

Lab of Object-Oriented Programming:

Advanced Inheritance

黃威、陳岳紘、邱彥翔

2022

使用 moodle 點名

請登入實習課的 moodle 課程


點擊出缺席並完成今日的點名

- 邱彥翔 - 108703017@nccu.edu.tw

E-mail 格式

- 標題：[OOP111] + 問題
- 必須包含系級學號姓名
- 請附上有問題的**部分**程式碼或截圖

 討論區

 出缺席

多型 (Polymorphism)

多型 (Polymorphism)

- 一個訊息 (message, event) 的意義是由接收者來定義
 - 同樣的訊息, 會根據不同的接收者而有不同的行為
- 實現多型的方法
 - 在子類別重新定義函式
 - Overriding, Overloading
 - 使用虛擬的成員函式
 - Virtual function

多型 (Polymorphism)

- 多型的種類
 - Compile-time / static binding
 - Method overriding
 - Operator overloading
 - Runtime / dynamic binding
 - Virtual functions

Static Binding vs. Dynamic Binding

- Static Binding

- 由編譯器決定要呼叫哪個成員函式
- 在程式開始執行前就決定呼叫的對象
- 使用 overloading 實作

- Dynamic Binding

- 在程式執行時才決定要呼叫哪個成員函式
- 使用 virtual function 實作

Virtual Function

- 沒有 virtual 宣告的情況
 - 編譯時不會決定呼叫的函式
 - 不會判斷指向的物件的形態

```
#include <iostream>
using namespace std;

class Parent {
public:
    Parent(){};
    void print() {
        cout << "Parent is called." << endl;
    }
};

class Child : public Parent {
public:
    Child(){};
    void print() {
        cout << "Child is called." << endl;
    }
};

int main() {
    Parent parent;
    Child child;
    Parent* p = &parent;
    Parent* c = &child;
    p->print();
    c->print();

    return 0;
}
```

Virtual Function

- 有 virtual 宣告的情況
 - 編譯時會依據指標變數的形態決定呼叫的函式
 - 執行時依據物件形態決定

```
#include <iostream>
using namespace std;

class Parent {
public:
    Parent(){};
    virtual void print() {
        cout << "Parent is called." << endl;
    }
};

class Child : public Parent {
public:
    Child(){};
    virtual void print() {
        cout << "Child is called." << endl;
    }
};

int main() {
    Parent parent;
    Child child;
    Parent* p = &parent;
    Parent* c = &child;
    p->print();
    c->print();

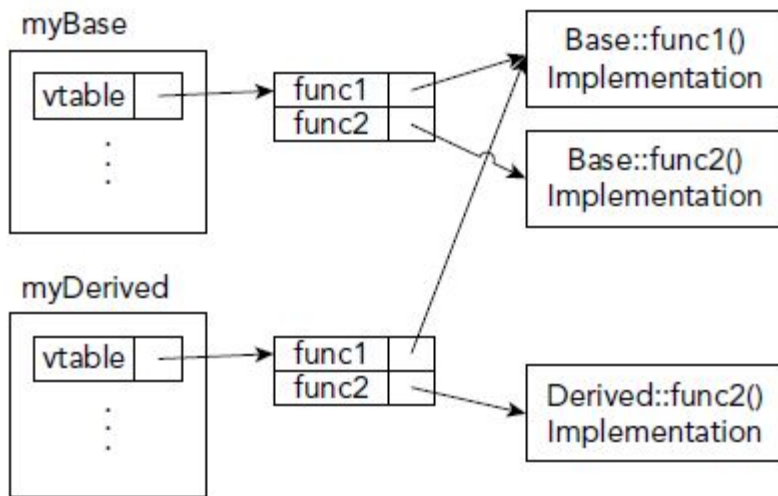
    return 0;
}
```


Virtual Table & Virtual Pointer

- C++ 使用 virtual table 和 virtual pointer 實作 virtual function
 - 當類別中有一個以上的函式被宣告 virtual 時自動產生
 - Virtual table: 存放類別中所有的虛擬函式之位址
 - Virtual pointer: 隱藏的成員變數, 指向 virtual table

Virtual Table & Virtual Pointer

- 父類別生成時會產生 virtual table
 - 如果子類別有重新定義虛擬函式，就會更新 virtual table
- Virtual table 在建構子產生
 - Virtual pointer 在 table 產生時指向 table
 - 不能將建構子定義成虛擬函式
 - 避免在 constructor 中呼叫虛擬函式



抽象類別 (Abstract Class)

抽象類別

- 物件爲抽象概念時使用
 - 例如「水果」物件，不存在一種水果叫做「水果」
- 使用純虛擬函式 (pure virtual function) 實作

抽象類別

- 純虛擬函式
 - 未實作的函式
- 抽象類別
 - 含有純虛擬函式
 - 只能被繼承，不能生成物件
 - 子物件必須實作虛擬函式

```
#include <iostream>
using namespace std;

class Base {
public:
    virtual void show() = 0;
};

class Derived : public Base {
public:
    void show() { cout << "In derived." << endl; }
};

int main() {
    Base *bp = new Derived();
    bp->show();
    return 0;
}
```

Private Inheritance

Private Inheritance

- 子類別用 private 的方式繼承父類別
 - 將父類別的 public、protected 以 private 方式繼承
 - 外部無法透過子類別存取父類別的函式
 - 子類別能夠以 public、protected 覆寫函式

```
class Derived : private Base {  
    Derived(){};  
};
```

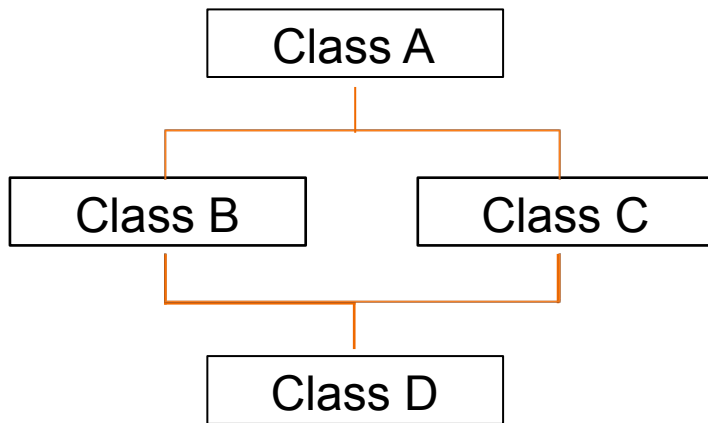
Protected Inheritance

- 子類別用 protected 的方式繼承類別
 - 將父類別的 public 以 protected 的方式繼承

多重繼承 (Multiple Inheritance)

多重繼承

- 繼承多個父類別
- 可能會有 ambiguous access 的問題
 - 有很多路徑會呼叫到同一個函式



Ambiguous Access

- 使用 virtual 函式繼承
 - 同樣名稱的函式只會有一個基底類別

```
class B : virtual public A {  
    public:  
    int b;  
};  
class C : virtual public A {  
    public:  
    int c;  
};  
class D : public B, public C {  
    public:  
    int b;  
};
```

Assign 6

Assign6

項目	配分
有交	20
可編譯	15
實作草的行爲	15
實作羊的行爲	15
可以順利完成模擬	15
實作 display	5
實作 passes	5
實作 seed	10

Exercise 7

題目說明

人員種類	薪資計算方式
Salaried employee	(Weekly salary)
Commision employee	(Gross sales * Commission rate)
Base-salaried commission employee	(Gross sales * Commission rate) + Base salary