



C/C++程式設計

使用STL (Standard Template Library)

講師：張傑帆

CSIE, NTU

大綱

- 標準樣板函式庫(STL)
 - 抽象指標(Iterator)
 - 容器(Container)
 - 演算法(Algorithm)

標準樣板函式庫(STL)

- STL(Standard Template Library)
- 提供多種類別樣板以供使用
- 由以下三大部分所組成
 - **Iterator**(抽象指標)
 - **Container**(容器):
 - vector, list, stack, queue, map,...
 - **Algorithm**(演算法):
 - find, sort, count,...

大綱

- 標準樣板函式庫(STL)
 - 抽象指標(Iterator)
 - 容器(Container)
 - 演算法(Algorithm)

example

抽象指標(Iterator)

- Iterator是一種可以讓使用者逐一存取container中元素的工具
- 所有STL的container C都有(甚至自己寫的class)
- 使用者均可利用`C::iterator it;`的方式來宣告iterator，以取得容器內元素的位址
- 容器均有提供`begin()`,`end()`成員函數，讓使用者取得容器的第一個元素的位址與最後一個元素後一個的位址

抽象指標(Iterator)

```
#include<iostream>
#include<vector>

using namespace std;
int main()
{
    int a[7] = { 8, 1, 3, 2, 5, 1, 4};
    vector<int> v(a,a+7);
    vector<int>::iterator it ;
    for ( it = v.begin( ); it != v.end( ) ; it++)
        cout<< *it <<" ";
    cout<< endl;

    return 0;
}
```

<https://goo.gl/2icHzN>
<https://goo.gl/AnHMEg>

大綱

- 標準樣板函式庫(STL)
 - 抽象指標(Iterator)
 - 容器(Container)
 - Vector
 - List
 - Map
 - 演算法(Algorithm)

容器(Containers)

11	22	33	44				
11	22	33	44				
11	22	33	44	55	66	77	88

99?

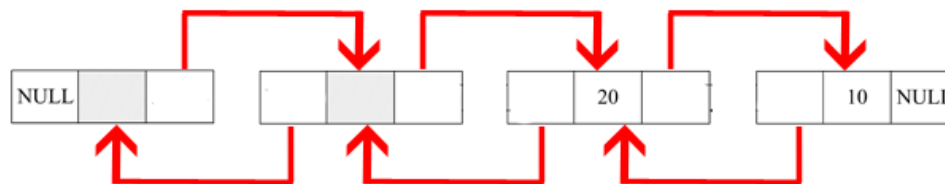
copy

11	22	33	44	55	66	77	88	99		
----	----	----	----	----	----	----	----	----	--	--

- **Vector**

- 動態陣列，此陣列可任意增長其大小，並提供隨機存取

- **List**

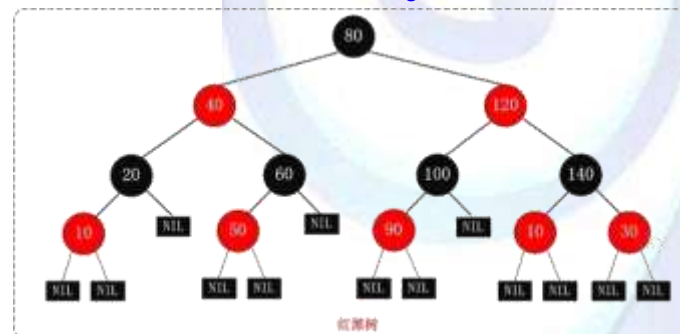


- 雙向linked list，支援循序雙向存取(無法隨機存取)

- **Map**

- 支援查表功能之資料結構，資料均以key/value方式存入

- 其他... (stack, queue, set...)



向量vector

- vector其特性如同陣列一樣，但又比內建的陣列多了許多好用的功能
- vector可以動態成長，可以將一個vector指定給另一個vector
- vector知道自己內含元素的個數

常用方法

[n]	取得第索引值n之資料
size	計算長度
front	取得開頭元素
back	取得結尾元素
pop_back	移除結尾資料
push_back	新增資料於結尾
begin	取得開頭元素之抽象指標
end	取得結尾元素之抽象指標
insert	插入資料
erase	刪除資料

vector::vector

```
#include <iostream>
#include <vector>
using namespace std;
//將現成的陣列放入vector中
int main ()
{
    int i;
    int myints[] = {16,2,77,29};
    vector<int> v1 (myints, myints+ 4);
    vector<int> v2;
    v2 = v1;
    for (i=0; i< v2.size(); i++)
        cout<< " " << v2[i];
    cout<< endl;
    return 0;
}
```



```
C:\Windows\system32\cmd.exe
16 2 77 29
請按任意鍵繼續 . . .
```

