第十二、十三章
類別與物件

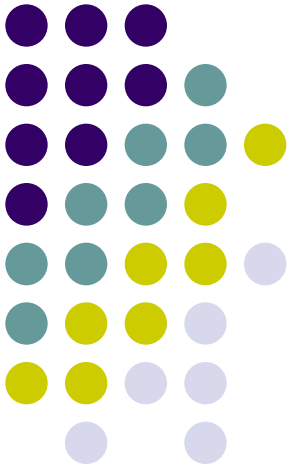
學習類別的觀念

認識與撰寫成員函數

認識引數的傳遞與多載的方式

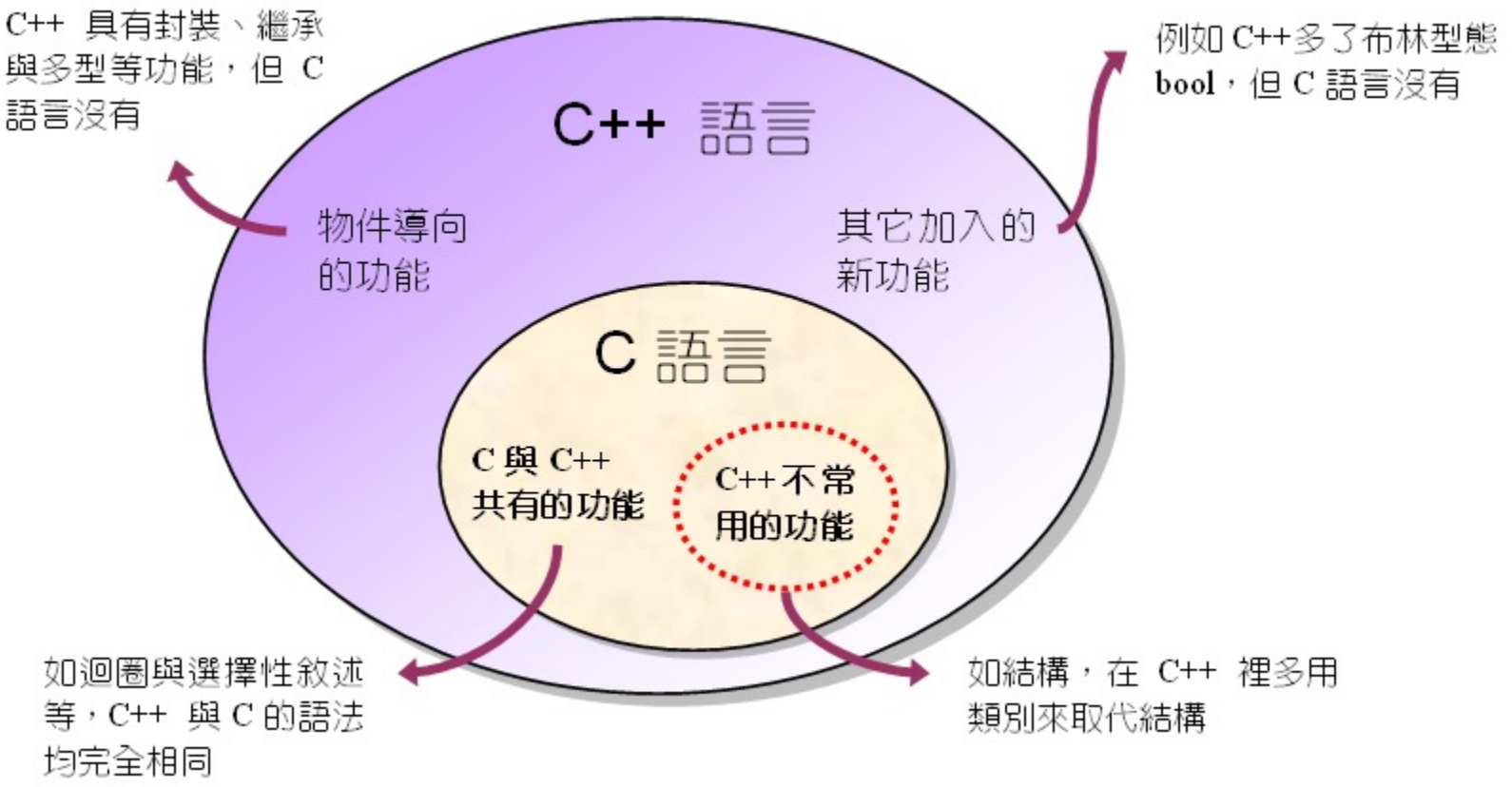
建立公有與私有成員

認識建構元及其引數的使用

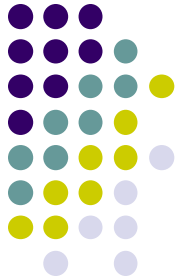


C 與 C++

- C++ 可看成是具備了物件導向的功能的 C 語言



差別1. 關於ANSI/ISO C++的標準



- 新版的ANSI/ISO C++ 於1997年頒佈
  - 捨棄標頭檔的副檔名.h。

    include <iostream> -> 沒有副檔名.h
  - 把原先從C語言移植到C++ 的函數庫，在其標頭檔名稱之前加上小寫的字母c。

    math.h -> cmath
  - 將所有的函數、類別與物件名稱放在特定的名稱空間std內，所以必須利用using namespace來設定名稱空間為std。

    using namespace std

差別2. 布林型態

- C++ 有 bool 型態，C 則沒有
- 布林（boolean）型態的值只有 true 和 false 兩種

```
bool status=true;    //  宣告布林變數status，並設值為true
bool flag=false;     //  宣告布林變數flag，並設值為false
bool test=1;         //  宣告布林變數test，並設定為true
```

