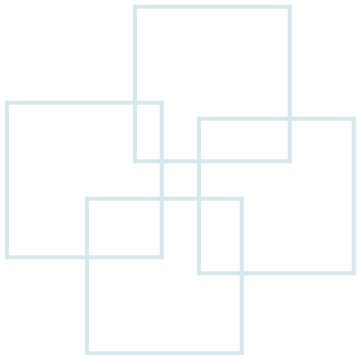


Chapter 17

C to C++

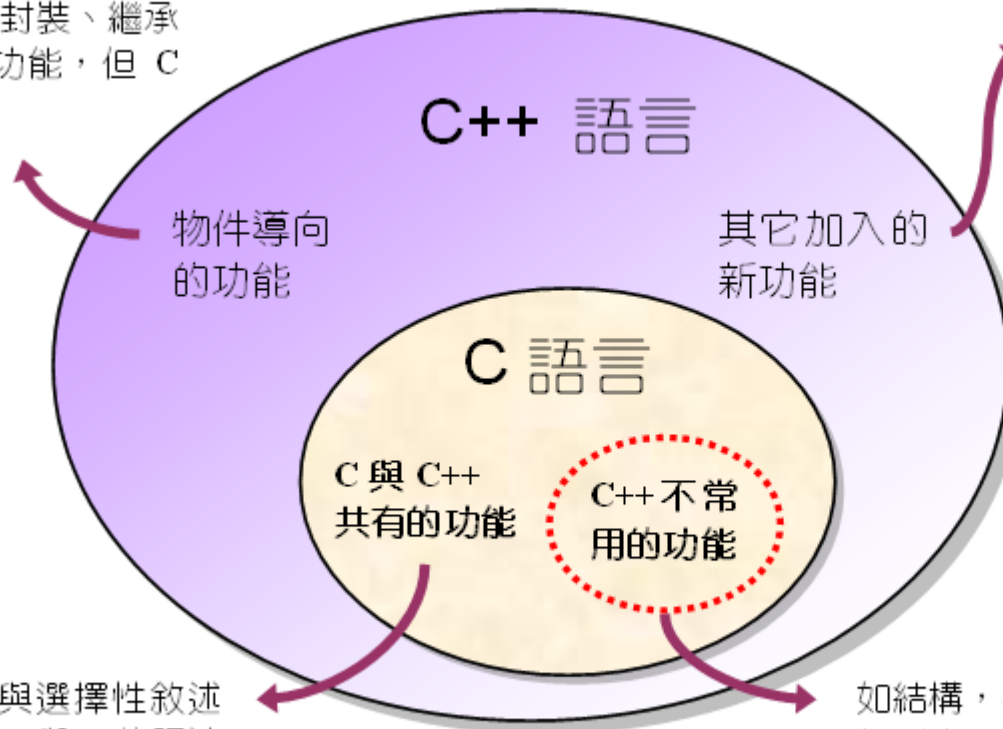




Relation Between C and C++

C++ 具有封裝、繼承與多型等功能，但 C 語言沒有

例如 C++ 多了布林型態 `bool`，但 C 語言沒有



如迴圈與選擇性敘述等，C++ 與 C 的語法均完全相同

如結構，在 C++ 裡多用類別來取代結構



New Features in C++

- **Encapsulation** (封装)

- Data hiding
- Data protection

- **Inheritance** (繼承)

- **Polymorphism** (多型)

- **Overloading**: Different functions are called when different parameter are used.
- **Overriding**: Different functions are called when different objects are used.
- **Dynamic binding**:
 - Virtual function: Determine the invoked function at run time.
 - Abstract class: Abstract class can only be inherited. The classes inheriting abstract classes can be implemented.
 - RRTI (Run-Time Type Identification): Determine object types at run time



