第十二章
類別的基本架構

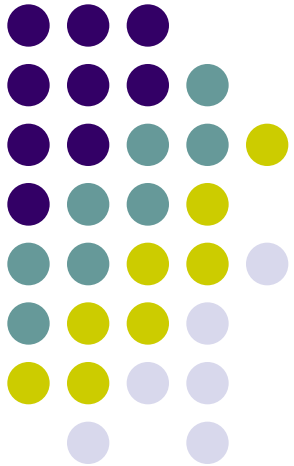
學習類別的觀念

認識與撰寫成員函數

認識引數的傳遞與多載的方式

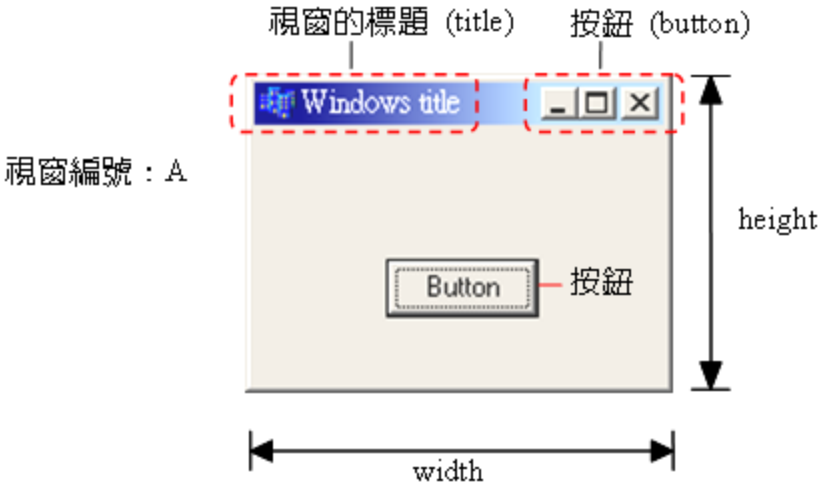
建立公有與私有成員

瞭解友誼函數

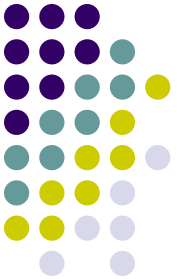


類別 (1/2)

- 類別可看成是結構的擴充，它不只擁有結構的所有功能，還可定義函數在類別裡。
- 下圖為視窗示意圖，稍後將分別以結構和類別的概念來撰寫它



類別 (2/2)