差別3. 變數的位置

- C++ 可以在迴圈裡宣告變數

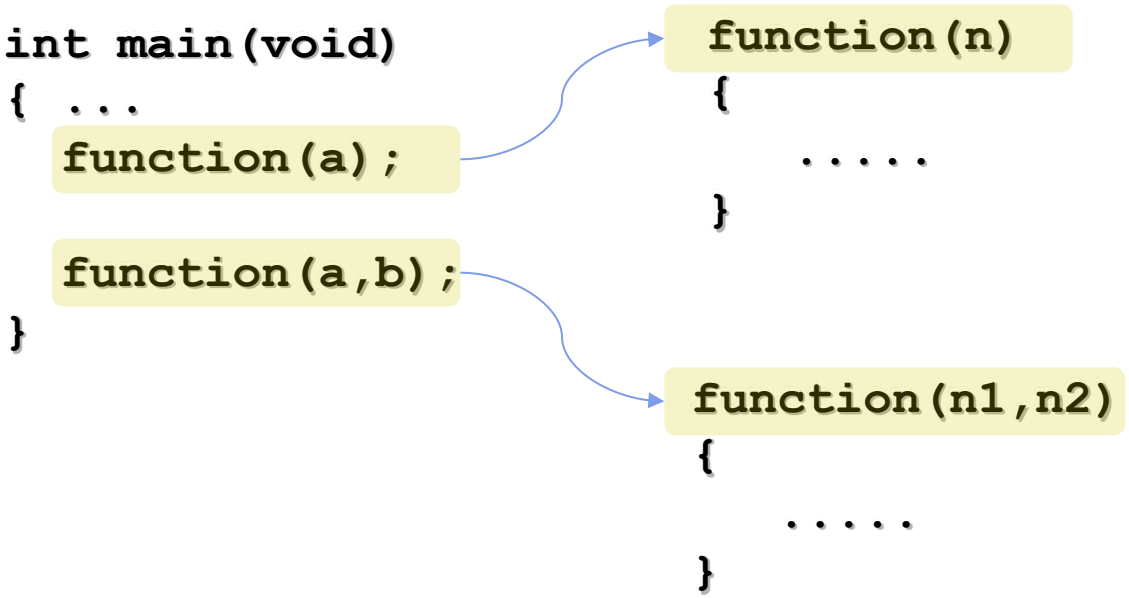
```
01  // prog16_4, 變數宣告的位置
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  int main(void)
06  {
07     int i=20;      // 宣告變數 i，並設值為 20
08
09     for(int i=0;i<3;i++)
10        cout<<"在 for 迴圈裡,i="<<i<<endl;
11
12     cout<<"for 迴圈執行完後,i="<<i<<endl;  // 執行完迴圈後，印出 i 的值
13
14     system("pause");
15     return 0;
16  }
```

變數 i 的有效範圍

```
/* prog16_4 OUTPUT--

在 for 迴圈裡,i=0
在 for 迴圈裡,i=1
在 for 迴圈裡,i=2
for 迴圈執行完後,i=20
----------------------*/
```

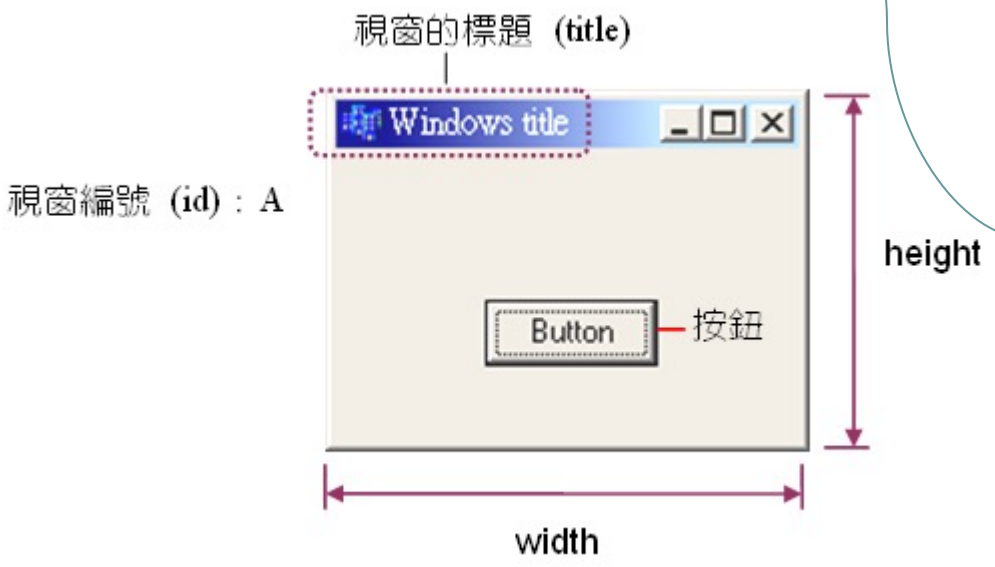
差別4. 函數的多載

- 多載（overloading）
  - 編譯器便會根據引數的個數與型態，自動執行相對應的函數



差別5. 結構與類別

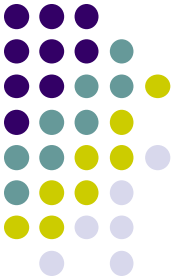
- 下圖是一個典型的視窗
  - 利用結構來定義視窗
  - 利用類別來定義視窗




```
struct Win
{
    char id;
    int width;
    int height;
};
```

```
class CWin
{
  public:
    char id;
    int width;
    int height;
    int area()
    {
      return
width*height;
    }
};
```

以結構表示視窗



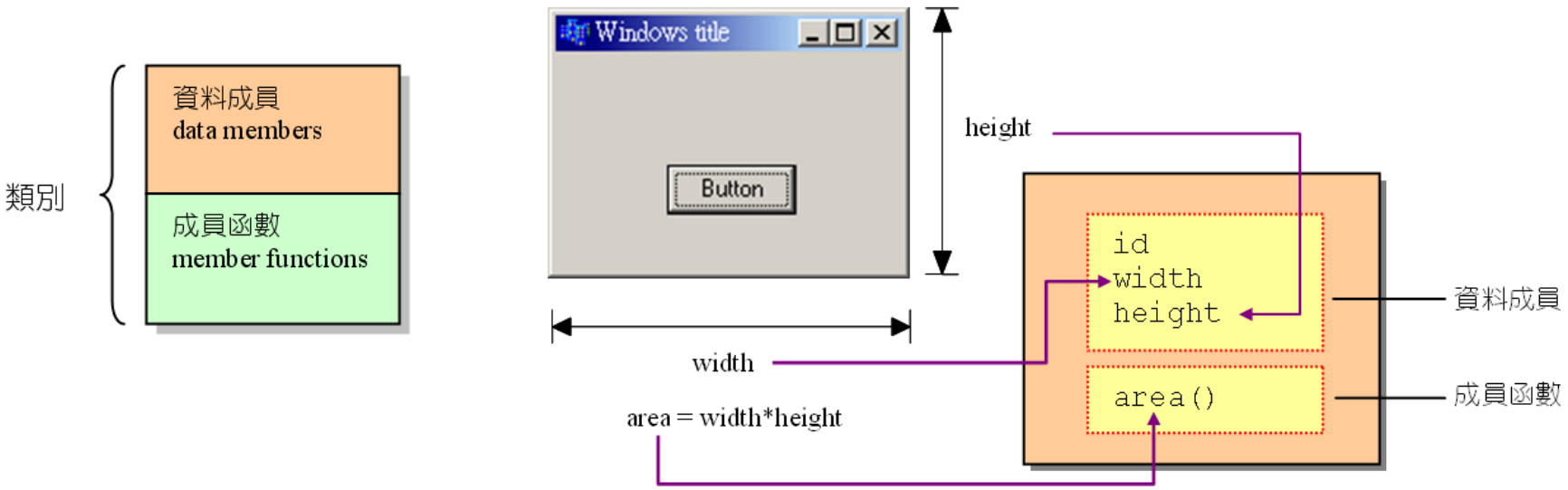
- 下面的範例是利用結構來表示視窗

```
01  // prog12_1, 利用結構來表示視窗
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  struct Win                          // 利用結構來定義視窗(Window)
06  {
07     char id;
08     int width;                       // Win 結構的 width 成員
09     int height;                      // Win 結構的 height 成員
10  };
11
12  int area(struct Win w)              // 面積函數
13  {
14     return w.width*w.height;         // 面積=寬*高
15  }
16
17  int main(int)
18  {
19     struct Win win1;                 // 宣告 Win 結構的物件 win1
20
21     win1.id='A';
22     win1.width=50;                   // 設定寬為 50
23     win1.height=40;                  // 設定高為 40
24
25     cout << "Window " << win1.id << ", area=" << area(win1) << endl;
26     system("pause");
27     return 0;
28  }
```

```
/* prog12_1 OUTPUT---

Window A, area=2000
----------------------*/
```

