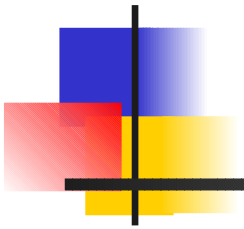
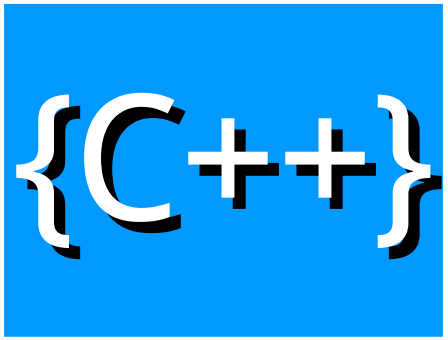


Polymorphism

Week 10



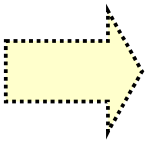
Yang-Cheng Chang
Yuan-Ze University
yczhang@saturn.yzu.edu.tw

隱藏實作細節的做法

- 優點
  - 編譯時可以減少因爲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

```
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 };
```

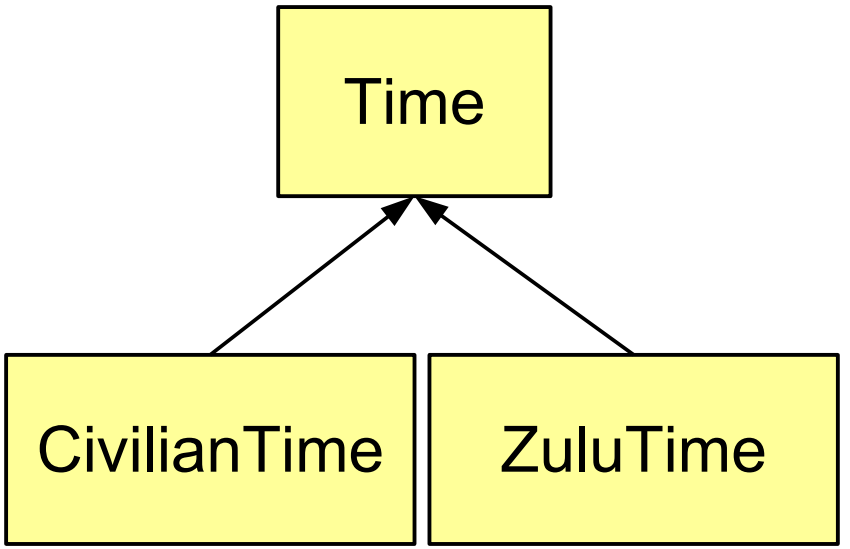
範例 1

```
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

```cpp
#include "Time.h"

// Implementation Class
class TimeImp {
    public:
        TimeImp(int hr, int min);
        virtual void tell();
    protected:
        int hour, minite;
};

TimeImp::TimeImp(int hr, int min) {
        hour = hr;
        minite = min;
}

void TimeImp::tell()
{
        cout << "time is " << setw(2) << hour << minite << endl;
}

```

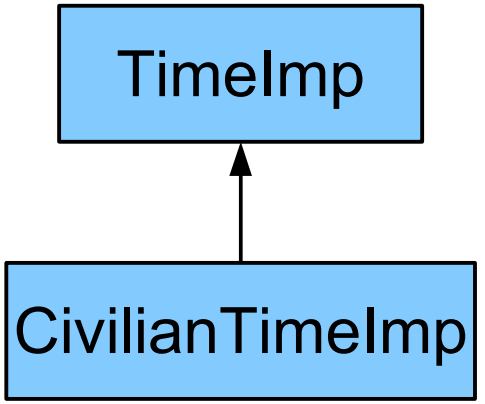
範例 2 Time.cpp

```cpp
// Civilian
class CivilianTimeImp: public TimeImp {
    public:
        CivilianTimeImp(int hr, int min, int pm);
        void tell();
    protected:
        string whichM;
};

