

Object Oriented Programming

Lec02: Class Introduction

Sun Chin-Yu (孫勤昱)

cysun@ntut.edu.tw

2023/09/22



大綱

- Schedule Review
- 什麼是類別
- 實作方法
- 生命週期
- RAI



Schedule Review



Schedule Review

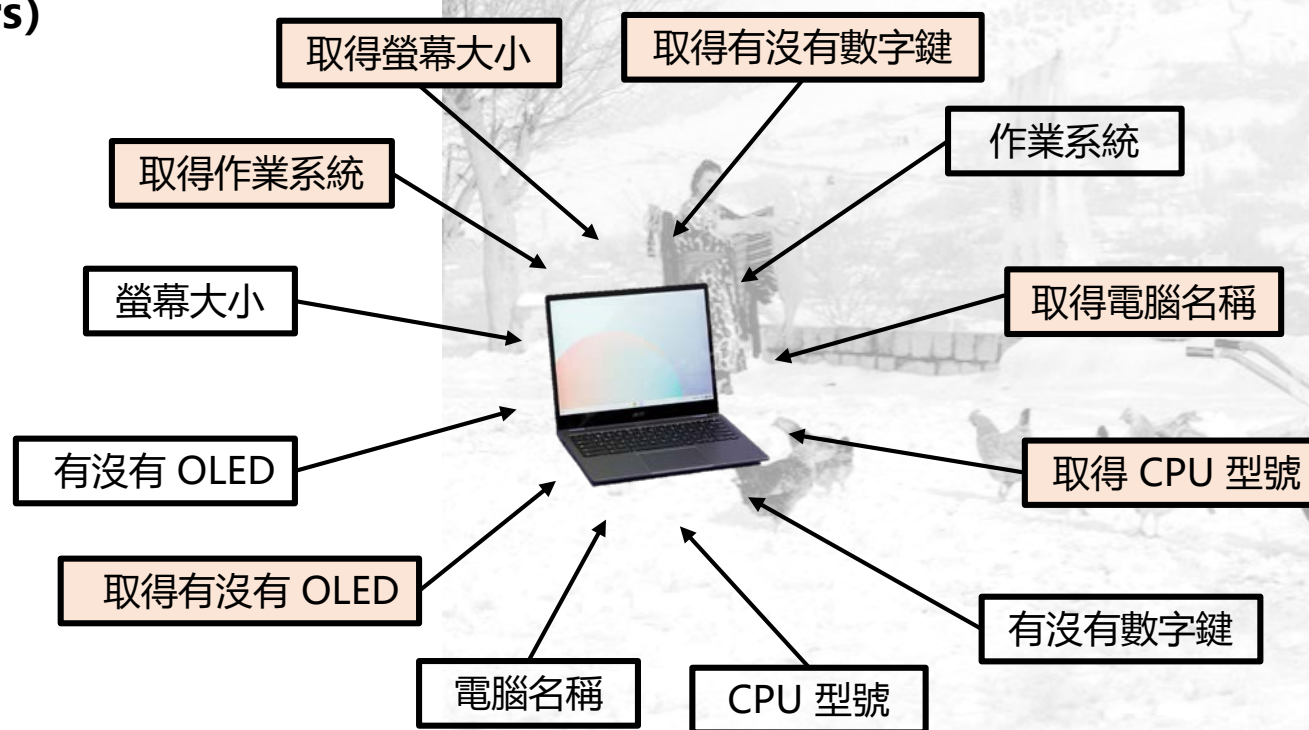
W	Date	Lecture	Homework
1	09/11, 09/15	Lec01: Course Information, Environment Introduction, and OOP Concept	Homework 00 (Fri.)
2	09/18, 09/22	Lec01: OOP Concept / Lec02: Class Introduction	
3	09/25, 09/29	Lec02: Class Introduction, and Essential STL Introduction / No class due to Moon Festival	Homework 01 (Mon.)
4	10/02, 10/06	Lec02: Class Introduction, and Essential STL Introduction	
5	10/09 , 10/13	No class due to the bridge holiday of Nation day / Lec03: Encapsulation	Homework 02 (Mon.)
6	10/16, 10/20	Lec03: Encapsulation / Lec04: Inheritance	
7	10/23, 10/27	Lec04: Inheritance	Homework 03 (Mon.)
8	10/30, 11/03	Lec04: Inheritance	
9	11/06, 11/10	Physical Hand-Written Midterm / Physical Computer-based Midterm	Midterm
10	11/13, 11/17	Lec05: Polymorphism	
11	11/20, 11/24	Lec05: Polymorphism	Homework 05 (Mon.)
12	11/27, 12/01	Lec05: Polymorphism	
13	12/04, 12/08	Lec06: Composition & Interface	Homework 06 (Mon.)
14	12/11, 12/15	Lec06: Composition & Interface	
15	12/18, 12/22	Lec07: Efficiency + Dependency Injection	Homework 07 (Mon.)
16	12/25, 12/29	Physical Hand-Written Final , Flexible time	
17	01/01 , 01/05	No class / Physical Computer-based Final	Final
18	01/08, 01/12	No class here	

什麼是類別 (Class)



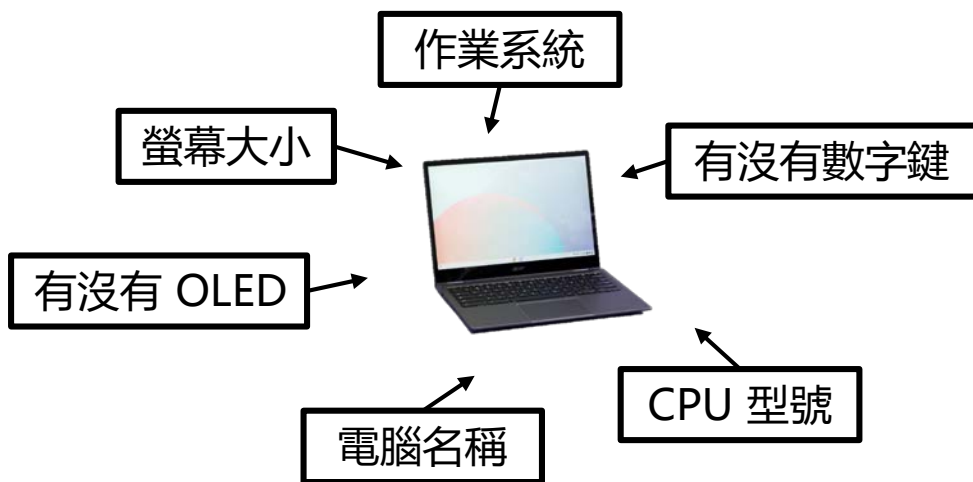
什麼是類別

- 用來描述物件 (Object) 的成員 (Members) 與方法 (Methods)
- 成員也稱為屬性 (Attributes) 、狀態 (States)
 - 是類別中的變數，用於儲存與物件相關的資料
- 方法也稱為函數 (Functions) 、行為 (Behaviors)
 - 是類別中，用於執行與物件相關的操作