類別的基本概念 – 封裝

- 類別（class）包含「資料成員」與「成員函數」



類別的宣告

- 類別定義的語法如下：

```
class 類別名稱
{
  public:
    資料型態 變數名稱;
    ...

    傳回值型態  函數名稱(型態 1 引數 1, 型態 2 引數 2,...)
    {
       程式敘述 ;
       return 運算式;
    }
  ...
};
```

定義函數成員

宣告資料成員

函數的本體（body）

定義成員函數的內容

這兒必須要加分號

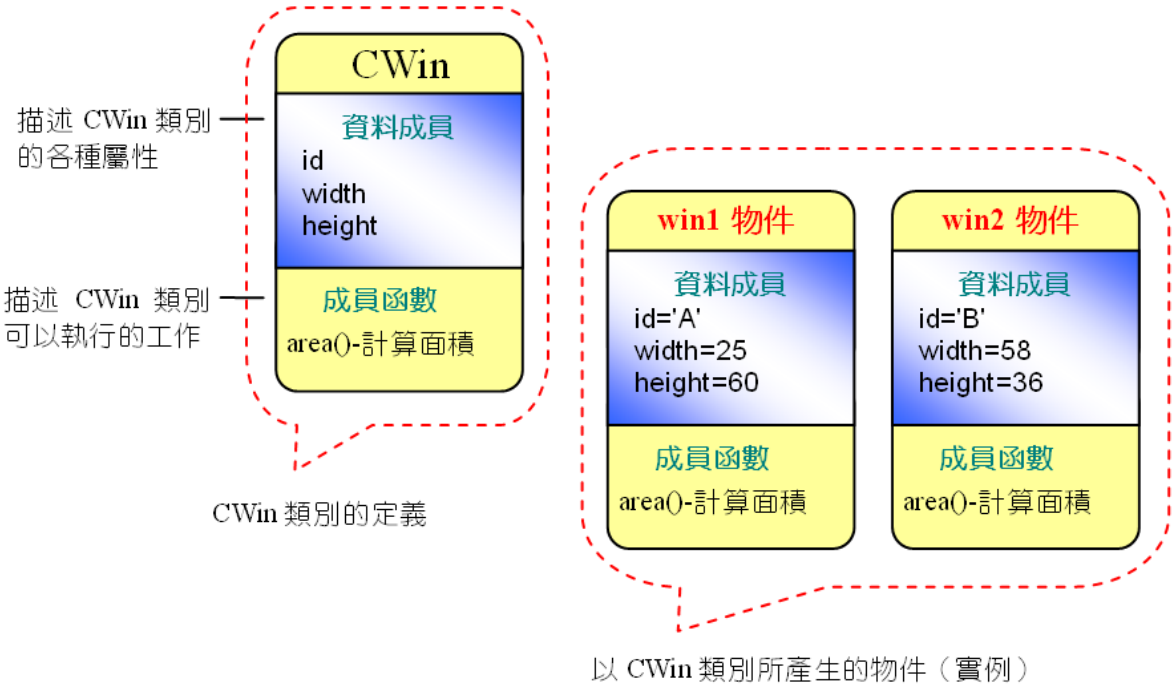
```
class CWin   // 定義視窗類別CWin
{
   public:    // 宣告公有屬性之成員
      char id;
      int width;
      int height;
      int area()
      {
         return width*height;
      }
};
```

宣告資料成員

建立新的物件 (1/2)

- 建立物件

```
CWin win1;          // 建立 CWin 型態的物件 win1

CWin win1,win2;     // 同時建立物件 win1 與 win2
```



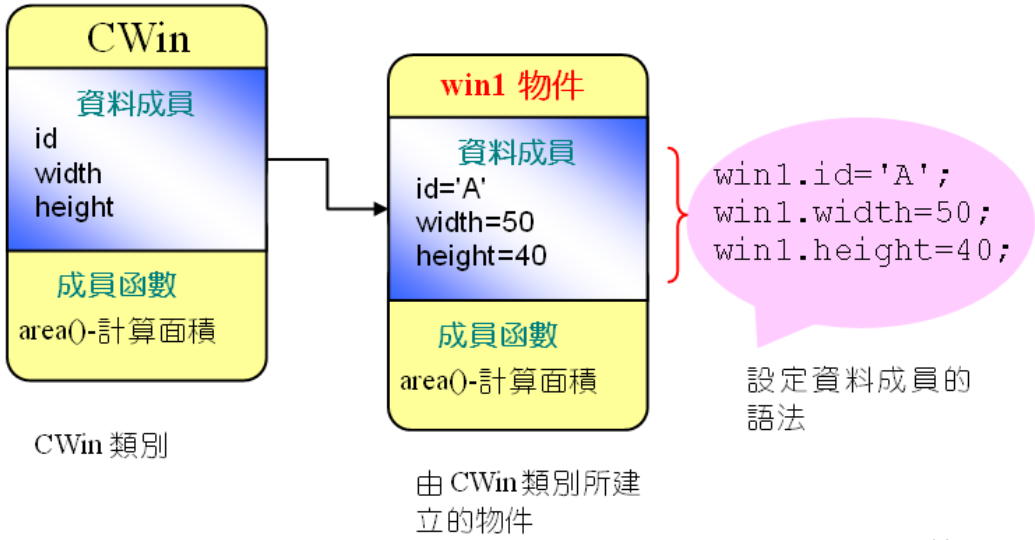
建立新的物件 (2/2)