<https://goo.gl/udJkiB>
<https://goo.gl/zU6mD2>

vector::pop_back, push_back

```
#include <iostream>
#include <vector>
using namespace std;

int main ()
{
    vector<int> v;
    int sum=0;
    v.push_back(100);
    v.push_back(200);
    v.push_back(300);
    while (!v.empty()){
        sum+=v.back();
        v.pop_back();
    }
    cout<< "sum = " << sum << endl;
    return 0;
}
```



vector::begin, end

```
#include <iostream>
#include <vector>
using namespace std;
int main ()
{
    vector<int> v;
    vector<int>::iterator it;
    for (int i=1; i<=5; i++)
        v.push_back(i);
    for ( it=v.begin() ; it < v.end(); it++ )
        cout<< " " << *it;
    cout<< endl;

    return 0;
}
```



<https://goo.gl/EkjswM>

vector::insert

// inserting into a vector

```
#include <iostream>
```

```
#include <vector>
```

```
using namespace std;
```

```
int main ()
```

```
{
```

```
    int i;
```

```
    vector<int> v;
```

```
    int a[] = { 10,20,30 };
```

```
    v.insert(v.begin(), a, a+3); //插入一段資料於開頭
```

```
    v.insert(v.begin()+1,15); //插入一筆資料於第二個位子
```

```
    v.insert(v.end(),100); //插入一筆資料於最後
```

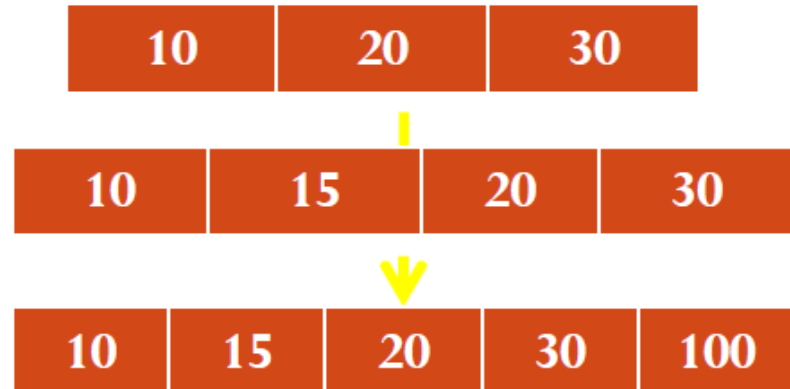
```
    for (i=0; i<v.size(); i++)
```

```
        cout<< " " << v[i];
```

```
    cout<< endl;
```

```
    return 0;
```

```
13 }
```



<https://goo.gl/h4PzfD>



begin	begin+1	begin+2	begin+3	begin+4	begin+5	begin+6	begin+7	begin+8	begin+9
1	2	3	4	5	6	7	8	9	10

vector::erase

```
#include <iostream>
#include <vector>
using namespace std;
```

```
int main ()
```

```
{
```

```
    int i;
```

```
    vector<int> v;
```

```
    for (i=1; i<=10; i++)
```

```
        v.push_back(i);
```

```
    v.erase(v.begin()+5); //指定一筆資料刪除
```

```
    v.erase(v.begin(),v.begin()+3); //刪除一段資料
```

```
    for (i=0; i<v.size(); i++)
```

```
        cout<< " " << v[i];
```

```
    cout<< endl;
```

```
    return 0;
```

```
}
```



小練習

- 請利用迴圈及vector的push_back() 和pop_back()功能
- 將「資料i ($i=1, \dots, n$)」倒序放入一vector folder 中，
例如 $n=10$ (10,9,8...1)
然後，再將資料順序取出，例如(1,2,3...10)

Sample Output

Sample Input

5

<https://jgirl.ddns.net/problem/0/3062>

```
data 5  
data 4  
data 3  
data 2  
data 1
```

```
data 1  
data 2  
data 3  
data 4  
data 5
```


串列list list其特性主要為實作串列資料結構

常用函式	
size	計算長度
front	取得開頭元素
back	取得結尾元素
begin	取得開頭元素之抽象指標
end	取得結尾元素之抽象指標
pop_back, (pop_front)	移除結尾(開頭)資料
push_back, (push_front)	新增資料於結尾(開頭)
insert	插入資料
erase	刪除資料
remove	指定一個資料內容,並刪除
reverse	資料反置
merge	合併資料

list::remove

```
#include <iostream>
#include <list>
using namespace std;

int main ()
{
    int a[] = {17, 89, 7, 14};
    list<int> L (a, a+4);
    list<int>::iterator it;
    L.remove(89);
    for (it=L.begin(); it!=L.end(); ++it)
        cout<< " " << *it;
    cout<< endl;
    return 0;
}
```



