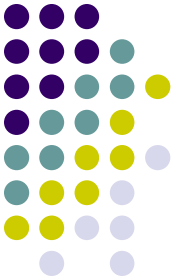
```
22        int area(void)                    // 成員函數，用來顯示資料成員的值
23        {
24           return width*height;
25        }
26    };
27
28    int main(void)
29    {
30       CWin win1('A',70,80);              // 建立 win1 物件
31       CWin win2('B',60,90);              // 建立 win2 物件
32
33       win1.compare(win2);
34
35       system("pause");
36       return 0;
37    }
```

```
/* prog13_7 OUTPUT---
Window A is larger
--------------------*/
```

由函數傳回物件 (1/3)

- 想設計compare() 函數，可傳遞物件到函數內，經比較後傳回面積較大的物件，則函數宣告的語法如下



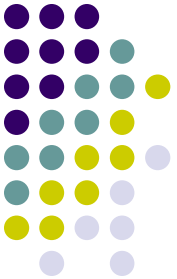


由函數傳回物件 (2/3)

- 下面的範例是compare() 函數實際的撰寫

```
01  // prog13_8, 由函數傳回類別型態的變數
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  class CWin                              // 定義視窗類別 CWin
06  {
07    private:
08      char id;
09      int width, height;
10
11    public:
12      CWin(char i,int w,int h):id(i),width(w),height(h)   // 建構元
13      {}
14
15      CWin compare(CWin win)
16      {
17        if(this->area() >= win.area())
18          return *this;                   // 傳回呼叫 compare() 的物件
19        else
20          return win;                     // 傳回 compare() 所接收的引數
21      }
```

```
/* prog13_8 OUTPUT---
Window A is larger
--------------------*/
```



由函數傳回物件 (3/3)

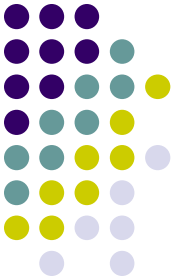
```
22        int area(void)
23        {
24           return width*height;
25        }
26        char get_id(void)                  // 成員函數，顯示資料成員 id 的值
27        {
28           return id;
29        }
30    };
31
32    int main(void)
33    {
34       CWin win1('A',70,80);
35       CWin win2('B',60,90);
36
37       cout << "Window " << (win1.compare(win2)).get_id();
38       cout << " is larger" << endl;
39
40       system("pause");
41       return 0;
42    }
```

傳回面積較大的物件

```
(win1.compare(win2)).get_id()
```

呼叫 get_id()函數，取得面積較大之物件的 id 成員

```
/* prog13_8 OUTPUT---
Window A is larger
--------------------*/
```

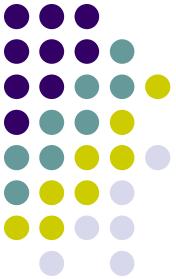


建立物件陣列 (1/2)

- 下面的程式碼是物件陣列的使用範例

```
01  // prog13_9, 建立物件陣列
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  class CWin
06  {
07     private:
08       char id;
09       int width, height;
10
11     public:
12       CWin(char i='D',int w=100,int h=100):id(i),width(w),height(h)
13       {
14          cout << "建構元被呼叫了..." << endl;
15       }
16       void show member(void)    // 成員函數，用來顯示資料成員的值
17       {
18          cout << "Window " << id << ": ";
19          cout<< "width=" << width << ", height=" << height << endl;
20       }
21  };
```

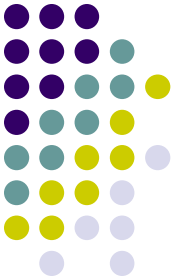
```
/* prog13_9 OUTPUT---------------

建構元被呼叫了...
建構元被呼叫了...
建構元被呼叫了...
建構元被呼叫了...
Window A: width=50, height=40
Window D: width=100, height=100
---------------------------------*/
```



