# 第1章 電腦程式語言及主控台應用程式介紹

- 當人類在日常生活中遇到問題時，常會開發一些工具來解決它。
  - 例：發明筆來寫字、發明腳踏車來替代雙腳行走等。
- 電腦程式語言也是解決問題的一種工具
  - 例：過去的車子都是手排車，是由駕駛人手動控制變速箱的檔位；現在的自排車，都是經由電腦程式根據當時的車速來控制變速箱的檔位
  - 人類必須借助共通語言交談溝通；同樣地，當人類要與電腦溝通時，也必須使用彼此都能理解的語言，像這樣的語言我們稱為電腦程式語言(Computer Programming Language)

# 電腦程式語言分成三大類

- 編譯式程式語言
  - 編譯式程式語言：凡是必須經過編譯器(Compiler)編譯成機器碼(Machine Code)的原始程式所隸屬之程式語言，稱之為編譯式程式語言。例：COBOL、C、C++等
    - 編譯式的程式語言，從原始程式變成可執行檔的過程分成編譯(Compile)及連結(Link)兩部分，分別由編譯程式(Compiler)及連結程式(Linker)負責

# 電腦程式語言分成三大類

- 編譯式程式語言（續）
  - 編譯程式負責檢查程式中的語法是否正確，以及程式中所使用的函式是否有定義。
  - 當原始程式編譯正確後，接著才由連結程式連結程式中之函式(或方法)所在的位址，若連結正確，進而產生原始程式之可執行檔
  - 若原始程式編譯無誤，就可執行它且下次無須重新編譯；否則必須修改程式且重新編譯
    - 執行效率高

# 電腦程式語言分成三大類

- **直譯式程式語言**
  - 直譯式程式語言：凡是必須經過直譯器(Interpreter)將指令一列一列翻譯成機器碼並執行的原始程式所隸屬之程式語言，稱之為直譯式程式語言。例：BASIC、HTML等
  - 利用直譯式程式語言所撰寫的原始程式，每次執行都要重新經過直譯器翻譯成機器碼，若執行過程發生錯誤就停止運作
    - 執行效率差

# 電腦程式語言分成三大類

- 結合編譯式及直譯式的程式語言，其執行速度比純編譯式語言慢一些
  - C#語言屬於編譯式及直譯式的程式語言之一
  - 利用C#語言所撰寫的原始程式必須經過C#編譯器(Compiler)編譯成位元組碼(Byte code)，再經由C#直譯器翻譯位元組碼並執行它
  - 位元組碼與電腦之作業系統( 例：UNIX/Linux、Windows及Mac OS)無關，只要在電腦的作業系統中，安裝C#的虛擬機器(C# Virtual Machine：JVM)，就能執行它
    - 因此，C# 屬於跨平台的程式語言

# 1-1 .NET Framework架構

- .NET Framework:
  - 主要的目的，是提供作業系統一個共同的語言執行環境平台，讓程式開發者只要專注在與共同語言執行環境的互動，而與作業系統溝通及呼叫系統相關函式，則交由共同語言執行環境來負責
    - 是Microsoft 公司所開發的一種架構
    - 它由Common Language Runtime（CLR，共同語言執行環境）及.NET Framework Class Library（.NETFramework 類別庫）所組成
    - 支援.NET Framework的程式語言有Visual C++、Visual C# 及Visual Basic

- 利用支援.NET Framework的程式語言所撰寫的「原始程式碼」，必須透過
  - 「.NET Framework 編譯器」將它編譯成副檔名為.dll或.exe的「中間程式語言」（Microsoft Intermediate Language，簡稱MSIL）
  - 再經由「共同語言執行環境」的「即時直譯程式」（Just in Time，JIT）及連結「.NET 類別庫」，將中間程式語言直譯成「原生碼」（Native Code）後才能執行。

- 中間程式語言與電腦之作業系統（例：UNIX/Linux、Windows及Mac OS）無關，只要在電腦的作業系統中有安裝「共同語言執行環境」就能執行它
- 因此，支援.NET Framework 的程式語言屬於跨平台的程式語言

- .NET 提供的共同語言執行環境，除了
  - 能讓支援.NET Framework的程式語言所撰寫的程式在不同的平台上執行外
  - 還能讓不同程式語言所開發的原始程式碼在編譯之後，產生相同的語法及資料型態名稱，使不同程式語言彼此間能夠相互溝通

# 1-2 物件導向程式設計

- 電腦程式：利用任何一種電腦程式語言所撰寫的指令集
- 程式設計：撰寫程式的整個過程
- 程式設計方式可分成下列兩種類型：
  - 第一類為程序導向程式設計(Procedural Programming)
    - 設計者依據解決問題的程序，完成電腦程式撰寫
    - 程式執行時電腦會依據流程進行各項工作的處理

- 第二類為物件導向程式設計(Object Oriented Programming: OOP)
- 結合程序導向程式設計的原理與真實世界中的物件觀念
  - 建立物件與真實問題間的互動關係
  - 使得程式在維護、除錯，及新功能擴充上更容易

- 何謂物件（Object）呢？
  - 物件是具有屬性及方法的實體
    - 人、汽車、火車、飛機、電腦等
    - 實體具有屬於自己的特徵及行為，其中特徵以屬性（Properties）來表示，而行為則以方法（Methods）來描述
  - 物件可以藉由它所擁有的方法，改變它擁有的屬性值及與不同的物件溝通
    - 例：人具有胃、嘴巴等屬性，及吃、說等方法。可藉由「吃」這個方法，來降低胃的饑餓程度；可藉由「說」這個方法，與別人溝通或傳達訊息

- OOP就是模擬真實世界之物件運作模式的一種程式設計概念
  - 常見OOP的電腦程式語言有C++、C#等。

- 程式設計的步驟如下：
  1. 了解問題的背景知識
  2. 構思解決問題的程序，並繪出流程圖
  3. 選擇一種電腦程式語言，依據步驟2的流程圖撰寫指令集
  4. 編譯程式並執行，若編譯正確且執行結果符合問題的需求，則結束；否則必須重新檢視步驟1~3

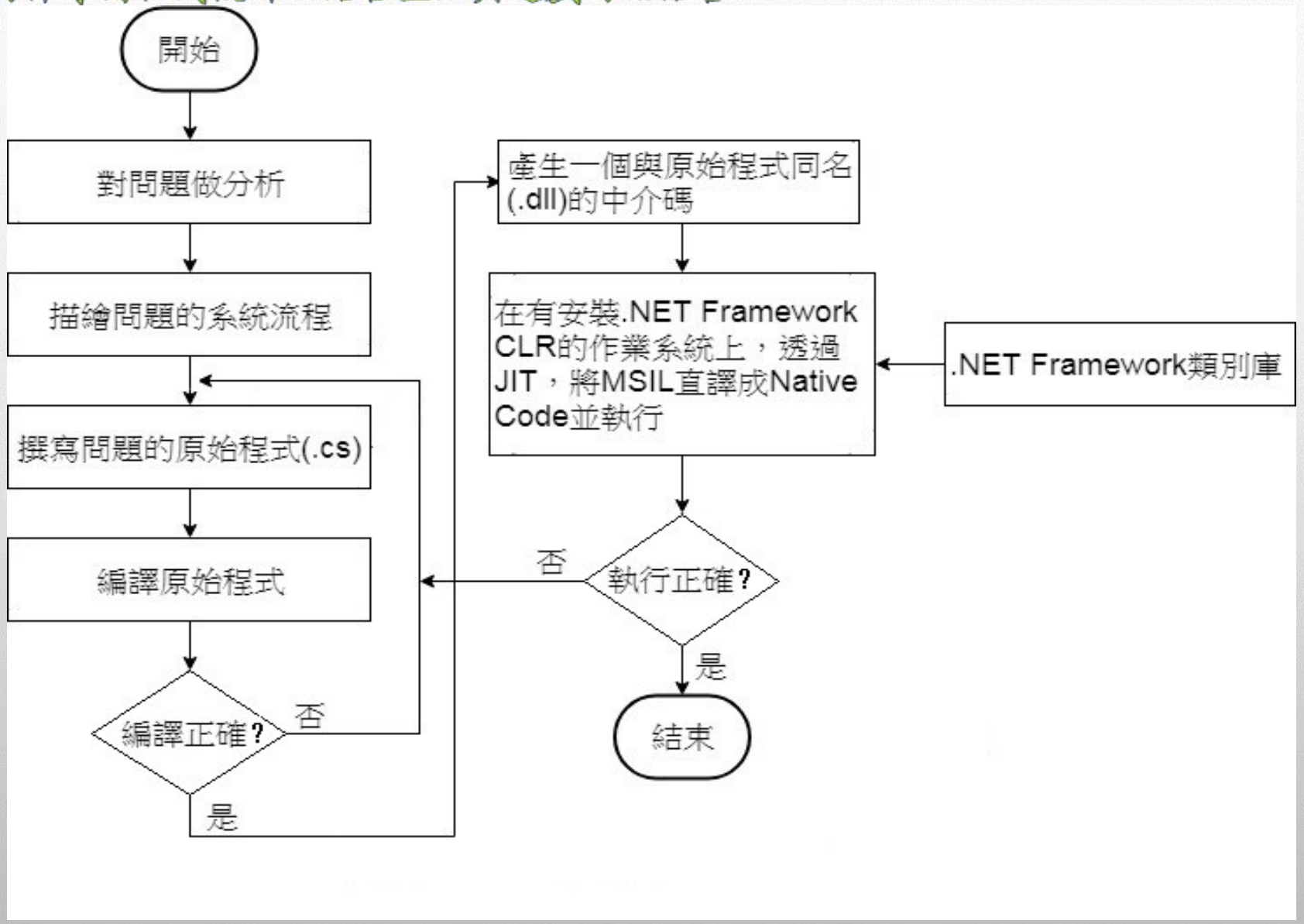
開始
對問題做分析
描繪問題的系統流程
撰寫問題的原始程式(.cs)
編譯原始程式
編譯正確?
否
是
產生一個與原始程式同名(.dll)的中介碼
在有安裝.NET Framework CLR的作業系統上，透過JIT，將MSIL直譯成Native Code並執行
.NET Framework類別庫
執行正確?
否
是
結束

圖1-1　程式設計流程圖

- 程式從撰寫階段到執行階段，可能產生的錯誤有編譯時期錯誤(compile error)及執行時期錯誤(run-time error)
  - 編譯時期錯誤是指程式敘述違反程式語言之撰寫規則，這類錯誤稱為語法錯誤
    - 例：在C#語言中，大多數的指令是以分號「;」做為該指令之結束符號。若違反此規則，就無法完成編譯

- 執行時期錯誤是指程式執行時產生的結果不符合需求或發生錯誤，這類錯誤稱為語意錯誤或例外
- 例：a=b/c; 在語法上是正確的，但執行時，若c為0，則會發生例外

「System.DivideByZeroException: '嘗試以零除。'」

# 1-3 Visual Studio簡介

- 現有的高階程式語言，都會提供「整合開發環境」（Integrated Development Environment，IDE）介面，以簡化開發應用程式的過程
- Visual Studio是微軟公司所開發的IDE介面
  - 它提供支援.NET Framework的Visual C++、Visual C#、Visual Basic及JavaScript程式語言的編輯、編譯、除錯、執行、管理及部署之整合開發平台
  - 讓程式開發和管理，更加快速且有效率

- 目前最新版的Visual Studio 是Visual Studio 2017
- Visual Studio 2017提供
  - Community（社群版）
  - Professional（專業版）
  - Enterprise（企業版）

- Visual Studio Community 2017為免費的初學者程式開發工具，它提供不同類型的應用程式開發
  - Windows Desktop單機應用程式開發
  - Web網頁應用程式開發
  - Azure雲端應用程式開發
  - Unity遊戲開發
  - Xamarin行動應用程式開發
  - 本書所有的Visual C#範例程式都是在Visual Studio Community 2017整合開發環境中所完成

# 1-3-1 安裝Visual Studio Community 2017

- 開發Visual C#主控台應用程式及視窗應用程式前，請先到Microsoft官方網站下載Visual C#整合開發環境（IDE）：Visual Studio Community（目前為2017版）
- 依下列程序下載Visual Studio Community 2017，並安裝：

1. 請到Microsoft 官方網站：
   https://www.visualstudio.com/zh-hant/
   並點選「Windows下載 / Community 2017」