Hello World C++

```
01 // 簡單的 C++ 程式
02 #include <iostream>           // 含括 iostream 檔案
03 #include <cstdlib>            // 含括 cstdlib 檔案
04 using namespace std;        // 使用 std 名稱空間
05 int main(void)
06 {
07     char ch='T';
08     int a=12;
09     float b=12.63F;
10
11     cout << ch << "是字元" << endl; // 印出字元 ch 的內容
12     cout << a << "是整數" << endl;  // 印出變數 a 的內容
13     cout << b << "是浮點數" << endl; // 印出變數 b 的內容
14
15     system("pause");
16     return 0;
17 }
```

Standard library

/* OUTPUT--

T 是字元
12 是整數
12.63 是浮點數

***/**



ANSI/ISO C++

- ANSI/ISO C++ was announced in 1997
 - Existing C libraries are also supported in C++
 - New libraries (only) supported by C++ exclude the file extension **.h**
 - `include <iostream>` -> No **.h**
 - The corresponding C++ libraries ported from C libraries add character **c** in the beginning of the header file name
 - `math.h` -> `cmath`
 - Every function, class, and object names are contained in the name space **std**
 - `using namespace std`



Inputs from Keyboard

```
01 // 利用 cin 輸入資料
02 #include <iostream>
03 #include <cstdlib>
04 using namespace std;
05 int main(void)
06 {
07     int x;
08     float y;
09     cout << "請輸入一個整數:";
10     cin >> x;           // 由鍵盤讀取一整數，並指定給變數 x 存放
11     cout << "請輸入一個浮點數:";
12     cin >> y;           // 由鍵盤讀取一浮點數，並指定給變數 y 存放
13     cout << x << "+" << y << "=" << x+y << endl; // 計算並輸出 x+y
14
15     system("pause");
16     return 0;
17 }
```

/* OUTPUT----

請輸入一個整數:**12**

請輸入一個浮點數:**26.87**

12+26.87=38.87

-----*/



New Data Type: bool

- bool: *true* or *false*

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



Example: bool

```
01 // 布林變數的使用
02 #include <iostream>
03 #include <cstdlib>
04 using namespace std;
05 int main(void)
06 {
07     bool is_odd;           // 宣告布林型態的變數 is_odd
08     int num;
09     cout << "請輸入一個正整數: ";
10     cin >> num;
11     if(num%2!=0)
12         is_odd=true;     // 如果 num 是奇數, 設定 is_odd 為 true
13     else
14         is_odd=false;    // 如果 num 是偶數, 設定 is_odd 為 false
15
16     if(is_odd)
17         cout << num << "是奇數" << endl;
18     else
19         cout << num << "是偶數" << endl;
20     system("pause");
21     return 0;
22 }
```

/* OUTPUT---

請輸入一個正整數: **53**

53 是奇數

-----*/



Variables

- Variables can be declared in any place (including in for loop).

```
01 // 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 }
```

/* 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
-----*/

} 變數 i 的有效範圍



Overloading (多載)

- Functions are called according to its parameters.

```
int main(void)
{
    ...
    function(a);
    function(a,b);
}

function(n)
{
    ....
}

function(n1,n2)
{
    ....
}
```



Overloading with Different Data Types

```
01 // 函數多載的範例--引數個數相同，但型態不同
02 #include <iostream>
03 #include <cstdlib>
04 using namespace std;
05 void show(int);
06 void show(double);
07 int main(void)
08 {
09     int a=26;
10     double b=3.14;
11     show(a); // 傳入整數到 show() 函數裡
12     show(b); // 傳入倍精度浮點數到 show() 函數裡
13     system("pause");
14     return 0;
15 }
16 void show(int num) // show() 函數，可接收一個整數
17 {
18     cout << num << "是一個整數" << endl;
19 }
20 void show(double num) // show() 函數，可接收一個倍精度浮點數
21 {
22     cout << num << "是一個倍精度浮點數" << endl;
23 }
```

