

Chapter 1

Introduction



Yu-Hsiang Cheng
鄭宇翔 (Slighten)
GOTC Instructor

http://m101.nthu.edu.tw/~s101021120/Myweb/index_ch.html



Outline

- Programming ABC
- Our First C++ Program
- Some Practices



Outline

- Programming ABC
- Our First C++ Program
- Some Practices



Programming ABC



What is a Program?

- 一個程式 (*program*) 是一個可執行 (*executable, runnable*) 檔。
- 在 Windows 上，通常會是以副檔名 `.exe` 結尾 (其他系統可能就不會)。
- 程式是由程式碼 (*code*) 經由編譯器 (*compiler*) 編譯 (*compile*) 後產生的。
- 還有一種不用經過編譯就可以執行的程式(碼)，叫作腳本 (*script*)。



What is a Code?

- 程式碼 (code) 是由某一(或多)種程式語言 (*programming language*) 編寫而成。
- 電腦讀不懂你的程式語言，只讀得懂機器語言 (*machine language*)。
- 程式語言就像現在存在於地球的語言一樣，有很多種。
- 程式碼會存成一(或多)個檔案，附檔名通常與該語言有關。
- 例如：
 - C — .c
 - C++ — .cpp
 - JAVA — .java
 - Python — .py
 - HTML — .htm or .html
 - ...



What is a Compiler?

- A compiler compiles your codes to an executable file.
- 編譯 (compile) 就像翻譯，把你寫的程式碼翻成電腦讀得懂的機器碼 (*machine code*)。
- 機器碼只由 0 跟 1 所構成 (2 進位)。
- 電腦只讀得懂機器碼。
- 因此你的程式碼經過編譯器編譯後產生的執行檔就是機器碼。
- 透過執行檔，電腦就可以讀懂你想要幹麻，執行你要它執行的動作。



What is a Machine Code?

1 位元 (1 bit)

1 位元組 (1 byte = 8 bits)

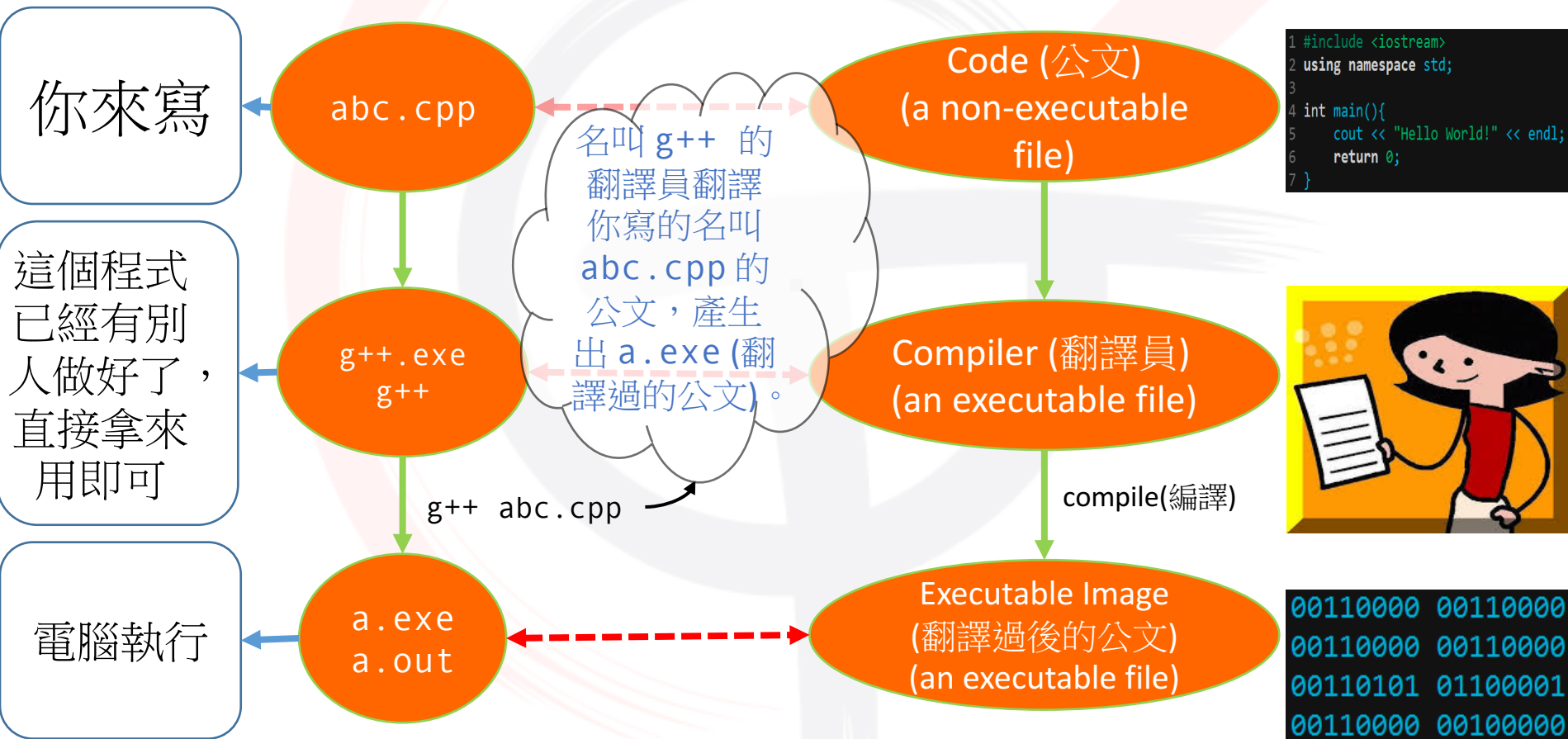
00110000 00110000 00110000 00110000 00110000 00110000
00110000 00110000 00111010 00100000 00110100 01100100
00110101 01100001 00100000 00111001 00110000 00110000
00110000 00100000 00110000 00110011 00110000 00110000
00100000 00110000 00110000 00110000 00110000 00100000
00110000 00110100 00110000 00110000 00100000 00110000
00110000 00110000 00110000 00100000 01100110 01100110
01100110 01100110 00100000 00110000 00110000 00110000
00110000 00100000 00100000 01001101 01011010 00101110
00101110 00101110 00101110 00101110 00101110 00101110
00101110 00101110 00101110 00101110 00101110 00101110

What the Hell? Please Make it Easier

- 想像你自己是老闆，你的助理是翻譯員，而電腦是你的手下。
- 簡單來說，你寫**程式**就像在寫**公文**，內有交辦事項。
- 用某種程式語言寫程式，就像在用某種語言寫公文。
- 編譯器**就像是你身邊的**翻譯員**，將你寫的公文翻譯成你手下讀得懂的機器語言(執行檔)。
- 你再將這篇**翻譯過的公文**給你的手下讀(將這個**執行檔**給電腦執行)，它就會去做裡面的交辦事項。

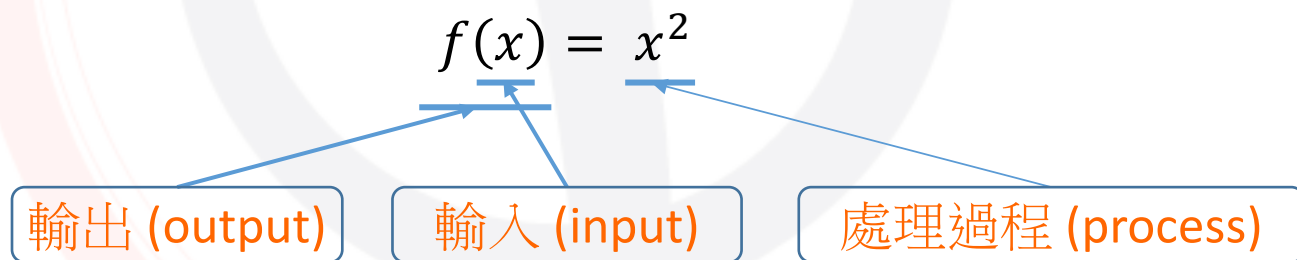


Illustration



Let's Observe a Program in Detail

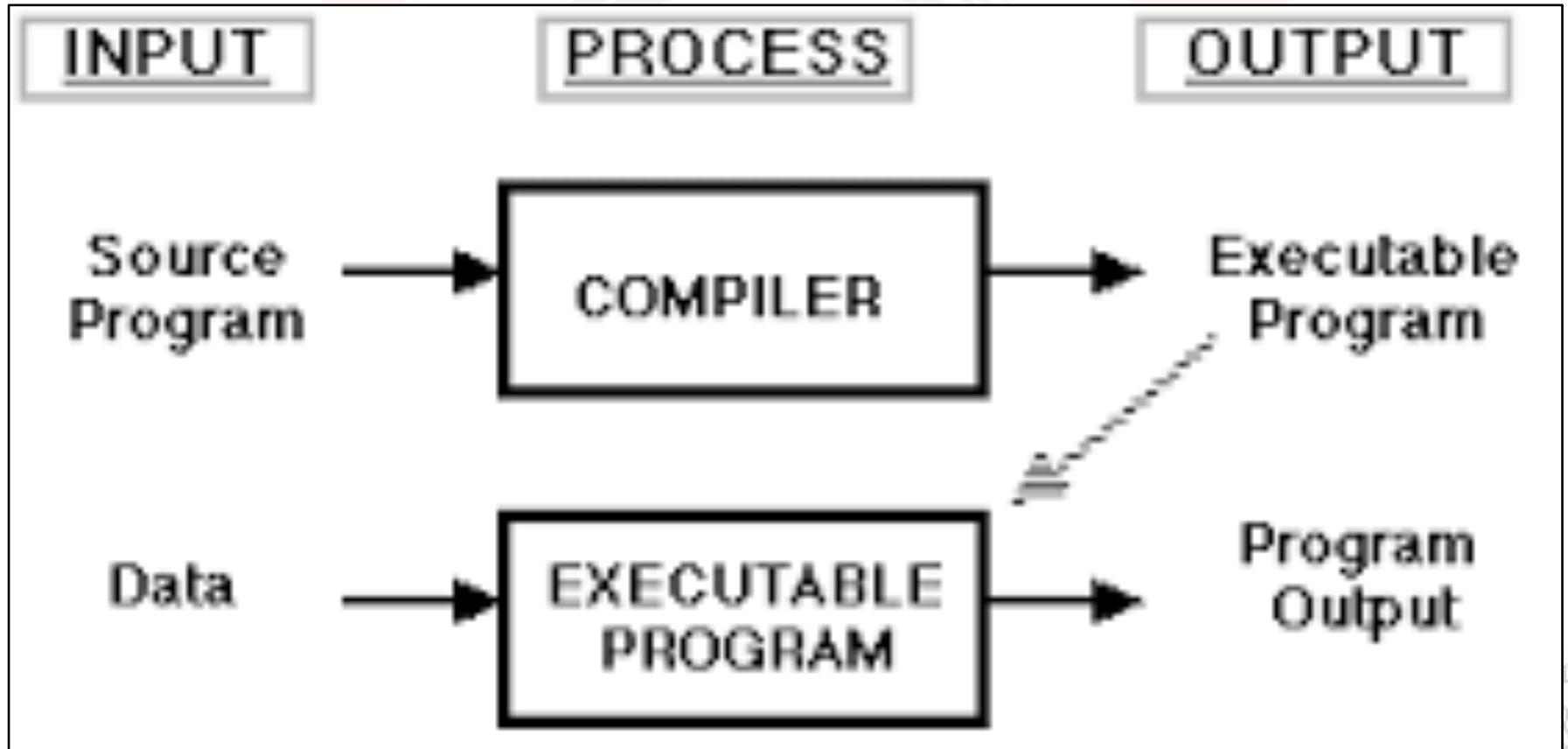
- 到底一個程式 (program) 是什麼？
- 一般來說，通常一個程式會有 3 個動作：
- 依序是讀進輸入 (read input) → 進行處理 (process) → 產生輸出 (write output)
- 拿各位以前在國中學過的「函數」來做舉例：



- x 的值就是輸入， x^2 是處理 (將輸入取平方)，而 $f(x)$ 的值就是輸出。
- 輸入：2、處理：將 2 取平方、輸出：4。
- 所以函數 f 就像一個程式，程式就像函數 f 。



A Program is like a Function



Conclusion I

- 寫作文就是用你會的語言，將你的想法，寫成文章，再經由翻譯，給外國人讀。
- 寫程式就是用你會的程式語言，將你的想法，寫成程式碼，再經由編譯，給電腦執行。
 - 寫程式的目的就是叫電腦幫你做事
- 學語言要學的是該語言的「單字」、「文法」、「片語」、「句型」……，還有怎麼樣把你腦袋裡的想法，用該語言表達出來。
- 學程式語言要學的是該程式語言的「關鍵字」、「語法」、「演算法」……，還有怎麼樣把你腦袋裡的想法，用該程式語言表達出來。



Normal Languages vs. Programming Languages

- 語言比較自由，程式語言比較嚴格。
- 你有點不符合語言的文法，翻譯員大致上還是能翻譯。
- 但若有**任何一點不符合**程式語言的語法，編譯器就不能編譯，發生**編譯錯誤** (*compile error*)。
- 好消息是通常編譯器會告訴你寫錯在哪裡（透過**錯誤訊息**，*error message*），所以一定要會看錯誤訊息。



Bugs?

