

標準樣版函式庫簡介 (Standard Template Library)

上課老師：莊啟宏

標準樣版函式庫簡介簡介(1/2)

- 標準樣版函式庫(Standard Template Library)簡稱STL是C++最重要的新增功能，它提供C++程式設計師常用的資料結構與演算法。
- 例如：Stack、Queue、Linked List…等資料結構，或是Swap、Sort、Find…等泛用演算法。標準樣版函式庫的發展已經非常成熟，程式的執行速度與記憶體使用空間都已經達到最佳化。

標準樣版函式庫簡介簡介(2/2)

- 標準樣版函式庫是屬於泛型函式庫，它是使用樣版(template)來實作的，也就是說標準樣版函式庫可以處理不同資料型別的資料。
- 使用標準樣版函式庫之前，建議您必須先具備樣版的觀念。
- 標準樣版函式庫包含三個部份，分別是容器(Container)、指位器(Iterator)與演算法(Generic Algorithm)，這裡將會為您介紹STL常用容器、指位器與演算法。

容器簡介(1/)

- 容器(Container)是指可用來儲存物件的物件，在STL中每一種容器都有不同儲存物件的方式。指位器可以用來存取容器內的資料，不同的容器必須配合相對應的指位器，下列是STL常用的容器；
- vector、list、deque、set、multiset、map、multimap

容器簡介(1/6)

- STL的容器是使用樣版(template)來實作，因此容器內可以存放不同資料型別的物件或資料，您可以將容器想像成傳統C語言的陣列一樣，但容器的使用又比C語言的陣列更方便更有效率。
- 例如：容器可以動態增加大小，不需要自行配置記憶體，可以使用STL提供的泛型函式來對容器做排序、反轉、搜尋等動作。

容器簡介(2/6)

- 例如下面示範如何宣告container為vector容器物件用來存放整數，container的範圍是container[0]~container[1]，如果想要再增加容器的大小，可使用push_back方法，在容器的最後面插入一個元素，此時container的範圍變成container[0]~container[2]。

```
vector<int> container(2); //容器範圍 container[0]~container[1]，可存放整數
container[0]=7;           //指定 container[0]等於 7
container[1]=2;           //指定 container[1]等於 2
container.push_back(4);   //在容器的最後面插入一個元素，其資料為 4
                           //即指定 container[2]等於 4
```


容器簡介(4/6)

- 如果容器內的元素想要進行由小到大排序，則可以使用泛用演算法的`sort()`樣版函式，`sort()`函式會指定容器中某個範圍內的元素進行由小到大排序，`begin()`方法會傳回第一個元素的指標，`end()`方法會傳回最後一個元素的指標。

容器簡介(5/6)

- 下面敘述 `sort(container.begin(), container.end());` 即是將container內的所有元素進行由小到大排序。經過`sort()`函式排序後，結果`contain[0]=2`，`contain[1]=4`，`contain[2]=7`。

```
vector<int> container(2); //容器範圍 container[0]~container[1]，可存放整數
container[0]=7;           //指定 container[0]等於 7
container[1]=2;           //指定 container[1]等於 2
container.push_back(4);   //在容器的最後面插入一個元素，其資料為 4
                           //即指定 container[2]等於 4
```

```
//begin()方法會傳回第一個元素的指標、end()方法會傳回最後一個元素的指標
//sort 函式可以對容器中某個範圍內的元素進行由小到大排序
sort(container.begin(), container.end());
```

容器簡介(6/6)

- 如果容器內的元素想要進行由小到大排序，則可以使用泛用演算法的sort()樣版函式，sort()函式會指定容器中某個範圍內的元素進行由小到大排序，begin()方法會傳回第一個元素的指標，end()方法會傳回最後一個元素的指標。

序列容器(1/2)

- 序列容器(Sequence Container)中的資料是以線性的方式來儲存，就好像傳統C語言的陣列，或是資料結構中的堆疊、佇列、鏈結串列一樣皆是以線性的方式來儲存，因此序列容器內的資料前後順序皆已經被確定，常用的序列容器如下說明：

序列容器(2/2)

容器名稱	功能說明
vector	屬於動態陣列。它在記憶體內是以連續空間來儲存的，且容器大小可以動態增加。資料插入到容器的最後面時速度較快，資料插入到容器前端(最前面)或中間時速度較慢。vector 可以使用 [] 中括號來存取資料，如 v1[0], v[1]...等。
deque	與 vector 類似。它在記憶體內是以不連續空間來儲存的，且容器大小可以動態增加。資料插入到容器的最前面與最後面(前後端)所花費時間較少，資料插入到容器的中間所花費的時間多。deque 可以使用 [] 中括號來存取資料，如 v1[0], v[1]...等。
list	屬於雙向鏈結串列。容器內的每一個元素都有指標會指向前一個元素與後一個元素，資料新增在容器的任何一個位置的速度都很快。list 無法使用 [] 中括號來存取資料。

關聯容器(1/2)

- 關聯容器(Associate Container)不是以線性的方式來儲存，容器內元素的順序並不是以資料元素的插入順序來排列，而是以元素的鍵值(key)來決定容器內元素的排列順序，因此當資料插入到關聯容器內的速度很快，常用的關聯容器說明如下：

關聯容器(2/2)

容器名稱	功能說明
set	容器中的資料只能儲存鍵值(Key)，不儲存對應的值，同一個容器中的資料，無法儲存相同的鍵值。
multiset	容器中的資料只能儲存鍵值(Key)，不儲存對應的值，同一個容器中的資料，可以儲存相同的鍵值。
map	容器中的資料可以儲存鍵值(Key)與對應的值，同一個容器中的資料，無法儲存相同的鍵值。
multimap	容器中的資料可以儲存鍵值(Key)與對應的值，同一個容器中的資料，可以儲存相同的鍵值。

指位器

指位器簡介(1/2)

- 指位器(iterator)可以指向容器內元素的位址，您可以將指位器想成是一種特殊指標，指位器可以用來存取容器內的資料元素，不同的容器必須配合適當的指位器，才能讓程式的執行更有效率，在STL中每一種容器皆定義不同功能的指位器，一般常用的指位器有下列五種，其說明如下：

指位器簡介(2/2)

指位器種類	功能說明
輸入(Input)	用來處理資料的輸入。指位器先往下移動一個元素的位置，再將新資料加入到目前指向的位置。
輸出(Output)	用來處理資料的輸出。先輸出指位器目前指向位置的資料，接著指位器再往下移動一個元素的位置。
向前(Forward)	同時具有輸入和輸出功能的指位器。
雙向(Bidirectional)	具有向前指位器的功能。但指位器的存取方向可向前移動一個元素的位置。
隨機存取(Random Access)	具有雙向指位器的功能。可以透過指位器直接存取容器內某個元素。

如何使用指位器(1/3)

- 如果要使用容器中的指位器，首先必須先透過下面語法來宣告才能使用指位器。
- 容器<資料型別>::指位器種類 指位器名稱;

例 1：vector<int>::iterator ptr1; //宣告可指向存放整數 vector 容器的 ptr1 指位器

例 2：vector<string>::iterator ptr2 //宣告可指向存放字串 vector 容器的 ptr2 指位器

例 3：list<float>::iterator ptr3; //宣告可指向存放浮點數 list 容器的 ptr3 指位器

如何使用指位器(2/3)

- for迴圈配合指位器將容器內的資料元素讀取出來。
- 如下簡例，宣告container為vector容器物件用來存放整數，容器範圍為container[0]~container[2]。
- 接著再宣告可以指向存放整數vector容器的ptr指位器，在for迴圈首先透過ptr=container.begin() 讓ptr指向容器的第一個元素，透過ptr!=container.end() 判斷ptr是否尚未指向容器的最後一個元素，如果ptr未指到最後一個元素則進入迴圈並透過*ptr取得目前指位器所指到的元素。
- 接著利用ptr++將指位器往下移一個位置，一直到ptr指到最後一個元素才離開for迴圈。

如何使用指位器(3/3)

```
vector<int> container(3); //容器範圍 container[0]~container[2]，可存放整數
container[0]=7;           //指定 container[0]等於 7
container[1]=2;           //指定 container[1]等於 2
container[2]=4;           //指定 container[2]等於 4
```

```
vector<int>::iterator ptr; //宣告可以指向存放整數 vector 容器的 ptr 指位器
//當 ptr 指位器尚未指到最後一個元素之後，則執行 for 迴圈內的敘述
for(ptr=container.begin() ; ptr!=container.end() ; ptr++)
{
    cout << *ptr << endl;    // *ptr 會取得目前指位器指到的元素
                               // 並將該元素顯示在螢幕上
}
```

STL常用容器： vector

- vector是屬於**序列容器**，它和傳統C語言的陣列很類似，兩者皆存放在連續的記憶體空間，差別在於傳統的陣列一經宣告記憶體大小即被固定，但vector宣告不需明確指定陣列大小，vector可以動態配置記憶體給容器使用，加入的資料會從容器的最後面(尾端)增加。
- 欲使用vector必須在程式的最開頭含入vector標頭檔。
- **#include <vector>**