建立物件陣列 (2/2)

```
22
23  int main(void)
24  {
25      CWin win1('A',50,40);
26      CWin my win[3];                 // 建立 3 個 CWin 型態的物件
27
28      win1.show member();             // 以 win1 物件呼叫 show member()
29      my win[2].show member();        // 以my win[2]物件呼叫 show member()
30
31      system("pause");
32      return 0;
33  }
```

```
/* prog13_9 OUTPUT---------------
建構元被呼叫了...
建構元被呼叫了...
建構元被呼叫了...
建構元被呼叫了...
Window A: width=50, height=40
Window D: width=100, height=100
--------------------------------*/
```

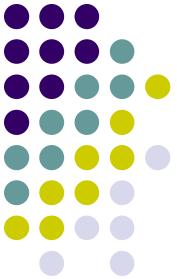


傳遞物件陣列到函數裡 (1/3)

- 我們也可以傳遞物件陣列到函數裡，如下面的程式

```
01  // prog13_10, 傳遞物件陣列到函數裡
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  class CWin                                    // 定義視窗類別 CWin
06  {
07     private:
08       char id;
09       int width, height;
10
11     public:
12       void set member(char i,int w,int h)
13       {
14          id=i;
15          width=w;
16          height=h;
17       }
18       int area(void)
19       {
20          return width*height;
21       }
22       friend void largest(CWin [], int);      // 友誼函數的原型
23  };
```

```
/* prog13_10 OUTPUT---
largest window= A
----------------------*/
```



傳遞物件陣列到函數裡 (2/3)

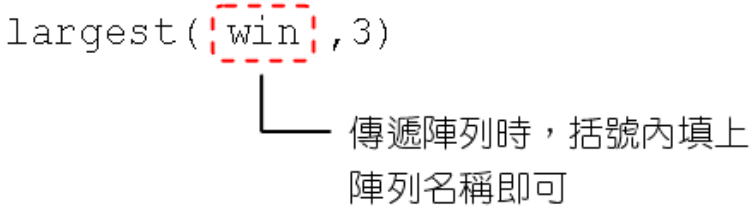
```
24
25  void largest(CWin win[], int n)              // 定義友誼函數 largest
26  {
27     int max=0,iw;
28     for(int i=0; i<n; i++)
29        if(win[i].area()>max)                  // 判別面積是否比 max 大
30        {
31           iw=i;
32           max=win[i].area();
33        }
34     cout << "largest window= " << win[iw].id << endl; // 印出 id 成員
35  }
36
37  int main(void)
38  {
39     CWin win[3];
40
41     win[0].set member('A',60,70);
42     win[1].set member('B',40,60);
43     win[2].set member('C',80,50);
44
45     largest(win,3);                           // 呼叫 largest()函數
46
47     system("pause");
48     return 0;
49  }
```

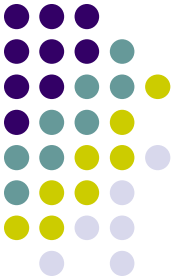
```
/* prog13_10 OUTPUT---
largest window= A
----------------------*/
```

傳遞物件陣列到函數裡 (3/3)

- 請注意，友誼函數largest() 原型的宣告語法撰寫如下

- 傳遞陣列時，largest() 函數的括號內填上陣列的名稱





資料成員與成員函數的複習 (1/3)

