C/C++基礎程式設計

C++: 物件導向程式設計-使用類別(Class)

張傑帆
CSIE, NTU

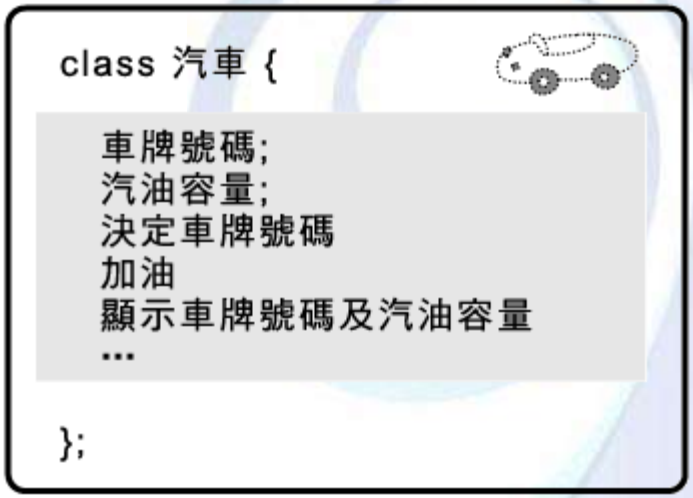
決不、決不、決不、決不放棄！
Never, never, never, never give up. -Steve Jobs

課程大綱

- **類別 (Class)**
  - 物件導向
  - 類別, 物件, 成員
  - 權限
  - 建構式與解構式
- 作業

C++物件導向

- 以物件為基礎的程式設計，將程式中互動的單元視為一個個的物件。
- 封裝（Encapsulation）
  - 封裝物件資訊是第一步，您要瞭解如何使用類別定義物件的屬性、方法(行為)
  - 類別是建構物件時所依賴的規格書。
  - 例如設計一個物件: 人
    - 屬性：姓名, 身高, 體重
    - 行為：輸入資料, 輸出資料

