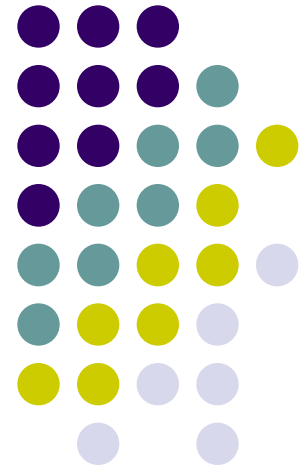


# 運算子的多載

認識運算子的多載

學習「+」的多載

熟悉「=」的多載





# 運算子的多載 (1/2)

- 運算子的多載，是指同一運算子在不同的時機，可進行不同的工作
- 「>」運算子的多載

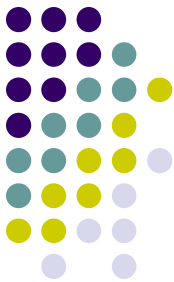
```
5 > 7;           // 判別整數 5 是否大於 7  
win1 > win2;      // 判別 win1 與 win2 物件面積的大小
```

- 「>」運算子經重新定義後，可用來判別數字的大小，也可用來判別物件面積大小的做法，即稱為「>」運算子的多載（overloading）



# 運算子的多載 (2/2)

- 不能多載的運算子
  - `::` 範籌解析運算子 (scope resolution operator)
  - `?:` 條件運算子 (conditional operator)
  - `.` 成員存取運算子 (direct member selection operator)
  - `sizeof` 運算子 (sizeof operator)



# 「>」運算子的多載 (1/4)

- 為運算子定義多載函數

```
01  int operator>(CWin &win)           // 定義運算子「>」的多載
02  {
03      return( this->area() > win.area());
04  }
```

- 呼叫operator>() 函數

```
win1.operator>(win2); // 呼叫 operator>() 函數
win1 > win2;          // 此敘述會呼叫 operator>() 函數
```



# 「>」運算子的多載 (2/4)

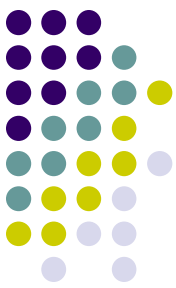
- 下圖說明在operator>() 函數裡，引數傳遞的情形

```
int main(void)
{
    ...
    if( win1 > win2 )
    {
        ...
    }
    ...
}
```

win2 是以參照的型式  
傳入 operator>()函數

```
class
{
    ...
    int operator>(CWin &win)
    {
        return( this->area() > win.area() );
    }
}
```

this 是指向 win1  
物件的指標



# 「>」運算子的多載 (3/4)

- 下面是將運算子「>」多載的範例

```

01 // prog15_1, 運算子「>」多載的範例
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         int operator>(CWin &win)                // 定義運算子「>」的多載
16         {
17             return(this->area() > win.area());
18         }
19         int area(void)
20         {
21             return width*height;
22         }
23 };

```

**/\* prog15\_1 OUTPUT-----**  
win1 is larger than win2  
**-----\*/**



# 「>」運算子的多載 (4/4)

```
24
25 int main(void)
26 {
27     CWin win1('A',70,80);
28     CWin win2('B',60,90);
29
30     if(win1>win2)                // 判別 win1 與 win2 物件面積的大小
31         cout << "win1 is larger than win2" << endl;
32     else
33         cout << "win2 is larger than win1" << endl;
34
35     system("pause");
36     return 0;
37 }
```

```
/* prog15_1 OUTPUT-----
win1 is larger than win2
-----*/
```

第 30 行的判斷敘述也可以把它寫成

```
if(win1.operator>(win2))    // 呼叫 operator>() 函數
```



# 再把operator>()函數多載 (1/5)

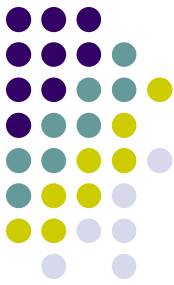
- 在prog15\_1裡，不能在main() 裡撰寫下面兩種敘述
  - (1) win1 > 7000; // 在prog15\_1裡，不能比較面積與常數的大小
  - (2) 7000 > win1; // 同上，但常數是置於「>」符號的前面