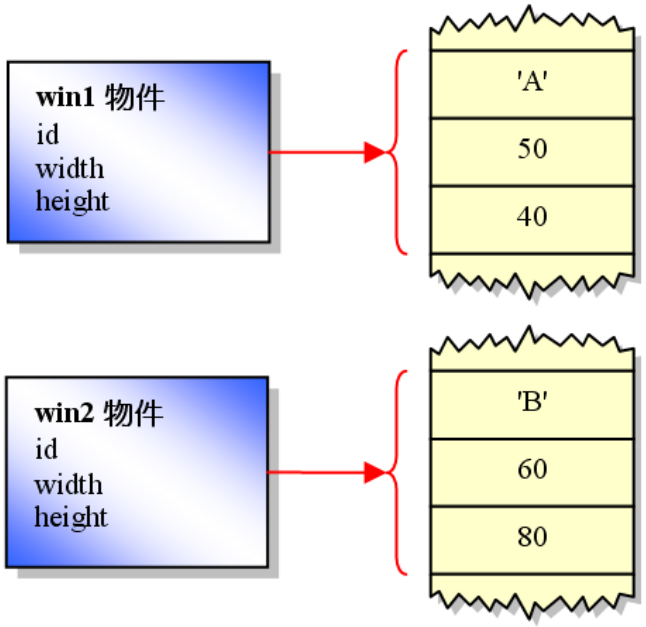
- prog13_11介紹實例變數與實例函數

```
01  // prog13_11, 簡單的範例, 實例變數與實例方法
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  class CWin                          // 定義視窗類別 CWin
06  {
07    private:
08      char id;
09      int width, height;
10
11    public:
12      CWin(char i,int w,int h):id(i),width(w),height(h)
13      {}
14
15      void show member(void)
16      {
17         cout << "Window " << id << ": ";
18         cout << "width=" << width << ", height=" << height << endl;
19      }
20  };
```

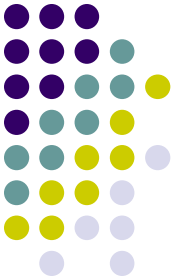
```
/* prog13_11 OUTPUT----------

Window A: width=50, height=40
Window B: width=60, height=80
-----------------------------*/
```

資料成員與成員函數的複習 (2/3)

```
21
22  int main(void)
23  {
24     CWin win1('A',50,40);
25     CWin win2('B',60,80);
26
27     win1.show member();
28     win2.show member();
29
30     system("pause");
31     return 0;
32  }
```




```
/* prog13_11 OUTPUT-----------

Window A: width=50, height=40
Window B: width=60, height=80
-----------------------------*/
```



資料成員與成員函數的複習 (3/3)

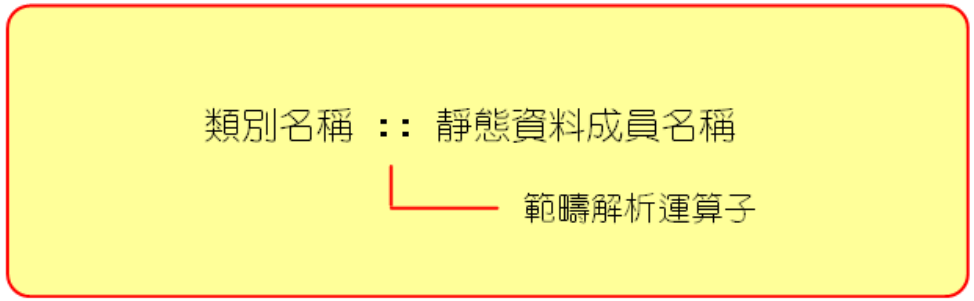
- 不同的物件之變數各自獨立，且存於不同的記憶體之內，具有此特性的變數稱為「實例變數」（instance variable）
- CWin類別裡的show_member() 必須透過物件來呼叫

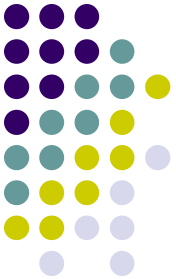
```
win1.show member();  // 透過物件 win1 呼叫 show member()
win2.show member();  // 透過物件 win2 呼叫 show member()
```

  - 必須利用物件來呼叫的函數，稱為「實例函數」（instance function）

靜態資料成員

- 靜態資料成員由所有的物件所共享
- 靜態資料成員可利用static來宣告
- 想在類別外存取靜態資料成員，可用下面的語法表示



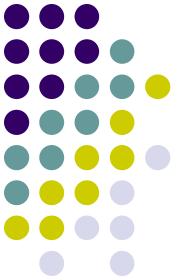


靜態資料成員的使用 (1/3)

