Course 01

Initial

開始之前

- 觀念一：簡單的英文單字但卻抽象
  - C 語言的指令都是簡單的英文單字(如：while、for)
  - 初學還是會覺得那些語法與規定很抽象
  - 這樣的反應是正常的

- 觀念二：語法的規定就是聖旨
  - 程式語法的規定除了遵守還是遵守
    - 每一個指令行後面都要有分號
  - 沒有可以自由創造的空間
  - 不能加分號的時候，就是不要加分號！宣告時必需要給初始值時，就是一定要給！
  - 背熟並且依照規定去使用就對了

一些故事

ANSI C

- ANSI C 的由來
  - K&R C 對於 C 的規範還是存在有模糊的灰色地帶
  - C 語言的發展變得百家爭鳴，產生些許的差異
  - 一統江湖
    - 美國國家標準局（American National Standard Institution）為了避免各開發廠商所發展的 C 語言產生差異，
    - 1980 年代訂定了一套 C 語言的國際標準語法 – ANSI C
    - 目前版本: C11
- 後續發展
  - 有關 C 語言的程式開發工具都會都支援符合ANSI C的語法
    - Linux、Unix以及微軟的 Windows 作業系統都有 C 的編譯環境
  - C 語言的設計也影響了許多後來的程式語言
    - Objective-C、C++、Java 與 C# 等。

- C++ 源自於 Bjarne Stroustrup 在 1979 年的「C with Classes」的構想
  - C 語言中加入類別的概念
  - 建立一個讓大型軟體開發變得容易而且又有效能的程式語言
  - 選擇 C 的原因
    - 執行效能高、可移植性強而且適於各種用途

- C++ 正式定名
  - 1983年 Rick Mascitti 正式取名為 (C plus plus)
  - ++ 在 C 的語法中代表變數內容增加的意思
  - 「+」也代表電腦程式提升的意義。
  - 新增功能被加入：虛擬函式(virtual function)、運算子多載(operator overloading)、參照(references)…等等

C 語言的特性

- 執行效能高與可攜性高
  - C 語言所開發的程式，只需（甚至不用）做部分修改就能在不同的平台(如：Liunx 或 Unix)上被編譯且執行
- ANSI C 只規範了 44個關鍵字（C11/C1x)
  - 變數與函式命名有更多彈性
- 字元（char）與整數（integer）可直接轉換
  - 邏輯判斷的運用更加的方便
- 非常適合
  - 由上而下的設計概念（top-down planning)
  - 結構化程式設計（structured programming）
  - 模組化的設計（modular design）

- 透過結構（struct）來組合不同的變數
  - 可衍生出更有用的資料型別
- 函式的運用更有彈性
  - 函式在呼叫時，可以在引數上使用傳值呼叫(call-by-value)的傳值(pass by value)或傳指標(pass by pointer)
- 巨集與前置處理器（preprocessor）
  - 讓程式撰寫更加簡潔而且容易維護
- 指標（pointer）與動態記憶體存取
  - 記憶體的低階控制變得更方便而且有效率

QT

- Write once, compile everywhere

- 是一個跨平台的C++應用程式開發框架
  - OPIE、Skype、VLC media player、Adobe Photoshop Elements、VirtualBox與Mathematica
  - Autodesk 、歐洲太空總署、夢工廠、Google、HP、KDE、盧卡斯影業、西門子公司、富豪集團, 迪士尼動畫製作公司、三星集團、飛利浦、Panasonic

- 分為**商用版**及**開放源始碼版**
  - 開放源始碼版採用LGPL授權

支援平臺

- 跨平臺
  - 相同的程式碼可以在任何支援的平台上編譯與執行，而不需要修改原始碼。
  - 會自動依平台的不同，表現平台特有的圖形介面風格。

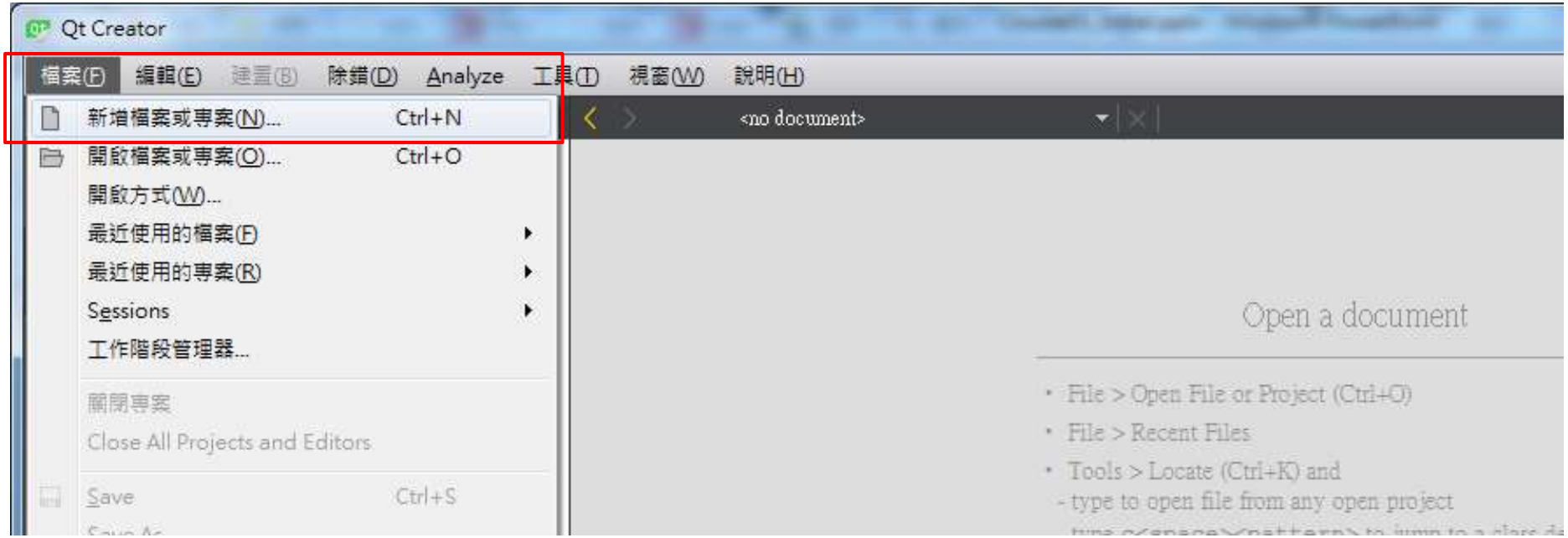
- 內部支援
  - Linux/X11--Window System
  - Mac--Apple Mac OS X。
  - Windows--用於Microsoft Windows。
  - 其他
    - Embedded Linux、Windows CE / Mobile、Symbian、Maemo/MeeGo、Wayland

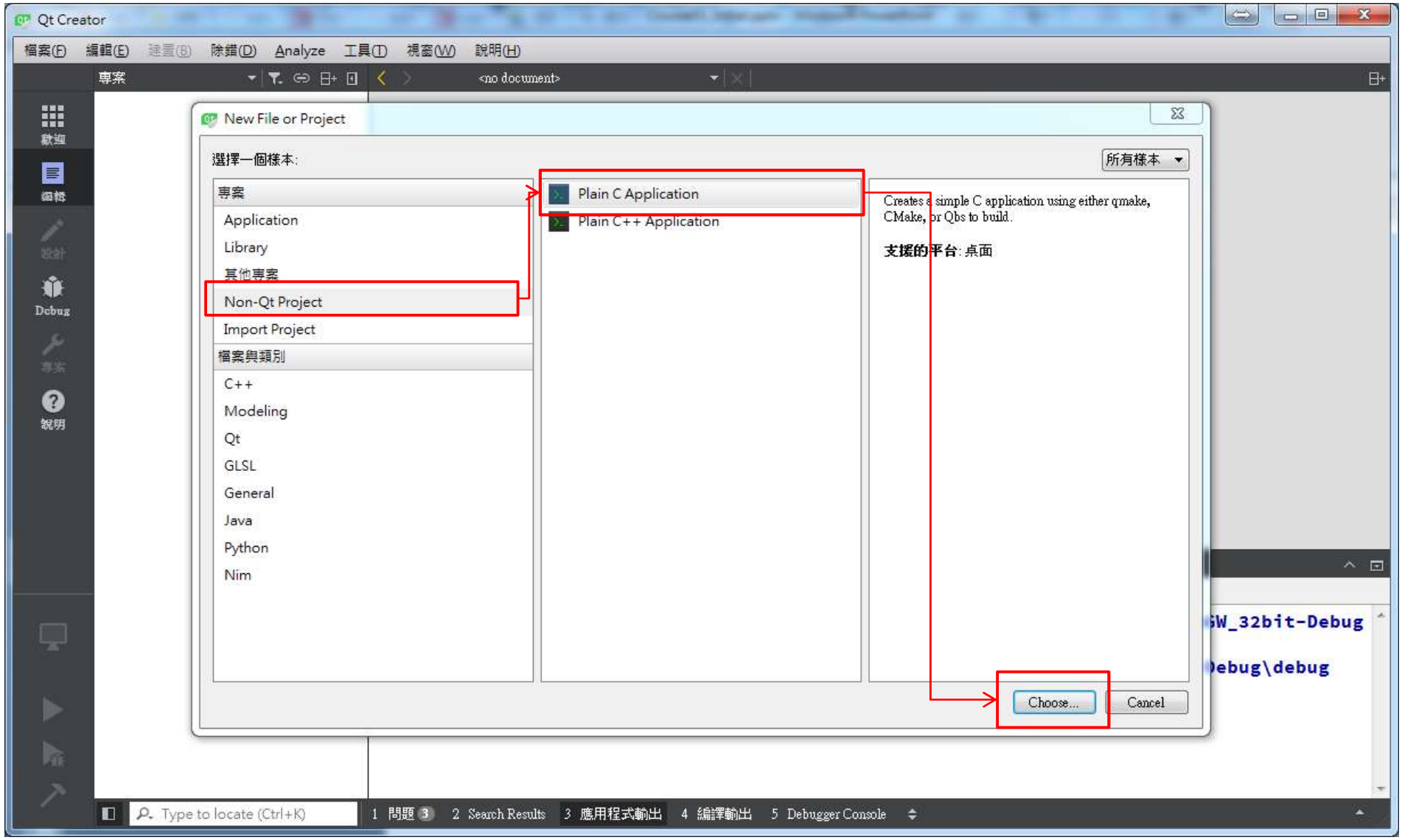
- 透過QT everywhere 將程式移至不同作業系統執行

第一個程式 – hello world

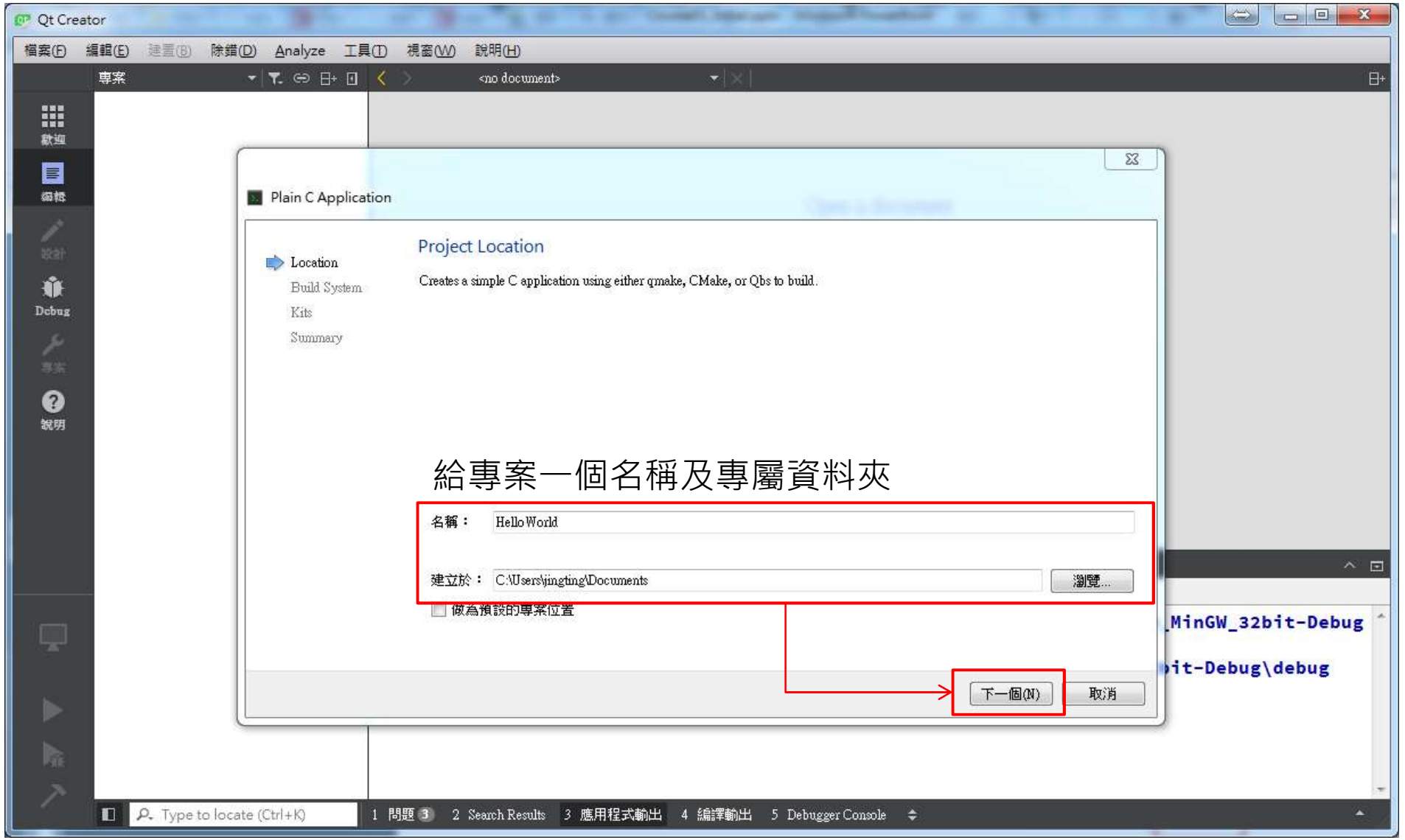
```
#include <stdio.h>

int main(int argc, char *argv[])
{
    printf("Hello World!\n");
    return 0;
}
```

