

Appendix

F

Dev-C++ 安裝、操作 環境介紹與除錯

- F-1 下載與安裝 Dev-C++
- F-2 設定 Dev-C++ 開發環境的字體
- F-3 設定 Dev-C++ 編譯器的 C++ 版本
- F-4 使用 Dev-C++ 建立程式
- F-5 Dev-C++ 除錯

F-1 下載與安裝 Dev-C++

Dev-C++ 也是一般常被使用的整合開發環境，截至目前為止所釋放出來的最後版本是 2015 年的 5.11 版，並不算是有持續在維護，甚至都不知道是否會有後續更新；但其也提供了一般性的功能與工具。雖然並不是最好用或功能最齊全，但因其檔案體積小、使用簡單，因此也成為一般人學習 C/C++ 的工具。讀者可在網路上搜尋 Dev-C++ 下載網頁，或由以下網址下載：

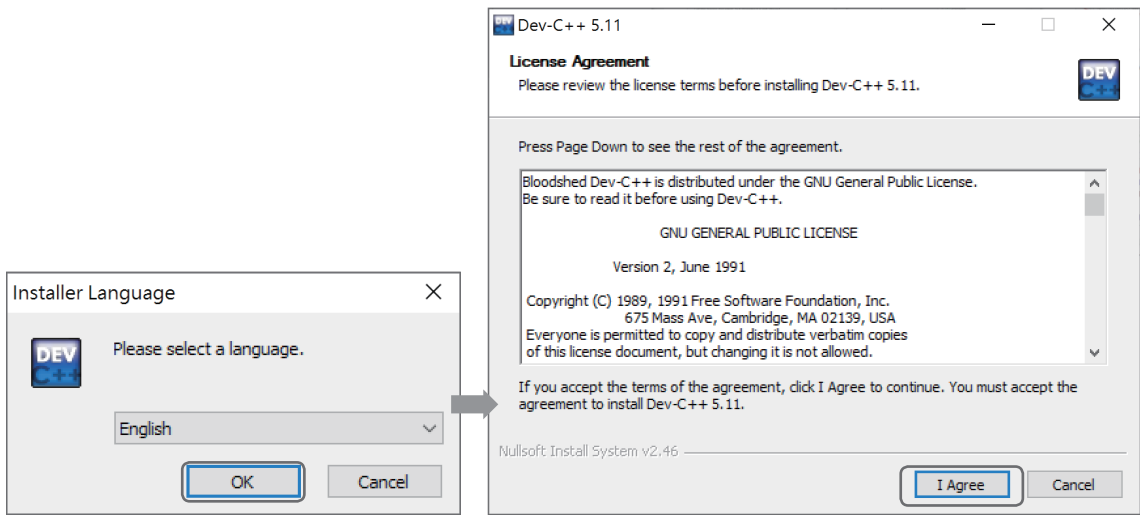
<http://orwelldevcpp.blogspot.com/>

進入網站後，將網頁往下捲動至 [**Download**] 部分，請下載有包含 TDM-GCC 4.2.9 的 Dev-C++ 安裝檔；如下圖所示。

Download

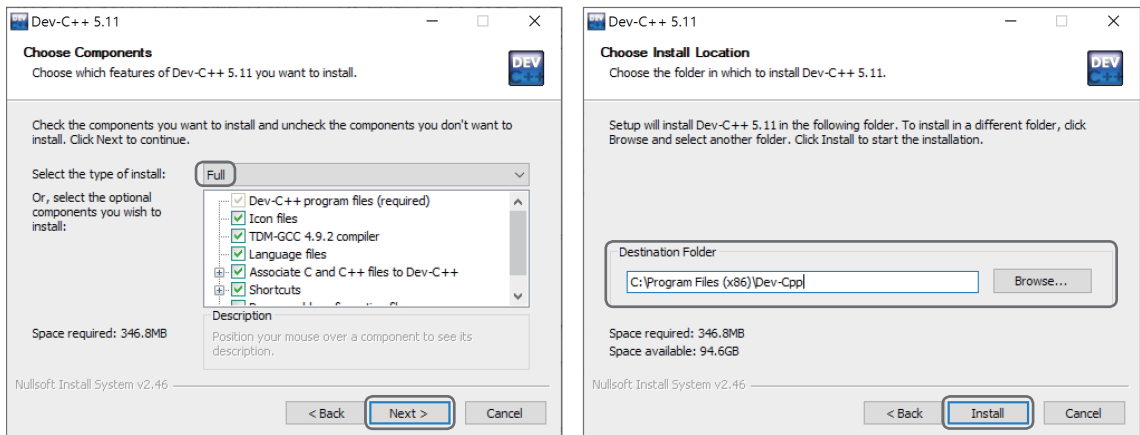
- The **setup** which includes **TDM-GCC 4.9.2** (32bit and 64bit) can be downloaded [here](#) (47MB).
- The **setup** which does not include a compiler can be downloaded [here](#) (2MB).
- The **portable** version which includes **TDM-GCC 4.9.2** (32bit and 64bit) can be downloaded [here](#) (34MB).
- The **portable** version which does not include a compiler can be downloaded [here](#) (2MB).
- The latest tested compilers can be downloaded [here](#).
- Lastly, the source code can be found [here](#) (1MB). Alternatively, one can use git to clone any commit. Instructions can be found [here](#).

下載完成後，執行此安裝檔。如果要選擇中文介面，則在安裝好之後再設定。按「OK」按鈕後進入版權畫面；如下圖左所示。在版權畫面按「I Agree」按鈕後進入元件安裝選擇畫面，如下圖右所示。

