《C++ Primer 3/e 中文版》勘誤

最後更新日期: 2000/05/18

原著：C++ Primer 3/e, by Stanly B. Lippman & Josee Lajoie, Addison Wesley/1998

注意，英文版 1999.08.10 之前的勘誤已直接修正於中文版內。原文書的 errata 在

(1) http://people.we.mediaone.net/stanlipp/index.html
(2) http://www.awl.com/cseng/titles/0-201-82470-1/

書籍內容更正，有兩種作法，一是在網際網路上做個專屬勘誤網頁，讓大家上去看。這是比較即時的作法。而更理想更負責的作法是：不但有勘誤網頁，並且在新刷中予以更正 -- 如果有新刷的話。

不過，理想與現實之間需要一點協調。書籍的製作是這樣的，製版與印刷時，是以檯（8 或 16 頁）爲單位。因此，每換一頁，同檯的各頁統統要換過。這便造成印製成本的大量增加。

以前，我從不考慮成本，只要我認爲書籍內容有修改必要，即使只是某個字詞用得不甚理想，我都會請出版社更新。出版社也都全力配合（這一點讓我非常感謝）。

慢慢地，我的行事不再這麼霹靂，我覺得我多少也要站在出版社的立場想想。所以我打算，如果是關係到對錯正誤的根本性問題，我便一定在新刷修正。如果是易判別的錯別字或排版誤失或用詞不很恰當…等等，我便先在勘誤網頁上明載，但不求立刻於新刷中更正。直到收集來的這類誤失較爲密集了，才一併於下一刷修正。

我會在勘誤網頁（網址見書封底）上很清楚地說明，哪些是新刷已修正的，哪些是暫請讀者自行動手更改的。這是個便宜法門，請讀者見諒。

書籍應該在出版前就詳細檢查，以完美之姿出現。但是完美很難達到。對於下列大大小小輕重不等的誤失，我謹向讀者說抱歉。

本檔歡迎廣爲流傳，謝謝。

以下爲更新記錄。如果您購買的是第 n 刷，請將以下「第 n+1 刷之後的更新內容」自行修正至書上。謝謝

二刷更正內容：（注意，Lm 表示第 m 行，L-n 表示倒數第 n 行）

■導讀 p23

原文：

```
template <typename T>
T func( T p1, T p2, int p3 )      // (T p1, T p2, int p3) 稱爲 template parameter list
{  /* ... */  }                   // p1, p2 稱爲型別參數（type parameter）
                                  // p3 稱爲非型別參數（nontype parameter）
```

更正：

```
template <typename T>             // <typename T> 稱爲 template parameter list
T func( T p1, T p2, int p3 )      // 其中 T 是 template type parameter（型別參數）
{  /* ... */  }                   // 另有所謂 template nontype parameter（非型別參數），
                                  // 本例未出現。
```

■p36 L-2

原文：當它後面緊跟著一個 class 名稱

更正：當它緊跟著一個 class 名稱後面

感謝：黃向陽先生

■p106 L17（原書筆誤）

原文：`int *&ptrVal2 = pi;`

更正：`int *&refPtr = pi;`

感謝：黃向陽先生

■p342: L-4（原書筆誤）

原文：以下示範利用 `rswap()` 交換兩個指標

更正：以下示範利用 `ptrswap()` 交換兩個指標

■p496: L13

原文：inline 或 extern 修飾詞應該放在 template 參數列之前，而非…

更正：inline 或 extern 修飾詞應該放在 template 參數列之後，而非…

■p1130: copy_backward() 下的第二段文字（原書錯誤）

原文：

例如，給予數列 {0,1,2,3,4,5}，我們可以複製最後三個元素 (3,4,5) 到最前三個元素 (0,1,2) 身上，作法是將 first 設定爲元素 0 的位址，last1 設定爲元素 3 的位址，last2 設定爲元素 5 的下一個位址。於是元素 5 會被指定到原來的元素 2 身上，元素 4 會被指定到原來的元素 1 身上，元素 3 會被指定到原來的元素 0 身上。最後的結果是 {3,4,5,3,4,5}。