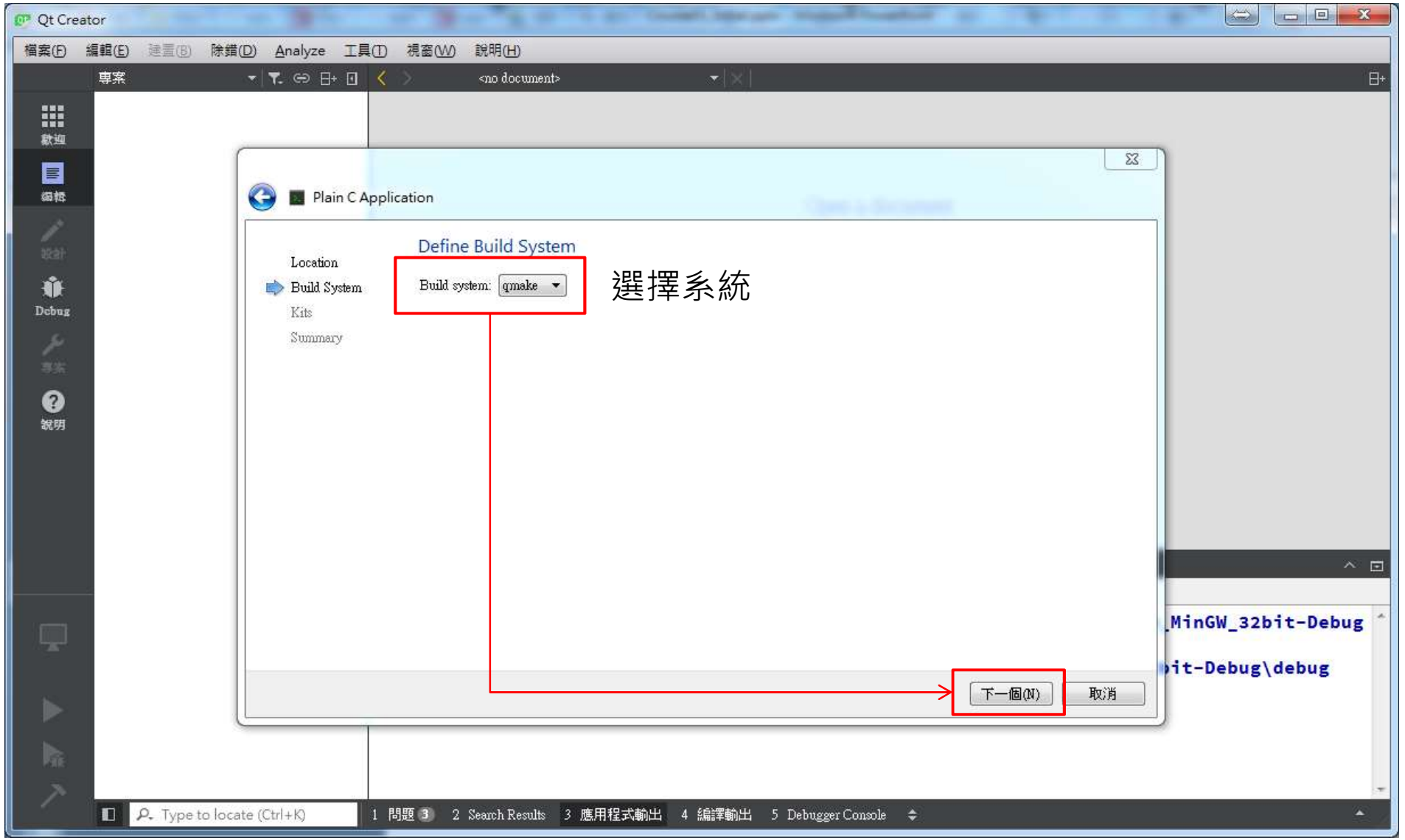

Qt Creator
檔案(F)
編輯(E)
建置(B)
除錯(D)
Analyze
工具(T)
視窗(W)
說明(H)
新增檔案或專案(N)... Ctrl+N
開啟檔案或專案(O)... Ctrl+O
開啟方式(W)...
最近使用的檔案(F)
最近使用的專案(R)
Sessions
工作階段管理器...
關閉專案
Close All Projects and Editors
Save Ctrl+S
<no document>
Open a document
File > Open File or Project (Ctrl+O)
File > Recent Files
Tools > Locate (Ctrl+K) and
- type to open file from any open project