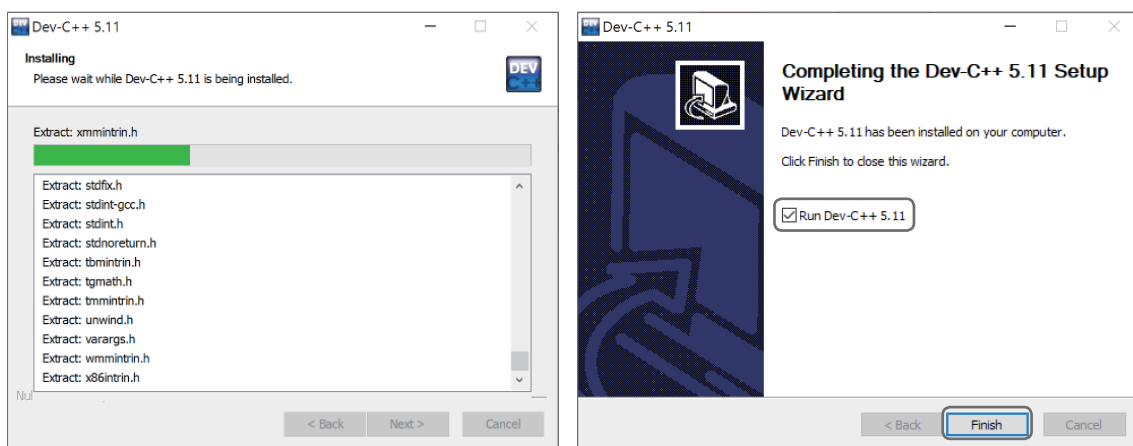


進入元件安裝選擇畫面之後，選擇 **[Full]** 安裝模式安裝所有建議的元件（若想要自訂安裝的元件，就選擇其他的安裝方式。），接著按「Next>」按鈕進入安裝位置的設定畫面。

接著進入安裝路徑設定，如下圖右所示。讀者可以自己決定是否要選擇不同的安裝路徑與目錄，或是使用預設的安裝路徑與目錄。

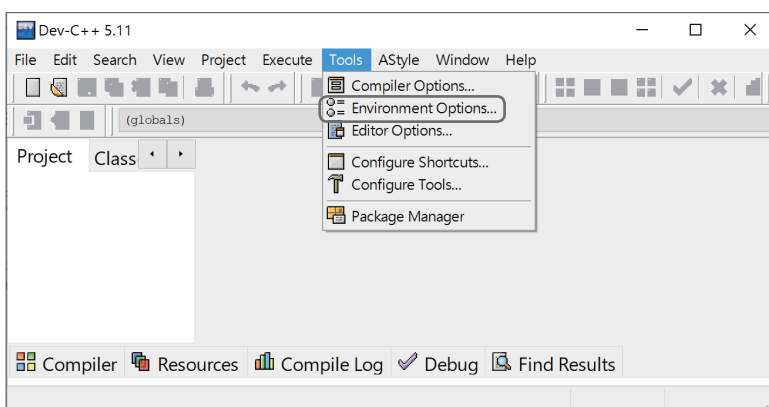


接著按「Install」按鈕開始安裝 Dev-C++。安裝完畢後，按「Finish」按鈕結束安裝，並執行 Dev-C++；如下圖所示。

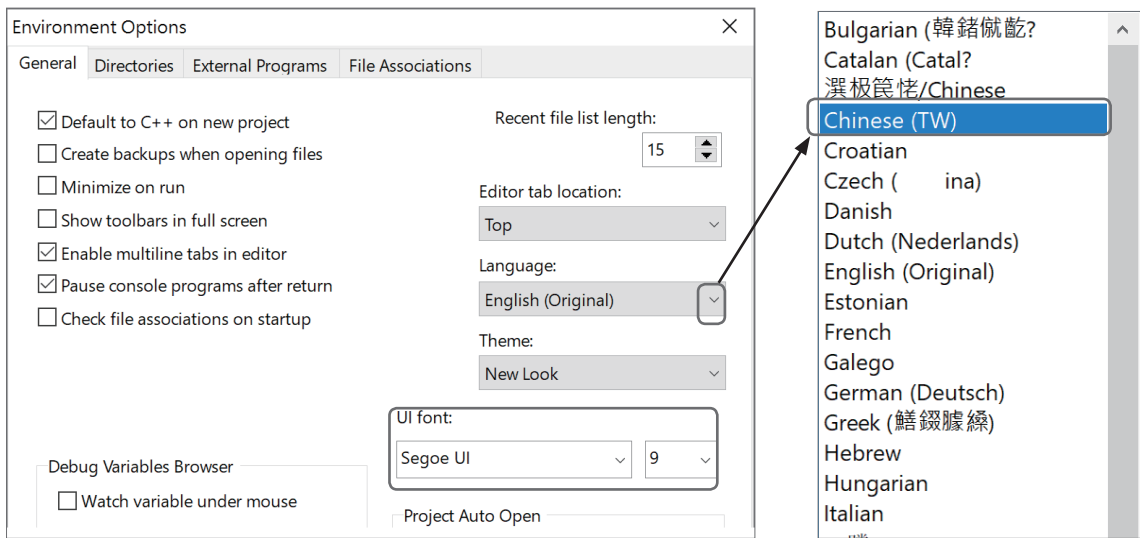


F-2 設定 Dev-C++ 開發環境的字體

執行並進入 Dev-C++ 畫面之後，先將介面設定為中文；如想保持原來的英文介面，則可以略過此步驟。選擇主功能表 [**T**ools]>[**E**nvironment Options...] 項目，如下圖所示。



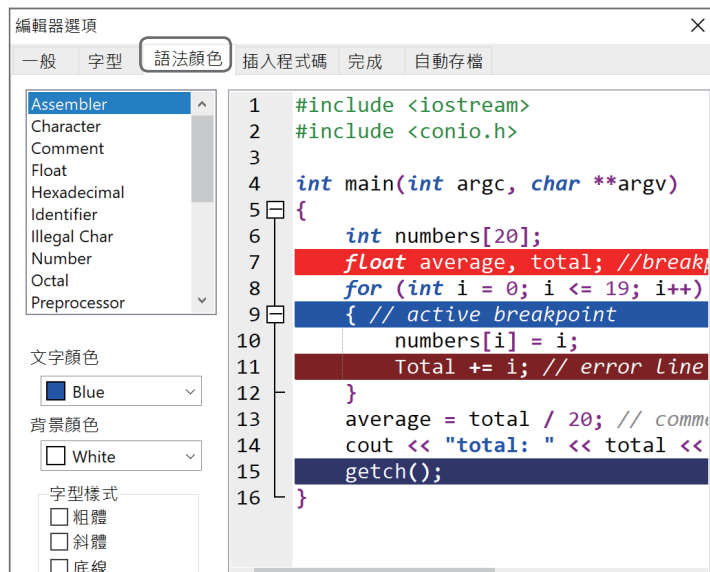
接著在 [**L**anguage] 按一下，在彈出的語言清單中選擇 [**C**hinese(TW)] 後，再按「OK」按鈕將 Dev-C++ 的介面換成中文；如下圖所示。除此之外，[UI font] 項目，也能夠改變介面的字體樣式與大小。



