



# 05 Inheritance (II)

Fundamental Computer Programming- C++ Lab (II)

元智大學 | C++ 程式設計實習 (II)

張維元

課程投影片：[請從元智個人 Portal 下載](#)

2015



維元

(@v123582)

Web Development

Data Science

#遠端 #斜槓 #教學  
#資料科學 #網站開發

擅長網站開發與資料科學的雙棲工程師，熟悉的語言是 Python 跟 JavaScript。同時也是程式社群 JSDC 核心成員及資料科學家的 12 堂心法養成計畫發起人，擁有多次國內大型技術會議講者經驗，持續在不同的平台發表對 #資料科學、 #網頁開發 或 #軟體職涯 相關的分享。

- 元智大學 C++/CPE 程式設計課程 講師
- ALPHACamp 全端 Web 開發 / Leetcode Camp 課程講師
- CUPOY Python 網路爬蟲實戰研習馬拉松 發起人
- 中華電信學院 資料驅動系列課程 講師
- 工研院 Python AI 人工智慧資料分析師養成班 講師
- 華岡興業基金會 AI/Big Data 技能養成班系列課程 講師

site: v123582.tw / line: weiwei63

mail: weiyuan@saturn.yzu.edu.tw

2015



維元

(@v123582)

Web Development

Data Science

#遠端 #斜槓 #教學  
#資料科學 #網站開發

擅長網站開發與資料科學的雙棲工程師，熟悉的語言是 Python 跟 JavaScript。同時也是程式社群 JSDC 核心成員及資料科學家的 12 堂心法養成計畫發起人，擁有多次國內大型技術會議講者經驗，持續在不同的平台發表對 #資料科學、 #網頁開發 或 #軟體職涯 相關的分享。

- 2018 總統盃黑客松 冠軍隊伍
- 2017 資料科學愛好者年會(DSCONF) 講師
- 2017 行動科技年會(MOPCON) 講師
- 2016 微軟 Imagine Cup 台灣區冠軍

site: v123582.tw / line: weiwei63  
mail: weiyuan@saturn.yzu.edu.tw

# 什麼是程式？

程式語言是用來命令電腦執行各種作業的工具，是人與電腦溝通的橋樑。當電腦藉由輸入設備把程式讀入後，會儲存在主記憶體內，然後指令會依序被控制單元提取並解碼或翻譯成電腦可以執行的信號，並把信號送到各個裝置上，以執行指令所指派的動作。也就是說，人類與電腦溝通的語言稱為程式語言。

程式 = 利用一系列的指令告訴電腦如何執行工作



# 物件導向程式設計

物件導向程式設計（Object-oriented programming，OOP）是種具有物件概念的程式設計典範，同時也是一種程式開發的抽象方針。它可能包含資料、屬性、程式碼與方法。物件則指的是類別（class）的實例。它將物件作為程式的基本單元，將程式和資料封裝其中，以提高軟體的重用性、靈活性和擴充性。

把物件作為程式最小的單位，模擬實體世界的運作



# 類別與物件

- 類別 (Class) 利用屬性與方法定義了物件的抽象樣板/結構
- 物件 (Object) 是類別的實體 (instance) ，可以使用類別中的方法

```
1 class Book {  
2     public:  
3         string title;  
4         int price;  
5 };  
6
```

```
int main( ){  
  
    Book b1, b2;  
  
    b1.title = "C++ How to Program(1)";  
    b2.title = "C++ How to Program(2)";  
  
    cout << b1.title << endl;  
    cout << b2.title << endl;  
    return 0;  
}
```

# 成員變數與成員函式

(member variable and member function)

```
1  class Book {  
2      public:  
3          string title; ← 成員變數  
4          int price; ← 成員變數  
5          void f(void); ← 成員函式  
6  };  
7  
8  void Book::f(void){  
9      cout << "I'm a book: " + title;  
10     };  
11     ← 成員函式可存取成員變數
```

```
1  int main( ){  
2      Book book1;  
3      book1.title = "C++ How to Program";  
4      book1.f();  
5  
6      return 0;  
7  }
```