- 存取物件的內容

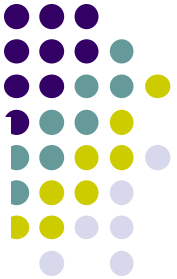
物件名稱.特定的資料成員

- 設定資料成員的範例

```
int main(void)
{
   CWin win1;

   win1.id='A';
   win1.width=50;
   win1.height=40;
    ....
}
```



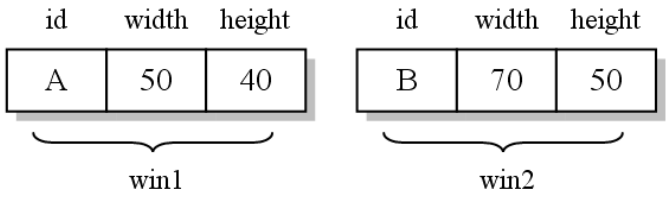
使用類別來設計完整的程式



```
01  // prog12_2, 第一個類別程式
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  class CWin                          // 定義視窗類別 CWin
06  {
07    public:                           // 設定資料成員為公有
08      char id;
09      int width;
10      int height;
11  };
12  int main(void)
13  {
14    CWin win1;                        // 宣告 CWin 類別型態的變數 win1
15
16    win1.id='A';
17    win1.width=50;                    } 設定資料成員
18    win1.height=40;
19
20    cout << "Window " << win1.id << ":" << endl;
21    cout << "win1.width = " << win1.width << endl;
22    cout << "win1.height = " << win1.height << endl;
23
24    system("pause");
25    return 0;
26  }
```

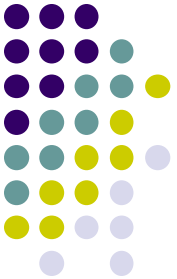
```
/* prog12_2 OUTPUT---

Window A:
win1.width = 50
win1.height = 40
----------------------*/
```

同時建立多個物件




```
12  int main(void)
13  {
14      CWin win1,win2;          // 宣告 CWin 類別型態的變數 win1 與 win2
15
16      win1.id='A';
17      win1.width=50;           } 設定 win1 物件的資料成員
18      win1.height=40;
19
20      win2.id='B';
21      win2.width=win1.width+20;    } 設定 win2 物件的資料成員
22      win2.height=win1.height+10;
23
24      cout << "Window " << win2.id << ":" << endl;
25      cout << "win2.width = " << win2.width << endl;
26      cout << "win2.height = " << win2.height << endl;
27
28      system("pause");
29      return 0;
30  }
```

```
/* prog12_3 OUTPUT---

Window B:
win2.width = 70
win2.height = 50
--------------------*/
```

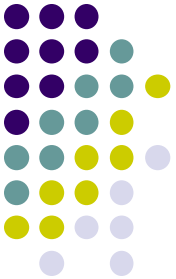


物件所佔的位元組

```
01  // prog12_4, 物件與類別所佔的位元組
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  class CWin                          // 定義視窗類別 CWin
06  {
07     public:                          // 設定資料成員為公有
08       char id;
09       int width;
10       int height;
11  };
12  int main(void)
13  {
14     CWin win1;                       // 宣告 CWin 類別型態的變數 win1
15
16     cout << "sizeof(win1) = " << sizeof(win1) << " bytes" << endl;
17     cout << "sizeof(CWin) = " << sizeof(CWin) << " bytes" << endl;
18
19     system("pause");
20     return 0;
21  }
```

- 利用sizeof() 函數可查詢物件與類別所佔的位元組

```
/* prog12_4 OUTPUT------

sizeof(win1) = 12 bytes
sizeof(CWin) = 12 bytes
----------------------*/
```



定義與使用函數

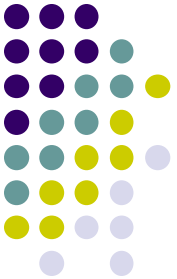
- 類別裡的函數可用如下的語法來定義

```
傳回值型態 函數名稱(型態 1 引數 1, 型態 2 引數 2,...)
{
    程式敘述 ;
    return 運算式;
}
```

函數的主體（body）

- 物件要呼叫封裝在類別裡的函數時，可用下列的語法

```
物件名稱.函數名稱(引數 1, 引數 2,...)
```



使用成員函數 (1/2)

- 加入area()函數到CWin類別裡

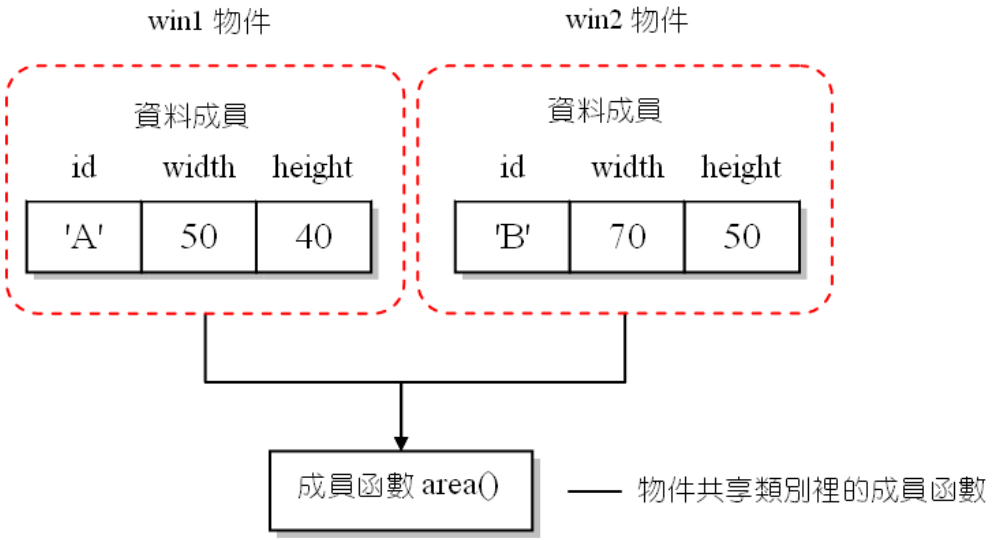
```
01  // prog12_5, 加入 area()函數到類別的定義裡
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  class CWin                          // 定義視窗類別 CWin
06  {
07    public:
08      char id;
09      int width;
10      int height;
11
12      int area(void)                  // 定義成員函數 area(), 用來計算面積
13      {
14          return width*height;
15      }
16  };
17
```