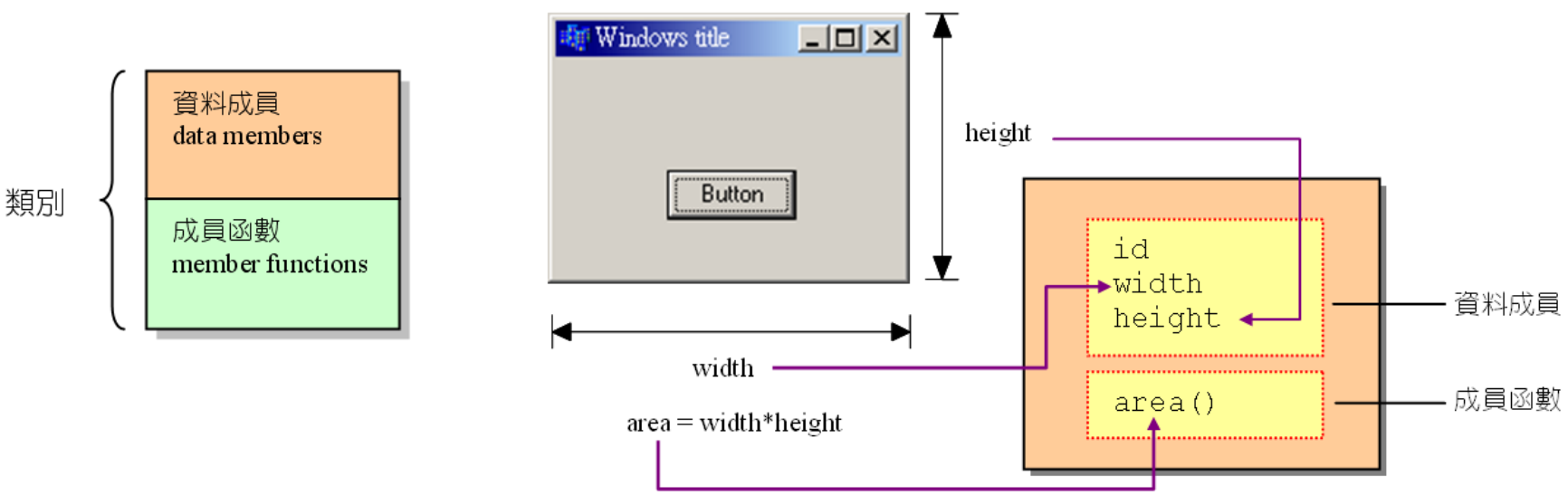
- 下面的範例是利用結構來表示視窗

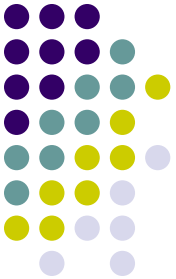
```
01  // prog12_1, 利用結構來表示視窗
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  struct Win                          // 利用結構來定義視窗(Window)
06  {
07     char id;
08     int width;                       // Win 結構的 width 成員
09     int height;                      // Win 結構的 height 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

- 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：

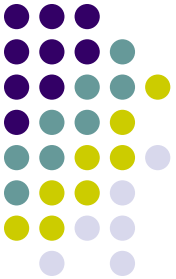
```
class 類別名稱
{
  public:
    資料型態 變數名稱;
    ...

    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;
    }
  ...
};
```

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



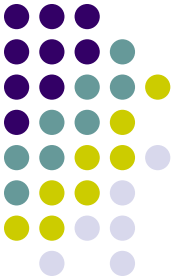
定義「視窗」類別 (1/2)

- 以前述的視窗為例，可定義如下的「視窗」類別

```
class CWin                          // 定義視窗類別 CWin
{
   public:                          // 在此以下宣告之成員的屬性皆屬公有
      char id;                      // 宣告資料成員 id
      int width;                    // 宣告資料成員 width
      int height;                   // 宣告資料成員 height

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

宣告資料成員

定義成員函數



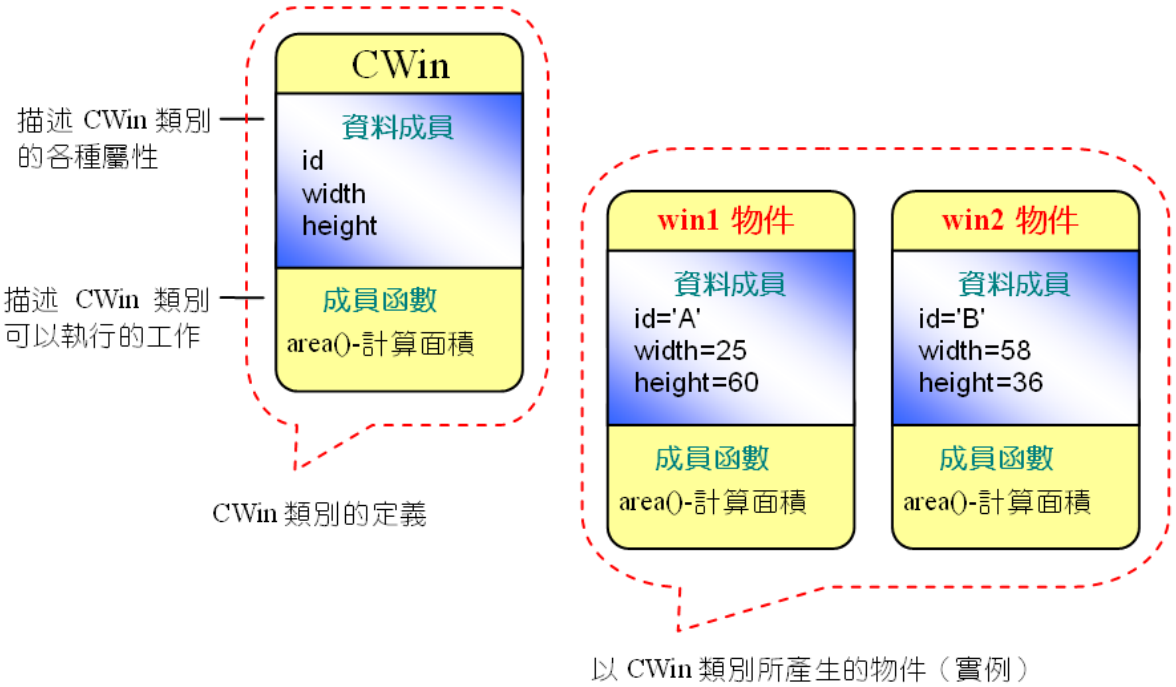
定義「視窗」類別 (2/2)

