



Polymorphism

Week 10



Yang-Cheng Chang
Yuan-Ze University
yczhang@saturn.yzu.edu.tw



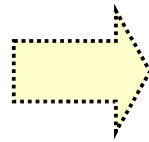
隱藏實作細節的做法

■ 優點

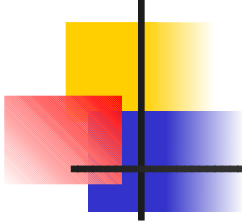
- 編譯時可以減少因為修改而牽動的檔案數量，大幅降低檔案重新編譯的規模
- 在提供別人編譯所需要的檔案時，既可以成功讓人編譯
 - 但不會曝露實作細節的原始碼，而破壞類別的封裝性
 - 或被修改到實作的程式碼，破壞了類別的整體概念性

範例 1 - num.h

```
1 class num
2 {
3     public:
4         num();
5         ~num();
6         void set (int param);
7         int get (void) const;
8         void addTwo (void);
9     private:
10         int n;
11 };
```



```
1 class num
2 {
3     public:
4         num();
5         ~num();
6         void set (int param);
7         int get (void) const;
8         void addTwo (void);
9     private:
10         class implementation;
11         implementation * impl;
12 };
```



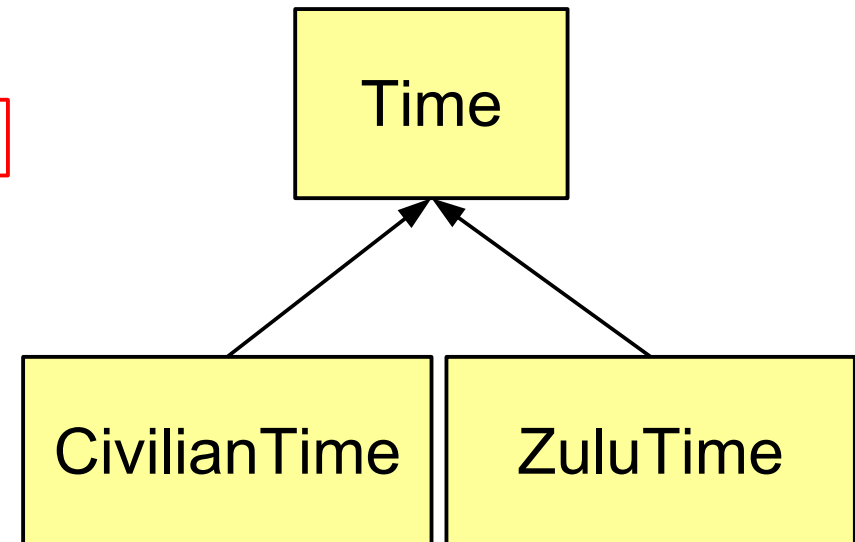
範例 1 - num.cpp

```
1 #include "num.h"
2
3 // implementation class
4 class num::implementation
5 {
6     public:
7         int n;
8     public:
9         void addOne (void);
10 };
11
12 void num::implementation::addOne (void)
13 {
14     ++n;
15 }
```

```
1 // normal class
2 num::num()
3 {
4     impl = new implementation;
5 }
6
7 num::~~num()
8 {
9     delete impl;
10 }
11
12 void num::set (int param)
13 {
14     impl->n = param;
15 }
16
17 int num::get (void) const
18 {
19     return (impl->n);
20 }
21
22 void num::addTwo (void)
23 {
24     impl->addOne();
25     impl->addOne();
26 }
```

範例 2 Time.h

```
1 // Hide Implementation in C++
2 #ifndef TIME_H
3 #define TIME_H
4
5 #include <iostream>
6 #include <iomanip>
7 #include <string>
8
9 using namespace std;
10
11 class TimeImp;
12
13 class Time {
14     public:
15         Time(){}
16         Time(int hr, int min);
17         virtual void tell();
18     protected:
19         TimeImp *impl;
20 };
21
22 class CivilianTime: public Time {
23     public:
24         CivilianTime(int hr, int min, int pm);
25 };
26
27 class ZuluTime: public Time {
28     public:
29         ZuluTime(int hr, int min, int zone);
30 };
31
32 #endif // TIME_H
```