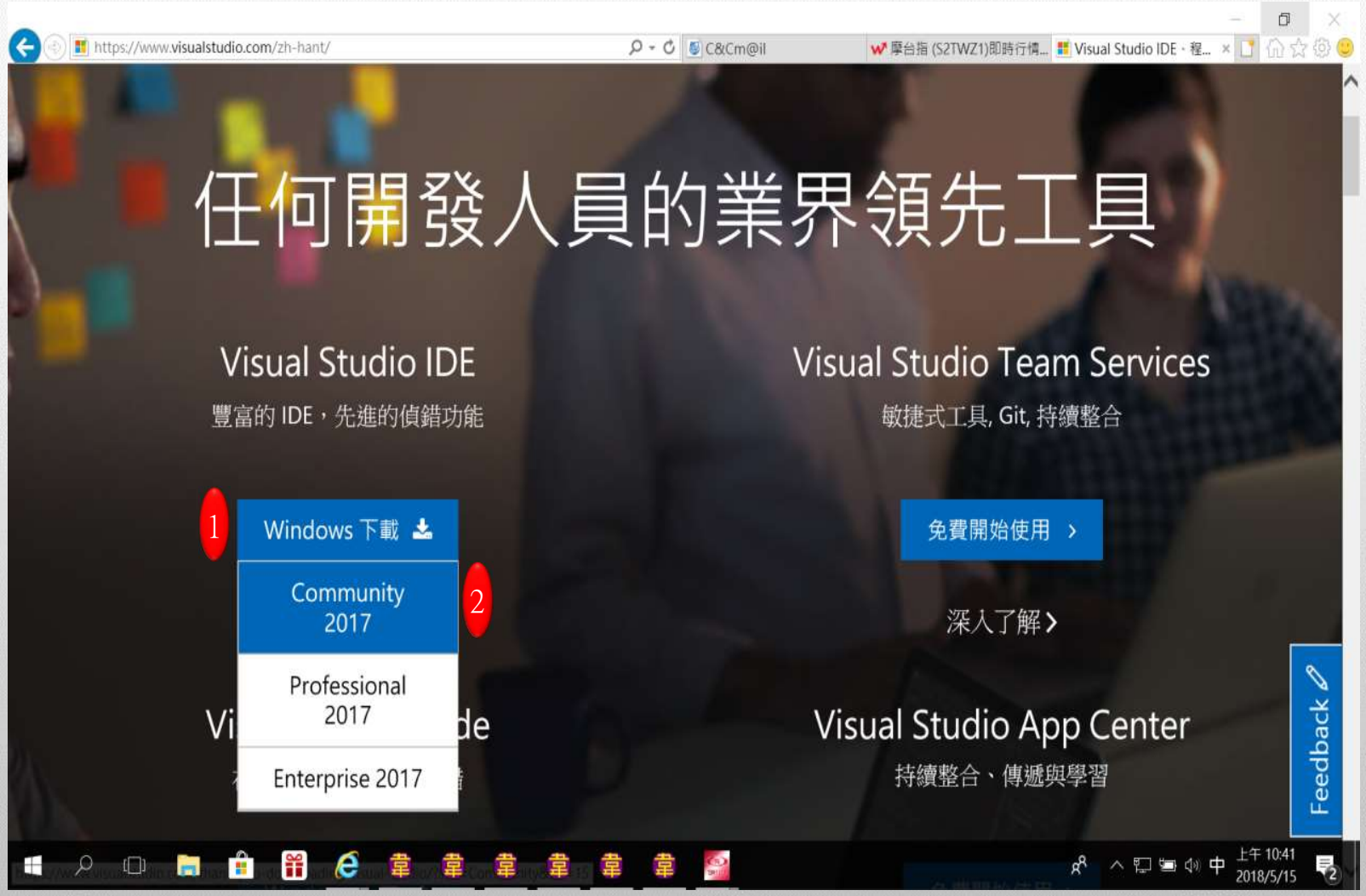


圖1-2　Microsoft官方網站

2. 按「儲存(S)」，將安裝程式 vs_community_1794283963.1520649367.exe，儲存於電腦磁碟中

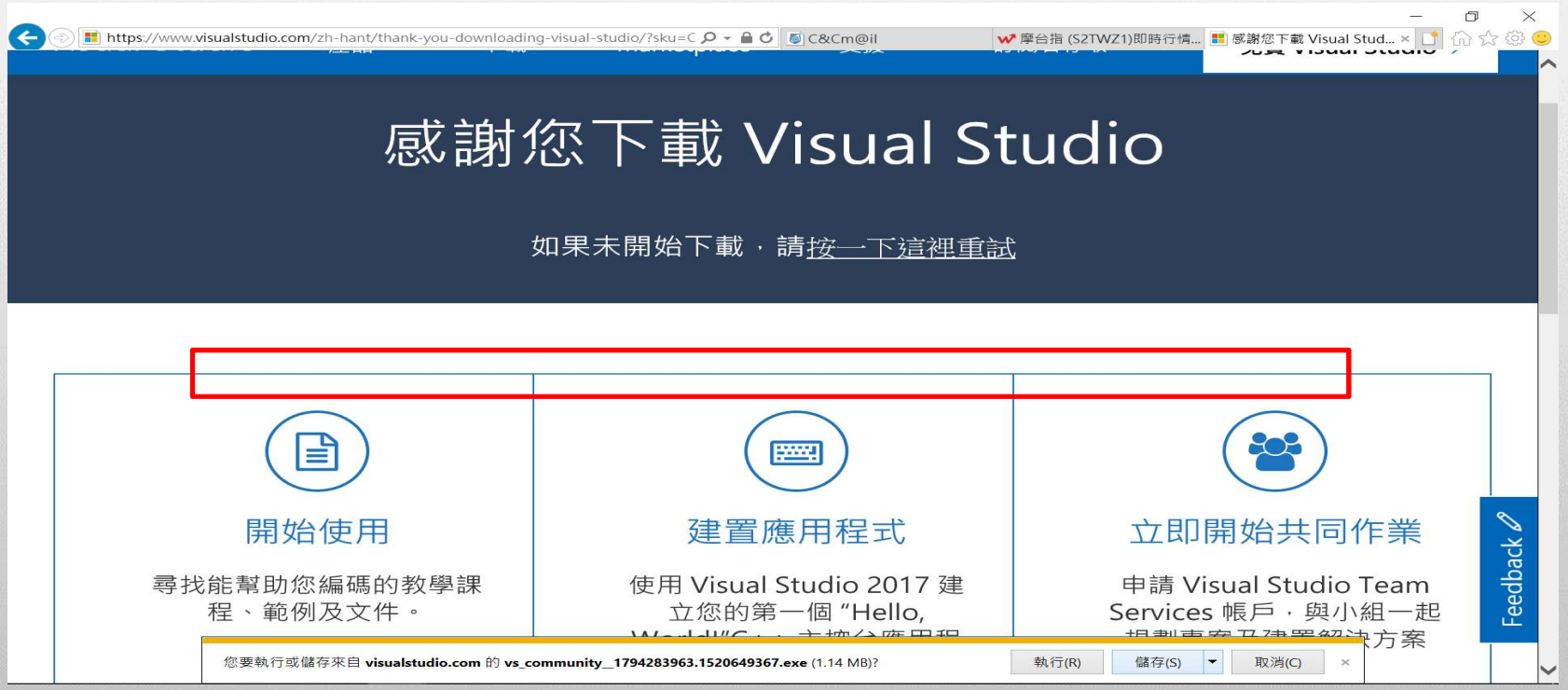


圖1-3　Visual Studio Community 2017下載

■ 註：不同時期下載的安裝程式，檔名會有所不同

3. 執行vs_community_1794283963.1520649367.exe


圖1-4　Visual Studio Community 2017安裝(一)

4. 按「繼續」

## 5. 稍待一下...正在擷取您的檔案

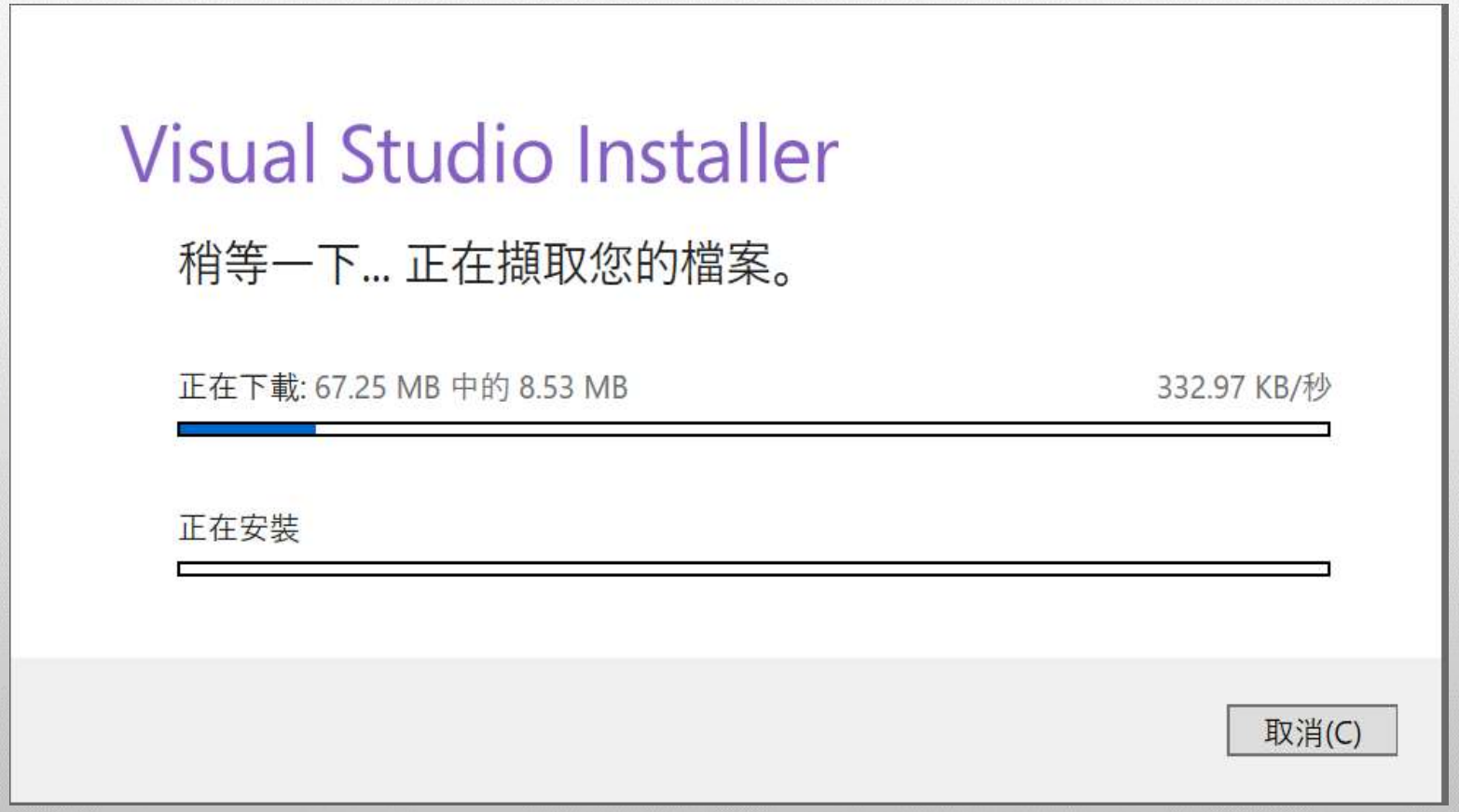


圖1-5　Visual Studio Community 2017安裝(二)

6. 點選「通用Windows 平台開發」及「.NET 桌面開發」選項，並按「安裝」

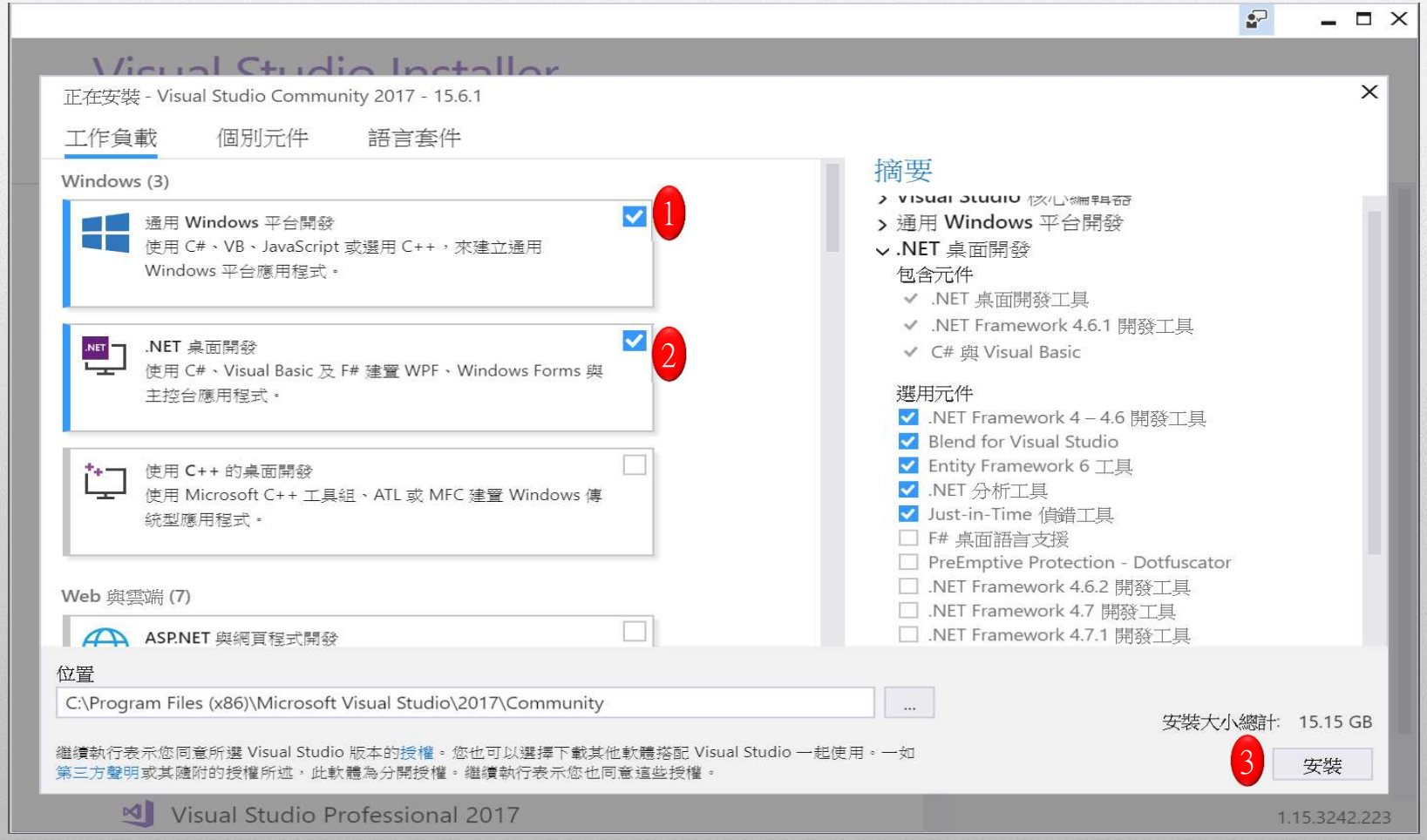


圖1-6　Visual Studio Community 2017安裝(三)

# 7. 安裝進行中，請稍候

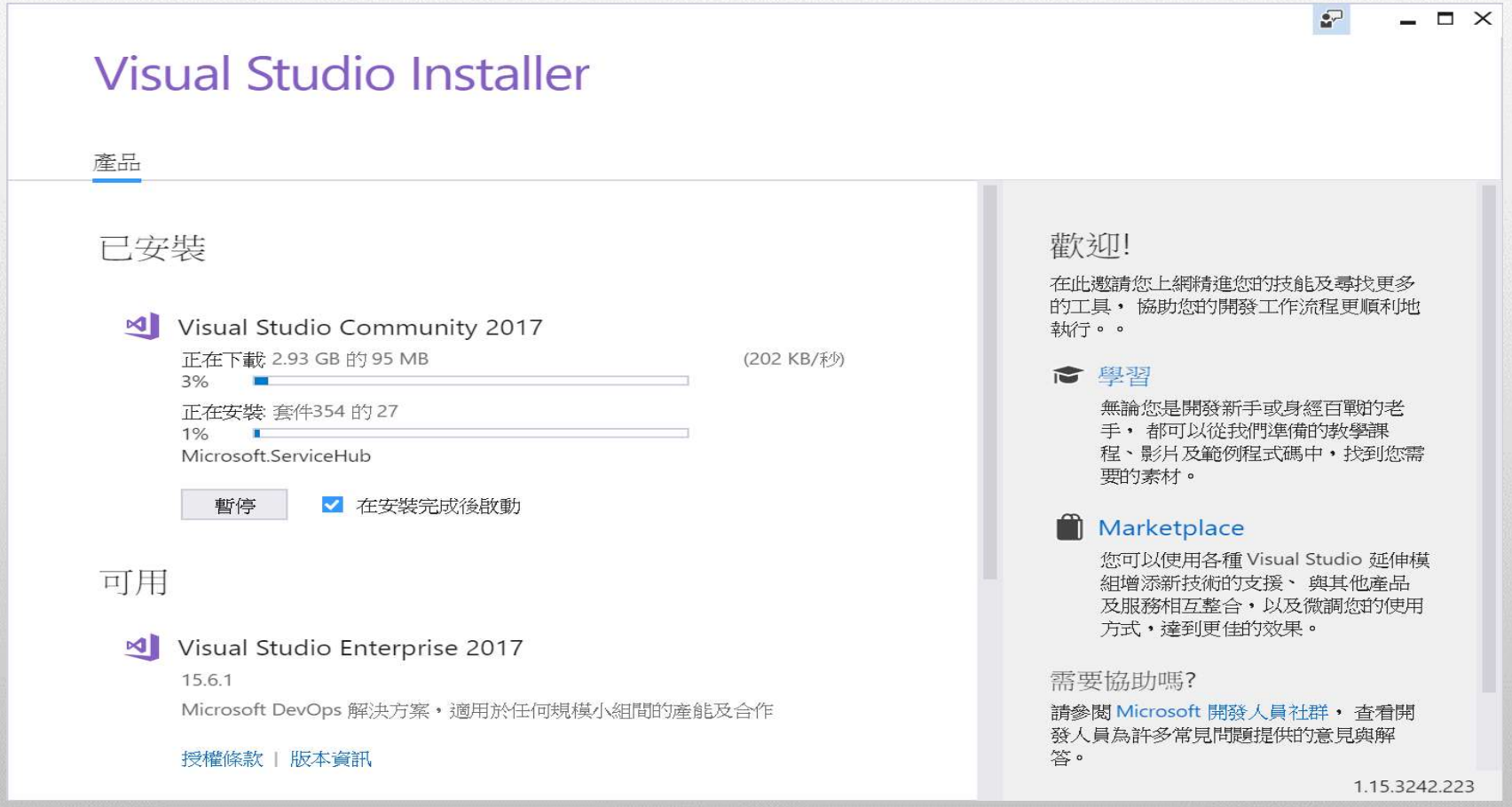


圖1-7　Visual Studio Community 2017安裝(四)