- 成員存取的控制權
  - public設定在它之後的成員，其屬性均為公有（public），成員可隨意的在類別外部做存取
  - 類別成員的屬性設定為private（私有），則成員只能在類別內部做存取的動作
  - 類別內的成員，若無宣告為public，則預設的屬性為private

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
```



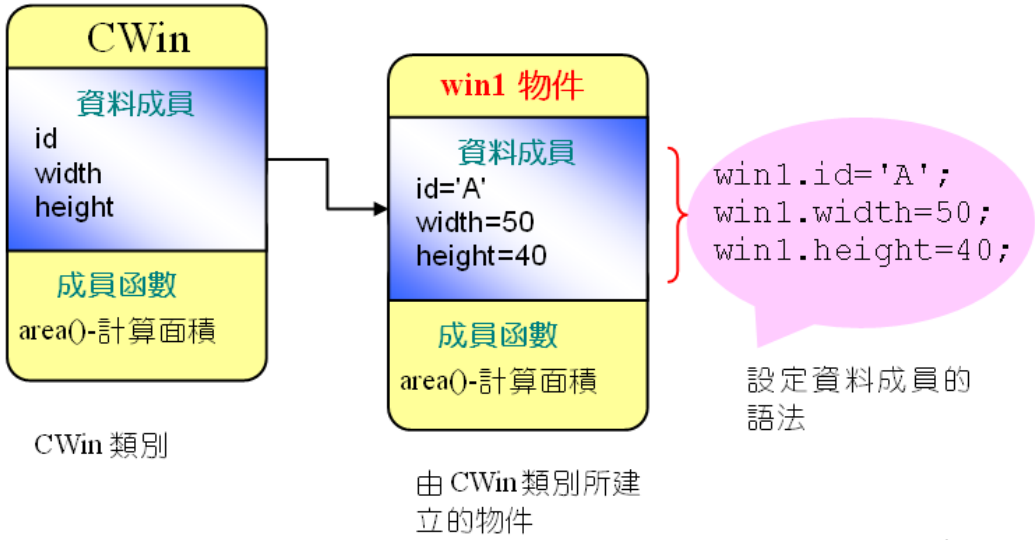
建立新的物件 (2/2)

- 存取物件的內容

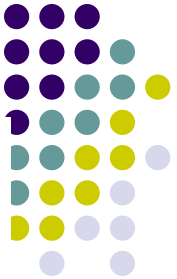
物件名稱.特定的資料成員

- 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



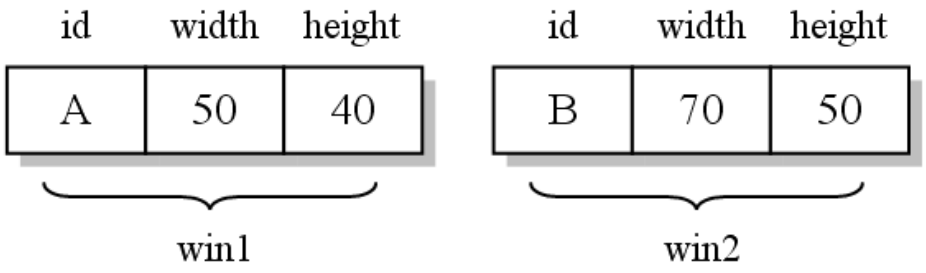
```
01  // prog12_2, 第一個類別程式
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  class CWin                          // 定義視窗類別 CWin
06  {
07    public:                           // 設定資料成員為公有
08      char id;
09      int width;
10      int height;
11  };
12  int main(void)
13  {
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  }
```

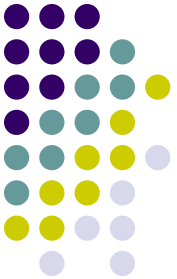
```
/* prog12_2 OUTPUT---

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

同時建立多個物件 (1/2)

- 下面的程式示範同時建立數個物件，並存取資料成員

```
01  // prog12_3, 建立物件與資料成員的存取
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  };
  /* prog12_3 OUTPUT---
  Window B:
  win2.width = 70
  win2.height = 50
  ---------------------*/
```