什麼是類別

- 我們可以使用 C++ 的 class 來描述類別
- 例如我們要描述筆電 ...



```
1  #ifndef LAPTOP_HPP
2  #define LAPTOP_HPP
3
4  #include <string>
5
6  class Laptop {
7  public:
8      std::string operatingSystem = "Windows";           // 作業系統 (初始值為 "Windows")
9      double screenSize = 15.7;                          // 螢幕大小 (初始值為 15.7)
10     bool isOLED = false;                                // 是否為 OLED (初始值為 false)
11     std::string name = "Default Laptop";                // 筆電名稱 (初始值為 "Default Laptop")
12     std::string cpuName = "Letni i5-8700";              // CPU 型號 (初始值為 "Letni i5-8700")
13     bool haveNumberKey = false;                         // 是否有數字鍵 (初始值為 false)
14
15     // 建構子，輸入六個值，用於初始化上面的成員
16     Laptop(std::string operatingSystem, double screenSize, bool isOLED,
17           std::string name, std::string cpuName, bool haveNumberKey);
18
19     // 解構子，用來在物件要被釋放時釋放內部資源
20     ~Laptop();
21
22     std::string GetOpeartingSystem();                    // 取得作業系統
23     double GetScreenSize();                              // 取得螢幕大小
24     bool IsOLED();                                       // 是否為 OLED
25     std::string GetName();                               // 取得筆電名稱
26     std::string GetCPUName();                           // 取得 CPU 型號
27     bool HaveNumberKey();                               // 是否有數字鍵
28 };
29
30 #endif
31
```

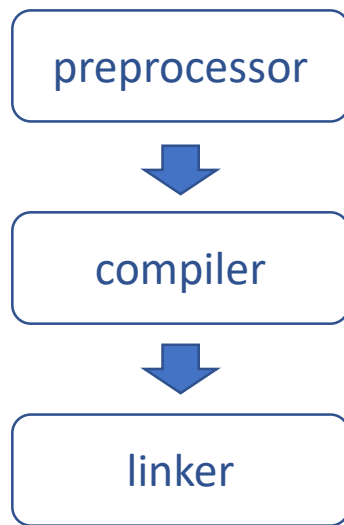

什麼是類別

- 紅框的部分是directives(指導語句)
 - 預處理器的指令

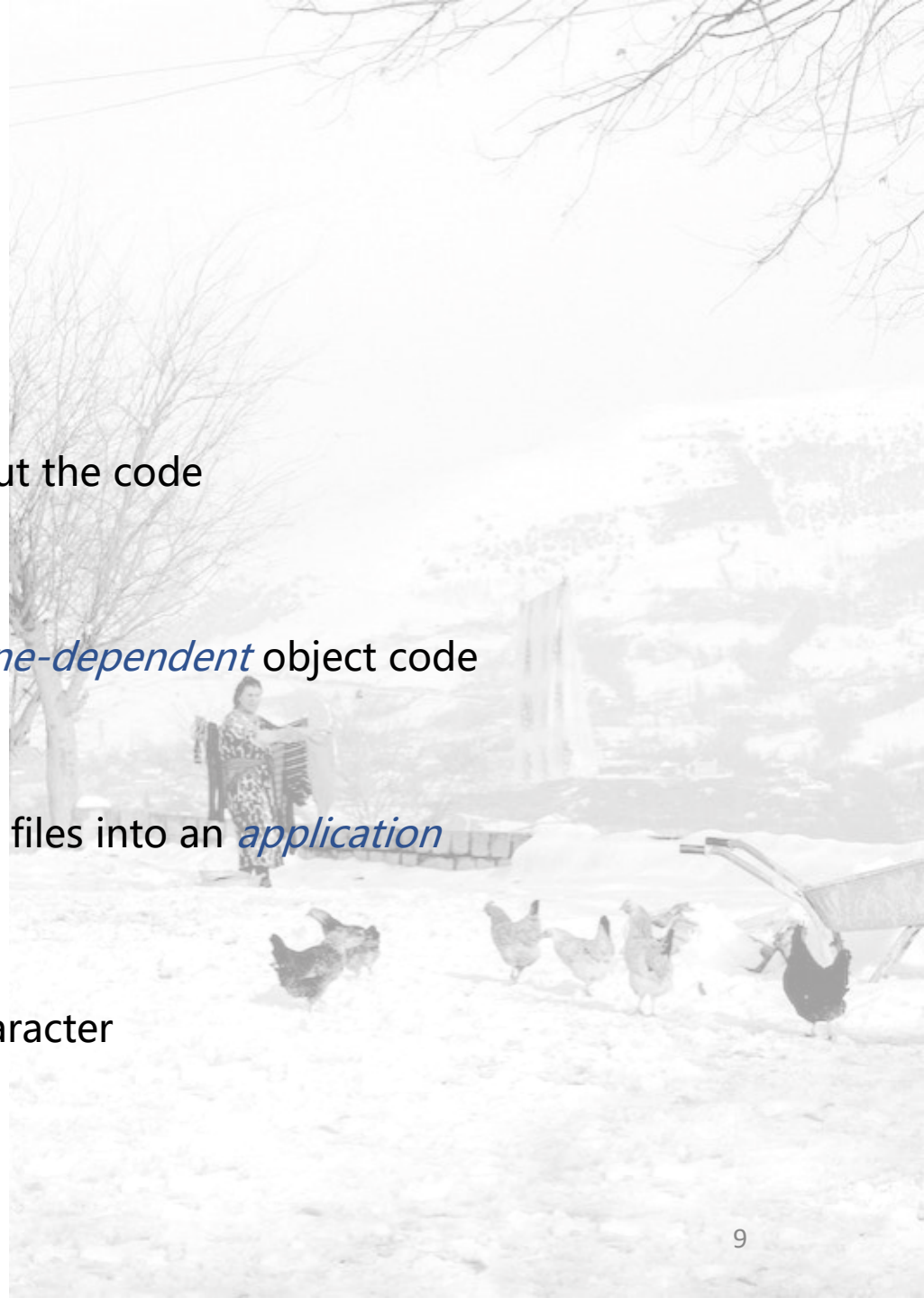
```
1  #ifndef LAPTOP_HPP
2  #define LAPTOP_HPP
3
4  #include <string>
5
6  class Laptop {
7  public:
8      std::string operatingSystem = "Windows";           // 作業系統 (初始值為 "Windows")
9      double screenSize = 15.7;                          // 螢幕大小 (初始值為 15.7)
10     bool isOLED = false;                                // 是否為 OLED (初始值為 false)
11     std::string name = "Default Laptop";                // 筆電名稱 (初始值為 "Default Laptop")
12     std::string cpuName = "Letni i5-8700";              // CPU 型號 (初始值為 "Letni i5-8700")
13     bool haveNumberKey = false;                         // 是否有數字鍵 (初始值為 false)
14
15     // 建構子，輸入六個值，用於初始化上面的成員
16     Laptop(std::string operatingSystem, double screenSize, bool isOLED,
17           std::string name, std::string cpuName, bool haveNumberKey);
18
19     // 解構子，用來在物件要被釋放時釋放內部資源
20     ~Laptop();
21
22     std::string GetOpeartingSystem();                   // 取得作業系統
23     double GetScreenSize();                             // 取得螢幕大小
24     bool IsOLED();                                     // 是否為 OLED
25     std::string GetName();                             // 取得筆電名稱
26     std::string GetCPUName();                          // 取得 CPU 型號
27     bool HaveNumberKey();                              // 是否有數字鍵
28 };
29
30 #endif
31
```


建構一個C++

- 是一個三步驟的過程

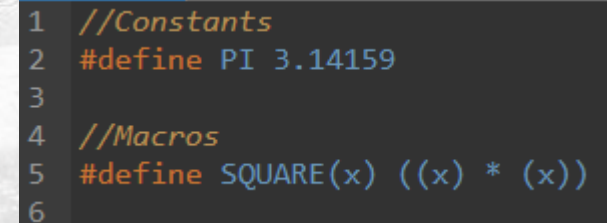


- Recognize *meta-information* about the code
 - Translate source code into *machine-dependent* object code
 - Link together all individual object files into an *application*
- Preprocessor aims at *directives*(指導語句) which starts with the # character
 - Ex. #include <iostream>



