

程式設計與實習(二)

BY 孫茂勛

EMAIL:JOHN85051232@GMAIL.COM

一些雜七雜八的事情

- 下週5/9(二)要協助進行高中生APCS檢測試做，屆時會算一次小考分數，請記得來上課
- 本次上課是最後一次晚上的加強班課程
- 想看自己期中考成績的下課來找我
- 明天5/4(四)8:45AM法院L516有一場「物聯網、行動應用、雲端服務與大數據分析之關鍵技術」種子教師培訓研討會...有空的同學可以來參加...參加的簽名屆時可以加分

已經沒有什麼可以教你們了！>>

議程在這裡->

會議簡章	
時間	地點
08:00-08:50	報名、開會
09:00-09:30	第一場：專題演講
09:30-10:30	第二場：專題演講
10:30-11:30	第三場：專題演講
11:30-12:00	第四場：專題演講
12:00-13:00	午餐休息
13:00-14:00	第五場：專題演講
14:00-15:00	第六場：專題演講
15:00-16:00	第七場：專題演講
16:00-17:00	散會

一些雜七雜八的事情

- 5/23(二)CPE考試(5/9開始報名)，期末E平台會開上傳讓各位把題數截圖上傳做為加分的依據

Q1：想寫一個程式，同時擁有Stack和Queue該怎麼寫？

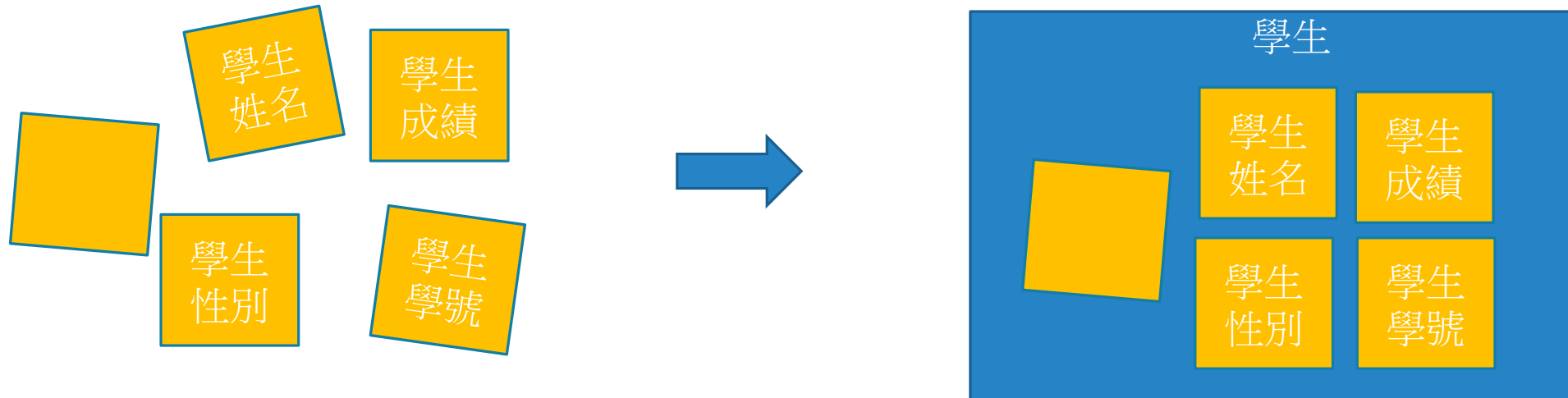
A：把stack和queue的struct、function都寫在一起

Q2：stack和queue都有push()、pop()，電腦怎麼知道你要push/pop的是stack和queue？

OOP

物件導向程式設計(Object-Oriented Programming)

- 大二上學期會教
- 把物件(Object)都組裝再一起的概念
- 現在可以先想成把許多變數用一個更大的變數包裝再一起



OOP

OOP(Object Oriented Programming)