/* OUTPUT--

26 是一個整數

3.14 是一個倍精度浮點數

***/**



Overloading with Different Number of Parameters

```
01 // 函數多載的範例--引數個數不同
02 #include <iostream>
03 #include <cstdlib>
04 using namespace std;
05 void star(void);
06 void star(int);
07 int main(void)
08 {
09     star();           // 呼叫沒有引數的 star()
10     star(9);         // 呼叫有一個整數引數的 star()
11     system("pause");
12     return 0;
13 }
14 void star(void)      // 定義 star(void) 函數
15 {
16     cout << "印出 5 個星號: *****" << endl;
17 }
18 void star(int num)  // 定義 star(int) 函數
19 {
20     cout << "印出" << num << "個星號: ";
21     for(int i=1;i<=num;i++)
22         cout << "*";
23     cout << endl;
24 }
```

/* OUTPUT----



印出 5 個星號: *****

印出 9 個星號: ***********

***/**



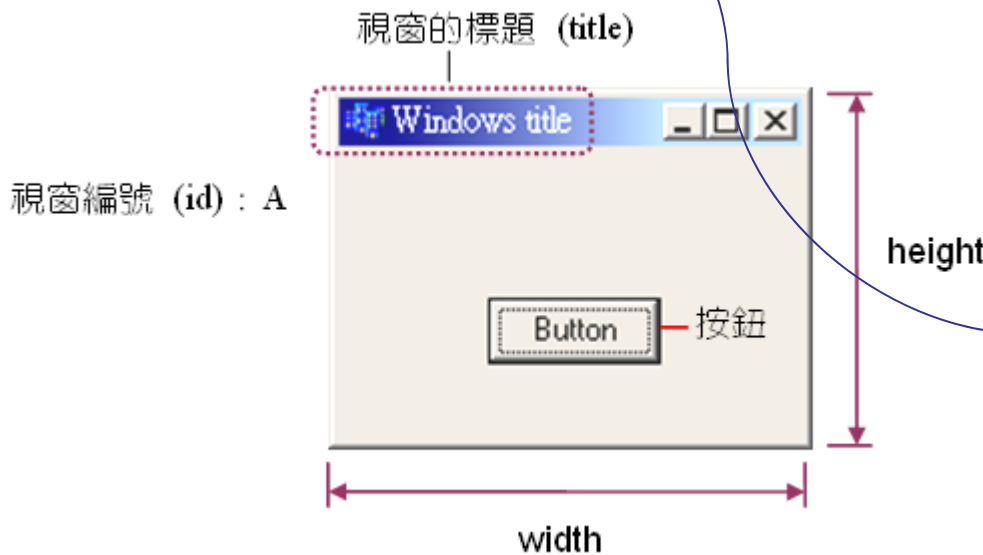
Class

- Using a typical window as an example:
 - Using **struct** 
 - Using **class** 

```

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

int area()
{
    reutrn width*height
}
  
```

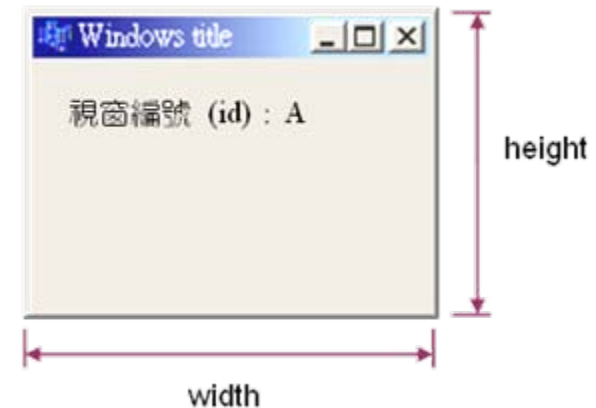


```

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



Example: *struct*



```

01 // 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
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) // 定義函數 area(), 用來計算面積
13 {
14     return w.width*w.height; // 面積=寬*高
15 }
16

```



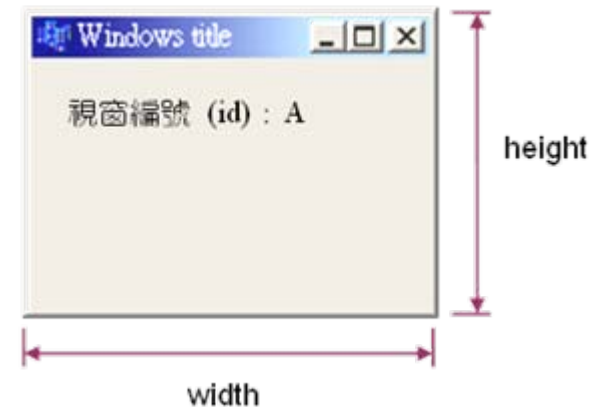
Example: *struct* (Cont.)