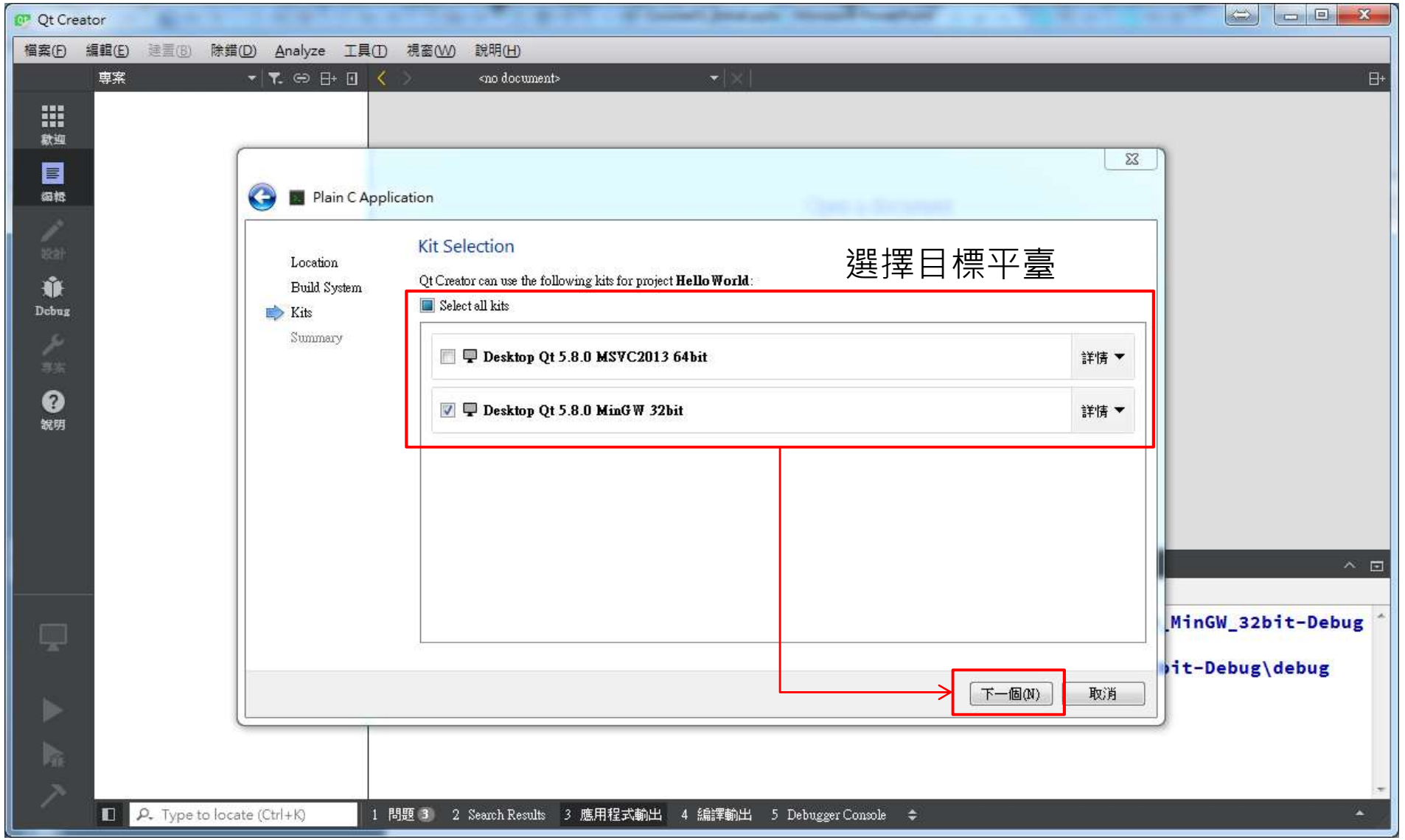
/Course/1-01-0011-01/c/Course01_Initial

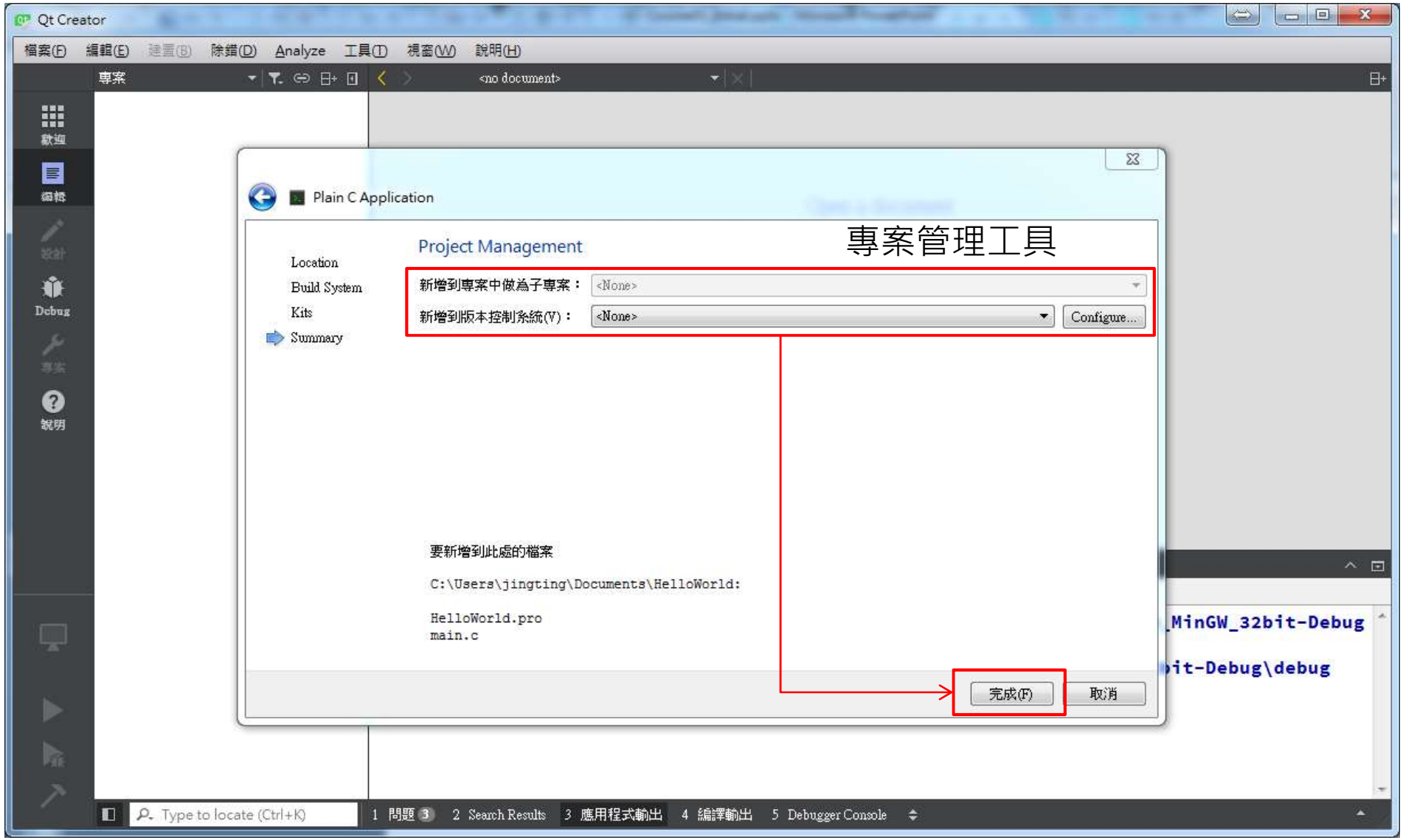


/Course/1-01-0011-01/c/Course01_Initial

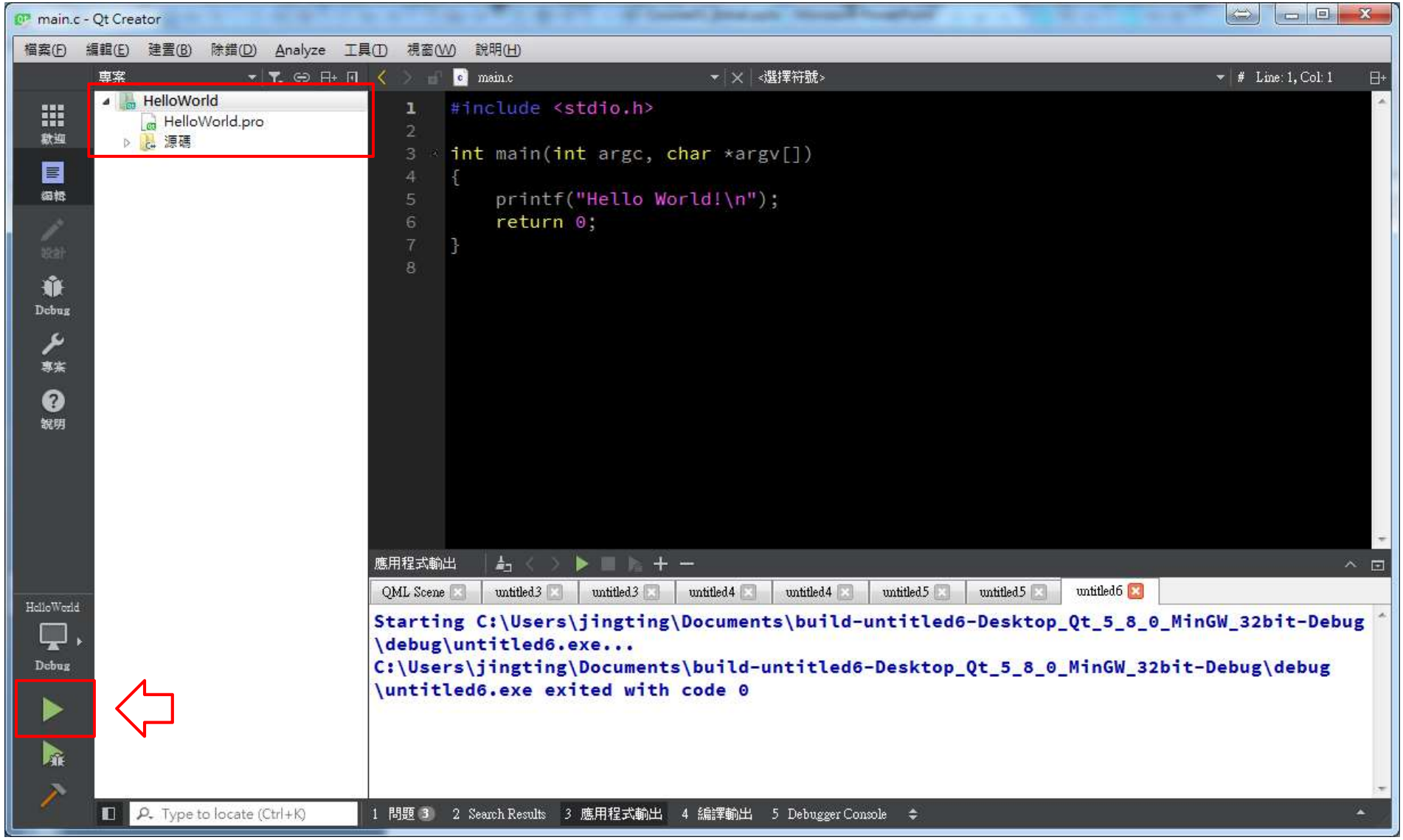


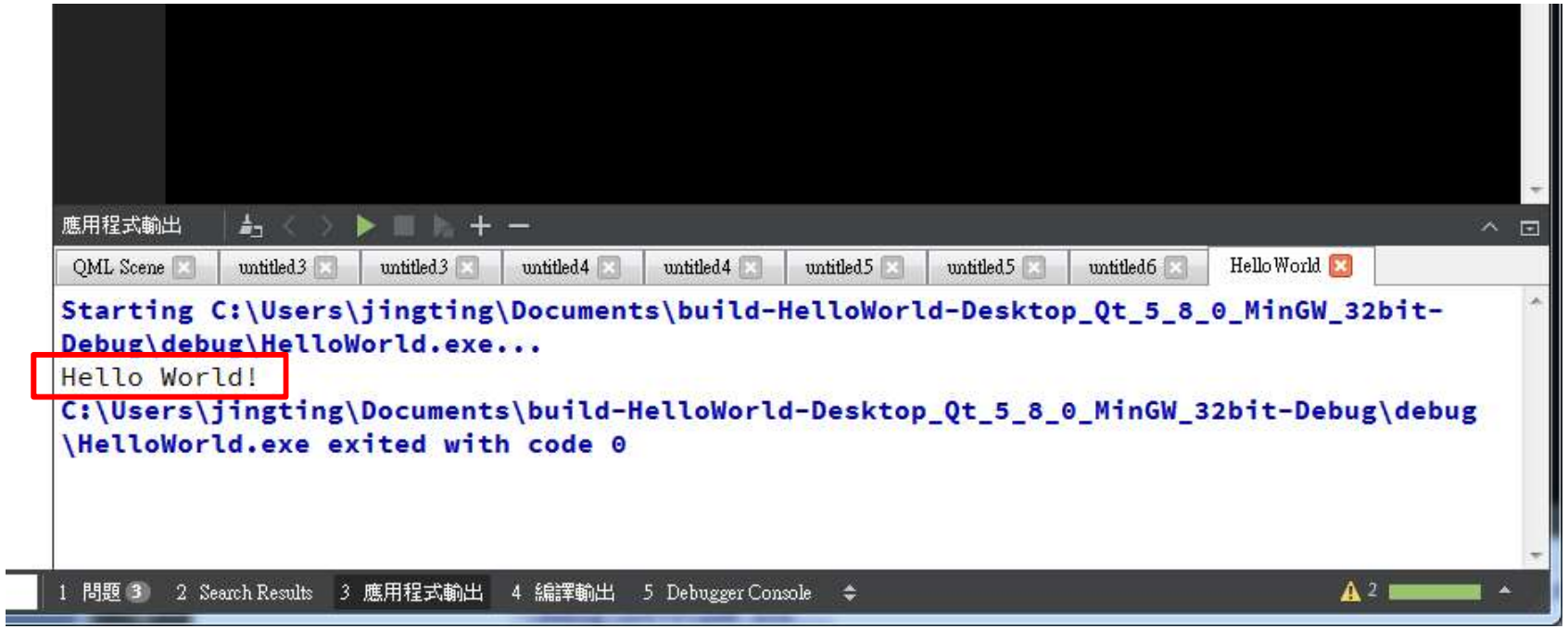
/Course/1-01-0011-01/c/Course01_Initial


Qt Creator
檔案(F) 編輯(E) 建置(B) 除錯(D) Analyze 工具(T) 視窗(W) 說明(H)
專案
<no document>
歡迎
編輯
設計
Debug
專案
說明
Plain C Application
Location
Build System
Kits
Summary
Kit Selection
Qt Creator can use the following kits for project Hello World:
Select all kits
Desktop Qt 5.8.0 MSVC2013 64bit
詳情
Desktop Qt 5.8.0 MinGW 32bit
詳情
選擇目標平臺
下一個(N)
取消
_MinGW_32bit-Debug
it-Debug\debug
Type to locate (Ctrl+K)
1 問題 3
2 Search Results
3 應用程式輸出
4 編譯輸出
5 Debugger Console

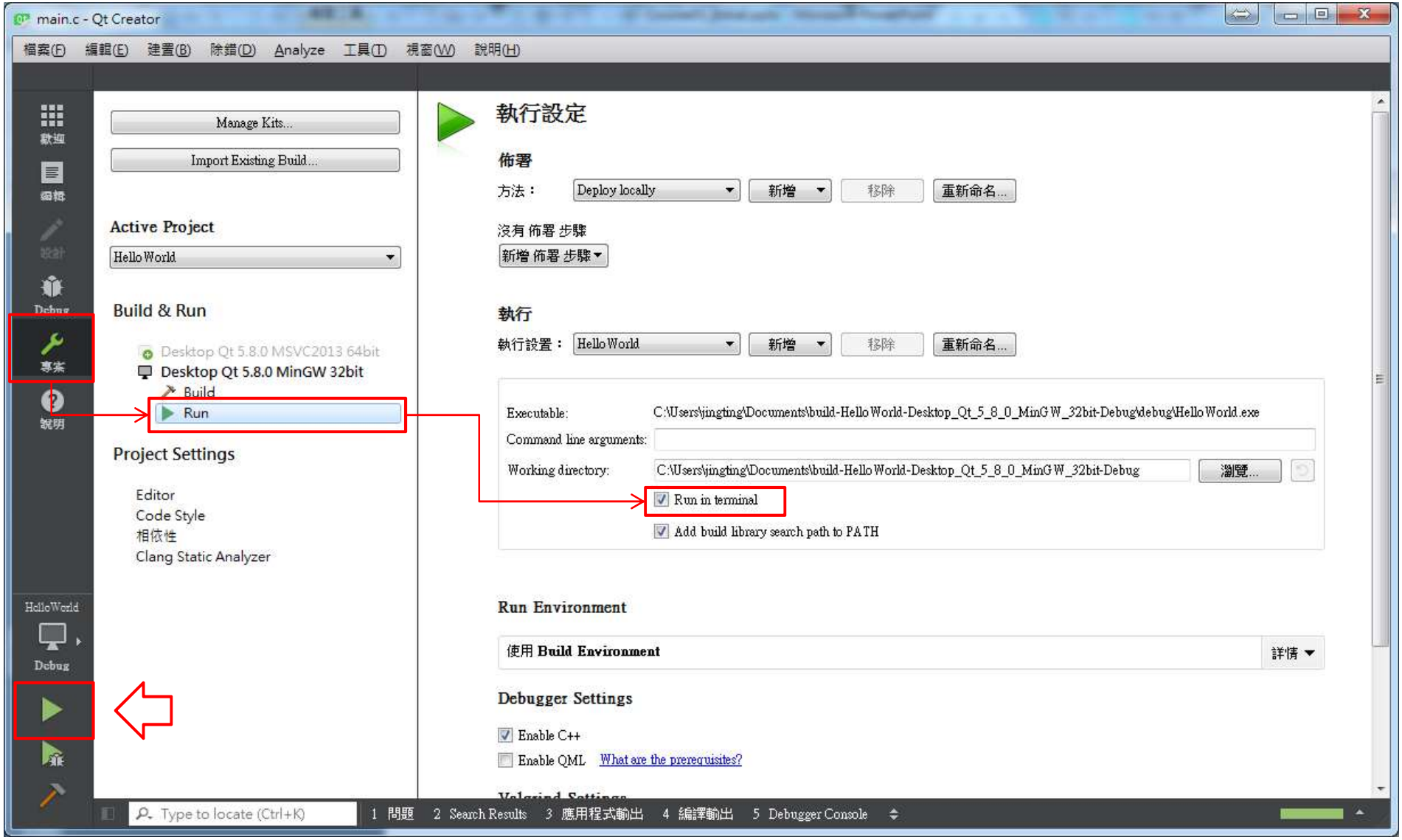

Qt Creator
檔案(F) 編輯(E) 建置(B) 除錯(D) Analyze 工具(T) 視窗(W) 說明(H)
專案
<no document>
歡迎
編輯
設計
Debug
專案
說明
Plain C Application
Project Management
專案管理工具
Location
Build System
Kits
Summary
新增到專案中做為子專案：
<None>
新增到版本控制系統(V)：
<None>
Configure...
要新增到此處的檔案
C:\Users\jingting\Documents\HelloWorld:
HelloWorld.pro
main.c
完成(F)
取消
MinGW_32bit-Debug
it-Debug\debug
Type to locate (Ctrl+K)
1 問題 3
2 Search Results
3 應用程式輸出
4 編譯輸出
5 Debugger Console