- 編譯器就不能編譯的情形通常就稱作你的程式「有臭蟲 (*bug*)」。
- 而找尋錯在哪裡並修改錯誤的過程稱作「除錯 (*debug*)」。
- 還有一種 bug 是：編譯器就能編譯，但產生的執行檔執行結果卻是錯的。
- 這代表你的程式碼語法 (*syntax*) 正確，但語意 (*semantic*) 錯誤。
- 語法錯誤 (*syntax error*): Yesterday, I **write** a program.
- 語意錯誤 (*semantic error*): Yesterday, I **ate** a program.
- 演算法錯誤 (*algorithm error*): 如何把大象放進冰箱裡？
 - 會再之後細談這三種的差別
- 語法錯誤編譯器會告訴你錯在哪裡，其他錯誤不會。



Conclusion II

- 接下來好好學。
- Make sure that you perfectly understand.
- Feel free to ask any problems, even if it's subtle.



More about a Compiler

- 「編譯器 (compiler)」 我剛剛說它像「翻譯員」。(一次翻譯整篇文章)
- 還有一種東西叫作「直譯器 (interpreter)」，它像「口譯員」。(一次翻譯一句)
- 哪些是語言可以直譯： MATLAB, python
- 哪些是語言只能編譯： C, C++, JAVA



Okay! Let's Start Writing a Program!

But... What do I need?

- 寫作文需要「筆」跟「稿紙」，還有剛剛說的「翻譯員」。
- 寫程式需要「鍵盤」跟「純文字編輯器 (*plain text editor*)」，還有剛剛說的「編譯器 (*compiler*)」。
- 筆 = 鍵盤
- 紙 = 純文字編輯器 (*plain text editor*)
- 翻譯員 = 編譯器 (*compiler*)



What is a Plain Text Editor?

- 事實上你電腦裡的「記事本 (notepad.exe)」就是一個純文字編輯器。(文字編輯.app in OS X)
- 它讀你用鍵盤打的字當輸入，處理方式就是將字存起來，輸出一個你儲存的檔案。
- 存在電腦的長期記憶區 (LTM)，也就是硬碟 (HDD, Hard Drive Disk) 裡。(關於電腦構造下次會講)
- 因此你可以用記事本寫 code (not recommended)，存成 .cpp 檔，再用編譯器 g++ 來編譯，就可以了。



Why not?

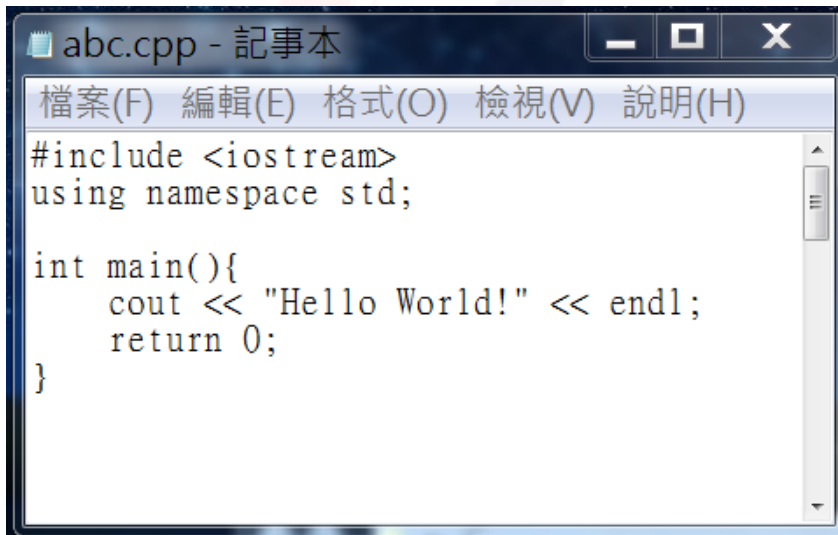
- 理由有二

1. 記事本沒有語法高亮 (syntax highlight)。
2. 你還不知道編譯器怎麼載怎麼裝，有沒有裝錯版本，裝在哪等……。
3. 你不會操作像 Terminal (OS X) or cmd (Windows) 這類 CUI。(command user interface)



What is Syntax Highlight

- 就是將你的 code 重點用螢光筆上色。

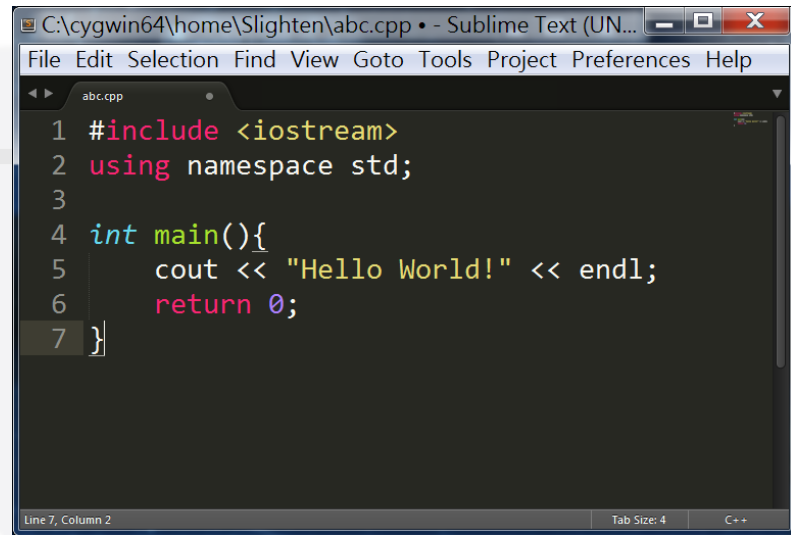


A screenshot of a Windows Notepad window titled "abc.cpp - 記事本". The menu bar includes "檔案(F)", "編輯(E)", "格式(O)", "檢視(V)", and "說明(H)". The code is as follows:

```
#include <iostream>
using namespace std;

int main(){
    cout << "Hello World!" << endl;
    return 0;
}
```

without syntax highlight



A screenshot of a Sublime Text window titled "C:\cygwin64\home\Slighten\abc.cpp - Sublime Text (UN...)". The menu bar includes "File", "Edit", "Selection", "Find", "View", "Goto", "Tools", "Project", "Preferences", and "Help". The code is as follows:

```
1 #include <iostream>
2 using namespace std;
3
4 int main(){
5     cout << "Hello World!" << endl;
6     return 0;
7 }
```

with syntax highlight



哪些 text editor 有語法高亮功能？

- Sublime Text (Windows/OS X), notepad++ ... (Windows)
- 所以用上述軟體加上自行下載 g++ 在 cmd 或 terminal 上進行編譯就可以寫程式。
- 但還有一種更棒的東西，叫作整合開發環境 (IDE)。



Integrated Development Environment (IDE)

- 將文字編輯器與編譯器，加上一堆其他好用的輔助功能，通通包在一起，讓你寫程式、編譯程式、產生執行檔、執行執行檔，都在同一個環境下執行。
- C/C++ 有哪些 IDEs：
 - [Dev-C++](#), [Code::Blocks](#) ... (Windows)
 - [Xcode](#) (OS X)
- 我個人是習慣在 CUI 上操作：
 - Windows: 裝 Cygwin，再裝 g++，用 vim 或 Sublime Text 寫 code，在 Cygwin 下執行 g++ 做編譯。
 - OS X: 裝完 Xcode 時 g++ 已同時裝完，用 vim 或 macvim 寫 code，在 terminal 下執行 g++ 做編譯。
- 欲知詳情請看VCR我網路上提供的影片。

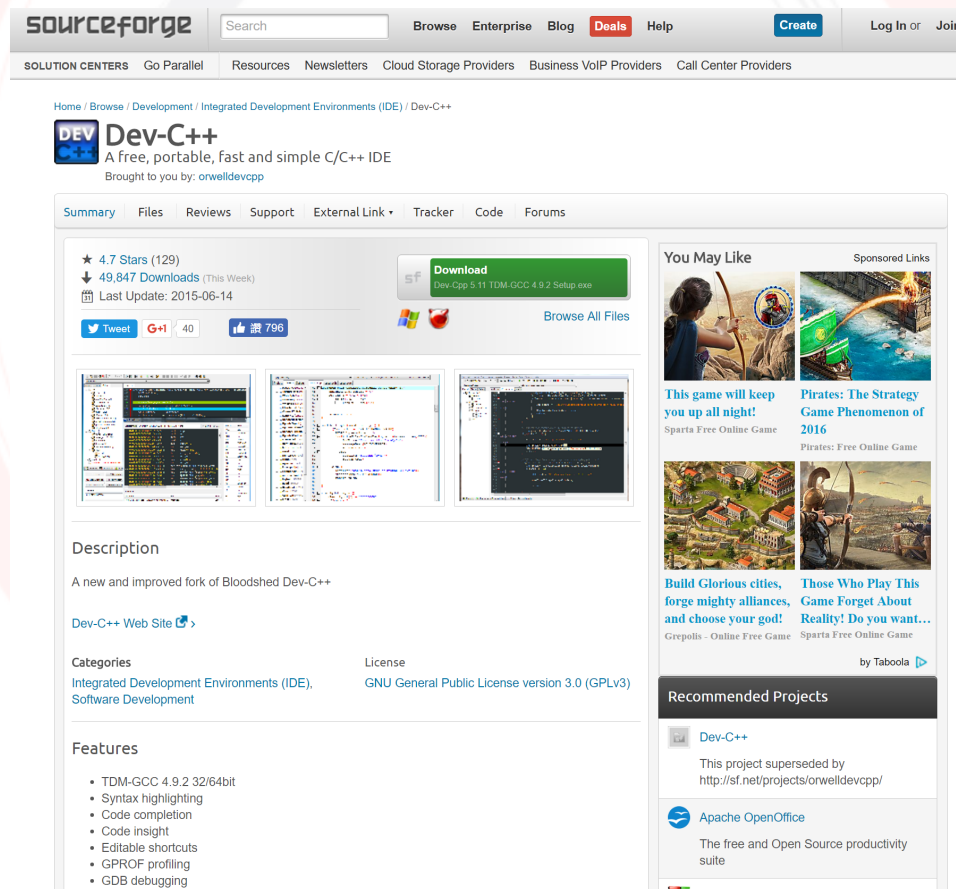


DEMO: Install an IDE (Dev-C++)



Google Dev-C++ and Download

- 4.9.2, 5.1.1, ... 隨便



The screenshot shows the SourceForge project page for Dev-C++. The page header includes the SourceForge logo, a search bar, and navigation links like 'Browse', 'Enterprise', 'Blog', 'Deals', 'Help', 'Create', 'Log In or Join'. Below the header, there are links for 'SOLUTION CENTERS', 'Go Parallel', 'Resources', 'Newsletters', 'Cloud Storage Providers', 'Business VoIP Providers', and 'Call Center Providers'.