```
17 int main(void)
18 {
19     Win win1;           // 宣告 Win 結構的物件 win1
20
21     win1.id='A';
22     win1.width=50;     // 設定 width 為 50
23     win1.height=40;   // 設定 height 為 40
24
25     cout<<"Window "<< win1.id<<" , area="<<area(win1)<<endl;
26     system("pause");
27     return 0;
28 }
```

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

```
Window A, area=2000
```

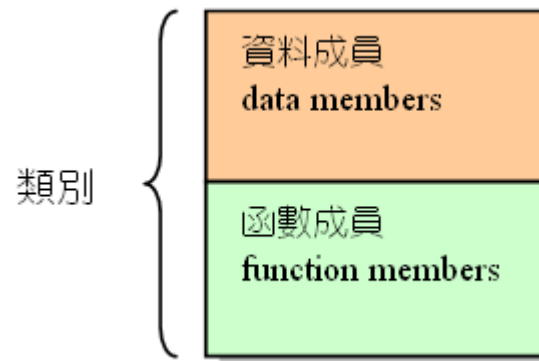
```
---*/
```





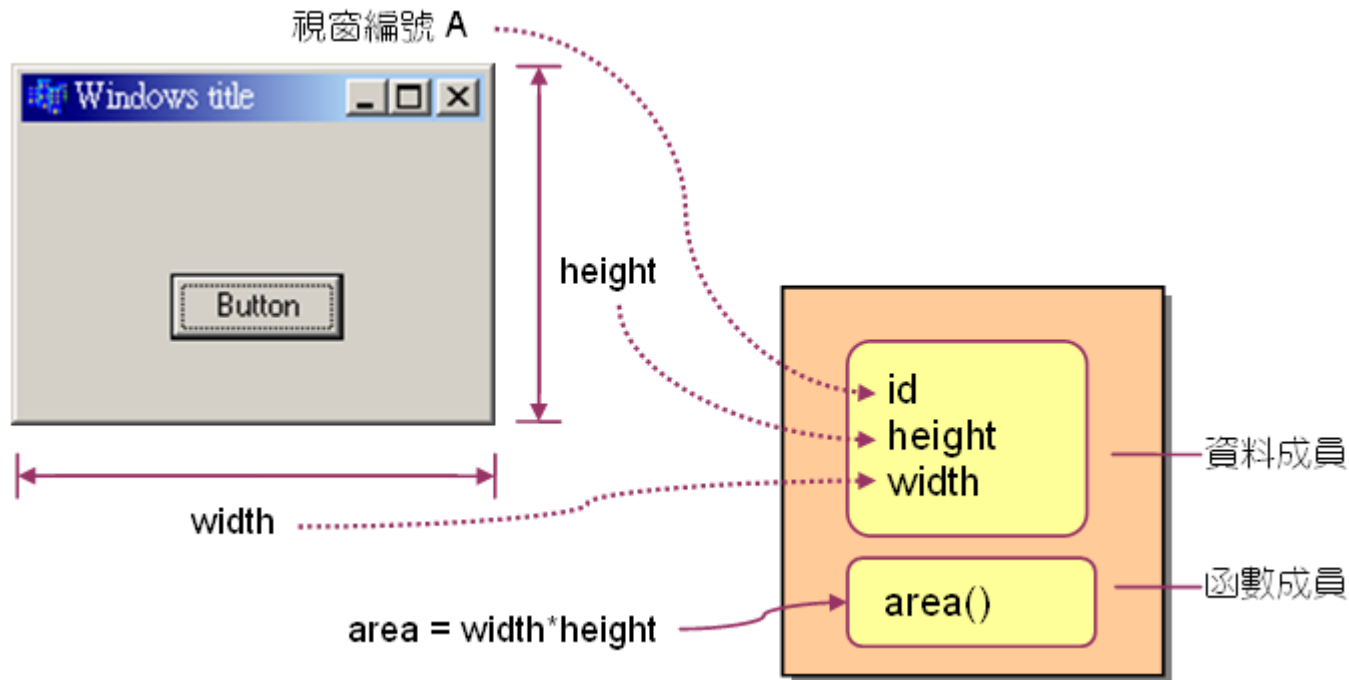
Basic Concept of Class

- A class includes ***data members*** and ***function members***
 - ***Data members (attributes)***
 - E.g., width, height
 - ***Function members (methods)***
 - E.g., area()





Encapsulation (封装)





Class Declaration

Class Declaration