執行看看

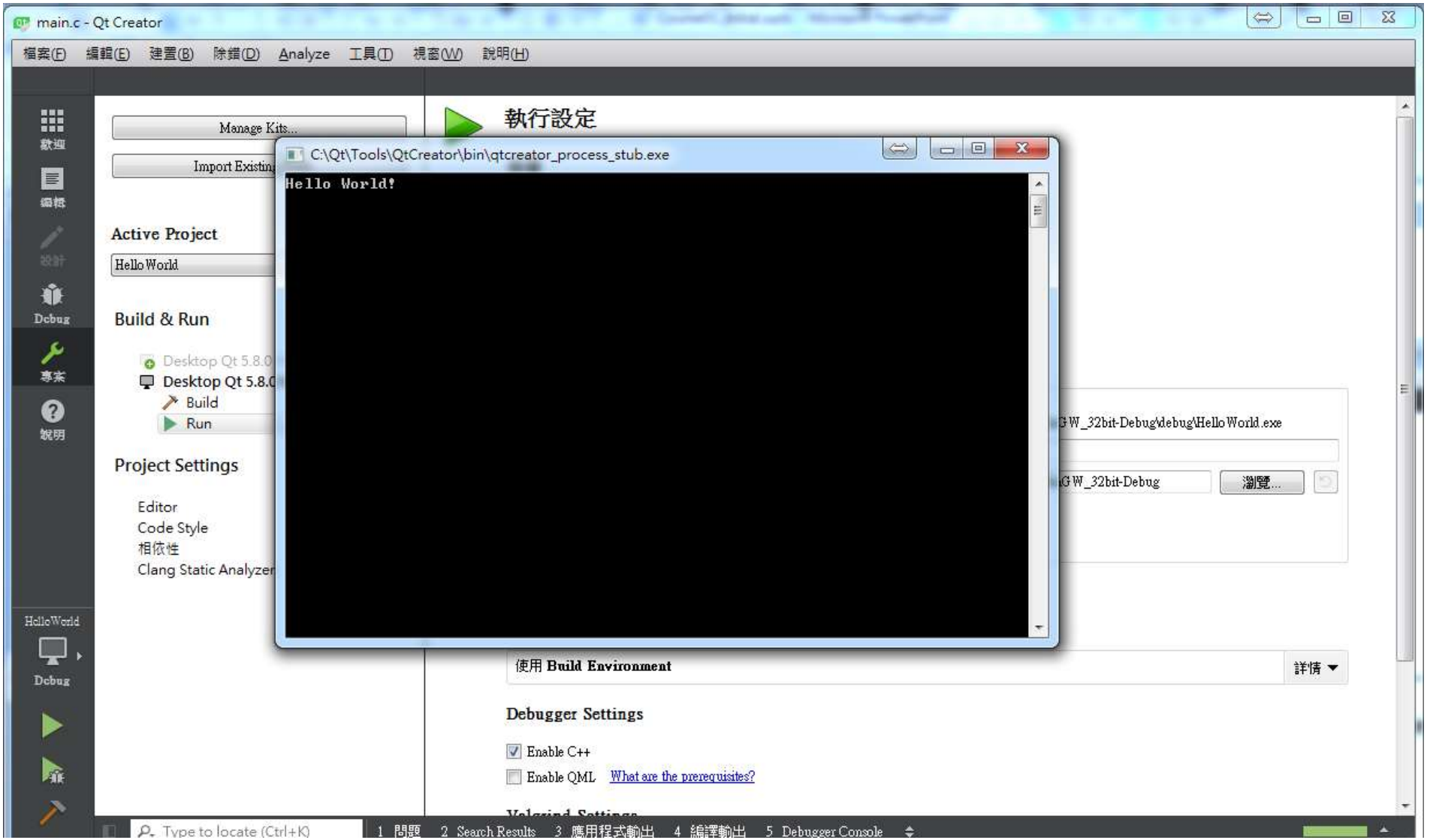

main.c - Qt Creator
檔案(F) 編輯(E) 建置(B) 除錯(D) Analyze 工具(T) 視窗(W) 說明(H)
專案
HelloWorld
HelloWorld.pro
源碼
main.c
<選擇符號>
Line: 1, Col: 1
#include <stdio.h>
int main(int argc, char *argv[])
{
printf("Hello World!\n");
return 0;
}
歡迎
編輯
設計
Debug
專案
說明
HelloWorld
Debug
應用程式輸出
QML Scene
untitled3
untitled3
untitled4
untitled4
untitled5
untitled5
untitled6
Starting C:\Users\jingting\Documents\build-untitled6-Desktop_Qt_5_8_0_MinGW_32bit-Debug\debug\untitled6.exe...
C:\Users\jingting\Documents\build-untitled6-Desktop_Qt_5_8_0_MinGW_32bit-Debug\debug\untitled6.exe exited with code 0
Type to locate (Ctrl+K)
1 問題 3
2 Search Results
3 應用程式輸出
4 編譯輸出
5 Debugger Console


應用程式輸出
QML Scene
untitled3
untitled3
untitled4
untitled4
untitled5
untitled5
untitled6
Hello World
Starting C:\Users\jingting\Documents\build-HelloWorld-Desktop_Qt_5_8_0_MinGW_32bit-Debug\debug\HelloWorld.exe...
Hello World!
C:\Users\jingting\Documents\build-HelloWorld-Desktop_Qt_5_8_0_MinGW_32bit-Debug\debug\HelloWorld.exe exited with code 0
1 問題 3
2 Search Results
3 應用程式輸出
4 編譯輸出
5 Debugger Console
2