8. 安裝成功後，按「啟動」，進入「註冊」視窗

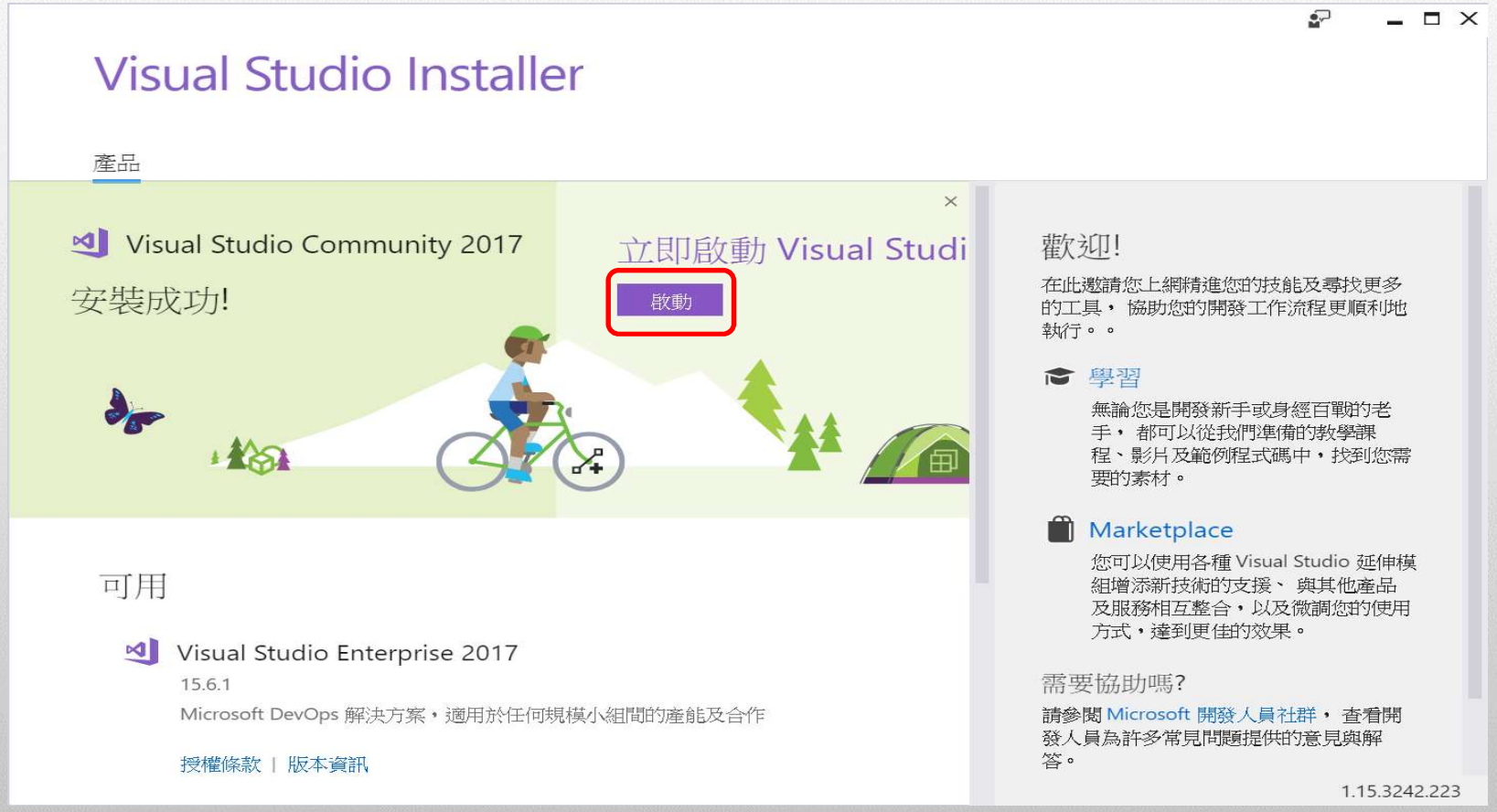


圖1-8　Visual Studio Community 2017註冊(一)

9. 按「不是現在，以後再說。」，進入環境及色彩佈景設定視窗


圖1-9　Visual Studio Community 2017註冊(二)

10. 按「啟動Visual Studio(S)」，進入「Visual Studio 2017」的「起始頁」畫面

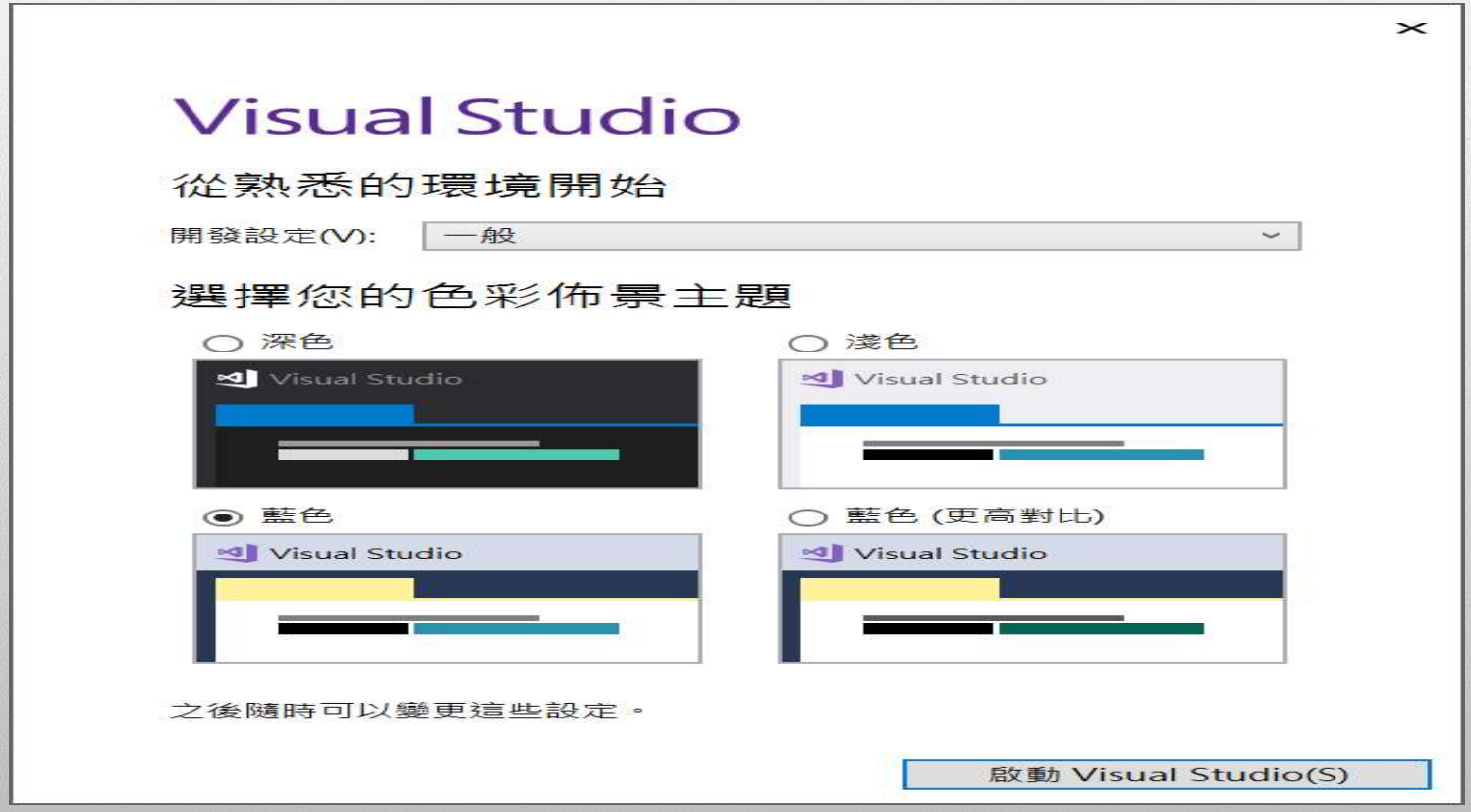


圖1-10　Visual Studio Community 2017啟動(一)

11. 按「起始頁」頁籤旁的X關閉這個頁面，進入Visual Studio 2017 整合開發環境視窗

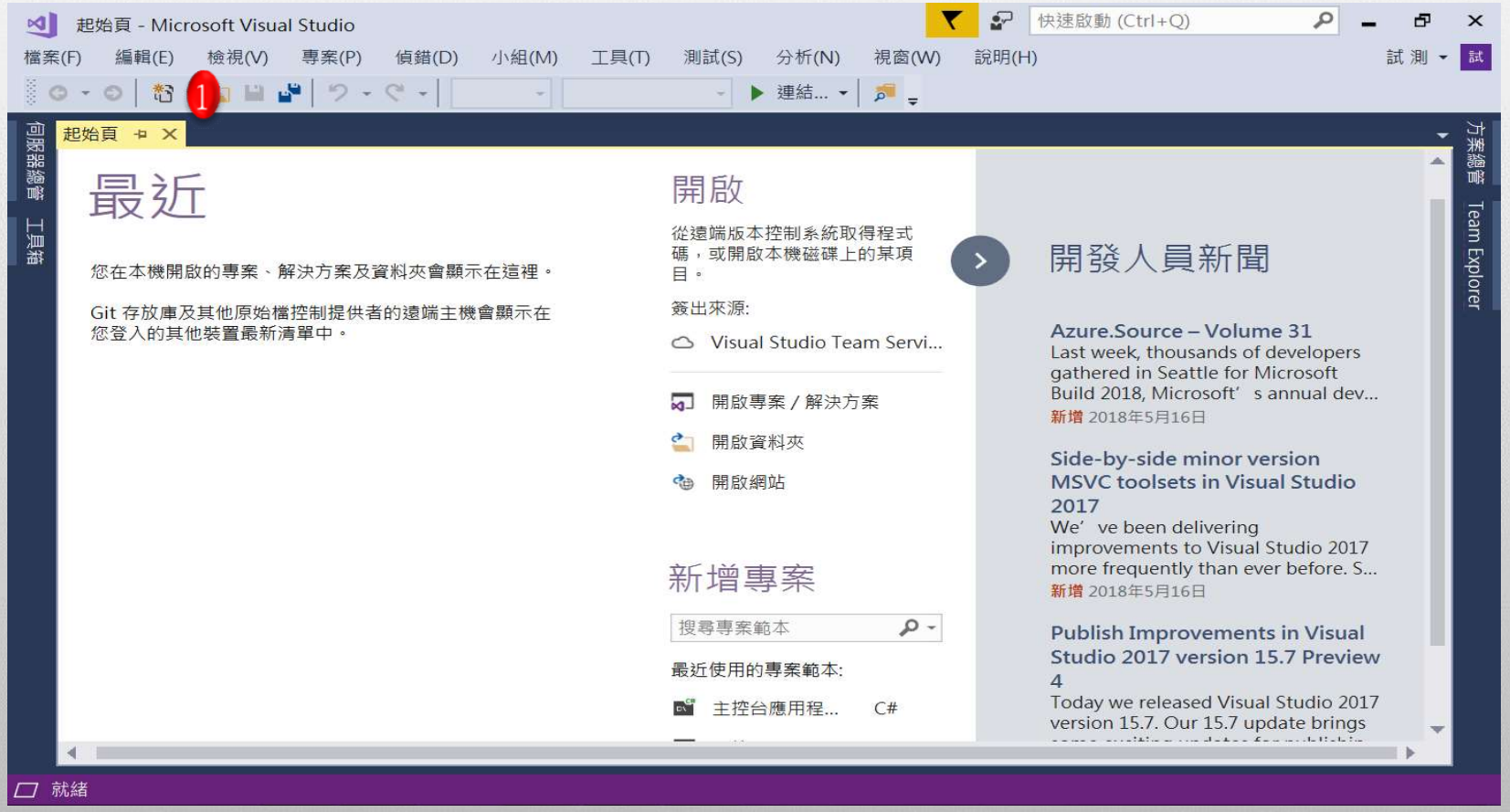


圖1-11　Visual Studio Community 2017啟動(二)

# 12. Visual Studio 2017 整合開發環境視窗

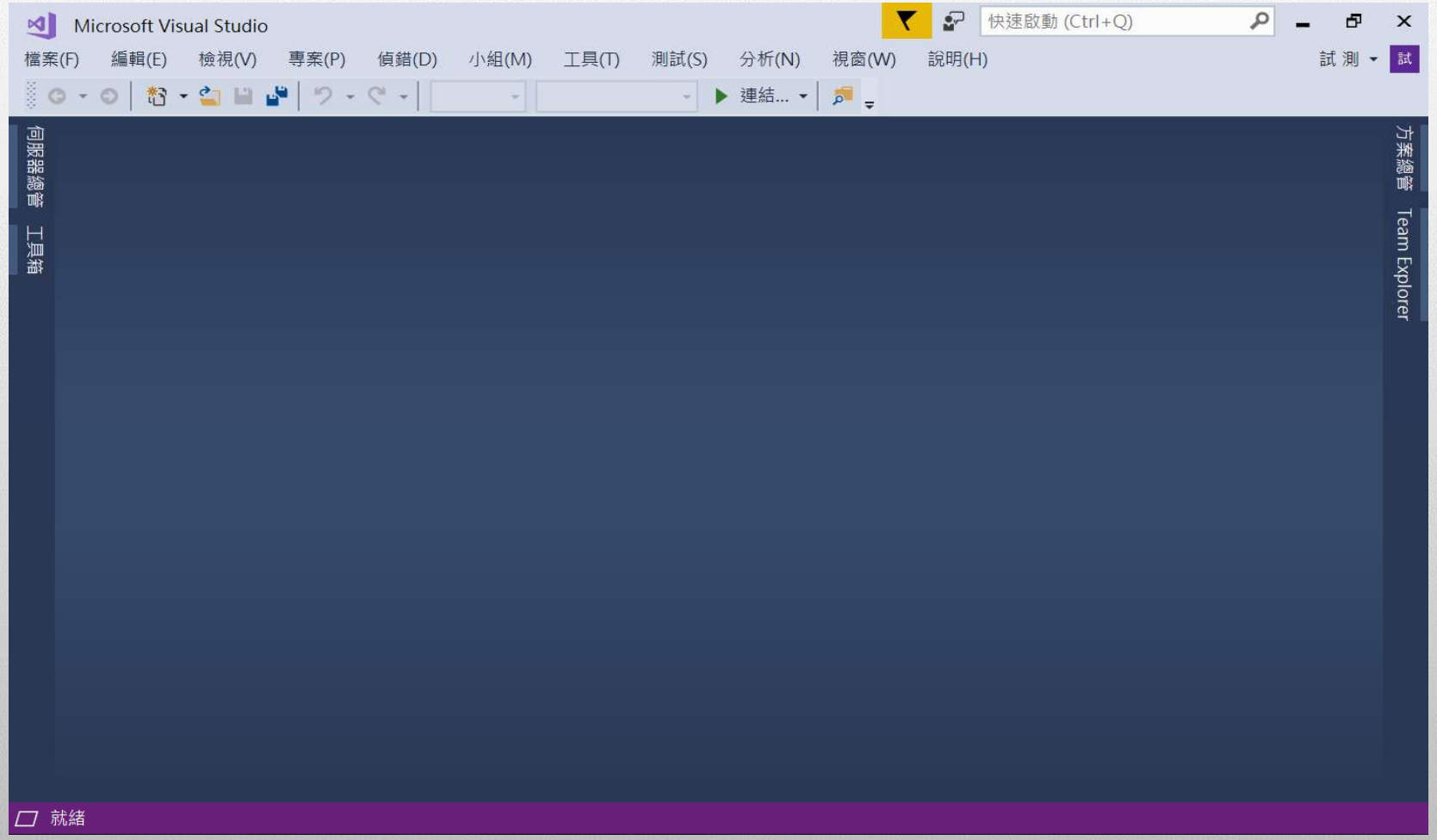


圖1-12　Visual Studio Community 2017啟動(三)

# 1-3-2 Visual Studio 2017 操作環境設定

- 第一次執行Visual Studio 2017 時，請依下列程序，分別設定
  - 專案位置（預設在安裝的磁碟機:\Users\ 使用者名稱\Documents\Visual Studio 2017）
  - 程式文字的字型大小（預設為10 點）

- 設定專案位置：

  (1) 點選功能表中的「工具(T) / 選項(O)」。

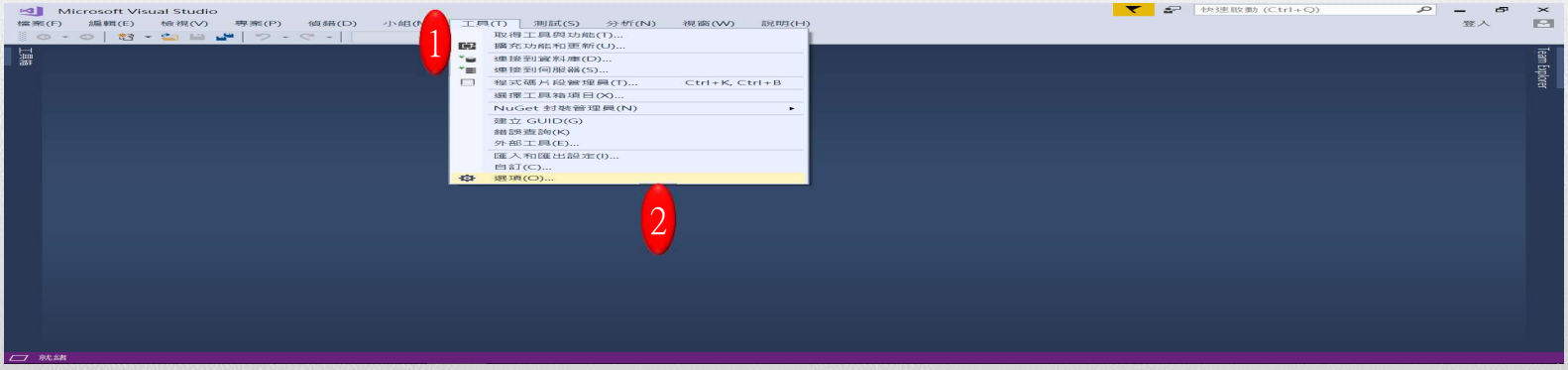


圖1-15　Visual Studio Community 2017環境設定(一)