```

class ClassName
{
    public:
        DataType VariableName;
        ...
        ReturnType FuncName(DataType1 Para1, DataType2 Para2,...)
        {
            Statement ;
            return (value);
        }
        ...
};

```

```

class CWin // Define Class CWin
{
    public: // Declare public members
        char id;
        int width; } Declare data members
        int height; }
        int area()
        {
            return width*height;
        }
};

```

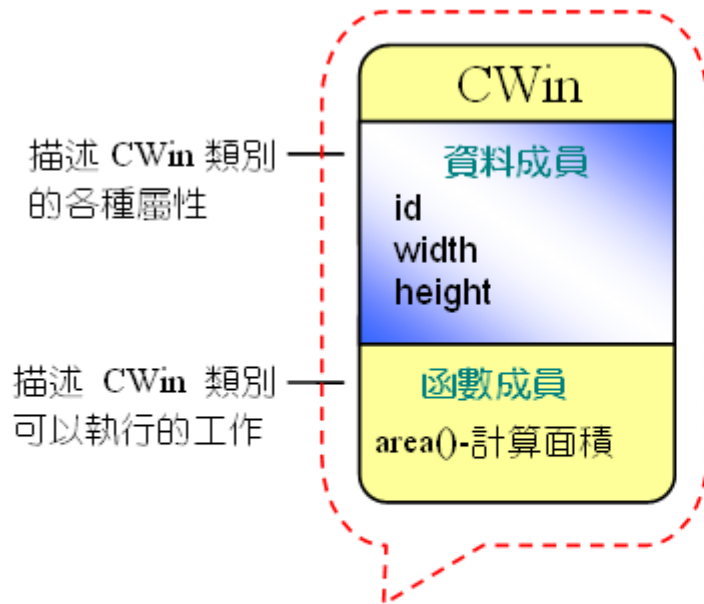
Function members

Public members can be accessed outside of the class.

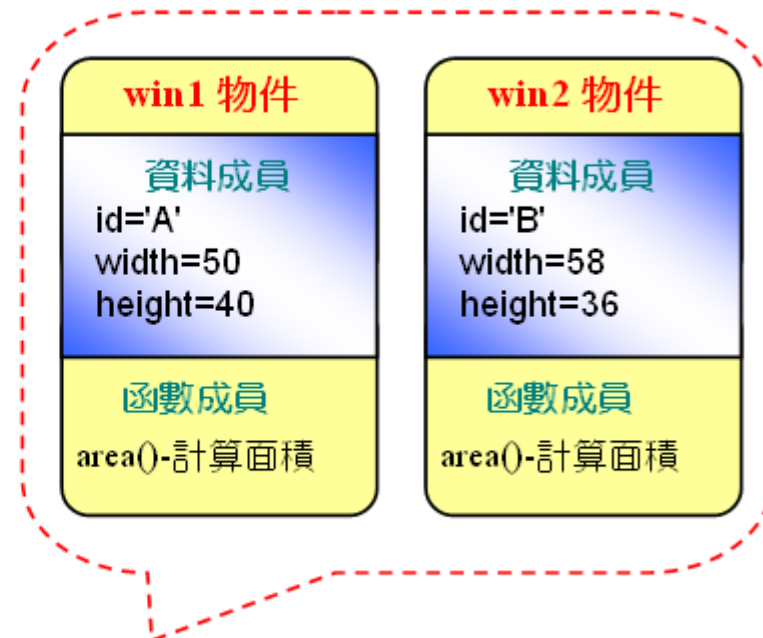


Object Creation