Most Common Directives

- **#include <[file]>**
 - Inserts the specified header file into the code at the location of directives
 - Ex. #include <iostream>
- **#define [key] [value]**
 - Replace the key with the value everywhere
 - Define constants or macros
- **#ifdef [key] #ifndef [key] #endif**
 - Includes/omits the code within ifdef/ifndef and endif blocks based on if key has been defined before with #define
 - Protects against circular includes



```
1 //Constants
2 #define PI 3.14159
3
4 //Macros
5 #define SQUARE(x) ((x) * (x))
6
```

Include Guards

- `#ifndef LAPTOP_HPP`: 這個指令會檢查是否已經定義了名為 `LAPTOP_HPP`，如果還未定義，則預處理器會繼續執行接下來的code
- `#define LAPTOP_HPP`: 這個指令定義了一個 `LAPTOP_HPP`。這樣做的目的是為了讓預處理器「記住」這個頭文件已經被包含過了，以避免未來重複包含
- `#endif`: 這個指令表示 `#ifndef` 條件的結束

```
1  #ifndef LAPTOP_HPP
2  #define LAPTOP_HPP
3
4  #include <string>
5
6  class Laptop {
7  public:
8      std::string operatingSystem = "Windows";           // 作業系統 (初始值為 "Windows")
9      double screenSize = 15.7;                          // 螢幕大小 (初始值為 15.7)
10     bool isOLED = false;                                // 是否為 OLED (初始值為 false)
11     std::string name = "Default Laptop";                // 筆電名稱 (初始值為 "Default Laptop")
12     std::string cpuName = "Letni i5-8700";               // CPU 型號 (初始值為 "Letni i5-8700")
13     bool haveNumberKey = false;                         // 是否有數字鍵 (初始值為 false)
14
15     // 建構子，輸入六個值，用於初始化上面的成員
16     Laptop(std::string operatingSystem, double screenSize, bool isOLED,
17           std::string name, std::string cpuName, bool haveNumberKey);
18
19     // 解構子，用來在物件要被釋放時釋放內部資源
20     ~Laptop();
21
22     std::string GetOpeartingSystem();                    // 取得作業系統
23     double GetScreenSize();                             // 取得螢幕大小
24     bool IsOLED();                                       // 是否為 OLED
25     std::string GetName();                              // 取得筆電名稱
26     std::string GetCPUName();                           // 取得 CPU 型號
27     bool HaveNumberKey();                               // 是否有數字鍵
28 };
29
30 #endif
31
```


什麼是類別

- 紅框的部分是類別名稱 (Class Name)
 - 用來當作這個類別 (Class) 的名稱

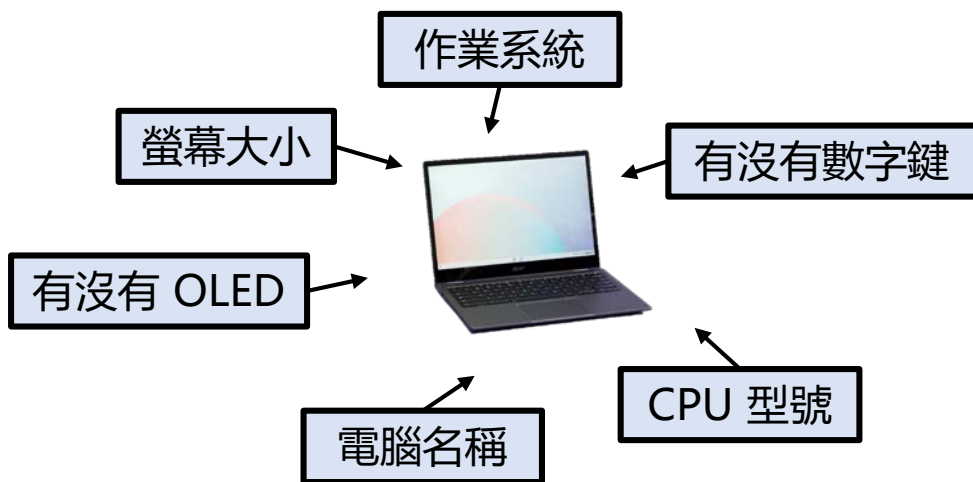


Laptop

```
1  #ifndef LAPTOP_HPP
2  #define LAPTOP_HPP
3
4  #include <string>
5
6  class Laptop {
7  public:
8      std::string operatingSystem = "Windows"; // 作業系統 (初始值為 "Windows")
9      double screenSize = 15.7; // 螢幕大小 (初始值為 15.7)
10     bool isOLED = false; // 是否為 OLED (初始值為 false)
11     std::string name = "Default Laptop"; // 筆電名稱 (初始值為 "Default Laptop")
12     std::string cpuName = "Letni i5-8700"; // CPU 型號 (初始值為 "Letni i5-8700")
13     bool haveNumberKey = false; // 是否有數字鍵 (初始值為 false)
14
15     // 建構子，輸入六個值，用於初始化上面的成員
16     Laptop(std::string operatingSystem, double screenSize, bool isOLED,
17           std::string name, std::string cpuName, bool haveNumberKey);
18
19     // 解構子，用來在物件要被釋放時釋放內部資源
20     ~Laptop();
21
22     std::string GetOperatingSystem(); // 取得作業系統
23     double GetScreenSize(); // 取得螢幕大小
24     bool IsOLED(); // 是否為 OLED
25     std::string GetName(); // 取得筆電名稱
26     std::string GetCPUName(); // 取得 CPU 型號
27     bool HaveNumberKey(); // 是否有數字鍵
28 };
29
30 #endif
31
```

什麼是類別

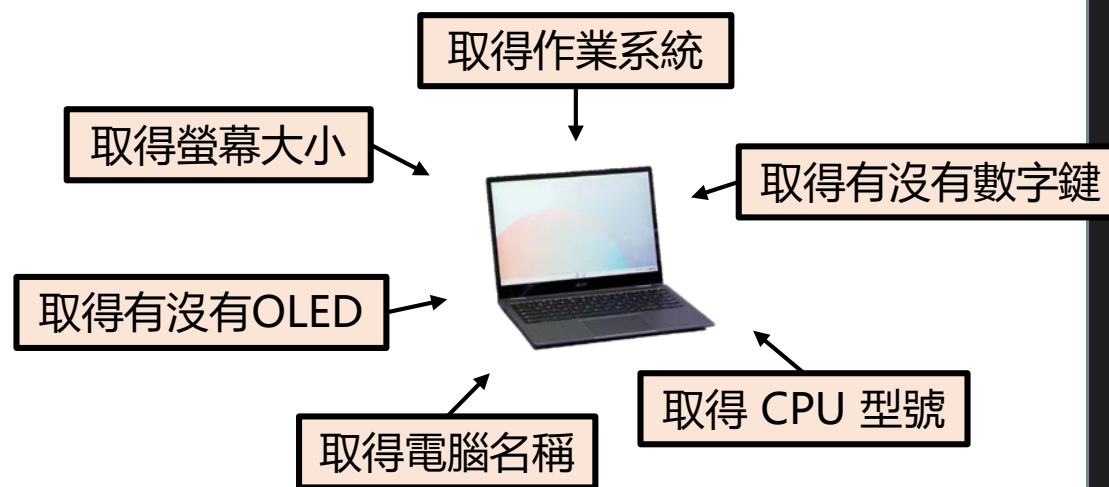
- 紅框的部分是類別**成員** (Class Member)
 - 用來描述這個類別所擁有的成員



```
1  #ifndef LAPTOP_HPP
2  #define LAPTOP_HPP
3
4  #include <string>
5
6  class Laptop {
7  public:
8      std::string operatingSystem = "Windows";           // 作業系統 (初始值為 "Windows")
9      double screenSize = 15.7;                         // 螢幕大小 (初始值為 15.7)
10     bool isOLED = false;                               // 是否為 OLED (初始值為 false)
11     std::string name = "Default Laptop";               // 筆電名稱 (初始值為 "Default Laptop")
12     std::string cpuName = "Letni i5-8700";              // CPU 型號 (初始值為 "Letni i5-8700")
13     bool haveNumberKey = false;                        // 是否有數字鍵 (初始值為 false)
14
15     // 建構子，輸入六個值，用於初始化上面的成員
16     Laptop(std::string operatingSystem, double screenSize, bool isOLED,
17           std::string name, std::string cpuName, bool haveNumberKey);
18
19     // 解構子，用來在物件要被釋放時釋放內部資源
20     ~Laptop();
21
22     std::string GetOpeartingSystem();                   // 取得作業系統
23     double GetScreenSize();                             // 取得螢幕大小
24     bool IsOLED();                                     // 是否為 OLED
25     std::string GetName();                             // 取得筆電名稱
26     std::string GetCPUName();                          // 取得 CPU 型號
27     bool HaveNumberKey();                             // 是否有數字鍵
28 };
29
30 #endif
31
```