若選擇主功能表的 [工具] > [編輯器選項 (E)]，開啓 [編輯器選項] 後其中 [字型] 頁可以設定編寫程式碼時所使用的字體；如下所示。



在 [語法顏色] 頁面則可以逐一地設定 C/C++ 程式的每項顏色設定。此功能有助閱讀程式，因此讀者可以端視自己的喜好去設定每個項目的顏色。



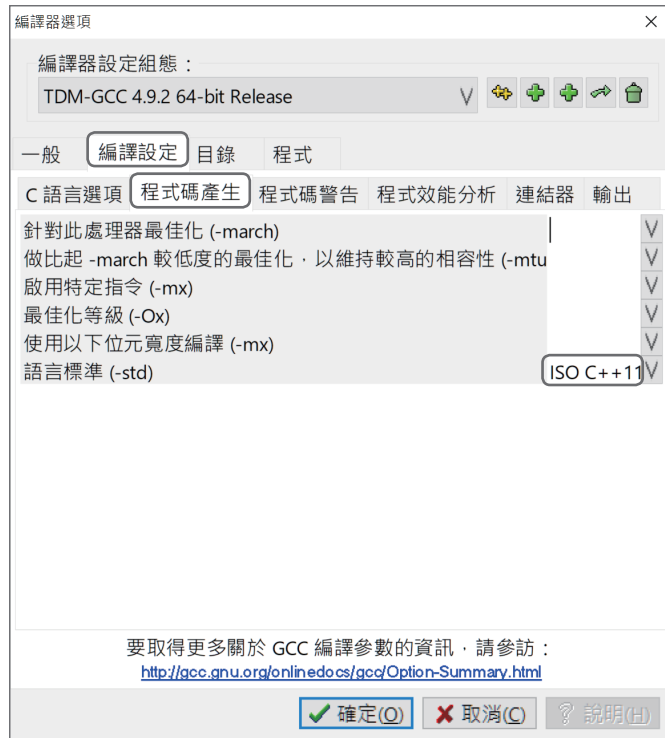
F-3 設定 Dev-C++ 編譯器的 C++ 版本

Dev-C++ 5.11 版本支援 C++ 11 版本，但安裝好之後預設的 C++ 版本為 C++ 98；因此，有些方便的函式以及新的功能無法使用。所以可以依照下列步驟變更預設的 C++ 版本；本書所有的程式碼皆使用 C++ 11 的版本所撰寫。

首先開啓 Dev-C++，在主功能表中選擇 [工具 (T)] > [編譯器選項 (C)]。接著在 [一般] 標籤頁勾選 [呼叫編譯器時使加入以下的命令] 項目；並在下方輸入：-std=c++11；如下圖所示。

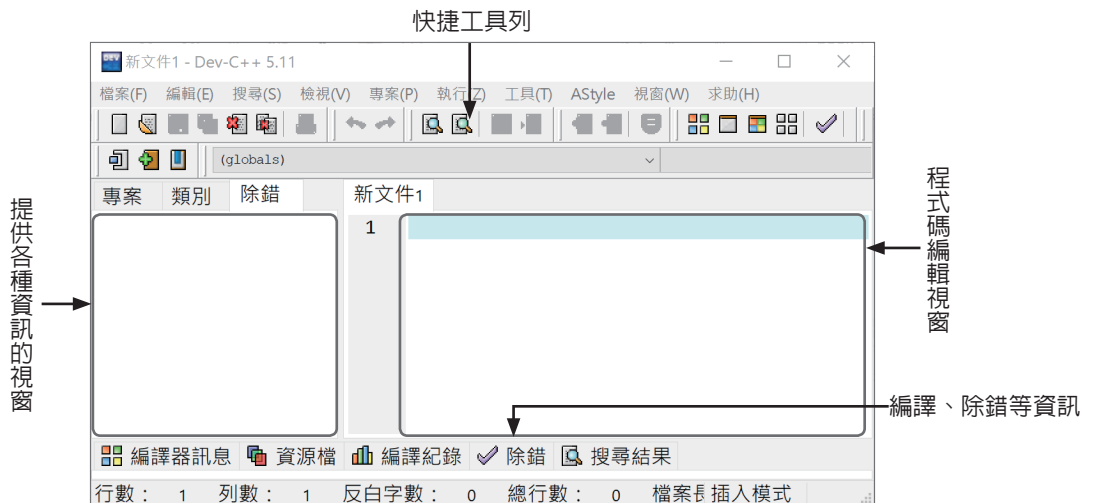


接著，切換到 [編譯設定] 頁面，再點選 [程式碼產生] 子頁面，在語言標準 (-std) 的欄位選擇 IOS C++11；最後點選「確定」按鈕完成設定，如下圖所示。





F-4 使用 Dev-C++ 建立程式

使用 Dev-C++ 開發的程式一樣有兩種類型：獨立程式與專案。獨立程式指的是只有單獨一支的 C++ 程式；而專案指的是包含了一支或一支以上的程式所組合而成的大型程式或系統。開啓 Dev-C++ 後的畫面，如下圖所示。



