C++全方位學習-第四版_勘誤表

- 勘誤日期：2022/7/19

頁碼	原來內容	修正為

P3-1	● cin 物件包含於 iostream~~.h~~ 標題檔中，所以使用前須先插入 iostream~~.h~~ 檔。
	
	下面範例是讀取鍵盤輸入的數值資料，並存入整數變數 length 中。
	
	`int length;  //宣告整數變數 length`
	`cin >> length;  //將輸入資料存入 length`
		刪除.h

P3-8	`float number = 35.7`
	`cout << setprecision(4) << number;  //顯示 35.7`
	`cout << setiosflags(ios::fixed)`
	`  << setprecision(2) << number;  //顯示 35.7`
		`float number = 35.7`
		`cout << setprecision(4) << number;  //顯示 35.7`
		`cout << setiosflags(ios::fixed)`
		`  << setprecision(2) << number;  //顯示 35.70`

P3-9	**3.2.4 cout 成員函數** **成員函數（member function）**是屬於類別的觀念，本書將在第 11 章詳細介紹類別與其成員函數，在此只簡單說明如何使用 cout 的成員函數。 除了使用 setw()、setprecision()、setiosflags() 等函數來設定輸出格式化以外，還可利用 cout 的成員函數更改 cout 的預設輸出格式。這些可更改 cout 預設輸出格式的成員函數如 .width()、.precision()、.setf()、.unsetf() 函數包含於 cout 函數中，所以使用前只須插入 iostream~~.h~~ 標題檔即可，其用法與說明如下：	刪除.h
P3-12	**3.3.2 cin 成員函數** 除了使用 setw() 函數設定輸出格式化以外，還可利用 cin 的成員函數更改 cin 的預設輸入格式。這些可更改 cin 輸入格式的成員函數	**3.3.2 cin 成員函數** 除了使用 setw() 函數設定輸入格式化以外，還可利用 cin 的成員函數更改 cin 的預設輸入格式。這些可更改 cin 輸入格式的成員函數

P3-14	`char key;` `//宣告字元變數 key` `cin.get(key);` `//取得鍵盤按鍵`	`char key;` `//宣告字元變數 key` `cin.get(key);` `//取得鍵盤按鍵`
P3-20	● **指定型態**是要轉換的目的型態。雖然 C++ 非常的寬容大量，但有些人閱讀別人的程式時並不是那麼的 Nice，所以為了替閱讀程式的人著想，或者說為了自己的分數著想，使用型態指定符號讓程式更容易閱讀。	● **指定型態**是要轉換的目的型態。雖然 C++ 非常的寬容大量，但有些人閱讀別人的程式時並不是那麼的 nice，所以為了替閱讀程式的人著想，或者說為了自己的分數著想，使用型態指定符號讓程式更容易閱讀。

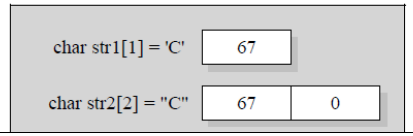
P3-22	程式 **3-16**：型態由大轉小練習
	`//儲存檔名：d:\C++03\C0316.cpp`
	`#include <iostream>`
	`using namespace std;`
	
	`int main(int argc, char** argv)`
	`{`
	`  int intVar = 500;  //intVar=500`
	`  short shortVar = short(intVar);  //shortVar=-12`
		程式 **3-16**：型態由大轉小練習
		`//儲存檔名：d:\C++03\C0316.cpp`
		`#include <iostream>`
		`using namespace std;`
		
		`int main(int argc, char** argv)`
		`{`
		`  int intVar = 500;  //intVar=500`
		`  short shortVar = short(intVar);  //shortVar=500`

P3-23	`  float floatNum = 70000.0f;  //floatNum=70000.0`
	`  int intNum = int(floatNum);  //intNum=70000`
	`  short shortNum = short(floatNum);  //shortNum=32767`
		`  float floatNum = 70000.0f;  //floatNum=70000.0`
		`  int intNum = int(floatNum);  //intNum=70000`
		`  short shortNum = short(floatNum);  //shortNum=4464`

P3-26	`  << "\nd /= x => d = " << (d /= x)  // 輸出跳行、字串、d 值` `  << "\nf %= x => f = " << (f %= 6)  // 輸出跳行、字串、f 值`	`  << "\nd /= x => d = " << (d /= x)  // 輸出跳行、字串、d 值` `  << "\nf %= x => f = " << (f %= x)  // 輸出跳行、字串、f 值`
P3-31	● sin 函數是傳回指定徑度的 sin 函數值。 ● cos 函數是傳回指定徑度的 cos 函數。 ● tan 函數值是傳回指定徑度的 tan 函數值。	● sin 函數是傳回指定徑度的 sin 函數值。 ● cos 函數是傳回指定徑度的 cos 函數值。 ● tan 函數是傳回指定徑度的 tan 函數值。

P3-35	程式 **3-24**：次方與開方練習
	`//儲存檔名：d:\C++03\C0324.cpp`
	`#include <iostream>`
	`#include <iomanip>`
	`#include <cmath>  //數值函數標題檔`
	`using namespace std;`
	