# 利用公開函式存取私有成員

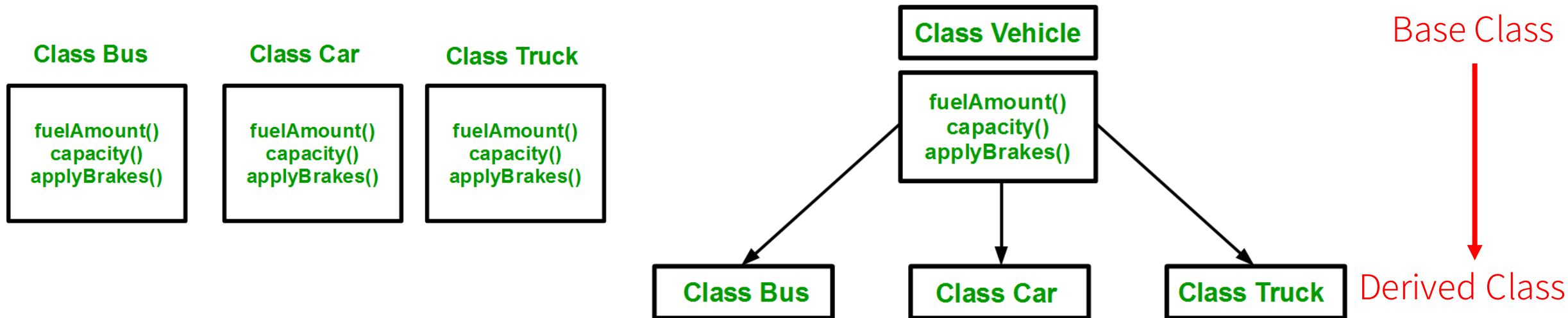
```
1 class Book {  
2     private: ← 私有成員  
3     int price;  
4     public: ← 公開函式  
5     void set(int p);  
6     double get(void);  
};  
  
void Book::set(int p){  
    price = p;  
};  
  
double Book::get(void){  
    return price * 0.9;  
};
```

```
1 int main( ){  
2     Book book1;  
3     book1.set(600);  
4     cout << "price: " << book1.get();  
5  
6     return 0;  
}
```



# 繼承 (Inheritance)

繼承是物件導向的一個概念，繼承可以使得子類別具有父類別的各種屬性和方法，而不需要再次編寫相同的代碼。子類別也可以重新定義某些屬性，並重寫某些方法。



# 繼承

## (Inheritance)

繼承是物件導向的一個概念，繼承可以使得子類別具有父類別的各種屬性和方法，而不需要再次編寫相同的代碼。子類別也可以重新定義某些屬性，並重寫某些方法。

```
1 class Shape {  
2     protected:  
3         int width, height;  
4     public:  
5         void setWidth(int w){  
6             width = w;  
7         }  
8         void setHeight(int h){  
9             height = h;  
10        }  
11    };
```

```
1  
2  
3 class Rectangle: public Shape {  
4     public:  
5         int getArea(){  
6             return (width * height);  
7         }  
8 };  
9
```

# 繼承

## (Inheritance)

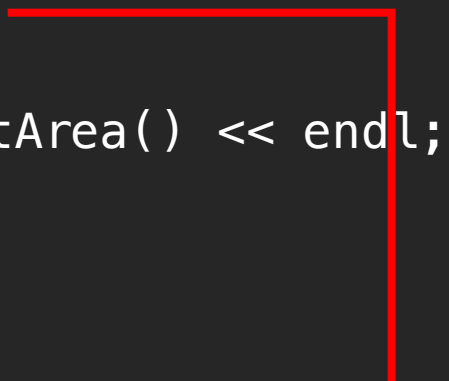
繼承是物件導向的一個概念，繼承可以使得子類別具有父類別的各種屬性和方法，而不需要再次編寫相同的代碼。子類別也可以重新定義某些屬性，並重寫某些方法。

```
1 int main(void){
2     Rectangle Rect;
3
4     Rect.setWidth(5);
5     Rect.setHeight(7);
6     cout << Rect.getArea() << endl;
7
8     return 0;
9 }
```

```
1
2
3 class Rectangle: public Shape {
4     public:
5         int getArea(){
6             return (width * height);
7         }
8 };
9
```