解決的方式是，再定義一個operator() 函數

接收整數常數的參照

```
int operator>(const int &var)  
{  
    return(this->area() > var);  
}
```





## 再把operator>()函數多載 (2/5)

- 有兩個引數的operator>() 函數，引數接收的情形如下

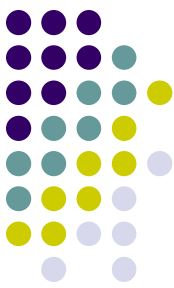
```
int main(void)
{
    ...
    if( 7000 > win2 ) {
        ...
    }
    ...
}
```

整數 7000 會傳入  
operator>() 的第一個  
引數

CWin 物件的參照傳入  
operator>()的第二個引數

```
int operator>(const double &val, CWin &win)
{
    return( val > win.area());
}
```

一般的函數，非類別裡的成員函數



# 再把operator>()函數多載 (3/5)

- prog15\_2是使用operator>() 函數的完整範例

```
01 // prog15_2, 運算子多載的範例(二)
```

```
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):id(i),width(w),height(h) // 建構元
```

```
13     {}
```

```
14     int operator>(CWin &win)
```

```
15     {
```

```
16         return(this->area() > win.area());
```

```
17     }
```

```
18     int operator>(const int &var)
```

```
19     {
```

```
20         return(this->area() > var);
```

```
21     }
```

```
/* prog15_2 OUTPUT-----
```

```
win1 is larger than win2
```

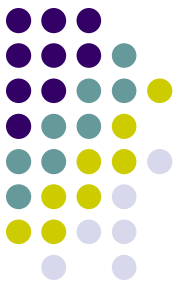
```
win1 is smaller than 7000
```

```
win2 is smaller than 4500
```

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

} 第一個 operator>()函數

} 第二個 operator>()函數



# 再把operator>()函數多載 (4/5)

```
22     int area(void)
23     {
24         return width*height;
25     }
26 };
27
```

```
/* prog15_2 OUTPUT-----
win1 is larger than win2
win1 is smaller than 7000
win2 is smaller than 4500
-----*/
```

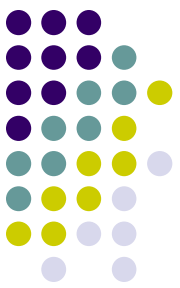
```
28 int operator>(const int &var, CWin &win)
29 {
30     return( var > win.area());
31 }
```

第三個 operator>()函數，注意  
這個函數定義在 CWin 類別  
之外，因此它是一般的函數

```
32
33 int main(void)
34 {
35     CWin win1('A',70,80);
36     CWin win2('B',60,70);
37
```

```
38     if(win1>win2)                // 呼叫第一個 operator>()函數
39         cout << "win1 is larger than win2" << endl;
40     else
41         cout << "win1 is smaller than win2" << endl;
```

```
42
```



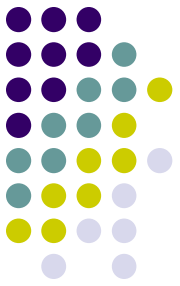
# 再把operator>()函數多載 (5/5)

```
43     if(win1>7000)                // 呼叫第二個 operator>() 函數
44         cout << "win1 is larger than 7000" << endl;
45     else
46         cout << "win1 is smaller than 7000" << endl;
47
48     if(4500>win2)                // 呼叫第三個 operator>() 函數
49         cout << "win2 is smaller than 4500" << endl;
50     else
51         cout << "win2 is larger than 4500" << endl;
52
53     system("pause");
54     return 0;
55 }
```