更正：
例如，給予數列 {0,1,2,3,4,5}，我們可以複製最前三個元素 (0,1,2) 到最後三個元素 (3,4,5) 身上，作法是將 first 設定爲元素 0 的位址，last1 設定爲元素 3 的位址，last2 設定爲元素 5 的下一個位址。於是元素 2 會被指定到原來的元素 5 身上，元素 1 會被指定到原來的元素 4 身上，元素 0 會被指定到原來的元素 3 身上。最後的結果是 {0,1,2,0,1,2}。

三刷更正內容：（注意，Lm 表示第 m 行，L-n 表示倒數第 n 行）

■p252 圖片過於粗糙，重製。

■p318 最後一段第二行（誤譯）
原文：這種作法之所以能夠成功，是因爲這些 containers 保證其元素實體會以連續的方式出現（存在）。
更正：這種作法之所以能夠成功，是因爲這些 containers 保證其鍵值相同的元素會連續出現（存在）。

■p322 最後一段第二行（修潤）
原文：我們稱 stack 爲一種所謂的 container adapter，因爲它在底層的 container 集合體身上課徵 stack 抽象性質。
更正：我們稱 stack 爲一種所謂的 container adapter，因爲它係利用底層各類型 container 加工完成 stack 抽象性質。

■p495, L4（原書筆誤）
原文：// ok: Type used many times in template parameter list
更正：// ok: Type used many times in function parameter list
感謝：kylin

■p501, L12（中譯本誤植）
原文：`min2()` 的第一個函式參數是個型別爲 `*Type` 的指標。
更正：`min2()` 的第一個函式參數是個型別爲 `Type*` 的指標。

■p503, L-5（原書筆誤）
原文：上述的 template 引數 T 從第一個函式引數中推導得 int，
　　又自第二個函式引數中推導得 unsigned int，因此 template 引數推導失敗。
更正：上述的 template 引數 T 從第一個函式引數中推導得 unsigned int，
　　又自第二個函式引數中推導得 int，因此 template 引數推導失敗。
感謝：leetron

■p565, L7（原書筆誤）
原文：Exception declaration 是函式介面的一部份，所以它必須...
更正：Exception specification 是函式介面的一部份，所以它必須...

■p576, 最上（中譯稿漏印，缺少兩個右大括號）
原文：
```
    // ...
```
更正：
```
    }
    // ...
}
```
注意：我在本書第三刷（3/p）中將 p575 和 p576 相鄰處的 layout 做了一點點挪移，所以第三刷的讀者在 p576 最上所見並非上述那樣子。請勿疑慮。

■p595, L-5
原文：當然，「令一個 non-const iterator 指向一個 const iterator」總是可以的。
更正：當然，「將一個 non-const iterator 指派給一個 const iterator」總是可以的。

■p596（原書筆誤）
說明：本頁下方有三點，討論 `back_inserter`, `front_inserter`, `inserter`.
　　並均利用 `unique_copy()` 做爲使用範例。每一個 unique_copy() 都
　　遺漏了最後的右大括弧。
更正：請爲 `unique_copy()` 加上最後的右大括弧。
感謝：Megadeth

■p652, L-1（中譯稿漏印，少了最後一行）
原文：有著以下的型別：
更正：有著以下的型別： `int (Screen::*)()`

■p731, 第一段（中譯誤失）
原文：
舉個例子，如果我再次修改 Account class 的定義，令 _name 的型別爲 string，那麼預設的 memberwise 指派動作：

```
newAcct = oldAcct;
```

就會被編譯器喚起，猶如編譯器爲我們產生了以下的 copy assignment 運算子

更正：
舉個例子，如果我再次修改 Account class 的定義，令 _name 的型別爲 string，那麼當：

```
newAcct = oldAcct;
```

預設的 memberwise 指派動作就會被編譯器喚起，猶如編譯器爲我們產生了以下的 copy assignment 運算子

■p915, 第二段文字，第一行（譯筆不佳，重譯）
原文：一個 derived class constructor 只能合法地喚起其
「直接 base class」的 constructor
更正：一個 derived class constructor 能夠合法直接喚起的 construtor 只有其
「直接 base class」的 constructor

■p915, 17.4.4 的標題（修潤）
原文：17.4.4 惰式錯誤偵測（Lazy Error Detection）
更正：17.4.4 緩式錯誤偵測（Lazy Error Detection）
注意：請同時修改
p.ix, 目錄
p.916, L6
p.981, 18.3.2 標題前兩行
p.984, L-6
p.1224, 索引

