



03 類別與物件

Fundamental Computer Programming- C++ Lab(II)

元智大學 | C++ 程式設計實習 (二)

張維元

課程投影片：[請從元智個人 Portal 下載](#)

Outline

- 01 物件導向與開發環境
- 02 陣列、向量與結構
- 03 類別與物件
- 04 物件導向程式設計: 多載
- 05 物件導向程式設計: 繼承
- 06 樣板
- 07 C++ 進階用法

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



結構 (Struct)

C/C++ 而結構 (Struct) 提供在一個變數內自定義儲存的資料屬性。

```
1  
2  
3 struct Book {  
4     string title;  
5     int price;  
6 };  
7  
8  
9
```

```
1 int main( ){  
2  
3     Book book1;  
4  
5     book1.title = "C++ How to Program";  
6     book1.price = 636;  
7  
8     cout << book1.title << endl;  
9     cout << book1.price << endl;  
10    return 0;  
11  
12 }
```

→ Struct 是一種可以自定義格式型態

結構 (Struct)


C/C++ 而結構 (Struct) 提供在一個變數內自定義儲存的資料屬性。

```
1  
2  
3 struct Book {  
4     string title;  
5     int price;  
6 };  
7  
8  
9
```

```
1 int main( ){  
2  
3     Book book1;  
4  
5     book1.title = "C++ How to Program";  
6     book1.price = 636;  
7  
8     cout << book1.title << endl;  
9     cout << book1.price << endl;  
10    return 0;  
11  
12 }
```

→ 定義了一個叫做「Book」的變數型態

Struct 就像是自定義的「變數」型態

| | | |
|---|--|---|
| <pre>1 struct Book { 2 string title; 3 int price; 4 }; 5 6 void printBook(Book b){ 7 cout << b.title << endl; 8 cout << b.price << endl; 9 } 10</pre> |  | <pre>1 int main(){ 2 3 Book book1; 4 5 book1.title = "C++ How to Program"; 6 book1.price = 636; 7 8 printBook(book1); 9 return 0; 10 }</pre> |
|---|--|---|

→ 宣告的物件實體也可以被當成參數傳入 function

成員函式

(member function)

```
1 struct Book {  
2  
3     string title;  
4     int price;  
5     void f(){  
6         cout << "I'm a book";  
7     }  
8  
9 };  
10
```

```
1  
2  
3 int main( ){  
4  
5     Book book1;  
6     book1.f();  
7  
8     return 0;  
9 }  
10
```

→ Struct 中也能夠定義函式，稱為成員函式

成員函式

(member function)

```
1 struct Book {  
2  
3     string title;  
4     int price;           ← 成員變數  
5     void f();           ← 成員函式  
6     cout << "I'm a book";  
7 }  
8  
9 };  
10
```

```
1  
2  
3 int main( ){  
4     Book book1;  
5     book1.f();  
6  
7     return 0;  
8 }  
9  
10
```

Struct 另一種初始化方法

```
1
2 int main( ){
3
4     Book book1;
5     book1.title = "C++ How to Program";
6     book1.price = 636;
7
8     return 0;
9 }
10
```

```
1
2 int main( ){
3
4     Book book1 = {
5         .title = "C++ How to Program",
6         .price = 656
7     };
8
9     return 0;
10 }
```

Struct 就像是自定義的「變數」型態

```
1
2
3 struct Book {
4     int id;
5     void f(int i){
6         cout << "I'm a book #";
7     }
8 };
9
10
```

```
1
2 int main( ){
3
4     Book books[10];
5     for(int i = 0; i < 10; i++){
6         books[i].id = i;
7         books[i].f(books[i].id);
8     }
9     return 0;
10 }
```

→ 宣告的實體也可以被當成元素組成一個陣列

Struct 就像是自定義的「變數」型態

```
1
2
3 struct Book {
4     int id;
5     void f(int i){
6         cout << "I'm a book #";
7     }
8 };
9
10
```

```
1 int main( ){
2
3     Book books[10];
4     for(int i = 0; i < 10; i++){
5         books[i].id = i;
6         books[i].f(books[i].id);
7     }
8     return 0;
9
10 }
```