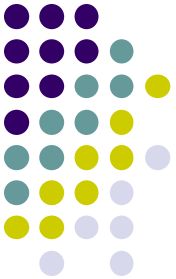
```
/* prog12_5 OUTPUT------
Window A:
Area = 2000
sizeof(win1) = 12 bytes
----------------------*/
```

使用成員函數 (2/2)

```
18  int main(void)
19  {
20     CWin win1;                    // 宣告 CWin 類別型態的變數 win1
21     win1.id='A';
22     win1.width=50;                // 設定 win1 的 width 成員為 50
23     win1.height=40;               // 設定 win1 的 height 成員為 40
24
25     cout << "Window " << win1.id << ":" << endl;
26     cout << "Area = " << win1.area() << endl;          // 計算面積
27     cout << "sizeof(win1) = " << sizeof(win1) << " bytes" << endl;
28
29     system("pause");
30     return 0;
31  }
```

```
/* prog12_5 OUTPUT------
Window A:
Area = 2000
sizeof(win1) = 12 bytes
----------------------*/
```





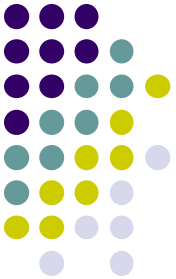
於類別裡定義多個函數 (1/2)

- 看一個同時具有兩個成員函數的例子

```
01  // prog12_6, 於類別裡定義多個函數
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  class CWin                    // 定義視窗類別 CWin
06  {
07    public:
08      char id;
09      int width;
10      int height;
11
12      int area()                // 定義成員函數 area(), 用來計算面積
13      {
14          return width*height;
15      }
16      int perimeter()           // 定義成員函數 perimeter(), 用來計算周長
17      {
18          return 2*(width + height);
19      }
20  };
```

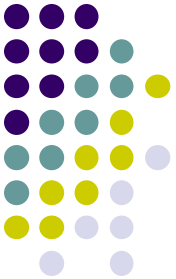
```
/* prog12_6 OUTPUT---
Window A:
Area = 2000
Perimeter = 180
--------------------*/
```

於類別裡定義多個函數 (2/2)



```
21
22  int main(void)
23  {
24     CWin win1;                  // 宣告 CWin 類別型態的變數 win1
25
26     win1.id='A';
27     win1.width=50;              // 設定 win1 的 width 成員為 50
28     win1.height=40;             // 設定 win1 的 height 成員為 40
29
30     cout << "Window " << win1.id << ":" << endl;
31     cout << "Area = " << win1.area() << endl;              // 計算面積
32     cout << "Perimeter = " << win1.perimeter() << endl; // 計算周長
33
34     system("pause");
35     return 0;
36  }
```

```
/* prog12_6 OUTPUT---

Window A:
Area = 2000
Perimeter = 180
--------------------*/
```

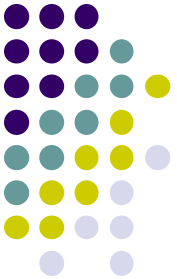


類別內資料成員的存取方式 (1/2)

- 在類別宣告的內部使用資料成員，可直接取用其名稱

```
class CWin                          // 定義視窗類別 CWin
{
   public:
     char id;
     int width;
     int height;

     int area()                     // 定義成員函數 area()
     {
         return width * height;
     }
};
```

可直接使用資料成員的名稱

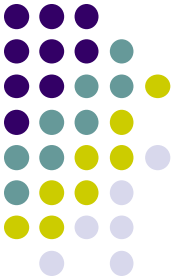


在類別定義的內部呼叫函數 (1/2)

- 在類別定義的內部呼叫函數

```
01  // prog12_8, 在類別定義的內部呼叫函數
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  class CWin                          // 定義視窗類別 CWin
06  {
07    public:                                          /* prog12_8 OUTPUT---
08      char id;
09      int width;                                     Window A, area=2000
10      int height;                                    ----------------------*/
11
12      int area(void)                  // 定義成員函數 area(), 用來計算面積
13      {
14        return width*height;
15      }
16      void show_area(void)            // 定義成員函數 show_area(), 顯示面積
17      {
18        cout << "Window " << id << ", area=" << area() << endl;
19      }
20  };
```

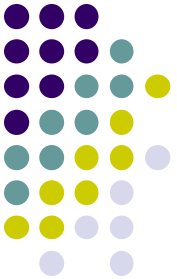
呼叫 area() 函數



在類別定義的內部呼叫函數 (2/2)

```
22  int main(void)
23  {
24     CWin win1;
25
26     win1.id='A';
27     win1.width=50;
28     win1.height=40;
29     win1.show_area();            // 顯示面積
30
31     system("pause");
32     return 0;
33  }
```

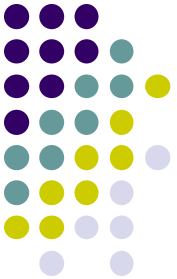
```
/* prog12_8 OUTPUT---
Window A, area=2000
--------------------*/
```



引數的傳遞 (1/2)

- 下面的程式是傳遞數值到函數中的範例

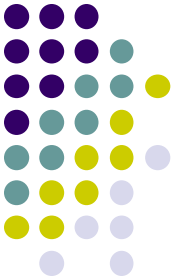
```
01  // prog12_9, 傳遞引數到函數裡
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  class CWin                          // 定義視窗類別 CWin
06  {
07     public:                                          /* prog12_9 OUTPUT---
08       char id;
09       int width;                                     Window B, area=2000
10       int height;                                    ----------------------*/
11
12       int area()                     // 定義成員函數 area(), 用來計算面積
13       {
14          return width*height;
15       }
16       void show_area(void)
17       {
18          cout << "Window " << id << ", area=" << area() << endl;
19       }
```



引數的傳遞 (2/2)

```
20        void set_data(char i,int w,int h)       // set data() 函數
21        {
22           id=i;                          // 設定 id 成員
23           width=w;                       // 設定 width 成員
24           height=h;                      // 設定 height 成員
25        }
26    };
27
28    int main(void)
29    {
30       CWin win1;
31
32       win1.set_data('B',50,40);
33       win1.show_area();
34
35       system("pause");
36       return 0;
37    }
```

```
/* prog12_9 OUTPUT---
Window B, area=2000
--------------------*/
```