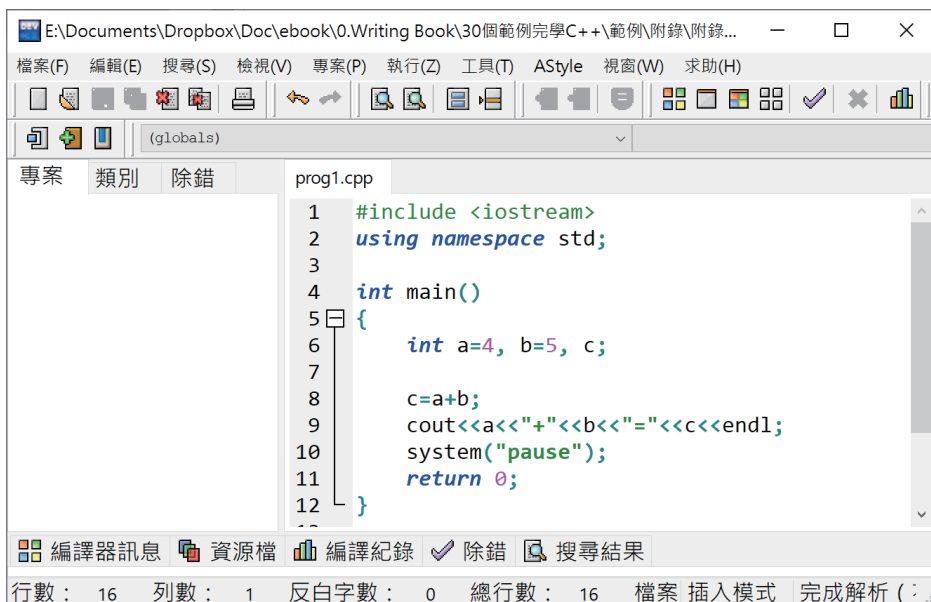
建立獨立程式

開啓 Dev-C++ 之後，點選主功能表的 [檔案 (F)] > [開啓新檔 (N)] > [原始碼 (S)]，接著 Dev-C++ 便會進入程式碼編輯的狀態，如上圖所示。或者按快捷工具列的  圖示後，再點選  原始碼(S) 一樣可以建立獨立原始碼。



接著在程式碼編輯視窗內輸入以下程式碼（不需要輸入行號，英文字母大小寫要一樣）：

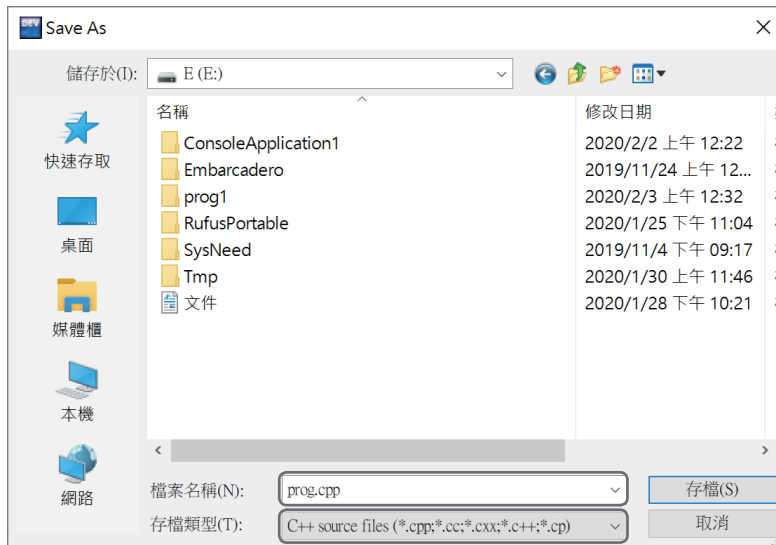
```
1 #include <iostream>
2 using namespace std;
3
4 int main()
5 {
6     int a=4, b=5, c;
7
8     c=a+b;
9     cout<<a<<"+"<<b<<"="<<c<<endl;
10    system("pause");
11    return 0;
12 }
```

輸入完畢後，程式碼編輯視窗內的程式碼應如下圖所示。



儲存程式檔案



接著練習儲存剛才寫好的程式碼。點選主功能表的 [檔案 (F)] > [儲存 (S)]，或者是按快捷工具列的  或  開啟儲存的對話框，如下圖所示。




在儲存對話框裡的 [檔案名稱 (N)] 可以自行輸入檔名。如果是 C 程式則檔案的副檔名為 ".c"；如果是 C++ 程式則副檔名則為 ".cpp"。[存檔類型 (T)] 可以挑選儲存檔案的類型。例如：挑選「C source files (*.c)」的話，則儲存的檔名會自動加上 ".c" 的副檔名；如果挑選的是「C++ source files (*.cpp...)」，則儲存的檔名會自動加上 ".cpp" 的副檔名。

C++ 程式可以接受 C 的語法與函式，而 C 的程式無法辨識 C++ 的語法與函式；因此，通常會以 C++ 的形式存檔，會比較方便。



關閉程式

程式儲存完畢之後，若要關閉正在編輯中的程式，則點選主功能表 [檔案 (F)] > [關閉檔案 (C)] 或 [全部關閉 (U)] 都可以；也可以點選快捷工具上的  或  關閉檔案。

載入程式檔案

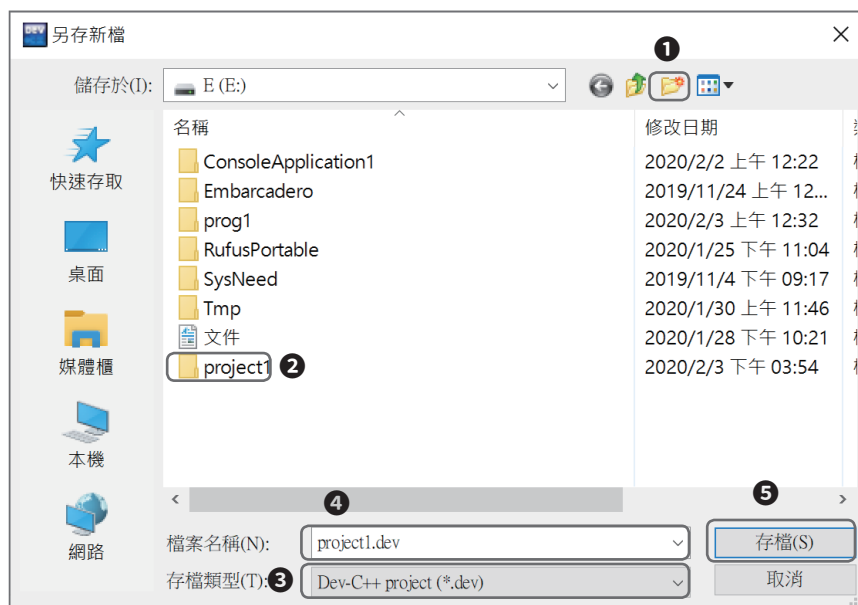
點選主功能表 [檔案 (F)] > [開啟舊檔 (O)]，或是點選快捷工具上的  載入程式，接著選擇欲載入的程式檔案即可。