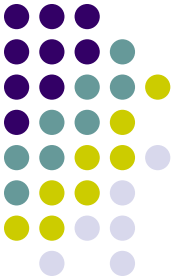




同時建立多個物件 (2/2)

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

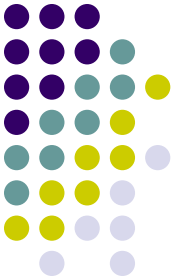


物件所佔的位元組

```
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

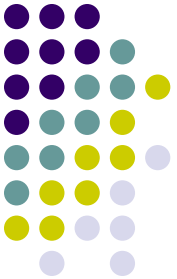
- 類別裡的函數可用如下的語法來定義

```
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

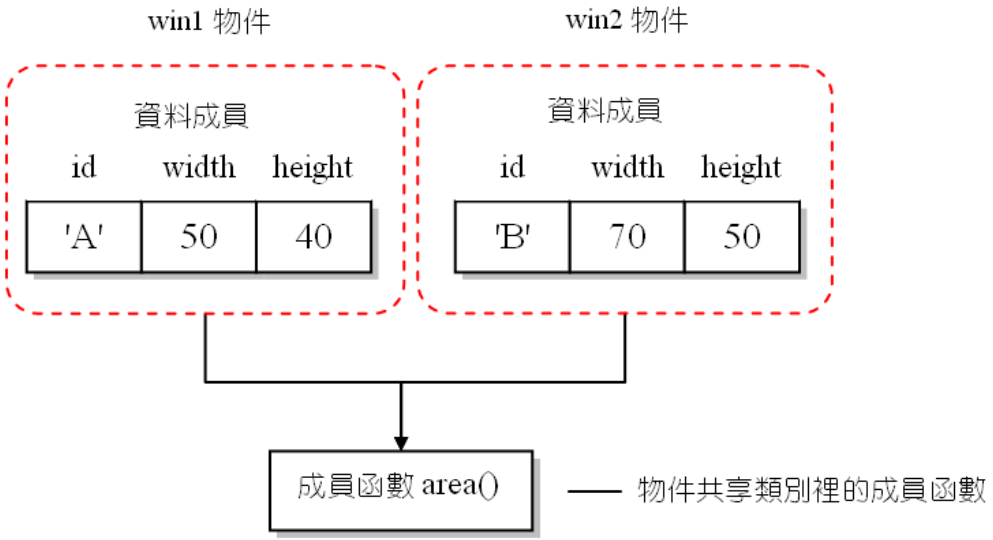
```
01  // prog12_5, 加入 area()函數到類別的定義裡
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  class CWin                          // 定義視窗類別 CWin
06  {
07    public:
08      char id;
09      int width;
10      int height;
11
12      int area(void)                  // 定義成員函數 area(),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

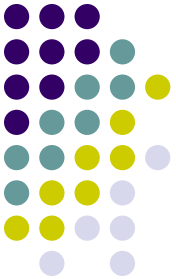
```
/* prog12_5 OUTPUT------
Window A:
Area = 2000
sizeof(win1) = 12 bytes
----------------------*/
```

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





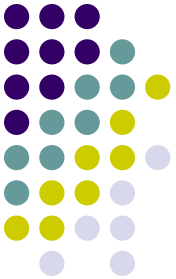
於類別裡定義多個函數 (1/2)

- 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  };
```

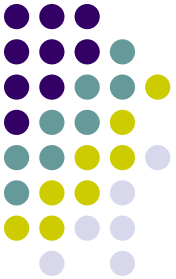
```
/* prog12_6 OUTPUT---
Window A:
Area = 2000
Perimeter = 180
--------------------*/
```

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

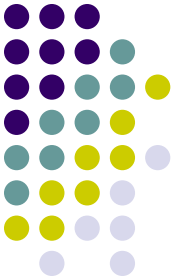


函數的位置 (1/2)

- 成員函數定義在類別之外，只需在類別的定義內加入函數的原型即可

```
01  // prog12_7, 將函數定義於類別之外
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       int area(void);                // 成員函數 area()的原型
12  };
13
14  int CWin::area(void)                // 定義 area()函數
15  {
16     return width*height;
17  }
18
```

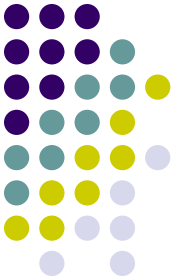
```
/* prog12_7 OUTPUT---

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



函數的位置 (2/2)