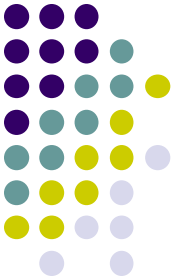


函數的多載 (1/2)

```
01  // prog12_11, 函數的多載
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  class CWin                          // 定義視窗類別 CWin
06  {
07    public:
08      char id;
09      int width;
10      int height;
11
12      int area()                      // 定義成員函數 area(), 用來計算面積
13      {
14        return width*height;
15      }
16      void show_area(void)
17      {
18        cout << "Window " << id << ", area=" << area() << endl;
19      }
20      void set_data(char i,int w,int h)     // 第一個 set_data()函數
21      {
22        id=i;
23        width=w;
24        height=h;
25      }
```

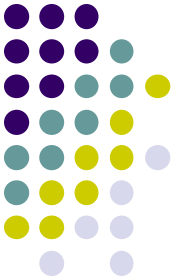
有三個引數

- 在類別裡定義的成員函數也可以多載，如下面的範例



函數的多載 (2/2)

```
26      void set_data(char i)                    // 第二個 set_data()函數
27      {
28         id=i;     只有一個引數
29      }
30      void set_data(int w,int h)               // 第三個 set_data()函數
31      {
32         width=w;     有兩個引數
33         height=h;
34      }
35  };
36
37  int main(void)
38  {
39     CWin win1,win2;
40
41     win1.set_data('A',50,40);
42     win2.set_data('B');
43     win2.set_data(80,120);
44
45     win1.show_area();
46     win2.show_area();
47
48     system("pause");
49     return 0;
50  }
```



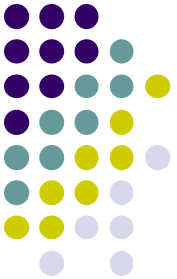
可能導致的危險 (1/2)

```
01  // prog12_12, 在類別定義的內部呼叫函數
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  class CWin              // 定義視窗類別 CWin
06  {
07     public:
08       char id;
09       int width;
10       int height;
11
12       int area(void)
13       {
14          return width*height;
15       }
16       void show_area(void)
17       {
18          cout << "Window " << id;
19          cout << ", area=" << area() << endl;
20       }
21  };
```

- 下面的範例示範從類別外部存取資料時，可能導致的危險

CWin 類別內部

```
/* prog12_12 OUTPUT---
Window A, area=-2000
----------------------*/
```



可能導致的危險 (2/2)

```
22
23  int main(void)
24  {
25     CWin win1;
26
27     win1.id='A';
28     win1.width=-50;    // 刻意將 width 成員設為-50
29     win1.height=40;
30     win1.show_area();  // 顯示面積
31
32     system("pause");
33     return 0;
34  }
```

CWin 類別外部

```
/* prog12_12 OUTPUT---
Window A, area=-2000
----------------------*/
```

建立私有成員

- 私有成員（private member）的設定方式如下

私有與公有成員的定義

```
class 類別名稱
{
    private:
        // 此處定義私有成員
    public:
        // 此處定義公有成員
}
```

```
class CWin      // 定義視窗類別CWin
{
   private:
      char id;
      int width;
      int height;
   public:
      int area(void)
      {
         return width*height;
      }
};
```

私有

公有

此處如果省略private關鍵字，id、width與height成員還是視為私有

錯誤示範 (1/2)

```
01  // prog12_13, 私有成員的使用範例
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;
10       int height;
11
12     public:
13       int area(void)                         // 成員函數 area()
14       {
15          return width*height;                // 在 CWin 類別內部，故可存取私有成員
16       }
17       void show_area(void)                   // 成員函數 show area()
18       {
19          cout << "Window "<< id <<", area=" << area() << endl;
20       }                                      // 在 CWin 類別內部，故可存取私有成員
21  };
22
```

- prog12_13為私有成員的使用範例（錯誤示範）

錯誤示範 (2/2)

```
23  int main(void)
24  {
25     CWin win1;
26
27     win1.id='A';
28     win1.width=-5;
29     win1.height=12;
30
31     win1.show_area();
32     system("pause");
33     return 0;
34  }
```

錯誤，在 CWin 類別外部，無法直接更改私有成員

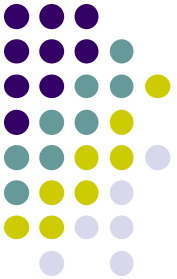
CWin 類別內部

```
class CWin
{
   private:
     char id;
     int width;
     int height;
      ....
};
```

在 CWin 類別外部，無法直接更改類別內部私有成員

CWin 類別外部

```
int main(void)
{
   CWin win1;
   win1.id='A';
   win1.width=-5;
   win1.height=12;
```

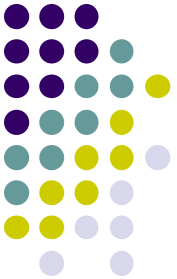


建立公有成員 (1/3)

- 下面的範例是利用公有函數存取私有成員

```
01  // prog12_14, 利用公有函數存取私有成員
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;                               // 私有資料成員
09       int width;                             // 私有資料成員
10       int height;                            // 私有資料成員
11
12     public:
13       int area(void)                         // 公有成員函數 area()
14       {
15           return width*height;
16       }
17       void show_area(void)                   // 公有成員函數 show area()
18       {
19          cout<<"Window "<< id <<", area=" << area() << endl;
20       }
```

```
/* prog12_14 OUTPUT---

Window A, area=2000
----------------------*/
```