建立專案

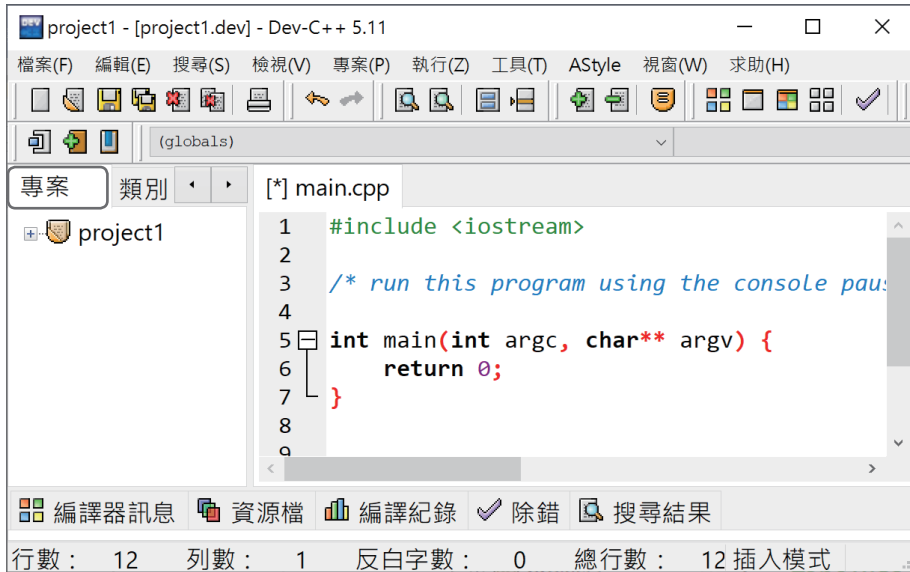
開啓 Dev-C++ 之後，點選主功能表的 [檔案(F)]>[開啓新檔(N)]>[專案(P)...]，接著 Dev-C++ 便會開啓「建立新專案」對話框，如下圖所示。或者按快捷工具列的  圖示後，再點選  專案(P)... 一樣可以建立專案。



對話框上半部分為「Basic」、「Multimedia」、「Win32」與「Console」四大類的程式樣板；請點選 Basic 類的「Console Application」，接著點選「C++ 專案」，再輸入專案名稱之後按「確定」按鈕進入「另存新檔」對話框，如下圖所示。通常專案裡會有多支程式；因此，都會新增一個資料夾來存放這些檔案；通常資料夾的名稱會與專案名稱相同。



新增資料夾 "project1" 之後，存檔類型請選擇「Dev-c++ project(*.dev)」，接著輸入專案的檔名，最後點選「存檔 (S)」按鈕儲存專案。接著 Dev-C++ 會替新的專案建立新的 C++ 程式樣板，如下圖所示。「專案」視窗中會有 project1 專案，以及專案裡有一支程式 main.cpp。

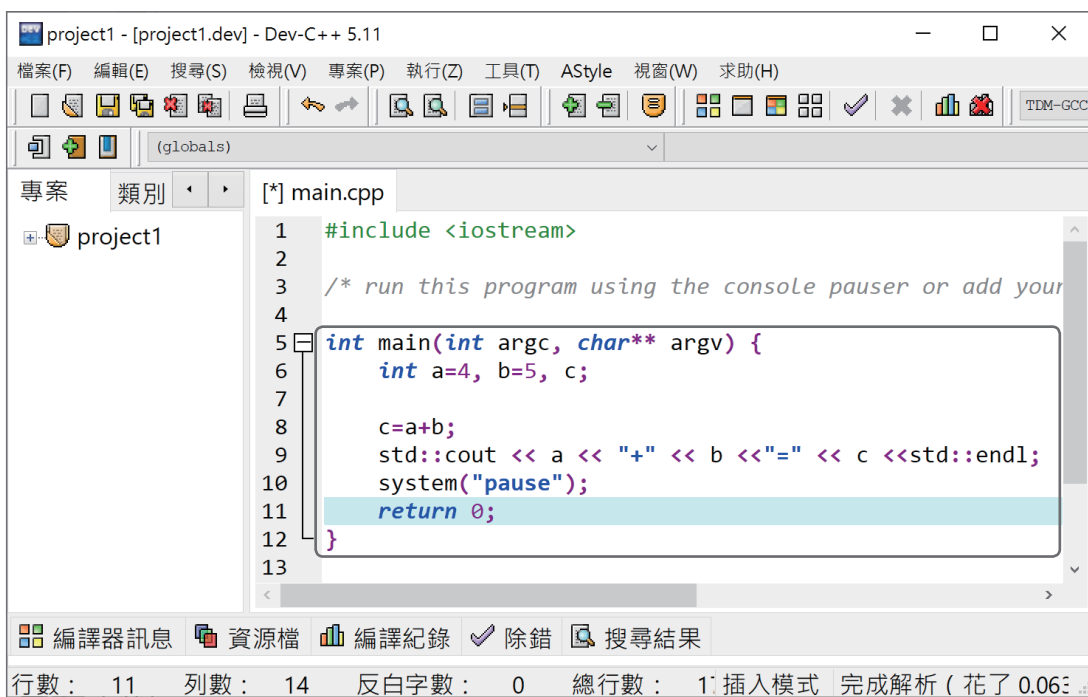


接著輸入程式碼。在程式碼編輯視窗中的程式碼敘述 "return 0" 之前，空出幾列空白列，並鍵入第 6-10 行程式碼：



```

1  #include <iostream>
2
3  /* run this program using the console...input loop */
4
5  int main(int argc, char** argv) {
6      int a=4, b=5, c;
7
8      c=a+b;
9      std::cout << a << "+" << b << "=" << c << std::endl;
10     system("pause");
11     return 0;
12 }
```

輸入完畢後，程式碼編輯視窗內的程式碼應如下圖所示。



儲存專案

接著練習儲存剛才建立的專案。點選主功能表的 [檔案(F)] > [儲存所有檔案(S)]，或者是按快捷工具列的  開啟儲存的對話框（注意，不是按 ）。輸入程式檔名之後，按「存檔(S)」按鈕儲存專案中的程式碼。


此專案裡有一支 C++ 程式，若開啟檔案總管觀察，便可看見此專案資料夾「project1」裡面包含 3 個檔案，如下圖所示。其中 project1.dev 是專案檔，以及專案中的程式碼檔案 main.cpp。



