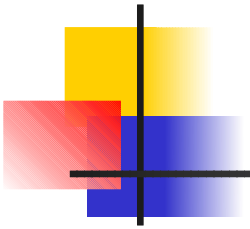
Creating and using Libraries in C++

Week 16

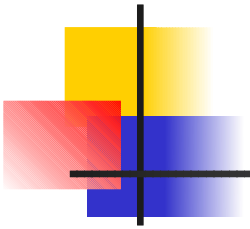
Yang-Cheng Chang
Yuan-Ze University
yczhang@saturn.yzu.edu.tw



Libraries

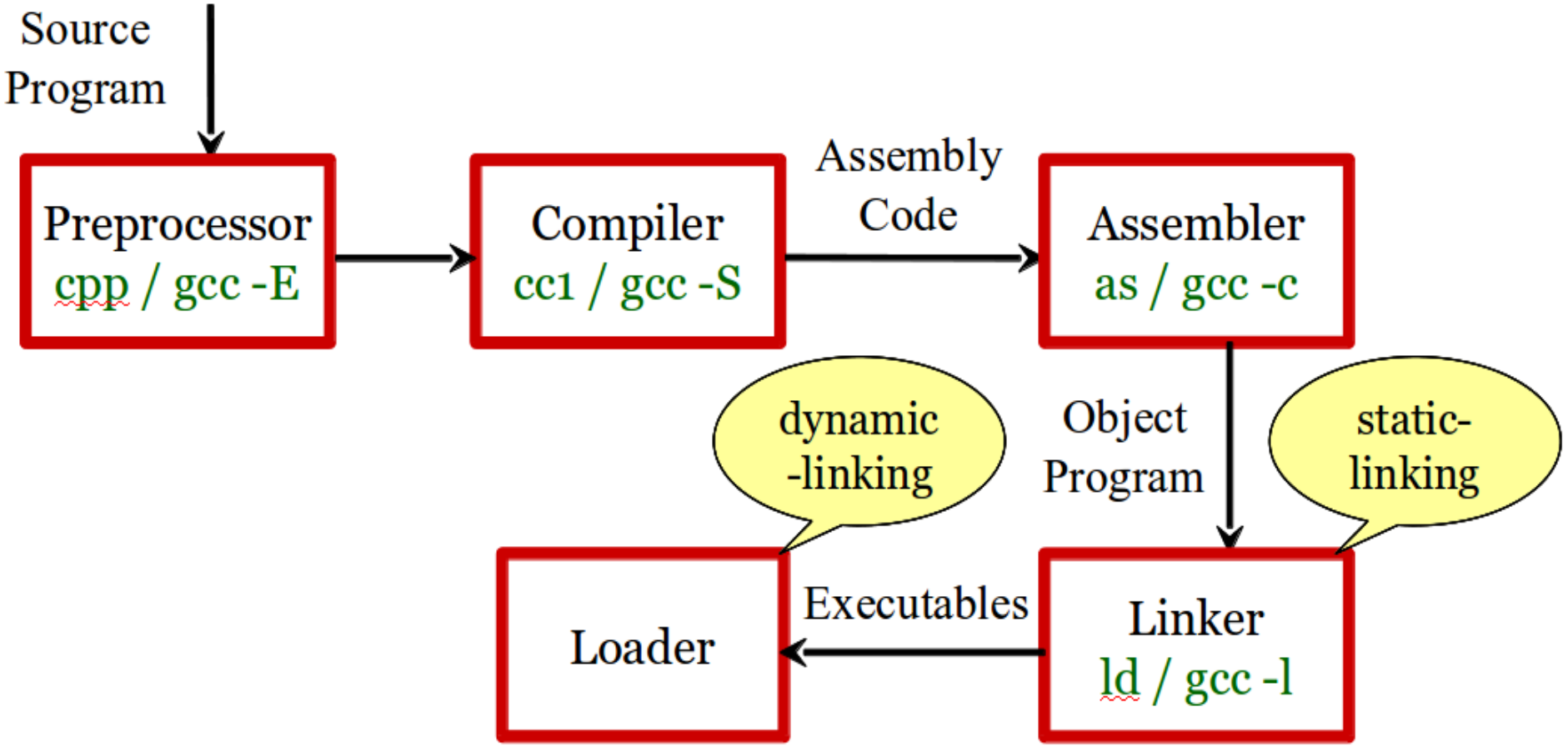
- A library is a collection of subprograms used to develop software.
  - Allows code and data to be reused, shared and changed in a modular fashion.
  - Linking: A linker resolves the references between executables and libraries.