Vector

vector 的宣告語法如下：

```
vector<資料型別> 變數名稱;
```

vector 的建構式如下。

<code>vector()</code>	//建立容器，不指定大小
<code>vector(size_type n)</code>	//建立大小為 <code>n</code> 的容器
<code>vector(size_type n, const T& t)</code>	//建立大小為 <code>n</code> 的容器，初值使用 <code>t</code> 來指定
<code>vector(const vector& x)</code>	//建立大小、初值與 <code>x</code> 相同的容器

vector常用的成員函式(方法)如下：

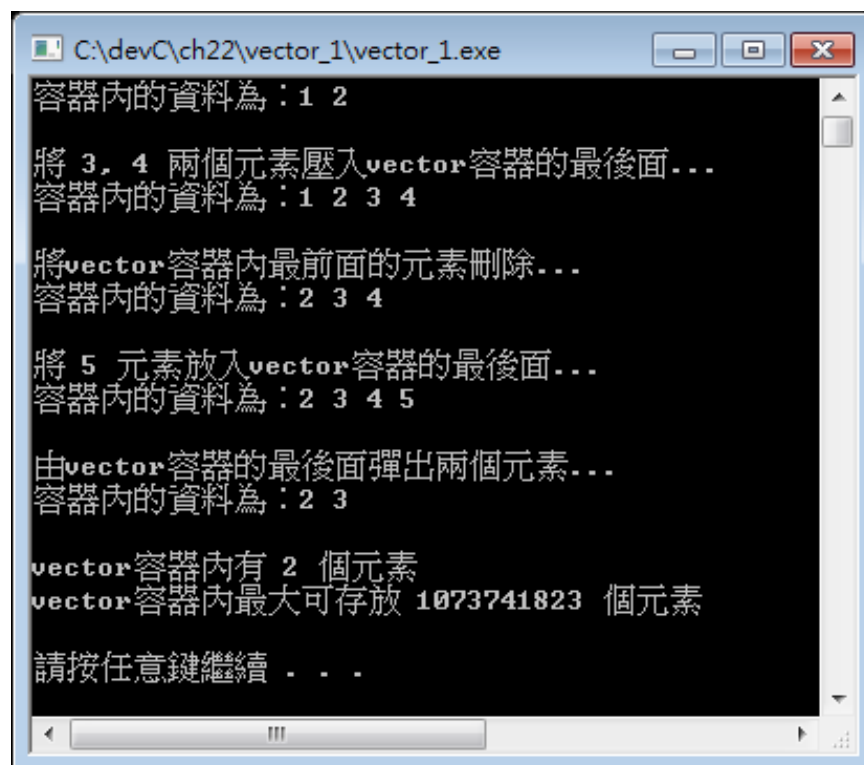
成員函式	功能說明
push_back	語法：void push_back(const T& x) 功能：在容器內最後面插入資料 x。
insert	語法：void insert(iterator pos, const T& x) 功能：在容器內第 pos 位置的前面插入資料 x。
pop_back	語法：void pop_back() 功能：刪除容器內最後面的資料。
erase	語法 1：void erase(iterator pos) 功能 1：刪除容器內第 pos 位置的資料。 語法 2：void erase(iterator first, iterator last) 功能 2：刪除容器內 first 到 last 範圍的資料。
clear	語法：void clear() 功能：刪除容器內所有的資料。
at	語法：reference at(size_type pos) 功能：傳回容器內第 pos 個元素的資料。

front	語法：reference front() 功能：傳回容器內第一個元素的資料。
back	語法：reference back() 功能：傳回容器內最後一個元素的資料。
begin	語法：iterator begin() 功能：傳回指向第一個元素位置的指位器。
end	語法：iterator end () 功能：傳回指向最後一個元素位置的指位器。
empty	語法：bool empty() 功能：判斷容器是否為空。若容器為空則傳回 true，若容器不為空則傳回 false。
size	語法：size_type size() 功能：傳回目前容器內的元素數目。
max_size	語法：size_type max_size() 功能：傳回目前容器內可存放最大的元素數目。
swap	語法：void swap(vector& x) 功能：將容器的內容與 x 容器交換。



範例：vector_1.cpp

練習使用 vector 容器，並透過 vector 的方法將容器內的資料進行插入與取出。首先在容器先置入初值 1 和 2，然後在容器尾端置入 3 和 4，刪除容器前端的資料 1，在容器尾端置入 5，刪除容器尾端兩個元素 4 和 5，最後顯示容器有多少個元素以及容器最多可存放多少個元素。試觀察容器插入與取出的情形。



```
C:\devC\ch22\vector_1\vector_1.exe
容器內的資料為：1 2
將 3, 4 兩個元素壓入vector容器的最後面...
容器內的資料為：1 2 3 4
將vector容器內最前面的元素刪除...
容器內的資料為：2 3 4
將 5 元素放入vector容器的最後面...
容器內的資料為：2 3 4 5
由vector容器的最後面彈出兩個元素...
容器內的資料為：2 3
vector容器內有 2 個元素
vector容器內最大可存放 1073741823 個元素
請按任意鍵繼續 . . .
```


程式碼 FileName : vector_1.cpp

```
01 #include <cstdlib>
02 #include <iostream>
03 #include <vector>
04 using namespace std;
05
06 template <class T>
07 void PrintOut(T& container);
08 int main(int argc, char *argv[])
09 {
10     vector<int> container(2);
11     container[0]=1;
12     container[1]=2;
13     PrintOut(container);
14
15
16     cout << "將 3, 4 兩個元素壓入 vector 容器的最後面...\n";
17     container.push_back (3);
18     container.push_back (4);
19     PrintOut(container);
```

```
21  cout << "將 vector 容器內最前面的元素刪除...\n";
22  container.erase (container.begin ());
23  PrintOut(container);
24
25  cout << "將 5 元素放入 vector 容器的最後面...\n";
26  container.insert (container.end(), 5);
27  PrintOut(container);
28
29  cout << "由 vector 容器的最後面彈出兩個元素...\n";
30  container.pop_back ();
31  container.pop_back ();
32  PrintOut(container);
33
34  cout << "vector 容器內有 " << container.size () << " 個元素\n";
35  cout << "vector 容器內最大可存放 " << container.max_size () << " 個元素\n\n";
36  system("PAUSE");
37  return EXIT_SUCCESS;
38 }
```

```
40 template <class T>
41 void PrintOut(T& container)
42 {
43     if(container.empty())
44     {
45         cout << "容器為空";
46     }
47     else
48     {
49         vector<int>::iterator ptr;
50         cout << "容器內的資料為：";
51         for(ptr=container.begin();ptr!=container.end();ptr++)
52         {
53             cout << *ptr << " ";
54         }
55         cout << "\n\n";
56     }
57 }
```

deque

- deque是屬於序列容器，和vector很類似，但deque將資料存在不連續的記憶體空間，就是將資料分散在不同記憶體位址，新增資料時可加入到容器的最前面(前端)與最後面(尾端)。
- 欲使用deque必須在程式的最開頭含入deque標頭檔：
- `#include <deque>`

deque 的宣告語法如下：

```
deque<資料型別> 變數名稱;
```

deque 的建構式如下：

deque()	//建立容器，不指定大小
deque(size_type n)	//建立大小為 n 的容器
deque(size_type n, const T& t)	//建立大小為 n 的容器，初值使用 t 來指定
deque(const vector& x)	//建立大小、初值與 x 相同的容器

deque常用的成員函式(方法)如下：

成員函式	功能說明
push_front	語法：void push_front(const T& x) 功能：在容器內最前面插入資料 x。
push_back	語法：void push_back(const T& x) 功能：在容器內最後面插入資料 x。
insert	語法：void insert(iterator pos, const T& x) 功能：在容器內第 pos 位置的前面插入資料 x。
pop_front	語法：void pop_front() 功能：刪除容器內最前面的資料。
pop_back	語法：void pop_back() 功能：刪除容器內最後面的資料。
erase	語法 1：void erase(iterator pos) 功能 1：刪除容器內第 pos 位置的資料。 語法 2：void erase(iterator first, iterator last) 功能 2：刪除容器內 first 到 last 範圍的資料。
clear	語法：void clear() 功能：刪除容器內所有的資料。
at	語法：reference at(size_type pos) 功能：傳回容器內第 pos 個元素的資料。

front	<p>語法：reference front()</p> <p>功能：傳回容器內第一個元素的資料。</p>
back	<p>語法：reference back()</p> <p>功能：傳回容器內最後一個元素的資料。</p>
begin	<p>語法：iterator begin()</p> <p>功能：傳回指向第一個元素位置的指位器。</p>
end	<p>語法：iterator end()</p> <p>功能：傳回指向最後一個元素位置的指位器。</p>
empty	<p>語法：bool empty()</p> <p>功能：判斷容器是否為空。若容器為空則傳回 true，若容器不為空則傳回 false。</p>
size	<p>語法：size_type size()</p> <p>功能：傳回目前容器內的元素數目。</p>
max_size	<p>語法：size_type max_size()</p> <p>功能：傳回目前容器內可存放最大的元素數目。</p>
swap	<p>語法：swap(deque& x)</p> <p>功能：將容器的內容與 x 容器交換。</p>