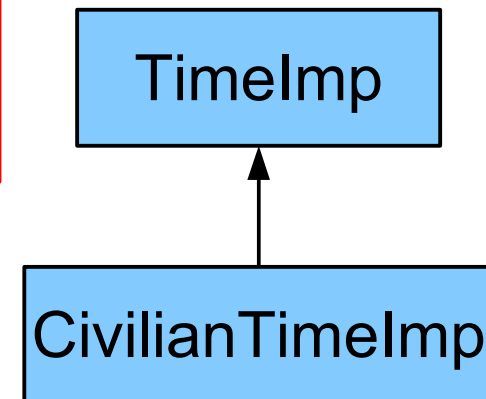
範例 2 Time.cpp

```
1 #include "Time.h"
2
3 // Implementation Class
4 class TimeImp {
5     public:
6         TimeImp(int hr, int min);
7         virtual void tell();
8     protected:
9         int hour, minite;
10 };
11
12 TimeImp::TimeImp(int hr, int min) {
13     hour = hr;
14     minite = min;
15 }
16
17 void TimeImp::tell()
18 {
19     cout << "time is " << setw(2) << hour << minite << endl;
20 }
21
```

所有的 Imp 類別
不會放在 Time.h
因為不希望將其
實作曝露給使用
者

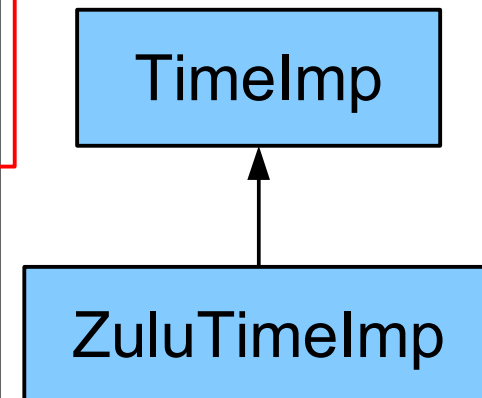
範例 2 Time.cpp

```
1 // Civilian
2 class CivilianTimeImp: public TimeImp {
3     public:
4         CivilianTimeImp(int hr, int min, int pm);
5         void tell();
6     protected:
7         string whichM;
8 };
9
10 CivilianTimeImp::CivilianTimeImp(int hr, int min, int pm): TimeImp(hr, min)
11 {     if (pm)
12         whichM=" PM";
13     else
14         whichM=" AM";
15 }
16
17 void CivilianTimeImp::tell() {
18     cout << "time is " << hour << ":" << minite << whichM << endl;
19 }
20
```



範例 2 Time.cpp

```
1 // Zulu
2 class ZuluTimeImp: public TimeImp {
3     public:
4         ZuluTimeImp(int hr, int min, int zone);
5         void tell();
6     protected:
7         string zone;
8 };
9
10 ZuluTimeImp::ZuluTimeImp(int hr, int min, int zn): TimeImp(hr, min) {
11     if (zn == 5)
12         zone=" Eastern Standard Time";
13     else if (zn == 6)
14         zone=" Central Standard Time";
15 }
16
17 void ZuluTimeImp::tell() {
18     cout << "time is " << setw(2) << hour << minite << zone << endl;
19 }
20
```





範例 2 Time.cpp

```
1 // Normal Class
2 Time::Time(int hr, int min) {
3     impl = new TimeImp(hr, min);
4 }
5
6 void Time::tell() {
7     impl->tell();
8 }
9
10 CivilianTime::CivilianTime(int hr, int min, int pm) {
11     impl = new CivilianTimeImp(hr, min, pm);
12 }
13
14 ZuluTime::ZuluTime(int hr, int min, int zone) {
15     impl = new ZuluTimeImp(hr, min, zone);
16 }
17
```

impl 的型別是
TimeImpl，這
是多型的應用



注意事項

- 由於 impl 是採用指標與動態記憶體配置的方式來實作
- 實務上需要實作自己的
 - 拷貝建構子 (copy constructor)
 - 給值運算子 (assignment operator)