Categories of Libraries (by linking time)



- Static linking libraries
- Dynamic linking libraries
  - Run-Time Environment libraries
  - Programming Language libraries

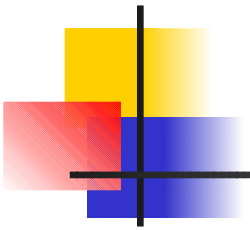
From Source to Execution



Static Linking Libraries

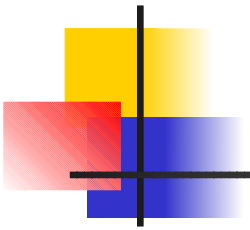
- The code segments will be copy to each executables.
- Pros:
  - Easy to use; no dependency problem after compilation.
- Cons:
  - The executable size will be larger.
  - Require re-linking when libraries changed.

Dynamic Linking Libraries (1/2)



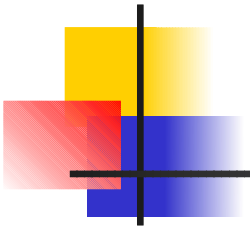
- Allow multiple processes to share the same code segment.
- Pros:
  - Greater flexibility
  - Possible support for plugins.
- Cons:
  - Slow application at start time.
  - Dependent on the libraries when execution.

Dynamic Linking Libraries (2/2)



- The references can be resolved either at:
  - Load-time
  - Run-time
- UNIX Platform
  - “shared-object”: *.so
- Windows Platform
  - “dynamic-linking library”: *.dll

Location of Libraries



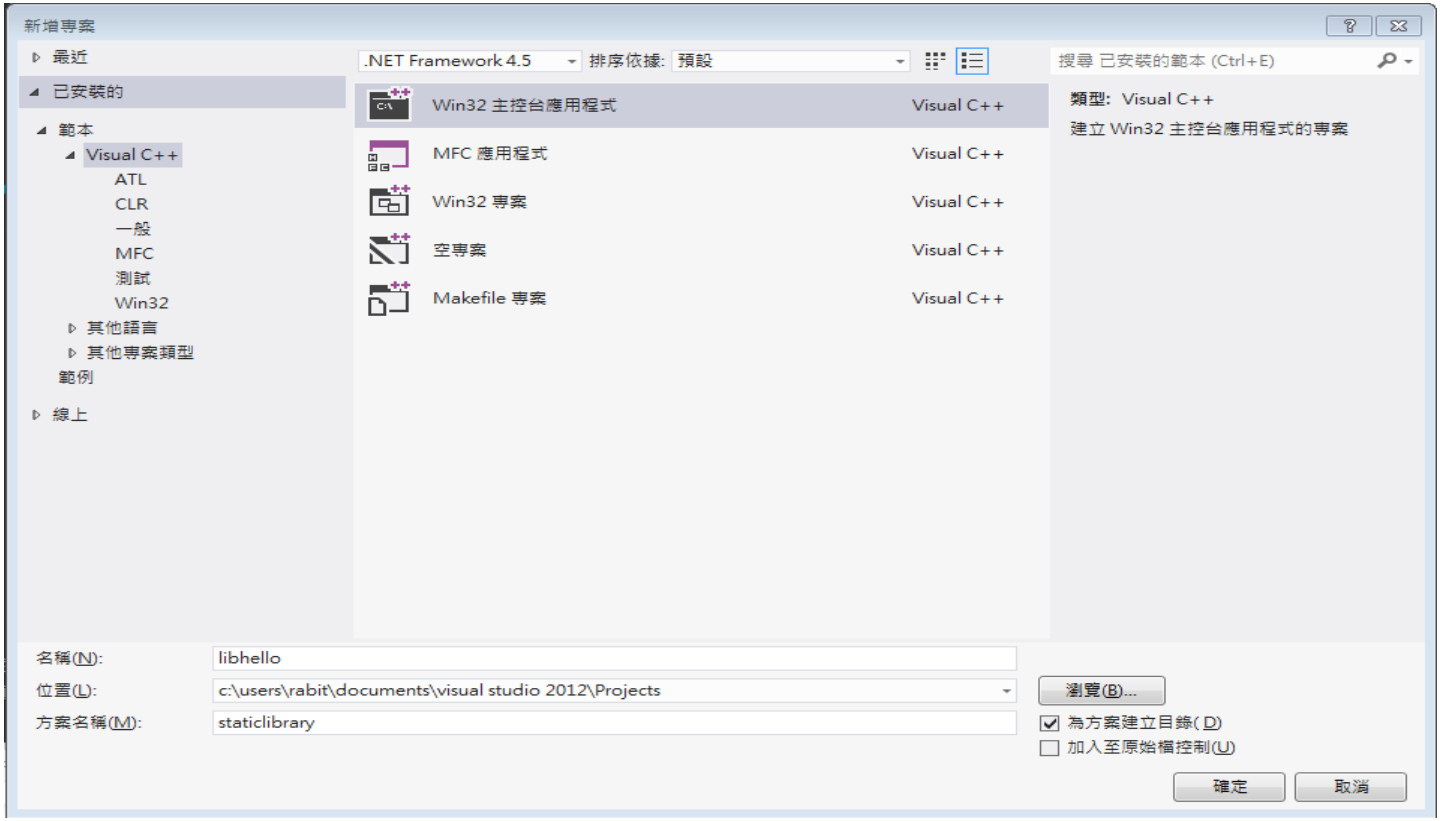
- UNIX Platform
  - /lib: runtime environment libraries
  - /usr/lib: for program development
- Windows Platform
  - C:\WINDOWS\system32\
  - The libraries for program development will be accompanies with compiler, like: Visual C++.