範例：deque_1.cpp

練習使用 deque 容器，並透過 deque 的方法將容器內的資料進行插入與取出。首先在容器先置入初值 1 和 2，然後在容器尾端置入 3 和 4，刪除容器前端的資料 1，在容器尾端置入 5，在容器前端插入兩個元素 6 和 7，刪除容器前端的 7 和 6，刪除容器尾端 5 和 4，最後顯示容器有多少個元素以及容器最多可存放多少個元素。試觀察容器插入與取出的情形。

```
C:\devC\ch22\deque_1\deque_1.exe
容器內的資料為：1 2
將 3, 4 兩個元素壓入deque容器的最後面...
容器內的資料為：1 2 3 4
將deque容器內最前面的元素刪除...
容器內的資料為：2 3 4
將 5 元素放入deque容器的最後面...
容器內的資料為：2 3 4 5
將 6, 7 兩個元素壓入deque容器的最前面...
容器內的資料為：7 6 2 3 4 5
由deque容器內最前面彈出兩個元素...
容器內的資料為：2 3 4 5
由deque容器內最後面彈出兩個元素...
容器內的資料為：2 3
deque容器內有 2 個元素
deque容器內最大可存放 4294967295 個元素
請按任意鍵繼續 . . .
```


程式碼 FileName : deque_1.cpp

```
01 #include <cstdlib>
02 #include <iostream>
03 #include <deque>
04 using namespace std;
05
06 template <class T>
07 void PrintOut(T& container);
08
09 int main(int argc, char *argv[])
10 {
11     deque<int> container(2);
12     container[0]=1;
13     container[1]=2;
14     PrintOut(container);
15
16     cout << "將 3, 4 兩個元素壓入 deque 容器的最後面...\n";
17     container.push_back (3);
18     container.push_back (4);
19     PrintOut(container);
20 }
```

```
21  cout << "將 deque 容器內最前面的元素刪除...\n";
22  container.erase (container.begin ());
23  PrintOut(container);
24
25  cout << "將 5 元素放入 deque 容器的最後面...\n";
26  container.insert (container.end(), 5);
27  PrintOut(container);
28
29  cout << "將 6,7 兩個元素壓入 deque 容器的最前面...\n";
30  container.push_front (6);
31  container.push_front (7);
32  PrintOut(container);
33
34  cout << "由 deque 容器內最前面彈出兩個元素...\n";
35  container.pop_front ();
36  container.pop_front ();
37  PrintOut(container);
38
39  cout << "由 deque 容器內最後面彈出兩個元素...\n";
40  container.pop_back ();
41  container.pop_back ();
42  PrintOut(container);
```

```
45  cout << "deque 容器內有 " << container.size () << " 個元素\n";
46  cout << "deque 容器內最大可存放 " << container.max_size () << " 個元素\n\n";
47  system("PAUSE");
48  return EXIT_SUCCESS;
49 }
50 template <class T>
51 void PrintOut(T& container)
52 {
53     if(container.empty())
54     {
55         cout << "容器為空";
56     }
57     else
58     {
59         deque<int>::iterator ptr;
60         cout << "容器內的資料為 :";
61         for(ptr=container.begin();ptr!=container.end();ptr++)
62         {
63             cout << *ptr << " ";
64         }
65         cout << "\n\n";
66     }
67 }
```

list

- list容器是雙向鏈結串列，使用方式和vector、deque很類似，list容器內的元素會指向前一個元素與後一個元素，因此資料插入到容器的任何一個位置都很快，但是list容器不能使用 `[]` 中括號來存取容器內元素的資料。
- 欲使用list必須在程式的最開頭含入list標頭檔。
- `#include <list>`

list 的宣告語法如下：

```
list<資料型別> 變數名稱;
```

list 的建構式如下：

list()	//建立容器，不指定大小
list(size_type n)	//建立大小為 n 的容器
list(size_type n, const T& t)	//建立大小為 n 的容器，初值使用 t 來指定
list(const vector& x)	//建立大小、初值與 x 相同的容器

list常用的成員函式(方法)：

成員函式	功能說明
push_front	語法：void push_front(const T& x) 功能：在容器內最前面插入資料 x。
push_back	語法：void push_back(const T& x) 功能：在容器內最後面插入資料 x。
insert	語法：void insert(iterator pos, const T& x) 功能：在容器內第 pos 位置的前面插入資料 x。
pop_front	語法：void pop_front() 功能：刪除容器內最前面的資料。
pop_back	語法：void pop_back() 功能：刪除容器內最後面的資料。
erase	語法 1：void erase(iterator pos) 功能 1：刪除容器內第 pos 位置的資料。 語法 2：void erase(iterator first, iterator last) 功能 2：刪除容器內 first 到 last 範圍的資料。
clear	語法：void clear() 功能：刪除容器內所有的資料。
at	語法：reference at(size_type pos) 功能：傳回容器內第 pos 個元素的資料。

front	語法：reference front() 功能：傳回容器內第一個元素的資料。
back	語法：reference back() 功能：傳回容器內最後一個元素的資料。
begin	語法：iterator begin() 功能：傳回指向第一個元素位置的指位器。
end	語法：iterator end() 功能：傳回指向最後一個元素位置的指位器。
empty	語法：bool empty() 功能：判斷容器是否為空。若容器為空則傳回 true，若容器不為空則傳回 false。
size	語法：size_type size() 功能：傳回目前容器內的元素數目。
max_size	語法：size_type max_size() 功能：傳回目前容器內可存放最大的元素數目。
sort	語法：void sort() 功能：容器內的元素進行遞增排序。
swap	語法：void swap(list& x) 功能：將容器的內容與 x 容器交換。



範例：list_1.cpp

練習使用 list 容器，並透過 list 的方法將容器內的資料進行插入、取出與排序。首先在容器先置入初值 3 和 4，然後刪除容器前端的資料 3，在容器的尾端置入 5，在容器前端插入兩個元素 6 和 7，將容器的所有元素進行遞增排序，刪除容器前端的 4 和 5，刪除容器內的所有元素，最後顯示容器有多少個元素以及容器最多可存放多少個元素。試觀察容器插入與取出的情形。

```
C:\dev\ch22\list_1\list_1.exe
將 3, 4 兩個元素壓入list容器的最後面...
容器內的資料為：3 4

將list容器內最前面的元素刪除...
容器內的資料為：4

將 5 元素放入list容器的最後面...
容器內的資料為：4 5

將 6, 7 兩個元素壓入list容器的最前面...
容器內的資料為：7 6 4 5

將list容器內的元素進行由遞增排序...
容器內的資料為：4 5 6 7

由list容器內最前面彈出兩個元素...
容器內的資料為：6 7

由list容器內的元素清空...
容器為空list容器內有 0 個元素
list容器內最大可存放 4294967295 個元素

請按任意鍵繼續 . . .
```


程式碼 FileName : list_1.cpp

```
01 #include <cstdlib>
02 #include <iostream>
03 #include <list>
04 using namespace std;
05
06 template <class T>
07 void PrintOut(T& container);
08
09 int main(int argc, char *argv[])
10 {
11     list<int> container;
12
13     cout << "將 3, 4 兩個元素壓入 list 容器的最後面...\n";
14     container.push_back (3);
15     container.push_back (4);
16     PrintOut(container);
17
18     cout << "將 list 容器內最前面的元素刪除...\n";
19     container.erase (container.begin ());
20     PrintOut(container);
```

```
22  cout << "將 5 元素放入 list 容器的最後面...\n";
23  container.insert (container.end(), 5);
24  PrintOut(container);
25
26  cout << "將 6,7 兩個元素壓入 list 容器的最前面...\n";
27  container.push_front (6);
28  container.push_front (7);
29  PrintOut(container);
30
31  cout << "將 list 容器內的元素進行由遞增排序...\n";
32  container.sort ();
33  PrintOut(container);
34
35  cout << "由 list 容器內最前面彈出兩個元素...\n";
36  container.pop_front ();
37  container.pop_front ();
38  PrintOut(container);
39
40  cout << "由 list 容器內的元素清空...\n";
41  container.clear ();
42  PrintOut(container);
```

```
44  cout << "list 容器內有 " << container.size () << " 個元素\n";
45  cout << "list 容器內最大可存放 " << container.max_size () << " 個元素\n\n";
46  system("PAUSE");
47  return EXIT_SUCCESS;
48 }
49
50 template <class T>
51 void PrintOut(T& container)
52 {
53     if(container.empty())
54     {
55         cout << "容器為空";
56     }
57     else
58     {
59         list<int>::iterator ptr;
60         cout << "容器內的資料為：";
61         for(ptr=container.begin();ptr!=container.end();ptr++)
62         {
63             cout << *ptr << " ";
64         }
65         cout << "\n\n";
66     }
67 }
```

set與multiset

- set與multiset兩者的功能類似，兩者皆可以存放鍵值(key)，容器內的資料是依鍵值來做排列的順序，差別在於set無法儲存相同的鍵值，但multiset可以儲存相同的鍵值。
- 欲使用set或multiset必須在程式的最開頭含入set標頭檔：
- #include <set>

set與multiset

- set 和 multiset 常用的宣告寫法如下：
set<資料型別, 函式物件<排序資料型別>△> 變數名稱;
multiset<資料型別, 函式物件<排序資料型別>△> 變數名稱