關閉專案檔

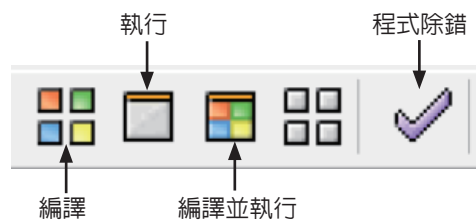
程式儲存完畢之後，若要關閉正在編輯中的專案，則點選主功能表 [檔案 (F)] > [關閉專案 (L)]。

載入專案

點選主功能表 [檔案 (F)] > [開啓舊檔 (O)]，或是點選快捷工具上的  打開「開啓檔案」對話框，接著選擇專案檔 project1.dev 即可。

編譯與執行程式

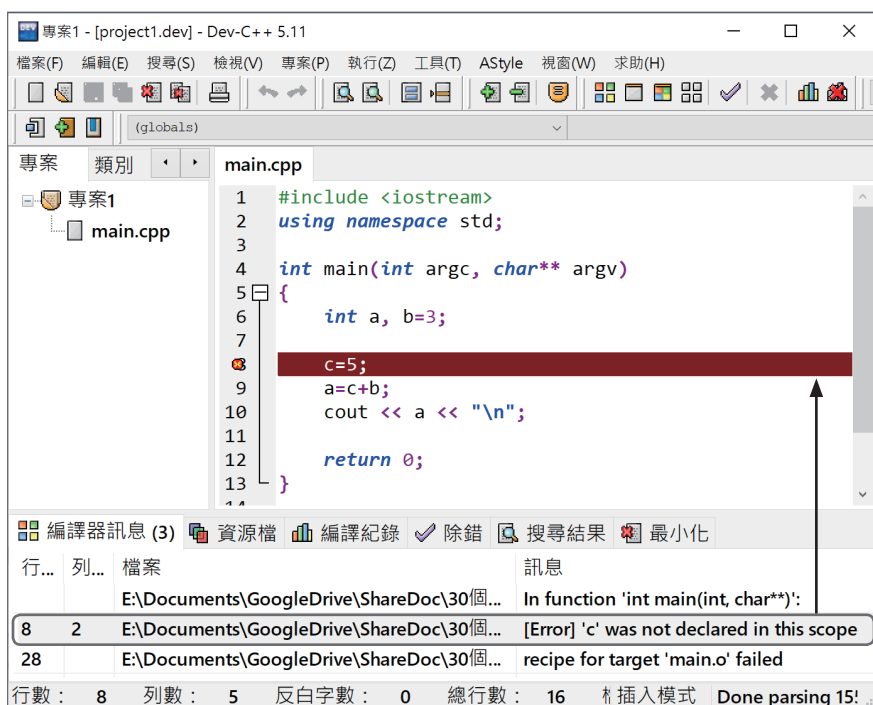
當載入獨立程式或是專案之後，在快捷按鈕可以選擇「編譯」、「執行」、「編譯並執行」。選擇「編譯」，則不會執行編譯後檔案。選擇「執行」，則會執行上一次已經編譯成功的程式，而不是目前在編輯視窗中修改過的程式。因此，通常都會選「編譯並執行」，讓程式重新編譯之後，接著執行程式。



F-5 Dev-C++ 除錯

在撰寫程式時，最常發生的錯誤就是語法錯誤（Syntax error），語法錯誤就是所撰寫的程式敘述不符合 C++ 所規定的語法所產生的錯誤。例如：大小寫錯誤、變數名稱寫錯、程式敘述結尾忘了加上分號、函式寫法錯誤等，有太多的情形會發生語法錯誤。

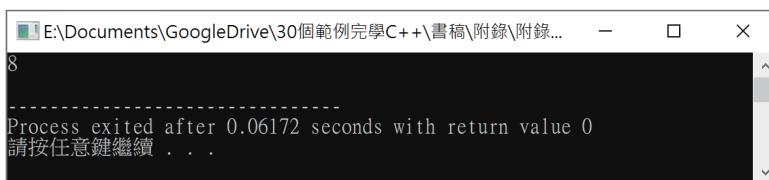
請載入範例程式 01 資料夾裡的 project1 專案並執行，會發現程式因為發生錯誤而中斷，除了在下方的 [編譯器訊息] 視窗會出現錯誤訊息，並且會跳至第 1 個錯誤的程式敘述；或是按 2 下 [編譯器訊息] 視窗中錯誤的訊息，也能跳至發生錯誤的程式碼；如下圖所示。



此錯誤訊息：「'c' was not declared in this scope」表示變數 *c* 沒有在程式中宣告就直接使用。因此，將第 6 行程式敘述加上宣告整數變數 *c*，即修正為：

```
int a, b=3, c;
```

再重新編譯與執行後程式便能正常執行，得到輸出結果如下圖所示。

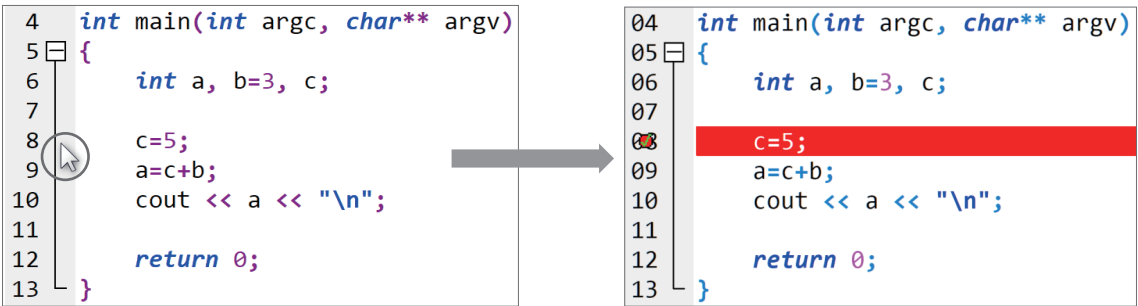


除錯功能



程式執行時，若遇到設有中斷點（Break point）的程式敘述時會暫停執行，並等待後續的處理。中斷點是程式除錯過程中經常被使用的技巧；在懷疑有錯誤的程式敘述設定中斷點，並予以追蹤後續的程式碼、顯示變數的值、檢查程式敘述是否符合預想的結果等。