→ 用 [...] 取得元素，用 . 取得成員

作業 #02

■#練習：請利用一個結構去記錄字串中每個字母的出現次數。

■Requirements：

1. 可重複輸入一個字串（假設字串內只會有大小寫英文字母）
2. 請用指定結構記錄字串中每個字母的出現次數
3. 輸出每個字母與其出現次數

■Sample Input：參考下頁

■Sample Output：參考下頁

參考程式碼與結果

- 你可以參考下列程式碼修改，也可以自己從頭開始寫。只要執行結果必須符合右邊的格式即可。

作業 #03.cpp

```
1  #include <iostream>
2  using namespace std;
3
4  struct Node {
5      char c;
6      int ascii;
7      int count;
8  };
9
10 int main (){
11     string s;
12     while (getline(cin,s)){
13         Node nodes[52];
14         // Your Code
15     }
16 }
```

Console

Shell

```
> clang++-7 -pthread -std=c++17 -o main Q x \
le01.cpp example02.cpp example03.cpp example0
4.cpp example05.cpp example06.cpp example07.c
pp example08.cpp example09.cpp main.cpp 作業
#01.cpp 作業 #02.cpp 作業 #03.cpp
+ ./main
aABbbzZz
A=> 2
B=> 1
Z=> 1
a=> 1
b=> 2
z=> 2
[]
```

```
1
2
3
4 int nodes[52];
5 char c = 'A';
6
7 nodes[int(c)] = nodes[int(c)]++
8
9
10
```

```
1
2
3
4 map<char, int> nodes;
5 char c = 'A';
6
7 nodes[c] = nodes[c]++
8
9
10
```

作業 #03.cpp

```
1  #include <iostream>
2  using namespace std;
3
4  struct Node {
5      char c;
6      int ascii;
7      int count;
8  };
9
10 int main (){
11     string s;
12     while (getline(cin,s)){
13         Node nodes[52];
14         // Your Code
15     }
16 }
```

```
1
2
3 Node nodes[52];
4 char c = 'A';
5
6 nodes[int(c)].char = ...
7 nodes[int(c)].ascii = ...
8 nodes[int(c)].count += 1
9
10
```


結構 (Struct)

C/C++ 而結構 (Struct) 提供在一個變數內自定義儲存的資料屬性。

```
1  
2  
3 struct Book {  
4     string title;  
5     int price;  
6 };  
7  
8  
9
```

```
1 int main( ){  
2  
3     Book book1;  
4  
5     book1.title = "C++ How to Program";  
6     book1.price = 636;  
7  
8     cout << book1.title << endl;  
9     cout << book1.price << endl;  
10    return 0;  
11  
12 }
```

→ Struct 是一種可以自定義格式型態

從結構 (struct) 到類別 (class)

C/C++ 而結構 (Struct) 提供在一個變數內自定義儲存的資料屬性。

```
1  
2 struct Book {  
3     string title;  
4     int price;  
5 };  
6
```

```
1  
2 class Book {  
3     public:  
4         string title;  
5         int price;  
6     };  
7
```

結構與類別的差別

| 差別 | C 語言 Struct | C++ 語言 Struct | C++ 語言 Class |
|--------|-------------|------------------------|------------------------|
| 成員變數 | 不支援靜態變數 | 支援靜態變數 | 支援靜態變數 |
| 成員函式 | 不支援成員函式 | 支援成員函式 | 支援成員函式 |
| 成員預設值 | 不支援成員預設值 | 支援成員預設值 | 支援成員預設值 |
| 成員範圍 | public | public、protect、private | public、protect、private |
| 預設成員範圍 | X | public | private |
| 物件導向特性 | X | 繼承、多型、建構子 ... | 繼承、多型、建構子 ... |