```
19  int main(void)
20  {
21     CWin win1;                          // 宣告 CWin 類別型態的變數 win1
22
23     win1.id='A';
24     win1.width=50;
25     win1.height=40;
26
27     cout << "Window " << win1.id << ":" << endl;
28     cout << "Area = " << win1.area() << endl;
29
30     system("pause");
31     return 0;
32  }
```

```
/* prog12_7 OUTPUT---

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



範疇解析運算子

- 範疇解析運算子的用法

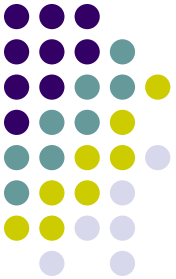
範疇解析運算子，用來表示 area()函數
是屬於 CWin 類別

```
int CWin::area(void)      // 定義 area()函數
{
    return width*height;
}
```

- 也可以在函數前面加上inline關鍵字

也可以在函數前面加上 inline 關鍵字

```
inline int CWin::area(void)      // 定義 area()函數
{
    return width*height;
}
```

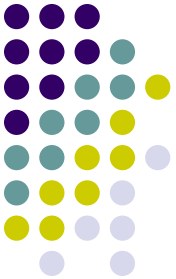


類別內資料成員的存取方式 (1/2)

- 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



類別內資料成員的存取方式 (2/2)

- 在資料成員前面的this代表指向「取用此一資料成員之物件」的指標

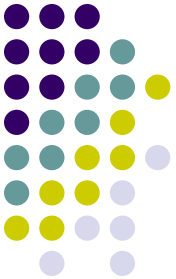
```
class CWin      // 定義視窗類別 CWin
{
   public:
     char id;
     int width;
     int height;
     int area()     // 定義成員函數 area()
     {
         return this->width * this->height;
     }
};
```

```
win1.area(); /* 此時程式碼裡的 this
                代表 win1 物件   */
```

在資料成員前面加上 this，此時的 this 即代表指向「取用此一資料成員之物件」的指標

```
this.資料成員名稱   // 錯誤，this 是指標

this->資料成員名稱 // 正確的使用 this 指標
```

在類別定義的內部呼叫函數 (1/3)

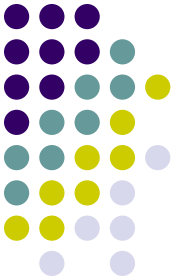


- 在類別定義的內部呼叫函數

```
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:
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      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  };
```

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

呼叫 area() 函數

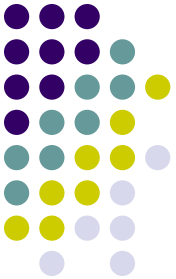


在類別定義的內部呼叫函數 (2/3)

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

在類別定義的內部呼叫函數 (3/3)



- 把prog12_8的show_area() 改成如下的敘述

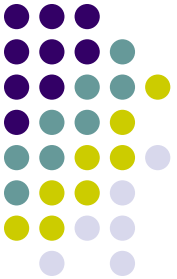
```
void show area(void)              // 定義成員函數 show area(), 用來顯示面積
{
    cout << "Window " << id << ", area=" << this->area() << endl;
}
```

在類別的定義內呼叫其它的函數，可在該函數之前加上 this 關鍵字，此時的 this 即代表指向「取用此一函數的物件」之指標

- 假設在main() 主程式裡有這麼一行敘述

```
win1.show_area();          // 利用 win1 物件呼叫 show_area()函數
```

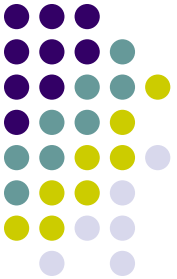
- 則this即代表指向win1物件的指標



引數的傳遞 (1/3)

- 下面的程式是傳遞數值到函數中的範例

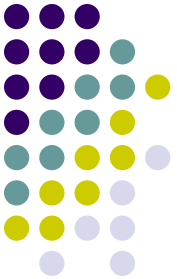
```
01  // prog12_9, 傳遞引數到函數裡
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  class CWin                          // 定義視窗類別 CWin
06  {
07     public:                                          /* prog12_9 OUTPUT---
08       char id;
09       int width;                                     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 (2/3)

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

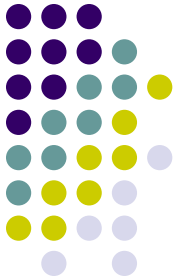


引數的傳遞 (3/3)

- 下面的程式片段是將prog12_9中的區域變數i、w與h標示出來