Assignment 10 – Part 1

- 請參考主程式的流程與物件宣告，完成三個類別的實作
 - 一個基礎類別 pet
 - 需要有建構子，用來初始化成員變數 name，設定寵物名字
 - 一個成員變數 name，用於儲存寵物的名字
 - 兩個成員函式 Name()，makeSound()
Name() 傳回寵物的名字
makeSound() 傳回寵物的叫聲，用字串代表
 - 兩個衍生類別 cat，dog，繼承於基礎類別 pet
 - cat，dog 兩個類別的 makeSound() 必須不同，cat 的 makeSound() 傳回 miau，dog 的 makeSound() 傳回 won



主程式

```
#include <iostream>
#include "pet.h"

using namespace std;

void examinePet(pet* p){
    cout << "My name is " << p->Name() << " and I make " << p->makeSound() << endl;
}

int main(){
    pet* marry = new cat("marry");
    pet* tom = new dog("tom");
    examinePet(marry); // output: My name is marry and I make miau
    examinePet(tom); // output: My name is tom and I make won
}
```



Assignment 10 – Part 2

- 將程式修改為實作與介面分離的做法
- 建立一個函數庫專案，名稱為 `libpet`，將 `pet.h` 與 `pet.cpp` 放入專案，產生 `libpet.lib` 函式庫
- 建立另一個專案，將 `main.cpp` 放入專案，設定方案屬性，引入 `pet.h` 及 `libpet.lib`，產生使用 `libpet` 的程式