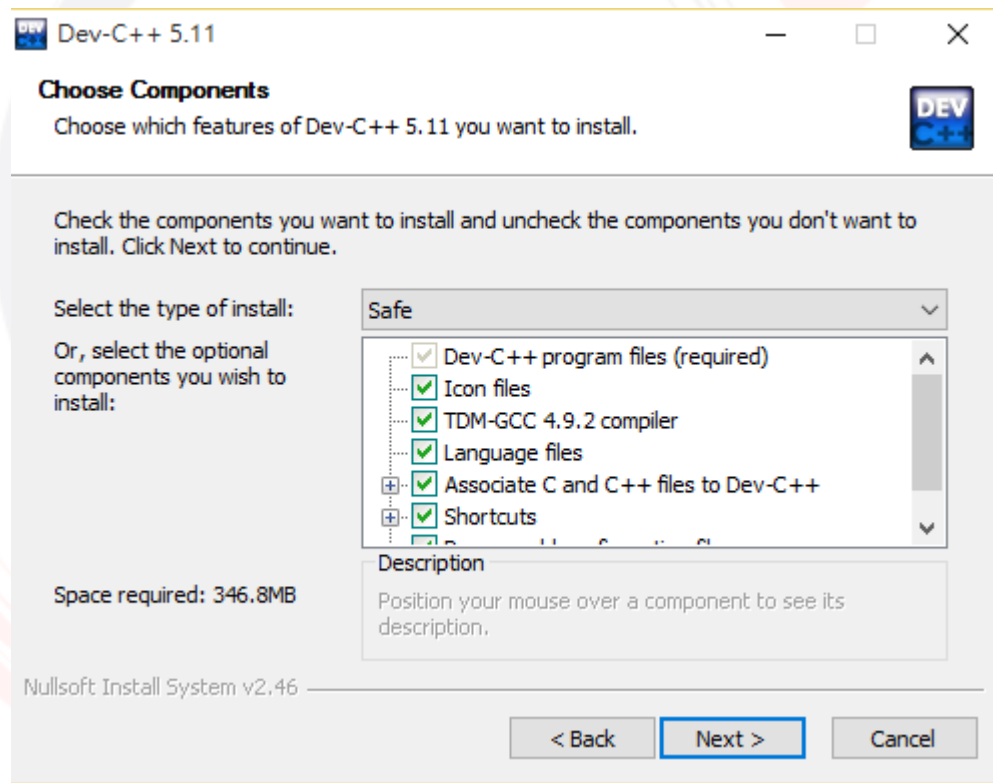
The main content area for the Dev-C++ project includes:

- DEV Dev-C++**: A free, portable, fast and simple C/C++ IDE. Brought to you by: orwelldevcpp.
- Summary**: 4.7 Stars (129), 49,847 Downloads (This Week), Last Update: 2015-06-14. Includes social media links for Twitter, Google+, and Facebook.
- Download**: A green button labeled 'Download' with the text 'Dev-Cpp 5.11 TDM-GCC 4.9.2 Setup.exe'.
- You May Like**: A section with sponsored links for games like 'Sparta Free Online Game' and 'Pirates: The Strategy Game Phenomenon of 2016'.
- Description**: A new and improved fork of Bloodshed Dev-C++.
- Dev-C++ Web Site**: A link to the project's website.
- Categories**: Integrated Development Environments (IDE), Software Development.
- License**: GNU General Public License version 3.0 (GPLv3).
- Features**:
 - TDM-GCC 4.9.2 32/64bit
 - Syntax highlighting
 - Code completion
 - Code insight
 - Editable shortcuts
 - GPROF profiling
 - GDB debugging



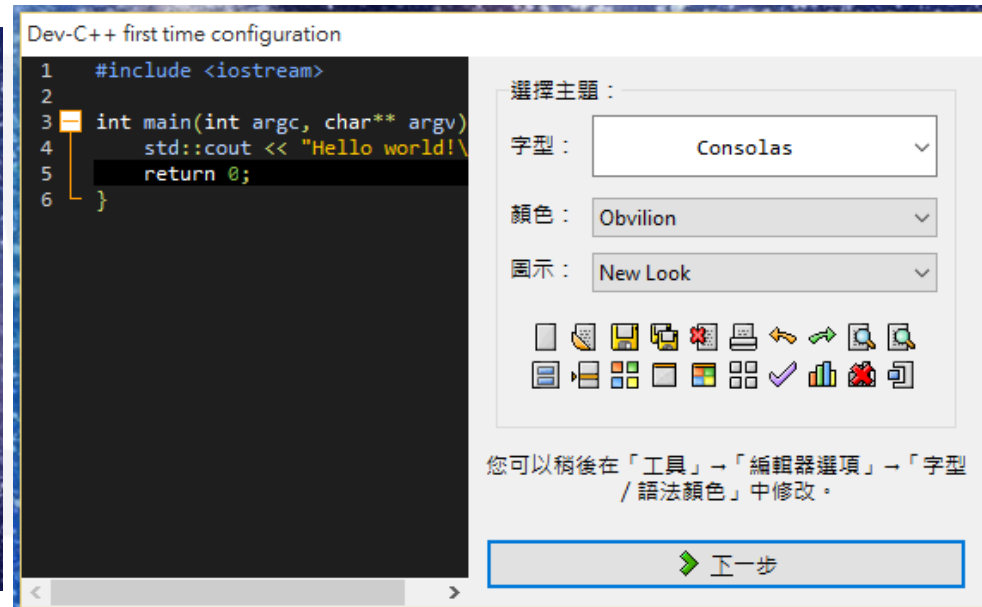
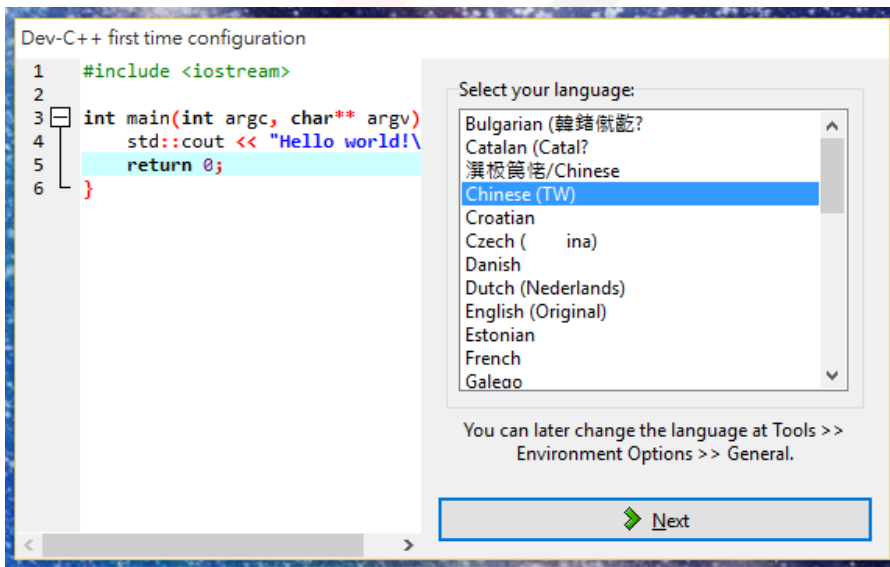
Install

- Keep Next and Done



Initial Setting

- Chinese (TW)



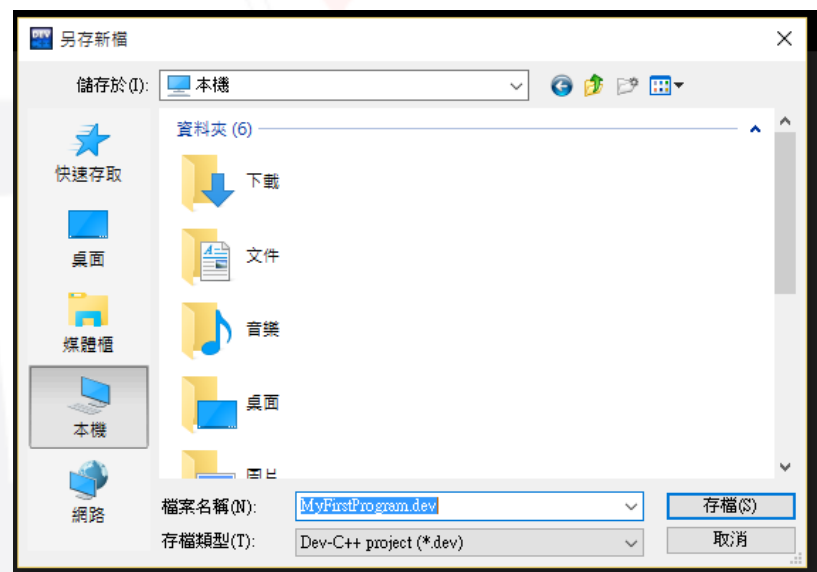
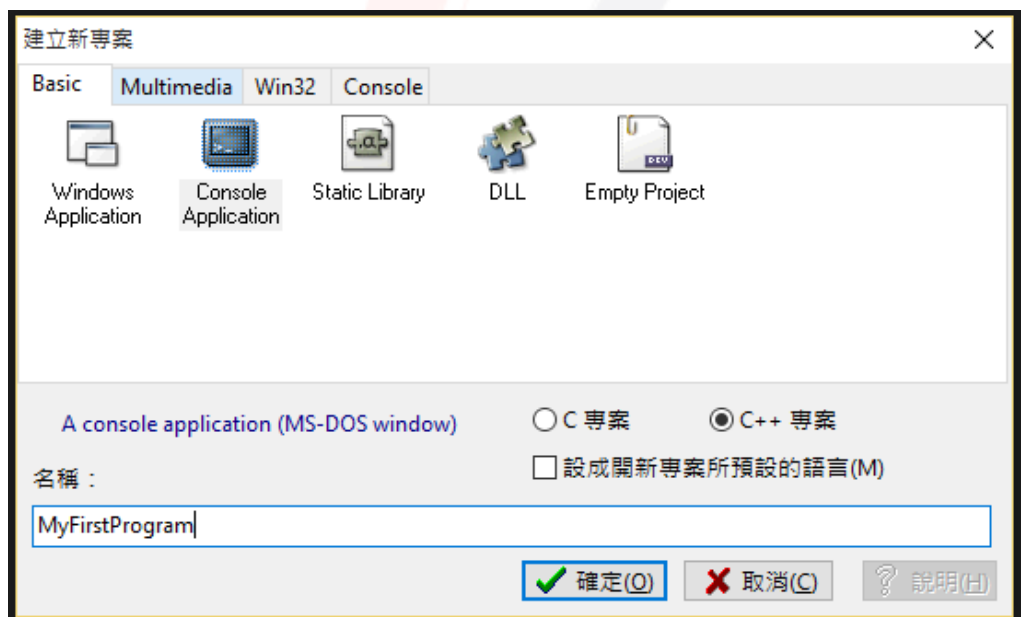
Larger Font Size