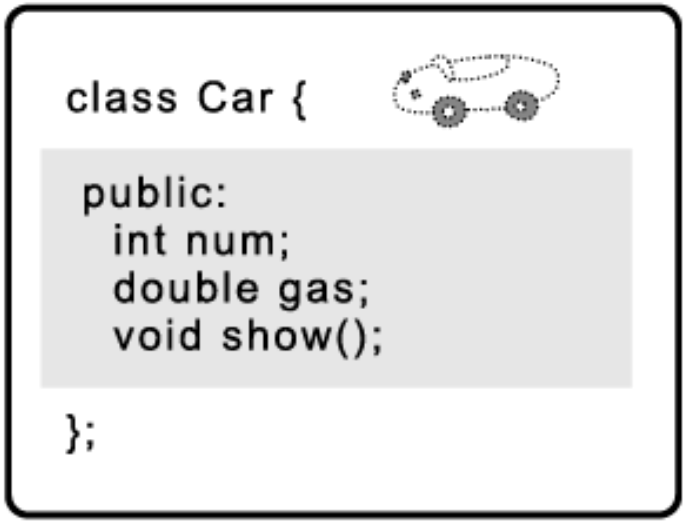


類別（Class）

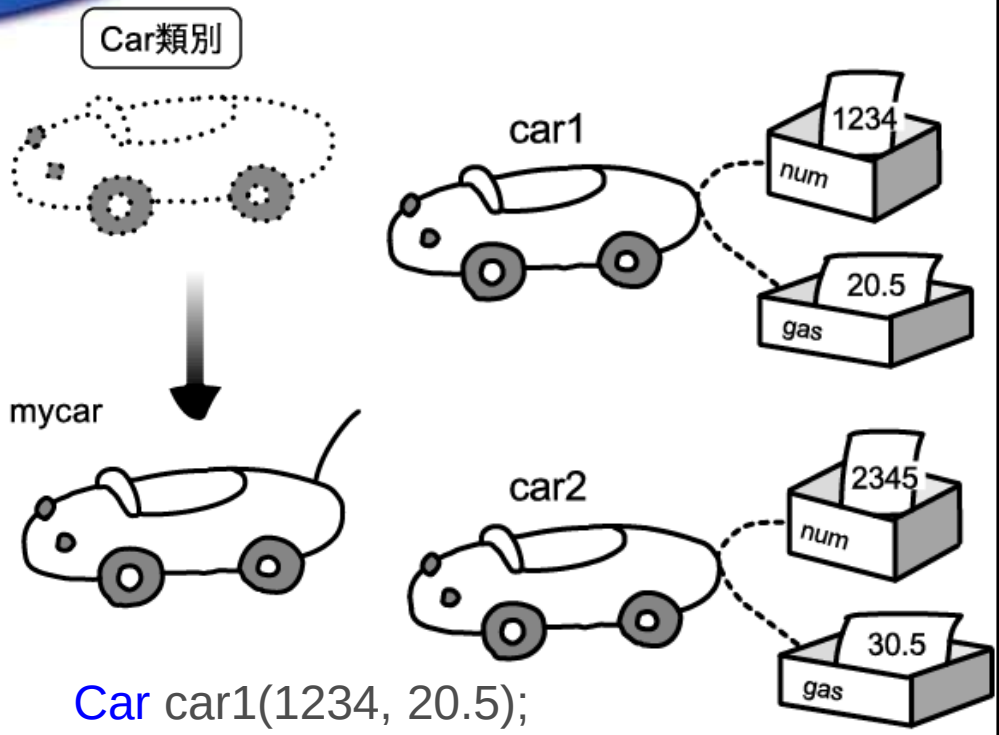
- **類別class是C++中用來封裝資料的關鍵字**
- **當使用類別來定義一個物件時，考慮這個物件可能擁有的「屬性」與「方法」成員**
  - 屬性是物件的靜態描述
  - 方法是可施加於物件上的動態操作
- **使用類別定義出這個物件的規格書，之後就可依這個規格書製作出一個個的物件實例，並在製作過程中設定個別物件的專屬特性資料。**
- **要訣：** Example1 Example2
  - 屬性→宣告要存放的資料 (每個物件有自己的屬性)
  - 方法→寫要執行的函式　(通常用來操作物件的屬性)

類別（Class）

- 宣告一個類別
  (類似定義一個結構struct)
- 語法：
  - class 類別名稱
    {
    public:
    類別名稱(); //建構式, 用來做物件的初始化
    ~類別名稱(); //解構式, 用來做物件的善後工作
    公開的方法或屬性;
    protected: // 只有在同一繼承架構中可以使用的資料
    受保護的方法或屬性;
    private: // 只有在此類別中可以使用的資料
    私有的方法或屬性;
    };



物件的產生與使用



- 使用類別宣告物件
  (類似宣告一個變數)
- 語法：
  Car mycar;
  Car car1,car2;
  - 類別名稱 物件名稱;
  - 類別名稱 物件名稱(參數1, 參數2, … , 參數n);

Car car1(1234, 20.5);

- 物件可透過 . 來使用或存取該方法或屬性
  (類似C語言的結構struct)

```
car1.num = 10;
car1.show();
```

- 若為物件指標，可透過 -> 來使用或存取該方法或屬性

從一個簡單的例子開始

- 範例: 輸入兩個人資料(姓名, 身高, 體重)並印出

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

class Person
{
   public:
      void input()
      {
         cin >> name;
         cin >> height;
         cin >> weight;
      }
      void output()
      {
         cout << "Name:" << name << endl;
         cout << "Height:" << height << " cm" << endl;
         cout << "Weight:" << weight << " kg" << endl;
      }
      string name;
      int  height;
      int  weight;
};
```