```
class CWin                  // 定義視窗類別 CWin
{
   public:
     ......
     void set data(char i, int w, int h)          // set data() 函數
     {
        id=i;               // 設定 id 成員
        width=w;            // 設定 width 成員
        height=h;           // 設定 height 成員
     }
};
```

i、w 與 h 均為區域變數，一離開此範圍，變數 i、w 與 h 即屬無效



傳遞物件到函數裡 (1/2)

- 下面是傳遞物件到函數裡的範例

```
01  // prog12_10, 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     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
17       void set data(char i,int w,int h)       // set data() 函數
18       {
19          id=i;                       // 設定 id 成員
20          width=w;                    // 設定 width 成員
21          height=h;                   // 設定 height 成員
22       }
23  };
```

```
/* prog12_10 OUTPUT---
Window B, area=2000
----------------------*/
```

傳遞物件到函數裡 (2/2)

```
24
25  void show area(CWin win)        // 把 show area()定義成一般的函數
26  {
27     cout << "Window " << win.id <<", area=" << win.area() << endl;
28  }
29
30  int main(void)
31  {
32     CWin win1;
33
34     win1.set data('B',50,40);    // 由 win1 物件呼叫 set data()函數
35     show area(win1);             // 傳遞 win1 物件到 show area()函數裡
36
37     system("pause");
38     return 0;
39  }
```

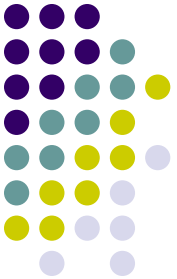
由 win1 物件呼叫 set_data()函數

```
win1.set data('B',50,40);
```

```
show_area(win1);
```

傳遞 win1 物件到 show_area()函數裡

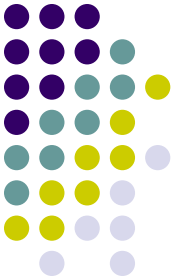
```
/* prog12_10 OUTPUT---
Window B, area=2000
----------------------*/
```



函數的多載 (1/2)

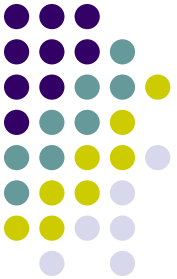
- 在類別裡定義的成員函數也可以多載，如下面的範例

```
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      }
```



函數的多載 (2/2)

```
26        void set_data(char i)                    // 第二個 set_data()函數
27        {
28           id=i;
29        }
30        void set_data(int w,int h)               // 第三個 set_data()函數
31        {
32           width=w;
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    }
```



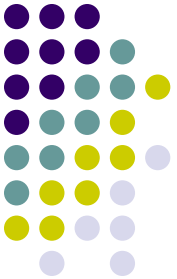
可能導致的危險 (1/2)

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

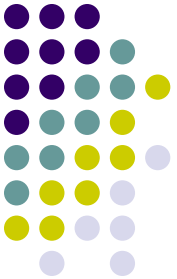


可能導致的危險 (2/2)

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



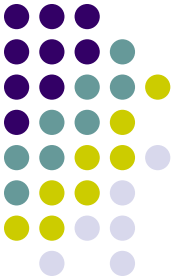
建立私有成員 (1/3)

- 私有成員（private member）的設定方式如下

```
class 類別名稱
{
   private:
      私有的成員 (包含資料與成員函數)
      ....
   public:
      公有的成員 (包含資料與成員函數)
      ....
}
```

定義於此部份的成員皆為私有

定義於此部份的成員皆為公有



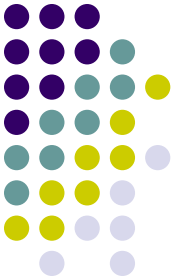
建立私有成員 (2/3)

- 下面的程式碼設定id、width與height資料成員為私有，area() 函數為公有

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

   public:
     int area(void)
     {
        return width*height;
     }
};
```

id、width 與 height 成員皆為私有

成員函數 area()為公有



建立私有成員 (3/3)

- C++成員的預設屬性為私有，因此id、width與height成員會被視為私有

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

   public:
     int area(void)
     {
        return width*height;
     }
};
```

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

成員函數 area()為公有

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
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

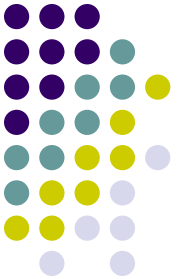
CWin 類別內部