(2) 點選「專案和方案 / 位置」，並在「專案位置(P)」中輸入「D:\C#」，最後按「確定」

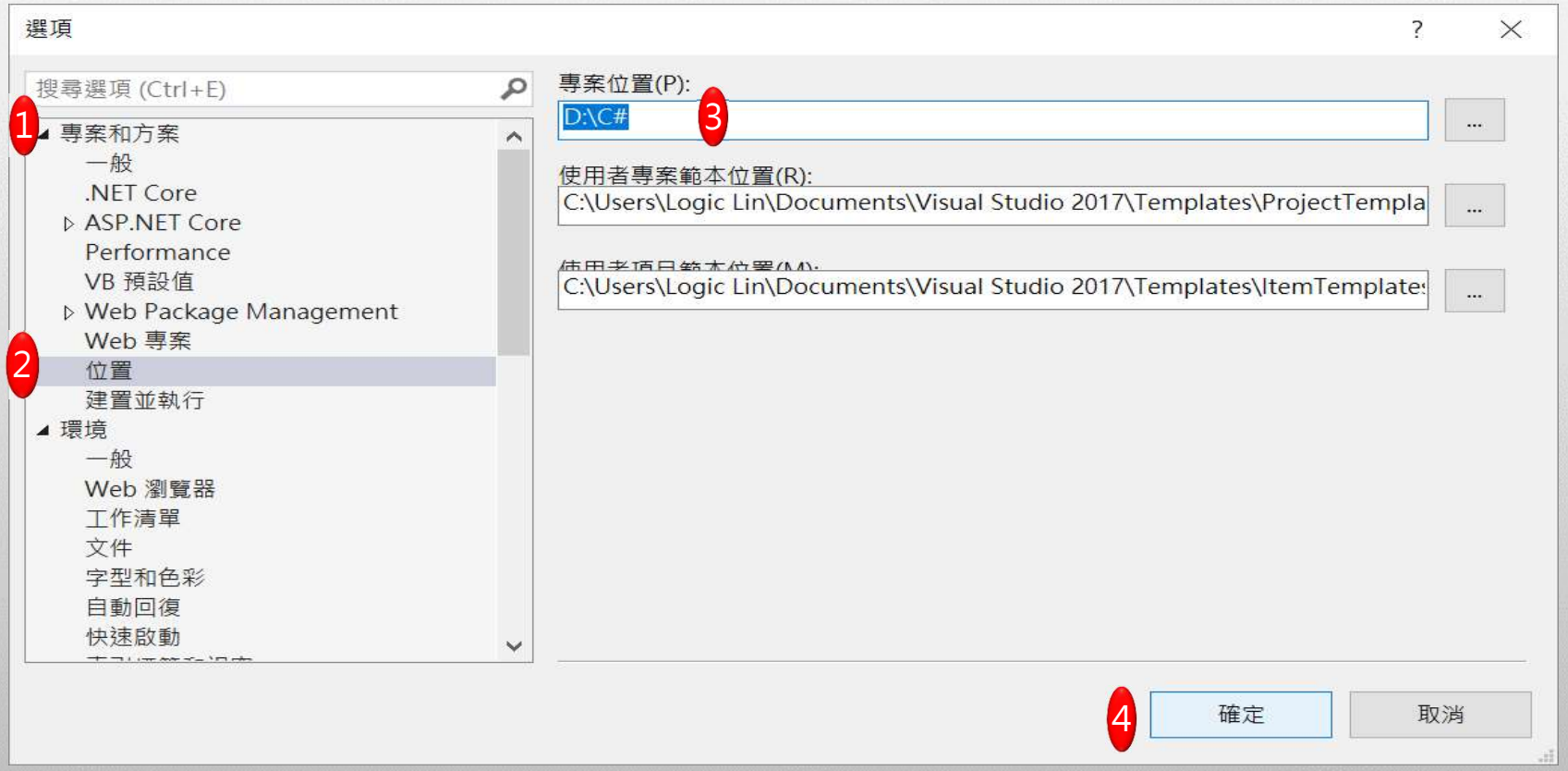


圖1-16　Visual Studio Community 2017環境設定(二)

- 設定程式文字的字型大小：
  (1) 點選功能表中的「工具(T) / 選項(O)」。

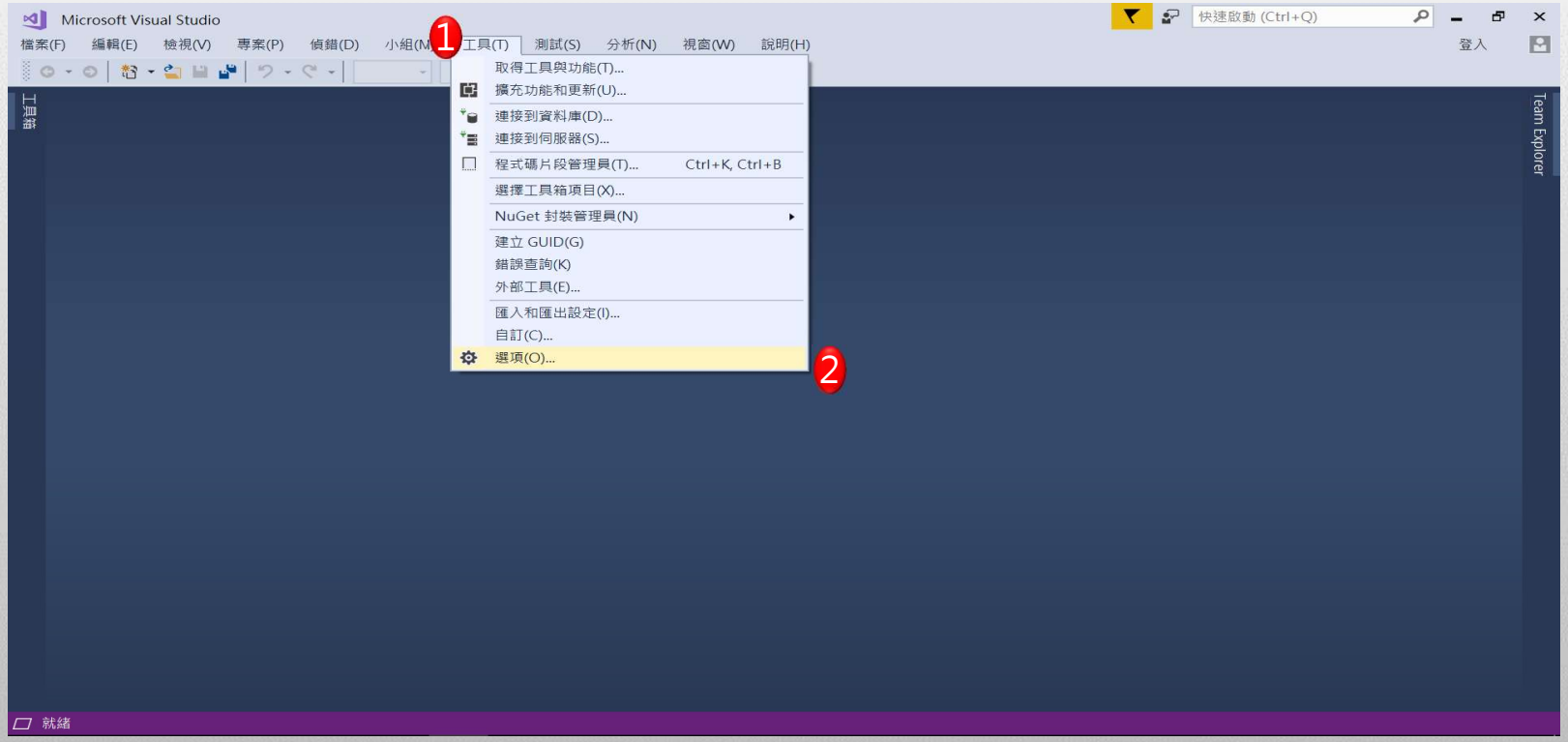


圖1-17　Visual Studio Community 2017環境設定(三)

(2) 點選「環境 / 字型和色彩」，及點選「顯示項目(D) / 純文字」，並在「大小(S)」中輸入「15」，最後按「確定」

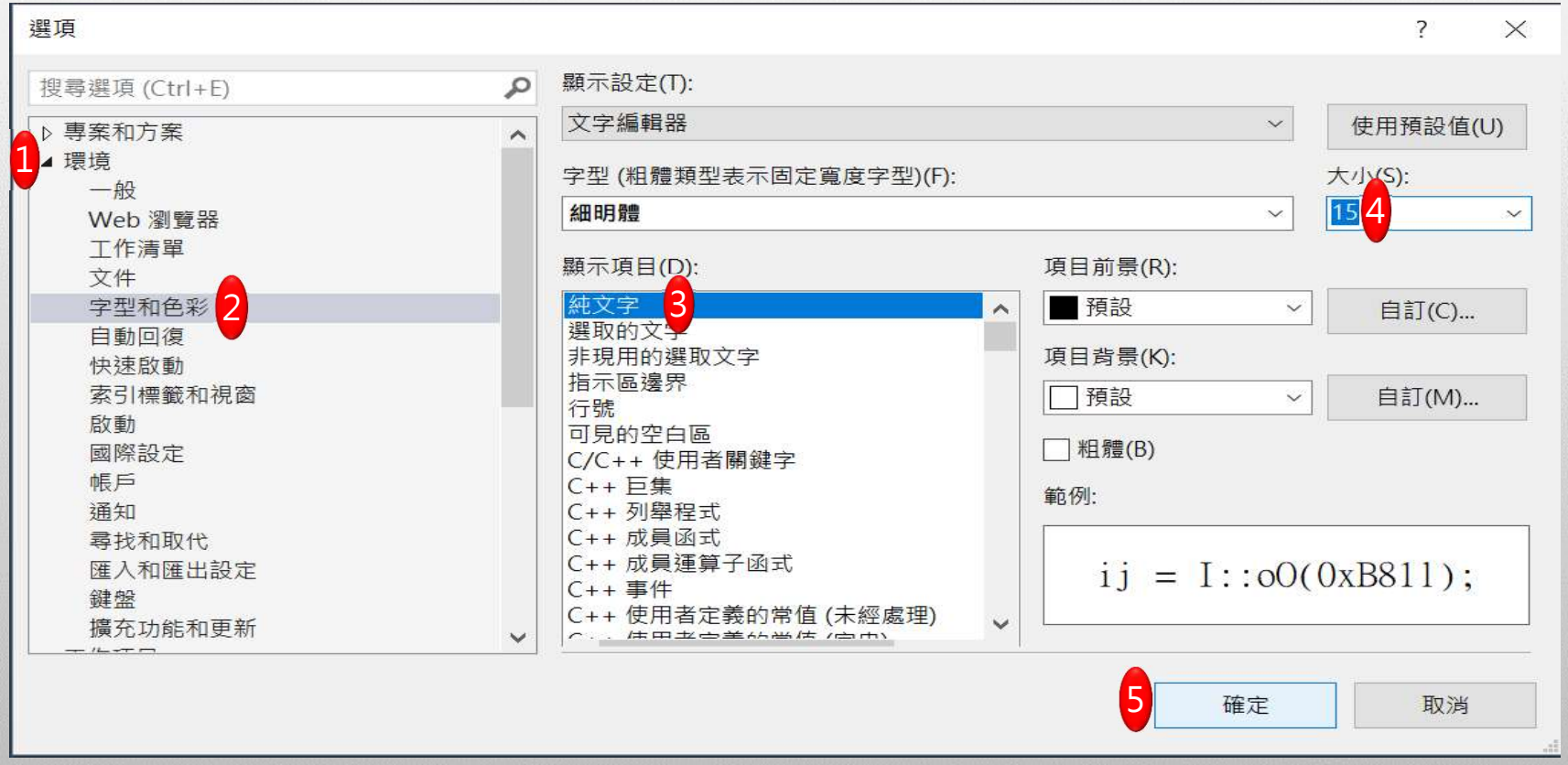


圖1-18　Visual Studio Community 2017環境設定(四)

# 1-3-3 建立Visual C# 主控台應用程式

- **Visual Studio 2017 是以專案模式來建立及管理 Visual C# 應用程式及相關的資源檔與參考檔**
  - 因此，開發應用程式時，會將應用系統分成多個專案程式來撰寫，方便日後團隊合作（或功能獨立）設計及維護
- **進入Visual Studio 2017 整合開發環境的程序：**
  - **點選「開始」中的「Visual Studio 2017」，進入 Visual Studio 2017 整合開發環境的「起始頁」視窗**

2. 關閉「起始頁」（即點按「起始頁」頁籤上的X），即可進入「Visual Studio 2017整合開發環境」

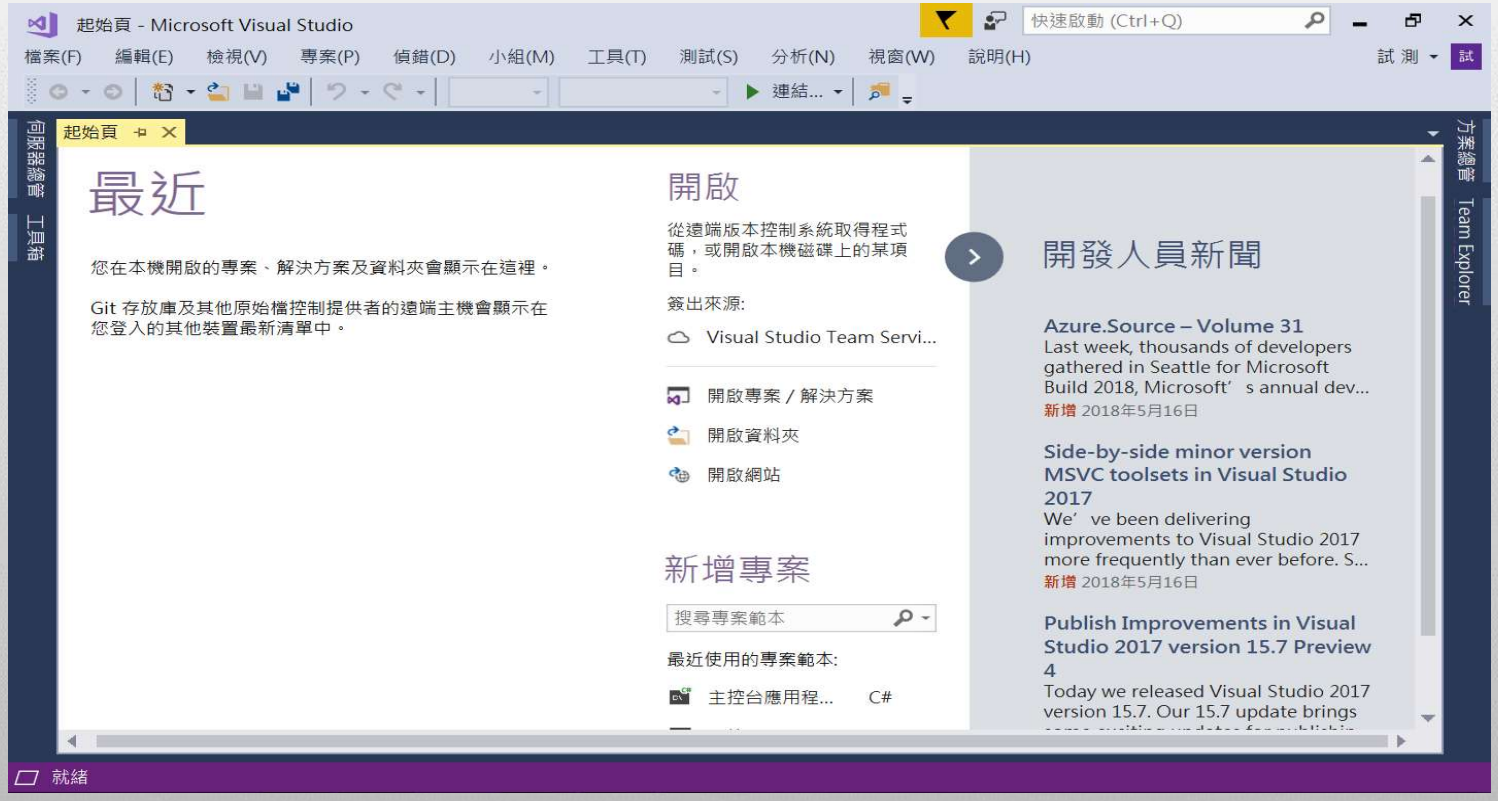


圖1-19　建立主控台應用程式專案(一)