- 工具->編輯器選項->字型->大小



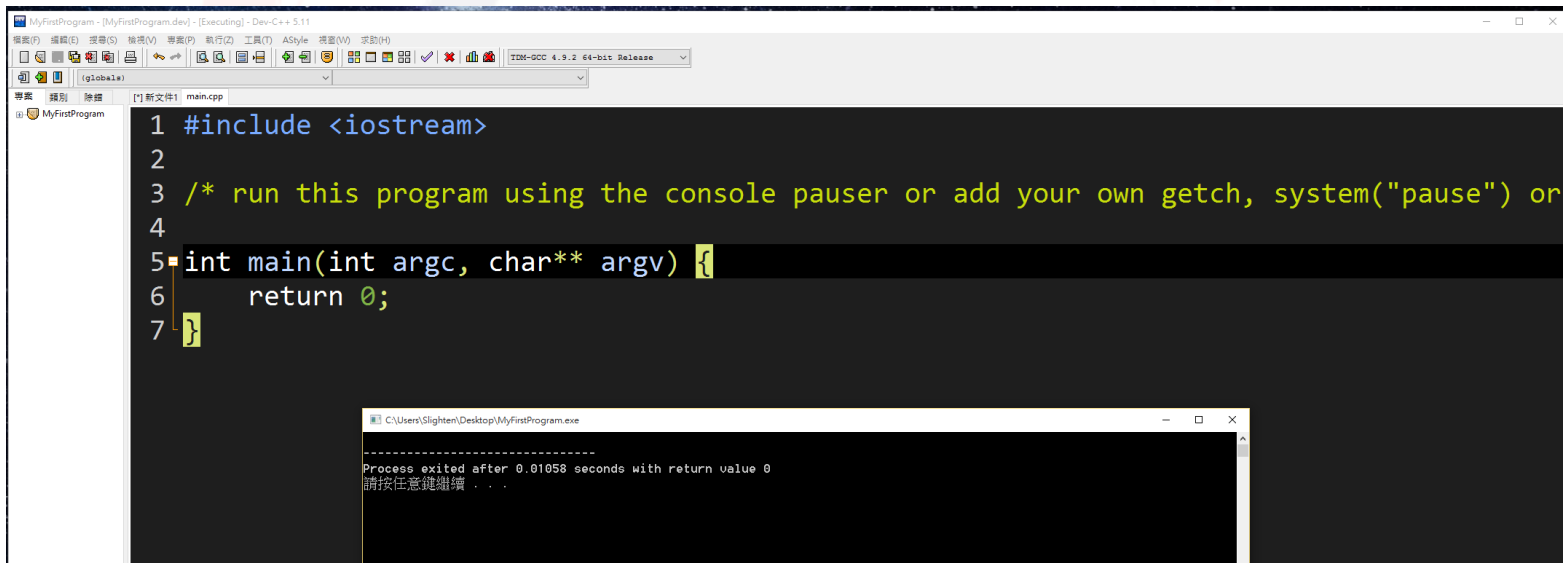
Create new Project

- 檔案->開新檔案->專案



Compile and Run

- 按 F11 (或上方按鈕) 編譯並執行



The screenshot shows a C++ IDE window titled "MyFirstProgram - [MyFirstProgram.dev] - [Executing] - Dev-C++ 3.11". The code editor displays the following C++ code:

```
1 #include <iostream>
2
3 /* run this program using the console pauser or add your own getch, system("pause") or
4
5 int main(int argc, char** argv) {
6     return 0;
7 }
```

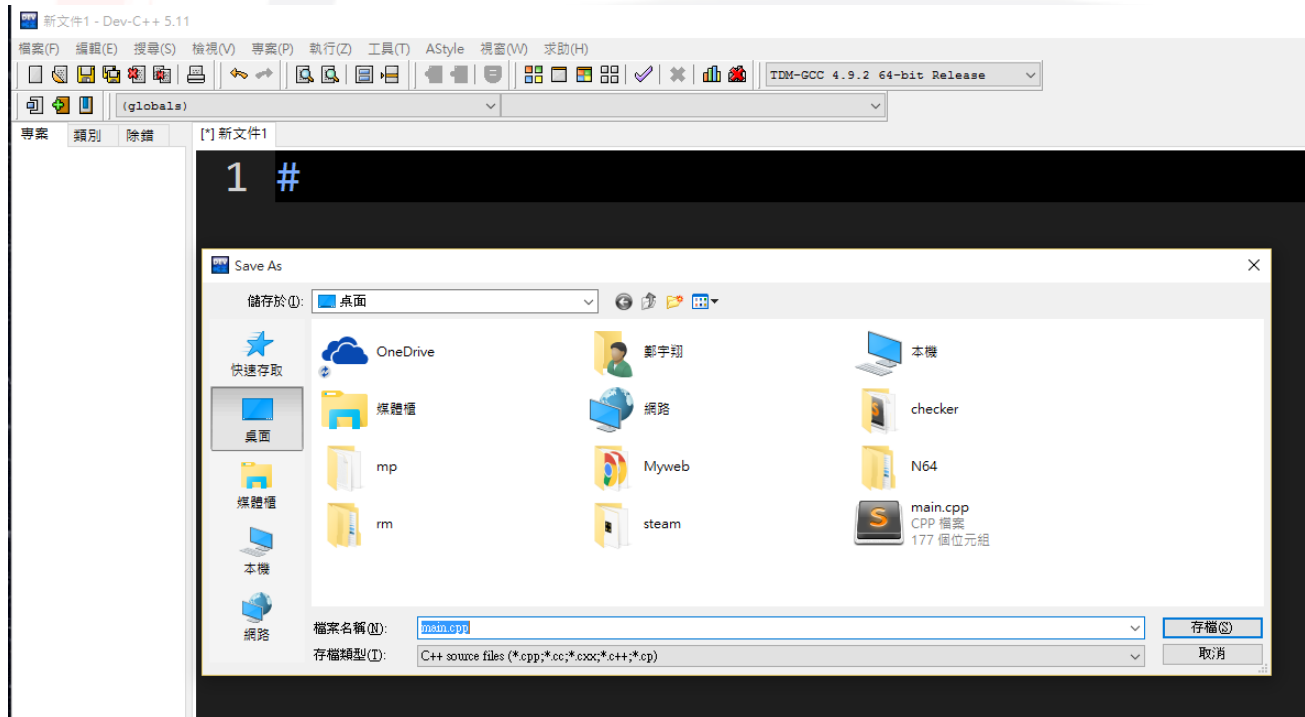
Below the code editor, a console window titled "C:\Users\Slighten\Desktop\MyFirstProgram.exe" shows the output:

```
-----
Process exited after 0.01058 seconds with return value 0
請按任意鍵繼續 . . .
```



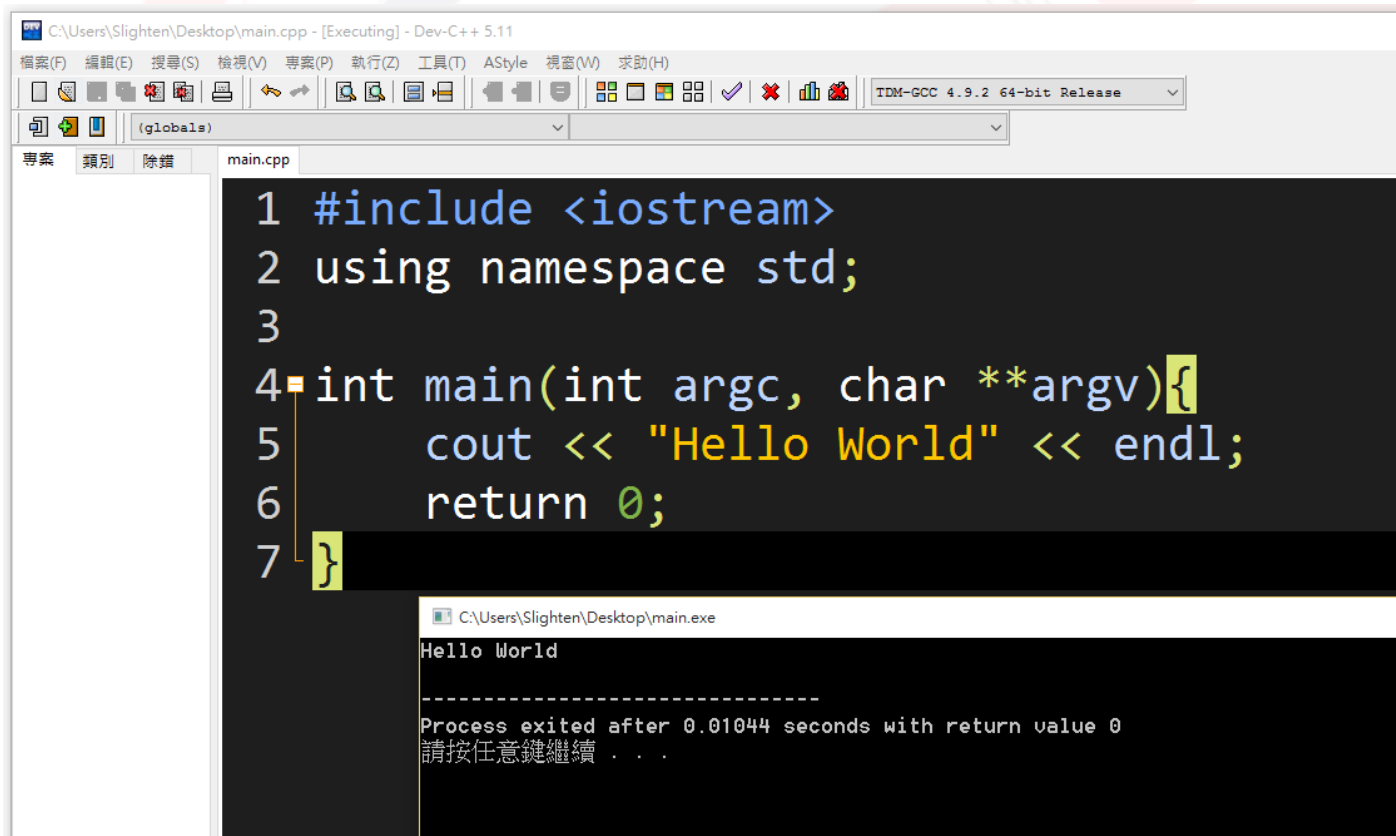
Create an Empty File instead of a new Project

- 檔案->開新檔案->原始碼
- 隨便打幾個字然後 Ctrl + s 另存成 main.cpp



Compile and Run

- 按 F11 (或上方按鈕) 編譯並執行



The screenshot shows the Dev-C++ IDE interface. The title bar indicates the file path is C:\Users\Slighten\Desktop\main.cpp and it is being executed with Dev-C++ 5.11. The menu bar includes File (F), Edit (E), Search (S), View (V), Project (P), Run (Z), Tools (T), AStyle, Window (W), and Help (H). The toolbar contains icons for file operations, editing, and running. The compiler is set to TDM-GCC 4.9.2 64-bit Release. The project list on the left shows 'main.cpp'. The main editor displays the following C++ code:

```
1 #include <iostream>
2 using namespace std;
3
4 int main(int argc, char **argv){
5     cout << "Hello World" << endl;
6     return 0;
7 }
```

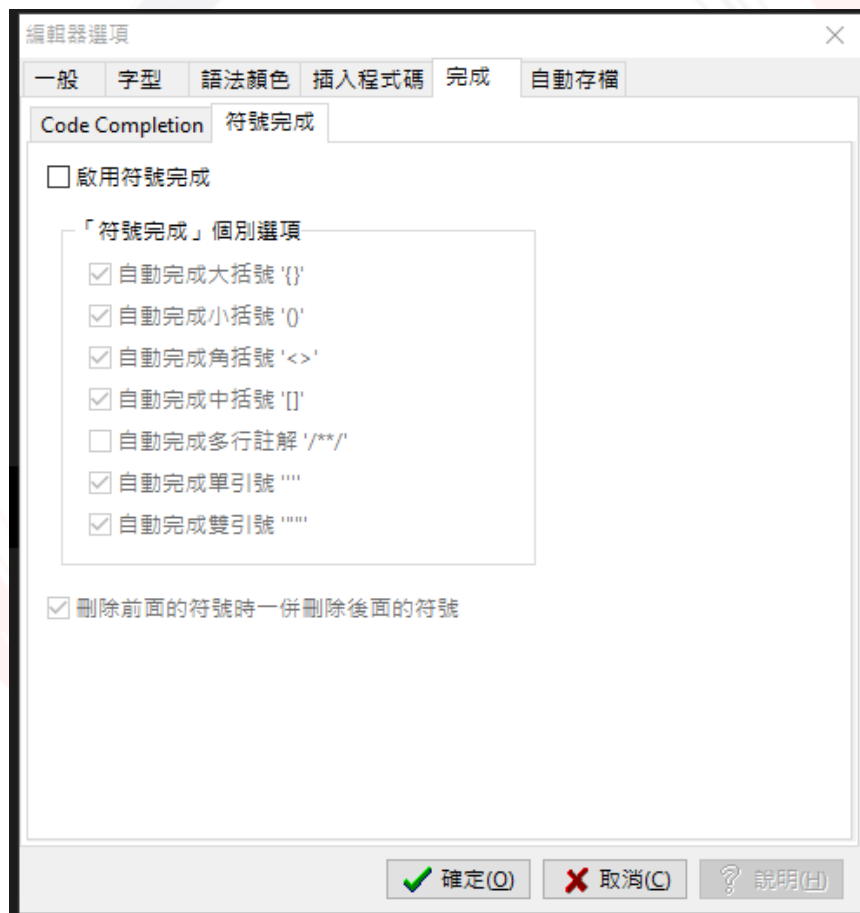
Below the code editor, a console window titled 'C:\Users\Slighten\Desktop\main.exe' shows the output:

```
Hello World
-----
Process exited after 0.01044 seconds with return value 0
請按任意鍵繼續 . . .
```