→ Class 就像是多了繼承特性的 Struct

結構與類別的差別

| 差別 | C 語言 Struct | C++ 語言 Struct | C++ 語言 Class |
|--------|-------------|------------------------|------------------------|
| 成員變數 | 不支援靜態變數 | 支援靜態變數 | 支援靜態變數 |
| 成員函式 | 不支援成員函式 | 支援成員函式 | 支援成員函式 |
| 成員預設值 | 不支援成員預設值 | 支援成員預設值 | 支援成員預設值 |
| 成員範圍 | public | public、protect、private | public、protect、private |
| 預設成員範圍 | X | public | private |
| 物件導向特性 | X | O | O |

→ C++ Class 就像是多了繼承特性的 C 語言 Struct

C++ 的結構 (struct) 很像 C++ 的類別 (class)

→ C++ 結構支援了 C++ 類別大部分的性質與用法

```
1  
2 struct Book {  
3     string title;  
4     int price;  
5 };  
6
```

≠

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

C++ 的結構 (struct) 很像 C++ 的類別 (class)

→ 其中一個主要的差別在於預設的變數範圍不同

```
1  
2 struct Book {  
3     string title;  
4     int price;  
5 };  
6
```

≠

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

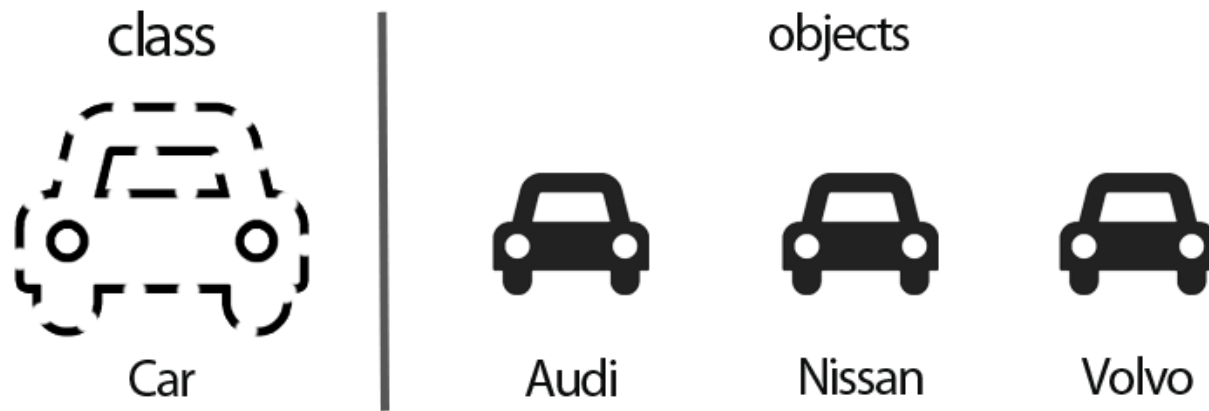
從結構 (struct) 到類別 (class)

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

```
int main( ){
    Book book1;
    book1.title = "C++ How to Program";
    book1.price = 636;
    cout << book1.title << endl;
    cout << book1.price << endl;
    return 0;
}
```

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) ， 可以使用類別方法



→ Class 是抽象的描述， Object 才是實體的存在

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) ， 可以使用類別方法

```
1  
2  
3 class Book {  
4     public:  
5         string title;  
6         int price;  
7 };  
8  
9
```

```
1 int main( ){  
2  
3     Book book1;  
4  
5     book1.title = "C++ How to Program";  
6     book1.price = 636;  
7  
8     cout << book1.title << endl;  
9     cout << book1.price << endl;  
10    return 0;  
11  
12 }
```

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) ， 可以使用類別方法

```
1  
2  
3 class Book {  
4     public:  
5         string title;  
6         int price;  
7 };  
8  
9
```

```
1 int main( ){  
2  
3     Book book1;  
4  
5     book1.title = "C++ How to Program";  
6     book1.price = 636;  
7  
8     cout << book1.title << endl;  
9     cout << book1.price << endl;  
10    return 0;  
11  
12 }
```