- 把東西都想像成一個物件
- 物件具有自己的資料以及行為
- 把資料跟行為包起來就叫做封裝 (Encapsulation)
- Ex：學生的Object



OOP

Ex：把人類當成物件，該具有身高、年齡的資料，並且人類可以進行跑步、吃飯、睡覺的動作

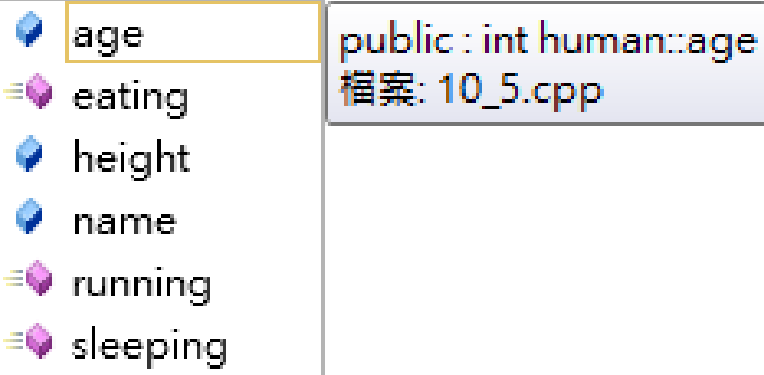
```
struct human
{
    int height;
    int age;
    char *name;

    void running()
    {
        printf("running\n");
    }
    void eating()
    {
        printf("eating\n");
    }
    void sleeping()
    {
        printf("sleeping\n");
    }
};
```

OOP

在main中用new宣告一個human的指標變數
透過->運算子可以執行對應的操作

```
int main()
{
    human *john = new john;
    john->
    return
}
```



age
eating
height
name
running
sleeping

public: int human::age
檔案: 10_5.cpp

OOP

把上禮拜的Stack跟Queue封裝看看

```
struct stack
{
    int num;
    stack *next;

    void push(stack *head, int data) { ... }
    void pop(stack *head) { ... }
    void print(stack *head) { ... }
    void top(stack *head) { ... }
    void size(stack *head) { ... }
    void empty(stack *head) { ... }
};
```

```
struct queue
{
    char num;
    queue *next;

    void push(queue *head, char data) { ... }
    queue *pop(queue *head) { ... }
    void print(queue *head) { ... }
    void front(queue *head) { ... }
    void size(queue *head) { ... }
    void empty(queue *head) { ... }
};
```

OOP

此時stack指標下可以存取struct內的資料以及函式

```
stack *head = new stack;  
head->next = NULL;  
head->
```

<pre>while({ pr</pre>	<ul style="list-style-type: none">emptynextnumpopprintpushsizetop	<pre>public : void stack::empty(stack *head) 檔案: 10_5.cpp</pre> <pre>=====</pre>
--------------------------------	--	--

OOP

然後改一下在main裡面的寫法

```
switch(n)
{
    case 0:head->print(head);break;
    case 1:
        printf("要新增的資料:");
        scanf("%d",&num);
        head->push(head,num);
        break;
    case 2:head->pop(head);break;
    case 3:head->top(head);break;
    case 4:head->size(head);break;
    case 5:head->empty(head);break;
}
```

```
switch(n)
{
    case 0:head->print(head);break;
    case 1:
        printf("要新增的資料:");
        scanf("%d",&num);
        head->push(head,num);
        break;
    case 2:head = head->pop(head);break;
    case 3:head->front(head);break;
    case 4:head->size(head);break;
    case 5:head->empty(head);break;
}
```

OOP

封裝 (Encapsulation)

- 把東西整合起來較不會混淆，使得撰寫程式觀念更清晰
- stack跟queue都有相同名稱的函式push()，如果兩個資料結構寫在同一個程式碼裡面會因為函式名稱重複而不易閱讀

就像這樣

```
int main()
{
    stack *a = new stack;
    queue *b = new queue;
    a->push(a);//執行stack a的push
    b->push(b);//執行stack q的push
    return 0;
}
```