避免括號自動完成

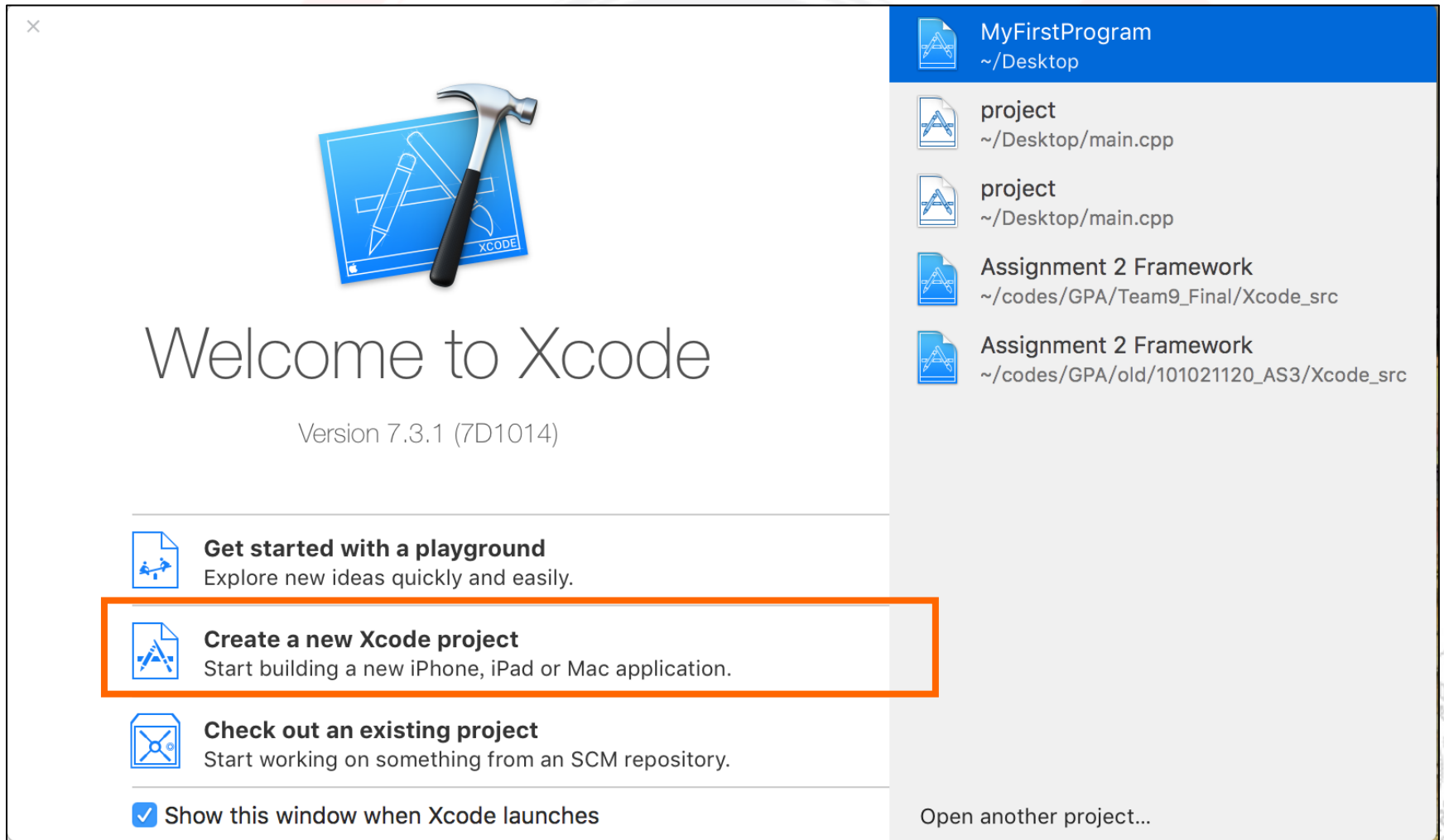
- 工具->編輯器選項->完成->符號完成->不啟用



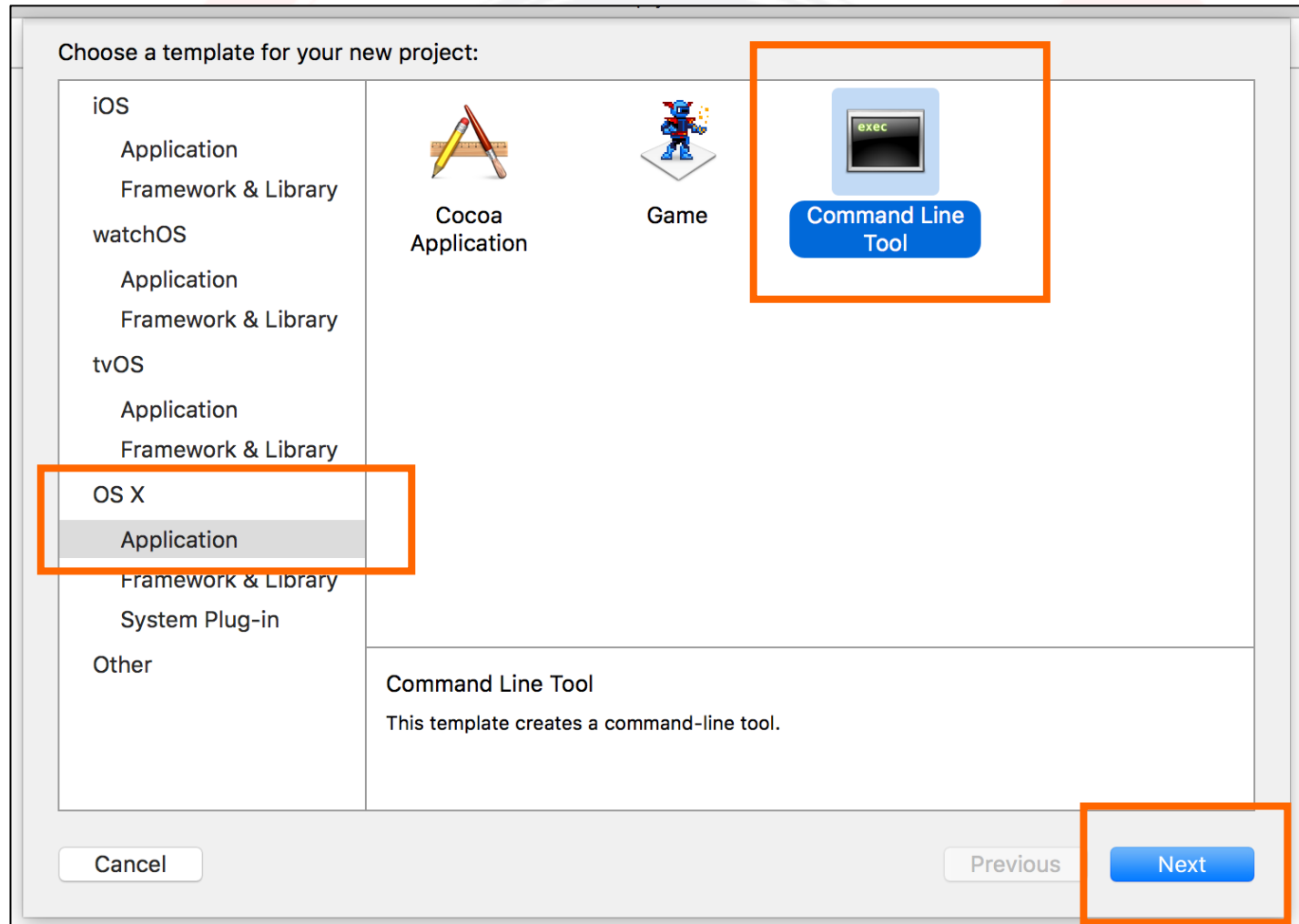
DEMO: Install an IDE (Xcode)



Create a new Xcode Project



Choose a Template for your new Project



Type the Product Name

Empty Tab

Choose options for your new project:

Product Name:

Organization Name:

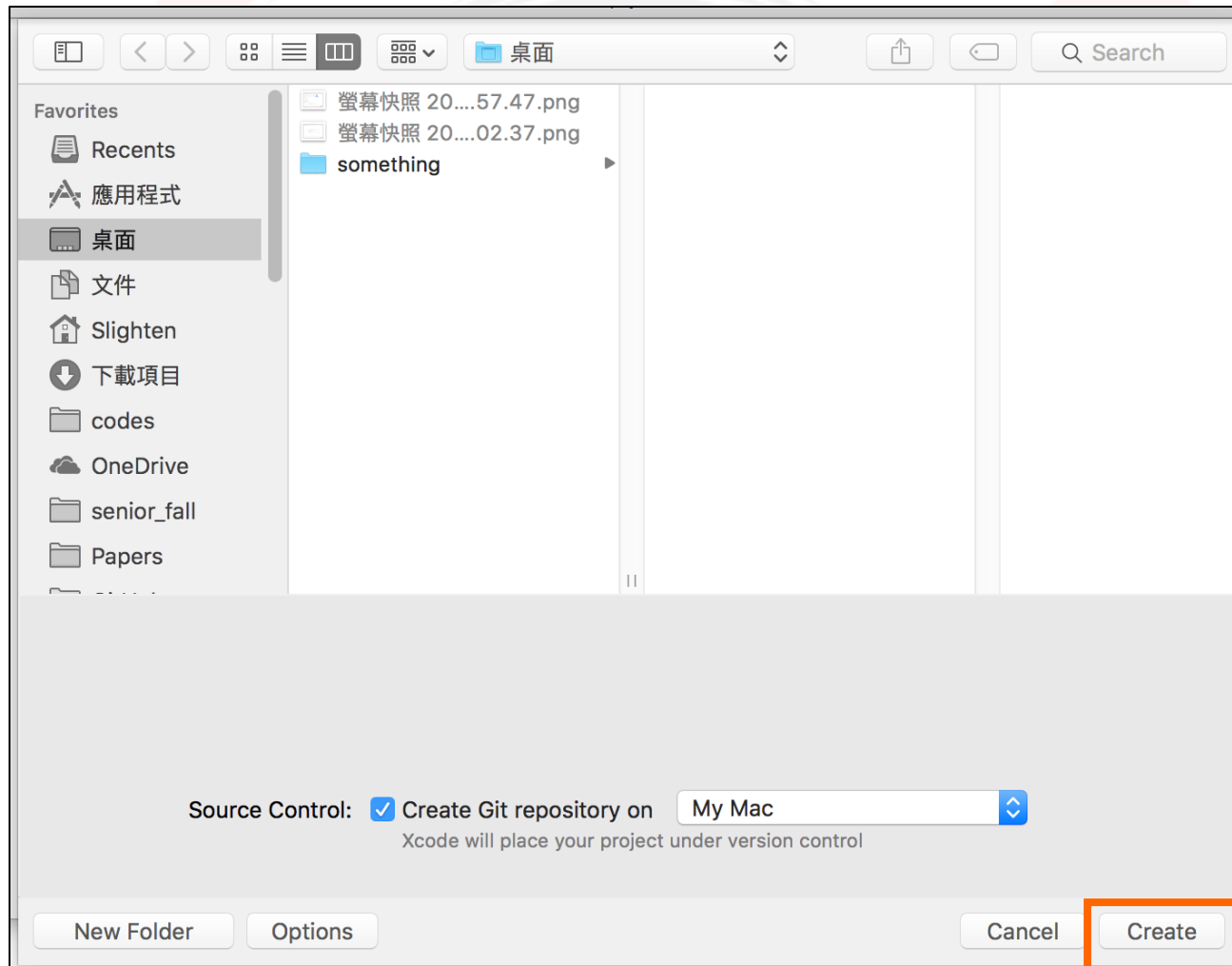
Organization Identifier:

Bundle Identifier:

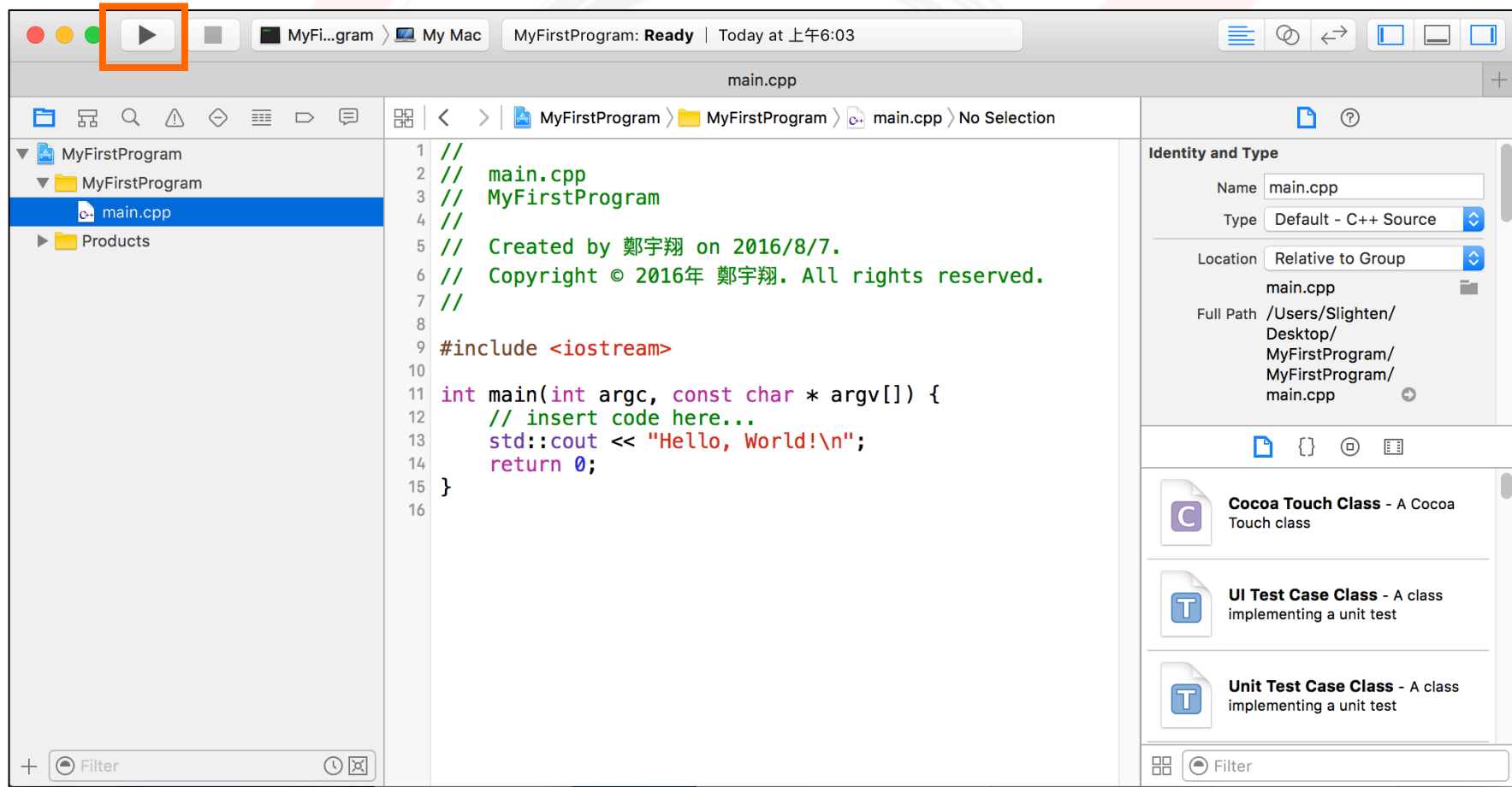
Language:



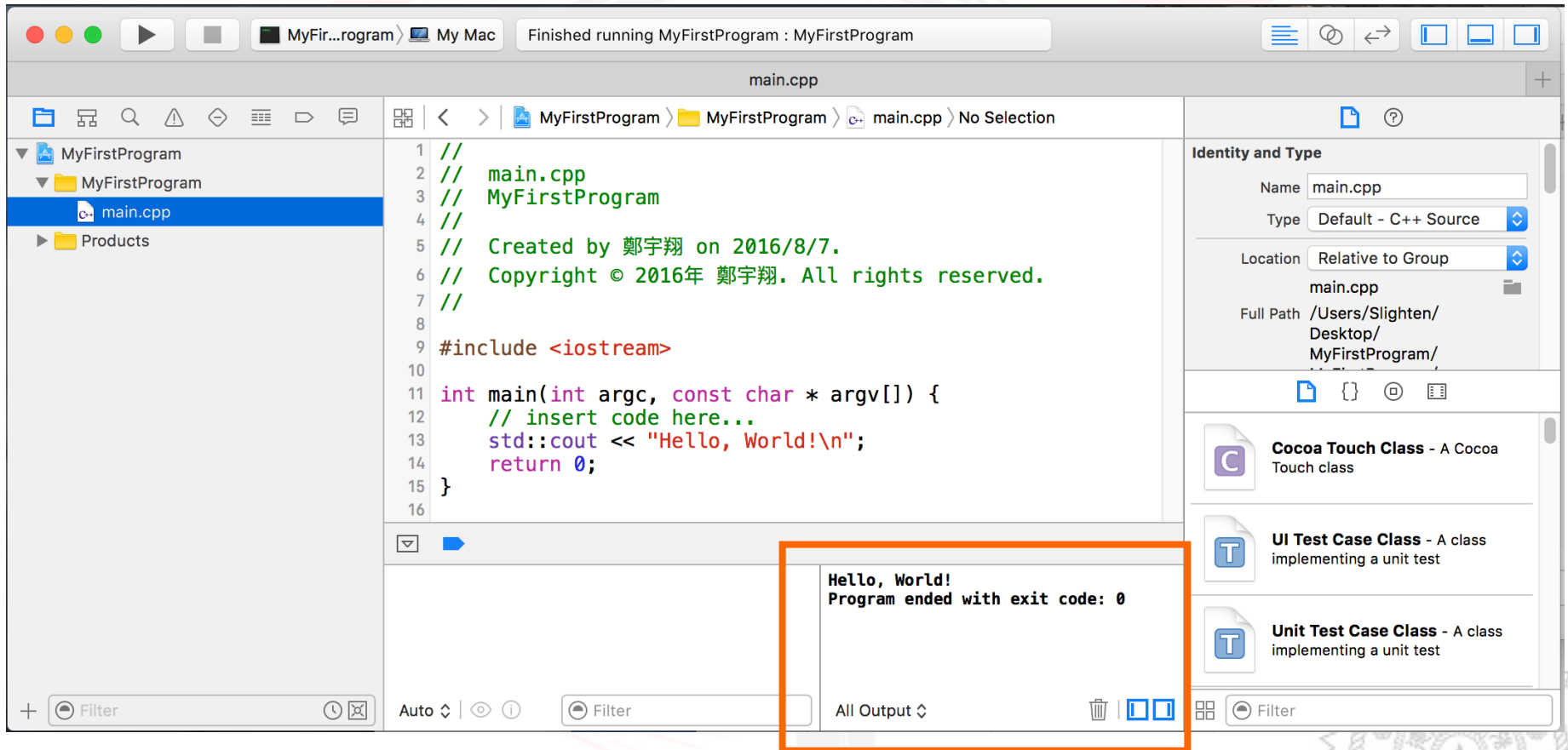
Find a Place to Save your Project



編譯並執行程式



Check the Output



Outline

- Programming ABC
- Our First C++ Program
- Some Practices



Our First C++ Program



Our First C++ Program

```
program1.cpp
1 ▾ /* File name: program1.cpp
2   * Author: Yu Hsiang Zheng
3   * Functionality:
4     This program takes no input,
5     and outputs a string: Hello World!
6   */
7
8  #include <iostream>
9  using namespace std;
10
11 // main function
12 ▾ int main(){
13     cout << "Hello World!" << endl;
14     return 0;
15 }
16 // End of File
```



編譯後執行結果

```
C:\cygwin64\home\Slighten\program1.exe
Hello World!
-----
Process exited after 0.0184 seconds with return value 0
請按任意鍵繼續 . . .
```



註解 (Comments)

```
program1.cpp
1 /* File name: program1.cpp
2  * Author: Yu Hsiang Zheng
3  * Functionality:
4   This program takes no input,
5   and outputs a string: Hello World!
6  */
7
8 #include <iostream>
9 using namespace std;
10
11 // main function
12 int main(){
13     cout << "Hello World!" << endl;
14     return 0;
15 }
16 // End of File
```

註解 (Comments) 有兩種

1. 多行註解：

用 `/*` 開頭，用 `*/` 結尾
(缺點：不能巢狀)

! 13 `/* /* */ */`

2. 單行註解：

用 `//` 開頭，行尾就是結尾
(缺點：一次只能註解一行)

- 編譯器完全忽略

- 功用：

解釋程式碼並增加可讀性
(readability)

- 1) 讓他人與幾日後的你自已看得懂這段程式碼
- 2) 方便 debug
- 3) 方便日後修改以增加新功能

巨集指令 (Macros)

```
program1.cpp x
1 ▾ /* File name: program1.cpp
2   * Author: Yu Hsiang Zheng
3   * Functionality:
4     This program takes no input,
5     and outputs a string: Hello World!
6   */
7
8 #include <iostream>
9 using namespace std;
10
11 // main function
12 ▾ int main(){
13     cout << "Hello World!" << endl;
14     return 0;
15 }
16 // End of File
```

巨集指令 (macro) 或前處理器指令 (preprocessor directive)

- 以 # (pound sign, number sign, hash...) 開頭，後接一個指令。
 - 如：#include, #define, #ifdef, #ifndef, #endif...
- 代表前處理器要做的動作。
- 第 3 章會教 #define
- 第 7 章類別會教 include guard 將用到 #ifndef, #endif



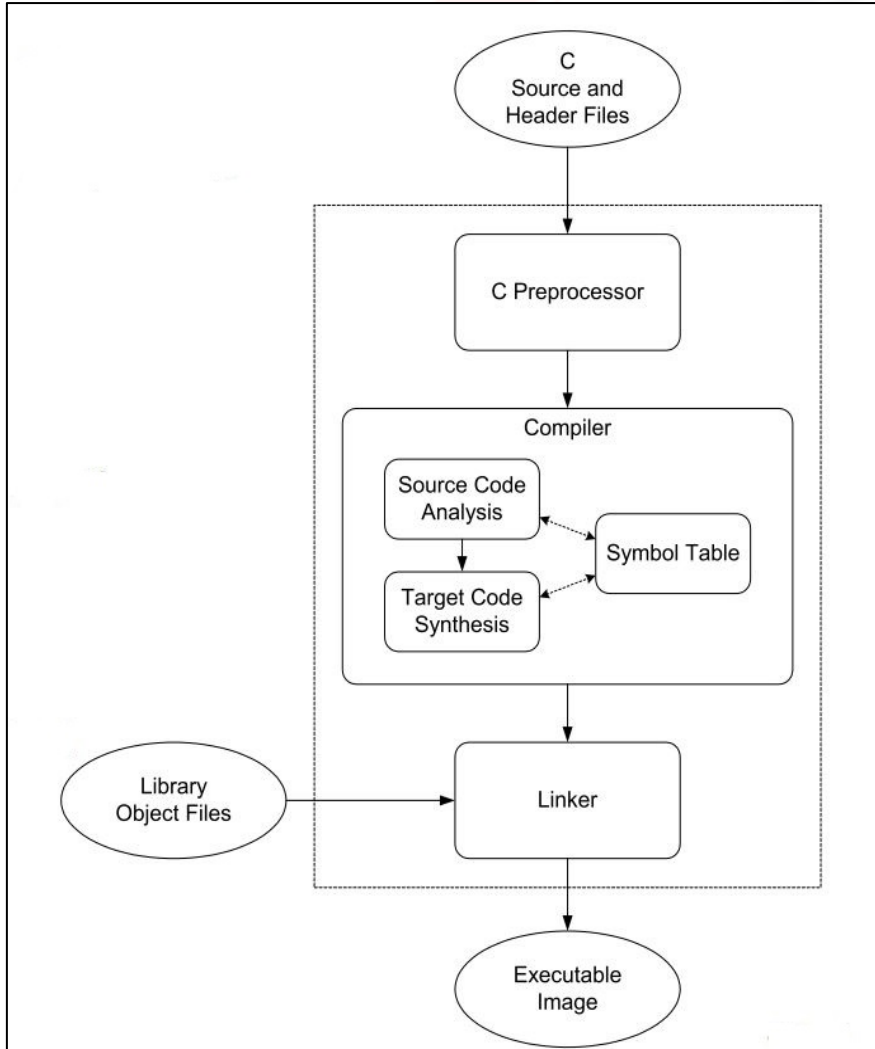
標頭檔 (Header Files)

```
program1.cpp x
1 ▾ /* File name: program1.cpp
2   * Author: Yu Hsiang Zheng
3   * Functionality:
4     This program takes no input,
5     and outputs a string: Hello World!
6   */
7
8   #include <iostream>
9   using namespace std;
10
11  // main function
12 ▾ int main(){
13     cout << "Hello World!" << endl;
14     return 0;
15 }
16 // End of File
```

iostream 是一個標頭檔 (header file)