→ 定義了一個叫做「Book」的變數型態

成員函式

(member function)

```
1
2 class Book {
3     public:
4         string title;
5         int price;
6         void f(void);
7 };
8
9 void Book::f(){
10     cout << "I'm a book: " + title;
11 };
12
```

```
1
2 int main( ){
3
4     Book book1;
5     book1.title = "C++ How to Program";
6     book1.f();
7
8     return 0;
9 }
10
11
12
```

成員函式

(member function)

```
1
2 class Book {
3     public:
4         string title;
5         int price;           ← 成員變數
6         void f(void);        ← 成員函式
7 };
8
9 void Book::f(){
10     cout << "I'm a book: " + title;
11 };
12
```

```
1
2 int main( ){
3
4     Book book1;
5     book1.title = "C++ How to Program";
6     book1.f();
7
8     return 0;
9 }
10
11
12
```

成員函式

(member function)

```
1
2 class Book {
3     public:
4         string title;
5         int price;           ← 成員變數
6         void f(void);       ← 成員函式
7 };
8
9 void Book::f(){
10     cout << "I'm a book: " + title;
11 };
12
```

→ 成員函式可存取成員變數

```
1
2 int main( ){
3
4     Book book1;
5     book1.title = "C++ How to Program";
6     book1.f();
7
8     return 0;
9 }
10
11
12
```

兩種建立物件的方式

```
1
2
3 int main( ){
4
5     Book book1;
6     book1.set(600);
7     cout << "price: " << book1.get();
8
9     return 0;
10 }
11
12
```

```
1
2
3 int main( ){
4
5     Book* book1 = new Book();
6     book1->set(600);
7     cout << "price: " << book1->get();
8
9     return 0;
10 }
11
12
```

由 Objects 組成的陣列

```
1 class Book {  
2     public:  
3         int id;  
4         void f();  
5     };  
6  
7     void Book::f(void){  
8         cout << "I'm a book: " << id;  
9     };  
10 }
```

```
1 int main( ){  
2  
3     Book books[10];  
4     for(int i = 0; i < 10; i++){  
5         books[i].id = i;  
6         books[i].f();  
7     }  
8     return 0;  
9  
10 }
```

→ 宣告的實體也可以被當成元素組成一個陣列

由 Objects 組成的陣列

```
1 class Book {  
2     public:  
3         int id;  
4         void f();  
5     };  
6  
7     void Book::f(void){  
8         cout << "I'm a book: " << id;  
9     };  
10
```

```
1 int main( ){  
2  
3     Book books[10];  
4     for(int i = 0; i < 10; i++){  
5         books[i].id = i;  
6         books[i].f();  
7     }  
8     return 0;  
9 }  
10
```

→ 用 [...] 取得元素，用 . 取得成員

結構與類別的差別

| 差別 | C 語言 Struct | C++ 語言 Struct | C++ 語言 Class |
|--------|-------------|------------------------|------------------------|
| 成員變數 | 不支援靜態變數 | 支援靜態變數 | 支援靜態變數 |
| 成員函式 | 不支援成員函式 | 支援成員函式 | 支援成員函式 |
| 成員預設值 | 不支援成員預設值 | 支援成員預設值 | 支援成員預設值 |
| 成員範圍 | public | public、protect、private | public、protect、private |
| 預設成員範圍 | X | public | private |
| 物件導向特性 | X | 繼承、多型、建構子 ... | 繼承、多型、建構子 ... |

→ 其中一個主要的差別在於預設的變數範圍不同

未指定的成員預設是私有變數

```
1
2 class Book {
3     int price;
4     public:
5         void set(int p);
6         double get(void);
7 };
8
9 void Book::set(int p){
10     price = p;
11 };
12
13 double Book::get(void){
14     return price * 0.9;
15 };
16
```

```
1 int main( ){
2
3     Book book1;
4     book1.price = 600;
5     cout << "price: " << book1.get();
6
7     return 0;
8 }
9
10
```