```
int main()
{
   Person p1;
   Person p2;

   p1. input();
   p1. output();
   p2. input();
   p2. output();
   return 0;
}
```

示範I example

- **定義一 Pokemon 類別如下：**
  - 屬性：
    - string Name: 名字
    - int Lv: 等級
    - int HpMax: 最大血量
    - int HpCurrent: 現存血量
  - 方法：
    - void ShowInfo(): 顯示資訊
- **使用Pokemon 類別宣告兩個物件並印出它們的資訊**

https://jgirl.ddns.net/problem/0/1107

類別的方法之描述

- 實作一個類別方法的內容*(類似寫一個函式)*
- 除了寫在類別定義中, 也可拿到**類別定義以外**的地方描述
- 語法：
  - 資料型態 類別名稱::方法名稱(參數1, 參數2, … ,參數n)
    {
    程式碼;
    }

範疇解析運算子
(scope resolution operator)

```
void Person::output(){
        cout << "Name:"<<name<<endl;
        cout << "Height:"<<height<<endl;
        cout << "Weight:"<<weight<<endl;
}
```

https://goo.gl/FLdXo8

示範II

- 定義一 Pokemon 類別如
- 使兩寶可夢物件可對戰
  - 屬性：
    - …同上
  - 方法：
    - void ShowInfo(): 顯示資訊
    - void Attack(Pokemon &Target): 攻擊
    - void Defence(int atkp): 防御(被攻擊？)
    - void Cure(): 治癒

example

https://jgirl.ddns.net/problem/0/1108

練習

- **設計一個Square(正方型)類別包含下列成員：**
  - 屬性：
    - int len: 邊長
  - 方法：
    - int area(): 計算面積
- **使用Square類別宣告兩個邊長為10與20的物件,並印出它們的面積**

https://jgirl.ddns.net/problem/0/1112

如何設計類別?

- **思考 (以功能角度)**
  - 每個物件需要什麼資料?
  - 每個物件需要什麼方法來操作資料?
- **進階思考 (以使用者角度)**
  - 如何讓使用類別的人方便簡單使用
  - 如何避免使用類別的人因資料操作不當而產生錯誤

存取成員的限制

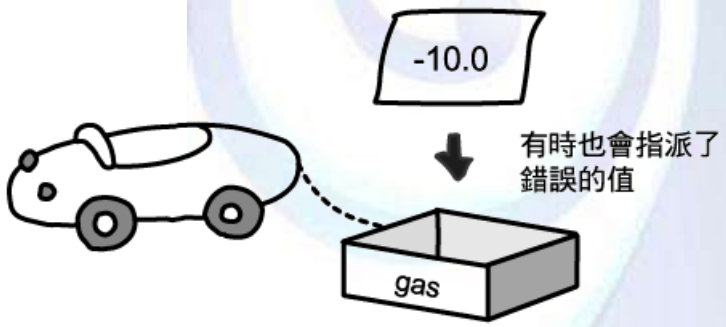
- 在設定資料成員的值的時候，有時候可能會出現一些問題，所以才有一些**防止錯誤的機制**。
- 例如：

```
int main()
{
   Car car1;