討論：我想，譯爲「緩式」可能比譯爲「惰式」更符合華人用語。在 Scott Meyers 的《More Effective C++》item17 "Consider using lazy evaluation" 中，對於 Lazy evaluation 有深刻的解說，並提出一個對應詞：eager evaluation。我把 eager evaluation 譯爲「急式評估」。

■p941, 17.5.8 標題（誤譯）
原文：虛擬函式、虛擬解構式 constructor、虛擬解構式 destructor
更正：在建構式（constructors）與解構式（destructors）中呼叫虛擬函式
注意：請同時修改目錄 (p.x)

■p1006, L-13（譯筆不佳，重譯）
原文：一旦成爲 base class，class template 必須完整列出其參數列。
修改：欲令一個 class template 扮演 base class 的角色，我們必須完整列出其參數列。

■p1045 19.2.7 標題（筆誤）
原文：19.2.7 Constructors（解構式）和 Function try Blocks 的關係
更正：19.2.7 Constructors（建構式）和 Function try Blocks 的關係
感謝：leetron
注意：請同時修改目錄 (p.x)

■p1126, adjacent_difference 內文第二行（誤譯）
原文：給予數列 {0,1,1,2,3,5,8}，新數列的 first 元素是
　　原數列的 first 元素的拷貝：0。
更正：給予數列 {0,1,1,2,3,5,8}，新數列的第一個元素是
　　原數列的第一個元素的拷貝：0。

■p1126, L-15, p1127, L9（原書錯誤）
說明：這兩行出現的 `times`，是某 function object 的舊名稱，
　　在 C++ standard 中已改名爲 `multiplies`. 見 p590
更正：將這兩行出現的 `times<int>` 改爲 `multiplies<int>`

■p1127, adjacent_find() 第二版本規格，最後一行（原書錯誤）
原文：`ForwardIterator last, Predicate pred);`
更正：`ForwardIterator last, BinaryPredicate pred);`

■p1128, binary_search() 第二版本規格，缺一行（原書遺漏）
原文：
```
bool
binary_search(ForwardIterator first,
              ForwardIterator last, const Type &value,
              Compare comp);
```
更正：請在 `bool` 前面加上一行
```
template <class ForwardIterator, class Type, class Compare>
```

■p1149, inner_product() 規格說明最後一行（原書錯誤）
原文： `(2+1) - (3+2) - (5+3) - (8+4)`
更正：`- (2+1) - (3+2) - (5+3) - (8+4)`

■p1166, L9, L20（原書錯誤）
■p1167, L1, L3 （原書錯誤）
說明：這四行出現的 `times`，是某 function object 的舊名稱，
在 C++ standard 中已改名爲 `multiplies`. 見 p590
更正：將這四行出現的 `times<int>` 改爲 `multiplies<int>`

■p1177, rotate() 規格說明第一行（原書錯誤）
原文：`rotate()` 會將 `[first,middle)` 範圍內的元素搬移到 container 尾端。
更正：`rotate()` 會將 `[first,middle)` 範圍內的元素搬移到 `last` 所指位置。

■p433, p434, p453, p454, p767, p769, p770, p771, p774, p781, p790, p849, p863, p865, p866, p881, p898, p911, p914
以上數頁製版不當（做了縮版動作）。第三刷已重新製版。

以下暫請讀者自行更正：

（注意，Lm 表示第 m 行，L-n 表示倒數第 n 行）

■範圍：全書
說明：Associative Containers 被我譯爲「聯合容器」，不甚妥當。
我想譯爲「關聯式容器」比較好，對比於「關聯式資料庫」。

■導讀 p7 L9（錯別字）
原文：以譯者的技術能力來撫平可能出現的閱讀上的坎砢崎嶇。
更正：以譯者的技術能力來撫平可能出現的閱讀上的坎坷崎嶇。
感謝：whizzkid

■導讀 p23, L-6
原文：以下造成上述 function template 產生出
函式實體 `func(float, float, int);`
更正：以下造成上述 function template 產生出
函式實體 `double func(double, double, int);`

■前言 p.xviii, L9（誤譯）
原文：最後我要說，當一個人寫了一本書，他決定略去的東西，
　　往往和他決定涵蓋的東西一樣重要。
更正：最後我要說，當一個人寫了一本書，他決定略去什麼東西，
　　往往和他決定涵蓋什麼東西一樣重要。