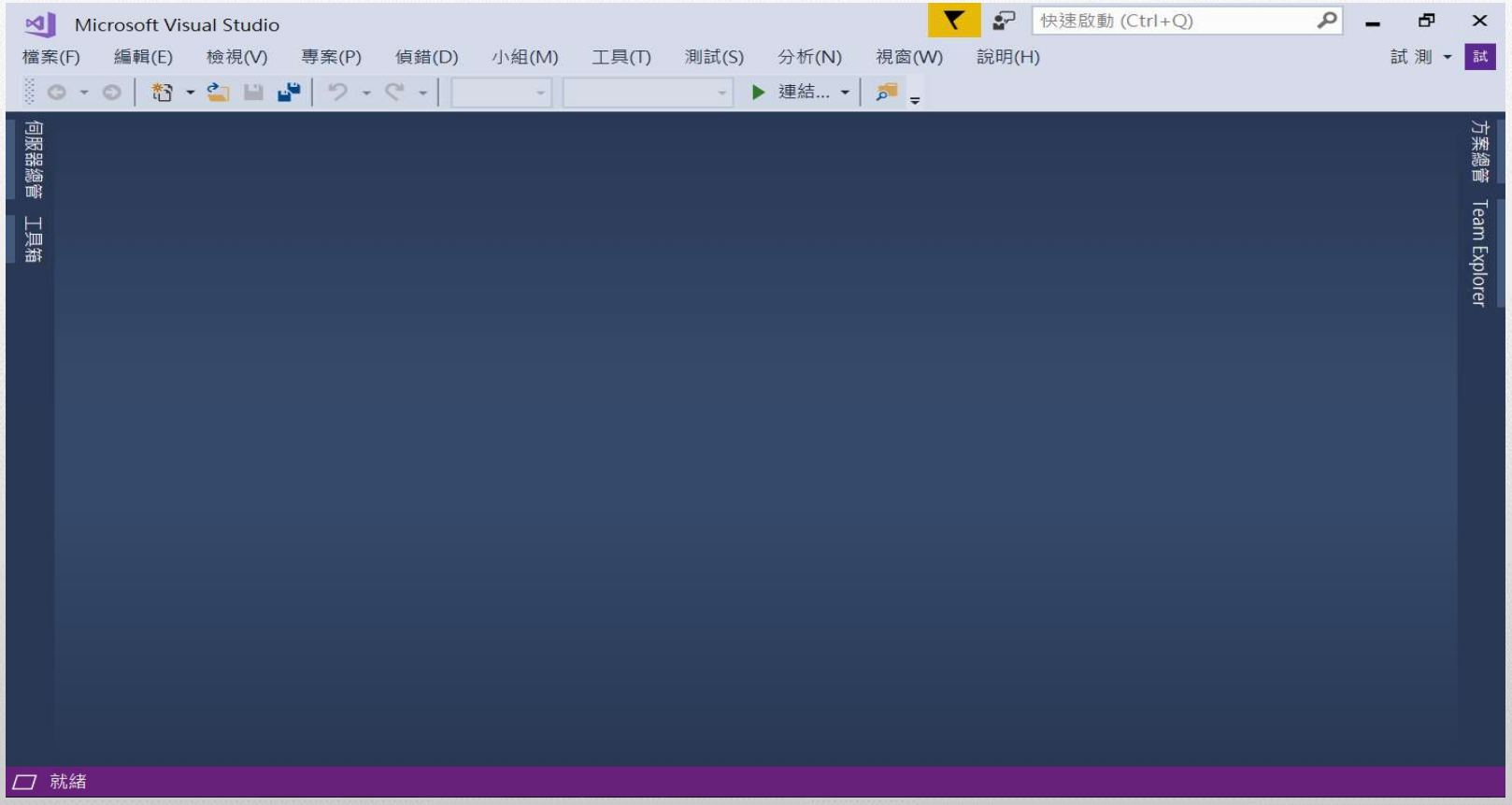


圖1-20　建立主控台應用程式專案(二)

- 應用程式使用的介面，有
  - 文字介面（Command-line Interface）
  - 圖形介面（GraphicUser Interface）
- 文字介面是以純文字方式來顯示使用者的電腦操作介面，使用這種介面的應用程式稱之為「主控台應用程式」（Console Applications）使用這種介面的應用程式稱之為「主控台應用程式」(Console Applications)。
- 圖形介面則是以圖形方式來顯示使用者的電腦操作介面，使用這種介面的應用程式稱之為「視窗應用程式」（Windows Applications）

- **建立Visual C#「主控台應用程式專案」的程序：**
  - (以專案名稱Ex1 為例，且將專案名稱Ex1 建立在資料夾「D:\C#\ch01」中)

1. **點選功能表中的「檔案(F) / 新增(N) / 專案(P)」**

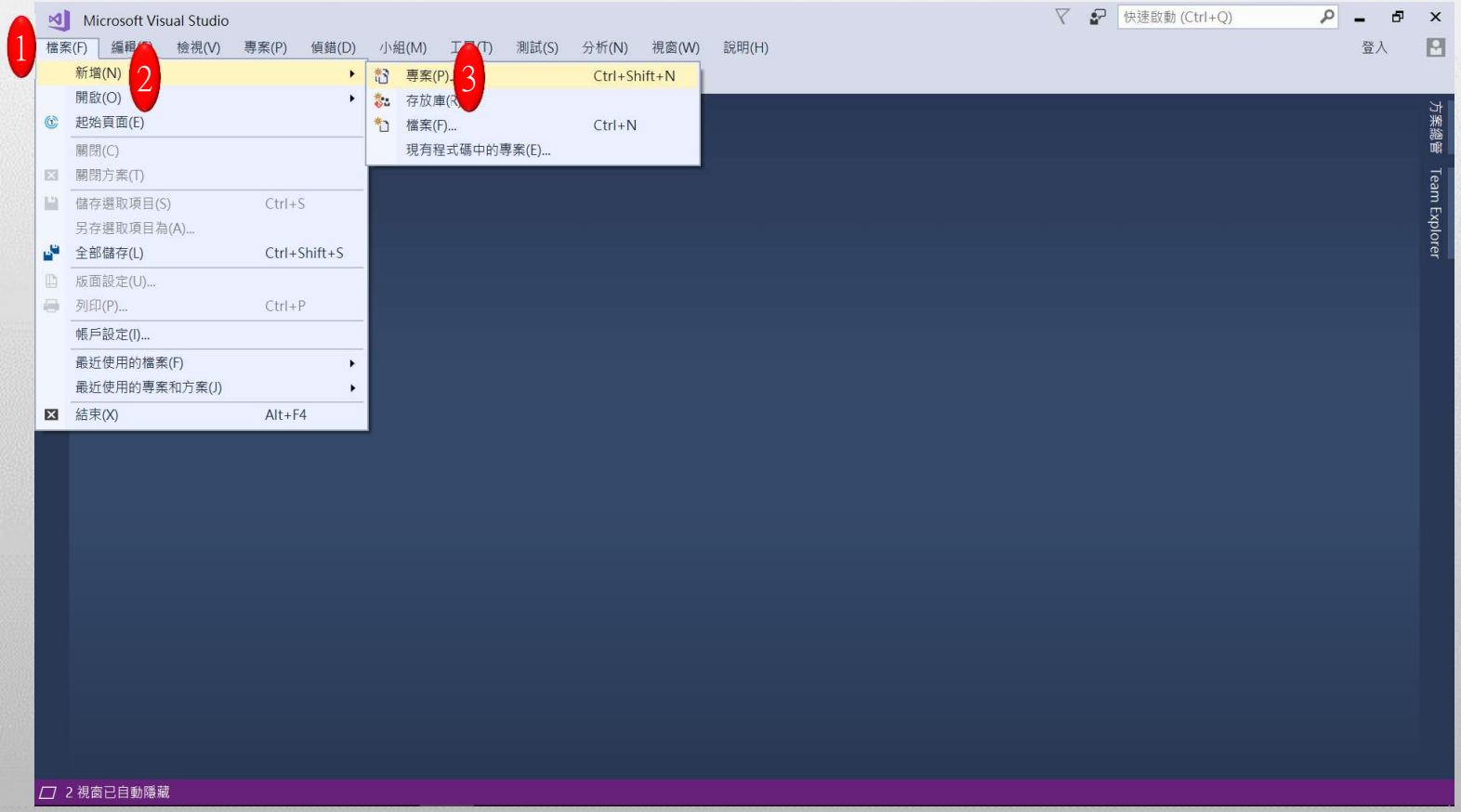


圖1-21　建立主控台應用程式專案(三)

2. (1) 點左邊的「Visual C#」

(2) 點中間的「主控台應用程式(.NET Framework)」

(3) 在「名稱(N)」欄位中，輸入「Ex1」

(4) 在「位置(L)」欄位中，輸入「D:\C#\ch01」

(5) 按「確定」，完成專案建立

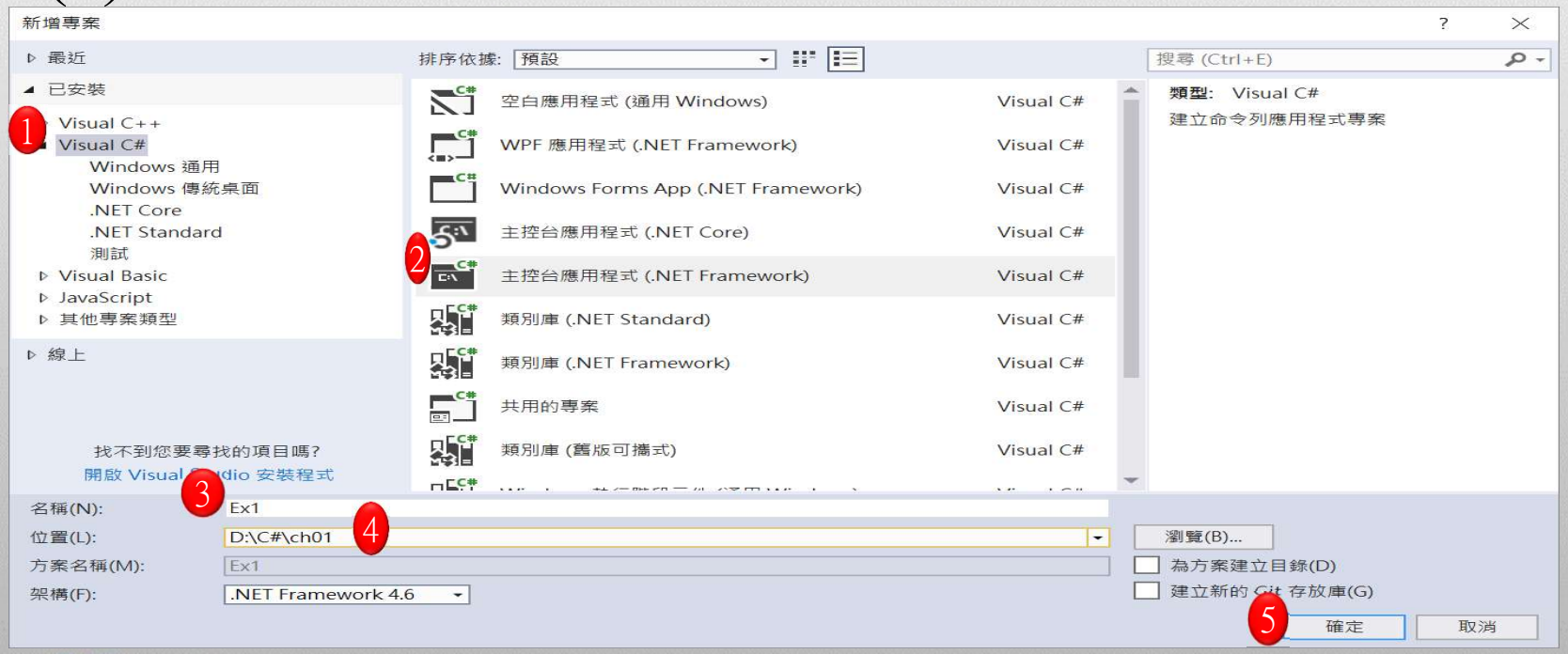


圖1-22　建立主控台應用程式專案(四)

- 完成圖1-22 之操作後，會出現如圖1-23的視窗：

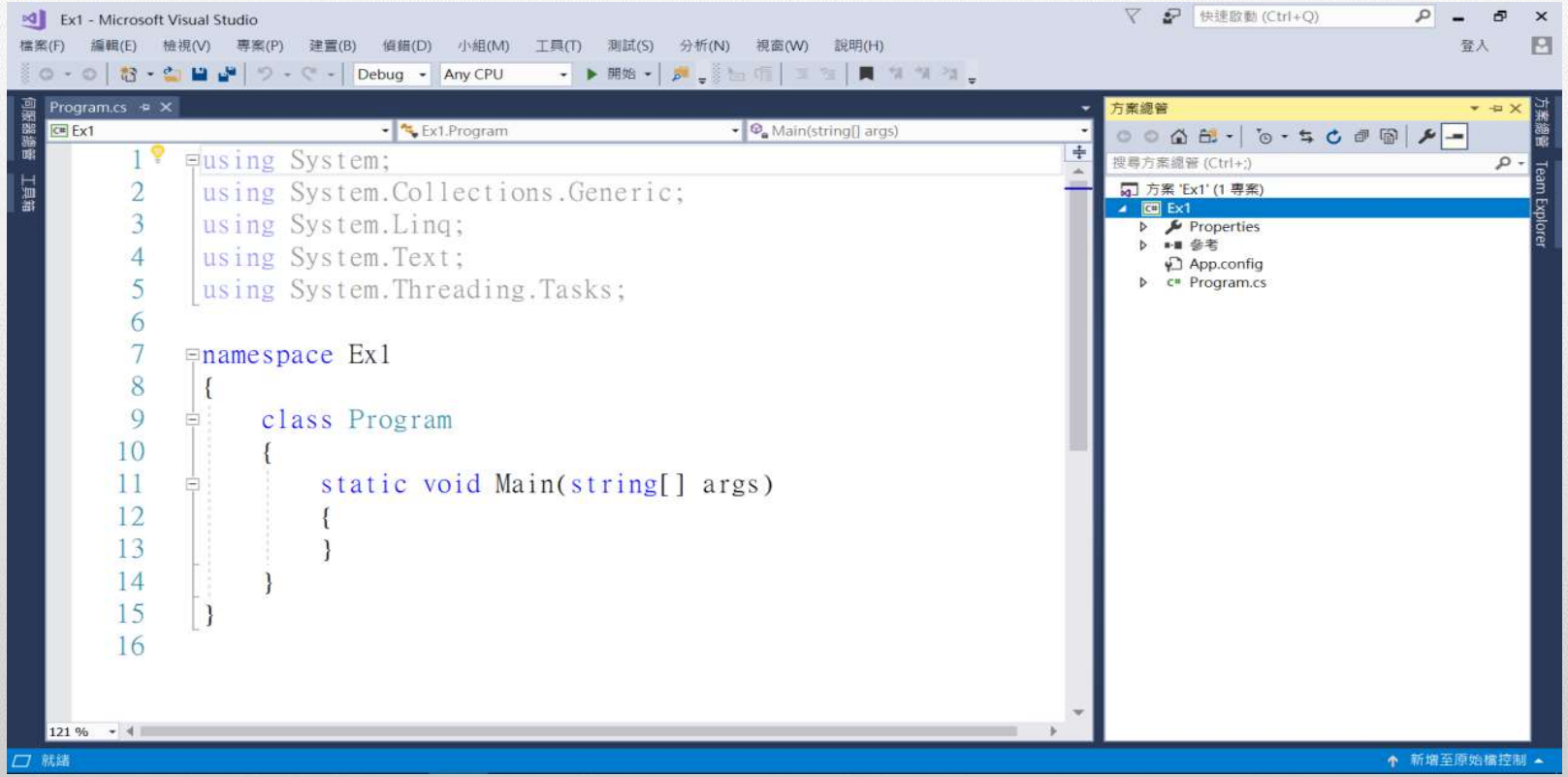


圖1-23　建立主控台應用程式專案(五)