1. 資料型別

用來指定 set 或 multiset 所要儲存鍵值的資料型別。

2. 函式物件<排序資料型別>

函式物件用來指定 set 或 multiset 所要儲存鍵值的排序方式，而<排序資料型別>可以指定鍵值是依哪種資料型別做排序。若函式物件未指定預設為 less，less 表示使用遞增排序(由小到大)，關於常用的函式物件可參閱 22.5 節。

3. 要注意的是，上述語法倒數第二個「>」和最後一個「>」中間要加一個空白，不然會被編譯器視為「>>」運算子，例如下面寫法：

```
set<string, less<string>△>container;
```

set 的建構式如下：

set()	//建立容器，不指定大小
set(const key_compare& comp)	//建立容器，不指定大小，依 comp 的排列方式
set(const set& x)	//建立大小、初值與 x 相同的容器

multiset 的建構式如下：

multiset()	//建立容器，不指定大小
multiset(const key_compare& comp)	//建立容器，不指定大小，依 comp 的排列方式
multiset(const set& x)	//建立大小、初值與 x 相同的容器

set與multiset常用的成員函式(方法)如下表：

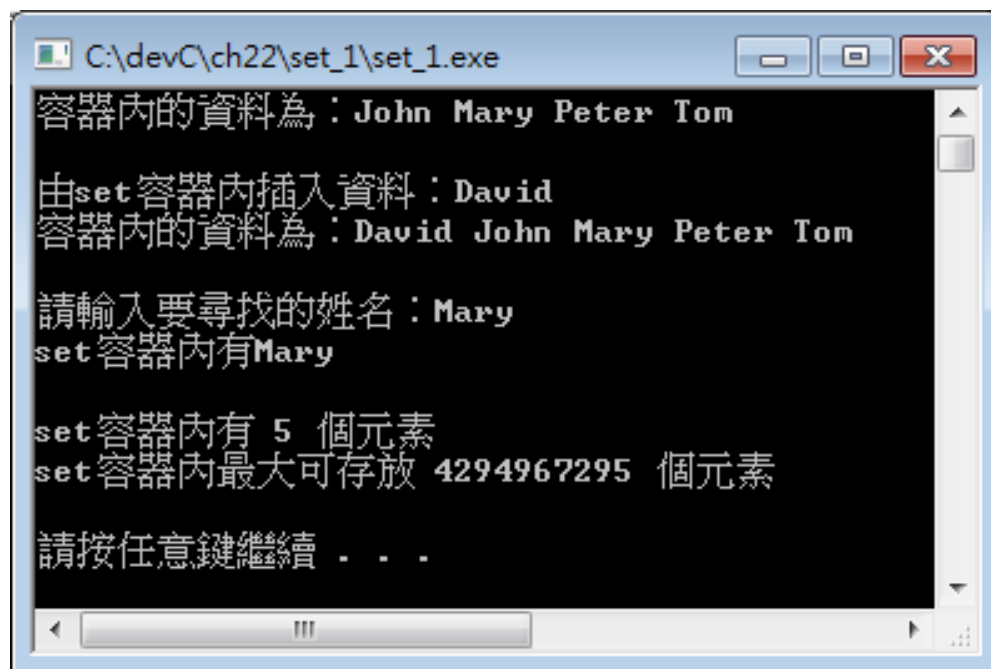
成員函式	功能說明
insert	語法：void insert(value_type x) 功能：在容器插入資料 x。
erase	語法 1：void erase(iterator pos) 功能 1：刪除容器內第 pos 位置的資料。 語法 2：void erase(const key_type& k) 功能 2：刪除容器內鍵值 k 的元素資料。
clear	語法：void clear() 功能：刪除容器內所有的資料。
begin	語法：iterator begin() 功能：傳回指向第一個元素位置的指位器。
end	語法：iterator end() 功能：傳回指向最後一個元素位置的指位器。
find	語法：iterator find(const key_type& k) 功能：傳回鍵值 k 元素位置的指位器。

upper_bound	<p>語法：iterator upper_bound(const key_type& k)</p> <p>功能：傳回第一個不小於鍵值 k 元素位置的指位器。</p>
lower_bound	<p>語法：iterator lower_bound(const key_type& k)</p> <p>功能：傳回第一個不大於鍵值 k 元素位置的指位器。</p>
empty	<p>語法：bool empty()</p> <p>功能：判斷容器是否為空。若容器為空則傳回 true，若容器不為空則傳回 false。</p>
size	<p>語法：size_type size()</p> <p>功能：傳回目前容器內的元素數目。</p>
max_size	<p>語法：size_type max_size()</p> <p>功能：傳回目前容器內可存放最大的元素數目。</p>
swap	<p>語法 1：void swap(set& x)</p> <p>語法 2：void swap(multiset& x)</p> <p>功能：將容器的內容與 x 容器交換。</p>



範例：set_1.cpp

練習使用 set 容器，並透過 set 的方法將容器內的資料進行插入與搜尋。首先在容器內先置入 John, Mary, Peter, Tom，接著讓使用者自行輸入要插入的資料，本例輸入「David」，再讓使用者輸入要搜尋的姓名「Mary」，最後容器印出搜尋的結果與容器內有多少元素，以及容器最多可存放的元素個數。由於使用 set 容器，因此無法置入相同鍵值的資料。



```
C:\devC\ch22\set_1\set_1.exe
容器內的資料為：John Mary Peter Tom
由set容器內插入資料：David
容器內的資料為：David John Mary Peter Tom
請輸入要尋找的姓名：Mary
set容器內有Mary
set容器內有 5 個元素
set容器內最大可存放 4294967295 個元素
請按任意鍵繼續 . . .
```

程式碼 FileName : set_1.cpp

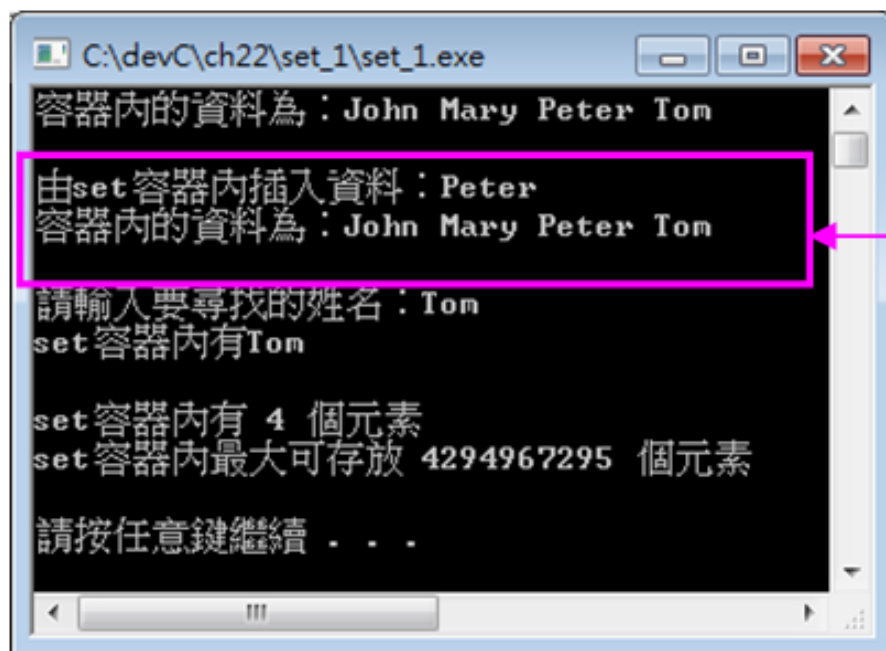
```
01 #include <cstdlib>
02 #include <iostream>
03 #include <string>
04 #include <set>
05
06 using namespace std;
07 template <class T>
08 void PrintOut(T& container);
09 int main(int argc, char *argv[])
10 {
11     string name[]={"Peter", "John", "Tom", "Mary"};
12     set<string, less<string> > container(name, name+4);
13     PrintOut(container);
14
15     string input_name;
16     cout << "由 set 容器內插入資料：";
17     cin >> input_name;
18     container.insert(input_name);
19     PrintOut(container);
```