```
CWin win1,win2; // 宣告CWin類別型態的變數win1與win2
```



CWin 類別的定義



以 CWin 類別所產生的物件(實例)



Accessing Object Contents

- Access data members: **ObjectName.DataMember**

```
win1.id='A';           // 設定 win1物件的id成員為 A
win1.width=50;        // 設定 win1物件的寬為 50
win1.height=40;       // 設定 win1物件的高為 40
```

- Invoke function members: **ObjectName.FunctionMember**

```
win1.area();          // 利用 win1物件呼叫函數成員 area()
```



Class Example (1/2)

```
01 // 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 area(void) // 定義函數成員 area()，用來計算面積
13         {
14             return width*height;
15         }
16 };
17
```



Class Example (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) << "個位元組" << endl;
28
29     system("pause");
30     return 0;
31 }
```

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

```
Window A:
```

```
Area = 2000
```

```
sizeof(win1) = 12 個位元組
```

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



Function Calls Between Function Members (1/2)

```

01 // 函數成員的相互呼叫
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) // 定義函數成員 area()，用來計算面積
12         {
13             return width*height;
14         }
15         void show_area(void) // 定義函數成員 show_area()，用來顯示面積
16         {
17             cout<<"Window " << id <<"", area=" << area() << endl;
18         }
19 };

```

/* OUTPUT--

Window A, area=2000

-----*/

呼叫 area() 函數



Function Calls Between Function Members (2/2)

```
20 int main(void)
21 {
22     CWin win1;
23
24     win1.id='A';
25     win1.width=50;
26     win1.height=40;
27     win1.show_area(); // 呼叫 show() 函數
28
29     system("pause");
30     return 0;
31 }
```

```
/* OUTPUT--
```

```
Window A, area=2000
```

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




Passing Objects to Function Members (1/2)

```
01 // 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     int area(void)  // 定義函數成員 area()，用來計算面積
12     {
13         return width*height;
14     }
15     void set_data(char i,int w,int h) // 定義 set_data() 函數
16     {
17         id=i;          // 設定 id 成員
18         width=w;      // 設定 width 成員
19         height=h;     // 設定 height 成員
20     }
21 };
```



Passing Objects to Function Members (2/2)

```
22 void show_area(CWin win) // 把 show_area() 定義成一般的函數
23 {
24     cout<<"Window " <<win.id<<" , area=" <<win.area() << endl;
25 }
26
27 int main(void)
28 {
29     CWin win1;
30
31     win1.set_data('B',50,40); // 由 win1 物件呼叫 set_data() 函數
32     show_area(win1); // 傳遞 win1 物件到 show_area() 函數裡
33
34     system("pause");
35     return 0;
36 }
```

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



Member Function Overloading (1/3)

```
01 // 函數成員的多載
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(void) // 定義函數成員 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         }
```



Member Function Overloading (2/3)

```
20 void set_data(char i,int w,int h) // 第一個 set_data() 函數
21 {
22     id=i;           3 parameters
23     width=w;
24     height=h;
25 }
26 void set_data(char i) // 第二個 set_data() 函數
27 {
28     id=i;           1 parameter
29 }
30 void set_data(int w,int h) // 第三個 set_data() 函數
31 {
32     width=w;       2 parameters
33     height=h;
34 }
35 };
36
```



Member Function Overloading (3/3)

```
37 int main(void)
38 {
39     CWin win1,win2;
40
41     win1.set_data('A',50,40);           // 呼叫有三個引數的 set_data()
42     win2.set_data('B');                // 呼叫有一個引數的 set_data()
43     win2.set_data(80,120);             // 呼叫有兩個引數的 set_data()
44
45     win1.show_area();                  // 利用 win1 物件呼叫 show_area()
46     win2.show_area();                  // 利用 win2 物件呼叫 show_area()
47
48     system("pause");
49     return 0;
50 }
```

```
/* OUTPUT--
```

```
Window A, area=2000
```

```
Window B, area=9600
```