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)

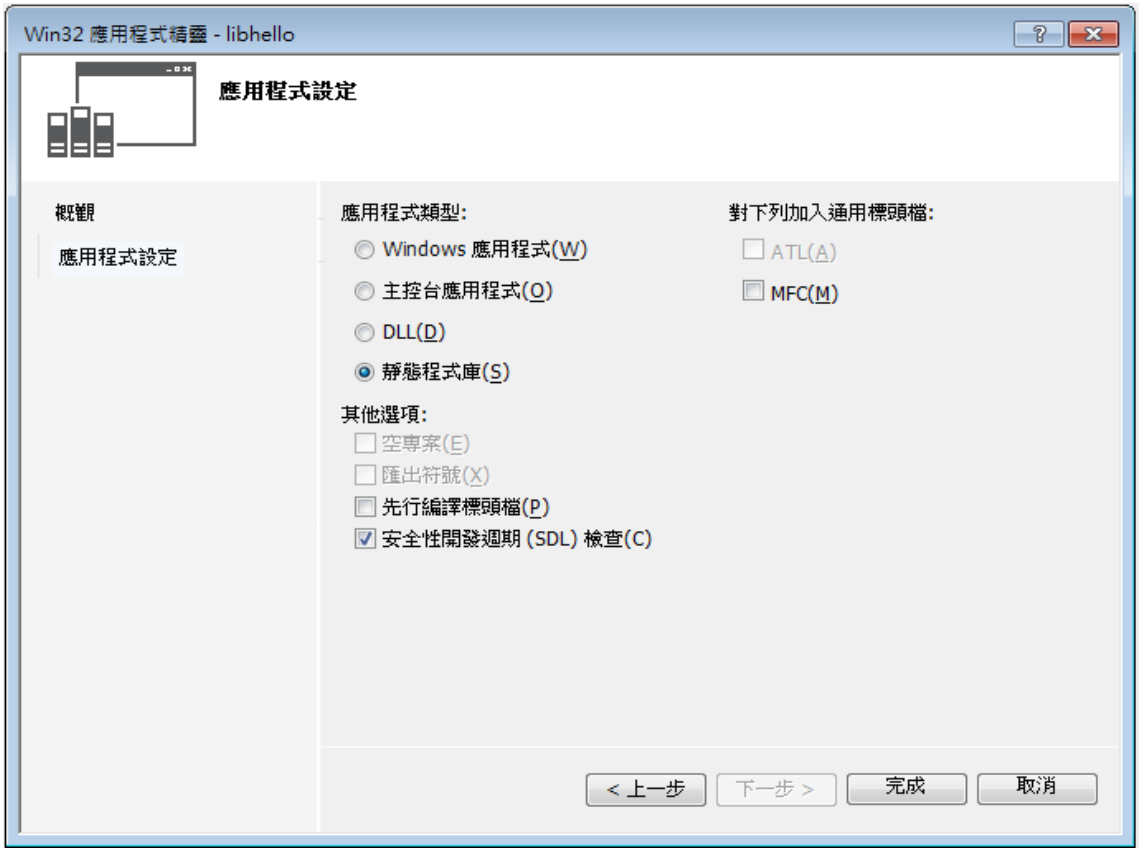
如何建靜態立函式庫 (1)