CivilianTimeImp::CivilianTimeImp(int hr, int min, int pm): TimeImp(hr, min)
{   if (pm)
      whichM=" PM";
    else
      whichM=" AM";
}

void CivilianTimeImp::tell() {
    cout << "time is " << hour << ":" << minite << whichM << endl;
}

```



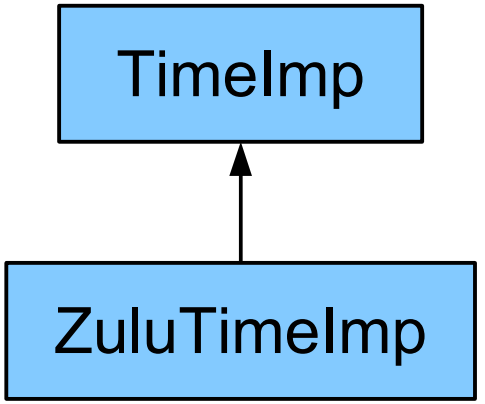
範例 2 Time.cpp

```cpp
// Zulu
class ZuluTimeImp: public TimeImp {
    public:
        ZuluTimeImp(int hr, int min, int zone);
        void tell();
    protected:
        string zone;
};

ZuluTimeImp::ZuluTimeImp(int hr, int min, int zn): TimeImp(hr, min) {
    if (zn == 5)
      zone=" Eastern Standard Time";
    else if (zn == 6)
      zone=" Central Standard Time";
}

void ZuluTimeImp::tell() {
    cout << "time is " << setw(2) << hour << minite << zone << endl;
}

```



範例 2 Time.cpp

```cpp
// Normal Class
Time::Time(int hr, int min) {
    impl = new TimeImp(hr, min);
}

void Time::tell() {
    impl->tell();
}

CivilianTime::CivilianTime(int hr, int min, int pm) {
    impl = new CivilianTimeImp(hr, min, pm);
}