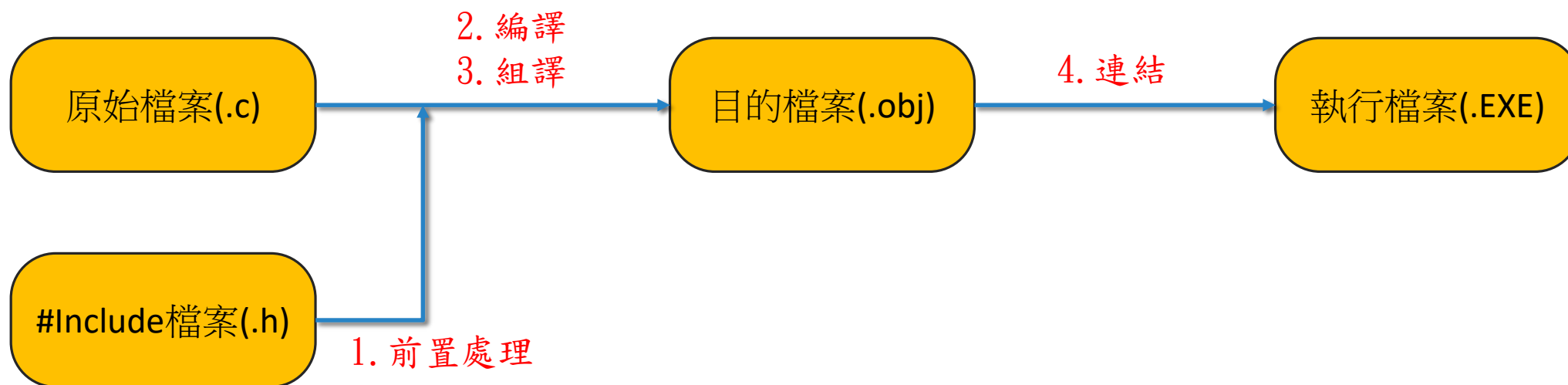
前置處理器

Q：編譯是什麼？

- 將程式原始碼透過編譯器(Compiler)轉成機械碼(二進位碼)讓電腦執行的過程

前置處理器

編譯一個程式的流程



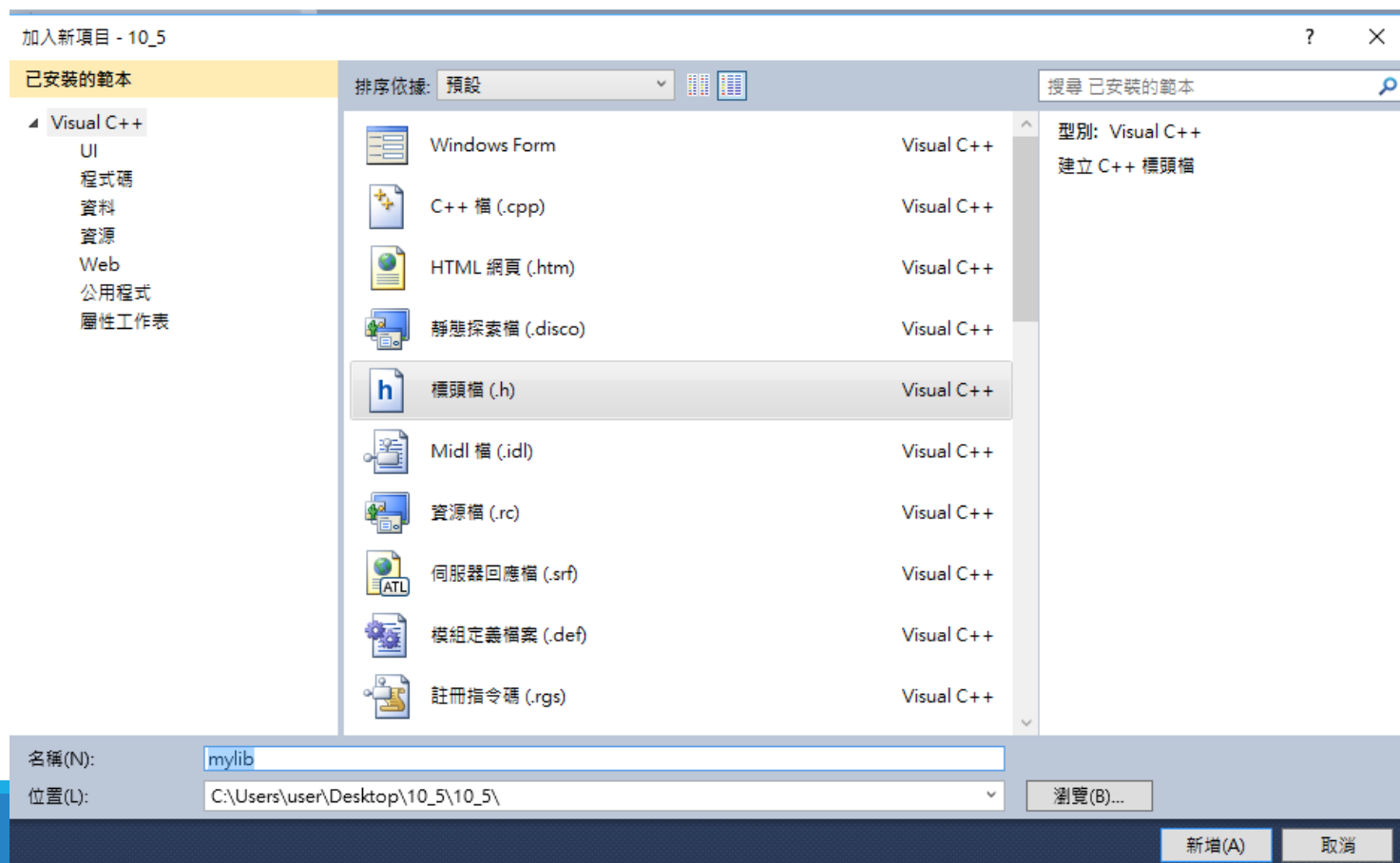
前置處理器

- Ex : `#include <stdio.h>` 、 `#include <stdlib.h>`
- 這些(.h)檔案叫做標頭檔
- 提供許多函式讓include該標頭檔的人使用



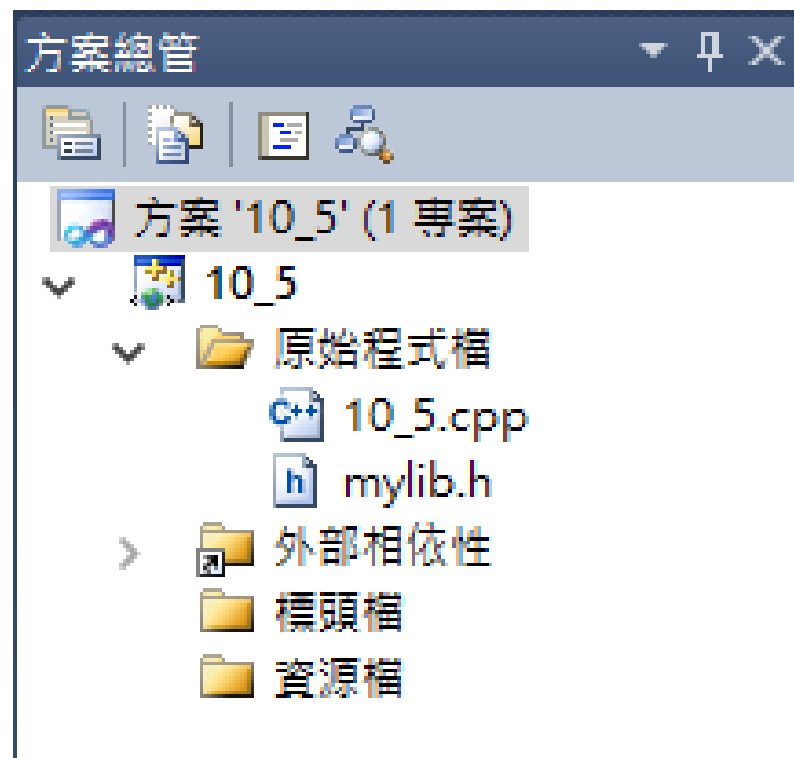
自己寫個標頭檔

- 原始程式檔點右鍵->加入->新增項目



自己寫個標頭檔