可用選擇性的第二個樣板引數來指定排列
關鍵值的比較函數或物件

```
21  string s_name;
22  cout << "請輸入要尋找的姓名：";
23  cin >> s_name;
24  set<string>::iterator ptr;
25  ptr = container.find (s_name);
26  if (ptr==container.end()){
27      cout << "set 容器內沒有" << s_name << "\n";
28  }else{
29      cout << "set 容器內有" << s_name << "\n";
30  }
31  cout << "\n";
32  cout << "set 容器內有 " << container.size () << " 個元素\n";
33  cout << "set 容器內最大可存放 " << container.max_size () << " 個元素\n\n";
34  system("PAUSE");
35  return EXIT_SUCCESS;
36 }
```

```
38 template <class T>
39 void PrintOut(T& container)
40 {
41     if(container.empty())
42     {
43         cout << "容器為空";
44     }
45     else
46     {
47         set<string>::iterator ptr;
48         cout << "容器內的資料為：";
49         for(ptr=container.begin();ptr!=container.end();ptr++)
50         {
51             cout << *ptr << " ";
52         }
53         cout << "\n\n";
54     }
55 }
```

3. 第 17~18 行：插入資料到 container(set 容器)內。若鍵值重複，則如下圖鍵值將無法放入 container 內。



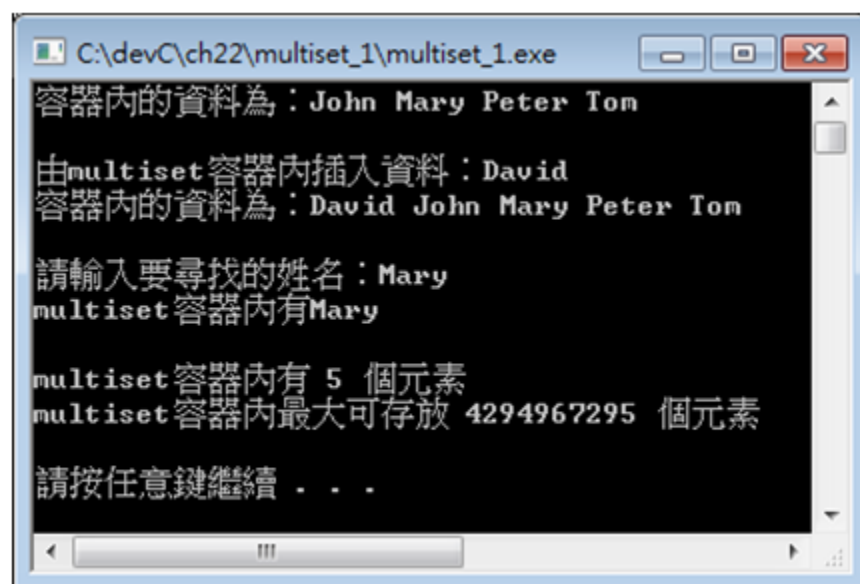
```
C:\devC\ch22\set_1\set_1.exe
容器內的資料為：John Mary Peter Tom
由set 容器內插入資料：Peter
容器內的資料為：John Mary Peter Tom
請輸入要尋找的姓名：Tom
set 容器內有Tom
set 容器內有 4 個元素
set 容器內最大可存放 4294967295 個元素
請按任意鍵繼續 . . .
```

插入重複的 Peter 鍵值，結果 Peter 無法放入 set 容器內

範例：multiset_1.cpp

練習使用 multiset 容器，並透過 multiset 的方法將容器內的資料進行插入與搜尋。首先在容器內先置入 John, Mary, Peter, Tom，接著讓使用者自行輸入要插入的資料，本例輸入「David」，再讓使用者輸入要搜尋的姓名「Mary」，最後容器印出搜尋的結果與容器內有多少元素，以及容器最多可存放的元素個數。由於使用 multiset 容器，因此可置入相同鍵值的資料。

執行結果



```
C:\devC\ch22\multiset_1\multiset_1.exe
容器內的資料為：John Mary Peter Tom
由multiset容器內插入資料：David
容器內的資料為：David John Mary Peter Tom
請輸入要尋找的姓名：Mary
multiset容器內有Mary
multiset容器內有 5 個元素
multiset容器內最大可存放 4294967295 個元素
請按任意鍵繼續 . . .
```

程式碼 FileName : multiset_1.cpp

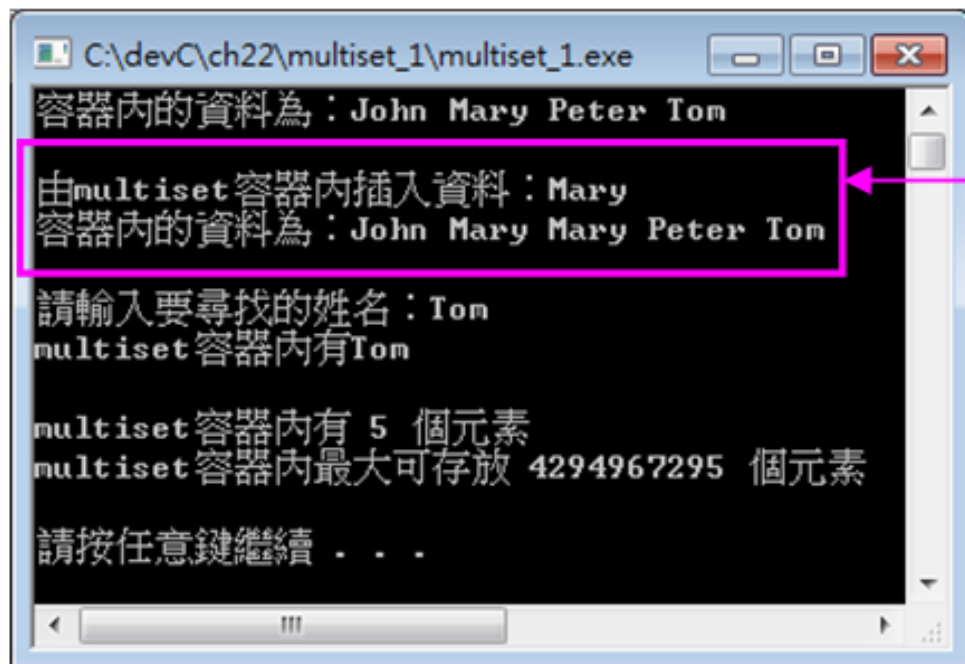
```
01 #include <cstdlib>
02 #include <iostream>
03 #include <string>
04 #include <set>
05
06 using namespace std;
07 template <class T>
08 void PrintOut(T& container);
09
10 int main(int argc, char *argv[])
11 {
12     string name[]={"Peter", "John", "Tom", "Mary"};
13     multiset<string, less<string> > container(name, name+4);
14     PrintOut(container);
15
16     string input_name;
17     cout << "由 multiset 容器內插入資料：";
18     cin >> input_name;
19     container.insert(input_name);
20     PrintOut(container);
```



```
22     string s_name;
23     cout << "請輸入要尋找的姓名：";
24     cin >> s_name;
25     multiset<string>::iterator ptr;
26     ptr = container.find (s_name);
27     if (ptr==container.end()){
28         cout << "multiset 容器內沒有" << s_name << "\n";
29     }else{
30         cout << "multiset 容器內有" << s_name << "\n";
31     }
32     cout << "\n";
33     cout << "multiset 容器內有 " << container.size () << " 個元素\n";
34     cout << "multiset 容器內最大可存放 " << container.max_size () <<
        " 個元素\n\n";
35     system("PAUSE");
36
37     return EXIT_SUCCESS;
38 }
```

```
40 template <class T>
41 void PrintOut(T& container)
42 {
43     if(container.empty())
44     {
45         cout << "容器為空";
46     }
47     else
48     {
49         multiset<string>::iterator ptr;
50         cout << "容器內的資料為：";
51         for(ptr=container.begin();ptr!=container.end();ptr++)
52         {
53             cout << *ptr << " ";
54         }
55         cout << "\n\n";
56     }
57 }
```

3. 第 18~19 行：插入資料到 container(multiset 容器)內。若鍵值重複，則如下圖鍵值可以放入 container 內。



```
C:\devC\ch22\multiset_1\multiset_1.exe
容器內的資料為：John Mary Peter Tom
由multiset容器內插入資料：Mary
容器內的資料為：John Mary Mary Peter Tom
請輸入要尋找的姓名：Tom
multiset容器內有Tom

multiset容器內有 5 個元素
multiset容器內最大可存放 4294967295 個元素
請按任意鍵繼續 . . .
```

插入重複的 Mary 鍵值，結果 Mary 可以放入 multiset 容器內

map與multimap

- `map/multimap`和`set/multiset`功能很類似。map和multimap的鍵值(key)還可以存放所對應的值。例如員工編號(鍵值)對應一筆員工基本資料，書號(鍵值)對應一本書籍資料。`map/multimap`容器內的資料是依鍵值來做排列的順序，差別在於map無法儲存相同的鍵值，但`multimap`可以儲存相同的鍵值。
- 欲使用map或multimap必須在程式的最開頭含入map標頭檔。
- `#include <map>`

map與multimap

- map和multimap常用的宣告語法如下：
- map<資料型別, 儲存資料型別, 函式物件
 <排序資料型別>△> 變數名稱;
- multimap<資料型別, 儲存資料型別, 函式物件
 <排序資料型別>△> 變數名稱;