	`int main(int argc, char** argv)`
	`{`
	`  float x;  //宣告整數變數`
		程式 **3-24**：次方與開方練習
		`//儲存檔名：d:\C++03\C0324.cpp`
		`#include <iostream>`
		`#include <iomanip>`
		`#include <cmath>  //數值函數標題檔`
		`using namespace std;`
		
		`int main(int argc, char** argv)`
		`{`
		`  float x;  //宣告浮點數變數`

● 勘誤日期：2022/7/11

<table>
<tr><th>頁碼</th><th>原來內容</th><th>修正為</th></tr>
<tr><td>P2-3</td><td>下面範例的第一式是 C++ 型態的新型註解，第二是則是 C 型態的舊型註解。
<pre>//儲存檔名：d:\C++02\C0201.cpp
/* 宣告整數變數練習 */</pre></td><td>第二式則是 C 型態的舊型註解</td></tr>
<tr><td>P2-19</td><td>表 2.6 整數變數的型態、功能、長度、與範圍
<table>
<tr><th>宣告型態</th><th>宣告功能</th><th>儲存空間</th><th>數值範圍</th></tr>
<tr><td>short</td><td>有號短整數</td><td>2 bytes</td><td>-32,768 至+32,767</td></tr>
<tr><td>unsigned short</td><td>無號短整數</td><td>2 bytes</td><td>0 至 65,535</td></tr>
<tr><td>int</td><td>有號整數</td><td>4 bytes</td><td>-2,147,483,648 至+2147483647</td></tr>
<tr><td>unsigned int</td><td>無號整數</td><td>4 bytes</td><td>0 至 4,294,967,295</td></tr>
<tr><td>long int</td><td>有號長整數</td><td>4 bytes</td><td>-2,147,483,648 至+2147483647</td></tr>
<tr><td>unsigned long int</td><td>無號長整數</td><td>4 bytes</td><td>0 至 4,294,967,295</td></tr>
<tr><td>long long in</td><td>有號倍長整數</td><td>8 bytes</td><td>- 9,223,372,036,854,775,808 至 9,223,372,036,854,775,807</td></tr>
</table></td><td>long long int</td></tr>
<tr><td>P2-21</td><td>事實上，C++ 會先將字元資料或字串資料轉成 ASCII 碼再存入記憶體中，所以字元 'C' 會先轉成 A 的 ASCII 碼 67 再存入記憶體中，而 '\0' 會先轉成 ASCII 碼 0 再存入記憶體中，如下圖則是實際存在記憶體中的資料。
 
</td><td>所以字元 'C' 會先轉成 C 的</td></tr>
<tr><td>P2-22</td><td><pre>char letter2 = 67; //以'C'的 ASCII 值定義字元
char letter3 = 0x43; //以'C'的 16 進位 ASCII 值
char tab = '\t'; //宣告字元並令 tab=定位符號
char string[] = "ANSI C++"; //字串 str="ANSI C++"</pre>註：0x43 表示十六進位數，其對應的十進位數為 4*161+3*160=67。</td><td>其對應的十進位數為 $4*16^1+3*16^0=67$</td></tr>
</table>

- 勘誤日期：2021/9/16

頁碼	原來內容	修正為
P12-4	24. int area(){ //計算面積或表面積函數 25. If (height == 0) 26. return length * width;	大寫 If 要改小寫 if

- 勘誤日期：2021/8/30

頁碼	原來內容	修正為

P7-20	55. cout << ((y == 0) ? "| " : " | ");
	56. for(x=0; x<3; x++) //內層迴圈,改變索引值 x
	57. cout << setw(2) << dd[y][x] << ' '; //輸出陣列元素
	58. cout << "|\n";
		55. cout << ((y == 0) ? "| " : " | ");
		56. for(x=0; x<3; x++) //內層迴圈,改變索引值 x
		57. cout << setw(2) << dd[y][x] << ' '; //輸出陣列元素
		58. cout << "|\n";

P7-28	a -= array[0][i]*array[1][(i+2)%3]*array[2][(i+1)%3];
	}
	return a; //傳回計算值
	}
	第一行程式碼文字的下半截不見了
		a -= array[0][i]*array[1][(i+2)%3]*array[2][(i+1)%3];
		}
		return a; //傳回計算值
		}

P7-28

程式 **7-17**：解三元一次聯立方程組

```
1.  //檔案名稱：d:\C++07\C0717.cpp
2.  #include <iostream>
3.  #include <iomanip>
4.  using namespace std;
5.
6.  void showArray(int [][3], int, char []); //宣告函數原型
7.  int calArray(int [][3]);                 //宣告函數原型
8.
9.  int main(int argc, char** argv)
10. {
11.    int eq[3][4] = { {7, 1, 3, 6},
12.                     {4, -1, 1, 1},
13.                     {5, 3, -2, 8} };     //宣告並起始二維陣列
14.    int dd[3][3] = {0};                   //宣告並起始二維陣列
15.    int d, dx, dy, dz, y;                 //宣告整數變數
16.    char tt[]="d  = | ";                  //宣告字串變數
17.                                          //輸出方程組
18.    cout << "三元一次聯立方程組\n";
```