修改執行設定



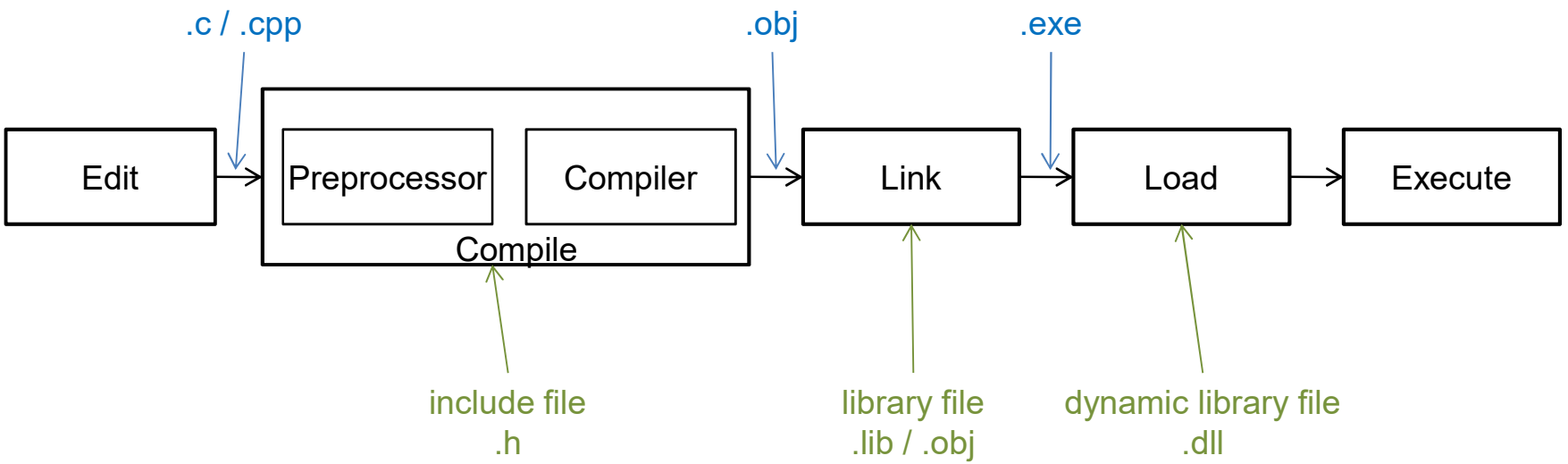
/Course/1-01-0011-01/c/Course01_Initial



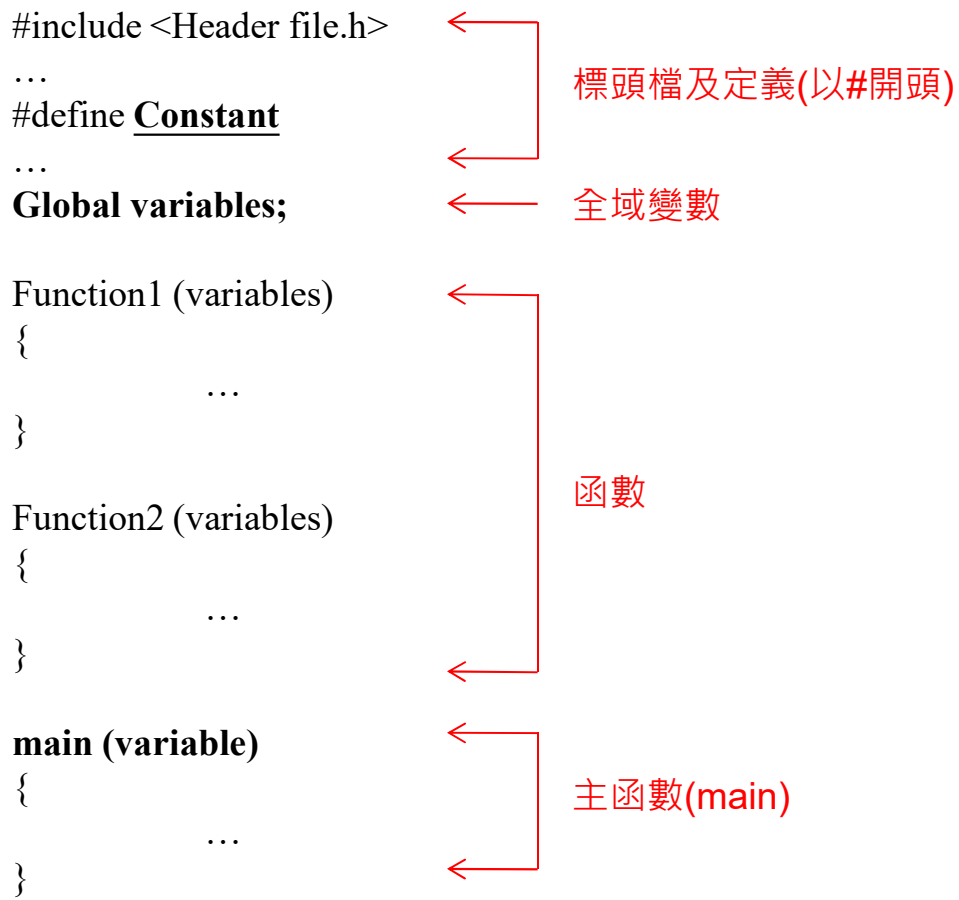
/Course/1-01-0011-01/c/Course01_Initial

執行程式的五個步驟

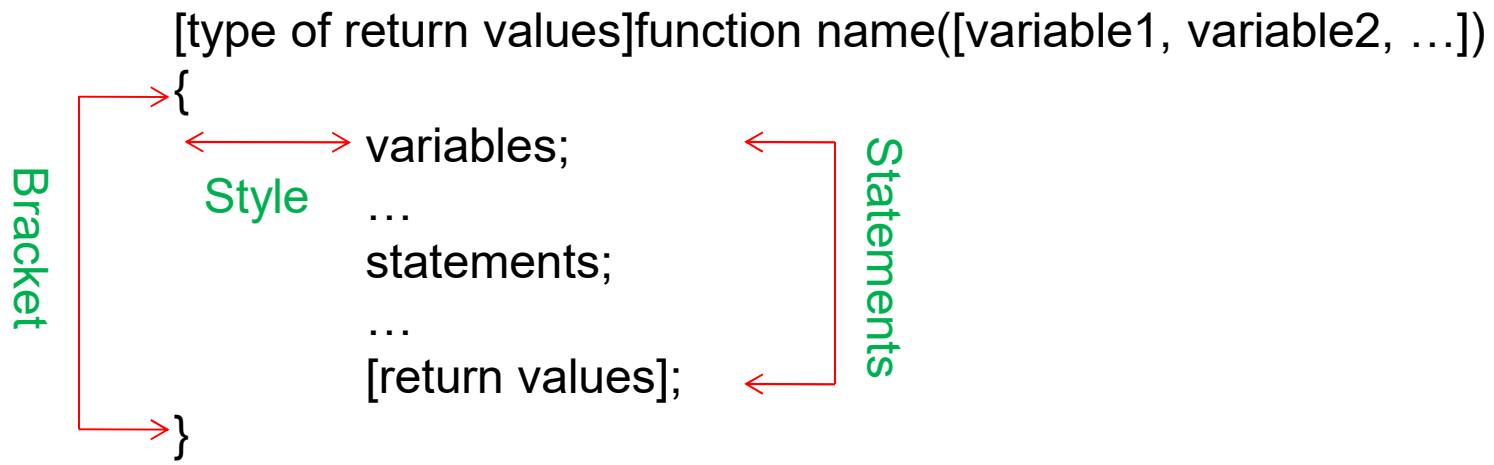
- 編輯 C or Cpp
  - 整合環境開發器或是文字編輯器
  - QT, Visual Studio, Dev-C
- 編譯 Compile
  - 物件檔
  - Preprocessor
  - Compiler
- 連結 Linker
  - Link object code and library
  - exe file
- 提取 Loader
  - Load exe file to memory
  - Connect dll file
- 執行 Execute


.c / .cpp
.obj
.exe
Edit
Preprocessor
Compiler
Compile
Link
Load
Execute
include file
.h
library file
.lib / .obj
dynamic library file
.dll

C語言程式結構


#include <Header file.h>
…
#define **Constant**
…
Global variables;
Function1 (variables)
{
…
}
Function2 (variables)
{
…
}
main (variable)
{
…
}
標頭檔及定義(以#開頭)
全域變數
函數
主函數(main)

函數結構


[type of return values]function name([variable1, variable2, …])
{
variables;
…
statements;
…
[return values];
}
Bracket
Style
Statements

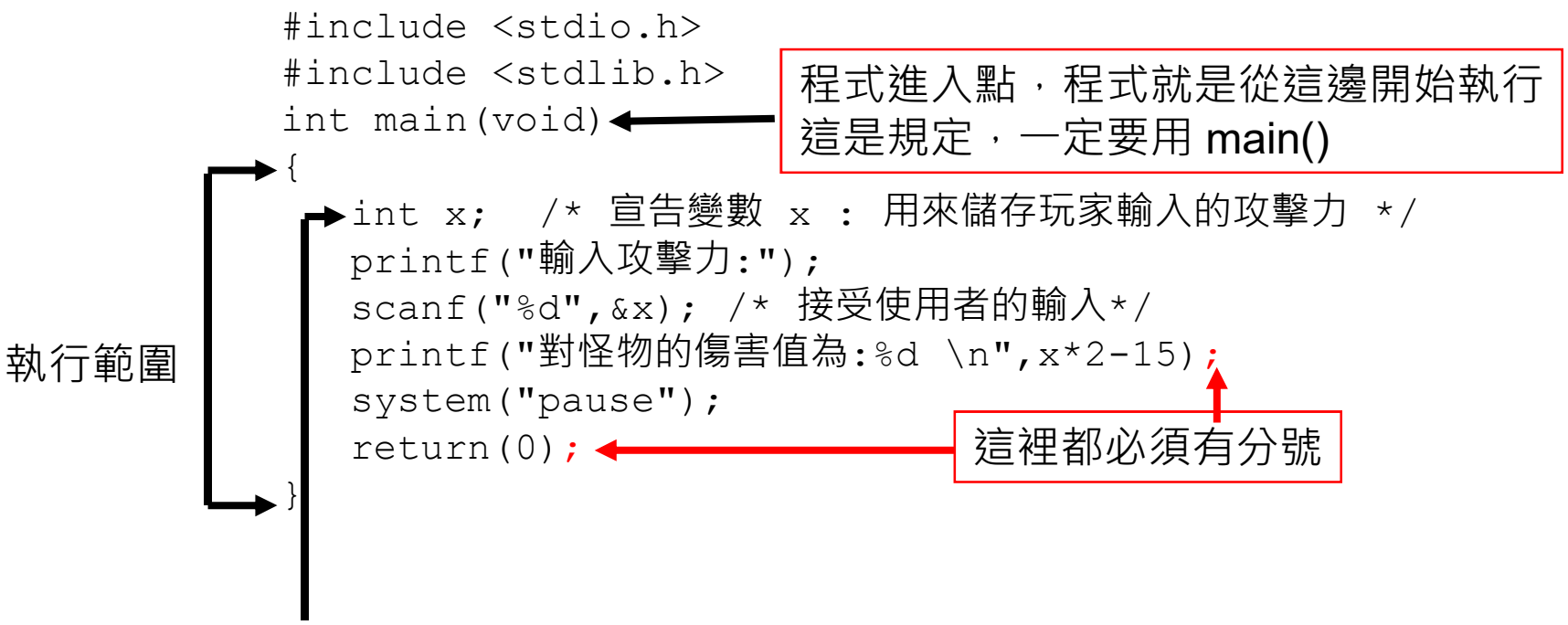
看一個小例子

```c
#include <stdio.h>

int main(int argc, char *argv[])
{
    int x; /* 宣告變數 */
    printf("Attack:");
    scanf("%d",&x); /* 接收使用者輸入的資料 */
    printf("You kill the monster:%d \n",x*2-15);
    return 0;
}
```

在 C 語言的世界裡，英文的字母的大小寫是不一樣的。
例如：int cat; int Cat; int CAT; 代表三個不同的變數

程式結構



int x;

代表宣告了一個名叫做 x 的變數，它只能儲存整數(integer)

- 程式解說
  - 註解 ─ 寫在 /* */ 之間的是註解
  - 註解是給人看的，程式開發軟體是不會去處理註解的
  - 記得 /* */ 一定要成對出現

養成良好的撰寫習慣
縮排與註解

縮排

使用 tab 或空白鍵內縮

```c
#include <stdio.h>
#include <stdlib.h>
#define NUM_PER_LINE 6
int main(void)
{
    int i,j,halfi;
    int flag;
    int num=0;
    for(i=2;i<=100;i++)
    {
        flag=1; halfi=i/2;
        if(i%j==0)
        {
            flag=0;
            break;
        }
        if(flag)
        {
            printf("%3d",i);
            num++;
            if(num%NUM_PER_LINE==0)
                printf("\n");
        }
    }
    system("pause");
    return 0;
}
```

層次分明，容易閱讀

```c
#include <stdio.h>
#include <stdlib.h>
#define NUM_PER_LINE 6
int main(void)
{
int i,j,halfi;
int flag;
int num=0;
for(i=2;i<=100;i++)
{
flag=1; halfi=i/2;
if(i%j==0)
{
flag=0;
break;
}
if(flag)
{
printf("%3d",i);
num++;
if(num%NUM_PER_LINE==0)
printf("\n");
}
}
system("pause");
return 0;
}
```

沒有層次，較難閱讀

註解

- 程式碼本身就是最好的說明
- 註解
  - /* */
    - 範圍所包含的文字都成是註解
    - 在這之間有換行一樣都會被當成是註解
  - //
    - C11開始也能支援以「 // 」為開頭的註解撰寫方式
    - 寫在「//」後面的文字都會被當成註解
    - 但範圍就是它它身後那一行文字

```
int x;        /* 宣告變數 x:
                 用來儲存玩家輸入的攻擊力 */
printf("輸入攻擊力:");
scanf("%d",&x); // 接受使用者的輸入
```

/Course/1-01-0011-01/c/Course01_Initial

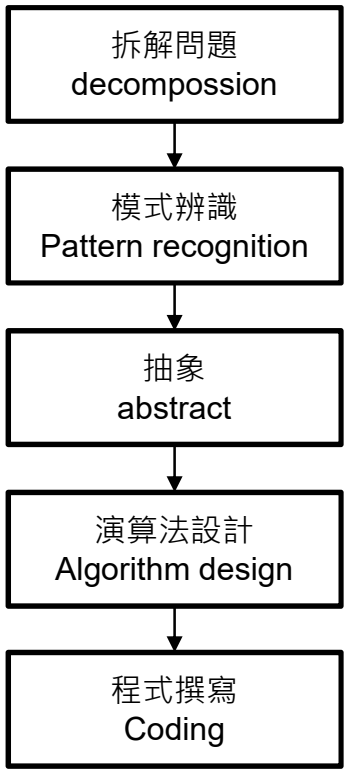
運算思維與程式邏輯

Computational Thinking
運算思維—讓電腦知道你在想什麼

運算思維 (Computational Thinking)

- 運算思維是一個思考的程序。它的目的是闡明問題，並呈現其解決方案，因而讓「運算器」(包括機器與人) 能夠有效率地執行。 -- 微軟周以真 (Jeannette Wing)
- 軟體工程師的工作就是轉換，把經營或管理的模糊概念轉換成程式碼。 --軟體工程師 Weiliang Chew

- 培養運算思維能訓練邏輯思考、提升問題解決能力。你會學習如何**拆解問題**，一步一步**找到最有效的解決方法**。學習運算思維也有助於了解電腦的運作模式，也就是電腦如何「思考」和執行指令。有了這些知識，你就更能有效運用科技解決問題。



理解問題→將大問題拆解成小問題

檢視每個小問題是否存在同樣的趨勢(Pattern)