- 下面的程式碼是靜態資料成員使用的範例

```
01  // prog13_12, public「靜態資料成員」的使用
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  class CWin                                  // 定義視窗類別 CWin
06  {
07     private:
08       char id;
09       int width, height;
10
11     public:
12       static int num;                        // 將靜態資料成員num宣告為public
13       CWin(char i,int w,int h):id(i),width(w),height(h)
14       {
15          num++;                              // 將靜態資料成員的值加 1
16       }
17       CWin()
18       {
19          num++;                              // 將靜態資料成員的值加 1
20       }
21  };
```

```
/* prog13_12 OUTPUT---
已建立 2 個物件了...
已建立 6 個物件了...
----------------------*/
```

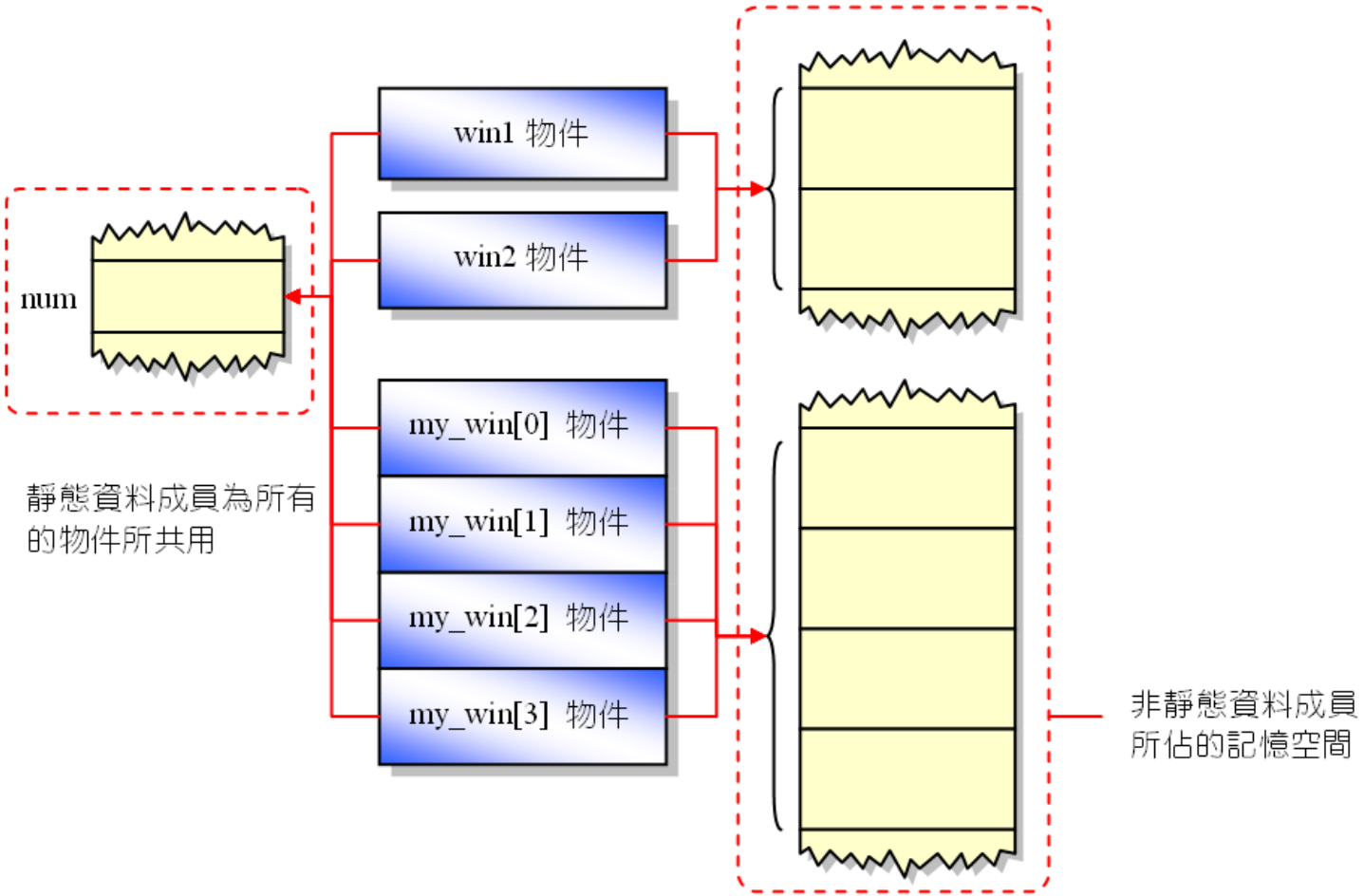


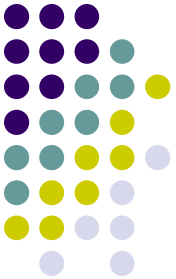
靜態資料成員的使用 (2/3)

```
22
23  int CWin::num=0;                          // 設定靜態資料成員 num 的初值
24
25  int main(void)
26  {
27     CWin win1('A',50,40);
28     CWin win2('B',60,80);
29     cout << "已建立 " << CWin::num << " 個物件了..." << endl;
30
31     CWin my win[4];
32     cout << "已建立 " << CWin::num << " 個物件了..." << endl;
33
34     system("pause");
35     return 0;
36  }
```

```
/* prog13_12 OUTPUT---

已建立 2 個物件了...
已建立 6 個物件了...
----------------------*/
```

靜態資料成員的使用 (3/3)



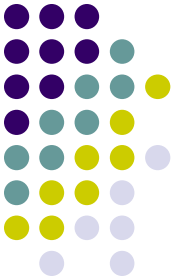


靜態資料成員宣告成private (1/2)

- 下面是將靜態資料成員宣告成private的範例：

```
01  // prog13_13, private「靜態資料成員」的使用
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  class CWin                          // 定義視窗類別 CWin