■p35 L13（原書筆誤）
原文：`#include <string>;`
更正：`#include <string>`
感謝：alberta

■p39 L2（原書筆誤）
原文：`{init(rhs.size, rhs.ia);}`
更正：`{init(rhs._size,rhs.ia);}`
感謝：chlin, Aua

■p39 L-3（原書筆誤）
原文：`assert( index >= 0 && index < size );`
更正：`assert( index >= 0 && index < _size );`
感謝：alberta

■p46 中間偏下（中譯本筆誤）
原文：（我們將在第 17 章...
更正：此段最後請加上小括號
感謝：黃向陽先生

■p108, L10
原文：
如果由右往左閱讀上述定義，我們會發現，pi_ref 是個 reference，代表一個指標，此指標指向一個型別爲 int 的 const object。但是我們的 reference 實際上卻未被用來代表一個常數，而是被用來代表一個非常數指標（該指標指向一個常數 object）。

更正：
如果由右往左閱讀上述定義，我們會發現，pi_ref 是個 reference，代表一個指標，此指標指向一個型別爲 int 的 const object。我們的 reference 代表的不是一個常數指標，而是一個非常數指標，指向一個常數 object。

■p125 練習 3.25（原書筆誤）
原文：`bool is_equal( const int*ia)`
更正：`bool is_equal( const int* ia)`　　請在 `ia` 之前加一空格
感謝：黃向陽先生

■p125 頁眉位置
說明：頁眉位置跑掉了

■p155 L6（原書錯誤）
原文：`while( ix_vec < 10 )`
更正：`while( ix_vec < 9 )`
感謝：李俊德先生

■p183 L3（錯別字）
原文：就某種意義而言，這說明了 C++ 語言一個自相矛頓的基礎議題。
更正：就某種意義而言，這說明了 C++ 語言一個自相矛盾的基礎議題。
感謝：whizzkid

■p183 L7（錯別字）
原文：Standard C++ 引入這些轉型運算子以強調（鮮明標示出）這個矛頓
更正：Standard C++ 引入這些轉型運算子以強調（鮮明標示出）這個矛盾
感謝：whizzkid

■p209 練習 5.7 (d) 第一行（原書筆誤）
原文：`int ival=512 jval=1024, kval=4096;`
更正：`int ival=512, jval=1024, kval=4096;` （原行少一個逗號）

■p230 L11（原書筆誤）
原文：`class ilist_item {`
更正：`class ilist {`
感謝：aven

■p242 L-4（筆誤）
原文：...，後面緊跟著一串以中括號爲界的參數。
更正：...，後面緊跟著一串以角括號爲界的參數。
感謝：rago

■p468, 第二大段程式碼的第五行（註解）出現一個中文亂碼
原文：potentially dangerous depending on i 掇 value
更正：potentially dangerous depending on i's value
感謝：edward

■p504, L-1
原文：根據各對應之「函式引數」所推導出來的「template 引數」，結果一定相同。
更正：根據各對應之「函式引數」所推導出來的「template 引數」，結果必須相同。

■p592, L16（原書筆誤）
原文：`Ires = IntNot( Ival1, Ival2 );`
更正：`Ires = IntNot( Ival1 );`
說明：`logical_not` 是一個 unary function object.
感謝：zychang（張振宇先生）

■p599, L4（中譯本筆誤）
原文：其中必須定義有一個 input 運算子（`operator<<`）
更正：其中必須定義有一個 input 運算子（`operator>>`）
感謝：zychang（張振宇先生）

■p600, L5（原書筆誤）
原文：其中必須定義有一個 output 運算子（`operator>>`）
更正：其中必須定義有一個 output 運算子（`operator<<`）
感謝：zychang（張振宇先生）

■p694, L6（原書筆誤）
原文：`Account *pacct;`
更正：`Account *pact;`

■p694, L9（原書筆誤）
原文：`pact->Acct.Account::Account(`
更正：`pact->Account::Account(`

■p709, 練習 14.8, L4（原書筆誤）
原文：`Accout acct;`
更正：`Account acct;`

■p889, 練習 17.1 之前兩行（錯別字）
原文：物件導向程式設計的主要形帽便是...
更正：物件導向程式設計的主要形貌便是...