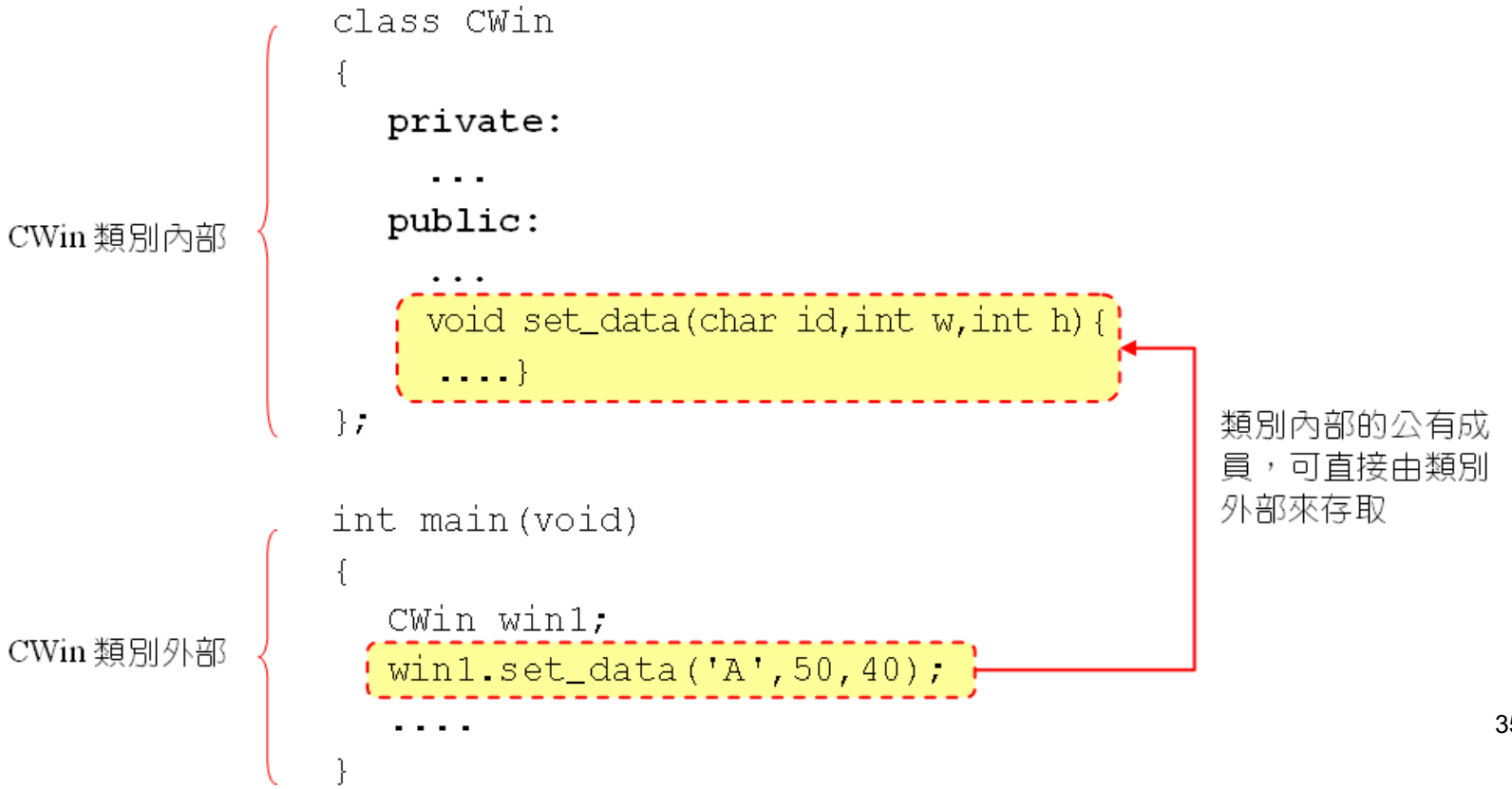
建立公有成員 (2/3)

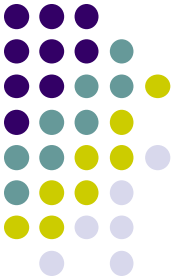
```
21        void set_data(char i,int w,int h) // 公有成員函數 set data()
22        {
23           id=i;
24           if(w>0 && h>0)
25           {
26              width=w;
27              height=h;
28           }
29           else
30              cout << "input error" << endl;
31        }
32    };
33
34    int main(void)
35    {
36       CWin win1;
37
38       win1.set_data('A',50,40);
39       win1.show_area();                    // 顯示面積
40       system("pause");
41       return 0;
42    }
```

```
/* prog12_14 OUTPUT---
Window A, area=2000
----------------------*/
```

建立公有成員 (3/3)

- 從prog12_14可看出，唯有透過公有成員函數，才能存取私有成員



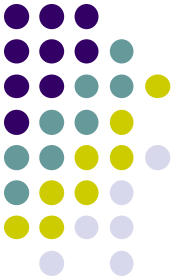


私有的成員函數 (1/2)

- 函數不想被外界所呼叫，一樣可設為私有，如下面的程式

```
01  // prog12_15, 私有的成員函數
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;                            // 私有資料成員
09      int width;                          // 私有資料成員
10      int height;                         // 私有資料成員
11
12      int area(void)                      // 私有成員函數 area()
13      {
14          return width*height;
15      }
16
17    public:
18      void show_area(void)                // 公有成員函數 show area()
19      {
20         cout << "Window " << id << ", area=" << area() << endl;
21      }
```

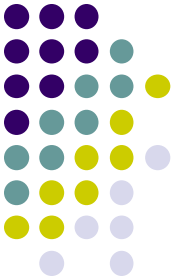
```
/* prog12_15 OUTPUT---
Window A, area=2000
----------------------*/
```



私有的成員函數 (2/2)

```
22        void set_data(char i,int w,int h)  // 公有成員函數 set_data()
23        {
24           id=i;
25           if(w >0 && h>0)
26           {
27              width=w;
28              height=h;
29           }
30           else
31              cout << "input error" << endl;
32        }
33    };
34
35    int main(void)
36    {
37       CWin win1;
38
39       win1.set_data('A',50,40);
40       win1.show_area();                    // 顯示面積
41       system("pause");
42       return 0;
43    }
```

```
/* prog12_15 OUTPUT---
Window A, area=2000
----------------------*/
```

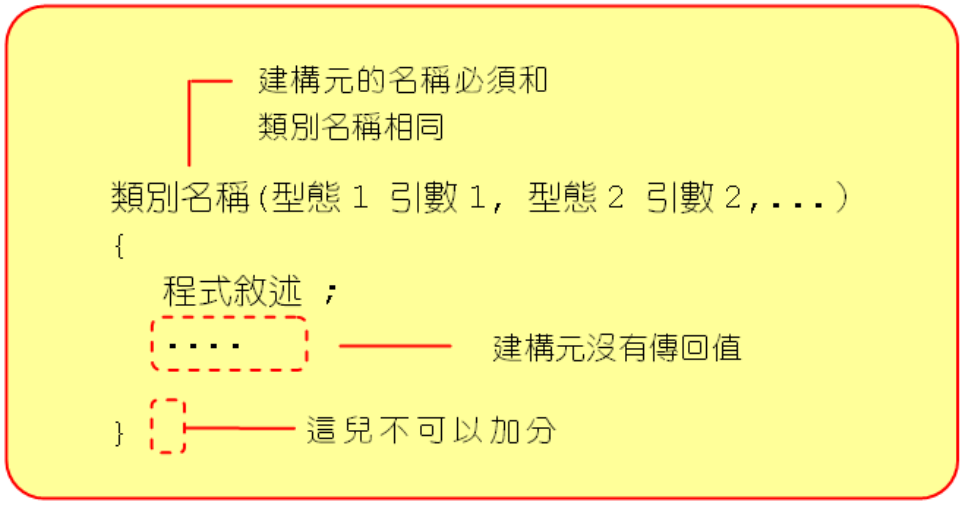


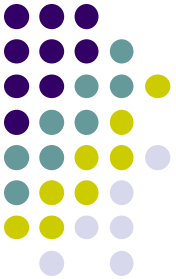
資料的封裝

- 封裝（encapsulation）
  - 把資料成員和成員函數依功能劃分為「私有」與「公有」，並且包裝在一個類別內來保護私有成員，使得它不會直接受到外界的存取