- 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

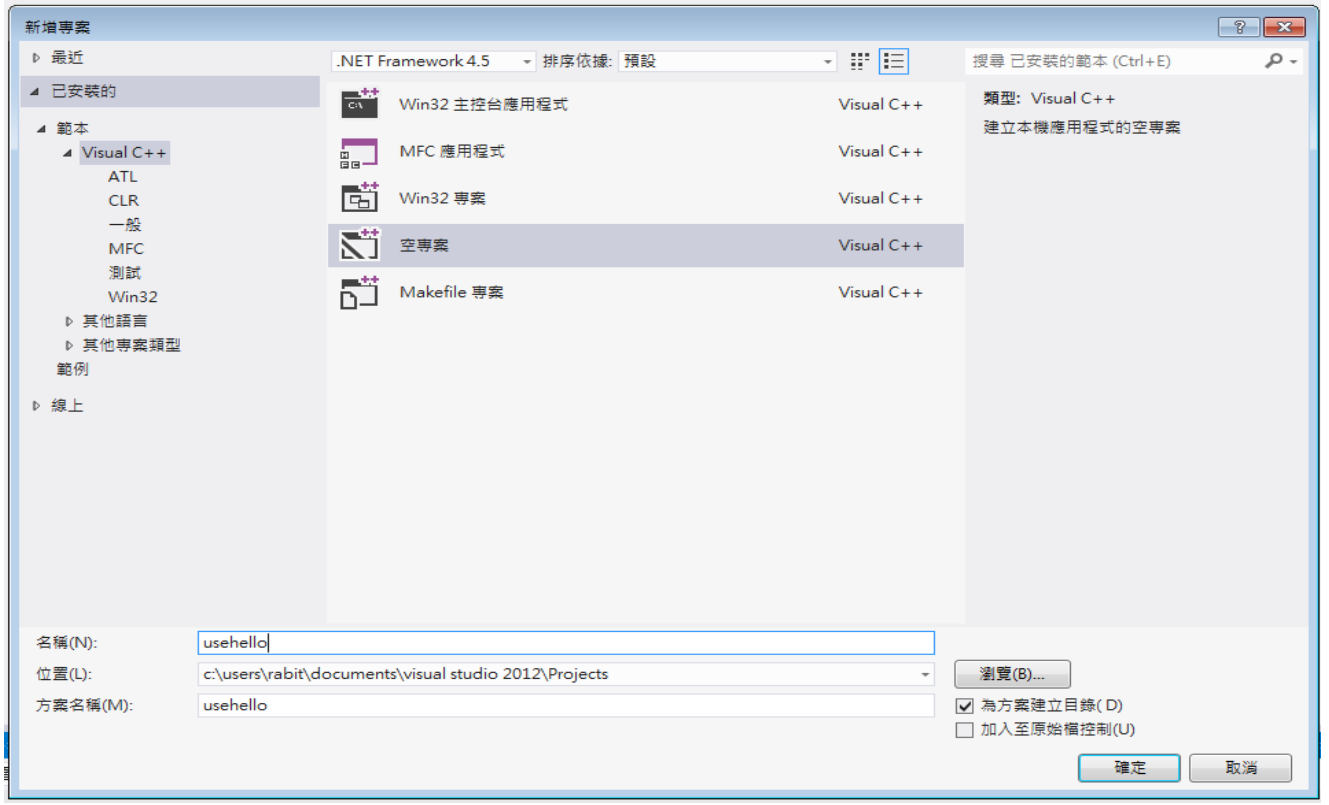
```
#include "hello.h"

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)

