# C++

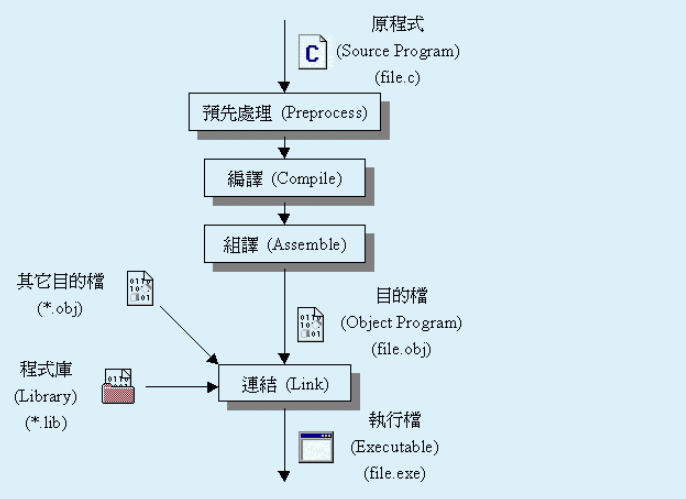
## 直譯與編譯

### 直譯

* 原始碼只透過直譯器(interpreter)，將程式碼轉為可執行碼。
* 直譯式語言(interpreted language)，不需要經過編譯，是在執行時才將原始碼逐行的轉成可執行碼。因此執行效率會比編譯過後程式碼慢。
* 直譯式語言必須依賴執行環境(execution context)，語言可用的功能只能由執行環境提供。例如JavaScript只能用瀏覽器提供的功能。

### 編譯

* 原始的程式碼經過編譯器(complier)與組譯器轉成目的碼(object code)，再由連結器(linker)，將所有目的碼與程式相關的程式庫連接起來，最後產生一個可執行檔(如.exe .lib)。
* 大型程式在編譯時，可能會耗費大量的時間。
* 編譯式語言產生的程式幾乎都是可獨立執行的，因為它們都是由Compiler進行型別／語意等檢查，以及經過連結器的處理，程式碼幾乎可以直接存取系統服務 (system service) 與 APIs。
* 編譯的過程
  + 預處理 : 做一些在編譯前要做的工作，之後就進行編譯。程式中與#相關的語句，就是預處理的範疇。



圖片來源: <http://www2.lssh.tp.edu.tw/~hlf/class-1/lang-c/compile.htm>

### 即時編譯（Just-in-time compilation）

* 為了改善編譯器與直譯器的缺點，並取得兩者個優點，因此發展出了即時編譯，此方法混合使用直譯以及編譯的特性來轉換程式碼。代表語言有C#與Java。

### 資料來源

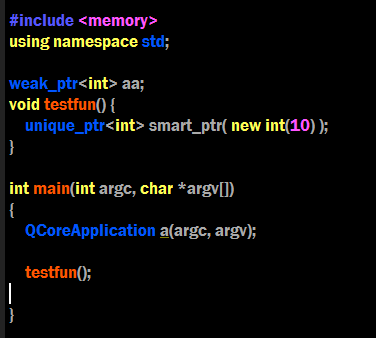
1. <http://www2.lssh.tp.edu.tw/~hlf/class-1/lang-c/compile.htm>
2. <https://dotblogs.com.tw/regionbbs/2012/05/11/compile_vs_interprete>
3. <https://medium.com/@totoroLiu/%E7%B7%A8%E8%AD%AF%E8%AA%9E%E8%A8%80-vs-%E7%9B%B4%E8%AD%AF%E8%AA%9E%E8%A8%80-5f34e6bae051>

# Smart pointer

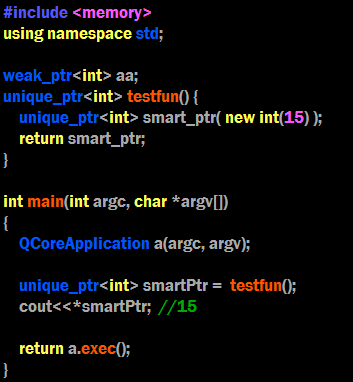
Smart pointer 為C++11新增的功能，可用來模擬傳統的pointer，它有自動delete的機制，因此可避免memory leak，使用者也可不用擔心會重複刪除或是沒有刪到new過的指標。其宣告方法與指標不同，使用方法與指標幾乎相同，C++11提供了三種Smart pointer，分別為unique\_ptr、shared\_ptr、weak\_ptr。

### unique\_ptr

* 確保一份資源（被配置出來的記憶體空間）只會被一個 unique\_ptr 物件管理的 smart pointer；當 unique\_ptr 物件消失時，就會自動釋放資源。如下圖，離開testfun()後smart\_ptr就消失了，所以其資源也會被釋放。

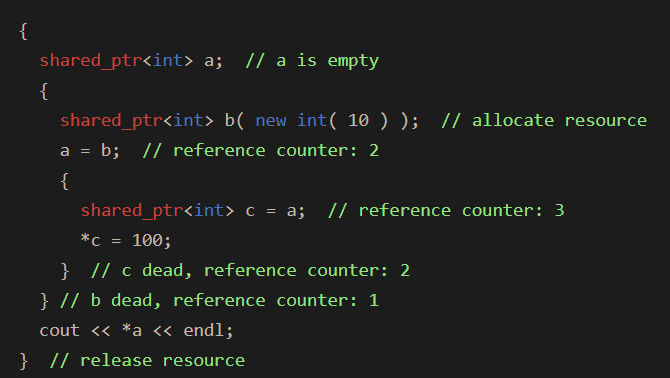


* 有move()的功能，可以轉移資源的所有權。



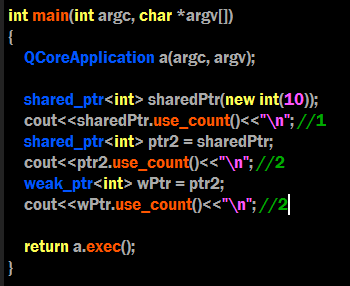
### shared\_ptr

* 多個 shared\_ptr 可以共用一份記憶體空間，並且在沒有要繼續使用的時候，可以自動把所用的資源釋放掉。

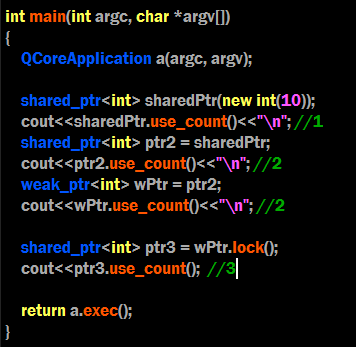


### weak\_ptr

* 可用來儲存shared\_ptr，但不會擁有資料（被配置出來的記憶體空間）的所有權，use\_count也不會增加，也沒有存取資料的功能。當 shared\_ptr 物件消失時，資源就會被釋放，weak\_ptr就無法取得資源。



* Weak\_ptr的lock()可將，weak\_ptr轉換成shared\_ptr。



### 資料來源

1. <https://kheresy.wordpress.com/2012/03/03/c11_smartpointer_p1/>
2. <https://kheresy.wordpress.com/2012/03/05/c11_smartpointer_p2/>