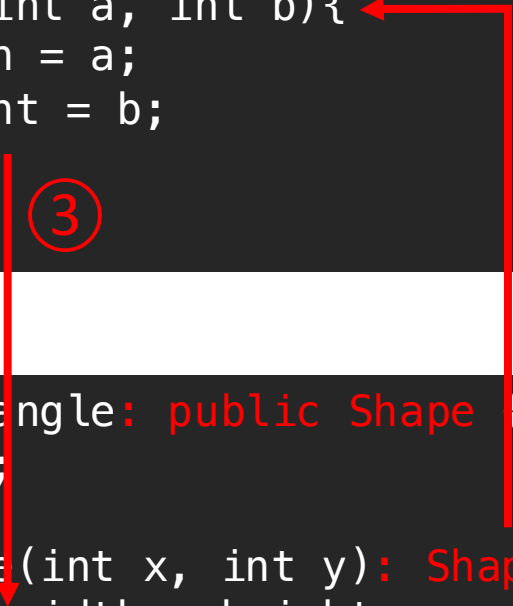
# 繼承的建構式

```
1
2 int main(void){
3     Rectangle Rect;
4
5     cout << Rect.getArea() << endl;
6
7     return 0;
8 }
9
```



A red line connects the `Rect.getArea()` call in the `main` function to the `getArea` method in the `Rectangle` class.

```
1 class Shape {
2     protected:
3         int width, height;
4     public:
5         Shape(int a, int b){
6             width = a;
7             height = b;
8         }
9 };
```



A red line connects the `Shape(x, y)` call in the `Rectangle` constructor to the `Shape` class definition.

```
1 class Rectangle: public Shape {
2     int area;
3     public:
4     Rectangle(int x, int y): Shape(x, y){
5         area = width * height;
6     }
7     int getArea(){
8         return area;
9     }
10 };
```

# 複製建構式

```
1  int main(void){  
2      Shape s1(3, 4);  
3      cout << s1.getArea() << endl;  
4  
5      Shape s2(s1);  
6      cout << s2.getArea() << endl;  
7  
8      Shape s3 = s1;  
9      cout << s3.getArea() << endl;  
  
    return 0;  
}
```

```
1  class Shape {  
2      public:  
3          int width, height;  
4          Shape(int a, int b);  
5          ~Shape();  
6          int getArea(){  
7              return width * height;  
8          }  
9  };
```

```
1  Shape::Shape(int a, int b){  
2      cout << "Shape constructor()" << endl;  
3      width = a;  
4      height = b;  
5  }  
6  
7  Shape::~~Shape(){  
8      cout << "Shape Dstructor()" << endl;  
9  }  
10
```

→ 複製建構子預設會把傳入的物件複製給新的物件

# 複製建構式

```
1  int main(void){
2      Shape s1(3, 4), s2(3, 5);
3
4      Shape s3 = s1;
5      cout << s3.getArea() << endl;
6
7      s3 = s2;
8      cout << s3.getArea() << endl;
9
10     return 0;
11 }
```

```
1  class Shape {
2      public:
3          int width, height;
4          Shape(int a, int b);
5          Shape &operator=(const Shape &obj){
6              cout << "operator=()" << endl;
7              // width = obj.width;
8              // height = obj.height;
9              return *this;
10     };
11 }
```