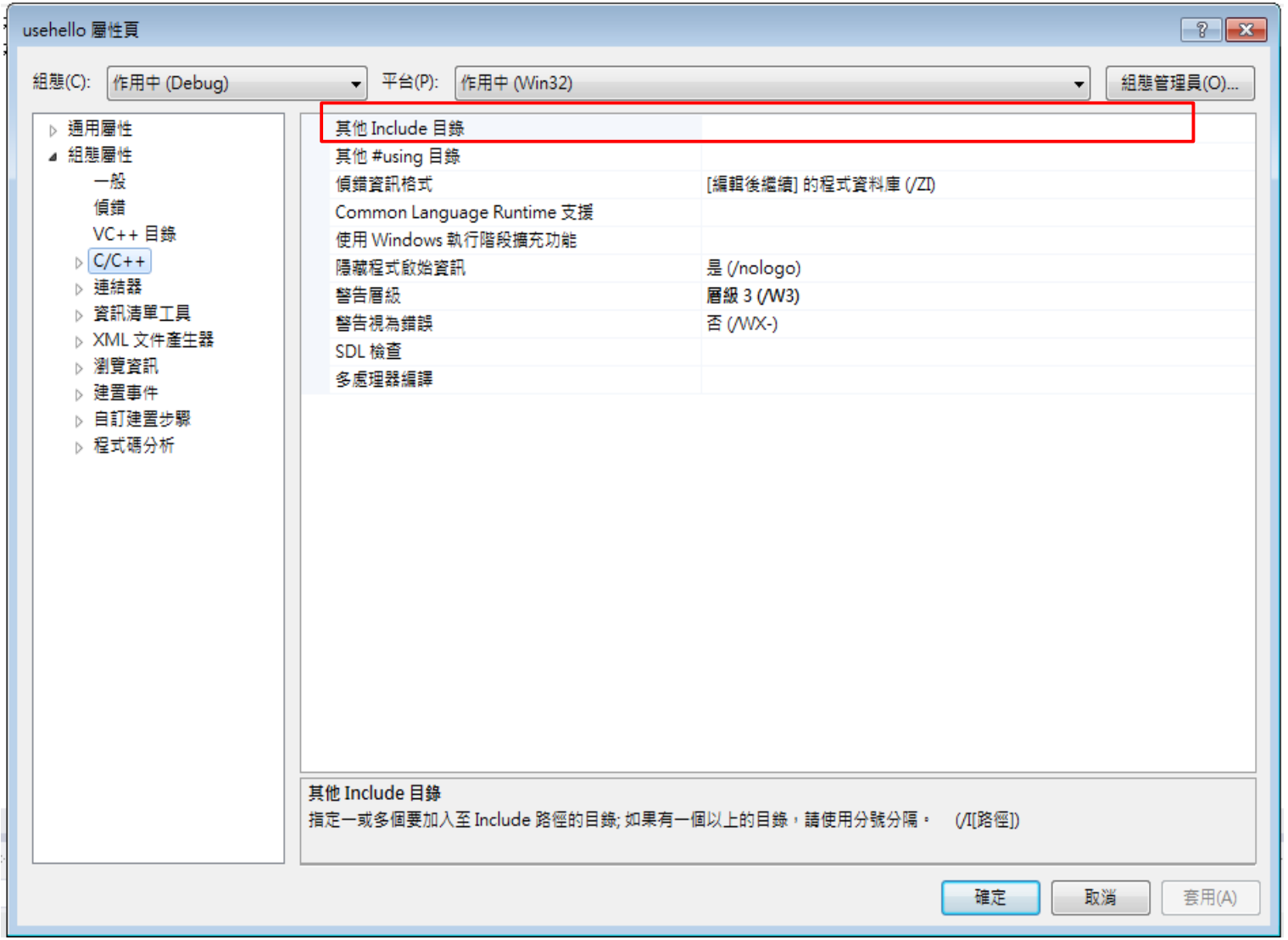
- 加入原始程式檔：usehello.cpp

```
#include <hello.h>

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

如何使用靜態函式庫 (3)