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



Problems of Public Data Members (1/2)

```
01 // 由類別外部直接設定資料成員所產生的錯誤
02 #include <iostream>
03 #include <cstdlib>
04 using namespace std;
05 class CWin // 定義視窗類別 CWin
06 {
07     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     };
```

CWin 類別內部



Problems of Public Data Members (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

/* OUTPUT---

Window A, area=-2000

-----*/



Private Members

- Private members can only be accessed inside of the class.

Public and private members

```
class ClassName
{
    private:
        // define private members
    public:
        // define public members
}
```

Without keyword **public**, members are private by default.

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




Private Member Example (1/2)

```

01 // 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) // 函數成員 area()
14         {
15             return width*height;
16         }
17         void show_area(void) // 函數成員 show_area()
18         {
19             cout<<"Window " << id <<"", area=" << area() << endl;
20         }
21 };
22

```

外部

在 CWin 類別內部，故可存取私有成員

在 CWin 類別內部，故可存取私有成員



Private Member Example (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 }

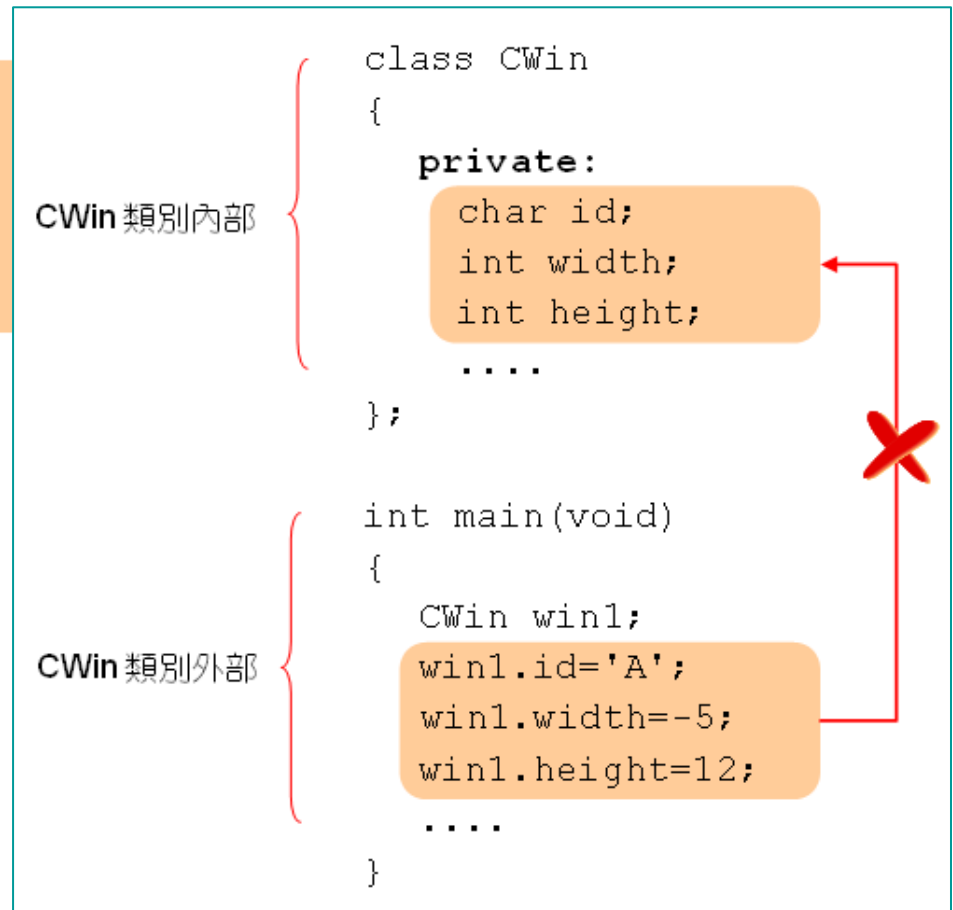
```

```

win1.id='A';
win1.width=-5;
win1.height=12;

```

錯誤，在 CWin 類別外部無法直接更改私有成員





Accessing Private Data Members through Public Function Members (1/3)