**/\* prog15\_2 OUTPUT-----**

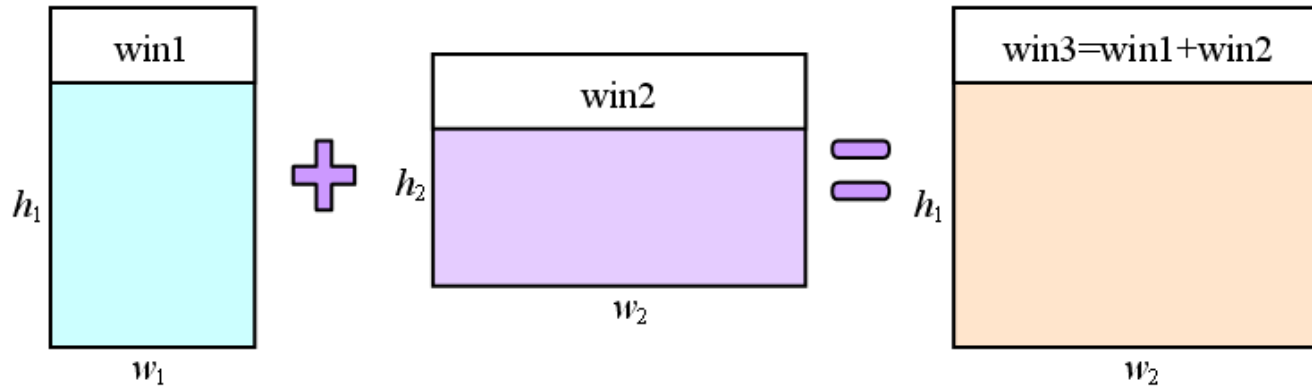
```
win1 is larger than win2
win1 is smaller than 7000
win2 is smaller than 4500
```

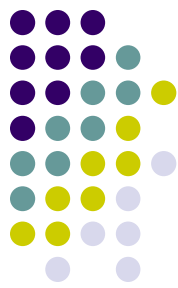
**-----\*/**



# 「+」的多載

- 定義CWin物件的相加
  - （取其width與height成員較大者）





# 「+」運算子多載的實例 (1/2)

01 // prog15\_3, 「+」運算子多載的範例

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=10,int h=10):id(i),width(w),height(h)

13 {}

14

15 CWin operator+(CWin &win) // 定義「+」運算子的多載

16 {

17 int w,h;

18 w = this->width > win.width ? this->width : win.width;

19 h = this->height > win.height ? this->height : win.height;

20 return CWin('D',w,h); // 呼叫建構元建立並傳回新的物件

21 }

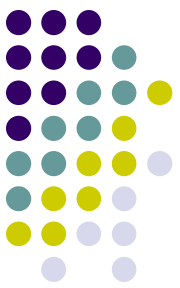
- 下面的程式是「+」運算子多載的實例

**/\* prog15\_3 OUTPUT-----**  
 Window D: width=70, height=90  
**-----\*/**



## 「+」運算子多載的實例 (2/2)

```
22     void show member(void)
23     {
24         cout << "Window " << id << ": ";
25         cout << "width=" << width << ", height=" << height << endl;
26     }
27 };
28
29 int main(void)
30 {
31     CWin win1('A',70,80);
32     CWin win2('B',60,90);
33     CWin win3;
34
35     win3=win1+win2;           // 物件的加法運算
36     win3.show member();
37
38     system("pause");         /* prog15_3 OUTPUT-----
39     return 0;                Window D: width=70, height=90
40 }                             -----*/
```

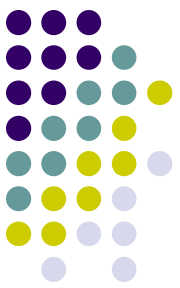


# 簡單的範例 (1/6)

- 如果物件裡的成員包含有指標，或是以動態記憶體方式配置變數時，使用預設的設定運算子「=」可能會發生錯誤

```
01 // prog15_4, 使用預設的設定運算子所發生的錯誤
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, *title;
09
10     public:
11         CWin(char i='D', char *text="Default window"):id(i)
12         {
13             title=new char[50]; // 配置可容納 50 個字元的記憶體空間
14             strcpy(title,text); // 將text所指向的字串拷貝給 title
15         }
```





## 簡單的範例 (2/6)

```

16     void set data(char i, char *text)
17     {
18         id=i;
19         strcpy(title,text);          // 將text所指向的字串拷貝給 title
20     }

21     void show(void)
22     {
23         cout << "Window " << id << ": " << title << endl;
24     }

25                                     /* prog15_4 OUTPUT-----
26     ~CWin(){ delete [] title; }      Window A: Main window
27                                     Window D: Default window
28     CWin(const CWin &win)           設定 win1=win2 之後...
29     {                                Window D: Default window
30         id=win.id;                  Window D: Default window
31         strcpy(title,win.title);
32     }
33 };
34

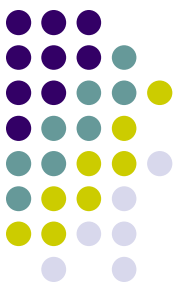
```

更改 win1 的資料成員之後...