未指定的成員預設是私有變數

```
1
2 class Book {
3     private:
4         int price;
5     public:
6         void set(int p);
7         double get(void);
8 };
9
10 void Book::set(int p){
11     price = p;
12 };
13
14 double Book::get(void){
15     return price * 0.9;
16 };
```

```
1 int main( ){
2
3     Book book1;
4     book1.price = 600;
5     cout << "price: " << book1.get();
6
7     return 0;
8 }
9
10
```

Private 和 Public

```
1
2 class Book {
3     private:      ← 私有成員
4         int price;
5     public:       ← 公開函式
6         void set(int p);
7         double get(void);
8 };
9
10 void Book::set(int p){
11     price = p;
12 };
13
14 double Book::get(void){
15     return price * 0.9;
16 };
```

```
1 int main( ){
2
3     Book book1;
4     book1.price = 600;
5     cout << "price: " << book1.get();
6
7     return 0;
8 }
9
10
```

不能直接存取私有變數

```
1
2 class Book {
3     private:      ← 私有成員
4         int price;
5     public:      ← 公開函式
6         void set(int p);
7         double get(void);
8 };
9
10 void Book::set(int p){
11     price = p;
12 };
13
14 double Book::get(void){
15     return price * 0.9;
16 };
```

```
1 int main( ){
2
3     Book book1;
4     book1.price = 600;
5     cout << "price: " << book1.get();
6
7     return 0;
8 }
9
10
```

利用公開成員函式存取私有成員變數

```
1
2 class Book {
3     private:      ← 私有成員
4         int price;
5     public:      ← 公開函式
6         void set(int p);
7         double get(void);
8 };
9
10 void Book::set(int p){
11     price = p;
12 };      → 成員函式可存取成員變數
13
14 double Book::get(void){
15     return price * 0.9;
16 };
```

```
1 int main( ){
2
3     Book book1;
4     book1.set(600);
5     cout << "price: " << book1.get();
6
7     return 0;
8 }
9
10
```

兩種建立物件的方式

```
1
2
3 int main( ){
4
5     Book book1;
6     book1.set(600);
7     cout << "price: " << book1.get();
8
9     return 0;
10 }
11
12
```

```
1
2
3 int main( ){
4
5     Book* book1 = new Book();
6     book1->set(600);
7     cout << "price: " << book1->get();
8
9     return 0;
10 }
11
12
```

本週上機練習

作業 #01

■#練習：請寫一個程式，讓使用者猜字元是否存在數字列表中。

■Requirements：

1. 請用一個容器記錄存放使用者輸入的字元（不可重複，輸入 0 終止）
2. 可重複輸入字元進行查詢，判斷是否存在容器內（猜中後要刪除）
3. 除了輸入輸出之外，不可使用其他迴圈

■Sample Input：參考下頁

■Sample Output：參考下頁

■Note：僅限 02/23、03/02 上課繳交

參考程式碼與結果

- 你可以參考下列程式碼修改，也可以自己從頭開始寫。只要執行結果必須符合右邊的格式即可。

作業 #01.cpp ×

```
1  #include <iostream>
2  using namespace std;
3
4  int main () {
5      char c;
6      while (cin >> c) {
7          /* Your Code */
8      }
9      cout << " ---- " << endl;
10     while (cin >> c) {
11         /* Your Code */
12     }
13 }
14
```

Console Shell

```
> clang++-7 -pthread -std=c++17 -o main archived/作業 #01.cpp Q ×
ved/作業 #02.cpp archived/作業 #03.cpp example01.cpp example02.cpp
example03.cpp example04.cpp example05.cpp example06.cpp example07.c
pp example08.cpp example09.cpp main.cpp 作業 #01.cpp
> ./main
1
f
c
0
----
d
d 不存在
f
f 有存在
1
1 有存在
[]
```