Assignment 10 – Part 2

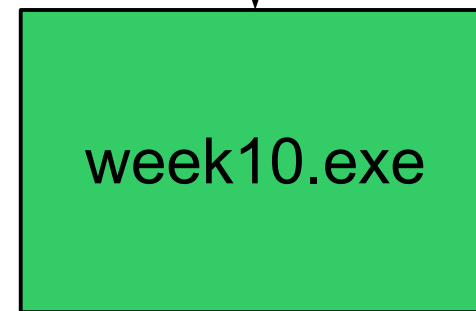
C++ 函式庫專案

libpet



C++ 一般專案

week10



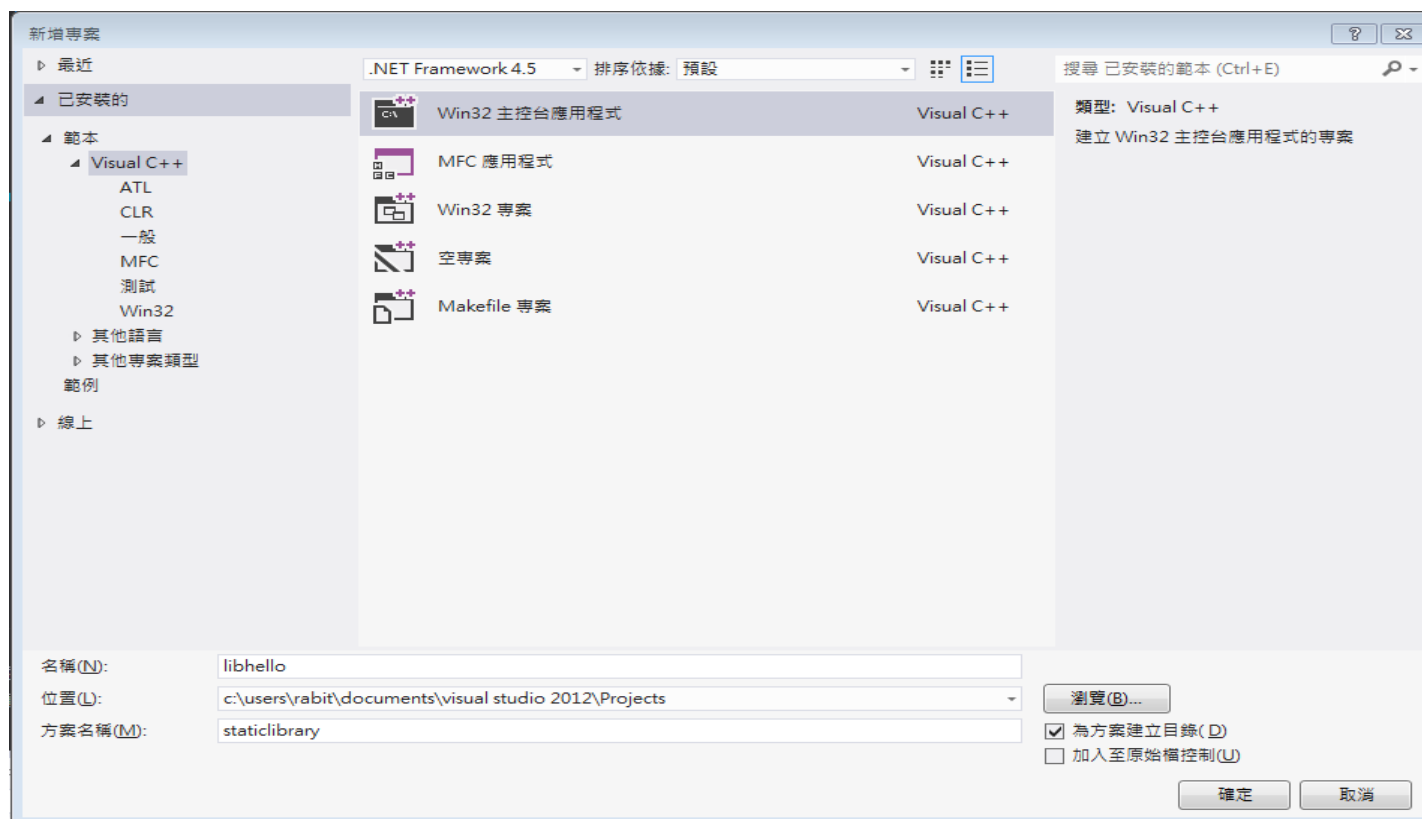


函式庫的散佈型式

- 原始碼 + 標頭檔
 - 檔案內容：.cpp .h
- 靜態函式庫 + 標頭檔
 - 檔案內容：.lib .h (windows)
.a .h (unix)
- 動態函式庫 + 標頭檔
 - 檔案內容：.dll .h (windows)
.so .h (unix)

如何建靜態立函式庫 (1)

- 新增專案
 - 選擇『Win32 主控台應用程式』
 - 名稱: libhello
 - 方案名稱: staticlibrary



如何建靜態立函式庫 (2)

- Win32 應用程式精靈
 - 應用程式類型：靜態函式庫
 - 其他選項：取消『先行編譯標頭檔』





如何建靜態立函式庫 (3)

- 加入標頭檔 : `hello.h`
- 加入原始程式檔 : `hello.cpp`

`hello.h`

```
#ifndef HELLO_H
#define HELLO_H
#include <iostream>
using namespace std;
void sayHello();
#endif
```

`hello.cpp`

```
#include "hello.h"

void sayHello()
{
    cout << "hello!" << endl;
}
```