- 標頭檔就像一本某科的參考書，裡面對於該科會用到的辭彙與用法(知識)有很詳細的解釋。
- iostream 意思是輸入/輸出串流 (I/O stream, I for input, O for output)。
- cout, cin, <<, >>, endl ... 都在 iostream 裡有定義 (直接拿來用就好)。
- #include <iostream> 代表前處理器會拿 iostream 這本參考書的內容原封不動直接替換掉這行，稱作巨集替代 (macro substitution)

What is a Preprocessor?



將編譯器再細分。

前處理器 (預處理器, Preprocessor)

- 巨集替代 (macro substitution): `#include`, `#define`
- 條件編譯 (conditional compilation): `#ifndef`, `#ifdef`, `#endif`
- 來源層級的轉換
 - 輸出仍為 C++ 語言



命名空間 (Namespace)

```
program1.cpp
1 /* File name: program1.cpp
2  * Author: Yu Hsiang Zheng
3  * Functionality:
4   This program takes no input,
5   and outputs a string: Hello World!
6  */
7
8 #include <iostream>
9 using namespace std;
10
11 // main function
12 int main(){
13     cout << "Hello World!" << endl;
14     return 0;
15 }
16 // End of File
```

命名空間 (namespace)

- 事實上剛剛說的 `cout`, `cin`, `<<`, `>>`, `endl`...是定義在標頭檔 `iostream` 裡的可作 `std` 的命名空間。
- 命名空間的設計是為了避免在同個或不同標頭檔同樣出現某個詞彙。
- 例如在同個標頭檔，可能有兩種不同的 `cout` 的定義。
- 就好像同一本參考書對同一個名詞可能有兩個定義，你可以用章節來區分。
 - Point: 2D vs. 3D
- 同一個標頭檔對同一個名詞可能有兩個定義，則用命名空間來區分。



A Substitution of Using namespace std

```
program1.cpp
1  /* File name: program1.cpp
2   * Author: Yu Hsiang Zheng
3   * Functionality:
4     This program takes no input,
5     and outputs a string: Hello World!
6   */
7
8  #include <iostream>
9
10 // main function
11 int main(){
12     std::cout << "Hello World!" << std::endl;
13     return 0;
14 }
15 // End of File
```

- 若沒寫這行，那你每當用到每個標頭檔裡的詞彙，都要標示出是哪個命名空間。
- 像 cout 就要改成 `std::cout`。
- 像 endl 就要改成 `std::endl`。



主函式 (Main Function)

```
program1.cpp
1 /* File name: program1.cpp
2  * Author: Yu Hsiang Zheng
3  * Functionality:
4   This program takes no input,
5   and outputs a string: Hello World!
6  */
7
8 #include <iostream>
9 using namespace std;
10
11 // main function
12 int main(){
13     cout << "Hello World!" << endl;
14     return 0;
15 }
16 // End of File
```

主函式 (main function)

- 程式由這裡開始執行
- 也在這裡結束
- 一個主函式的型式為
-

```
int main(...) {
    ...
    return 0;
}
```

- 若是自己寫的副函式名稱可以隨便叫。

Main Function (1/3)

- 第一個 `int` 代表這個函式的回傳值 (輸出) 的資料類型是整數(Integer)。
- Main function 回傳值就是程式的回傳值。
- 對應到第 14 行的 `return 0`，也就是回傳值為整數 0。
- 回傳值是自己設定的。
- 通常一個程式的回傳值是 0 代表這個程式「正確執行」，而回傳非 0 值代表這個程式執行過程中「有某種錯誤」，例如：讀不到輸入、網路連線失敗...
- `main` 是 函式名稱。
- 主函式一定要取 `main`，不然程式會找不到進入點。(編譯錯誤)
- `main()`，`()`裡放的是參數(輸入)類型與參數(輸入)名稱，沒寫代表主函式沒有輸入，等同於 `int main(void)`。
- 也可以寫 `int main(int argc, const char *argv[])`，這裡的參數意義以後再解釋。
- `{ }` 裡放的是處理過程。
- 注意裡面每行陳述句 (statement) 都要以 `;` (分號) 結尾。

```
12 int main(){
13     cout << "Hello World!" << endl;
14     return 0;
15 }
```



Main Function (2/3)

➤ 一個函式的型式是：

```
回傳型別 函式名稱 (參數1型別 參數1名稱, 參數2型別 參數2名稱, ...){  
    /* ... */  
    return 回傳值(須符合回傳型別);  
}
```

➤ 例如：

```
int add(int a, int b){  
    return a + b;  
}
```

```
12 int main(){  
13     cout << "Hello World!" << endl;  
14     return 0;  
15 }
```

➤ 第 4 章函數會詳談



Main Function (3/3)

- 注意一件很重要的事！
- 程式碼裡多餘的空白、縮排、換行，都是給人讀的，編譯器會把它們跟註解一樣忽略。
- 所以你可以寫成以下方式

```
12 int main(){cout<<"Hello World!"  
13 <<endl;  
14  
15 return  
16 0      ; }
```

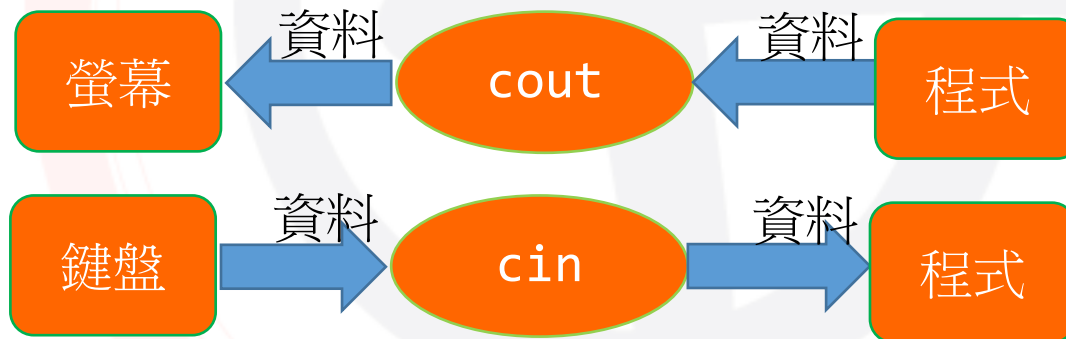
```
12 int main(){  
13     cout << "Hello World!" << endl;  
14     return 0;  
15 }
```

- 執行結果一模一樣，但非常難讀，請絕對不要這麼做。
- 這會牽扯到編程風格(coding style)，寫程式請務必有良好的編程風格 (之後會詳談)。
- 若不知道什麼是良好的編程風格，請先跟著我的範例程式碼做。



cout << "Hello World!" << endl; (1/2)

- cout 你可以想像是個程式的對外窗口，裡面是程式，外面是螢幕。
- (之後會遇到 cin，你可以想像是個電腦的對內窗口，裡面是程式，外面是鍵盤輸入。)



- 將 "Hello World!" 這串字用 << 丟給 cout，cout 會將這串字傳給螢幕（顯示在螢幕上）。
- 將 << 想像成會意字。



cout << "Hello World!" << endl; (2/2)

- 緊接著 "Hello World!"，程式再將 endl 丟給螢幕。
- endl 代表 end of line (換行)。
- '\n' 是換行字元。(註：程式碼裡的字串皆用 "" 括起，字元用 ' ' 括起。)
- 剛剛說了你程式碼裡的換行是會被編譯器忽略的，因此要有符號來表示換行。
- 因此此行也可改為

```
cout << "Hello World!\n";
```

或

```
cout << "Hello World!" << '\n';
```

- 也可以將 "Hello World!" 分成兩串字輸出：

```
cout << "Hello " << "World!" << endl;
```

- 也可以將 "Hello World!" 拆開，一個字元一個字元丟給 cout。

```
cout << 'H' << 'e' << 'l' << 'l' << 'o' << ' ' << 'W' << 'o' << 'r' << 'l' << 'd' << '!' << endl;
```



跳脫字元 (Escape Character) 與 跳脫序列 (Escape Sequence)

- 跳脫字元 ('\\') 讓後接的字元「有特殊功能變無特殊功能」或「無特殊功能變有特殊功能」，整體叫作跳脫序列

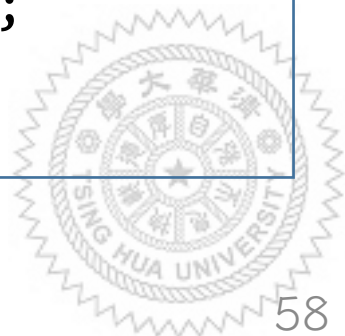
表 3.2.2 常用的跳脫序列

跳脫序列	所代表的意義	ASCII 十進位值	ASCII 十六進位值
\\a	警告音 (Alert)	7	0x7
\\b	倒退一格 (Backspace)	8	0x8
\\n	換行 (New line)	10	0xA
\\r	歸位 (Carriage return)	13	0xD
\\t	跳格 (Tab)	9	0x9
\\0	字串結束字元 (Null character)	0	0x0
\\\\	反斜線 (Backslash)	92	0x5C
\\'	單引號 (Single quote)	39	0x27
\\"	雙引號 (Double quote)	34	0x22

Return 0

- `return;` // 跳出這個函式，不帶任何回傳值。
- `return 0;` // 跳出這個函式，並回傳一個 0 值。
- 因為會跳出函式，所以碰到 `return`，就不會再繼續執行下面的東西。
- 例如：

```
int main(){  
    return 0;  
    cout << "This line won't be executed." << endl;  
}
```



Something you Should Know (1/2)

- 電腦跟人類有些不一樣。
- 人類看文章可以正著來、倒著來、整體看、個別看 ……。
- 人類讀的其實不是眼前的東西，而是「大腦裡」的東西。
- 也就是說，就算你一個字拼法順序有點顛倒，人類的大腦會把它視為它記憶中最相近的東西。
- 但只要有一個字拼反，電腦會認為是不一樣的東西。
- 電腦只會一行一行由上到下，一字一字由左到右「依序」讀。
- 「順序」(邏輯) 很重要，通常程式碼任兩行對調，結果會不同。
- 人類對「模式辨識」(pattern recognition) 很擅長，電腦只擅長「計算」(arithmetic, computation)。

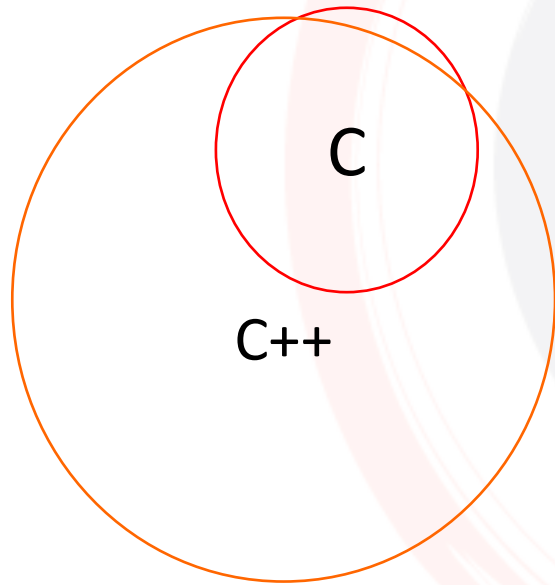


Something you Should Know (2/2)

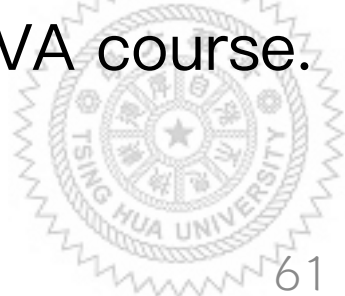
- 人類對「模式辨識」(pattern recognition) 很擅長，電腦只擅長「計算」(arithmetic, computation)。

Aoccdrnig to a rscheearch at Cmabrigde Uinervtisy, it de-
osn't mntaer in waht oredr the ltteers in a wrod are, the
olny iprmoatnt tihng is taht the frist and lsat ltteer be at
the rghit pclae. The rset can be a total mses and you can
sitll raed it wouthit porbelm. Tihs is bcuseae the huamn
mnid deos not raed ervey lteter by istlef, but the wrod as
a wlohe.