Window B: Hello window

Window D: Hello window

-----\*/



## 簡單的範例 (3/6)

```
35  int main(void)
36  {
37      CWin win1('A',"Main window");
38      CWin win2;
39
40      win1.show();
41      win2.show();
42
43      win1=win2;                // 設定 win1=win2
44      cout << endl << "設定 win1=win2 之後..." << endl;
45      win1.show();
46      win2.show();
47
48      win1.set data('B',"Hello window");
49      cout << endl << "更改 win1 的資料成員之後..." << endl;
50      win1.show();
51      win2.show();
52
53      system("pause");
54      return 0;
55  }
```

**/\* prog15\_4 OUTPUT-----**

Window A: Main window  
Window D: Default window

設定 win1=win2 之後...  
Window D: Default window  
Window D: Default window

更改 win1 的資料成員之後...  
Window B: Hello window  
Window D: Hello window

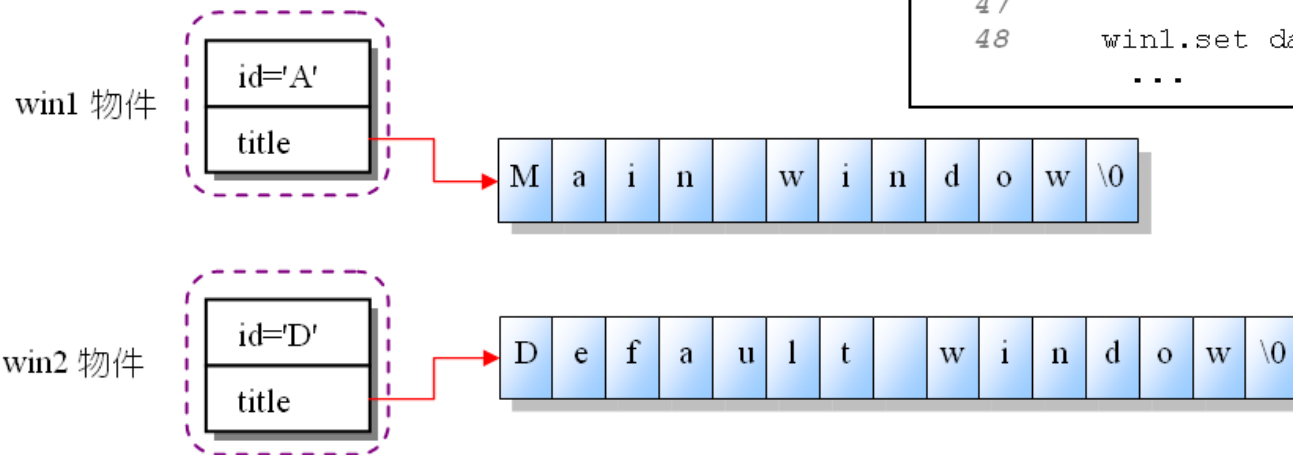
**-----\*/**



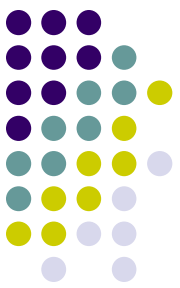
## 簡單的範例 (4/6)