什麼是類別

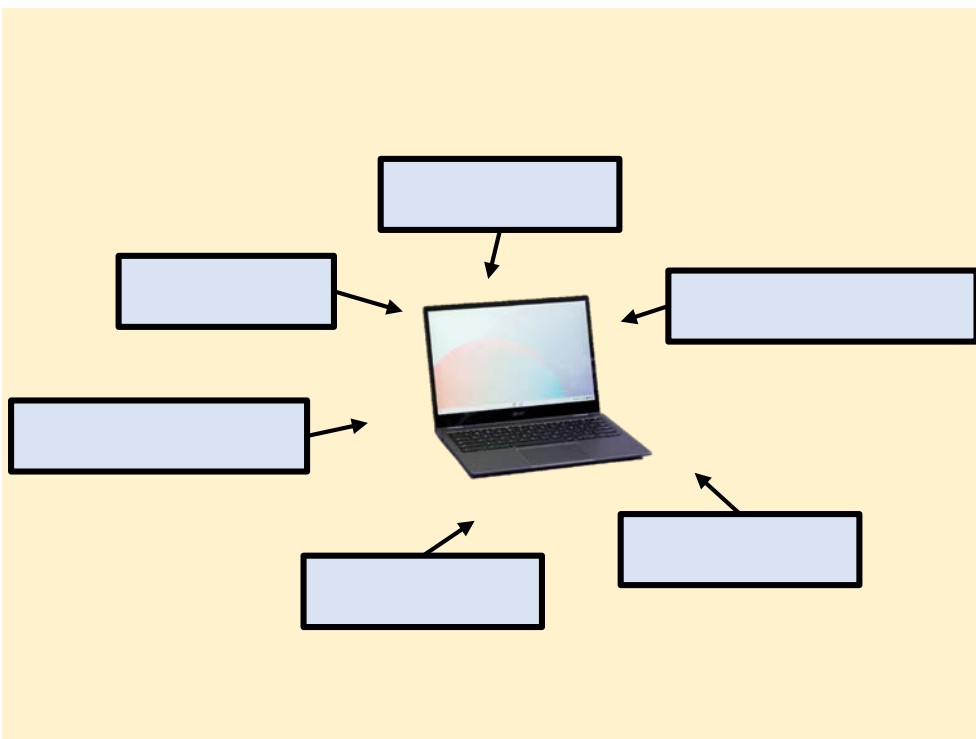
- 紅框的部分是類別**方法** (Class Method)
 - 用來描述這個類別所擁有的方法



```
1  #ifndef LAPTOP_HPP
2  #define LAPTOP_HPP
3
4  #include <string>
5
6  class Laptop {
7  public:
8      std::string operatingSystem = "Windows"; // 作業系統 (初始值為 "Windows")
9      double screenSize = 15.7; // 螢幕大小 (初始值為 15.7)
10     bool isOLED = false; // 是否為 OLED (初始值為 false)
11     std::string name = "Default Laptop"; // 筆電名稱 (初始值為 "Default Laptop")
12     std::string cpuName = "Letni i5-8700"; // CPU 型號 (初始值為 "Letni i5-8700")
13     bool haveNumberKey = false; // 是否有數字鍵 (初始值為 false)
14
15     // 建構子，輸入六個值，用於初始化上面的成員
16     Laptop(std::string operatingSystem, double screenSize, bool isOLED,
17           std::string name, std::string cpuName, bool haveNumberKey);
18
19     // 解構子，用來在物件要被釋放時釋放內部資源
20     ~Laptop();
21
22     std::string GetOpeartingSystem(); // 取得作業系統
23     double GetScreenSize(); // 取得螢幕大小
24     bool IsOLED(); // 是否為 OLED
25     std::string GetName(); // 取得筆電名稱
26     std::string GetCPUName(); // 取得 CPU 型號
27     bool HaveNumberKey(); // 是否有數字鍵
28 };
29
30 #endif
31
```


什麼是類別

- 我們使用**建構子**來初始化類別的成員

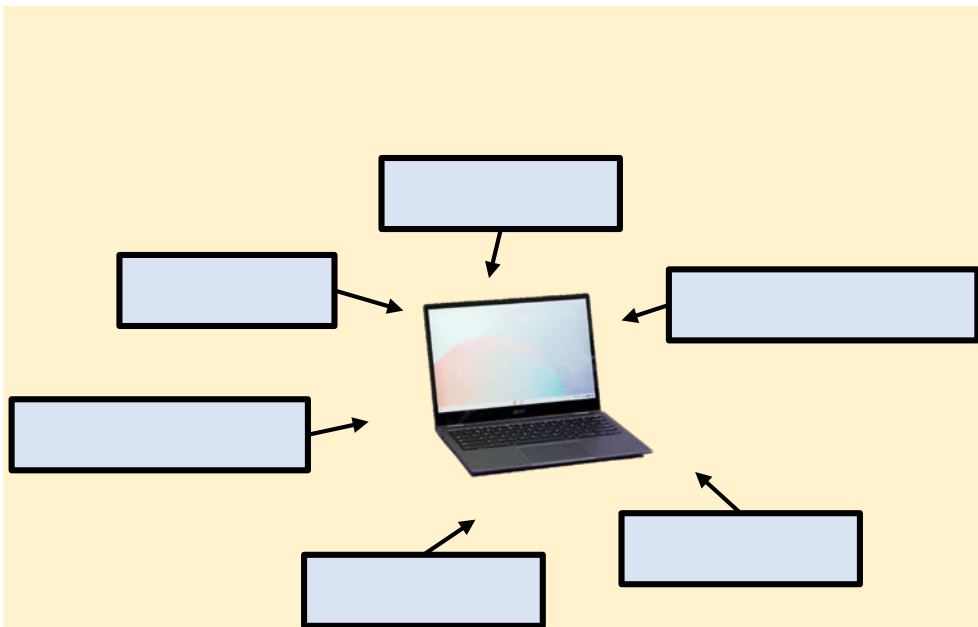


```
1  #ifndef LAPTOP_HPP
2  #define LAPTOP_HPP
3
4  #include <string>
5
6  class Laptop {
7  public:
8      std::string operatingSystem = "Windows";           // 作業系統 (初始值為 "Windows")
9      double screenSize = 15.7;                         // 螢幕大小 (初始值為 15.7)
10     bool isOLED = false;                               // 是否為 OLED (初始值為 false)
11     std::string name = "Default Laptop";               // 筆電名稱 (初始值為 "Default Laptop")
12     std::string cpuName = "Letni i5-8700";              // CPU 型號 (初始值為 "Letni i5-8700")
13     bool haveNumberKey = false;                        // 是否有數字鍵 (初始值為 false)
14
15     // 建構子，輸入六個值，用於初始化上面的成員
16     Laptop(std::string operatingSystem, double screenSize, bool isOLED,
17           std::string name, std::string cpuName, bool haveNumberKey);
18
19     // 解構子，用來在物件要被釋放時釋放內部資源
20     ~Laptop();
21
22     std::string GetOpeartingSystem();                   // 取得作業系統
23     double GetScreenSize();                             // 取得螢幕大小
24     bool IsOLED();                                     // 是否為 OLED
25     std::string GetName();                             // 取得筆電名稱
26     std::string GetCPUName();                          // 取得 CPU 型號
27     bool HaveNumberKey();                             // 是否有數字鍵
28 };
29
30 #endif
31
```