← 使用者輸入

作業 #03

■#練習：請定義一個 Book 的類別，根據亂數給值後排序。

■Requirements：

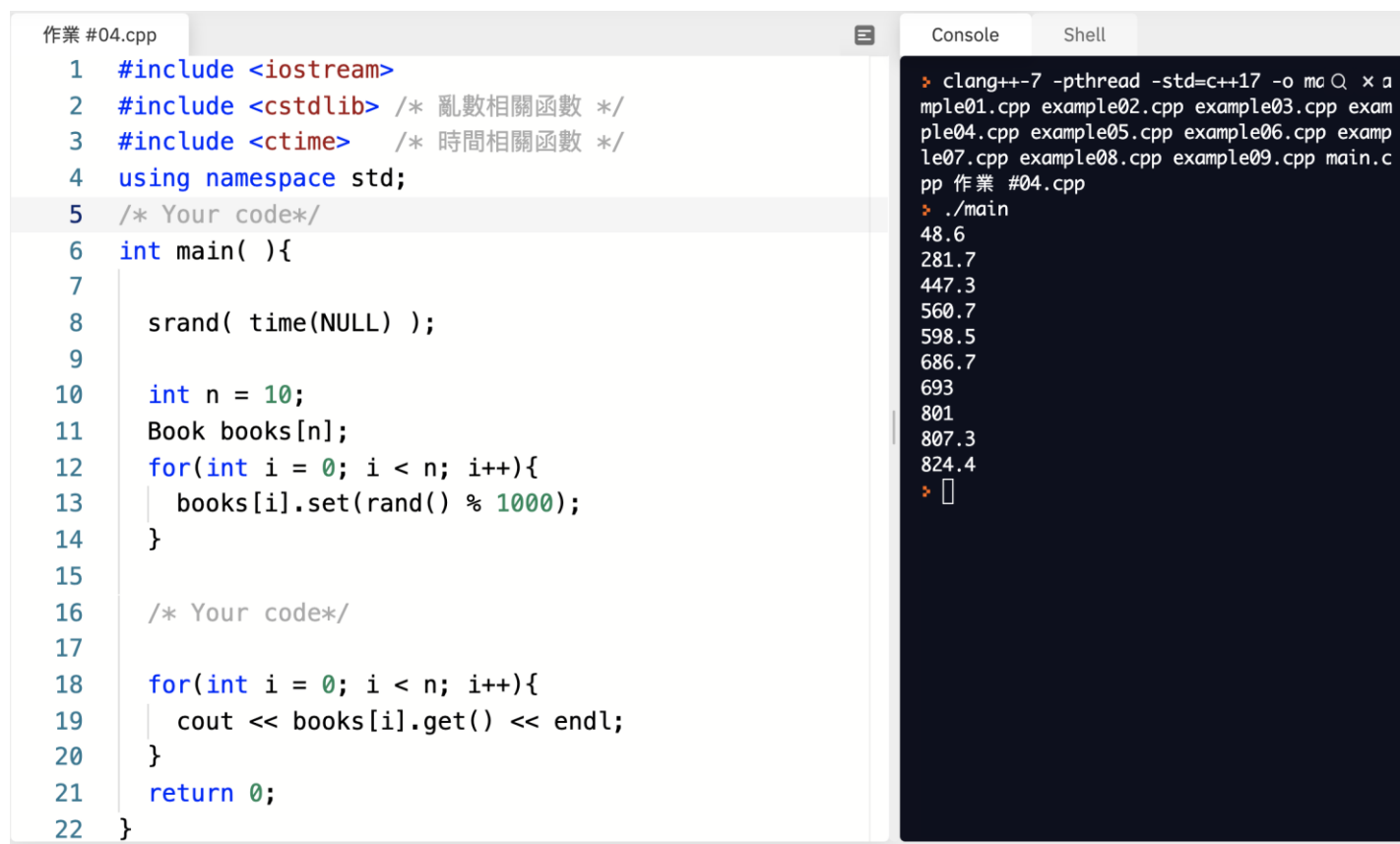
1. 定義一個 Book 類別包含私有變數：price、公有變數：get() 和 set()
2. 僅可利用公有變數 get() 和 set() 存取私有變數 price
3. 產生 10 個 Book 物件，並且根據亂數初始化 price
4. 將 10 個物件的 price 由小到大排序
5. (加分) 排序過程中使用到自定義的 swap 函數