- 下圖是執行完37~38行後的結果

執行完 37~38 行之後的結果



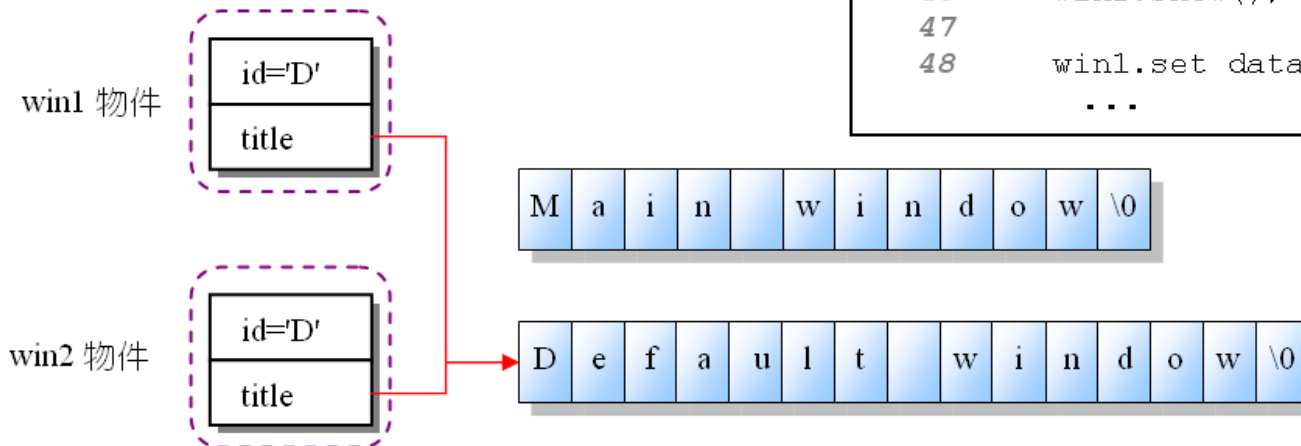
```
35  int main(void)
36  {
37      CWin win1('A', "Main window");
38      CWin win2;
39
40      win1.show();
41      win2.show();
42
43      win1=win2;    // 設定 win1=win2
44      cout << endl << "設定 win1=win2 之後...";
45      win1.show();
46      win2.show();
47
48      win1.set data('B', "Hello window");
    ...
```



# 簡單的範例 (5/6)

- 下圖是執行完43行後的結果

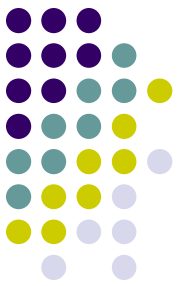
執行完 43 行之後的結果



```

35  int main(void)
36  {
37      CWin win1('A', "Main window");
38      CWin win2;
39
40      win1.show();
41      win2.show();
42
43      win1=win2;    // 設定 win1=win2
44      cout << endl << "設定 win1=win2 之後...";
45      win1.show();
46      win2.show();
47
48      win1.set data('B', "Hello window");
      ...

```

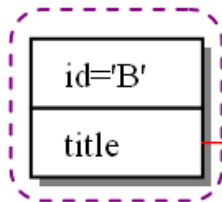


# 簡單的範例 (6/6)

- 下圖是執行完48行後的結果

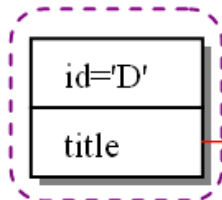
執行完 48 行之後的結果

win1 物件



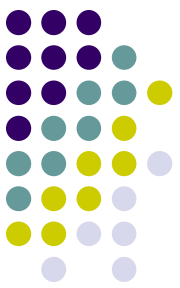
M a i n w i n d o w \0

win2 物件



H e l l o w i n d o w \0

```
35 int main(void)
36 {
37     CWin win1('A',"Main window");
38     CWin win2;
39
40     win1.show();
41     win2.show();
42
43     win1=win2; // 設定 win1=win2
44     cout << endl << "設定 win1=win2 之後...";
45     win1.show();
46     win2.show();
47
48     win1.set data('B',"Hello window");
    ...
```



# 修正錯誤 (1/6)

- 下面的範例，是使用設定運算子來修正錯誤

```
01 // prog15_5, 使用設定運算子來修正錯誤
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, *title;
09
10     public:
11         CWin(char i='D', char *text="Default window"):id(i)
12         {
13             title=new char[50];
14             strcpy(title,text);
15         }
16         void set data(char i, char *text)
17         {
18             id=i;
19             strcpy(title,text);
20 }
```

```
/* prog15_5 OUTPUT-----
Window A: Main window
Window D: Default window

設定 win1=win2 之後...
Window D: Default window
Window D: Default window

更改 win1 的資料成員之後...
Window B: Hello window
Window D: Default window
-----*/
```