什麼是類別

- 可以透過建構子初始化成員，並建立物件

```
1 #include "laptop.hpp"
2
3 int main() {
4     Laptop myLaptop = Laptop("Windows", 15.9, true, "My Laptop", "Letni i9-48763", true);
5 }
```



```
1 #ifndef LAPTOP_HPP
2 #define LAPTOP_HPP
3
4 #include <string>
5
6 class Laptop {
7 public:
8     std::string operatingSystem = "Windows"; // 作業系統 (初始值為 "Windows")
9     double screenSize = 15.7; // 螢幕大小 (初始值為 15.7)
10    bool isOLED = false; // 是否為 OLED (初始值為 false)
11    std::string name = "Default Laptop"; // 筆電名稱 (初始值為 "Default Laptop")
12    std::string cpuName = "Letni i5-8700"; // CPU 型號 (初始值為 "Letni i5-8700")
13    bool haveNumberKey = false; // 是否有數字鍵 (初始值為 false)
14
15    // 建構子，輸入六個值，用於初始化上面的成員
16    Laptop(std::string operatingSystem, double screenSize, bool isOLED,
17          std::string name, std::string cpuName, bool haveNumberKey);
18
19    // 解構子，用來在物件要被釋放時釋放內部資源
20    ~Laptop();
21
22    std::string GetOperatingSystem(); // 取得作業系統
23    double GetScreenSize(); // 取得螢幕大小
24    bool IsOLED(); // 是否為 OLED
25    std::string GetName(); // 取得筆電名稱
26    std::string GetCPUName(); // 取得 CPU 型號
27    bool HaveNumberKey(); // 是否有數字鍵
28 };
29
30 #endif
31
```

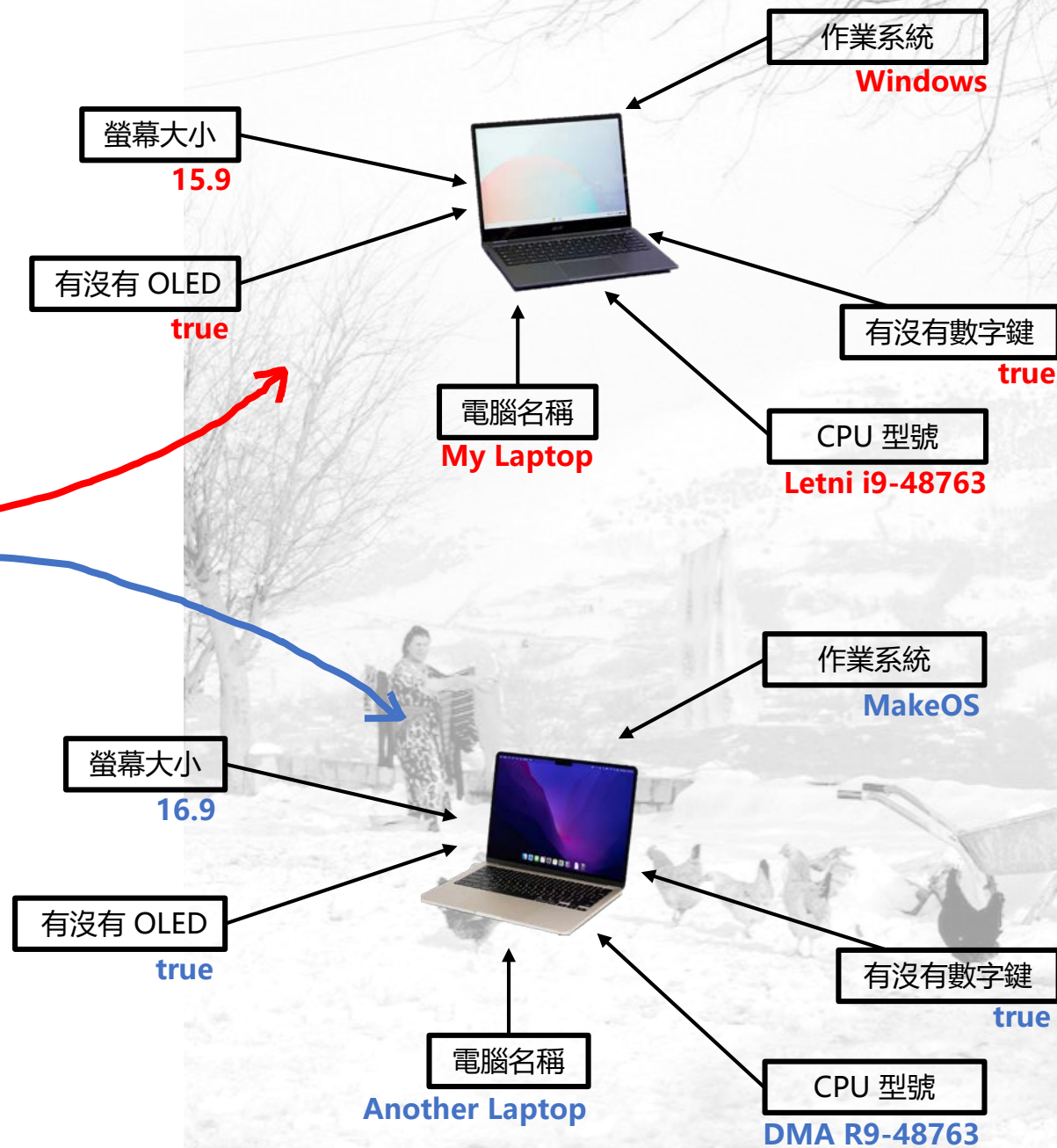
什麼是類別

- 然後就能用類別來建立出很多台電腦

```
1 #include "laptop.hpp"
2
3 int main(){
4     Laptop myLaptop = Laptop("Windows", 15.9, true, "My Laptop", "Letni i9-48763", true);
5     Laptop anotherLaptop = Laptop("MakeOS", 16.9, true, "Another Laptop", "DMA R9-48763", true);
6 }
```

- 並且用類別來取得筆電資訊

```
1 #include "laptop.hpp"
2
3 int main(){
4     Laptop myLaptop = Laptop("Windows", 15.9, true, "My Laptop", "Letni i9-48763", true);
5     Laptop anotherLaptop = Laptop("MakeOS", 16.9, true, "Another Laptop", "DMA R9-48763", true);
6
7     std::string myLaptopCPUName = myLaptop.cpuName;           // My Laptop
8     std::string anotherLaptopOperatingSystem = anotherLaptop.operatingSystem; // MakeOS
9 }
```