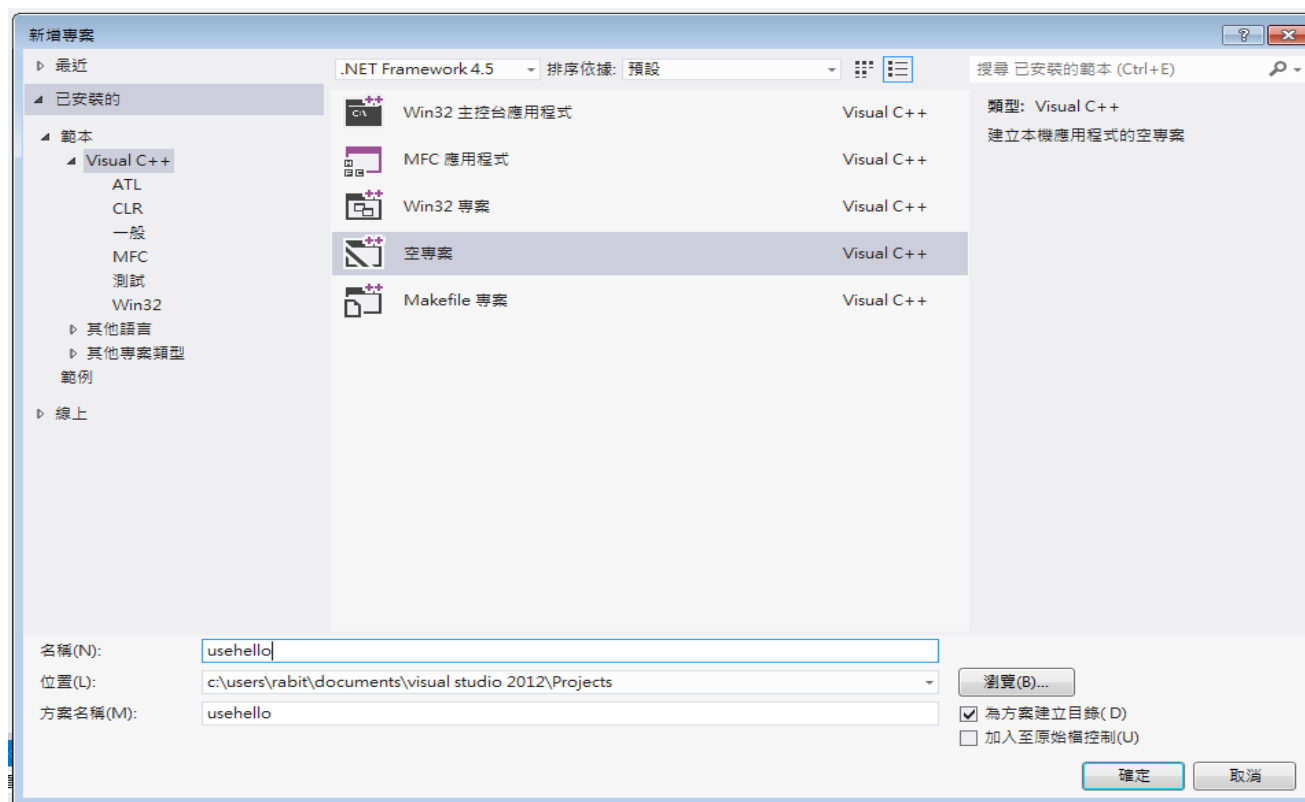


如何建靜態立函式庫 (4)

- 提供給其他程式使用時所需要的檔案
 - 標頭檔 `hello.h`
路徑：`staticlibrary\libhello`
 - 靜態函式庫 `libhello.lib`
路徑：`staticlibrary\Debug`

如何使用靜態函式庫 (1)

- 新增專案
 - 選擇『空專案』
 - 名稱: usehello
 - 方案名稱: usehello





如何使用靜態函式庫 (2)