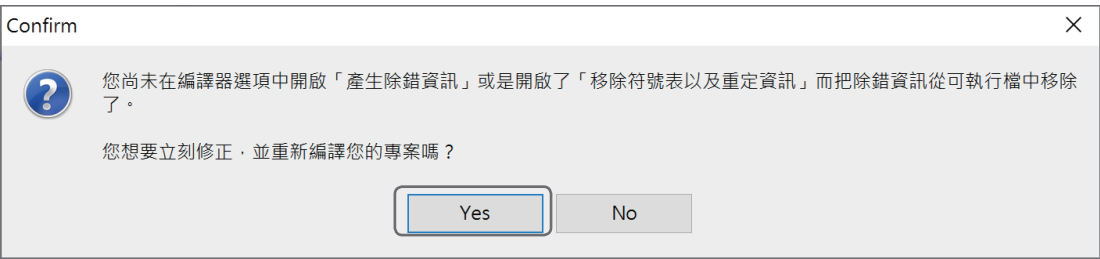
建立中斷點

載入範例程式 02 資料夾裡的 project2 專案，並在程式碼第 8 行的行號 "08" 右邊按一下，如下圖左所示。程式碼第 8 行會以紅色標記，表示此行程式碼建立了中斷點；如下圖右所示。也可以將滑鼠停在第 8 行程式碼的任何地方，接著點選主功能表 [執行]>[加入 / 移除中斷點]（或是按 F4），也能建立中斷點；只要在編號 "08" 右邊再按一下即可移除中斷點。



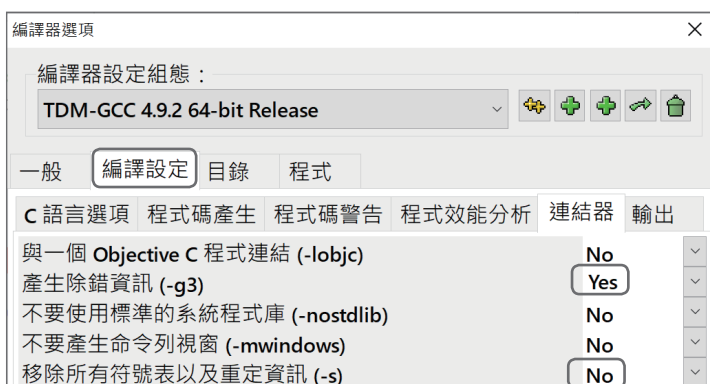
開始除錯

設定好中斷點之後，點選主功能表的 [執行]>[除錯]（或是按 F5），或是按快捷工具列上的  進行除錯。如果是第一次進行除錯，則會顯示如下畫面，表示尚未設定除錯資訊；則點選「Yes」按鈕進行自動設定即可。注意，若是程式碼有修改過，則必須重新編譯（或是按 F9），或是按快捷工具列的 ，否則仍然是針對舊的程式進行除錯。




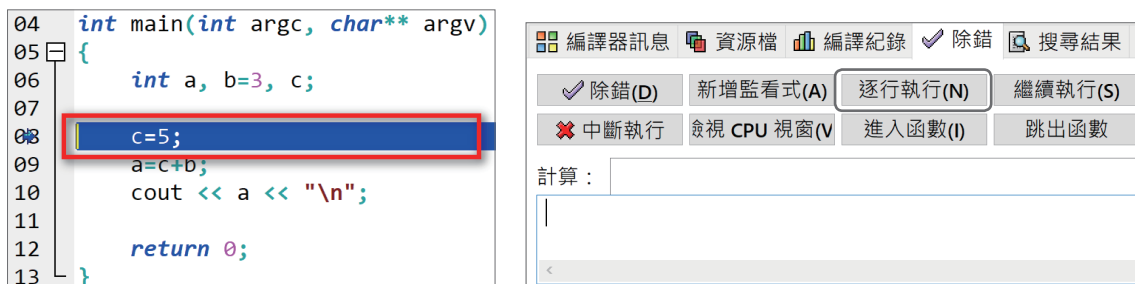
也可以手動進行設定。點選主功能表的 [工具]>[編譯器選項]，開啓「編譯器選項」視窗後切換到「編譯設定」頁面，再切換「連結器」頁面，設定以下 2 個項目，最後再按「確定」按鈕完成設定；如下圖所示。

項目	設定
產生除錯資訊 (-g3)	Yes
移除所有符號表以及重定資訊 (-s)	No

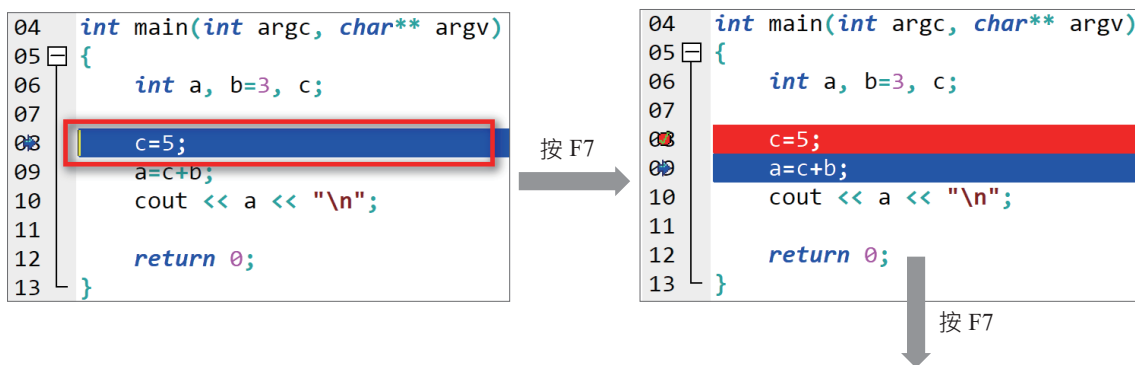


逐步執行

設定好中斷點以及除錯設定之後，便可以按  開始除錯。程式會開始執行一直遇到中斷點則停止執行。中斷點的顏色會變成藍色，如下圖左所示。此時可以按除錯視窗的 [逐步執行]（或是按 F7）以手動的方式逐步執行程式碼，藉以觀察程式碼執行的順序是否與原先的設定一樣。



每當按一次 F7 進行逐步執行，程式只會執行一行（藍色表示正要執行，但還沒執行的程式碼）。因此要執行到第 10 行程式碼，則需要按 3 次的 F7；如下圖所示。



```

04 int main(int argc, char** argv)
05 {
06     int a, b=3, c;
07
08     c=5;
09     a=c+b;
10     cout << a << "\n";
11
12     return 0;
13 }