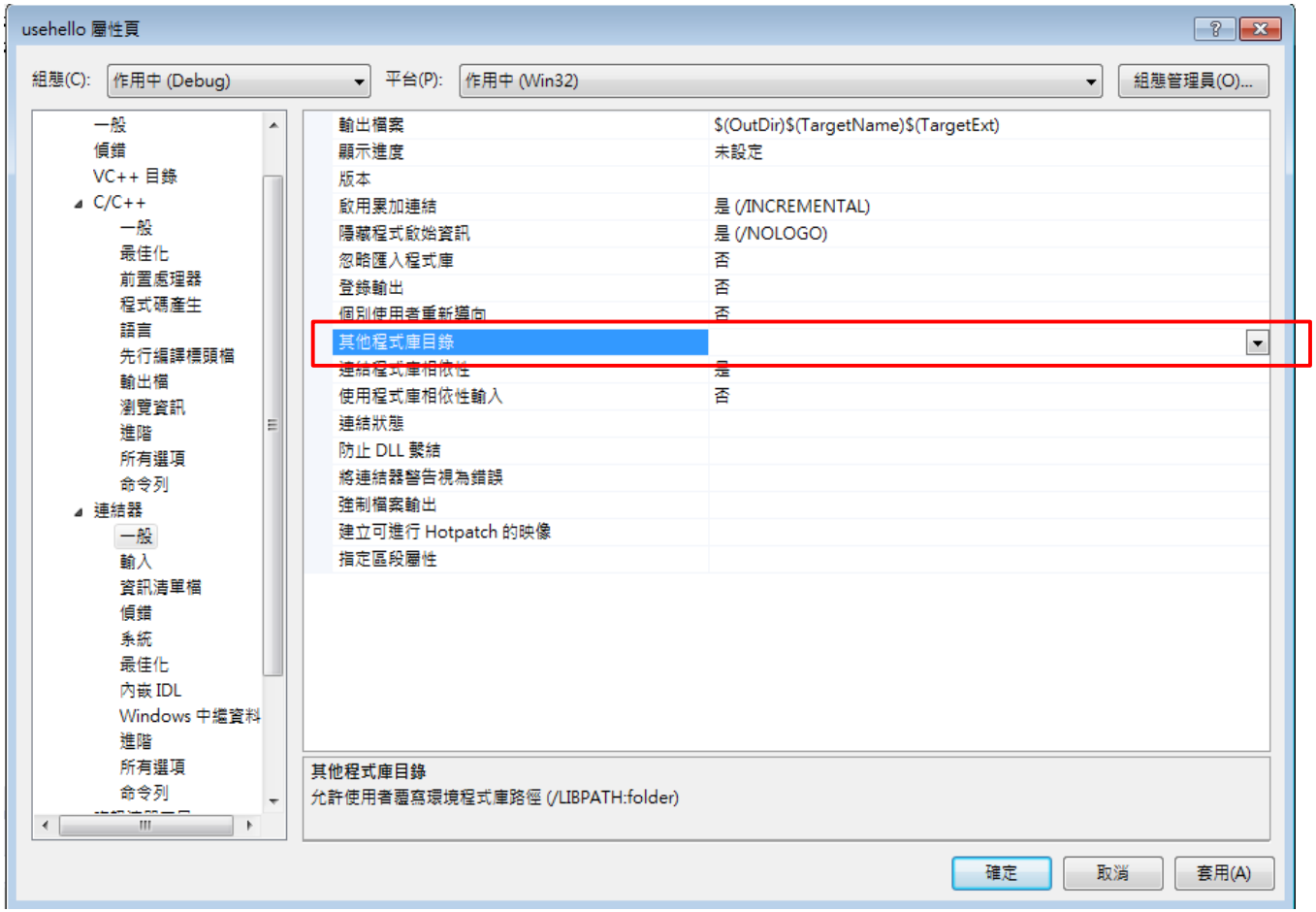
- 加入函式庫標頭檔 (hello.h) 所在的路徑
  - 方案總管 →在 usehello 按右鍵
  - 選擇最下方的選項：屬性
  - 選擇『組態屬性 > C/C++ > 一般』
    - 在其他 Include 目錄加入 hello.h 所在的路徑

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)