   car1.num = 1234;
   car1.gas = -10.0    ←—— 指派了錯誤的汽油容量

   car1.show();
}
```

- 但汽油容量不可能是負值。

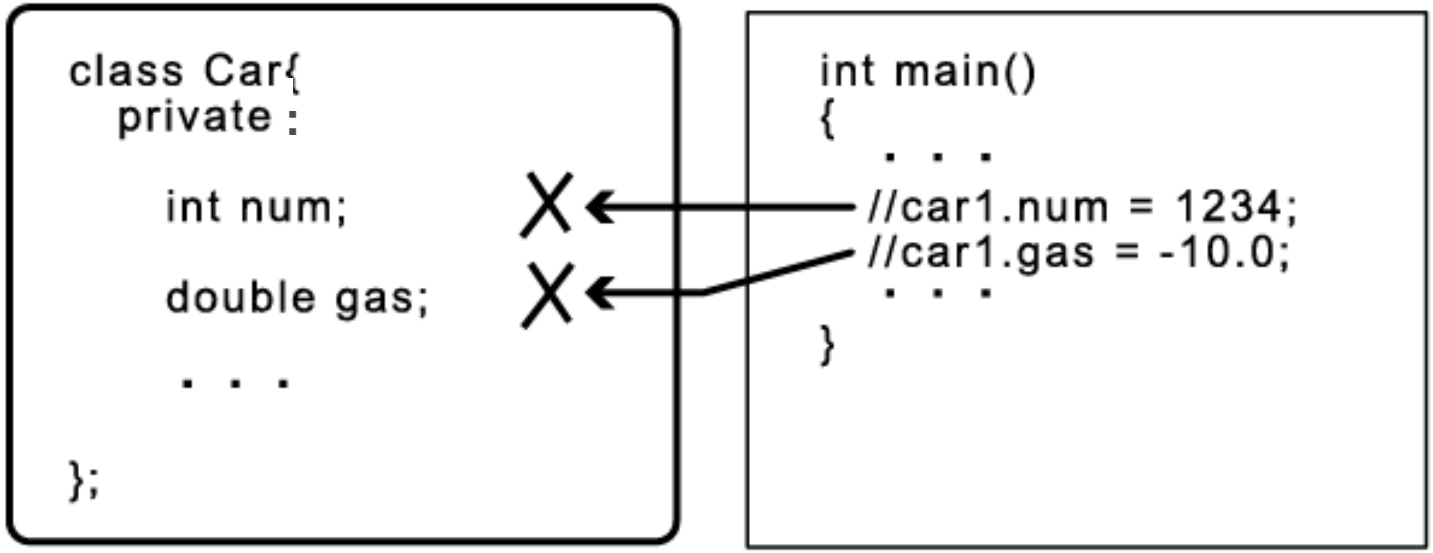


資料的權限

- 最重要的是別忘了在最後加上分號，初學C++的新手很常犯這個錯誤
- **public**這個關鍵字，它表示以下所定義的成員可以使用物件名稱直接被呼叫，稱之為「公開成員」
- **private**關鍵字下的則是「私有成員」，不可以透過物件名稱直接呼叫。
- 在類別封裝時，有一個基本原則是：**資訊的最小化公開**。如果屬性**可以不公開就不公開**，如果要取得或設定物件的某些屬性，也是儘量透過方法成員來進行。

製作Private成員

- 將成員設定成無法從類別外任意存取
- 這種成員稱之為**private成員**
- 設為**private**成員之後，就不可以從類別外任意存取了。



把資料成員設成**private**成員　　無法從類別外存取**private**成員

資料的權限

- 思考: 由上一個輸入兩個人資料(姓名, 身高, 體重)並印出的範例, 你希望Person產生的物件只能用input與output函式來輸入輸出資料，該如何達到此功能？
  - 使用private成員！

資料的權限

- 範例:

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

class Person
{
   public:
      void input()
      {
         cin >> name;
         cin >> height;
         cin >> weight;
      }
      void output()
      {
         cout << "Name:" << name << endl;
         cout << "Height:" << height << " cm" << endl;
         cout << "Weight:" << weight << " kg" << endl;
      }
   private:
      string name;
      int  height;
      int  weight;
};
```

```
int main()
{
   Person p1;
   Person p2;

   p1. input();
   //p1. height = 0;
   //這行不能執行!
   p1. output();
   p2. input();
   p2. output();
   return 0;
}
```

資料的權限 fix 修正版

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

class Person
{
   public:
      void setHW(int h, int w, string na);
      void input()
      {
         cin >> name;
         cin >> height;
         cin >> weight;
      }
      void output()
      {
         cout << "Name:" << name << endl;
         cout << "Height:" << height << " cm" << endl;
         cout << "Weight:" << weight << " kg" << endl;
      }
   private:
      string name;
      int  height;
      int  weight;
};
```