ZuluTime::ZuluTime(int hr, int min, int zone) {
    impl = new ZuluTimeImp(hr, min, zone);
}

```

Assignment 10

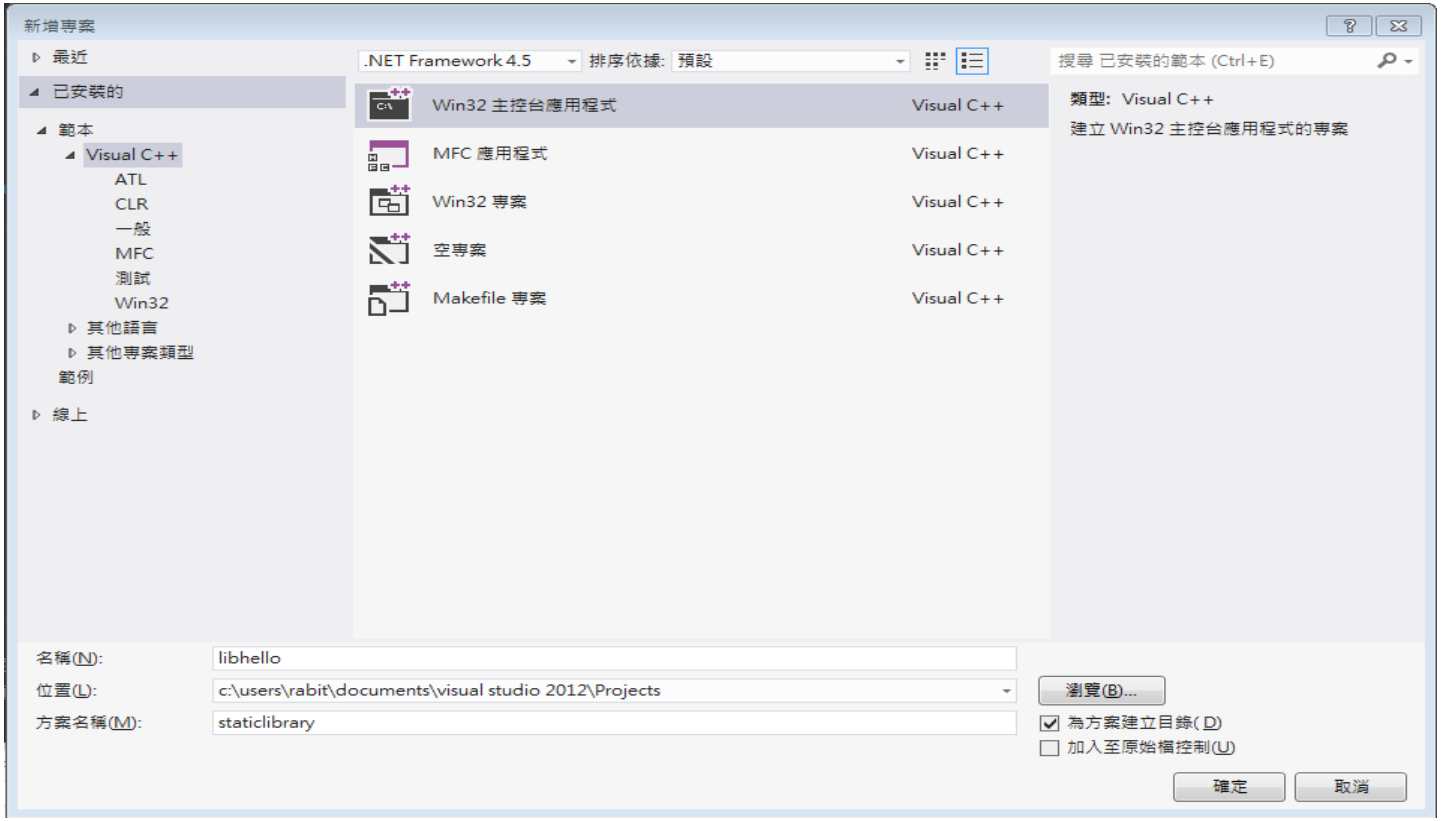
- 請參考 Assignment 9 的題目，將程式修改爲實作與介面分離的做法
- 建立一個函數庫專案，名稱爲 libpet，將 pet.h 與 pet.cpp 放入專案，產生 libpet.lib 函式庫
- 建立另一個專案，將 main.cpp 放入專案，設定方案屬性，引入 pet.h 及 libpet.lib，產生使用 libpet 的程式

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)

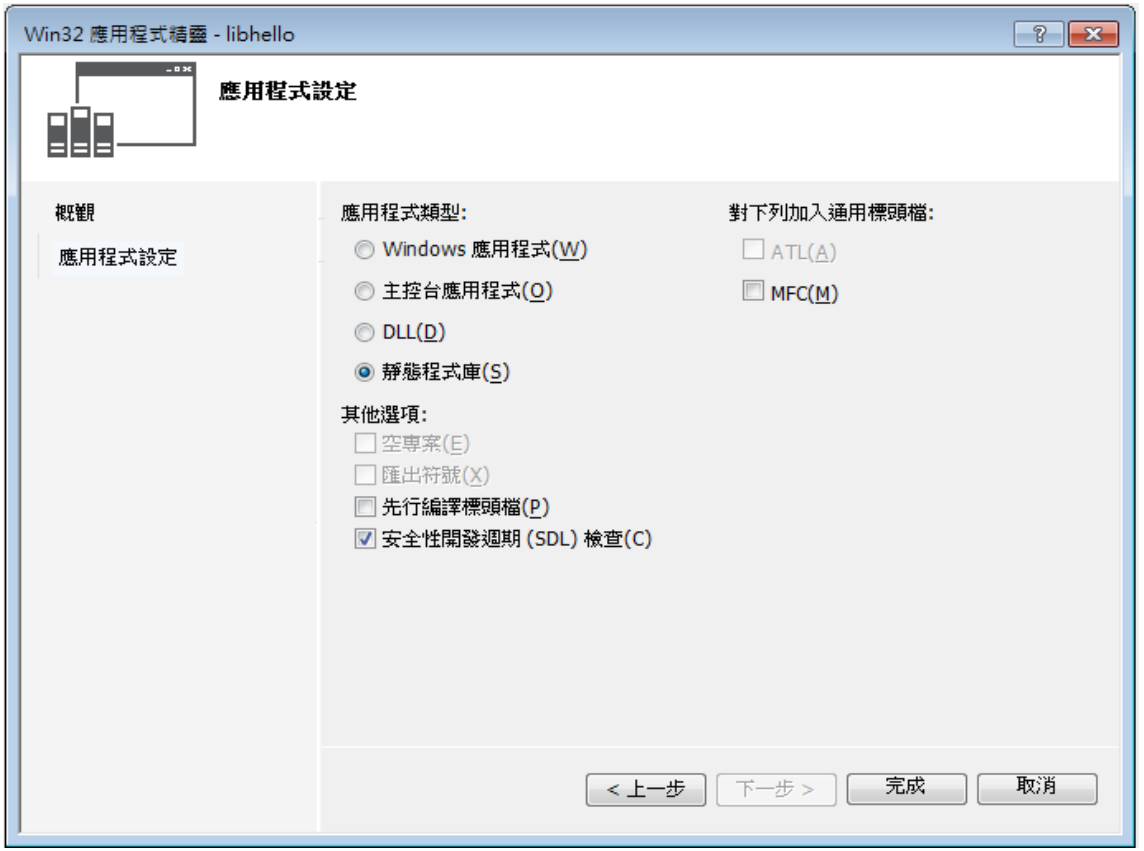
如何建靜態立函式庫 (1)

- 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;
}
```