1. 資料型別

指定 map 或 multimap 所要儲存鍵值的資料型別。

2. 儲存資料型別

指定 map 或 multimap 儲存鍵值所對應值的資料型別。

3. 函式物件<排序資料型別>

函式物件用來指定 map 或 multimap 所要儲存鍵值的排序方式，而<排序資料型別>可以指定鍵值是依哪種資料型別做排序。若函式物件未指定預設為 less，less 表示使用遞增排序(由小到大)

4. 要注意的是，上述語法倒數第二個「>」和最後一個「>」中間要加一個空白，不然會被編譯器視為「>>」運算子，例如下面寫法：

```
map<string, string, less<string>△>container;
```

map 的建構式如下：

map()	//建立容器，不指定大小
map(const key_compare& comp)	//建立容器，不指定大小，依 comp 的排列方式
map(const map& x)	//建立大小、初值與 x 相同的容器

multiset 的建構式如下：

multiset()	//建立容器，不指定大小
multiset(const key_compare& comp)	//建立容器，不指定大小，依 comp 的排列方式
multiset(const multimap& x)	//建立大小、初值與 x 相同的容器

map與multimap常用的成員函式(方法)：

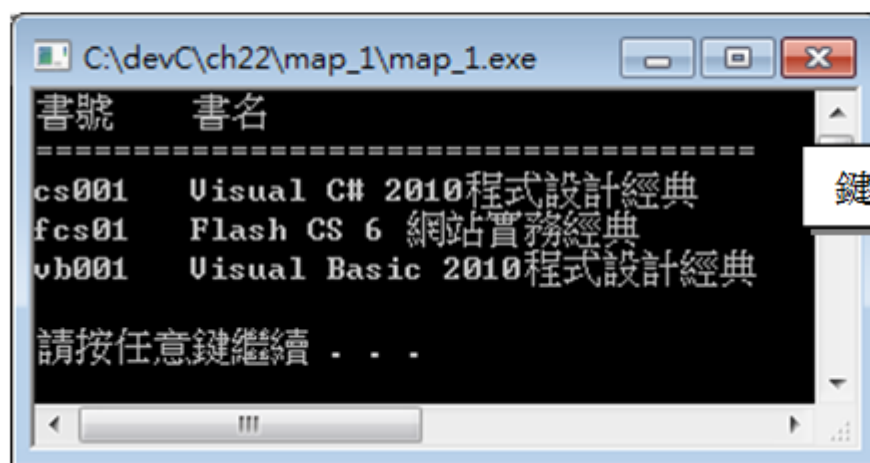
成員函式	功能說明
insert	語法：void insert(value_type x) 功能：在容器中插入 x，x 包含鍵值與對應值。
erase	語法 1：void erase(iterator pos) 功能 1：刪除容器內第 pos 位置的資料。 語法 2：void erase(const key_type& k) 功能 2：刪除容器內鍵值 k 的元素資料。
clear	語法：void clear() 功能：刪除容器內所有的資料。
begin	語法：iterator begin() 功能：傳回指向第一個元素位置的指位器。
end	語法：iterator end() 功能：傳回指向最後一個元素位置的指位器。
find	語法：iterator find(const key_type& k) 功能：傳回鍵值 k 元素位置的指位器。

upper_bound	<p>語法：iterator upper_bound(const key_type& k)</p> <p>功能：傳回第一個不小於鍵值 k 元素位置的指位器。</p>
lower_bound	<p>語法：iterator lower_bound(const key_type& k)</p> <p>功能：傳回第一個不大於鍵值 k 元素位置的指位器。</p>
empty	<p>語法：bool empty()</p> <p>功能：判斷容器是否為空。若容器為空則傳回 true，若容器不為空則傳回 false。</p>
size	<p>語法：size_type size()</p> <p>功能：傳回目前容器內的元素數目。</p>
max_size	<p>語法：size_type max_size()</p> <p>功能：傳回目前容器內可存放最大的元素數目。</p>
swap	<p>語法 1：void swap(map& x)</p> <p>語法 2：void swap(multimap& x)</p> <p>功能：將容器的內容與 x 容器交換。</p>

範例：map_1.cpp

練習使用 map 容器，並透過 map 的方法在容器放入書籍資料，使用鍵值存放書號，鍵值所對應的值存放書名，如下圖，請放入三本書籍資料。

執行結果



```
C:\devC\ch22\map_1\map_1.exe

書號    書名
=====
cs001   Visual C# 2010程式設計經典
fcs01   Flash CS 6 網站實務經典
vb001   Visual Basic 2010程式設計經典

請按任意鍵繼續 . . .
```

鍵值使用遞增排序

```

07 int main(int argc, char *argv[])
08 {
09     typedef map<string, string, less<string> > my_map ;
10
11     my_map container;
12     container.insert
13         (my_map::value_type ("vb001", "Visual Basic 2010 程式設計經典"));
14     container.insert
15         (my_map::value_type ("cs001", "Visual C# 2010 程式設計經典"));
16     container.insert
17         (my_map::value_type ("fcs01", "Flash CS 6 網站實務經典"));
18     my_map::iterator ptr;
19     cout << "書號\t書名\n";
20     cout << "=====\n";
21     for (ptr=container.begin (); ptr!=container.end(); ptr++){
22         cout << ptr->first << "\t" << ptr->second << "\n";
23     }
24     cout << "\n";
25     system("PAUSE");
26     return EXIT_SUCCESS;
27 }

```

這邊的 value_type，如果你是
std::map<string::string>，那 value_type 就會是
std::pair<string,string>

函式物件

- 函式物件是一種由樣版類別所產生的物件，函式物件的功能可以用來做大小比較、邏輯運算、算術運算等。例如關聯容器map、multimap、set、multiset中若使用less<string>，即表示容器的鍵值依字串型別由小到大進行遞增排序。
- 欲使用函式物件，必須在程式最開頭含入functional標頭檔。
- #include <functional>

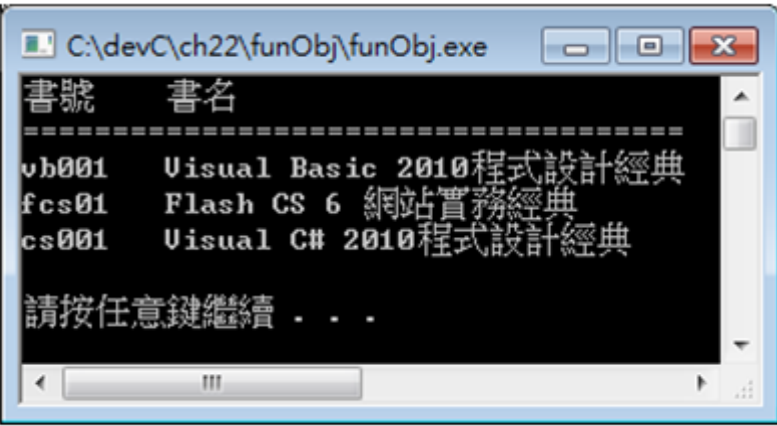
下表為常用的函式物件，其功能用來做比較大小，<T>表示欲指定的排序資料型別。

函式物件	功能說明
<code>equal_to<T></code>	接受兩個型別為 T 的引數(a, b)，若 a 和 b 兩個數相等傳回 true，否則傳回 false。
<code>not_equal_to<T></code>	接受兩個型別為 T 的引數(a, b)，若 a 和 b 兩個數不相等傳回 true，否則傳回 false。
<code>greater<T></code>	接受兩個型別為 T 的引數(a, b)，若 $a > b$ 則傳回 true，否則傳回 false。用於遞減排序。
<code>less<T></code>	接受兩個型別為 T 的引數(a, b)，若 $a < b$ 則傳回 true，否則傳回 false。用於遞增排序。
<code>greater_equal<T></code>	接受兩個型別為 T 的引數(a, b)，若 $a \geq b$ 則傳回 true，否則傳回 false。用於遞減排序。
<code>less_gual<T></code>	接受兩個型別為 T 的引數(a, b)，若 $a \leq b$ 則傳回 true，否則傳回 false。用於遞增排序。

函式物件

- 我們將前一個map_1.cpp範例，改使用greater<string>，結果如下圖發現，map容器的鍵值由大到小進行排序。修改後的範例請參閱funObj.cpp。