The Relationship Between C & C++



- $? \in C \setminus C+$
 - https://en.wikipedia.org/wiki/Compatibility_of_C_and_C%2B%2B
- OOP (物件導向程式設計, object-oriented programming) $\in C++ \setminus C$
- class (類別), object (物件), inheritance (繼承), override (覆寫), polymorphism (多形), overload (多載), generic (泛型) ...
- They will be included in Chapter 7.
- You can also learn OOP in JAVA course.



打字時或複製投影片上程式碼時請注意

- “ ” 這兩個是中文的雙引號，" 才是英文的雙引號（要用這個）
- ‘ ’ 這兩個是中文的單引號，' 才是英文的單引號（要用這個）
- ' ' 這裡面是中文全形空白，' ' 裡面才是英文的半形空白（要用這個）
- ？ ！ ， ； 「 」 （ ） % # @ & | + _ 這些都是中文符號，要用 ？ ！ ， ； { } [] () % # @ & | + _ -



Outline

- Programming ABC
- Our First C++ Program
- Some Practices



Some Practices



改錯 I

- 以下程式碼哪些部分會造成編譯錯誤，請找出並更正

```
1 #include <iostream>
2 using namespace std
3
4 void main()
5 {
6     cout << Hi! My name is Kyle Hyde << endl;
7     cout << "I'm working as a travelling salesman for Red Crown"
8     cout << "\n";
9     return 0;
10    return 1;
11 }
12
```



想一想！

- 填空題

```
1 #include <iostream>
2 using namespace std;
3 int main(){
4     int a;
5     cout << "Please input a number: ";
6     cin __ a;
7     cout << "The _____ of "<< _ << " is "
8     << a*a << endl;
9 }
10
```

- ____ 處應該要填什麼呢？



想一想 II

- 為什麼要 endl 呢？沒有會怎樣呢？

```
8 #include <iostream>
9 using namespace std;
10
11 // main function
12 int main(){
13     cout << "Hello World!" << endl;
14     cout << "I'm Superman!" << endl;
15     return 0;
16 }
17 // End of File
```

```
8 #include <iostream>
9 using namespace std;
10
11 // main function
12 int main(){
13     cout << "Hello World!";
14     cout << "I'm Superman!";
15     return 0;
16 }
17 // End of File
```

- 請猜猜兩左右兩段 code 執行結果有何差異，動手執行看看，並解釋為什麼。



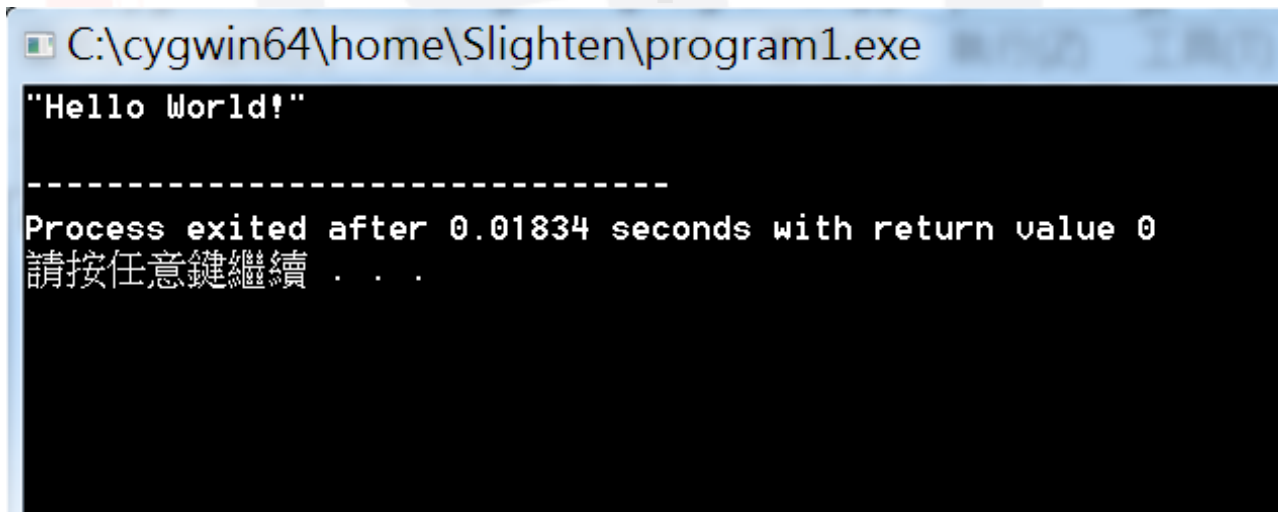
作業 Part 0

- 印出 `Hello World!` ◦
- 檔名叫做 `hw1_0.cpp`
- 目的：暖身練習



作業 Part I

- 印出橫的 `"Hello World!"`。(連同雙引號一起印出)
- 檔名叫做 `hw1_1.cpp`
- 目的：學會印雙引號

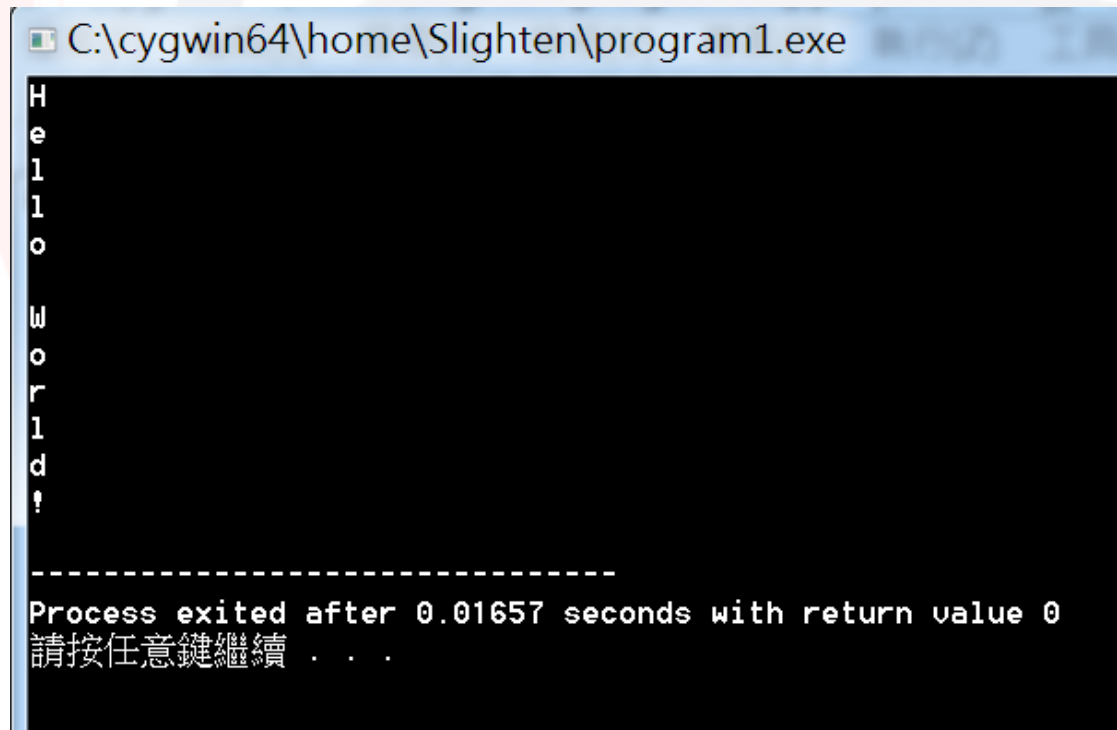
A screenshot of a Windows command prompt window. The title bar shows the path `C:\cygwin64\home\Slighten\program1.exe`. The command prompt displays the output `"Hello World!"` on the first line. Below it is a dashed line `-----`. The second line shows the message `Process exited after 0.01834 seconds with return value 0`. The third line shows the prompt `請按任意鍵繼續 . . .`.

```
C:\cygwin64\home\Slighten\program1.exe
"Hello World!"
-----
Process exited after 0.01834 seconds with return value 0
請按任意鍵繼續 . . .
```



作業 Part II

- 印出橫批的 "Hello World!" 不酷！我要印出直批。
- 檔名叫做 hw1_2.cpp
- 目的：學會適時換行



A screenshot of a Cygwin terminal window. The title bar shows the path 'C:\cygwin64\home\Slighten\program1.exe'. The terminal output displays 'Hello World!' with each character on a new line, forming a vertical column. Below the output, a dashed line separates it from the exit message: 'Process exited after 0.01657 seconds with return value 0' and '請按任意鍵繼續 . . .'. The background of the slide features a faint, stylized graphic of a person's head and shoulders.

```
C:\cygwin64\home\Slighten\program1.exe
H
e
l
l
o

W
o
r
l
d
!

-----
Process exited after 0.01657 seconds with return value 0
請按任意鍵繼續 . . .
```



作業 Part III

- 印出 hw1_1.cpp 的程式碼
- 檔名叫做 hw1_3.cpp
- 目的：學會印特殊符號



作業 Part IV (1/2)

- 事實上在 terminal 裡打 tree 可以看資料夾結構。

```
Slighten@SlightenCheng ttys000-bash 07:36 ~ $ tree codes/c++/  
codes/c++/  
├── hw1/  
│   ├── hw1_1.cpp  
│   ├── hw1_2.cpp  
│   ├── hw1_3.cpp  
│   └── hw1_4.cpp  
├── hw2/  
└── hw3/  
  
3 directories, 4 files
```

- 先不管顏色，寫一個程式 hw1_4.cpp 產生以下方結果。
- 你可能需要用到 `|`, `—`, `└`, `|`
- 目的：學會印特殊符號

```
codes/c++/  
├── hw1/  
│   ├── hw1_1.cpp  
│   ├── hw1_2.cpp  
│   ├── hw1_3.cpp  
│   └── hw1_4.cpp  
├── hw2/  
└── hw3/  
  
3 directories, 4 files
```



作業 Part IV (2/2)

- 如何在 Mac OS 裝 tree 指令？
- 先裝 Homebrew (<http://brew.sh/>)
 - Homebrew installs the stuff you need that Apple didn't.
- 成功後在 terminal 上打 **brew install tree**
- 以後可以用 brew 安裝你喜歡的東西



作業 Part V (OSX)

- 目的：印出 underlined text，學會查 stackoverflow
- 檔名：hw1_5.cpp
- 參考資料 (<http://stackoverflow.com/questions/24281603/c-underline-output>)：(OK for Linux)
 - 開啟 underline 模式："\33[4m"
 - 關閉 underline 模式："\33[0m"

```
Slighten@SlightenCheng tty000-bash 08:12 ~/codes/c++/hw1 $ ./a.out  
More info? Please click here.
```