■p1015~1030（裝訂顛倒）
少部份書品在這些頁次上裝訂顛倒。這是裝訂廠的誤失，請向經銷點
重換一冊。如經銷點不願配合，請向 service@pearsoned.com.tw 反應，
或向 http://www.gotop.com.tw 反應。

■p1183, L6（譯筆重修）
原文：傳回值 OutputIerator 指向被放進 result 所指之 container 內的
最後元素的下一位置。
更正：傳回值 OutputIerator 指向「result 所指之 container」內的
最後元素的下一位置。

■p1188, 小標題 swap_range()（原書錯誤）
原文：swap_range()
更正：swap_range**s**()
注意：該小段的函式原型、文字第一行、文字第四行各有一個 swap_range()，
皆應改爲 swap_range**s**()。同時請修改 p.vii 之目錄及 p.1235 之索引。
感謝：zychang（張振宇先生）

★英文 dimension 一詞用於陣列有兩義：(1) 維度 (2) 尺度（元素個數）我在翻譯過程中一時拘泥，譯得不好。現重新檢討如下（抱歉，頁數頗多，暫請讀者自行更正）：

■p24: L-13
原文：陣列的名稱是 fibon。這是一個整數陣列，維數爲 9。
更正：陣列的名稱是 fibon。這是一個整數陣列，尺度（元素個數）爲 9。

■p24: L-6
原文：最後一個元素，我們應該把維數減 1 做爲索引值
更正：最後一個元素，我們應該把尺度（元素個數）減 1 做爲索引值

■p28: L-9
原文：new 算式的第二個版本配置出一個特定型別和特定維數的陣列。
更正：new 算式的第二個版本配置出一個特定型別和特定尺度（元素個數）的陣列。

■p34: L7
原文：我把陣列的維數指定給 array_size。
更正：我把陣列的大小指定給 array_size。

■p114: 最後一段文字
原文：
陣列的定義係由型別符號、識別名稱、維度（dimension）三者構成。維度以一個中括號表示，指出陣列之中有多少元素。陣列的維度大小必須大於或等於 1。維度值必須是一個常數算式，也就是說，它必須能夠在編譯時期便被編譯器核定（evaluate）其值。換言之一個 non-const 變數不能夠用來指定陣列的維度大小。

更正：
陣列的定義係由型別符號、識別名稱、尺度（dimension）三者構成。尺度以一個中括號表示，指出陣列之中有多少元素。陣列的尺度必須大於或等於 1。尺度必須是一個常數算式，也就是說，它必須能夠在編譯時期便被編譯器核定（evaluate）其值。換言之一個 non-const 變數不能夠用來指定陣列的尺度大小。

■p115: L2
原文：只能夠在執行時期完成，所以它不能夠用來指定陣列維度。
更正：只能夠在執行時期完成，所以它不能夠用來指定陣列尺度。

■p115: L-13
原文：面對一個明白初始化的陣列，你不需要再指定其維度，
更正：面對一個明白初始化的陣列，你不需要再指定其尺度，

■p115: L-9
原文：如果維度被明白指出，那麼串列中的元素個數就不能夠超越該值，
更正：如果尺度被明白指出，那麼串列中的元素個數就不能夠超越該值，

■p116: L1
原文：ca1 的維度值是 3 而 ca2 的維度值是 4。
更正：ca1 的尺度（元素個數）是 3 而 ca2 的尺度是 4。

■p162: L8
原文：像是陣列的維數，或是 template 的 nontype 參數。
更正：像是陣列的尺度（元素個數），或是 template 的 nontype 參數。

■p415: L13
原文：以 new 算式配置獲得的陣列，其維度可被指定爲…
更正：以 new 算式配置獲得的陣列，其尺度（元素個數）可被指定爲…

■p616: L6
原文：也應該允許使用者在執行時期設定螢幕的實際維度。
更正：也應該允許使用者在執行時期設定螢幕的實際尺寸。

■p617: L-14
原文：使用者決定讓所有的 Screen class objects 維度爲 80 x 24，
更正：使用者決定讓所有的 Screen class objects 尺寸爲 80 x 24，

■p1079: L17
原文：其中的 bufSize 便是字元陣列 buf 的維度。
更正：其中的 bufSize 便是字元陣列 buf 的尺度（元素個數）。

■p1079: L22
原文：如果 buf 的宣告並未指定維度
更正：如果 buf 的宣告並未指定尺度（元素個數）

--- the end