06  {
07    private:
08      char id;
09      int width, height;
10      static int num;                 // 將靜態資料成員宣告為 private
11
12    public:
13      CWin(char i,int w,int h):id(i),width(w),height(h)
14      {
15        num++;                        // 將靜態資料成員的值加 1
16      }
17      CWin()
18      {
19        num++;                        // 將靜態資料成員的值加 1
20      }
```

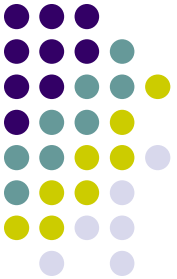
```
/* prog13_13 OUTPUT---
已建立 2 個物件了...
已建立 6 個物件了...
----------------------*/
```



靜態資料成員宣告成private (2/2)

```
21        void count(void)           // 成員函數，可讀取 private 靜態資料成員
22        {
23           cout << "已建立 " << num << " 個物件了..." << endl;
24        }
25    };
26
27    int CWin::num=0;               // 設定靜態資料成員 num 的初值
28
29    int main(void)
30    {
31       CWin win1('A',50,40);
32       CWin win2('B',60,80);
33       win1.count();               // 以 win1 物件呼叫 count
34
35       CWin my win[4];
36       win2.count();               // 也可用 win2 物件呼叫 count
37
38       system("pause");
39       return 0;
40    }
```

```
/* prog13_13 OUTPUT---

已建立 2 個物件了...
已建立 6 個物件了...
----------------------*/
```



靜態成員函數

- 於prog13_13中，所有的count() 均是透過物件來呼叫

```
win1.count();    // 用 win1 物件呼叫 count() 函數
win2.count();    // 用 win2 物件呼叫 count() 函數
```

- count() 也可以宣告成「靜態成員函數」

```
static void count(void)
{
   cout << "已建立了 " << num << " 個物件" << endl;
}
```