- 且在資料夾D:\C#\ch01 中，會新增一個子資料夾Ex1，且在資料夾Ex1中會產生Ex1.sln方案檔及Ex1.csproj專案檔

# 1-3-4 Visual C# 主控台應用程式的專案架構

- 圖1-23 右邊的「方案總管」視窗內容，是建立 **Ex1** 專案時所產生的專案架構
- 「方案總管」主要用來管理專案及其相關資訊
  - 透過「方案總管」，可以輕鬆存取專案中的檔案
- 專案架構中的項目，包括
  - 方案名稱**Ex1**
  - 專案名稱**Ex1**
  - Properties
  - 參考
  - App.config
  - Program.cs

- 方案：用來管理使用者所建立的專案
  - 一個方案底下可以同時建立多個專案，使用者可以點選方案中的專案名稱，並對它進行移除、更名、... 等作業處理
  - Visual Studio 將方案的定義，儲存在「方案名稱.sln」檔案和「.suo」( 方案使用者選項)檔案中
  - 圖1-23 右上方的「方案總管」視窗內的方案名稱Ex1 與專案名稱Ex1 同名，是建立Ex1 專案時自動產生的，同時在資料夾D:\C#\ch01\Ex1 中也會自動產生一個Ex1.sln 檔，其內容主要記錄專案和方案的相關資訊。「.suo」檔案是儲存方案時自動產生的一個二進位檔，用來記錄使用者處理方案時所做的選項設定，它位於資料夾D:\C#\ch01\Ex1\.vs\Ex1\v15 中。

- 專案：主要記錄與此專案相關的資訊，包括Properties、參考、App.config 及Program.cs。
  - 使用者可以點選專案中的檔案名稱或項目，並對它進行刪除、更名、移除、... 等作業處理
  - Visual Studio 將專案的定義，儲存在專案名稱.csproj中
    - 以圖1-23 右上方的「方案總管」視窗內的Ex1 專案為例，在資料夾「D:\C#\ch01\Ex1」中包含一個Ex1.csproj 檔

- Properties：用來記錄專案的版本等資訊。
- 參考：是存放Microsoft公司或個人或第三方公司所開發的組件（.dll）區
  - 若在專案程式中引用（using）「參考」中的組件，就能使用組建中的類別
- App.config：記錄專案的組態設定，記錄xml的版本、原始程式碼的字元編碼方式、.NET Framework 的版本、…等
- Program.cs：為專案預設的啟動程式檔名稱
  - 一個專案，可以同時建立多個類別檔(.cs)

- 圖1-23 左邊的「程式碼」視窗內容，是建立Ex1專案時，預設的程式碼
  - 「程式碼」開頭的using…敘述區，提供程式設計者引用Microsoft 公司或個人或第三方公司所開發的元件，這樣就不必重新撰寫已存在的類別，使撰寫程式更有效率
  - namespace Ex1是建立Ex1專案時，系統預設以專案名稱Ex1去定義Ex1命名空間，且會自動建立Ex1資料夾，將與Ex1專案有關的所有檔案都儲存在這裡
  - 「Program」類別中的「Main(…)」主方法，是核心程式撰寫區，且是應用程式的主要進入點

- 撰寫「主控台應用程式(.NET Framework)」時，在Visual Studio整合開發環境中，較常使用的工作視窗有
  - 「方案總管」視窗：可以瀏覽、新增或移除方案中的專案及專案所使用的相關檔案
  - 「程式撰寫區」視窗：是Visual C#的原始程式碼撰寫的地方
  - 「錯誤清單」視窗：主要是列出程式編譯時所產生的錯誤訊息

- 「方案總管」、「程式撰寫區」或「錯誤清單」視窗消失時，可分別點選功能表的「檢視(V) / 方案總管(P)」、「Program.cs」或「檢視(V) / 錯誤清單(I)」來開啟