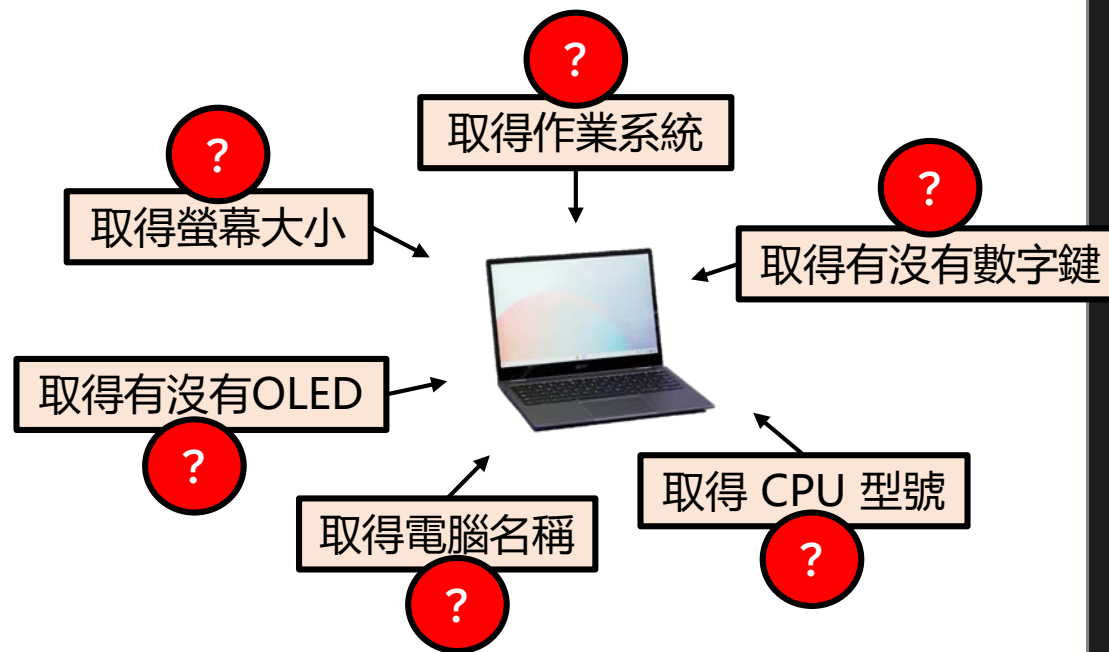


實作方法 (Method)



實作方法

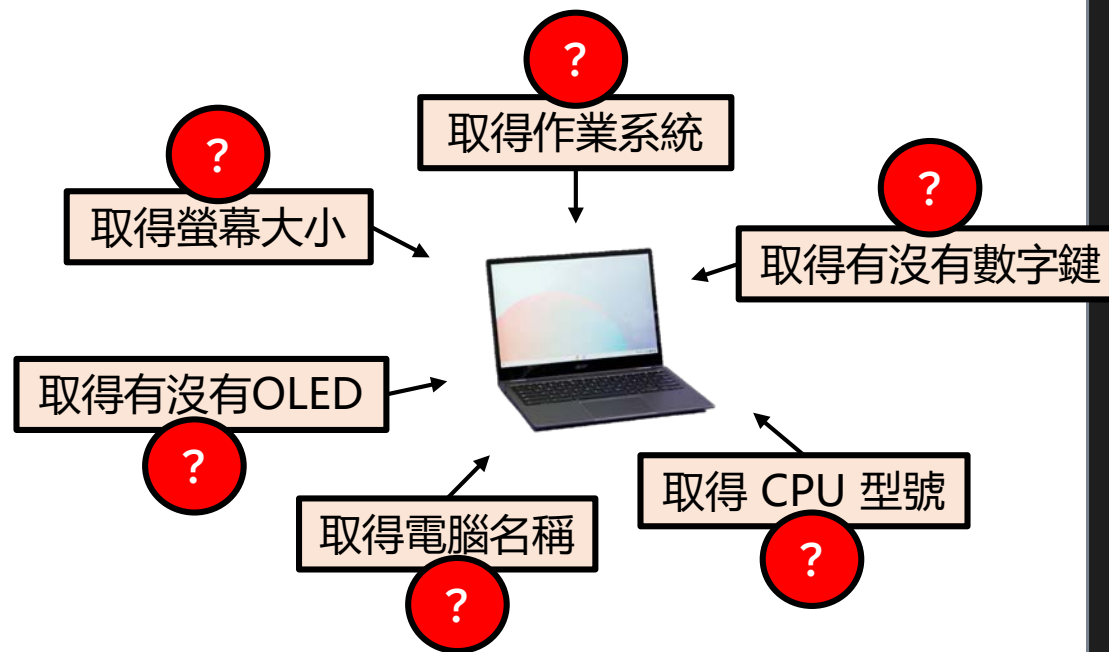
- 但電腦不會知道怎麼「取得作業系統」
- 所以我們需要實作方法的細節來讓電腦知道



```
1  #ifndef LAPTOP_HPP
2  #define LAPTOP_HPP
3
4  #include <string>
5
6  class Laptop {
7  public:
8      std::string operatingSystem = "Windows"; // 作業系統 (初始值為 "Windows")
9      double screenSize = 15.7; // 螢幕大小 (初始值為 15.7)
10     bool isOLED = false; // 是否為 OLED (初始值為 false)
11     std::string name = "Default Laptop"; // 筆電名稱 (初始值為 "Default Laptop")
12     std::string cpuName = "Letni i5-8700"; // CPU 型號 (初始值為 "Letni i5-8700")
13     bool haveNumberKey = false; // 是否有數字鍵 (初始值為 false)
14
15     // 建構子，輸入六個值，用於初始化上面的成員
16     Laptop(std::string operatingSystem, double screenSize, bool isOLED,
17           std::string name, std::string cpuName, bool haveNumberKey);
18
19     // 解構子，用來在物件要被釋放時釋放內部資源
20     ~Laptop();
21
22     std::string GetOpeartingSystem(); // 取得作業系統
23     double GetScreenSize(); // 取得螢幕大小
24     bool IsOLED(); // 是否為 OLED
25     std::string GetName(); // 取得筆電名稱
26     std::string GetCPUName(); // 取得 CPU 型號
27     bool HaveNumberKey(); // 是否有數字鍵
28 };
29
30 #endif
31
```

實作方法

- 在設計類別時，我們已經定義了方法



```
1  #ifndef LAPTOP_HPP
2  #define LAPTOP_HPP
3
4  #include <string>
5
6  class Laptop {
7  public:
8      std::string operatingSystem = "Windows"; // 作業系統 (初始值為 "Windows")
9      double screenSize = 15.7; // 螢幕大小 (初始值為 15.7)
10     bool isOLED = false; // 是否為 OLED (初始值為 false)
11     std::string name = "Default Laptop"; // 筆電名稱 (初始值為 "Default Laptop")
12     std::string cpuName = "Letni i5-8700"; // CPU 型號 (初始值為 "Letni i5-8700")
13     bool haveNumberKey = false; // 是否有數字鍵 (初始值為 false)
14
15     // 建構子，輸入六個值，用於初始化上面的成員
16     Laptop(std::string operatingSystem, double screenSize, bool isOLED,
17           std::string name, std::string cpuName, bool haveNumberKey);
18
19     // 解構子，用來在物件要被釋放時釋放內部資源
20     ~Laptop();
21
22     std::string GetOpeartingSystem(); // 取得作業系統
23     double GetScreenSize(); // 取得螢幕大小
24     bool IsOLED(); // 是否為 OLED
25     std::string GetName(); // 取得筆電名稱
26     std::string GetCPUName(); // 取得 CPU 型號
27     bool HaveNumberKey(); // 是否有數字鍵
28 };
29
30 #endif
31
```