- 加入原始程式檔：usehello.cpp

```
#include <hello.h>

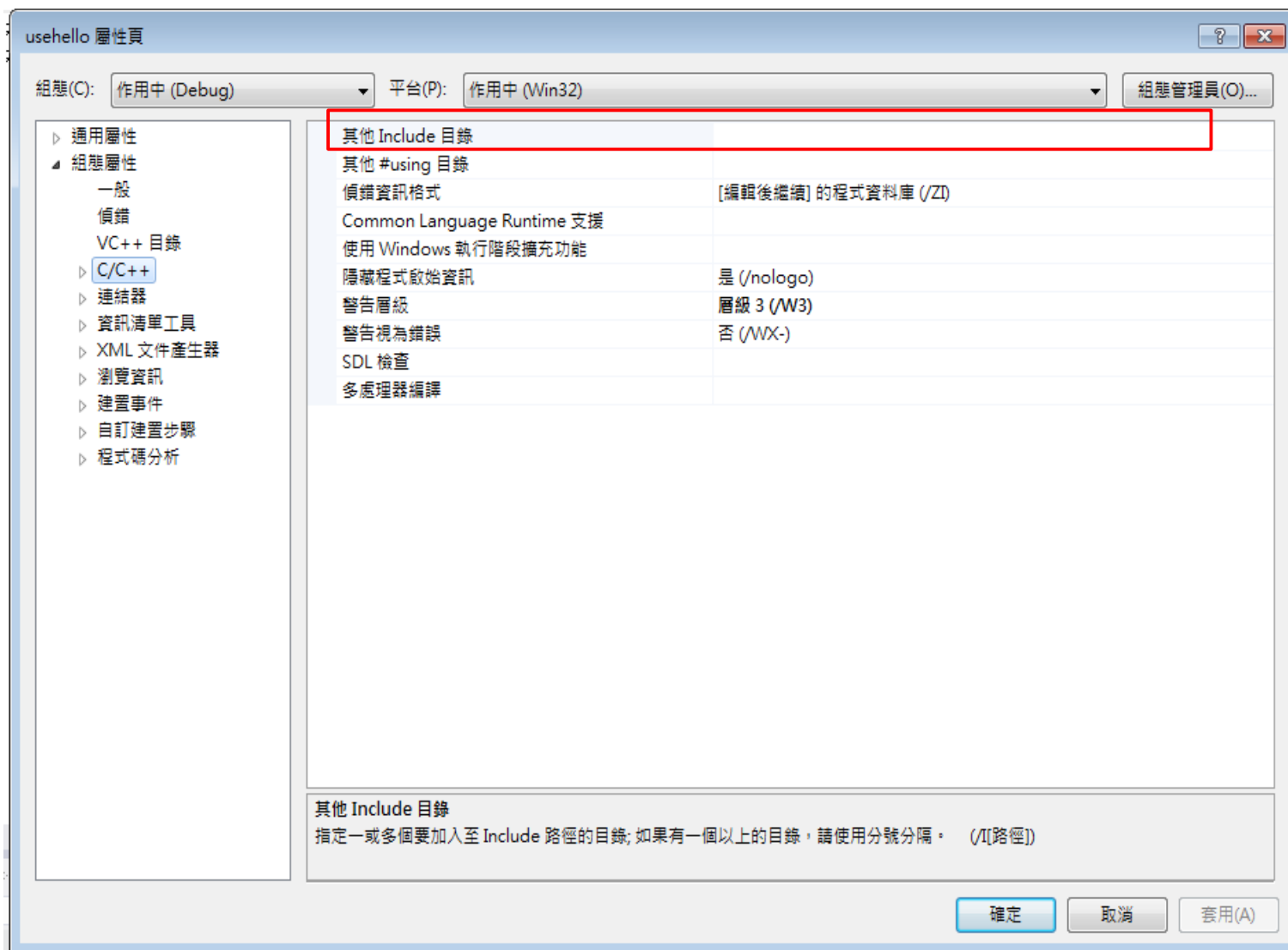
int main()
{
    sayHello();
    system("pause");
    return 0;
}
```



如何使用靜態函式庫 (3)