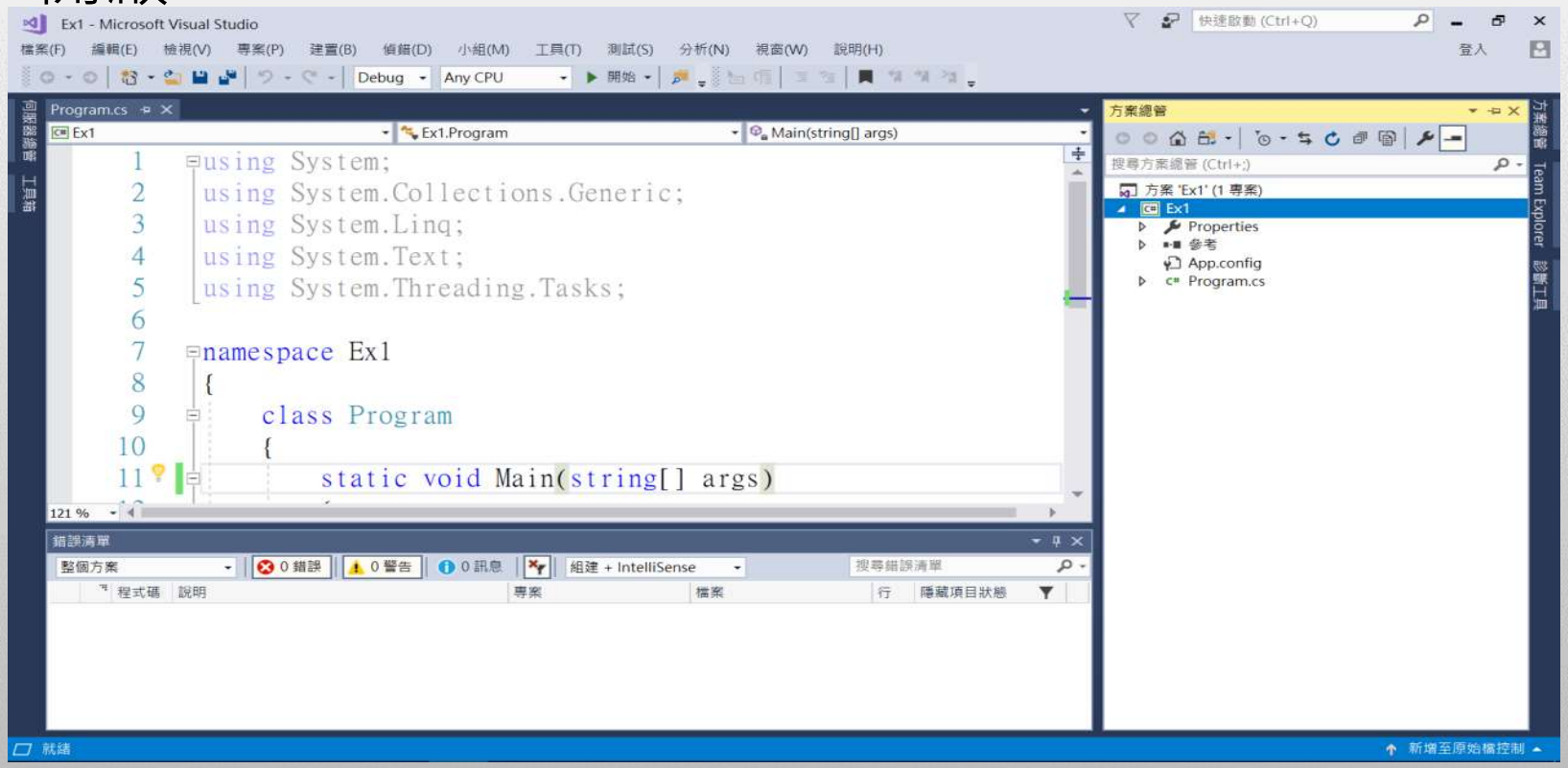


圖1-24方案總管視窗

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Ex1
8  {
9      class Program
10     {
11         static void Main(string[] args)
12         {
13                 // 在螢幕上顯示"歡迎您來到Visual C#的世界!"(並換列)
14                 Console.WriteLine("歡迎您來到歡迎您來到Visual C#的世界!")
15
16             // 程式暫時停止執行，等待使用者按下一按鍵，然後才又繼續執行
17             Console.ReadKey();
18         }
19     }
20 }
```

# ■程式說明

■程式撰寫過程中，當文字排列方式雜亂時，請點選功能表中的「編輯(E)/進階(V)/格式化文件(Ctrl+K，Ctrl+D)」，就能排列整齊

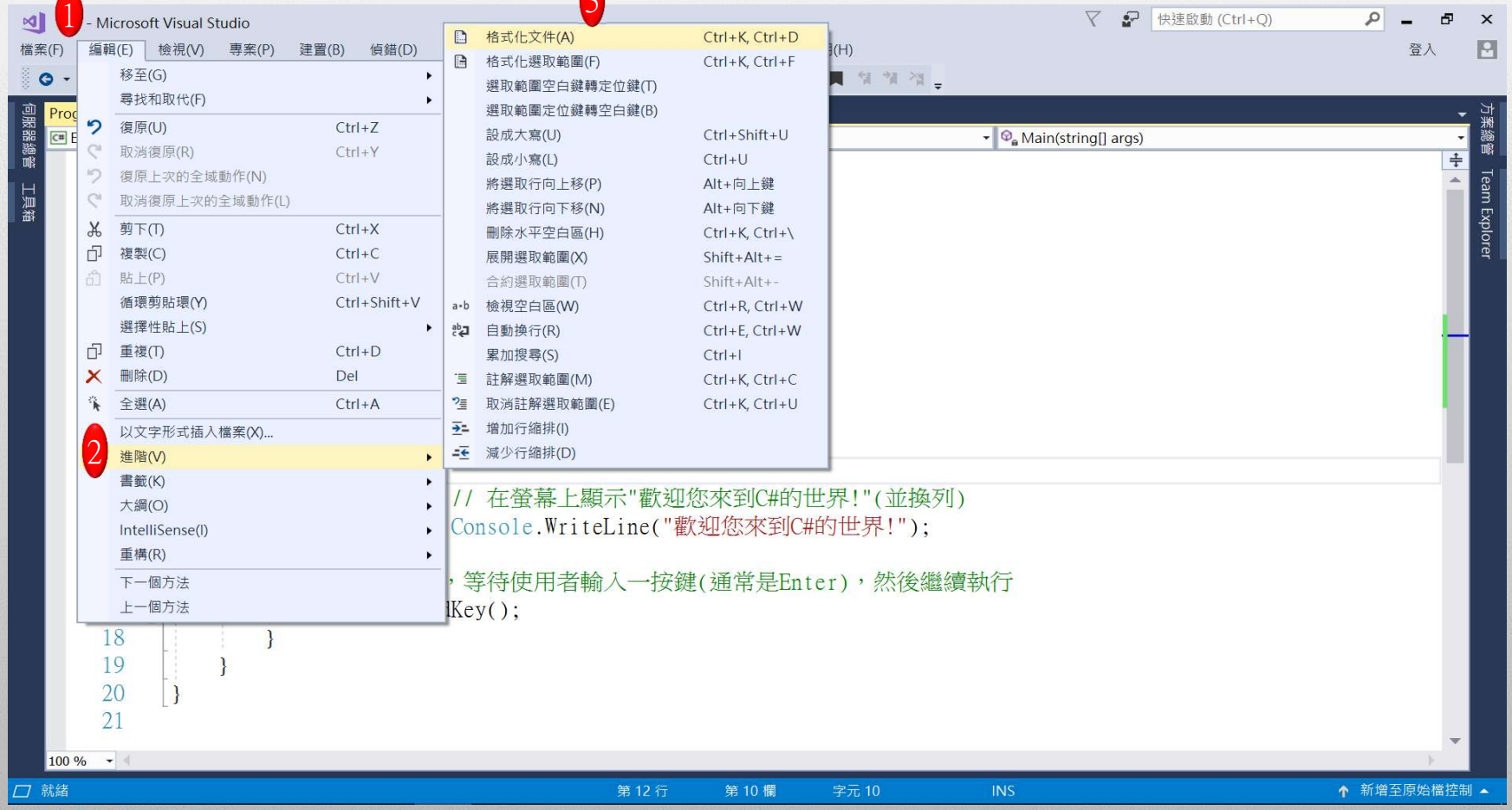


圖1-25　主控台應用程式程式碼排列美化

- 專案原始程式碼完成後，接著點選工具列的「開始」，編譯專案原始程式碼並執行
  - 若沒有產生錯誤，則會在看到執行結果；否則請修正原始程式碼，再重複此步驟

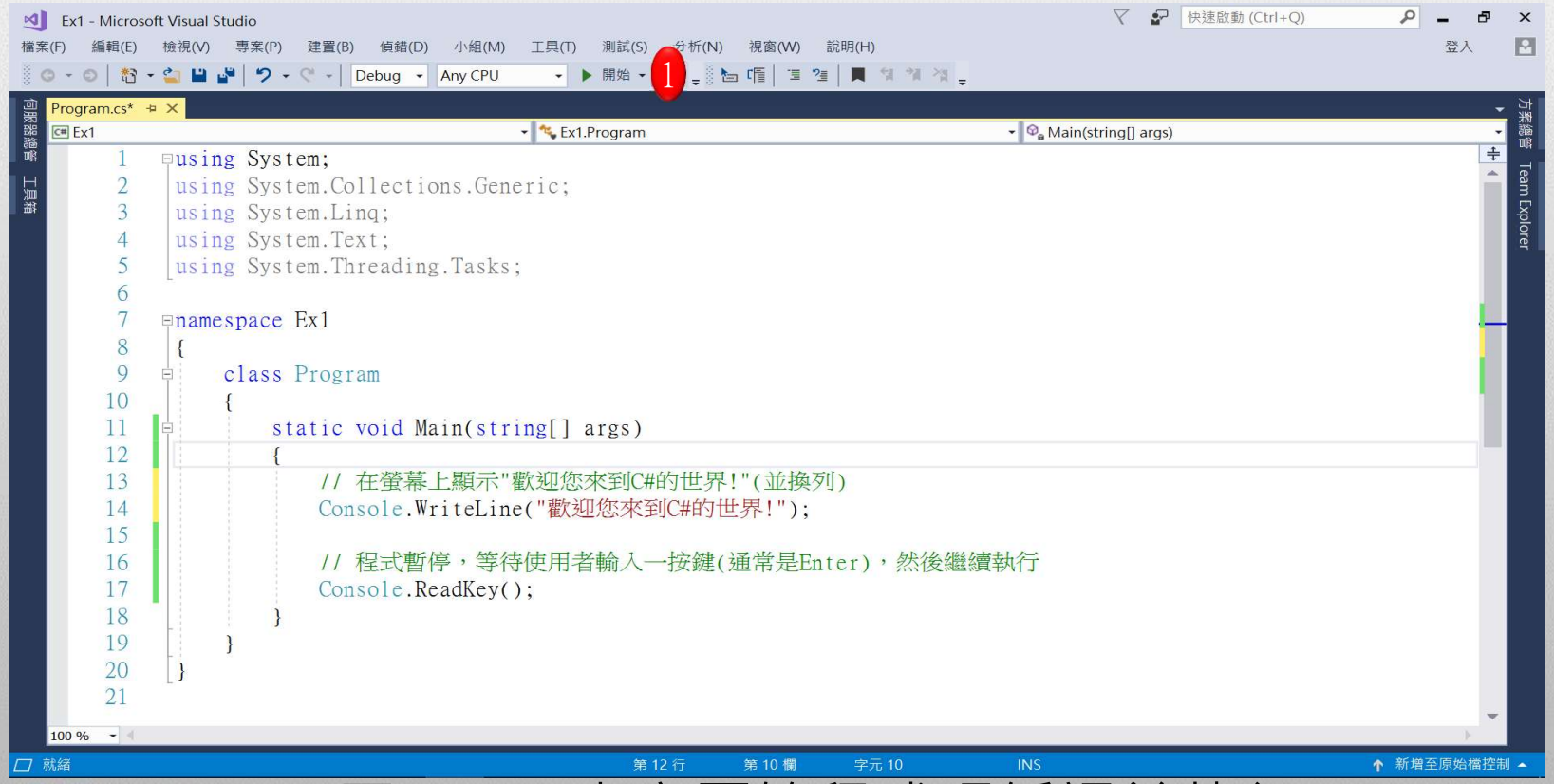


圖1-26　專案原始程式碼編譯並執行

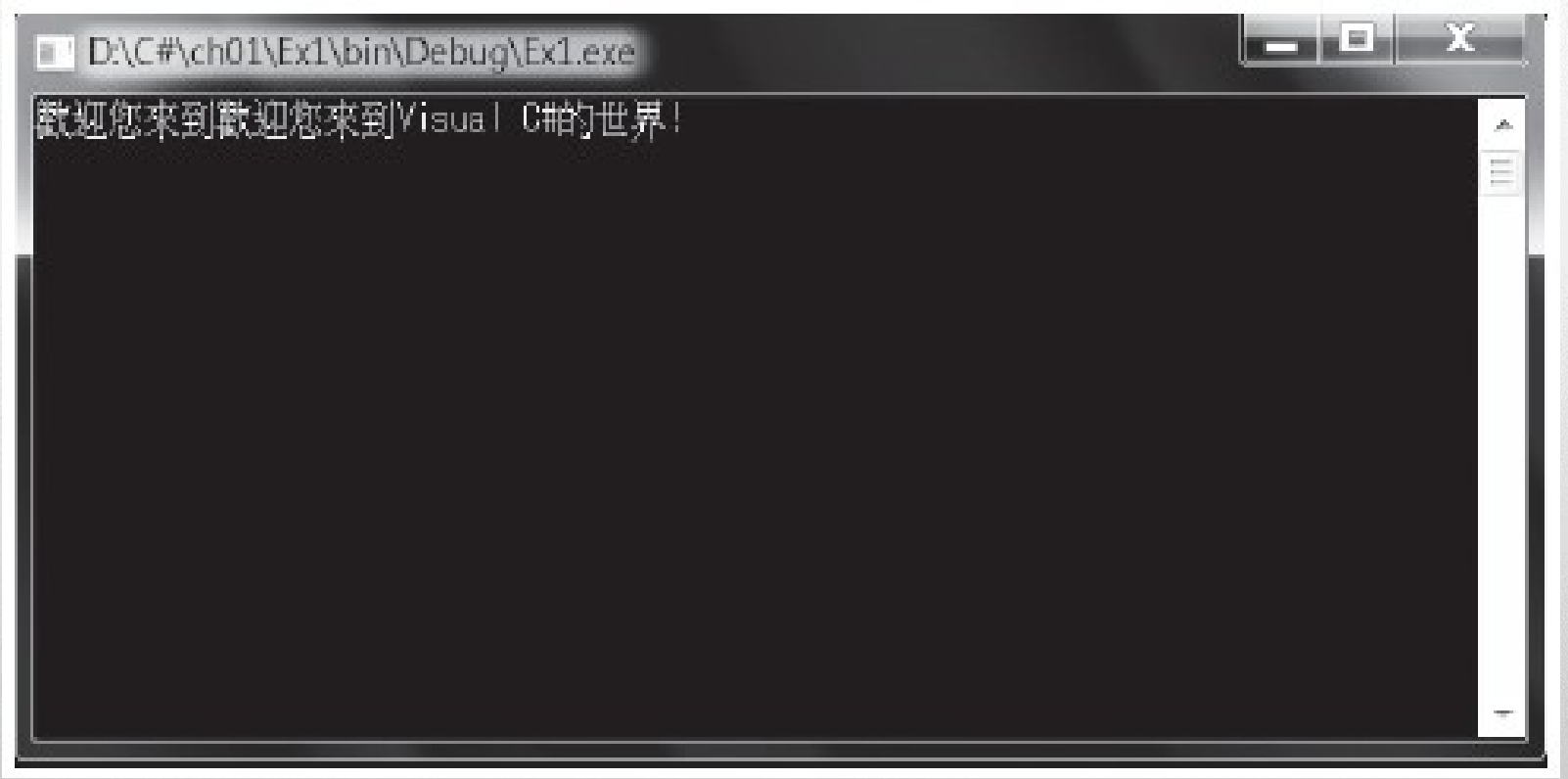


圖1-27　主控台應用程式執行結果

# 1-3-5 專案管理

■若要在專案Ex1中新增與專案相關的檔案，則可對著「方案總管」視窗中的專案名稱Ex1按滑鼠右鍵，點選「加入(D) / 新增項目(W)」

■ 以新增Second.cs類別檔為

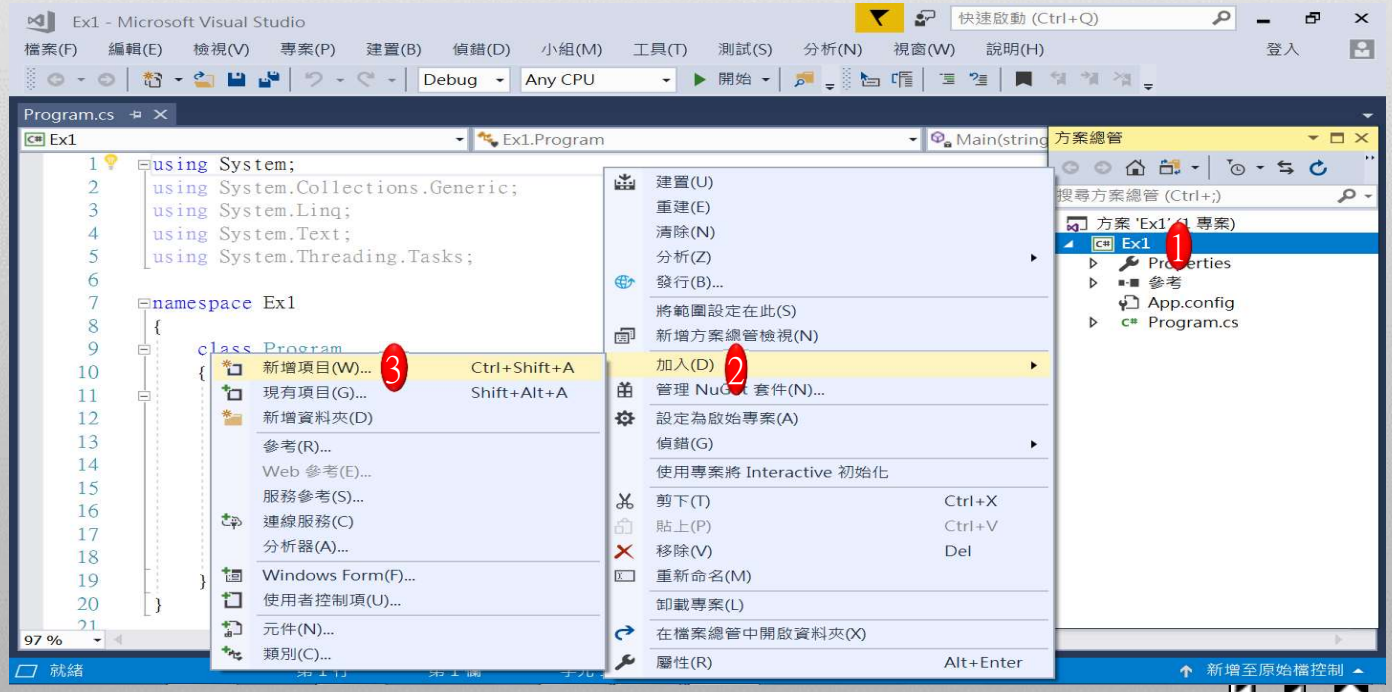


圖1-28　在Ex1專案中新增Second.cs檔案(一)

■ 點選「Visual C# 項目 / 類別」，在「名稱(N)」中輸入「Second.cs」，並按「新增(A)」

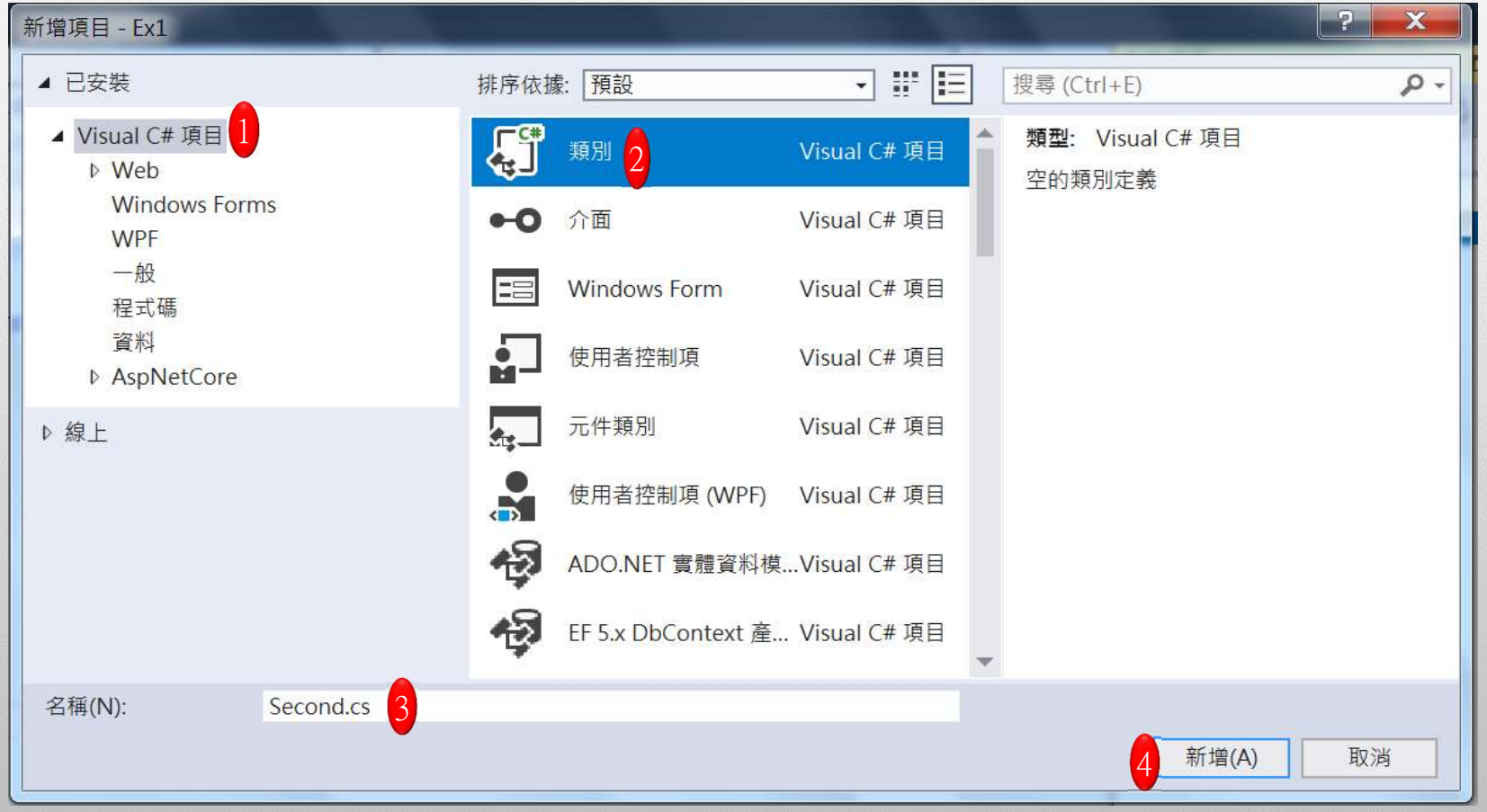


圖1-29　在Ex1專案中新增Second.cs檔案(二)

- 完成後，在「方案總管」視窗中，會產生「Second.cs」檔案名稱。

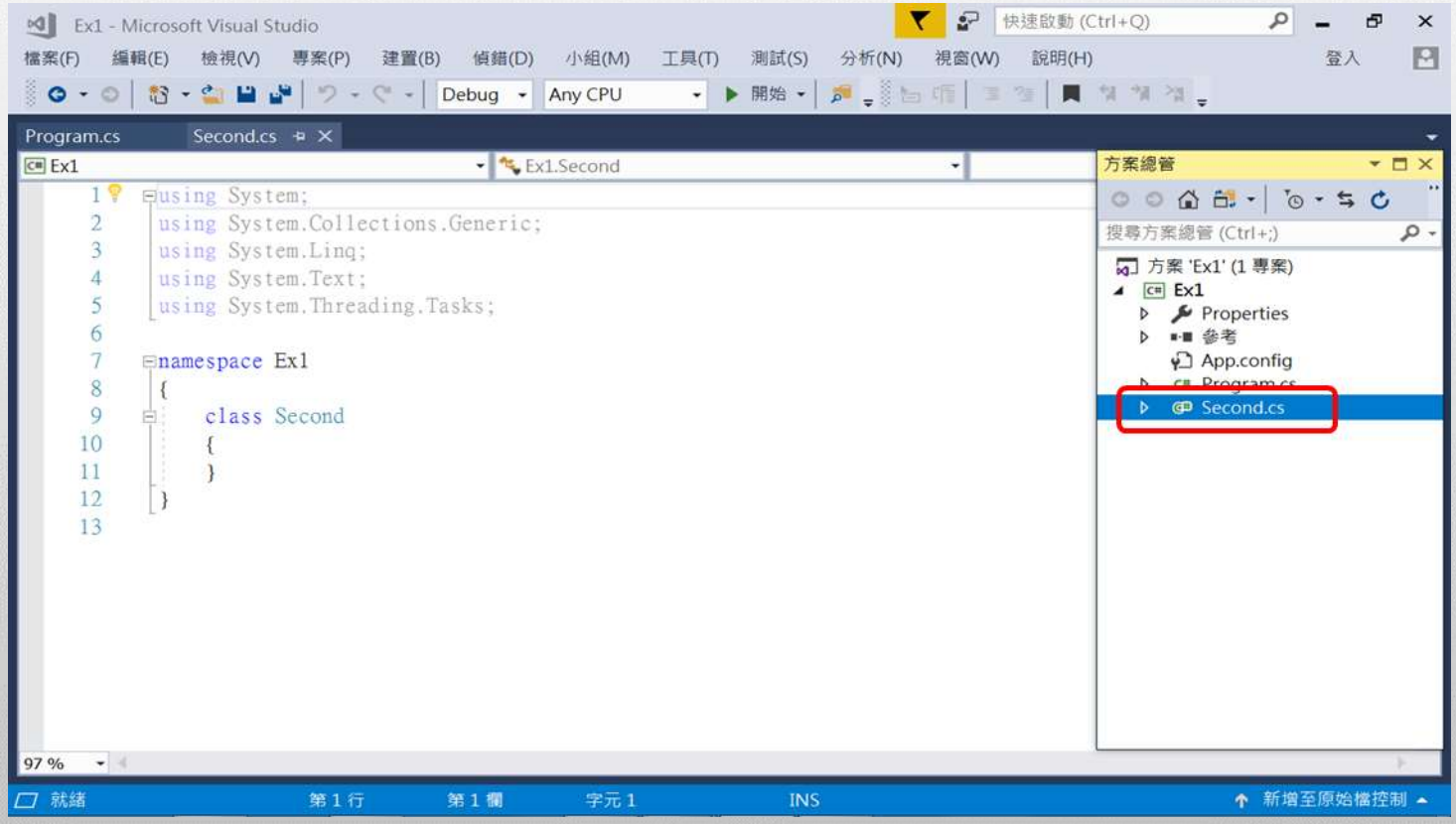


圖1-30　在Ex1專案中新增Second.cs檔案(三)

# 1-3-6 方案管理

- 一個Visual C#的方案，可以包含多個專案
  - 若要在方案內部新增一專案，則可對著「方案總管」視窗中的方案「名稱」按滑鼠右鍵，點選「加入(D) / 新增專案(N)」或「加入(D) / 現有專案(E)」
  - 若要設定方案中的一專案為啟始專案，則可對著「方案總管」視窗中的該專案「名稱」按「右鍵」，點選「設定啟始專案(A)」

# 1-4 Visual C#程式語言架構

- Visual C# 程式語言的「主控台應用程式(.NET Framework)」架構，撰寫順序依次為：
  - 命名空間引用區：
    1. 建立專案程式時，會在命名空間引用區中，自動產生以下程式敘述：

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

- **關鍵字「using」的目的，是告訴編譯器目前的專案程式檔（.csproj）引用哪些命名空間**
  - 引用「命名空間」名稱後，就能直接使用此「命名空間」中的類別，能簡化程式的撰寫及已存在程式碼的再利用
- **以上所列的命名空間名稱：System、…、**
  - System.Threading.Tasks，都是Visual C# 預設的命名空間
    - 其他Visual C#的命名空間， 可參考.NET Framework類別庫（https://msdn.microsoft.com/zh-w/tlibrary/gg145045(v=vs.110).aspx）

- 在原始程式碼中，除了能引用Visual C# 所提供的命名空間外，還能引用自行設計的命名空間。語法如下：

  using 命名空間名稱;

- 編譯原始程式碼過程中，遇到無法辨識的識別名稱，系統會自動比對已引用的「命名空間」中是否包含此無法辨識的識別名稱
  - 若包含，則可以通過編譯，否則會出現編譯錯誤

- 例：（以下為一程式的部分內容，假設命名空間Test中包含類別Welcome，不包含類別welcome）