實作方法

- 因此我們需要實作函數的細節

```
.hpp
1  #ifndef LAPTOP_HPP
2  #define LAPTOP_HPP
3
4  #include <string>
5
6  class Laptop {
7  public:
8      std::string operatingSystem = "Windows";    // 作業系統 (初始值為 "Windows")
9      double screenSize = 15.7;                  // 螢幕大小 (初始值為 15.7)
10     bool isOLED = false;                        // 是否為 OLED (初始值為 false)
11     std::string name = "Default Laptop";        // 筆電名稱 (初始值為 "Default Laptop")
12     std::string cpuName = "Letni i5-8700";      // CPU 型號 (初始值為 "Letni i5-8700")
13     bool haveNumberKey = false;                // 是否有數字鍵 (初始值為 false)
14
15     // 建構子, 輸入六個值, 用於初始化上面的成員
16     Laptop(std::string operatingSystem, double screenSize, bool isOLED,
17           std::string name, std::string cpuName, bool haveNumberKey);
18
19     // 解構子, 用來在物件要被釋放時釋放內部資源
20     ~Laptop();
21
22     std::string GetOperatingSystem();            // 取得作業系統
23     double GetScreenSize();                     // 取得螢幕大小
24     bool IsOLED();                              // 是否為 OLED
25     std::string GetName();                      // 取得筆電名稱
26     std::string GetCPUName();                   // 取得 CPU 型號
27     bool HaveNumberKey();                       // 是否有數字鍵
28 };
29
30 #endif
31
```

```
.cpp
1  #include "laptop.hpp"
2
3  #include <string>
4
5  Laptop::Laptop(std::string operatingSystem, double screenSize, bool isOLED,
6                std::string name, std::string cpuName, bool haveNumberKey){
7      this->operatingSystem = operatingSystem;
8      this->screenSize = screenSize;
9      this->isOLED = isOLED;
10     this->name = name;
11     this->cpuName = cpuName;
12     this->haveNumberKey = haveNumberKey;
13 }
14
15 Laptop::~~Laptop(){
16 }
17
18
19 std::string Laptop::GetOperatingSystem(){
20     return operatingSystem;
21 }
22
23 double Laptop::GetScreenSize(){
24     return screenSize;
25 }
26
27 bool Laptop::IsOLED(){
28     return isOLED;
29 }
30
31 std::string Laptop::GetName(){
32     return name;
33 }
34
35 std::string Laptop::GetCPUName(){
36     return cpuName;
37 }
38
39 bool Laptop::HaveNumberKey(){
40     return haveNumberKey;
41 }
```

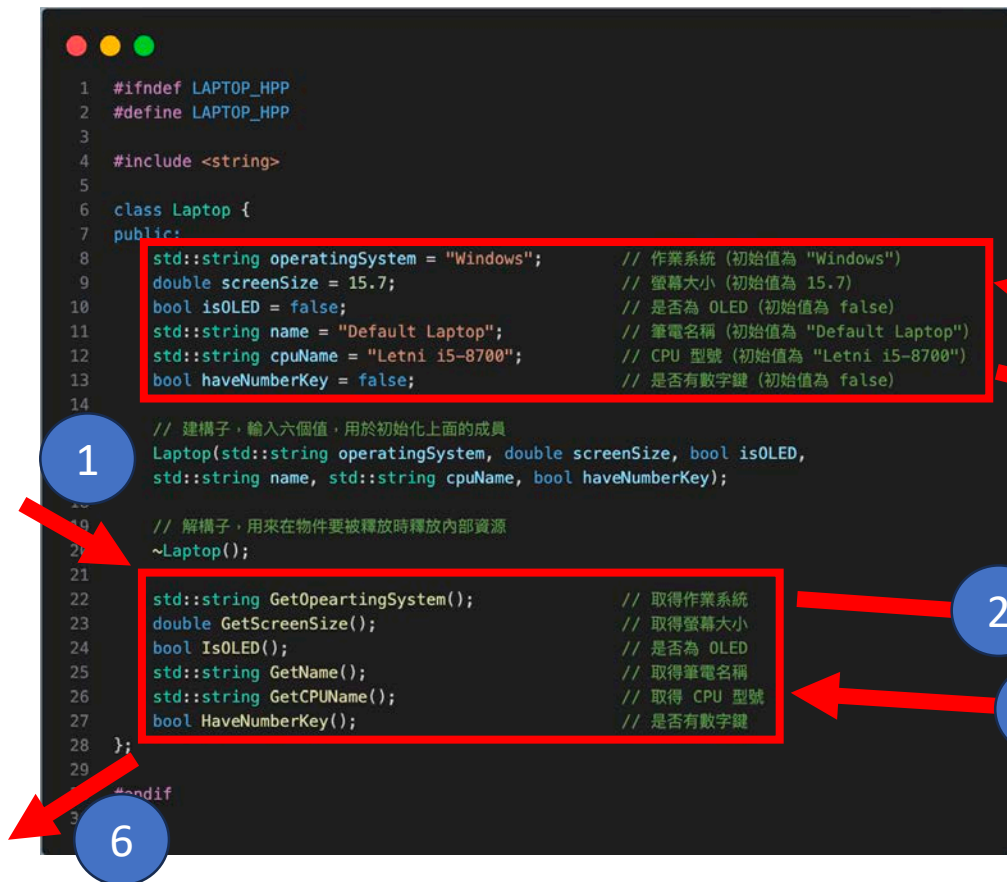
實作方法

- 為什麼我們需要區分 .hpp 與 .cpp?
 - 編譯效率
 - 當修改 .cpp 中的程式碼，只需要重新編譯這個文件，然後與其他已編譯過的對象文件進行鏈接（ Linking）
如果所有的代碼都放在一個文件中，即使是一個小變更也會需要重新編譯整個程式碼
 - 模組化（封裝）和重用
 - Head file通常只包含類別和函數的宣告，這樣其他文件就可以包含.hpp來使用這些類別和函數，而不需要知道它們是如何實現的
 - 將宣告和實作分開可以更容易地重用程式碼
 - 可讀性和維護
 - 當一個.hpp只包含類別和函數的宣告，會為讀程式碼地人提供了一個清晰、簡潔的視圖，不會被實現細節所干擾
 - 當一個程式碼由多人協作時，將介面和實作分開可以減少版本控制方面的衝突
- **慣例（大家的習慣）：會將宣告放在 .hpp 中，而將實作放在 .cpp 中**