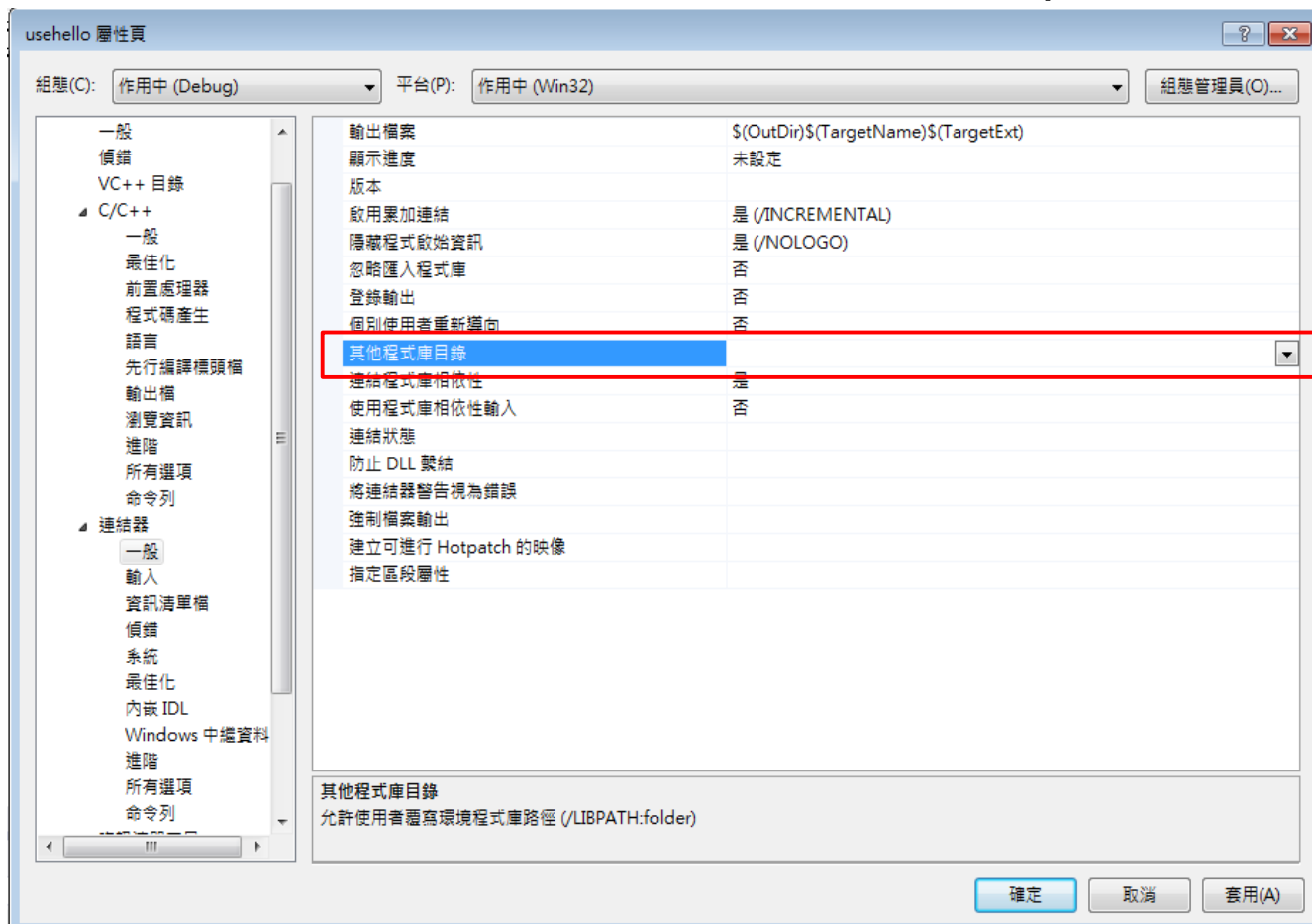
- 加入函式庫標頭檔 (hello.h) 所在的路徑
 - 方案總管 → 在 usehello 按右鍵
 - 選擇最下方的選項：屬性
 - 選擇『組態屬性 > C/C++ > 一般』
 - 在其他 Include 目錄加入 hello.h 所在的路徑

如何使用靜態函式庫 (3)



如何使用靜態函式庫 (4)

- 選擇『組態屬性 > 連結器 > 一般』
- 在其他程式目錄加入靜態函式庫 (libhello.lib) 路徑



如何使用靜態函式庫 (5)

- 選擇『組態屬性 > 連結器 > 輸入』
- 在其他相依性加入靜態函式庫名稱 (libhello.lib)