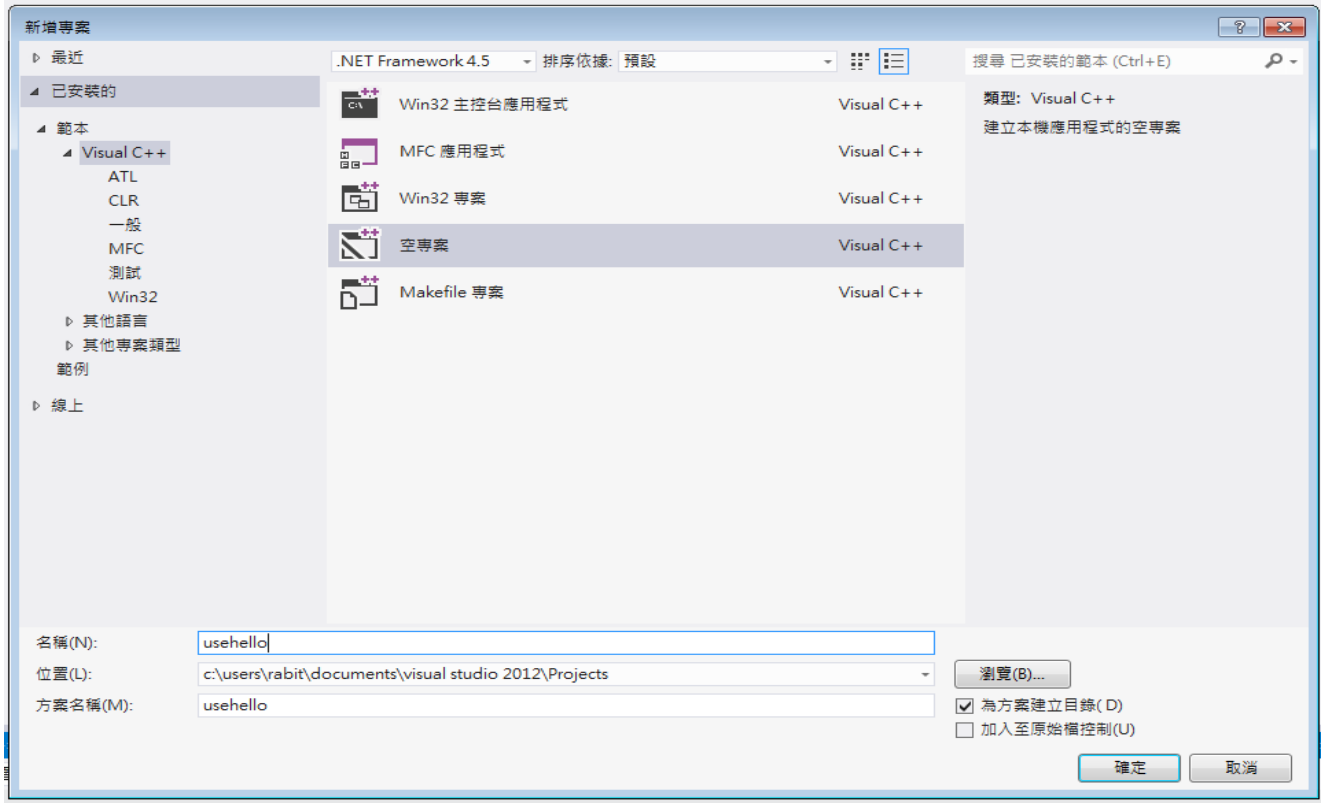
如何建靜態立函式庫 (4)