```
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;
```

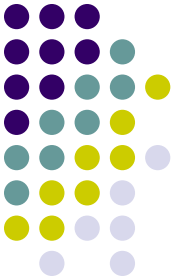


建立公有成員 (1/3)

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



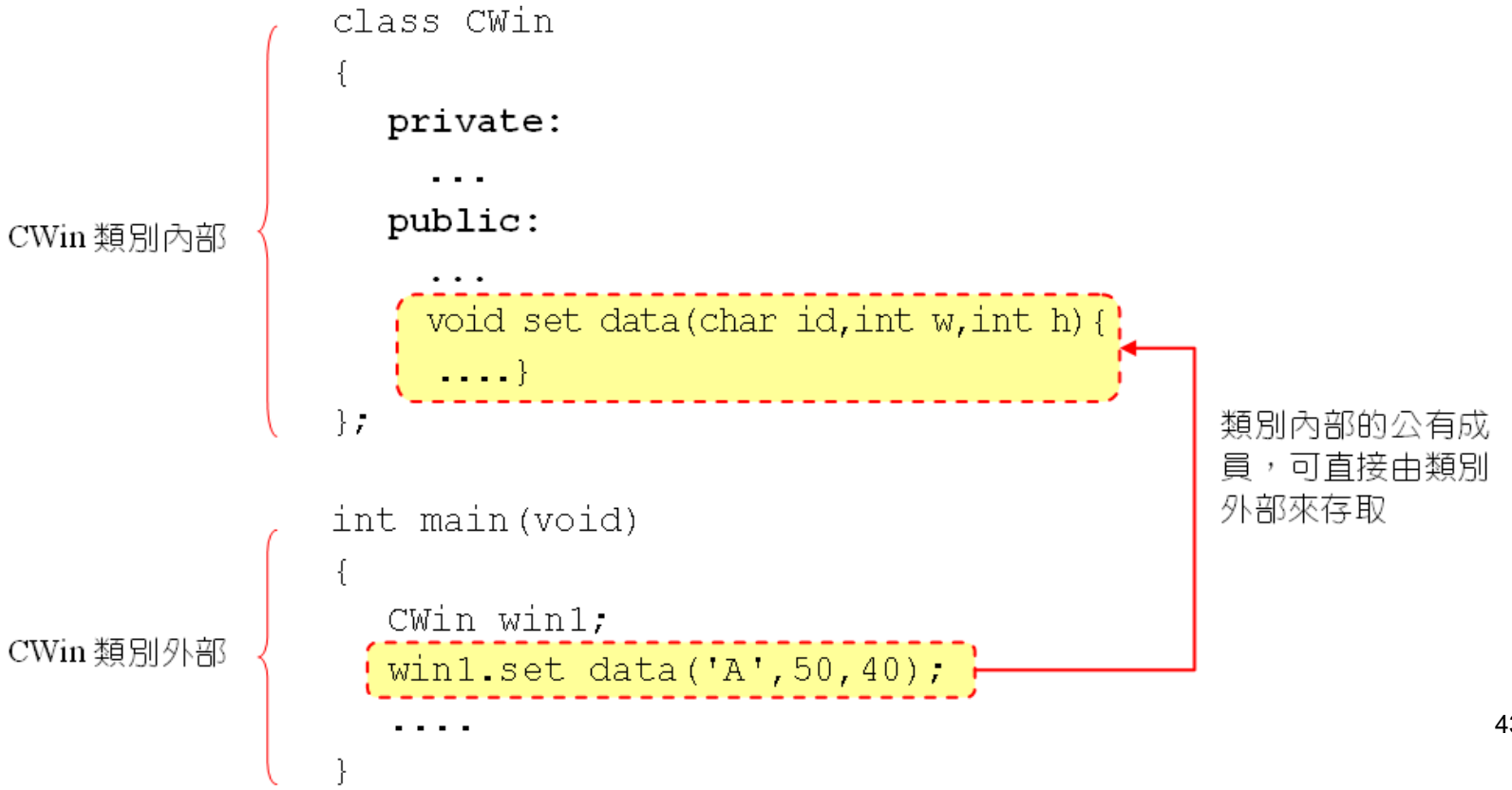
建立公有成員 (2/3)

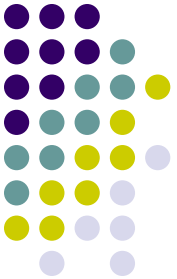
```
21        void set data(char i,int w,int h) // 公有成員函數 set data()
22        {
23           id=i;
24           if(w>0 && h>0)
25           {
26              width=w;
27              height=h;
28           }
29           else
30              cout << "input error" << endl;
31        }
32    };
33
34    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