<https://goo.gl/KsnD74>

list::reverse

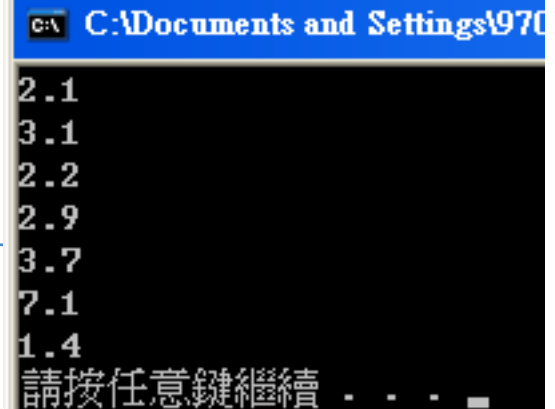
```
#include <iostream>
#include <list>
using namespace std;

int main ()
{
    list<int> L;
    list<int>::iterator it;
    for (int i = 1; i<10; i++)
        L.push_back(i);
    for (it = L.begin(); it != L.end(); ++it)
        cout << " " << *it;
    cout << endl;
    L.reverse(); //反轉串列
    for (it = L.begin(); it != L.end(); ++it)
        cout << " " << *it;
    cout << endl;
    return 0;
}
```



list::merge

```
#include <iostream>
#include <list>
using namespace std;
int main ()
{
    list<double> L1, L2;
    list<double>::iterator it;
    L1.push_back (3.1);
    L1.push_back (2.2);
    L1.push_back (2.9);
    L2.push_back (3.7);
    L2.push_back (7.1);
    L2.push_back (1.4);
    L1.merge(L2); ←請寫一段程式碼看看L2此時的內容
    L2.push_back (2.1);
    L1.merge(L2);
    for (it=L1.begin(); it!=L1.end(); ++it)
        cout<< *it << endl;
    return 0;
}
```



```
C:\Documents and Settings\970
2.1
3.1
2.2
2.9
3.7
7.1
1.4
請按任意鍵繼續 . . .
```

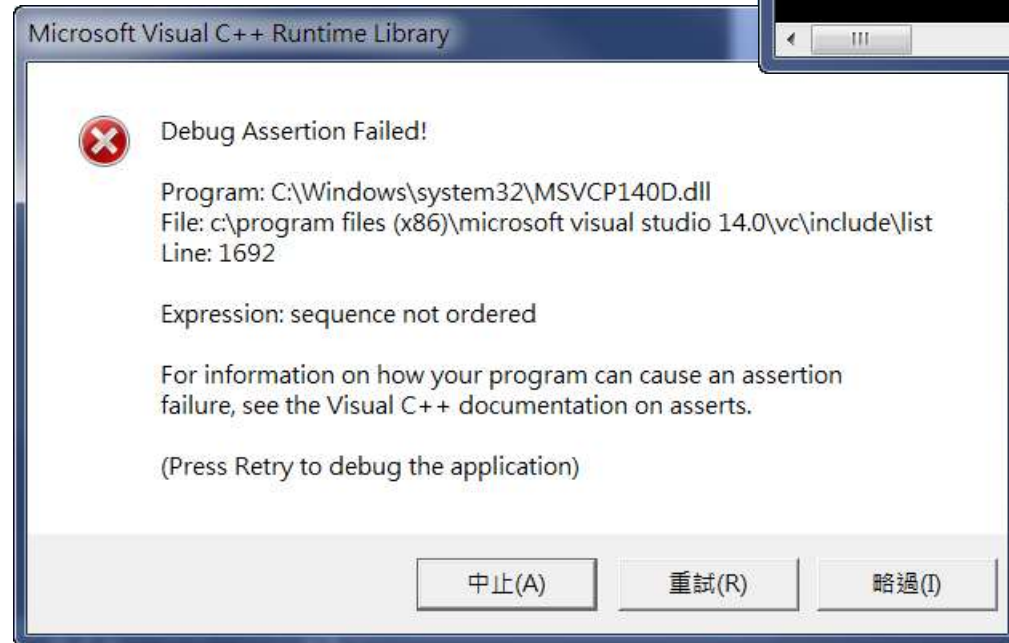
VS 沒排序 會出錯

```
L1.push_back(3.1);  
L1.push_back(2.2);  
L1.push_back(2.9);  
L2.pus_back(3.7);  
L2.push_back(7.1);  
L2.push_back(1.4);
```

```
L1.merge(L2);
```

```
L2.push_back(2.1);  
L1.merge(L2);
```

```
for (it = L1.begin(); it != L1.end(); ++it)  
    cout << *it << endl;
```