如此便可直接用類別來呼叫它

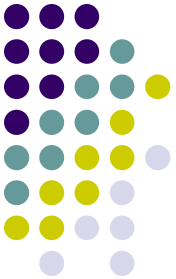
```
CWin::count();    // 直接用 CWin 類別呼叫「靜態成員函數」
```

靜態成員函數的使用 (1/2)

- 下面的範例是把count() 函數改寫成「靜態成員函數」

```
01  // prog13_14, 「靜態成員函數」的使用
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  class CWin                          // 定義視窗類別 CWin
06  {
07     private:
08       char id;
09       int width, height;
10       static int num;                // 靜態資料成員
11
12     public:
13       CWin(char i,int w,int h):id(i),width(w),height(h)
14       {
15          num++;
16       }
17       CWin()
18       {
19          num++;
20       }
```

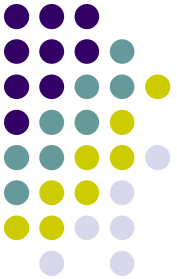
```
/* prog13_14 OUTPUT---

已建立 0 個物件了...
已建立 2 個物件了...
已建立 7 個物件了...
---------------------*/
```



靜態成員函數的使用 (2/2)

```
21        static void count(void)   // 靜態成員函數
22        {
23           cout << "已建立 " << num << " 個物件了..." << endl;
24        }
25    };
26
27    int CWin::num=0;                  // 設定靜態資料成員的初值
28
29    int main(void)
30    {
31       CWin::count();                 // 用類別呼叫靜態成員函數
32
33       CWin win1('A',50,40);
34       CWin win2('B',60,80);
35       CWin::count();                 // 用類別呼叫靜態成員函數
36
37       CWin my win[5];
38       CWin::count();                 // 用類別呼叫靜態成員函數
39
40       system("pause");
41       return 0;
42    }
```

```
/* prog13_14 OUTPUT---

已建立 0 個物件了...
已建立 2 個物件了...
已建立 7 個物件了...
----------------------*/
```



靜態成員函數的使用限制 (1/2)

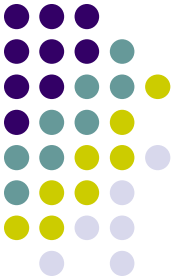
- 「靜態成員函數」不能取用類別內一般的變數或函數

  如果把prog13_14的count() 函數修改成如下的程式碼：

```
static void count(void)
{
   cout << "id= " << id << endl;   // 錯誤，無法對非靜態變數做存取
   cout << "已建立 " << num << " 個物件了..." << endl;
}
```

  則編譯時會產生錯誤

- 「非靜態變數」無法在靜態函數的內部來呼叫
- 「非靜態函數」也不能直接在靜態函數的內部呼叫



靜態成員函數的使用限制 (2/2)

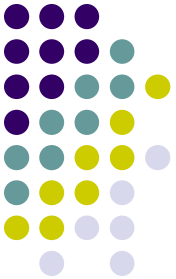
- 「靜態成員函數」內部不能使用this關鍵字

例如下面的程式碼是錯誤的：

```
static void count(void)
{
   cout << "id= " << this->id << endl;  // 錯誤，不能使用 this 關鍵字
   cout << "已建立 " << num << " 個物件了..." << endl;
}
```

如果編譯上面的程式碼，將會得到下列的錯誤訊息

```
'this' is unavailable for static member functions
```



指標與物件

- 把一個指標指向CWin物件

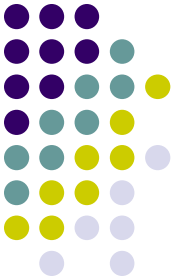
```
CWin *ptr = NULL;     // 宣告指向 CWin 物件的指標
ptr = &win1 ;         // 將 win1 的位址設給 ptr
```

- 也可以在宣告指標的同時，順便將它指向物件

```
CWin *ptr = &win1 ;
```

- 下面的語法來取用win1物件的成員函數與資料成員

```
ptr->area();     // 相當於用 win1 物件呼叫 area()
ptr->id;         // 相當於用 win1 物件取用 id 成員
```