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



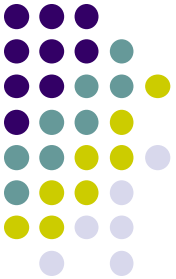


建構元的使用範例 (1/3)

- 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    }
```

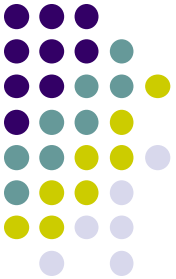
```
/* prog13_1 OUTPUT--------------

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

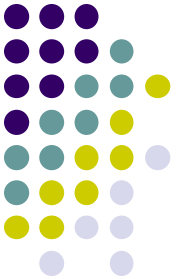


建構元的多載 (1/2)

- 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       }
```

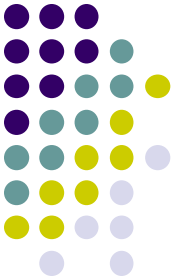
```
/* prog13_3 OUTPUT-----------------
CWin(char,int,int) 建構元被呼叫了...
CWin(int,int) 建構元被呼叫了...
Window A: width=50, height=40
Window Z: width=80, height=120
-----------------------------------*/
```



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



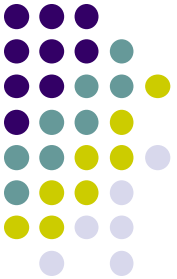
預設建構元 (1/5)

- 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

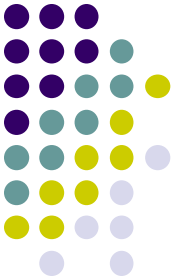
- 如果沒有撰寫建構元，C++會提供一個沒有引數的預設建構元
- 如果程式裡有撰寫建構元，C++不會再提供沒有引數的預設建構元



預設建構元 (2/5)

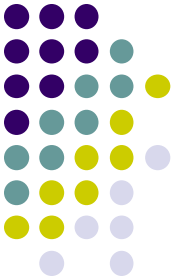
- prog13_4是加入一個預設建構元的完整範例

```
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  }
```

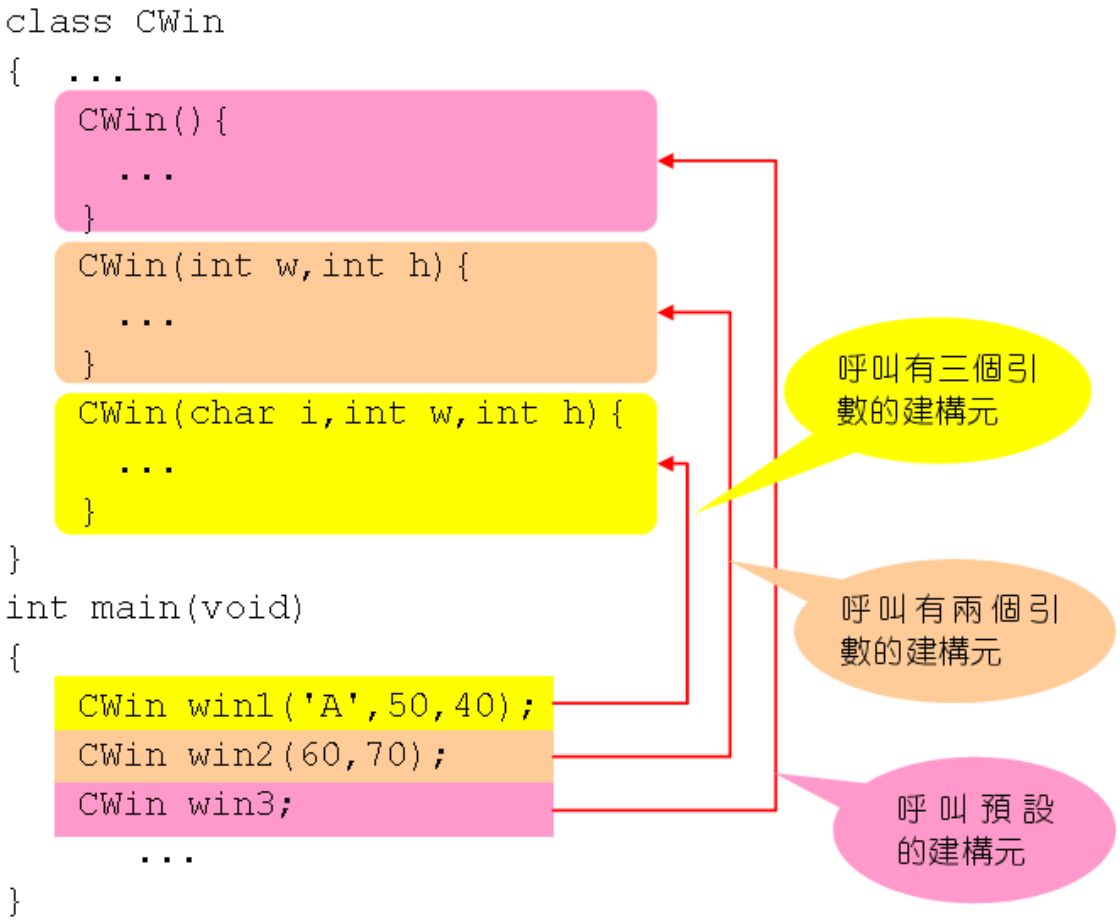
```
/* prog13_4 OUTPUT------------------
CWin(char,int,int) 建構元被呼叫了...
CWin(int,int) 建構元被呼叫了...
預設建構元被呼叫了...
Window A: width=50, height=40
Window Z: width=80, height=120
Window D: width=100, height=100
-----------------------------------*/
```

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




-The End-