實作方法

- 當我們完成了方法的定義後，就能夠呼叫方法。

```
1 #ifndef LAPTOP_HPP
2 #define LAPTOP_HPP
3
4 #include <string>
5
6 class Laptop {
7 public:
8     std::string operatingSystem = "Windows";    // 作業系統 (初始值為 "Windows")
9     double screenSize = 15.7;                  // 螢幕大小 (初始值為 15.7)
10    bool isOLED = false;                        // 是否為 OLED (初始值為 false)
11    std::string name = "Default Laptop";        // 筆電名稱 (初始值為 "Default Laptop")
12    std::string cpuName = "Letni i5-8700";      // CPU 型號 (初始值為 "Letni i5-8700")
13    bool haveNumberKey = false;                // 是否有數字鍵 (初始值為 false)
14
15    // 建構子，輸入六個值，用於初始化上面的成員
16    Laptop(std::string operatingSystem, double screenSize, bool isOLED,
17           std::string name, std::string cpuName, bool haveNumberKey);
18
19    // 解構子，用來在物件要被釋放時釋放內部資源
20    ~Laptop();
21
22    std::string GetOperatingSystem();            // 取得作業系統
23    double GetScreenSize();                     // 取得螢幕大小
24    bool IsOLED();                             // 是否為 OLED
25    std::string GetName();                     // 取得筆電名稱
26    std::string GetCPUName();                  // 取得 CPU 型號
27    bool HaveNumberKey();                      // 是否有數字鍵
28 };
29
30 #endif
```



```
1 #include "laptop.hpp"
2
3 #include <string>
4
5 Laptop::Laptop(std::string operatingSystem, double screenSize, bool isOLED,
6               std::string name, std::string cpuName, bool haveNumberKey){
7     this->operatingSystem = operatingSystem;
8     this->screenSize = screenSize;
9     this->isOLED = isOLED;
10    this->name = name;
11    this->cpuName = cpuName;
12    this->haveNumberKey = haveNumberKey;
13 }
14
15 Laptop::~~Laptop(){
16 }
17
18
19 std::string Laptop::GetOperatingSystem(){
20     return operatingSystem;
21 }
22
23 double Laptop::GetScreenSize(){
24     return screenSize;
25 }
26
27 bool Laptop::IsOLED(){
28     return isOLED;
29 }
30
31 std::string Laptop::GetName(){
32     return name;
33 }
34
35 std::string Laptop::GetCPUName(){
36     return cpuName;
37 }
38
39 bool Laptop::HaveNumberKey(){
40     return haveNumberKey;
41 }
```


生命週期 (Lifecycle)



生命週期

- 對於一個物件來說，會有所謂的生命週期
- 當離開物件所在的**程式碼區塊**後，物件將會被釋放



程式碼區塊

- 什麼是程式碼區塊？



```
1  TEST(LaptopTest, test_laptop) {  
2      Laptop laptop("Windows", 15.7, false, "Default Laptop", "Letni i5-8700", false);  
3  
4      ASSERT_EQ("Letni i5-8700", laptop.GetCPUName());  
5  }  
6
```

- 對於程式碼來說，裡面的 laptop 在這個測試函數的程式區塊內
- 當離開這個程式區塊（例如測試結束）後，laptop 會被自動釋放

解構子

- 怎麼釋放?
 - 使用解構子來描述物件哪些資源要被釋放（例如指標）

```
1  #include "laptop.hpp"
2
3  #include <string>
4
5  Laptop::Laptop(std::string operatingSystem, double screenSize, bool isOLED,
6                std::string name, std::string cpuName, bool haveNumberKey){
7      this->operatingSystem = operatingSystem;
8      this->screenSize = screenSize;
9      this->isOLED = isOLED;
10     this->name = name;
11     this->cpuName = cpuName;
12     this->haveNumberKey = haveNumberKey;
13 }
14
15 Laptop::~Laptop(){
16 }
17
18
19 std::string Laptop::GetOperatingSystem(){
20     return operatingSystem;
21 }
22
23 double Laptop::GetScreenSize(){
24     return screenSize;
25 }
26
27 bool Laptop::IsOLED(){
28     return isOLED;
29 }
30
31 std::string Laptop::GetName(){
32     return name;
33 }
34
35 std::string Laptop::GetCPUName(){
36     return cpuName;
37 }
38
39 bool Laptop::HaveNumberKey(){
40     return haveNumberKey;
41 }
```

生命週期

- 以生命週期來說：
 - 物件由建構子創立
 - 由解構子釋放



使用解構子釋放

```
1  #include "laptop.hpp"
2
3  #include <string>
4
5  Laptop::Laptop(std::string operatingSystem, double screenSize, bool isOLED,
6                std::string name, std::string cpuName, bool haveNumberKey){
7      this->operatingSystem = operatingSystem;
8      this->screenSize = screenSize;
9      this->isOLED = isOLED;
10     this->name = name;
11     this->cpuName = cpuName;
12     this->haveNumberKey = haveNumberKey;
13 }
14
15 Laptop::~Laptop(){
16 }
17
18
19 std::string Laptop::GetOperatingSystem(){
20     return operatingSystem;
21 }
22
23 double Laptop::GetScreenSize(){
24     return screenSize;
25 }
26
27 bool Laptop::IsOLED(){
28     return isOLED;
29 }
30
31 std::string Laptop::GetName(){
32     return name;
33 }
34
35 std::string Laptop::GetCPUName(){
36     return cpuName;
37 }
38
39 bool Laptop::HaveNumberKey(){
40     return haveNumberKey;
41 }
```

使用建構子創立



資源取得即初始化 (RAII)



RAII (Resource Acquisition Is Initialization)

- 資源使用所經歷三個步驟
 - 資源獲取
 - 資源使用
 - 資源釋放 (Programmer最常遺忘的環節)
- C++之父 (Bjarne Stroustrup) 提出了一個解決方案: RAII
 - 充分運用了 C++ 語言中區域物件自動銷毀的特性來控制資源的生命週期
- RAII精神
 - 期望利用建構子初始化的物件是**好的**
 - 簡化許多異常行為的判斷
 - 與其在取值時檢查是不是合法的, 不如讓這個不合法的狀況不存在



Bjarne Stroustrup

RAII - Bad Practice (1/2)

- 這段程式碼：
 - 期望飲料應該要有名字與大於零的容量
 - 若在取得時沒有名字與容量時，告訴使用者說沒有名字與容量
- 觀察：
 - 防止使用者輸出一些奇怪的情況
 - 在GetName()、GetVolume()檢常異常情況
 - 這還會發生什麼問題？
 - 只要每次用到這些值，就要額外再檢查一次
 - 使程式非常亂、不直觀

```
1  #include <string>
2
3  class Drink{
4  public:
5      std::string name;
6      int volume;
7
8      Drink(std::string name, int volume){
9          this->name = name;
10         this->volume = volume;
11     }
12
13     std::string GetName(){
14         if(name == ""){
15             throw std::string("No name");
16         }
17         return name;
18     }
19
20     int GetVolume(){
21         if(volume <= 0){
22             throw std::string("No volume");
23         }
24         return volume;
25     }
26 };
```

RAII - Bad Practice (2/2)

- 如果這時候我們要增加一個新的函式來計算飲料價格
 - 計算的方法為容量 (cc) * 0.03
 - Ex. $1000 * 0.03 = 30$ (元)
- 可以發現，由於我們需要額外保證 volume 是好的
- 導致重複出現了需要判斷 volume 是否大於 0 的程式區塊

```
1  #include <string>
2
3  class Drink{
4  public:
5      std::string name;
6      int volume;
7
8      Drink(std::string name, int volume){
9          this->name = name;
10         this->volume = volume;
11     }
12
13     std::string GetName(){
14         if(name == ""){
15             throw std::string("No name");
16         }
17         return name;
18     }
19
20     int GetVolume(){
21         if(volume <= 0){
22             throw std::string("No volume");
23         }
24         return volume;
25     }
26
27     int GetPrice(){
28
29
30
31
32     }
33 };
```


RAII - Good Practice

- 在建構子時即檢查並回饋使用者
 - 確保一開始所有的屬性都是使用合法的值
 - 未來就不需額外再次檢查
- 透過 RAII，我們可以確保物件內的成員都是使用合理的值。
- 透過不需額外判斷值是否合法而導致程式碼變得混亂。

```
1  #include <string>
2
3  class Drink{
4  public:
5      std::string name;
6      int volume;
7
8      Drink(std::string name, int volume){
9          if(name == ""){
10             throw std::string("No name");
11          }
12          if(volume <= 0){
13             throw std::string("No volume");
14          }
15          this->name = name;
16          this->volume = volume;
17      }
18
19      std::string GetName(){
20          return name;
21      }
22
23      int GetVolume(){
24          return volume;
25      }
26  };

```

Thanks!