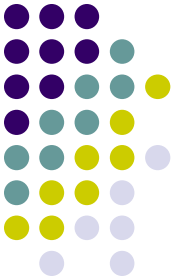


傳遞物件到函數 (1/2)

- 下面的範例練習將物件傳遞到函數裡

```
01  // prog13_15, 傳遞物件到函數裡
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  class CWin                                  // 定義視窗類別 CWin
06  {
07    private:
08      char id;
09      int width, height;
10
11    public:
12      CWin(char i,int w,int h):id(i),width(w),height(h)   // 建構元
13      {}
14
15      void compare(CWin *win)                 // 以指向物件的指標為引數
16      {
17        if(this->area() > win->area())
18          cout << "Window " << this->id << " is larger" << endl;
19        else
20          cout << "Window " << win->id << " is larger" << endl;
21      }
```

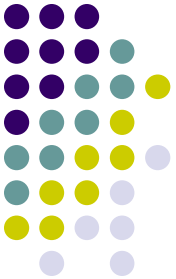
```
/* prog13_15 OUTPUT---
Window A is larger
----------------------*/
```



傳遞物件到函數 (2/2)

```
22        int area(void)                          // 成員函數 area()
23        {
24           return width*height;                 // 傳回物件的面積值
25        }
26    };
27
28    int main(void)
29    {
30       CWin win1('A',70,80);
31       CWin win2('B',60,90);
32       CWin *ptr1=&win1;              // 宣告 ptr1 指標，並將它指向物件 win1
33       CWin *ptr2=&win2;              // 宣告 ptr2 指標，並將它指向物件 win2
34
35       ptr1->compare(ptr2);           // 用 ptr1 呼叫 compare()，並傳遞 ptr2
36
37       system("pause");
38       return 0;
39    }
```

```
/* prog13_15 OUTPUT---
Window A is larger
----------------------*/
```

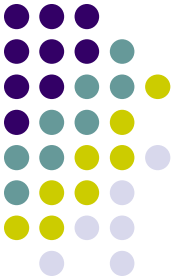


參照與物件

- 把「參照」指向某個物件

```
CWin &ref = win1; // 宣告 ref 為一參照到 win1 物件的參照變數
cout<<"area="<<ref.area()<<endl; // 相當於用 win1 呼叫 area()
cout<<"id="<<ref.id<<endl;       // 相當於用 win1 取用 id 成員
```

- 「參照」和物件一樣，利用「.」來存取物件內的成員

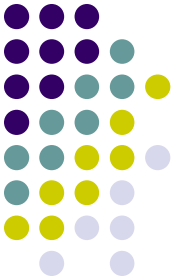


傳遞物件的參照到函數 (1/2)

- 下面的範例是在傳遞物件時，改採傳遞物件的參照

```
01  // prog13_16, 傳遞物件的參照到函數裡
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  class CWin                              // 定義視窗類別 CWin
06  {
07     private:
08       char id;
09       int width, height;
10
11     public:
12       CWin(char i,int w,int h):id(i),width(w),height(h)   // 建構元
13       {}
14
15       void compare(CWin &win)          // compare()可接收物件的參照
16       {
17          if(this->area() > win.area())
18            cout << "Window " << this->id << " is larger" << endl;
19          else
20            cout << "Window " << win.id << " is larger" << endl;
21       }
```

```
/* prog13_16 OUTPUT---
Window A is larger
----------------------*/
```



傳遞物件的參照到函數 (2/2)

```
22        int area(void)                    // 成員函數 area()
23        {
24           return width*height;           // 傳回物件的面積值
25        }
26    };
27
28    int main(void)
29    {
30       CWin win1('A',70,80);
31       CWin win2('B',60,90);
32
33       win1.compare(win2);              // 用 win1 呼叫 compare()，並傳遞 win2
34
35       system("pause");
36       return 0;
37    }
```

```
/* prog13_16 OUTPUT---
Window A is larger
---------------------*/
```


-The End-