```
void Person::setHW(int  h,
int  w, string na){
   if(h<0)
      h = 1;//.....
   name = na;
   height = h;
   weight = w;
}
int main()
{
   Person p1;
   Person p2;
   string name;
   int h,w;
   cin >> name >> h >> w;

   p1.setHW(h, w, name);
   //p1. input();
   //p1. height = 0;
   //這行不能執行!
   p1. output();
   p2. input();
   p2. output();
   return 0;
}
```

示範III

- 定義一 Pokemon 類別如下：
- 令物件實體不能存取其屬性
- 使方法設計更加合理
  - 屬性：(設為private權限)
    - …同上
  - 方法：
    - void ShowInfo(): 顯示資訊
    - void Attack(Pokemon &Target): 攻擊(並判斷其合理性)
    - void Defence(int n): 防御(並判斷其合理性)
    - void Cure(): 治癒
    - void setData(string name, ...): 設定其資料 (並判斷其合理性)

https://jgirl.ddns.net/problem/0/1109

練習

- 設計一個Square(正方型)類別包含下列成員：
  - 屬性 (私有)：
    - int len: 邊長
  - 方法 (公開)：
    - int area(): 計算面積
    - void setLen(int n): 設定邊長 (需判斷邊長值的正確性)
    - int getLen(): 取得邊長
- 使用Square類別宣告兩個邊長為10與20的物件,並印出它們的**邊長與面積**

https://jgirl.ddns.net/problem/0/1113

建構式與解構式

- 在定義類別時，您可以使用**建構函式(Constructor)**來進行物件的**初始化**
- 而在物件釋放資源之前，您也可以使用**解構函式(Destructor)**來進行一些**善後**的工作

建構式

- 思考: 由上一個輸入兩個人資料(姓名, 身高, 體重)並印出的範例，**你希望一開始姓名為No name, 身高與體重為0**，該如何達到此功能？
  - 使用建構式！

建構式

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

class Person
{
   public:
      Person() //建構式，名稱與class相同
      {
         name = "No name";
         height = 0;
         weight = 0;
      }
      void input()
      {
         cin >> name;
         cin >> height;
         cin >> weight;
      }
      void output()
      {
         cout << "Name:"
              << name << endl;
         cout << "Height:"
              << height << " cm" << endl;
         cout << "Weight:"
              << weight << " kg" << endl;
      }
private:
      string name;
      int  height;
      int  weight;
};
int main()
{
   Person p1; //物件實體建立時同時執行建構式
   Person p2;

   //p1. input();   //p1忘了輸入
   p1. output();    //p1印出No name
   p2. input();
   p2. output();
   return 0;
}
```


example

重載建構式

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

class Person
{
   public:
      Person()
      {
         name = "No name";
         height = 0;
         weight = 0;
      }
      Person(string n, int h, int w)
      {
         name = n;
         height = h;
         weight = w;
      }
      void input()
      {
         cin >> name;
         cin >> height;
         cin >> weight;
      }
      void output()
         {
            cout << "Name:"
                 << name << endl;
            cout << "Height:"
                 << height << " cm" << endl;
            cout << "Weight:"
                 << weight << " kg" << endl;
         }
   private:
      string name;
      int  height;
      int  weight;
};
int main()
{
   Person p1;
   Person p2("Andy", 180, 80); // 呼叫重載的建構式

   p1. output();
   p2. output();
   return 0;
}
```

example

解構式

- 範例：物件結束時印出ByeBye

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

class Person
{
   public:
      Person()
      {
         name = "No name";
         height = 0;
         weight = 0;
      }
      ~Person()//物件實體釋放前執行
      {
         cout << "ByeBye" << endl;
      }
      void input()
      {
         cin >> name;
         cin >> height;
         cin >> weight;
      }
      void output()
      {
         cout << "Name:"
              << name << endl;
         cout << "Height:"
              << height << " cm" << endl;
         cout << "Weight:"
              << weight << " kg" << endl;
      }
   private:
      string name;
      int  height;
      int  weight;
};
int main()
{
   Person p1;

   p1. output();
   return 0; //p1釋放前執行解構式
}
```

example

示範IV example

- 定義一 Pokemon 類別如下：
  - 屬性：
    - …同上
  - 方法：
    - Pokemon() :建構式
    - Pokemon(string name, ...): 重載建構式
    - ~Pokemon() :解構式
    - void Attack(): 攻擊
    - void Defence(): 防御(被攻擊？)
    - void Cure(): 治癒
    - void setData(string name, ...): 設定其資料 (並判斷其合理性)
    - void ShowInfo(): 顯示資訊