```
01 // 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;
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) // 公有函數成員 area()
14         {
15             return width*height;
16         }
17         void show_area(void) // 公有函數成員 show_area()
18         {
19             cout<<"Window " << id <<"", area=" << area() << endl;
20         }
}
```



Accessing Private Data Members through Public Function Members (2/3)

Using public function members to check error.

```
21 void set_data(char i,int w,int h) // 公有函數成員 set_data()
22 {
23     id=i;
24     if(w>0 && h>0) // 如果 w 與 h 均大於 0
25     {
26         width=w;
27         height=h;
28     }
29     else // 如果 w 與 h 任一個小於 0
30     {
31         width=0;
32         height=0;
33         cout << "input error" << endl;
34     }
35 }
```



Accessing Private Data Members through Public Function Members (3/3)

```

37 int main(void)
38 {
39     CWin win1;
40
41     win1.set_data('A', 50, 40);
42     win1.show_area(); // 顯示面積
43     system("pause");
44     return 0;
45 }

```

CWin 類別內部

```

class CWin
{
    private:
        ...
    Public:
        ...
    void set_data(char i,int w,int h)
    { .... }
};

```

CWin 類別外部

```

int main(void)
{
    CWin win1;
    win1.set_data('A', 50, 40);
    ....
}

```

類別內部的公有成員，可直接由類別外部來存取



Class Constructor

- The **constructor** is used to initialize objects.

Constructor's name should be the same as the class's name

Constructor's format

```

ClassName(DataType1 Para1, DataType2 Para2, ...)
{
    statement ;
    ...
}

```

建構元沒有傳回值

No return data

```

CWin(char i,int w,int h) // CWin()建構元，可接收三個引數
{
    id=i;
    width=w;
    height=h;
}

```



Constructor Example (1/2)

```

01 // 建構元的使用
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         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;
16             height=h;
17             cout << "CWin 建構元被呼叫了..." << endl;
18         }

```

```

class CWin
{
    ...
    CWin(char i,int w,int h)
    {
        id=i;
        width=w;
        height=h;
        ...;
    }
}
int main(void)
{
    CWin win1('A',50,40);
    CWin win2('B',60,70);
    ...
}

```



Constructor Example (2/2)

```
19     void show_member(void) // 函數成員，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     system("pause");
34     return 0;
35 }
```

/* 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

-----*/



Constructor Overloading (1/3)

```
01 // 建構元的多載
02 #include <iostream>
03 #include <cstdlib>
04 using namespace std;
05 class CWin // 定義視窗類別 CWin
06 {
07     private:
08         char id;
09         int width, height;
10
```

沒有引數的建構元

有三個引數的建構元

```
CWin(char i,int w,int h)
{
    id=i;
    width=w;
    height=h;
    ....
}

CWin(void)
{
    id='Z';
    width=10;
    height=10;
    ....
}
```



Constructor Overloading (2/3)

```
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     }
19     CWin(void)                // 沒有引數的建構元
20     {
21         id='Z';
22         width=10;
23         height=10;
24         cout <<"CWin() 建構元被呼叫了..."<<endl;
25     }
26     void show_member(void)    // 函數成員，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
```



Constructor Overloading (3/3)

```
33 int main(void)
34 {
35     CWin win1('A',50,40); // 建立 win1 物件，並呼叫有三個引數的建構元
36     CWin win2;           // 建立 win2 物件，並呼叫沒有引數的建構元
37
38     win1.show_member();
39     win2.show_member();
40
41     system("pause");
42     return 0;
43 }
```

```
/* OUTPUT-----
CWin(char,int,int) 建構元被呼叫了...
CWin() 建構元被呼叫了...
Window A: width=50, height=40
Window Z: width=10, height=10
-----*/
```