## 修正錯誤 (2/6)

```
21     void operator=(const CWin &win)    // 定義設定運算子「=」的多載
22     {
23         id=win.id;
24         strcpy(this->title,win.title); // 字串拷貝
25     }
26     void show(void)
27     {
28         cout << "Window " << id << ": " << title << endl;
29     }
30
31     ~CWin(){ delete [] title; }        // 解構元
32
33     CWin(const CWin &win)              // copy constructor
34     {
35         id=win.id;
36         strcpy(title,win.title);       // 拷貝建構元
37     }
38 };
39
40 int main(void)
41 {
```



## 修正錯誤 (3/6)

```
42     CWin win1('A',"Main window");
43     CWin win2;
44
45     win1.show();
46     win2.show();
47
48     win1=win2;
49     cout << endl << "設定 win1=win2 之後..." << endl;
50     win1.show();
51     win2.show();
52
53     win1.set data('B',"Hello window");
54     cout << endl << "更改 win1 的資料成員之後..." << endl;
55     win1.show();
56     win2.show();
57
58     system("pause");
59     return 0;
60 }
```

**/\* prog15\_5 OUTPUT-----**

Window A: Main window  
Window D: Default window

設定 win1=win2 之後...  
Window D: Default window  
Window D: Default window

更改 win1 的資料成員之後...  
Window B: Hello window  
Window D: Default window

**-----\*/**



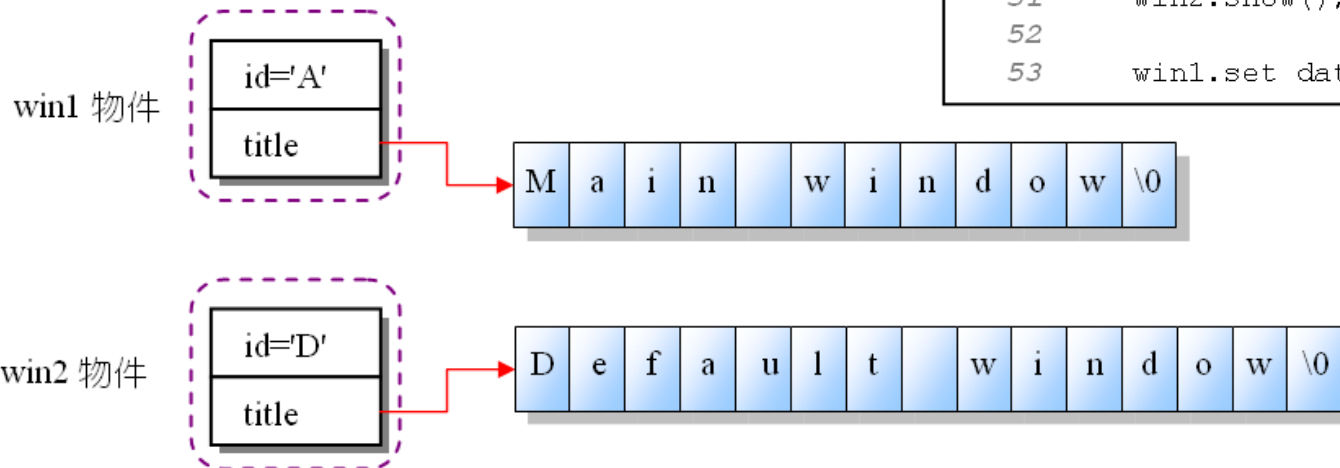


# 修正錯誤 (4/6)

- 下圖為執行完42~43行後的結果

```
40 int main(void)
41 {
42     CWin win1('A', "Main window");
43     CWin win2;
44
45     win1.show();
46     win2.show();
47
48     win1=win2;
49     cout << endl << "設定 win1=win2 之後...";
50     win1.show();
51     win2.show();
52
53     win1.set data('B', "Hello window");
```