```
using  Test;
  …
  welcome …
  …
```

- 因命名空間Test中，無welcome類別，故編譯器無法辨識welcome，編譯時產生以下錯誤訊息：

  名稱 'welcome' 不存在於目前的內容中

2. 命名空間（namespace）宣告區：

- 建立專案程式時，專案名稱若設定為Ex1（假設），則在專案原始程式碼中，會自動建立Ex1命名空間區段：namespace Ex1{ }
  - 且在Ex1命名空間區內自動建立Program主類別區段：class Program { }
  - 並在Program主類別定義區中，自動建立Program主類別的static void Main(…) { } 主方法區段

- 在主類別Program中的static void Main(string[] args) { } 主方法上方， 還可以加入Program主類別的屬性宣告， 代表Program主類別的特徵
- 在static voidMain(string[] args) { } 主方法下方，還可以定義Program主類別的其他方法，代表Program主類別的其他行為
- 在class Program { } 主類別區段的下方，還可以定義其他的類別區
  - 在這個類別區內，可以宣告屬於該類別的屬性及定義該類別的方法，分別代表該類別的特徵及行為
  - 此區可以同時連續定義多個類別區

- Visual C#語言的程式架構，是由class（類別）組成。每一個可被獨立執行的專案程式檔（.csproj），必須包含兩個成員：Program主類別及Main() 主方法
- 每個可以被直接執行的「主控台應用程式」，必須包含以下14列程式敘述：

- 「命名空間」名稱必須與專案程式檔（.csproj）的名稱相同
  - 「命名空間」名稱及類別名稱的字首以大寫為原則

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace 命名空間名稱
 {
 class Program  //主類別定義區
  {
  static void Main(string[] args)      //主方法定義區
    {
    }
  }
 }
```

- static void Main(string[] args) { }主方法是Visual C#程式的進入點
  - **Main(string[] args) 前面的void表示程式執行結束時，不會回傳任何資料給作業系統**
  - **以關鍵字static（靜態）定義的方法，稱為靜態方法**
    - 它在執行Visual C#程式時，會立刻自動被建立或執行
    - 因此，執行Visual C#程式時，static void Main(string[] args) { } 主方法會自動執行
  - **static void Main(string[] args)括號內的參數（args），是負責接收執行程式時所傳入的實際字串陣列，而實際字串陣列可有可無**
- class（類別）的相關說明，參考：第九章 自訂類別

- 註解是寫給人看的，主要是為了增加程式的可讀性，並降低程式維護時間。由於註解會被編譯器忽略，而不做任何處理
  - 註解可寫可不寫
  - 寫在「//」後的那些文字，稱為「單行文字註解」（Single comment），但文字不可超過一列
  - 也可用「/* 文字 */」來表示「多行文字註解」（Multiple comment）
  - 註解不能以巢狀形式呈現。例：/*.../*.../*...*/
  - 另外有一種多行註解「///」，用來說明類別、介面、方法等的目的及其他說明
  - 請參考「第九章　自訂類別」的範例1

- 「{」及「}」為程式區塊的開始敘述及結束敘述
- 「;」表示一個程式敘述的結束
  - 大多數的程式敘述尾部，都要加上「;」
  - 只有少數程式敘述，不必在尾部加上「;」
    - 例：「{」、「}」、「if」、「else」、「else if」、「switch」、「for」、「while」、「do while」、「class」定義的首列、「方法」定義的首列、「interface」定義的首列等。

# 1-5 良好的撰寫程式方式

- 良好的撰寫程式方式：
  - 一列一個指令敘述：方便程式閱讀及除錯
  - 程式碼的適度內縮：內縮是指程式碼往右移動幾個**空格**的意思
    - 當程式碼屬於多層結構時，適度內縮內層的程式碼，使程式具有層次感，方便程式閱讀及除錯
  - 善用註解：讓程式碼容易被了解及程式的維護和擴充更快速方便

# 1-5-1 撰寫程式常疏忽的問題

- 忘記使用關鍵字using 引用類別所在的命名空間
- 忘記加或多加分號「;」
- 忽略了大小寫字母的不同
- 忽略了不同資料型態間，在使用上的差異性
- 將字元常數與字串常數的表示法混淆。
- 忘記在一個區間的開始處加上「{」，或在一個區間結束處加上「}」
- 將「=」與「==」的用法混淆

# 1-6 隨書光碟之使用說明

- **首先請將隨書光碟內的程式檔，複製到D:\C#資料夾中**
- **接著依下列步驟，即可將光碟內的專案程式載入Visual Studio 2017 整合開發環境：**
  1. 依照1-3-3 節的步驟1 及2，進入Visual Studio 2017 整合開發環境
  2. 在Visual Studio 2017 整合開發環境中，點選功能表的「檔案(F) / 開啟(O) / 專案 /方案(P)」

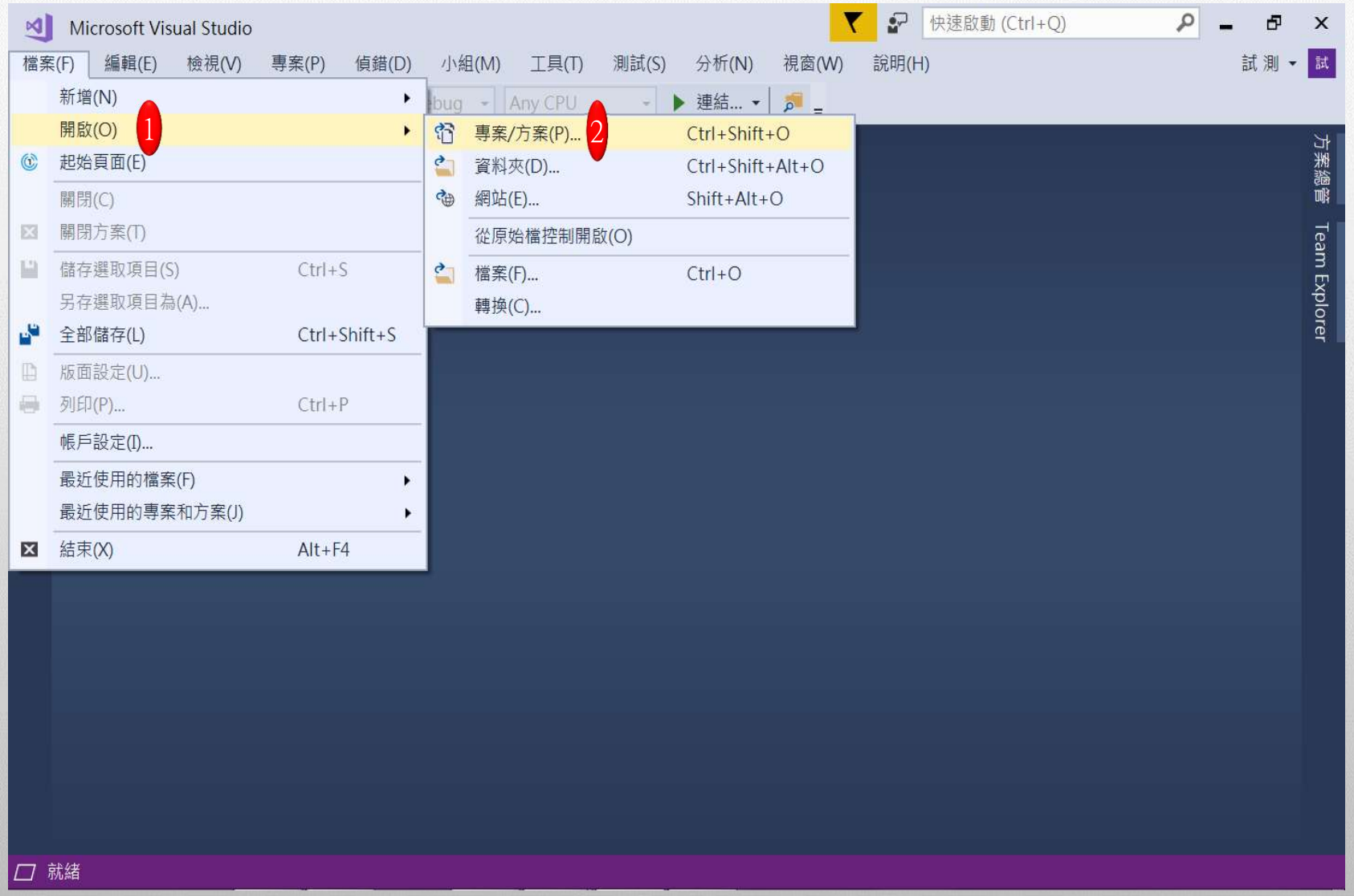


圖1-36　開啟已存在的專案程式(一)

3. 進入該專案所在的資料夾，選取專案檔名稱（.csproj）或方案檔名稱（.sln），並按「開啟(O)」，就能載入該專案程式

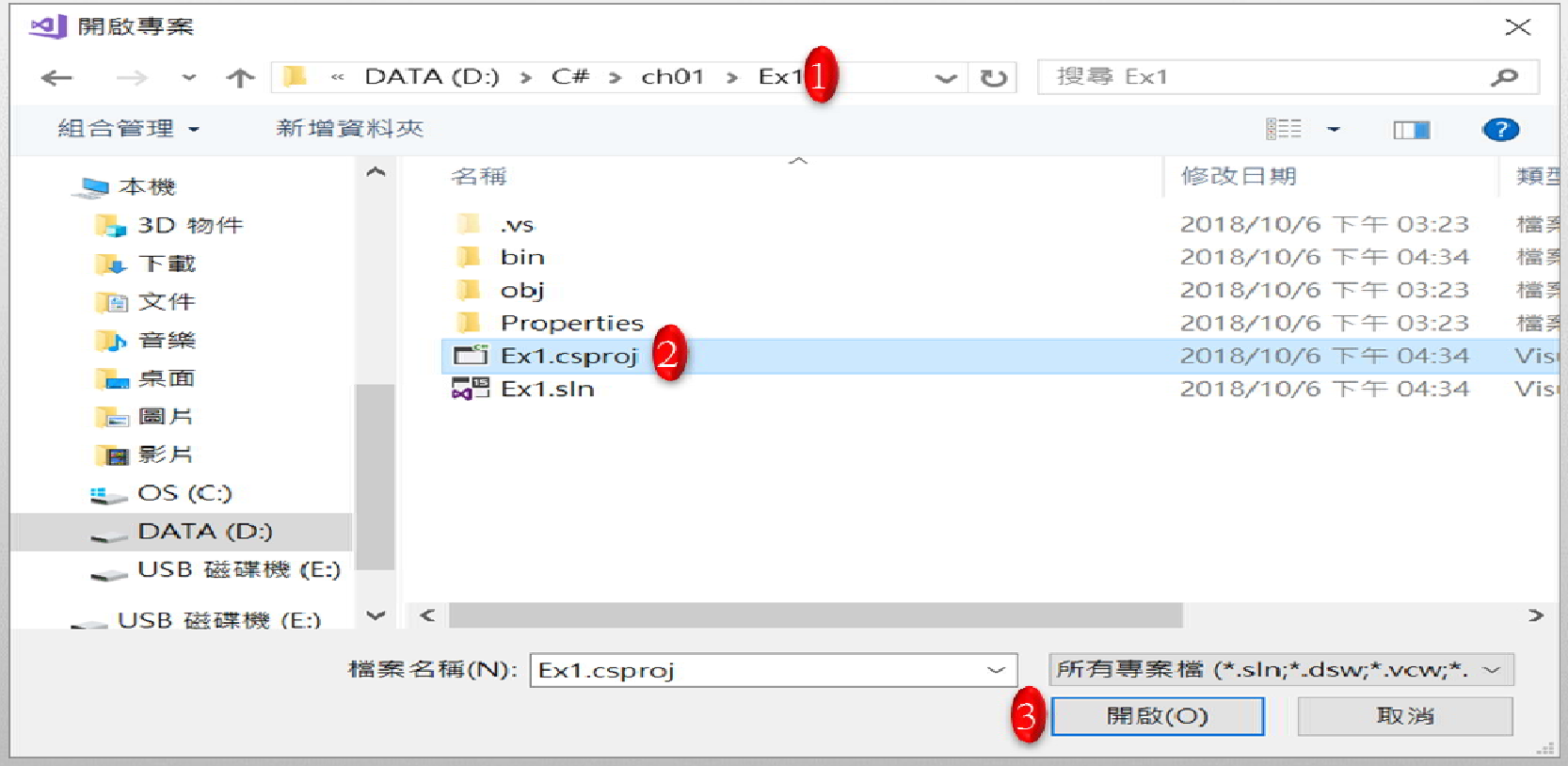


圖1-37　開啟已存在的專案程式(二)

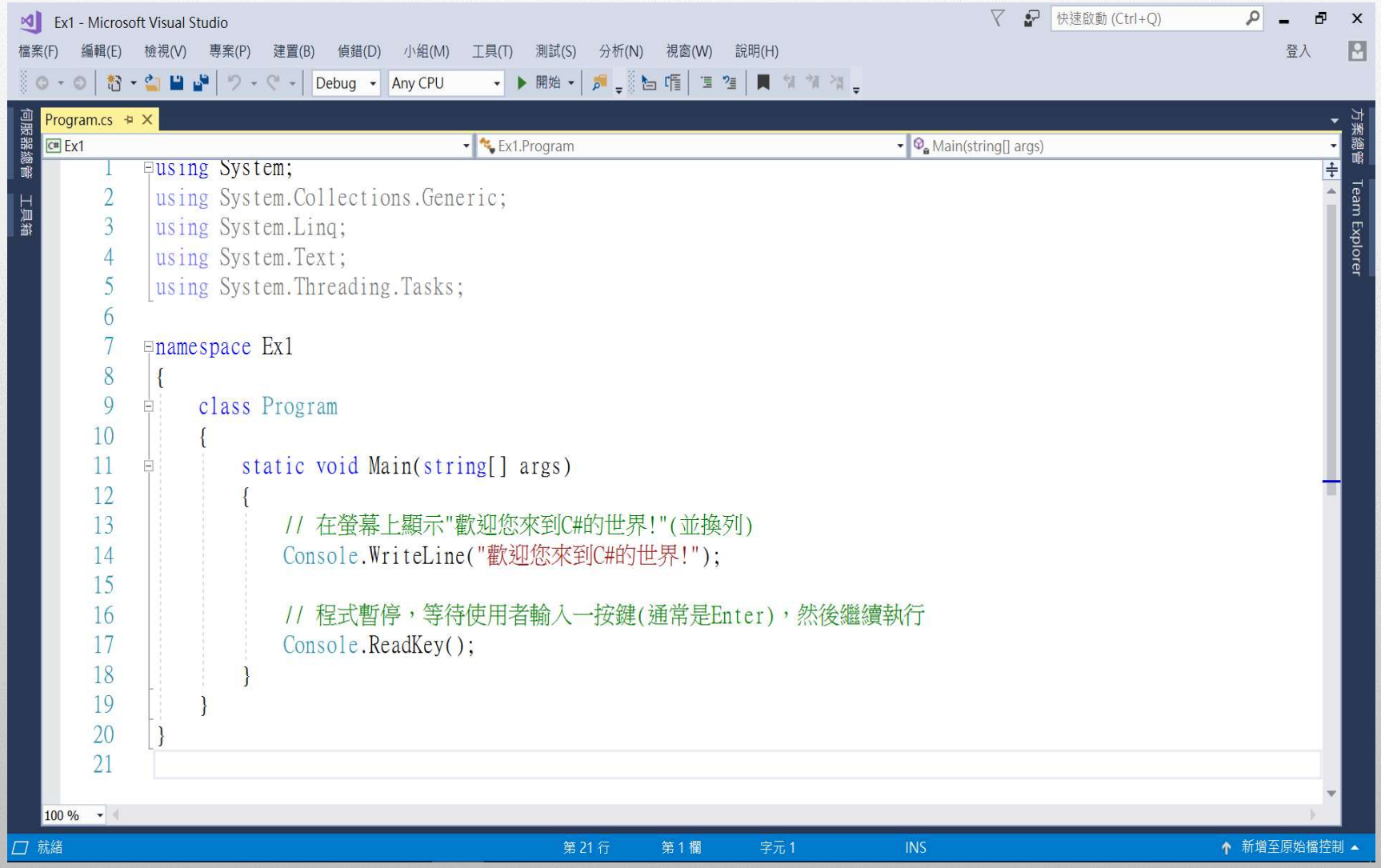


圖1-38　開啟已存在的專案程式(三)