```

按 F7

```

04 int main(int argc, char** argv)
05 {
06     int a, b=3, c;
07
08     c=5;
09     a=c+b;
10     cout << a << "\n";
11
12     return 0;
13 }

```

按 F7

繼續執行

在除錯的過程中，隨時可以按 [繼續執行] 按鈕以正常的方式繼續執行程式；若遇到下一個中斷點時仍然會中斷程式的執行。

如下圖所示，在程式碼第 8 與第 12 行分別設置中斷點，按 [除錯] 執行後程式停止在第 8 行，如下圖左。接著按 [繼續執行] 則程式繼續執行第 9、10 行，並在遇到第 12 行的中斷點時中斷執行並等待，如下圖右所示。

```

04 int main(int argc, char** argv)
05 {
06     int a, b=3, c;
07
08     c=5;
09     a=c+b;
10     cout << a << "\n";
11
12     return 0;
13 }

```

按 [繼續執行]

```

04 int main(int argc, char** argv)
05 {
06     int a, b=3, c;
07
08     c=5;
09     a=c+b;
10     cout << a << "\n";
11
12     return 0;
13 }

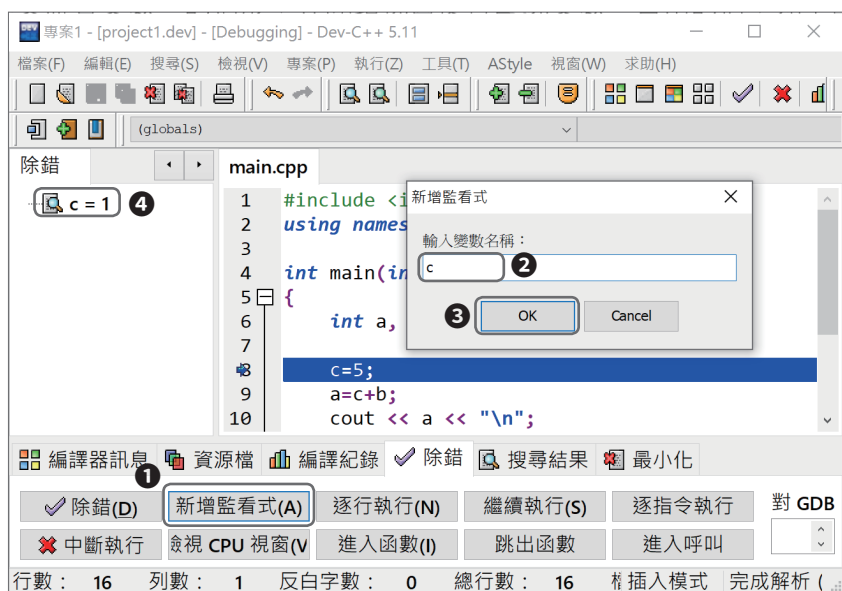
```

中斷執行

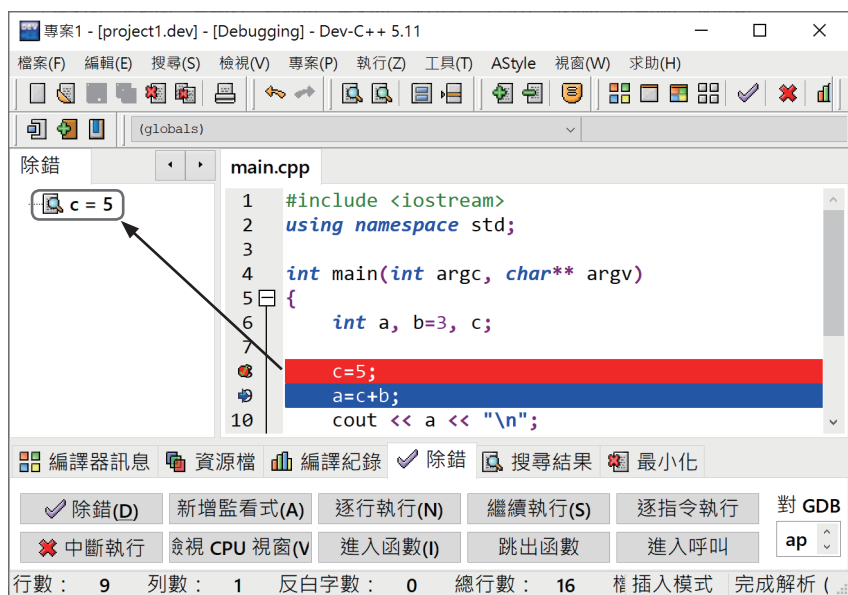
在除錯的過程中，隨時可以按 [中斷執行]（或是 F6），也可以按快捷工具列的  停止除錯。

新增監看式

在除錯的過程中，可以觀察、修改變數的值，這是一項很實用又方便的功能。請先中斷執行並移除所有中斷點後，重新在程式碼第 8 行設立中斷點，接著按 F4 開始除錯。程式遇到第 8 行的中斷點便會中斷執行，按 [新增監看式] 會彈出 [新增監看式] 對話框，請輸入變數 c，表示要監看變數 c 的情形；在除錯視窗便會出現變數 c 目前的值；如下圖所示。



這表示目前變數 **c** 的值等於 1；接著按 **F7** 讓程式執行第 8 行。程式碼第 8 行將數值 5 設定給變數 **c**，因此在除錯的視窗可看到變數 **c** 的值變成了 5；如下圖所示。



Dev-C++ 的除錯模式也能臨時修改變數的值。首先在除錯視窗上的變數 `c` 按滑鼠右鍵，彈出選單後選擇 [修改數值]，如下圖左所示。接著在「修改變數」對話框輸入新的值 8，並按 OK 按鈕，如下圖中所示。

雖然在除錯視窗上的變數 `c` 的值然仍然顯示為 5，但實際上的變數 `c` 的值已經改為 8；當把滑鼠移到程式碼第 7 行的變數 `c` 上時，便會顯示出變數 `c` 的值等於 8；如下圖右所示。當再次按 F7 逐行執行到下一行程式碼時，便可以看到除錯視窗上的變數 `c` 的值已經修改為 8 了。