- 專案裡面多了一個(.h)的檔案



自己寫個標頭檔

- 在標頭檔裡面寫一個function

```
#include <stdio.h>
#include <stdlib.h>
void print_helloworld()
{
    printf("hello world");
}
```

自己寫個標頭檔

- 在原本檔案include標頭檔

```
#include "mylib.h"//自定義的標頭檔用""引入
int main()
{
    print_helloworld();
    return 0;
}
```

自己寫個標頭檔

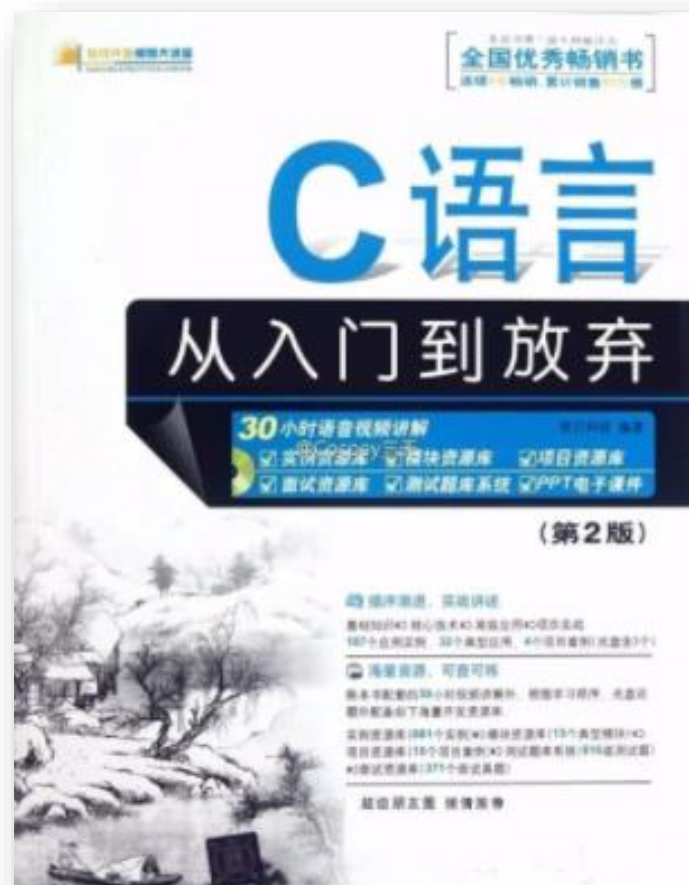
Q：為什麼main不用再include<stdio.h>和<stdlib.h>？