程式 **7-17**：解三元一次聯立方程組

```
1.  //檔案名稱：d:\C++07\C0717.cpp
2.  #include <iostream>
3.  #include <iomanip>
4.  using namespace std;
5.
6.  void showArray(int [][3], int, char []); //宣告函數原型
7.  int calArray(int [][3]);                 //宣告函數原型
8.
9.  int main(int argc, char** argv)
10. {
11.    int eq[3][4] = { {7, 1, 3, 6},
12.                     {4, -1, 1, 1},
13.                     {5, 3, -2, 8} };     //宣告並起始二維陣列
14.    int dd[3][3] = {0};                   //宣告並起始二維陣列
15.    int d, dx, dy, dz, y;                 //宣告整數變數
16.
17.    //輸出方程組
18.    cout << "三元一次聯立方程組\n";
```

P7-29

```
34.    showArray(dd, d, tt);                 //輸出行列式與值
35.                                          //計算與輸出 dx 值
36.    for(y=0; y<3; y++)                    //複製陣列元素
37.    {
38.       dd[y][0] = eq[y][3];               //複製 eq 第 3 到 dd 第 0 列
39.       dd[y][1] = eq[y][1];               //複製 eq 第 1 到 dd 第 1 列
40.       dd[y][2] = eq[y][2];               //複製 eq 第 2 到 dd 第 2 列
41.    }
42.    dx = calArray(dd);                    //計算行列式值
43.    showArray(dd, dx, tt);                //輸出行列式與值
44.                                          //計算與輸出 dy 值
45.    for(y=0; y<3; y++)                    //複製陣列元素
46.    {
47.       dd[y][0] = eq[y][0];               //複製 eq 第 0 到 dd 第 0 列
48.       dd[y][1] = eq[y][3];               //複製 eq 第 3 到 dd 第 1 列
49.       dd[y][2] = eq[y][2];               //複製 eq 第 2 到 dd 第 2 列
50.    }
51.    dy = calArray(dd);                    //計算行列式值
52.    showArray(dd, dy, tt);                //輸出行列式與值
53.                                          //計算與輸出 dz 值
54.    for(y=0; y<3; y++)                    //複製陣列元素
55.    {
56.       dd[y][0] = eq[y][0];               //複製 eq 第 0 到 dd 第 0 列
57.       dd[y][1] = eq[y][1];               //複製 eq 第 1 到 dd 第 1 列
58.       dd[y][2] = eq[y][3];               //複製 eq 第 3 到 dd 第 2 列
59.    }
60.    dz = calArray(dd);                    //計算行列式值
61.    showArray(dd, dz, tt);                //輸出行列式與值
```

```
34.    showArray(dd, d, "d  = | ");          //輸出行列式與值
35.                                          //計算與輸出 dx 值
36.    for(y=0; y<3; y++)                    //複製陣列元素
37.    {
38.       dd[y][0] = eq[y][3];               //複製 eq 第 3 到 dd 第 0 列
39.       dd[y][1] = eq[y][1];               //複製 eq 第 1 到 dd 第 1 列
40.       dd[y][2] = eq[y][2];               //複製 eq 第 2 到 dd 第 2 列
41.    }
42.    dx = calArray(dd);                    //計算行列式值
43.    showArray(dd, dx, "dx = | ");         //輸出行列式與值
44.                                          //計算與輸出 dy 值
45.    for(y=0; y<3; y++)                    //複製陣列元素
46.    {
47.       dd[y][0] = eq[y][0];               //複製 eq 第 0 到 dd 第 0 列
48.       dd[y][1] = eq[y][3];               //複製 eq 第 3 到 dd 第 1 列
49.       dd[y][2] = eq[y][2];               //複製 eq 第 2 到 dd 第 2 列
50.    }
51.    dy = calArray(dd);                    //計算行列式值
52.    showArray(dd, dy, "dy = | ");         //輸出行列式與值
53.                                          //計算與輸出 dz 值
54.    for(y=0; y<3; y++)                    //複製陣列元素
55.    {
56.       dd[y][0] = eq[y][0];               //複製 eq 第 0 到 dd 第 0 列
57.       dd[y][1] = eq[y][1];               //複製 eq 第 1 到 dd 第 1 列
58.       dd[y][2] = eq[y][3];               //複製 eq 第 3 到 dd 第 2 列
59.    }
60.    dz = calArray(dd);                    //計算行列式值
61.    showArray(dd, dz, "dz = | ");         //輸出行列式與值
```

P7-30	程式輸出 三元一次聯立方程組 7x +1y +3z= 6 4x -1y +1z= 1 5x +3y -2z= 8 d = | 7 1 3 | = 57 | 4 -1 1 | | 5 3 -2 |	程式輸出 三元一次聯立方程組 7x +1y +3z= 6 4x -1y +1z= 1 5x +3y -2z= 8 d = | 7 1 3 | = 57 | 4 -1 1 | | 5 3 -2 |