依鍵值做遞減排序



```
C:\devC\ch22\funObj\funObj.exe
書號  書名
=====
vb001  Visual Basic 2010程式設計經典
fcs01  Flash CS 6 網站實務經典
cs001  Visual C# 2010程式設計經典
請按任意鍵繼續 . . .
```

```
09 int main(int argc, char *argv[])
10 { //使用 greater<string>函式物件，故本例鍵值由大到小排序
11     typedef map<string, string, greater<string> > my_map ;
12     my_map container;
13     container.insert (my_map::value_type
14         ("vb001", "Visual Basic 2010 程式設計經典"));
15     container.insert (my_map::value_type
16         ("cs001", "Visual C# 2010 程式設計經典"));
17     container.insert (my_map::value_type
18         ("fcs01", "Flash CS 6 網站實務經典"));
19     my_map::iterator ptr;
20     cout << "書號\t書名\n";
21     cout << "=====\n";
22     for (ptr=container.begin (); ptr!=container.end(); ptr++){
23         cout << ptr->first << "\t" << ptr->second << "\n";
24     }
25     cout << "\n";
26     system("PAUSE");
27     return 0;
28 }
```

演算法

演算法簡介

- 演算法(Algorithm)是STL中提供給C++程式設計師使用的資料結構處理函式。
- 這些函式的建立方式，皆是使用樣版函式的技術來完成，透過STL的演算法，我們可以處理複雜的資料結構。
- 如排序、搜尋、比對、複製、合併…等機制，以達到快速操作STL容器內所儲存的資料。

演算法簡介

- 指位器是演算法操作容器內元素的媒介，不同的演算法會使用不同種類的指位器，也就是說演算法以隨機存取的方式存取容器內的元素，執行演算法的容器的指位器型別必須支援隨機存取指位器，否則該演算法無法執行。
- 例如演算法中的sort()函式可用來排序容器內的元素，使用sort()函式的容器指位器必須支援隨機存取指位器，STL的vector和deque支援隨機存取指位器，因此sort()只能排序vector和deque容器內的元素。
- 每一種容器都定義自己的指位器，下表列出各容器所支援指位器的功能。

演算法簡介

- 若要使用STL的演算法，必須在程式最開頭先含入algorithm標頭檔。其寫法如下：

#include <algorithm>

指位器種類	輸出	輸入	向前	雙向	隨機存取
vector	*	*	*	*	*
deque	*	*	*	*	*
list	*	*	*	*	
set	*	*	*	*	
multiset	*	*	*	*	
map	*	*	*	*	
multimap	*	*	*	*	

編輯演算法

- 本節介紹編輯演算法如何對容器的某個範圍進行複製、填滿、替換、反轉…等操作。

一. copy()演算法

若要將容器 A 的元素複製到容器 B 可以使用 copy()演算法，被複製容器 A 至少支援向前指位器，插入資料的容器 B 至少支援輸出指位器。其語法如下：

語法：`template<class InIt, class OutIt>OutIt copy(InIt first, InIt last, OutIt x)`
功能：將 first~last 範圍內的元素拷貝到 x 容器。

1. first：容器的起始指位器。
2. last：容器的終止指位器。
3. x：要複製的目的容器。

[例 1] 如下寫法將容器 v1 中的元素複製到容器 v2。

```
copy(v1.begin(), v1.end(), v2.begin());
```

[例 2] 如下寫法將容器 v1 中的元素複製到容器 v2 的第三個元素之後。

```
copy(v1.begin(), v1.end(), v2.begin()+3);
```

二. remove()演算法

remove()演算法可用來移除容器中的某一個元素，使用 remove()的容器的指位器至少支援向前指位器。其語法如下：

語法：`template<class FwdIt, class T>FwdIt remove(FwdIt first, FwdIt last ,
const T& val)`

功能：將 first~last 範圍內含有 val 元素移除。

1. first：容器的起始指位器。
2. last：容器的終止指位器。
3. val：要移除的元素。

[例] 如下寫法是移除容器 v1 中資料為 1 的元素。remove()的運作是將欲移除之後的元素往前移動並覆蓋欲移除的元素，使用 remove() 並不會縮短容器的長度，使用 remove 會傳回容器新尾端的指位器，因此必須再配合 erase()移除尾端的元素。其寫法如下：

```
ptr=remove(v1.begin(), v1.end(), 1);  
v1.erase(ptr, v1.end());
```

接著使用下面圖示說明上述兩行程式的執行過程：

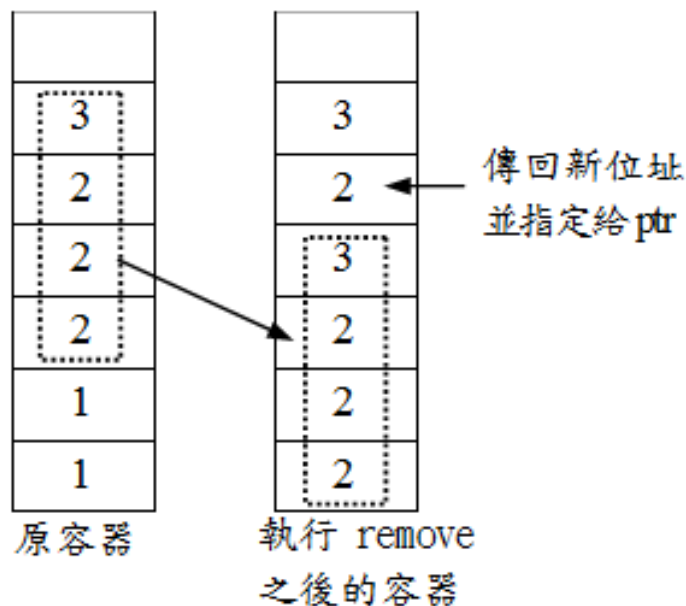
- ① 假設透過下面敘述在 v1 容器放置 6 個資料，
容器內如右圖。

```
int ary[]={1, 1, 2 ,2 ,2, 3};  
vector<int> v1(ary, ary+6);
```

3
2
2
2
1
1

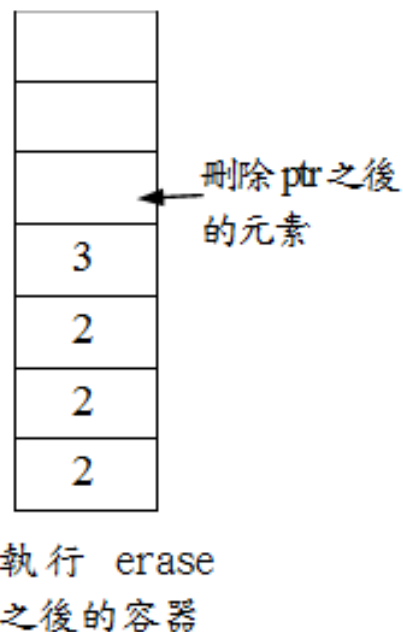
- ② 使用 `remove()` 函式將 `v1` 容器內的資料 1 移除，其方式是將欲移除之後的元素往前移動並覆蓋欲移除的元素，最後會傳回容器新的尾端的指位器，如右圖。

```
vector<int>::iterator ptr;  
ptr = remove(v1.begin(), v1.end(), 1);
```



- ③ 使用 `erase()` 函式將 `v1` 容器中 `ptr` 指位器之後的元素全部刪除。結果容器如右圖：

```
v1.erase(ptr, v1.end());
```



三. replace()演算法

replace()用來將容器某個範圍內的資料置換成新的資料，使用此演算法的容器至少支援向前指位器。其語法如下：

```
語法：template<class FwdIt, class T>OutIt replace  
        (FwdIt first, FwdIt last, const T& vold, const T& vnew);  
功能：將 first~last 範圍內含有 vold 取代為 vnew。
```

1. first：容器的起始指位器。
2. last：容器的終止指位器。
3. vold：容器內的舊資料。
4. vnew：設定取代容器內的新資料。

[例 1] 如下寫法將容器 v1 中的元素資料 2 替換為 33。

```
replace(v1.begin(), v1.end(), 2, 33);
```

[例 2] 如下寫法將容器 v1 的前三個元素資料 2 替換為 33。

```
replace(v1.begin(), v1.begin() + 3, 2, 33);
```