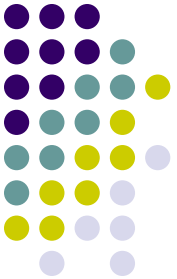


私有的成員函數 (1/2)

- 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       }
```

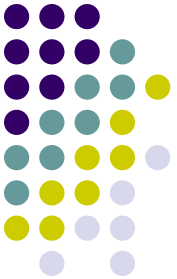
```
/* prog12_15 OUTPUT---
Window A, area=2000
----------------------*/
```



私有的成員函數 (2/2)

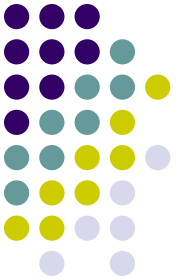
```
22        void set data(char i,int w,int h)  // 公有成員函數 set data()
23        {
24           id=i;
25           if(w >0 && h>0)
26           {
27              width=w;
28              height=h;
29           }
30           else
31              cout << "input error" << endl;
32        }
33    };
34
35    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
----------------------*/
```



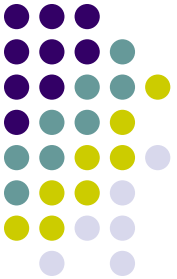
資料的封裝

- 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



友誼函數

- 友誼函數不屬於某個類別，但它可存取該類別內的成員
- 友誼函數以friend做宣告，放置位置可以有下列兩種：
  - 把友誼函數的原型放在類別的定義內，把定義放在類別外
  - 把友誼函數直接定義在類別內
- 友誼函數不會太複雜的話，編譯器會自動把它當成inline函數

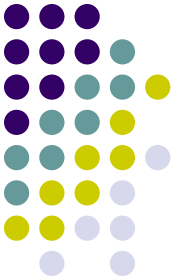


使用友誼函數 (1/3)

- 下面的程式是使用友誼函數的例子

```
01  // prog12_16, 友誼函數的使用
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  class CWin                                    // 定義視窗類別 CWin
06  {
07     public:
08       void set data(char i,int w, int h)       // 設定數值的函數
09       {
10          id=i;
11          width=w;
12          height=h;
13       }
14     private:
15       char id;
16       int width;
17       int height;
18
19     friend void show member(CWin);             // 友誼函數的原型
20  };
21
```

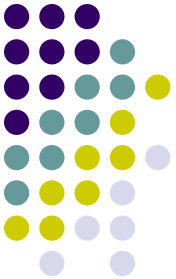
```
/* prog12_16 OUTPUT----------------

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



使用友誼函數 (2/3)

```
21
22  void show member(CWin w)                          // 定義友誼函數
23  {
24      cout << "Window " << w.id;
25      cout << ": width = " << w.width;
26      cout << ", height = " << w.height << endl;
27  }
28
29  int main(void)
30  {
31     CWin win1,win2;
32
33     win1.set data('A',50,40);                      // 呼叫 set data()設值
34     win2.set data('B',80,60);
35     show member(win1);
36     show member(win2);
37
38     system("pause");
39     return 0;
40  }
```

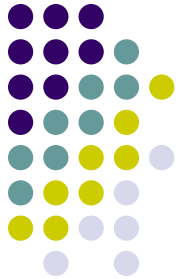
```
/* prog12_16 OUTPUT----------------
Window A: width = 50, height = 40
Window B: width = 80, height = 60
---------------------------------*/
```



使用友誼函數 (3/3)

- 下圖顯示友誼函數存取的權限

```
class CWin
{
   Public:
     ...
   private:
     ...
   friend void show member(CWin);
};

void show member(CWin w)        // 友誼函數
{
   cout<< "Window "<< w.id;
   cout<< ": width = "<< w.width;
   cout<< ", height = "<< w.height << endl;
}
```



友誼函數的注意事項

- 您可以在類別內只定義完整的友誼函數，或只定義它的原型，而將其完整的定義放置在類別外，如prog12_16即是
- 雖然友誼函數的原型或定義是放在類別內，但它並不屬於類別的成員，自然也就不具有公有或私有的特性


-The End-