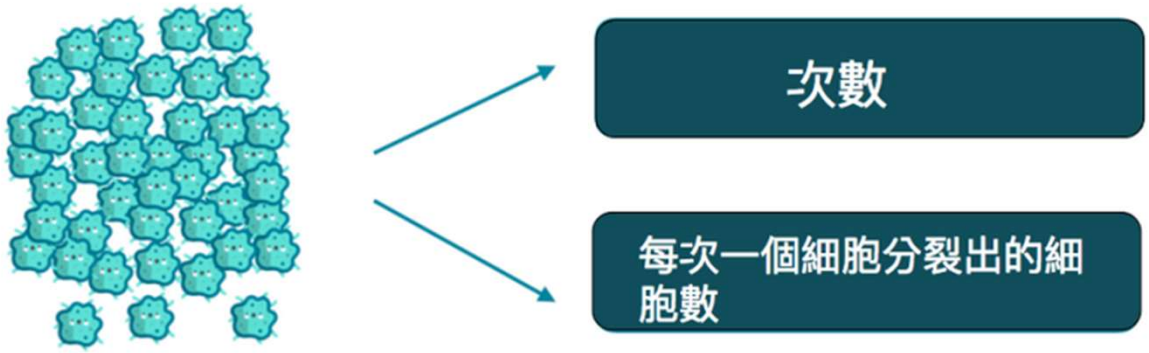
丟掉Pattern，找出其中的重點細節

設計演算法

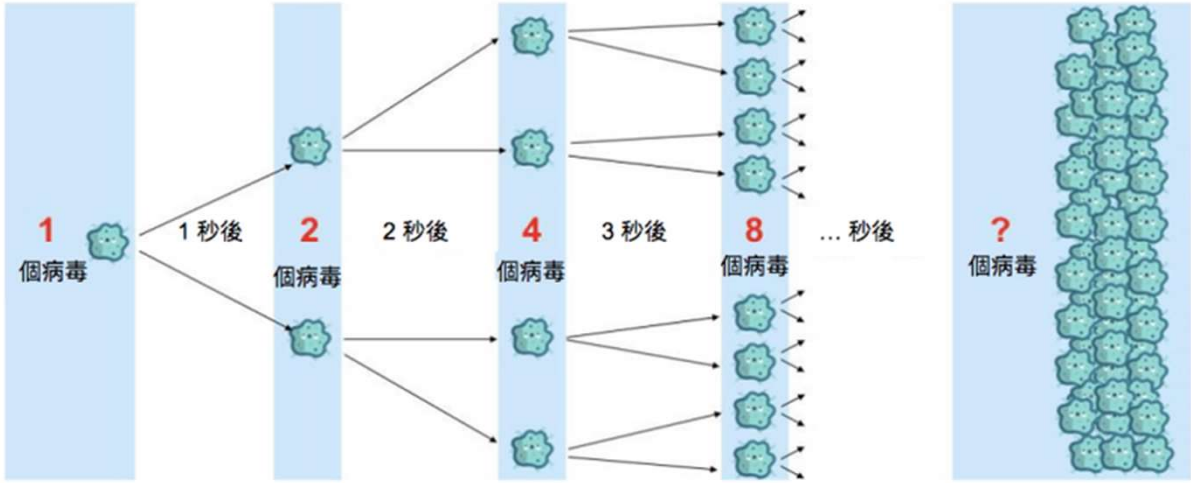
寫程式

以病毒擴散為例，我們想知道病毒擴散的數量

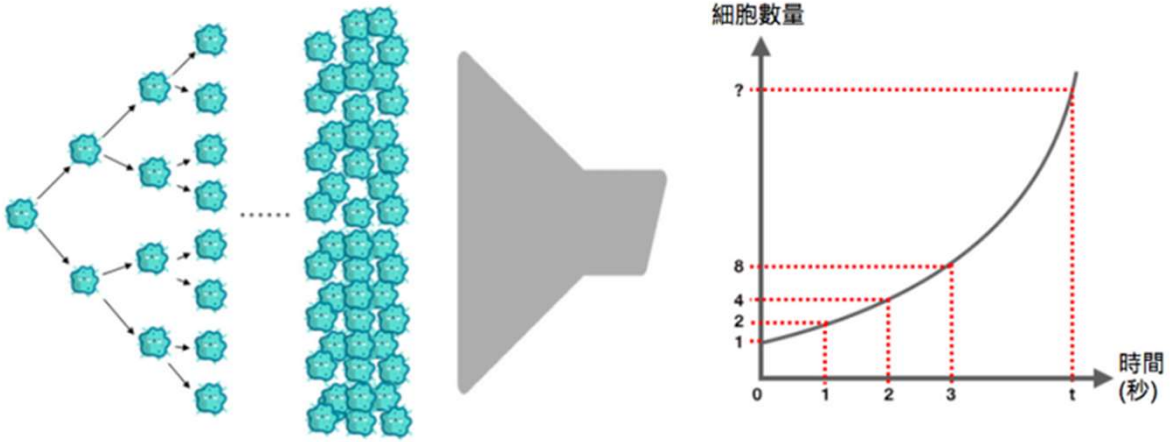
1. 拆解問題：病毒擴散的數量與分裂次數有關，而**分裂次數**與**時間**有關