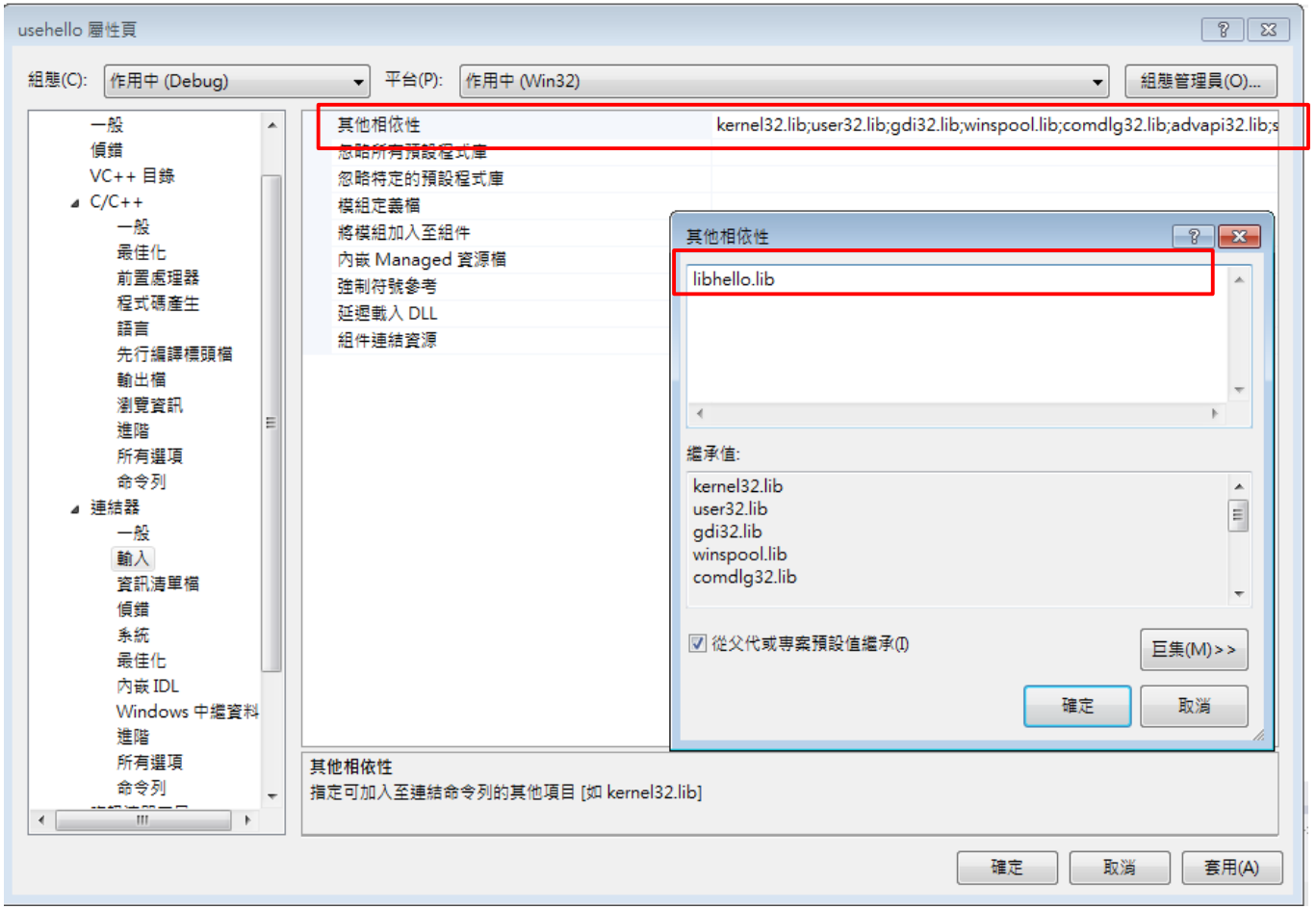
如何使用靜態函式庫 (4)

- 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)

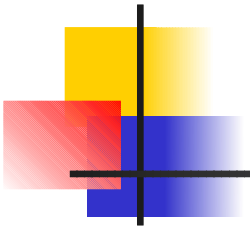


Assignment

- Write a function to count the occurrence of words in a given text file(IHaveADream.txt)
  - Sort the word list by the word occurrence in a descending order, and show the list in following format

```
102  the
 41  and
 38  a
 33  we
```

  - Ignore the following characters ,!."?;- and the given prepostions(preposions.txt)
  - Treat lower case letters the same as upper case letters



Assignment

- You should create a shared library, and implement this function in the library
- You should create a normal project, and write a program to use the function which has been implemented in the shared library