四. reverse()演算法

reverse()用來將容器中某個範圍內的元素進行反轉的動作，其語法如下：

語法：`template<class BidIt>void reverse(BidIt first, BidIt last)`

功能：將 first~last 範圍內的元素反轉。

1. first：容器的起始指位器。
2. last：容器的終止指位器。

五. fill()演算法

fill()可指定某個資料填滿容器中某個範圍，使用 fill()的容器的指位器至少支援向前指位器。

語法：`template<class FwdIt, class T>void fill(InIt first, InIt last, const T& x)`

功能：將 first~last 範圍內的元素改以 x 填滿。

1. first：容器的起始指位器。
2. last：容器的終止指位器。
3. x：設定要填滿的資料。

[例] 將 1 填滿 v1 容器。寫法如下：

```
fill(v1.begin(), v1.end(), 1);
```



範例：algorithm_1.cpp

下例示範使用 copy、remove、replace、fill 的演算法。

執行結果

```
C:\devC\ch22\algorithm_1\algorithm_1.exe
容器內的資料為：1 1 1 1 1 1 1 1 1 1
容器內的資料為：1 1 1 2 2 2 2 2 1 1
容器內的資料為：2 2 2 2 2
容器內的資料為：3 3 2 2 2
請按任意鍵繼續 . . .
```

```
05 using namespace std;
06 //PrintOut 用來印出容器內的元素
07 void PrintOut(vector<int>& container)
08 {
09     if(container.empty())
10     {
11         cout << "容器為空";
12     }
13     else
14     {
15         vector<int>::iterator ptr;
16         cout << "容器內的資料為：";
17         for(ptr=container.begin();ptr!=container.end();ptr++)
18         {
19             cout << *ptr << " ";
20         }
21         cout << "\n\n";
22     }
23 }
```

```
25 int main(int argc, char *argv[])
26 {
27     vector<int> container1(10);
28     //將 container1[0]~container1[9]指定為 1
29     fill(container1.begin(), container1.end(), 1);
30     PrintOut(container1);
31
32     int ary[]={2,2,2,2,2};
33     //將 container2[0]~container2[4]指定為 2
34     vector<int> container2(ary, ary+5);
35     //將 container2[0]~container2[4]拷貝到 container1[3]~container1[7]
36     copy(container2.begin(), container2.end(), container1.begin()+3);
37     PrintOut(container1);
38
39     vector<int>::iterator ptr;
40     //將 container1 容器內含 1 的元素移除
41     ptr = remove(container1.begin(), container1.end(), 1);
42     container1.erase(ptr, container1.end());
43     PrintOut(container1);
44     //將 container1[0]~container1[1]元素資料由原本的 2 改成 3
45     replace(container1.begin(), container1.begin()+2, 2, 3);
46     PrintOut(container1);
47
48     system("PAUSE");
49     return EXIT_SUCCESS;
50 }
```

搜尋演算法

一. find()演算法

透過 find()可以很方便的搜尋容器內是否有所要尋找的資料，演算法的容器指位器至少是輸入指位器，其寫法如下：

語法：template<class InIt, class T>InIt copy

(InIt first, InIt last, const T& val)

功能：將 first~last 範圍內的元素搜尋 val 資料，若有找到則傳回 val 資料的指位器，若找不到則傳回尾端的指位器。

1. first：容器的起始指位器。
2. last：容器的終止指位器。
3. val：要搜尋的資料。

[例] 如下寫法是在 v1 容器搜尋 1 整數資料。

```
ptr = find(v1.begin(), v1.end(), 1); //在 v1 中搜尋 1，並傳回 1 所在的 ptr 指位器
if(ptr==v1.end()){    //若 ptr 指到尾端表示沒有要找的資料
    cout << "容器內沒有您要找的資料\n";
}else{                //若 ptr 未指到尾端表示容器內有要尋找的資料
    cout << "容器內有 \n" << *ptr ;
}
```

二. search()演算法

search()可以找尋容器 A 中是否含有容器 B 相同片段的元素，使用 search()的 A、B 兩個容器的指位器至少支援向前指位器。其語法如下：

語法：`template<class FwdIt1, class FwdIt2>FwdIt1 search(FwdIt1 first1, FwdIt1 last1, FwdIt2 first2, FwdIt2 last2)`

功能：在 A 容器 first1~last1 範圍內是否含有 B 容器 first2~last2 的相同片段的元素。

1. first1：容器 A 的起始指位器。
2. last1：容器 A 的終止指位器。
3. first2：容器 B 的起始指位器。
4. last2：容器 B 的終止指位器。

[例] 判斷 v1 容器是否含有 v2 容器之相同片段的元素，其寫法如下：

```
ptr=search(v1.begin(), v1.end(), v2.begin(), v2.end());  
if(ptr==v1.end()){  
    cout << "容器 v1 沒有容器 v2\n";  
}else{  
    cout << "容器 v1 有容器 v2\n";  
}
```



範例：algorithm_2.cpp

下例示範使用 fill、search 的演算法。

執行結果



```
C:\devC\ch22\algorithm_2\algorithm_2.exe
容器內的資料為：1 5 6 7 4 10 9
容器內有 6
容器內有 7.4
請按任意鍵繼續 . . .
```



```
05 using namespace std;
06 //PrintOut 用來印出容器內的元素
07 void PrintOut(vector<int>& container)
08 {
09     if(container.empty())
10     {
11         cout << "容器為空";
12     }
13     else
14     {
15         vector<int>::iterator ptr;
16         cout << "容器內的資料為：";
17         for(ptr=container.begin();ptr!=container.end();ptr++)
18         {
19             cout << *ptr << " ";
20         }
21         cout << "\n\n";
22     }
23 }
```

```
25 int main(int argc, char *argv[])
26 {
27     int ary1[]={1,5,6,7,4,10,9};
28     //將 ary1 陣列放入 container1 容器
29     vector<int> container1(ary1, ary1+7);
30     PrintOut(container1);          //印出 container1 容器
31
32     vector<int>::iterator ptr;      //宣告指位器
33     //判斷 container1 容器內是否有 6
34     ptr = find(container1.begin(), container1.end(), 6);
35     if(ptr==container1.end()){
36         cout << "容器內沒有 6\n" ;
37     }else{
38         cout << "容器內有 6\n"    ;
39     }
```

```
40
41  int ary2[]={7,4};
42  //將 ary2 陣列放入 container2 容器
43  vector<int> container2(ary2, ary2+2);
44  //判斷 container1 容器內是否存在與 container2 容器相同的片段
45  ptr=search(container1.begin(), container1.end(),
46            container2.begin(), container2.end());
47  if(ptr==container1.end()){
48      cout << "容器內沒有 7,4\n";
49  }else{
50      cout << "容器內有 7,4\n";
51  }
52  cout << "\n";
53  system("PAUSE");
54  return EXIT_SUCCESS;
55 }
```

sort排序演算法

- Sort 是 STL 最常使用的演算法，其功能可用來將容器內的元素進行排序，此演算法的容器指位器至少支援隨機指位器。

語法 1： `template<class RanIt>void sort(RanIt first, RanIt last)`

語法 2： `template<class RanIt, class Pred>`

`void sort(RanIt first, RanIt last, Pred pr)`

功能：將 `first~last` 範圍內的元素進行排序。

1. `first`：容器的起始指位器。
2. `last`：容器的終止指位器。
3. `pr`：指定排序方式的函式物件，若設為 `less<資料型別>()` 表示遞增排序，若設為 `greater<資料型別>()` 表示遞減排序，關於函式物件可參閱 22.5 節。

[例 1] 將放置字串的容器 `v1` 中的元素進行遞增(由小到大)排序。

```
sort(v1.begin(), v1.end());
```

```
sort(v1.begin(), v1.end(), less<string>());
```

[例 2] 將放置字串的容器 `v1` 中的元素進行遞減(由大到小)排序。

```
sort(v1.begin(), v1.end(), greater<string>());
```



範例：algorithm_3.cpp

在 vector 容器內放入陣列元素{1, 5, 6, 7, 4, 10, 9}，接著使用 sort()函式對 vector 容器內的元素進行遞增及遞減排序。

執行結果

```
C:\devC\ch22\algorithm_3\algorithm_3.exe
未排序：
容器內的資料為：1 5 6 7 4 10 9

由小到大排序
容器內的資料為：1 4 5 6 7 9 10

由大到小排序
容器內的資料為：10 9 7 6 5 4 1

請按任意鍵繼續 . . .
```

```
06 using namespace std;
07 //PrintOut 用來印出容器內的元素
08 void PrintOut(vector<int>& container)
09 {
10     if(container.empty())
11     {
12         cout << "容器為空";
13     }
14     else
15     {
16         vector<int>::iterator ptr;
17         cout << "容器內的資料為：";
18         for(ptr=container.begin();ptr!=container.end();ptr++)
19         {
20             cout << *ptr << " ";
21         }
22         cout << "\n\n";
23     }
24 }
```

```
26 int main(int argc, char *argv[])
27 {
28     int ary1[]={1,5,6,7,4,10,9};
29     //將 ary1 陣列放入 container1 容器
30     vector<int> container1(ary1, ary1+7);
31     cout << "未排序：\n";
32     PrintOut(container1);
33     cout << "由小到大排序\n";
34     //以 less<int>整數做遞增排序
35     sort(container1.begin(), container1.end(), less<int>());
36     PrintOut(container1);
37     cout << "由大到小排序\n";
38     //以 greater<int>整數做遞減排序
39     sort(container1.begin(), container1.end(), greater<int>());
40     PrintOut(container1);
41     system("PAUSE");
42     return EXIT_SUCCESS;
43 }
```


課後練習(一)

- 使用vector容器來製作堆疊，程式有下列五個功能選項讓您可以操作堆疊內的元素。
 - ①壓入資料
 - ②彈出資料
 - ③遞增排序
 - ④遞減排序
 - ⑤印出堆疊資料

課後練習(二)

- 使用deque容器來製作佇列，程式有下列七個功能選項讓您可以操作佇列內的元素。
 - ①由前端插入資料
 - ②由後端插入資料
 - ③由前端刪除資料
 - ④由後端刪除資料
 - ⑤遞增排序
 - ⑥遞減排序
 - ⑦印出佇列資料