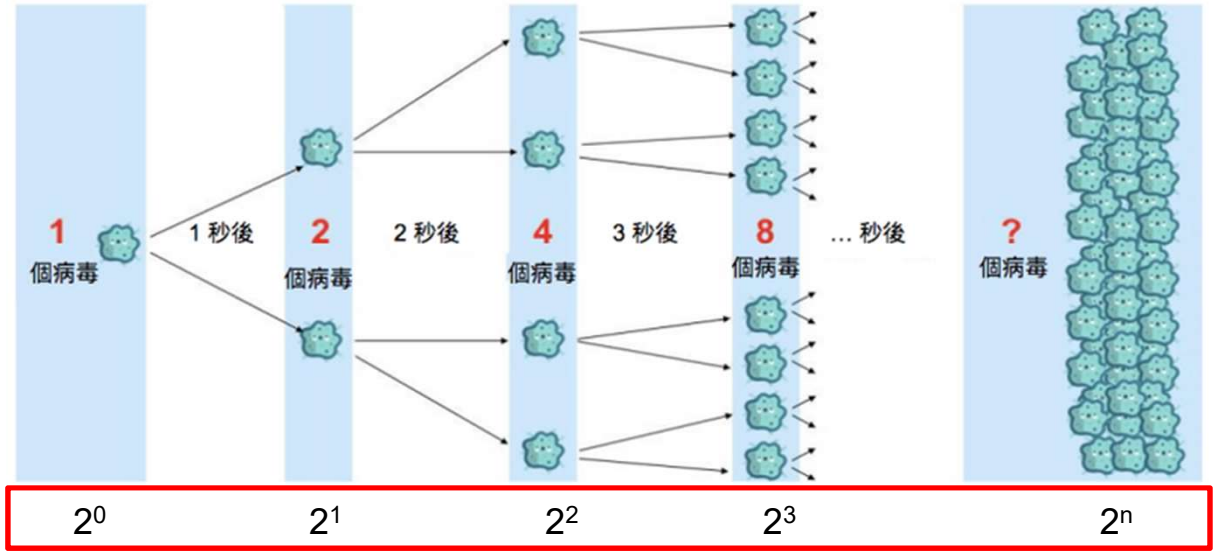
2. 模式辨識 Pattern recognition



病毒成長數量非常規律

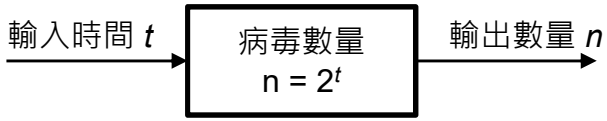


3. 抽象化 Abstract



病毒成長與時間有關，因此定義為 2^t

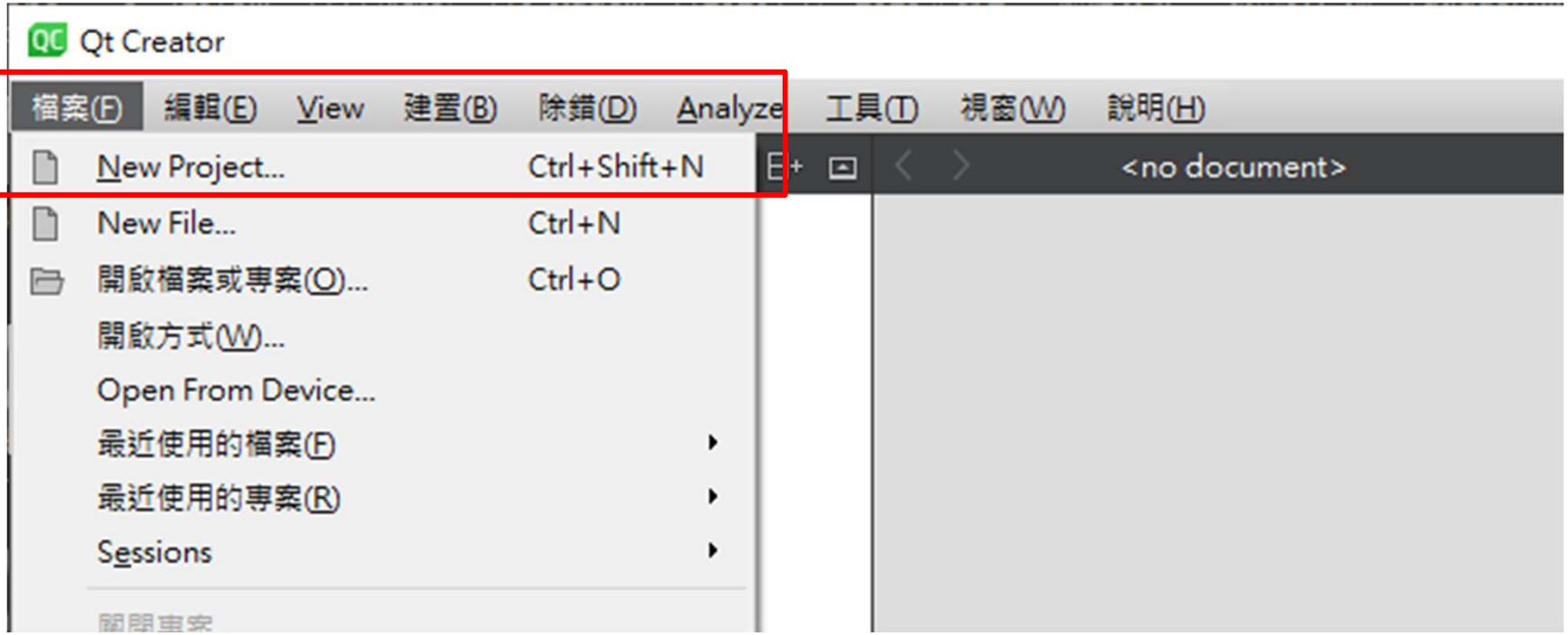
4. 演算法設計

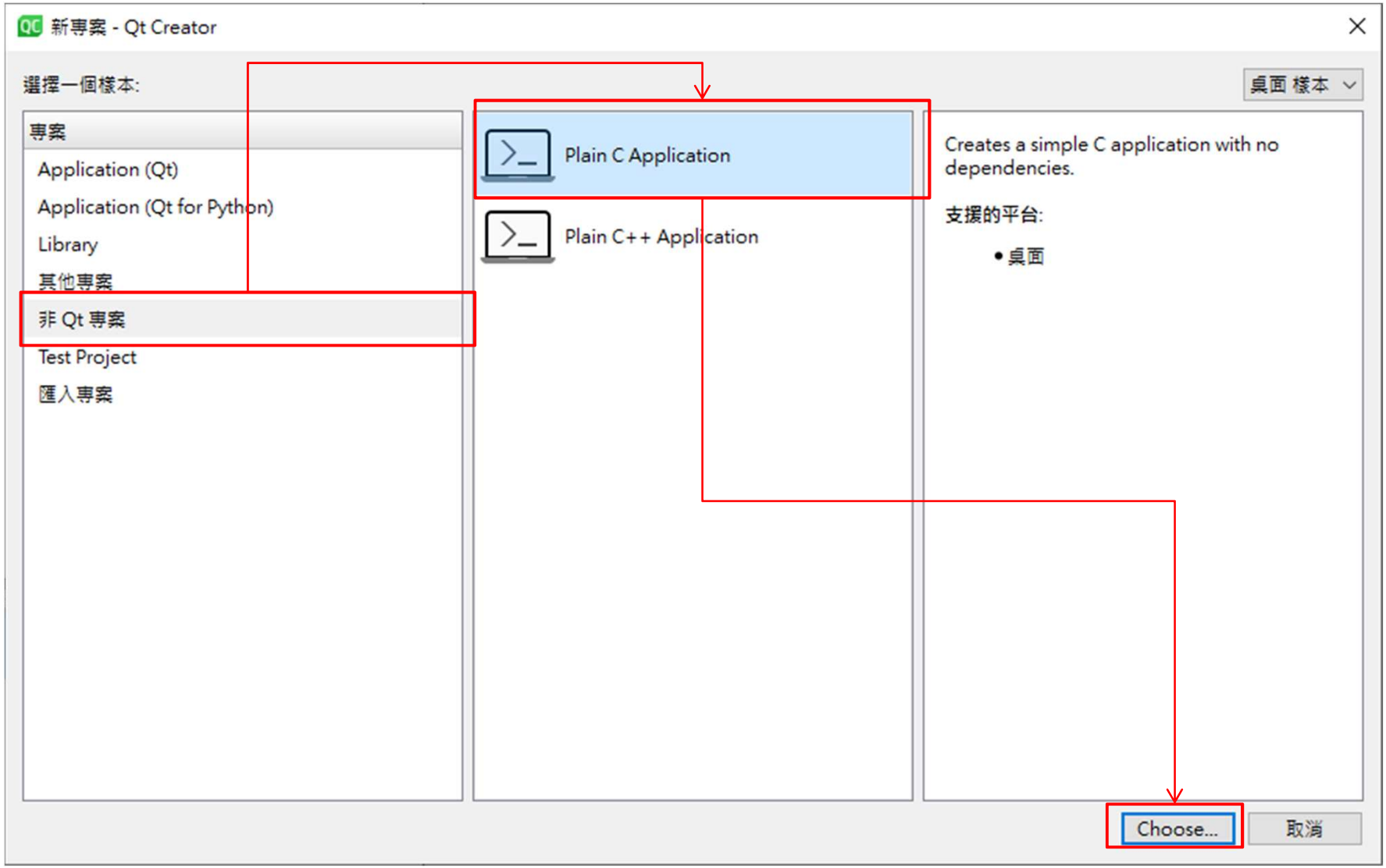


5. 寫程式

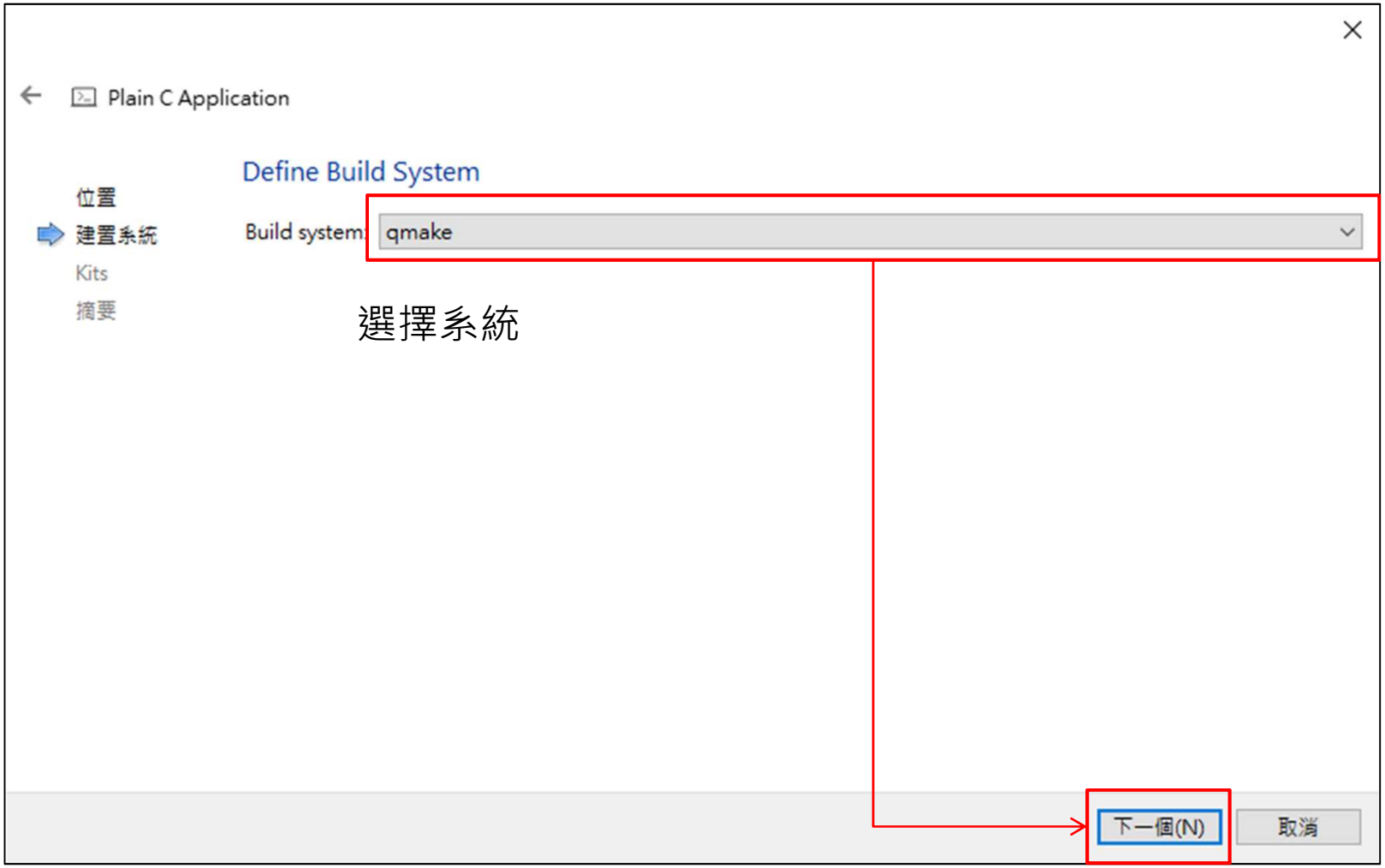
變數:

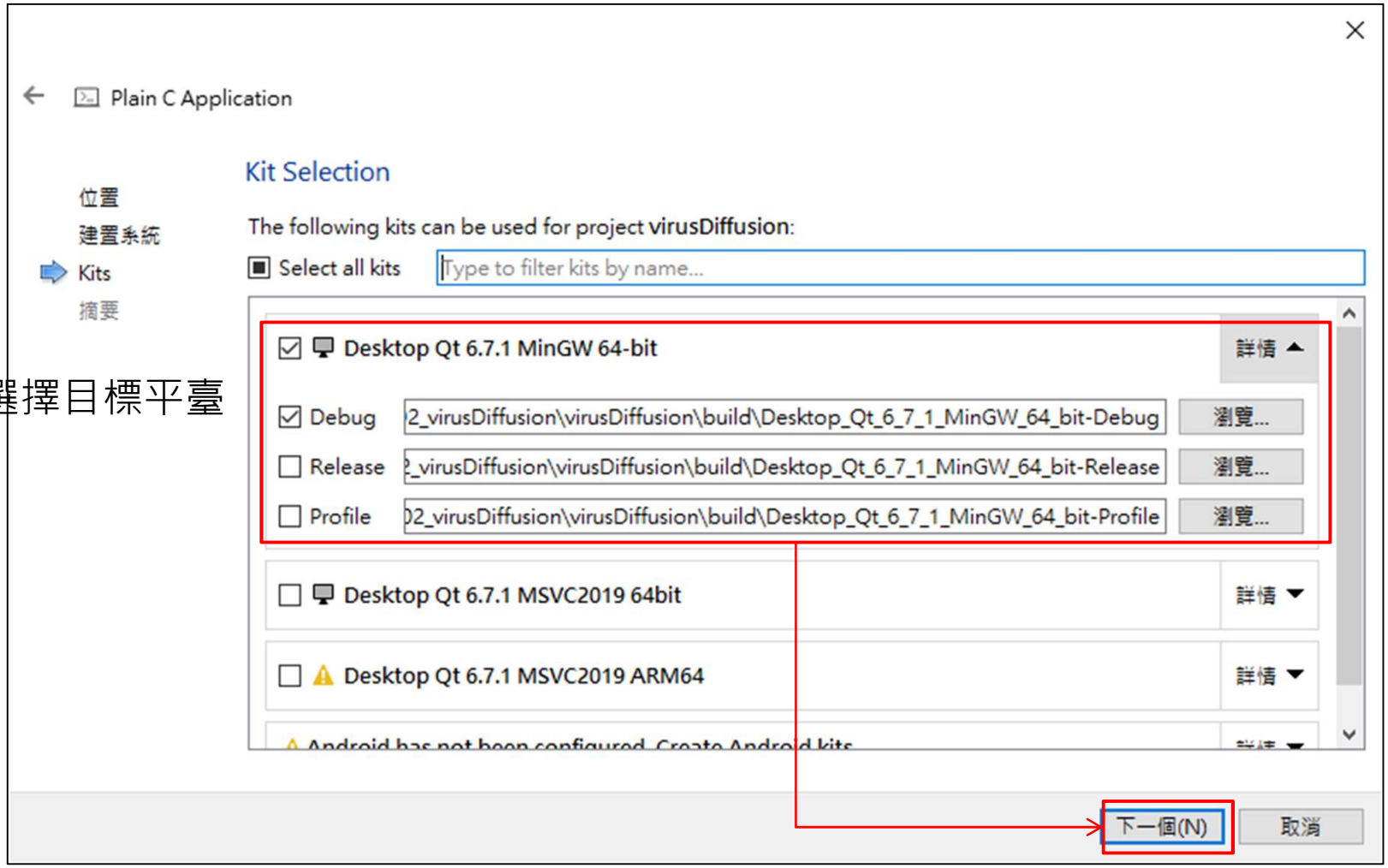
- 輸入變數: *t* → 整數型態
- 輸出變數: *n* → 整數或長整數型態


Qt Creator
檔案(F)
編輯(E)
View
建置(B)
除錯(D)
Analyze
工具(T)
視窗(W)
說明(H)
New Project... Ctrl+Shift+N
New File... Ctrl+N
開啟檔案或專案(O)... Ctrl+O
開啟方式(W)...
Open From Device...
最近使用的檔案(F)
最近使用的專案(R)
Sessions
<no document>