■Sample Input：參考下頁

■Sample Output：參考下頁

參考程式碼與結果

- 你可以參考下列程式碼修改，也可以自己從頭開始寫。只要執行結果必須符合右邊的格式即可。



The screenshot shows a C++ IDE with two panels. The left panel displays the source code for '作業 #04.cpp', and the right panel shows the console output.

```
作業 #04.cpp
1  #include <iostream>
2  #include <cstdlib> /* 亂數相關函數 */
3  #include <ctime>   /* 時間相關函數 */
4  using namespace std;
5  /* Your code*/
6  int main( ){
7
8      srand( time(NULL) );
9
10     int n = 10;
11     Book books[n];
12     for(int i = 0; i < n; i++){
13         books[i].set(rand() % 1000);
14     }
15
16     /* Your code*/
17
18     for(int i = 0; i < n; i++){
19         cout << books[i].get() << endl;
20     }
21     return 0;
22 }
```

Console

```
> clang++-7 -pthread -std=c++17 -o main example01.cpp example02.cpp example03.cpp example04.cpp example05.cpp example06.cpp example07.cpp example08.cpp example09.cpp main.cpp
> ./main
48.6
281.7
447.3
560.7
598.5
686.7
693
801
807.3
824.4
>
```

作業 #04

■#練習：根據 Rectangle Class 繪製出空心的矩形。

■Requirements：

1. 定義一個 Rectangle 類別包含長、寬私有變數（不可超過 20）
2. 定義一個畫出空心矩形的公有變數
3. 使用者可根據 Rectangle 類別產生物件並輸入長寬，最後畫出矩形

■Sample Input：參考下頁

■Sample Output：參考下頁

參考程式碼與結果

- 你可以參考下列程式碼修改，也可以自己從頭開始寫。只要執行結果必須符合右邊的格式即可。

作業 #05.cpp

```
1  #include <iostream>
2  using namespace std;
3
4  class Rectangle{
5      /* Your Code */
6  };
7
8  int main(){
9      Rectangle rectangle;
10
11      /* Your Code */
12
13      rectangle.draw();
14      return 0;
15  }
```

Console

Shell

```
> clang++-7 -pthread -std=c++17 -o mc Q x a
mple01.cpp example02.cpp example03.cpp exam
ple04.cpp example05.cpp example06.cpp exam
ple07.cpp example08.cpp example09.cpp main.c
pp 作業 #04.cpp 作業 #05.cpp
> ./main
6
4
*****
*   *
*   *
*****
> []
```

作業繳交說明

你需要繳交以下檔案到 Portal 作業：

- 1. cpp 程式碼
- 2. 程式碼內有文字說明的註解
- 3. 執行結果截圖

(若無法上傳多個檔案，請壓縮成 zip 或 rar 格式，
並且命名成「學號.zip」或「學號.rar」)

■ 上課驗收者只需上傳程式碼即可

■ 本次作業截止時間：

03/06 (日) 23:59

```
main.cpp
1 // 學號： 1091000 姓名：王大明
2 /* 文字說明：這個作業當中，我利用一個變數 x 存放名字並且利用
   cout 做印出。 */
3
4 #include <iostream>
5 #include <string>
6 using namespace std;
7
8 int main() {
9     string x = "王大明";
10    std::cout << "Hello World, " << x << endl;
11 }
```

```
> clang++-7 -pthread -std=c++17 -o main main.cpp
> ./main
Hello World, 王大明
> 
```

Thanks for listening.

元智大學 | C++ 程式設計實習

Wei-Yuan Chang