- 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

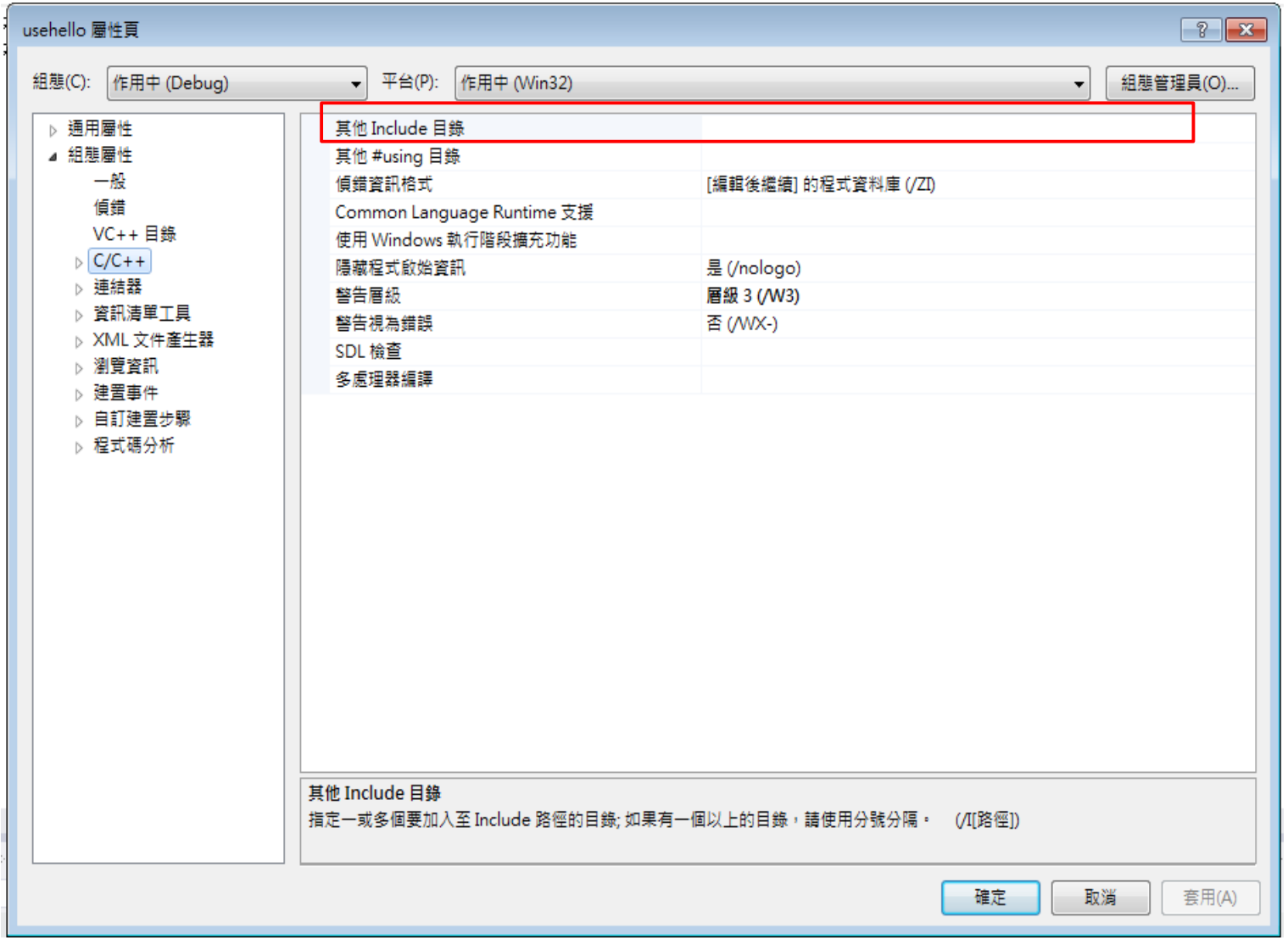
```
#include <hello.h>

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

如何使用靜態函式庫 (3)

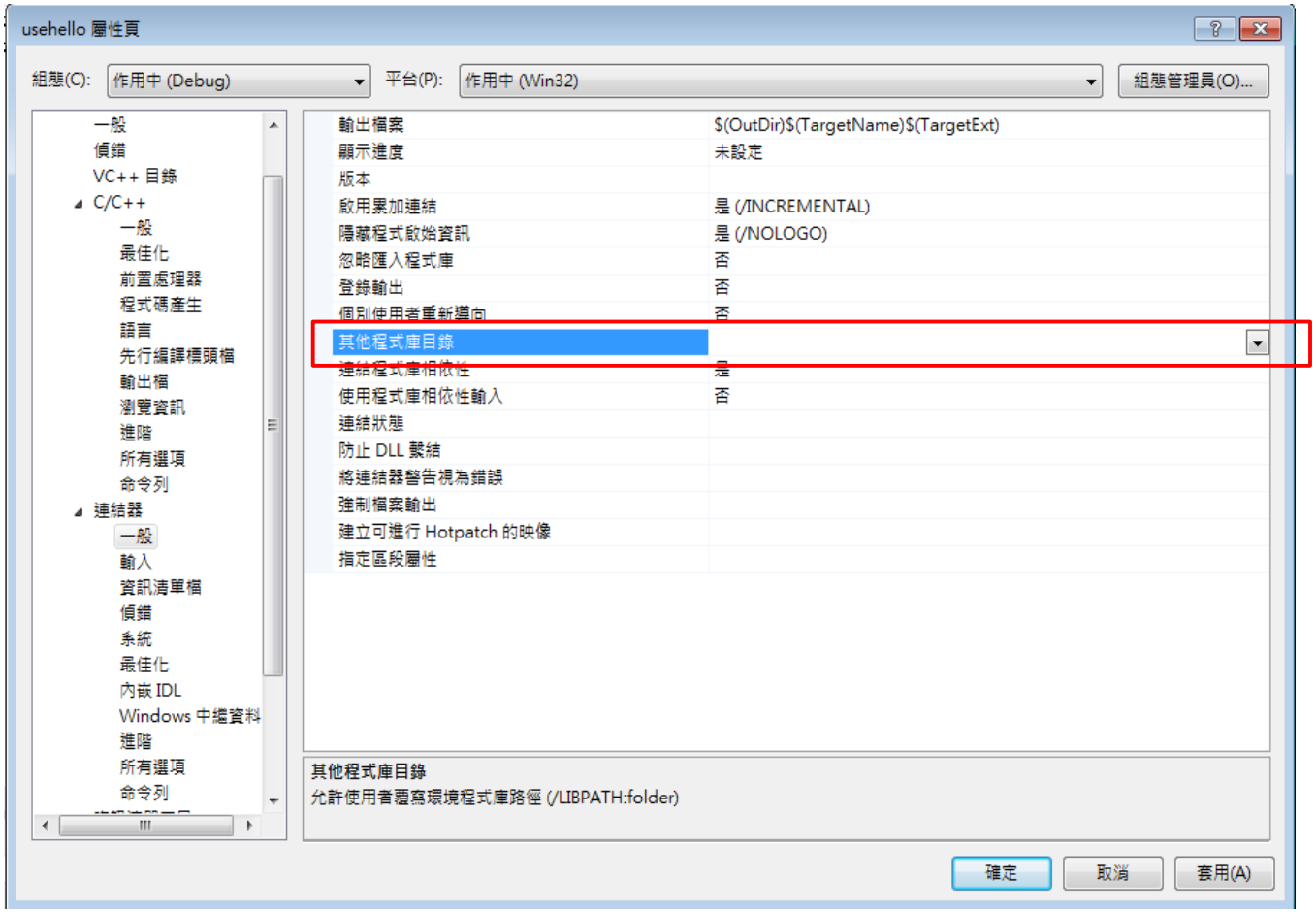
- 加入函式庫標頭檔 (hello.h) 所在的路徑
  - 方案總管 →在 usehello 按右鍵
  - 選擇最下方的選項：屬性
  - 選擇『組態屬性 > C/C++ > 一般』
    - 在其他 Include 目錄加入 hello.h 所在的路徑

如何使用靜態函式庫 (3)


usehello 屬性頁
組態(C): 作用中 (Debug)
平台(P): 作用中 (Win32)
組態管理員(O)...
通用屬性
組態屬性
一般
偵錯
VC++ 目錄
C/C++
連結器
資訊清單工具
XML 文件產生器
瀏覽資訊
建置事件
自訂建置步驟
程式碼分析
其他 Include 目錄
其他 #using 目錄
偵錯資訊格式
[編輯後繼續] 的程式資料庫 (/ZI)
Common Language Runtime 支援
使用 Windows 執行階段擴充功能
隱藏程式啟始資訊
是 (/nologo)
警告層級
層級 3 (/W3)
警告視為錯誤
否 (/WX-)
SDL 檢查
多處理器編譯
其他 Include 目錄
指定一或多個要加入至 Include 路徑的目錄; 如果有一個以上的目錄，請使用分號分隔。 (/I[路徑])
確定
取消
套用(A)

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)