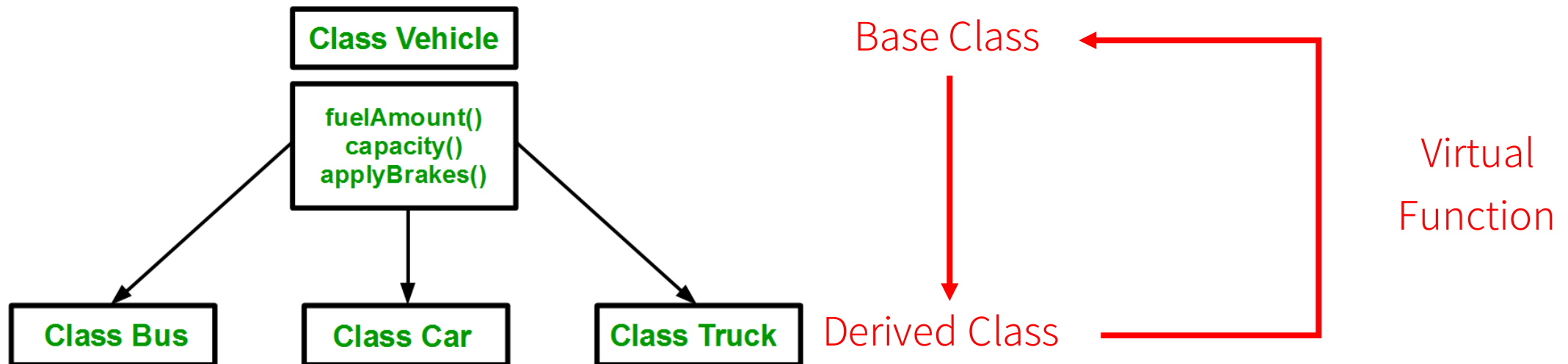
```
1  Shape::Shape(int a, int b){
2      cout << "Shape constructor()" << endl;
3      width = a;
4      height = b;
5  }
6
7  Shape::Shape(const Shape& obj){
8      // width = obj.width;
9      // height = obj.height;
10     cout << "Copy Constructor()" << endl;
11 }
```

→ 宣告的 = 會呼叫複製建構子，之後的 = 會呼叫 Operator

# 虛擬函式

## (Virtual Function)

虛擬函式是一種宣告在父類別類別的成員函式，執行時才會由子類別類別覆蓋後執行。這一概念是物件導向中（執行時）多型的實現，虛擬函式可以給出函式的定義，但該目標的具體指向在編譯期可能無法確定。





```

1 int main( ){
2
3     Rectangle rec(10,7);
4     rec.area();
5
6     return 0;
7 }
8
9

```

```

1 class Shape {
2     protected:
3         int width, height;
4     public:
5         Shape( int a=0, int b=0){
6             width = a;
7             height = b;
8         }
9         void area(){
10             cout << "Parent class area :" <<endl;
11         }
12 };

```

```

1 class Rectangle: public Shape{
2     public:
3         Rectangle(int a, int b):Shape(a, b){ }
4         void area (){
5             cout << "Rectangle class area :";
6             cout << (width * height) << endl;
7         }
8     };
9
10

```

→ 子類別可以沿用父類別的並且覆蓋成新的定義

```

1 int main( ){
2     Rectangle rec(10,7);
3
4     Shape s1 = rec;
5     s1.area();
6
7     return 0;
8 }
9

```

```

1 class Shape {
2     protected:
3         int width, height;
4     public:
5         Shape( int a=0, int b=0){
6             width = a;
7             height = b;
8         }
9         void area(){
10            cout << "Parent class area :" <<endl;
11        }
12    };

```

```

1 class Rectangle: public Shape{
2     public:
3         Rectangle(int a, int b):Shape(a, b){ }
4         void area (){
5             cout << "Rectangle class area :";
6             cout << (width * height) << endl;
7         }
8     };
9
10

```

→ 父類別無法直接存取子類別新的定義

```

1 int main( ){
2     Rectangle rec(10,7);
3
4     Shape *s1 = &rec;
5     s1->area();
6
7     return 0;
8 }
9

```

```

1 class Shape {
2     protected:
3         int width, height;
4     public:
5         Shape( int a=0, int b=0){
6             width = a;
7             height = b;
8         }
9         virtual void area(){
10
11     };
12

```

```

1 class Rectangle: public Shape{
2     public:
3         Rectangle(int a, int b):Shape(a, b){ }
4         void area (){
5             cout << "Rectangle class area :";
6             cout << (width * height) << endl;
7         }
8     };
9
10

```

→ 父類別可定義虛擬函式用來存取子類別中的定義

# 各種類別成員的修飾子

- public：所有人都可以存取的成員
- private：只有物件本身可以存取的成員
- protected：子類別可以存取父類別中的成員
- static：同一類別所有物件共用的成員
- friend：開放給其他類別的物件存取的成員
- virtual：虛擬函式可讓父類別用來存取子類別中的函式定義

# Thanks for listening.

元智大學 | C++ 程式設計實習

Wei-Yuan Chang