https://jgirl.ddns.net/problem/0/1110

練習 https://jgirl.ddns.net/problem/0/1114

- **設計一個Square(正方型)類別包含下列成員：**
  - 屬性 (私有)：
    - int len: 邊長
  - 方法 (公開)：
    - Square(): 建構式, 將邊長設為0
    - Square(int n): 建構式, 將邊長設為n(需判正確性)
    - int area(): 計算面積
    - void setLen(int n): 設定邊長 (需判斷邊長值的正確性)
    - int getLen(): 取得邊長
- **使用Square類別宣告三個邊長為10、20與不給初始值的物件，並印出它們的面積**

示範V example

- **定義一 Pokemon 類別同上再加入：**
  - 屬性：
    - …同上
    - string * Items;
    - int ItemNum;
  - 方法：
    - Pokemon() :建構式中配置100格空間給Items
    - Pokemon(string name, ...): 重載建構式中配置100格空間給Items
    - ~Pokemon() :解構式 釋放Items
    - void AddItem(): 增加身上攜帶的物品
    - void ShowInfo(): 顯示資訊時再列出身上所帶的物品

https://jgirl.ddns.net/problem/0/1111

課程大綱

- 類別 (Class)
- 作業

eCash程式

- 要求:
  - 設計一類別eCash模擬儲值卡的行為
  - 私有成員:
    - 屬性: Money 目前可用金額
  - 公開成員:
    - 方法: eCash() 建構式 (將Money初始化為0元)
    - 方法: void store(int m) 儲值 (將m存入Money中)
    - 方法: void pay(int m) 消費 (將Money消費m元)
      - (提醒: 需判斷目前eCash是否有足夠的金額消費)
    - 方法: void display() 顯示目前餘額 (將Money輸出於螢幕)
  - 主程式
    - 輸入's': 儲值
    - 輸入'p': 消費
    - 輸入'd': 查詢餘額
    - 輸入'q': 離開程式

- 範例：http://homepage.ntu.edu.tw/~jfanc/C/Demo/eCash.exe

https://jgirl.ddns.net/problem/0/1096

範例

- **Example:程式開始**

```
=== 歡迎使用eCash ===

您好，請選擇項目:
s: 儲值
p: 消費
d: 顯示餘額
q: 離開
>
```

範例

- **Example:儲值**

```
您好，請選擇項目:
s: 儲值
p: 消費
d: 顯示餘額
q: 離開
> s
請輸入儲存金額:
500
eCash: 您存了500元
```

```
您好，請選擇項目:
s: 儲值
p: 消費
d: 顯示餘額
q: 離開
> s
請輸入儲存金額:
-100
eCash: 請輸入大於0的金額
```

範例

- **Example:消 費**

```
您好，請選擇項目:
s: 儲值
p: 消費
d: 顯示餘額
q: 離開
> p
請輸入消費金額:
150
eCash: 您花了150元
```

```
您好，請選擇項目:
s: 儲值
p: 消費
d: 顯示餘額
q: 離開
> p
請輸入消費金額:
-100
eCash: 請輸入大於0的金額
```

```
您好，請選擇項目:
s: 儲值
p: 消費
d: 顯示餘額
q: 離開
> p
請輸入消費金額:
1000
eCash: 您的錢不夠
```

範例

- **Example: 查詢餘額**

```
您好，請選擇項目:
s: 儲值
p: 消費
d: 顯示餘額
q: 離開
> d
eCash: 您尚有350元
```

範例

- **Example:離開**

```
您好，請選擇項目:
s: 儲值
p: 消費
d: 顯示餘額
q: 離開
> q
謝謝，ByeBye!
Press any key to continue
```

延申閱讀

- 靜態成員
- Namespace
- https://docs.microsoft.com/zh-tw/cpp/cpp/namespaces-cpp?view=vs-2019
- https://jw1903.blogspot.com/2013/03/c-namespace.html
- https://zh.wikipedia.org/wiki/%E5%91%BD%E5%90%8D%E7%A9%BA%E9%97%B4
- https://openhome.cc/Gossip/CppGossip/Namespace.html