A：在標頭檔中已經做過引入了，重複的引入相同標頭檔可能會造成問題

自己寫個標頭檔

- 查看一下專案的資料夾，多了一個mylib.h
- 之後只要將該檔案傳給其他人，並讓他們include該標頭檔案，別人就可以使用你所撰寫的程式碼

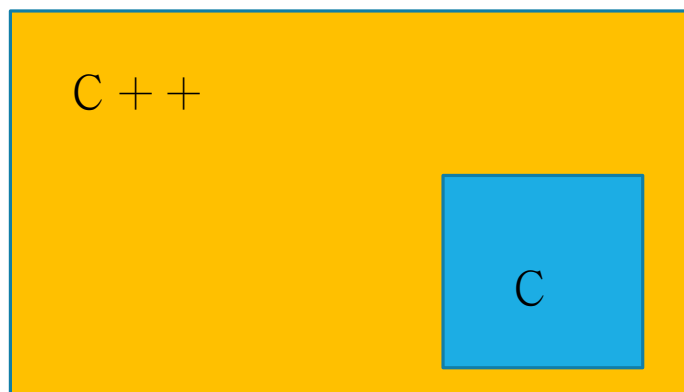
淺談C++



C++是什麼？

C 語言的超集合

- 也就是說，在C裡面的相關語法、函式都可以使用在C++



C與C++的差別

C的程序導向(procedure-oriented)/C++的物件導向(OOP)

C : procedure-oriented

- 以功能(function)為主要的核心概念

C++ : OOP

- 以物件為主要的核心概念
- 提供了class(與struct觀念相同)來撰寫物件

C與C++的差別

I/O的不同

- C : printf/scanf
- C++ : cin/cout

C與C++的差別

之前學過的動態宣告

- C : malloc/free
- C++ : new/delete

所以我們之前學的是C++的寫法(比較好寫)

I/O with C++

每個程式語言都有一個HELLO WORLD

```
#include <iostream>
#include <cstdio>
#include <cstdlib>
```

include的檔案不同了
<iostream>就是C++的I/O library

```
int main()
{
    int a = 0;
    std::cin >> a;
    std::cout << "hello world" << std::endl;
    return 0;
}
```

I/O with C++

輸入

- `std::cin;`
- 箭頭方向為 `>>`

輸出

- `std::cout;`
- 箭頭方向為 `<<`

換行

- `std::endl;`

```
#include <iostream>
#include <cstdio>
#include <cstdlib>
```

include的檔案不同了
`<iostream>`就是C++的I/O library

```
int main()
{
    int a = 0;
    std::cin >> a;
    std::cout << "hello world" << std::endl;
    return 0;
}
```

I/O with C++

省略std::

- using namespace std;

```
#include <iostream>
#include <cstdio>
#include <cstdlib>
using namespace std;
int main()
{
    int a = 0;
    cin >> a;
    cout << "hello world" << endl;
    return 0;
}
```

I/O with C++

一次輸入/輸出多個變數

- 不用%d、%c、%f，C++會自動判定變數的型態

```
using namespace std;
int main()
{
    int a = 0;
    char b = 0;
    float c = 0;
    cout << "按造順序輸入整數、字元、浮點數" << endl;
    cin >> a >> b >> c;
    cout << " a is " << a << endl << " b is " << b << endl << " c is " << c << endl;
    return 0;
}
```

YA！上完啦

給想多練一些程式的人

自學菜單

- 資料結構 (tree 、 graph)
- 基礎演算法(DFS 、 BFS 、 shortest path problem)
- OOP
- CPE一顆星48題 ★ ★ ★