執行完 42~43 行之後的結果



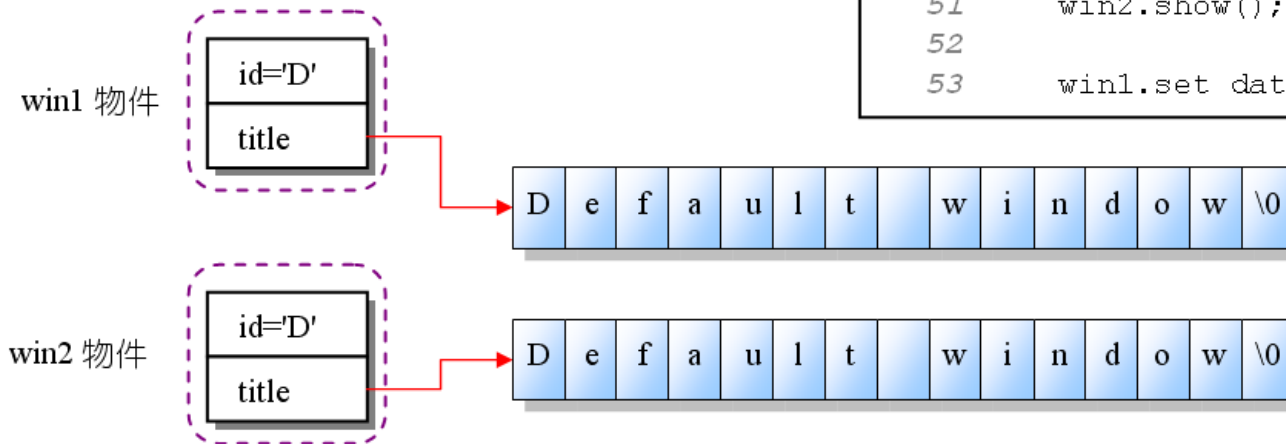


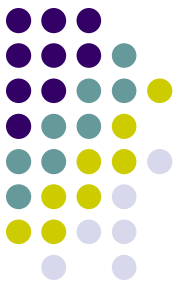
# 修正錯誤 (5/6)

- 下圖為執行完48行後的結果

```
40 int main(void)
41 {
42     CWin win1('A',"Main window");
43     CWin win2;
44
45     win1.show();
46     win2.show();
47
48     win1=win2;
49     cout << endl << "設定 win1=win2 之後...";
50     win1.show();
51     win2.show();
52
53     win1.set data('B',"Hello window");
```

執行完 48 行之後的結果

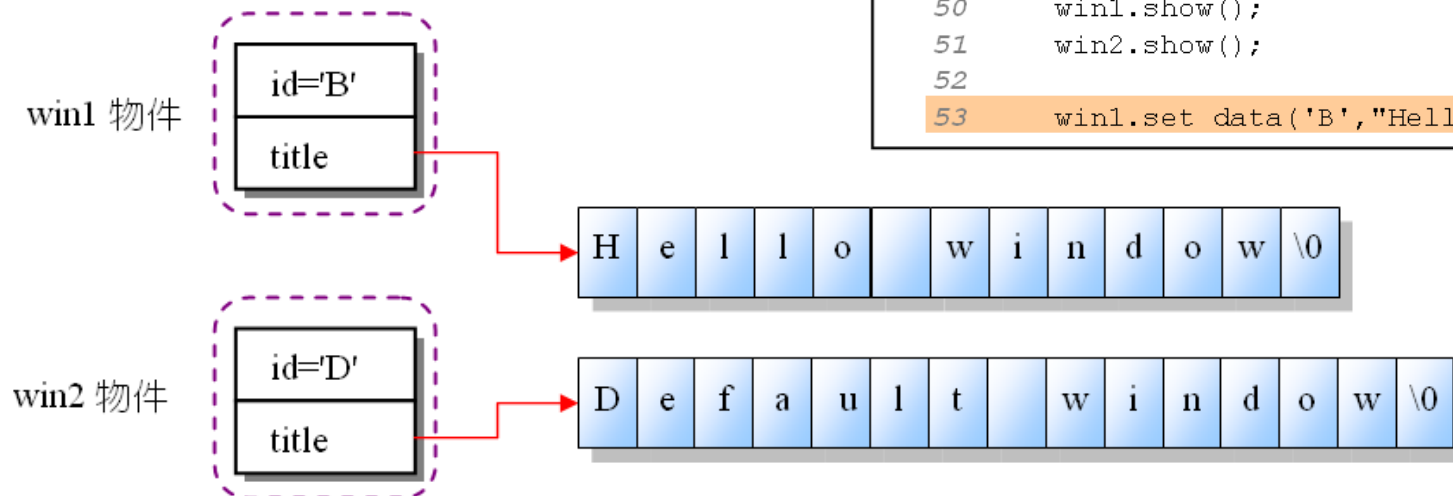




# 修正錯誤 (6/6)

- 下圖為執行完53行後的結果

執行完 53 行之後的結果



```
40 int main(void)
41 {
42     CWin win1('A', "Main window");
43     CWin win2;
44
45     win1.show();
46     win2.show();
47
48     win1=win2;
49     cout << endl << "設定 win1=win2 之後...";
50     win1.show();
51     win2.show();
52
53     win1.set data('B', "Hello window");
```