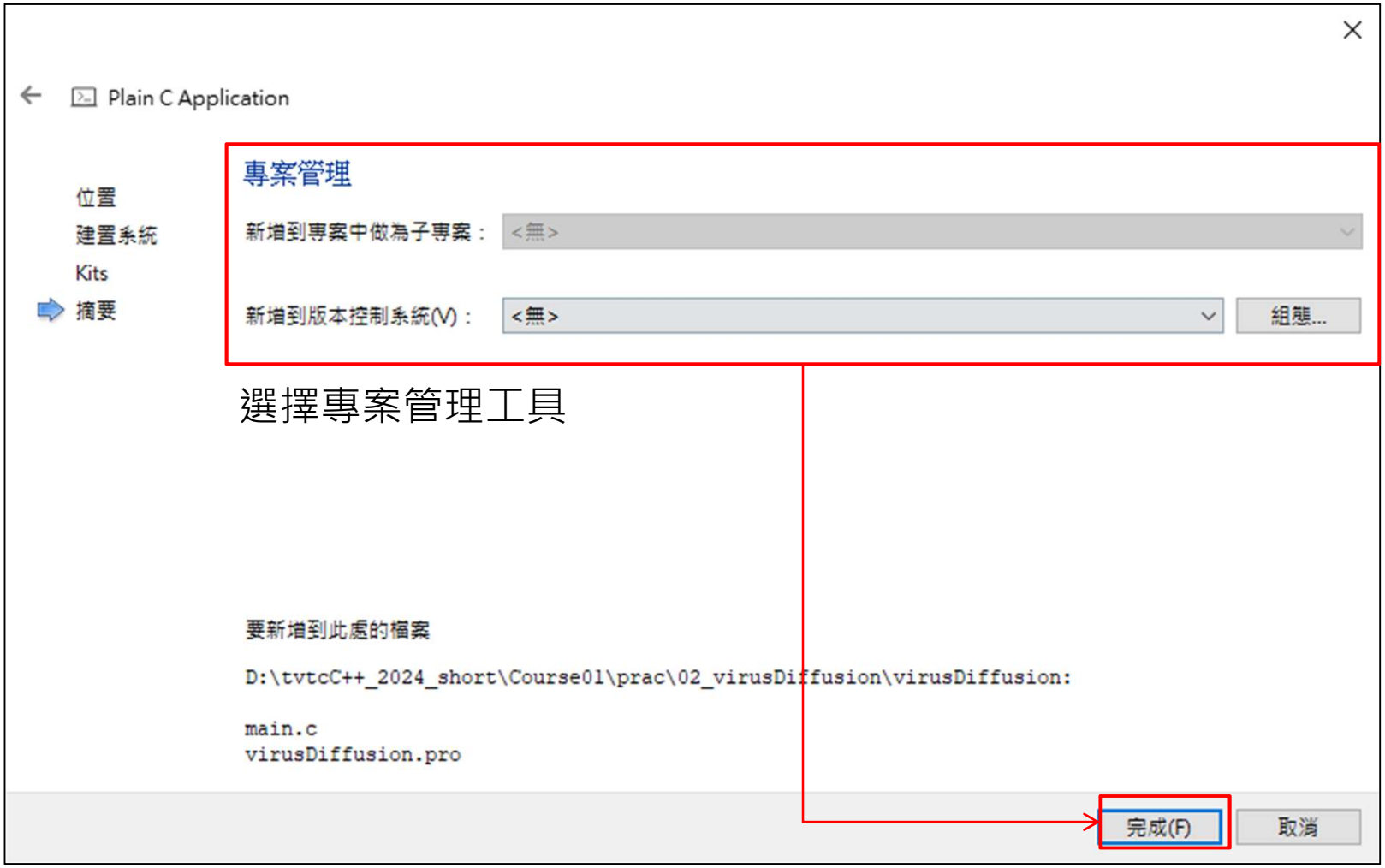


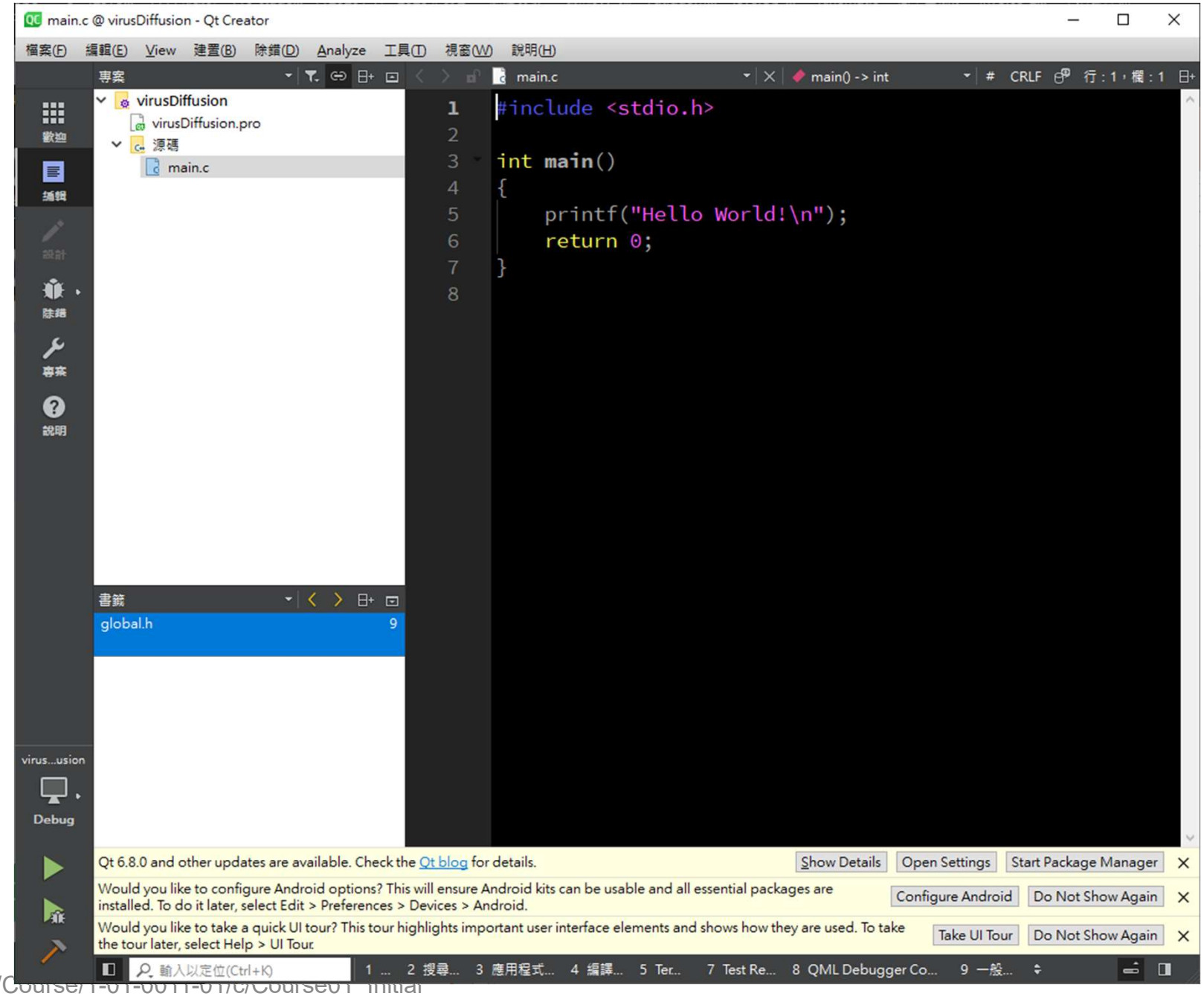
/Course/1-01-0011-01/c/Course01_Initial


Plain C Application
Define Build System
位置
建置系統
Kits
摘要
Build system:
qmake
選擇系統
下一個(N)
取消

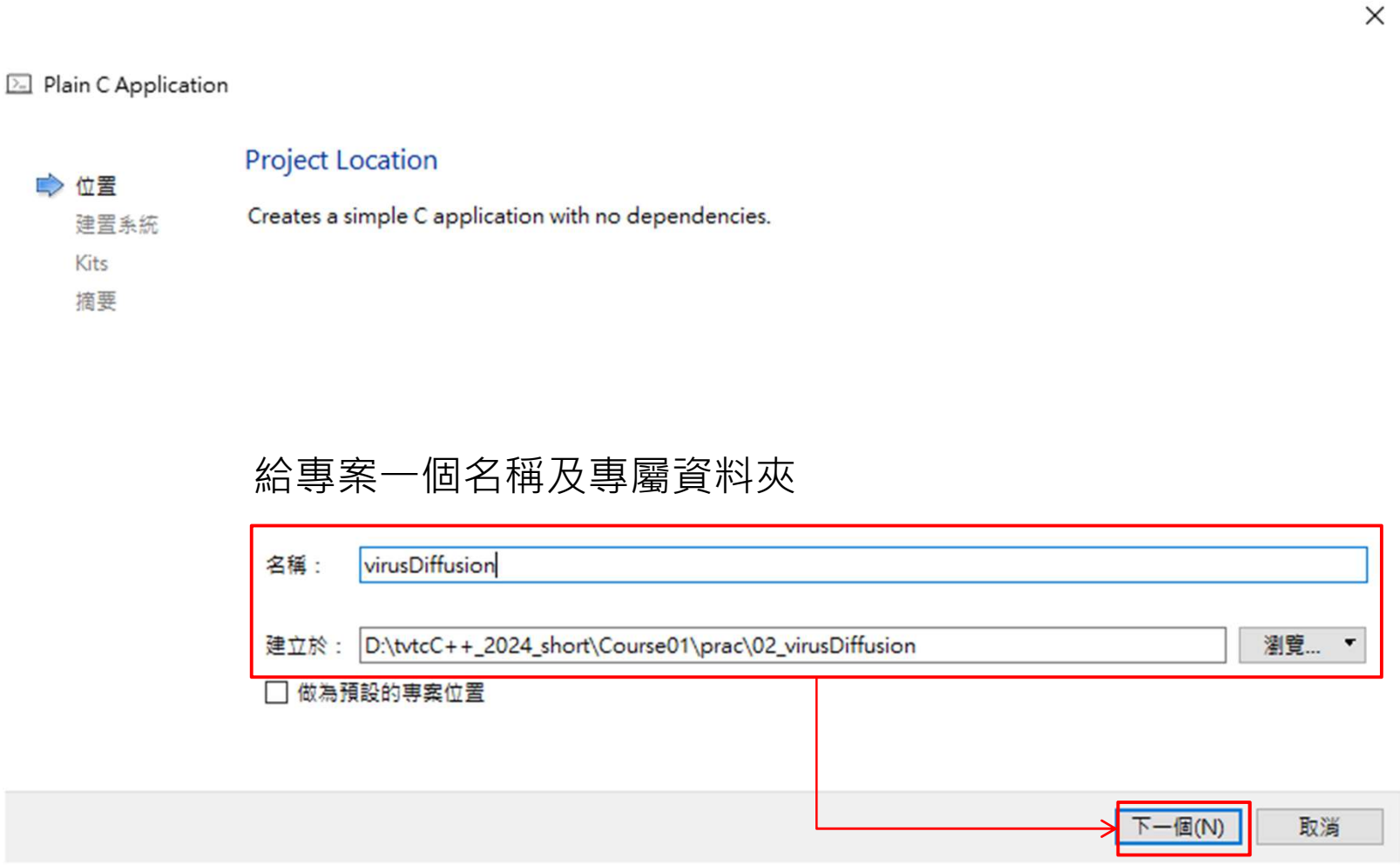

Plain C Application
位置
建置系統
Kits
摘要
Kit Selection
The following kits can be used for project virusDiffusion:
Select all kits
Type to filter kits by name...
Desktop Qt 6.7.1 MinGW 64-bit
詳情
Debug
2_virusDiffusion\virusDiffusion\build\Desktop_Qt_6_7_1_MinGW_64_bit-Debug
瀏覽...
Release
_virusDiffusion\virusDiffusion\build\Desktop_Qt_6_7_1_MinGW_64_bit-Release
瀏覽...
Profile
02_virusDiffusion\virusDiffusion\build\Desktop_Qt_6_7_1_MinGW_64_bit-Profile
瀏覽...
Desktop Qt 6.7.1 MSVC2019 64bit
詳情
Desktop Qt 6.7.1 MSVC2019 ARM64
詳情
下一個(N)
取消

選擇目標平臺

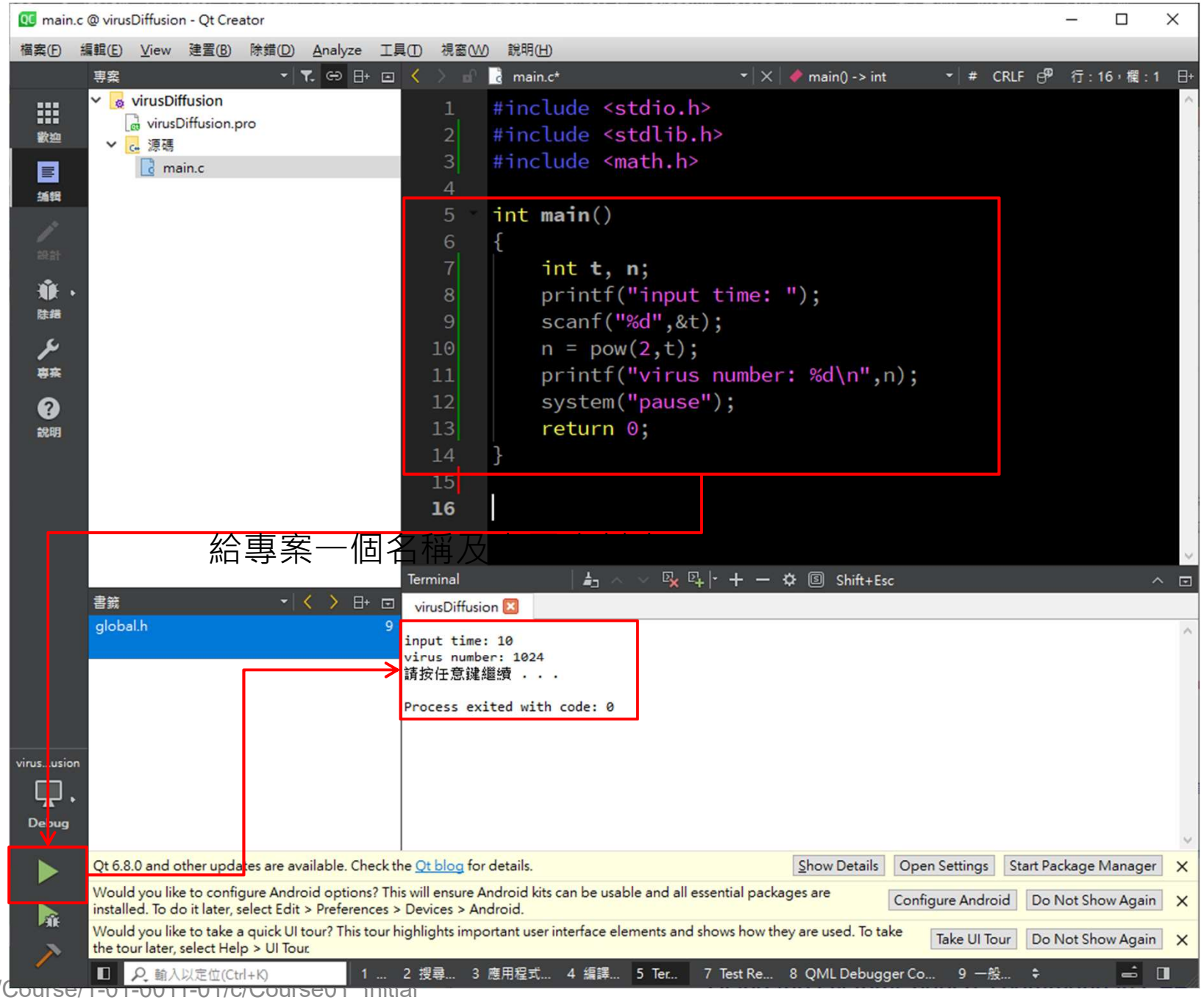

Plain C Application
位置
建置系統
Kits
摘要
專案管理
新增到專案中做為子專案：
<無>
新增到版本控制系統(V)：
<無>
組態...
選擇專案管理工具
要新增到此處的檔案
D:\tvtcC++_2024_short\Course01\prac\02_virusDiffusion\virusDiffusion:
main.c
virusDiffusion.pro
完成(F)
取消



/Course/1-01-0011-01/c/Course01_initial



/Course/1-01-0011-01/c/Course01_Initial



/Course/1-01-0011-01/c/Course01_Initial

發生什麼事了?


virusDiffusion
9
input time: 10
virus number: 1024
請按任意鍵繼續 . . .


virusDiffusion
9
input time: 100
virus number: -2147483648
請按任意鍵繼續 . . .