## 設定運算子多載的進階應用 (1/2)

- 如果能把設定運算子「=」多載成可以連續設值：

```
win1=win2=win3;           // 把win3的值設給win1與win2
```

可依下面的步驟進行設計：

- 先把win3的值設給win2，再把win2的值設給win1

```
win1=(win2=win3);
```

- 把上式改寫成：

```
win1=(win2.operator=(win3));
```

- 再改為成右邊的敘述：

```
win1.operator=(win2.operator=(win3));
```

以 win2 物件呼叫 operator=() 函數，  
並傳入 win3 物件

以 win2.operator=(win3) 的傳回值為引數，傳入  
由 win1 物件所呼叫的 operator=() 函數



# 設定運算子多載的進階應用 (2/2)

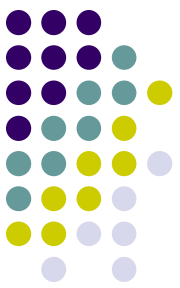
- 最後可以寫出「=」運算子的多載

傳回 CWin 物件的參照

傳入 CWin 物件的參照

```
CWin & operator=(const CWin &win)
{
    id=win.id;
    strcpy(this->title,win.title);
    return *this;
}
```

this 是指向呼叫此一函數的物件之指標，  
因此 \*this 即代表呼叫此函數的物件



## 「=」運算子多載的範例 (1/3)

- 下面的程式修改自prog15\_5，示範「=」運算子的多載

```
01 // prog15_6, 設定運算子多載的進階應用
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, *title;
09
10     public:
11         CWin(char i='D', char *text="Default window"):id(i)
12         {
13             title=new char[50];
14             strcpy(title,text);
15         }
16         void set data(char i, char *text)
17         {
18             id=i;
19             strcpy(title,text);
20         }
```



## 「=」運算子多載的範例 (2/3)

```

21     CWin &operator=(const CWin &win)    // 定義設定運算子「=」的多載
22     {
23         id=win.id;
24         strcpy(this->title,win.title);
25         return *this;
26     }
27     void show(void)
28     {
29         cout<<"Window "<< id <<" : "<< title <<endl;
30     }
31
32     ~CWin(){ delete [] title; }
33
34     CWin(const CWin &win)                // copy constructor
35     {                                     /* prog15_6 OUTPUT-----
36         id=win.id;                        Window A: Main window
37         strcpy(title,win.title);          Window B: Big window
38     }                                     Window D: Default window
39 };                                     設定 win1=win2=win3,並更改 win1 的成員之後 ...
40                                     Window A: Hello window
41     int main(void)                       Window D: Default window
42     {                                     Window D: Default window
                                         -----*/

```



## 「=」運算子多載的範例 (3/3)

```

41  int main(void)
42  {
43      CWin win1('A', "Main window");
44      CWin win2('B', "Big window");
45      CWin win3;
46
47      win1.show();
48      win2.show();
49      win3.show();
50
51      win1=win2=win3;           // 設定 win1=win2=win3
52      win1.set data('A', "Hello window"); // 修改 win1 的內容
53
54      cout << "設定 win1=win2=win3, 並更改 win1 的成員之後 ..." << endl;
55      win1.show();
56      win2.show();
57      win3.show();
58
59      system("pause");
60      return 0;
61  }

```

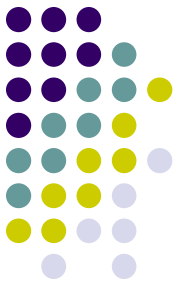
} 建立三個物件

```

/* prog15_6 OUTPUT-----
Window A: Main window
Window B: Big window
Window D: Default window
設定 win1=win2=win3, 並更改 win1 的成員之後 ...
Window A: Hello window
Window D: Default window
Window D: Default window
-----*/

```





-The End-