小練習

- 3064. List 手動/自動 排序

```
string books[] = {"Book1", "Book2", "Book6", "Book4", "Book5"};  
list<string> st(books, books+5);
```

- 將自己面前書櫃裡及掉到地上的書(Book3)按照次序排列整齊。
- 方法一 手動排序
 - 將shelf 中第 3 個元素加到並新增到串列之後
 - 將shelf 中第 3 個元素更改為Book3
- 方法二 自動排序
 - 加入Book3之後使用 list.sort() 排序

映射表map

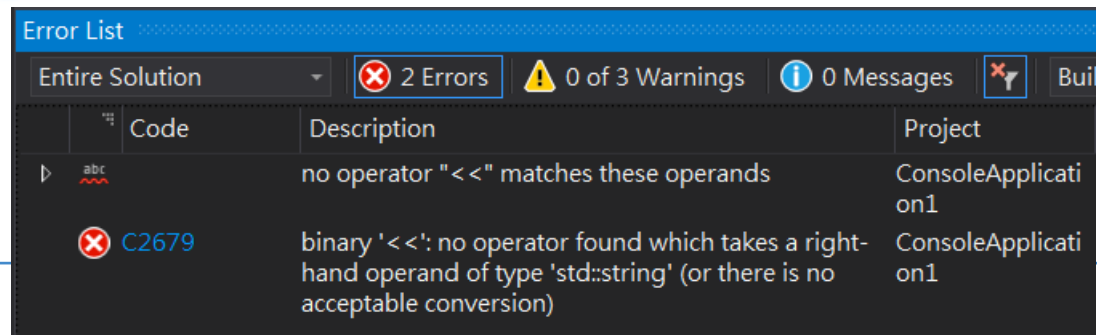
- map是一種關聯容器，支援查表功能之資料結構，資料均以key/value方式存入

常用函式	
[n]	取得key值為n之value
size	計算長度
find	尋找資料
erase	刪除資料
insert	插入資料
count	判斷資料是否存在
lower_bound, (upper_bound)	取得接近下限(上限)之資料

map::map

```
#include <iostream>
#include <map>
#include <string> //沒這行會錯
using namespace std;
```

```
int main() {
    map<char, string> m;
    map<char, string>::iterator it;
    m['a'] = "Andy Lee";
    m['b'] = "Joe Wang";
    m['c'] = "Mary Lin";
    for (it = m.begin(); it != m.end(); it++) {
        cout << "m[" << (*it).first << "\'] is " << (*it).second << endl;
    }
    cout << "m now contains " << m.size() << " elements." << endl;
    return 0;
}
```



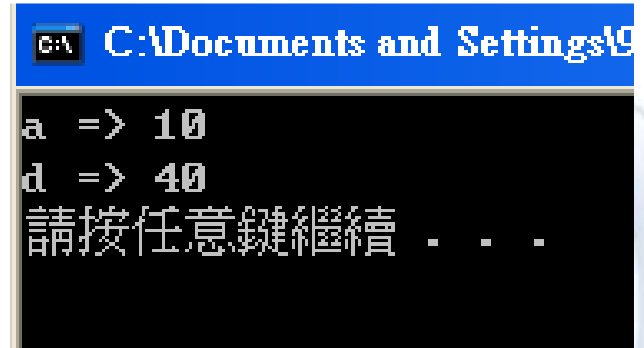
```
C:\Documents and Settings\970722\桌面\1.
m['a'] is Andy Lee
m['b'] is Joe Wang
m['c'] is Mary Lin
m now contains 3 elements.
請按任意鍵繼續 . . .
```

map::find, erase

```
// erasing from map
#include <iostream>
#include <map>
using namespace std;

int map_find_erase() {
    map<char, int> m;
    map<char, int>::iterator it; // insert some values:
    m['a']=10; m['b']=20; m['c']=30;
    m['d']=40; m['e']=50; m['f']=60;
    it=m.find('b');
    m.erase(it); // 刪除it位置之資料
    m.erase('c'); // 刪除key為'c'之資料
    it=m.find('e');
    m.erase( it, m.end() ); // 刪除一段資料

    for ( it=m.begin() ; it != m.end(); it++ )
        cout<< (*it).first << " => " << (*it).second << endl;
    return 0;
}
```



```
C:\Documents and Settings\9
a => 10
d => 40
請按任意鍵繼續 . . .
```

map::insert

```
#include <iostream>
#include <iomanip>
#include <string>
#include <map>
using namespace std;
int main()
{
    string s[][2] = { { "black", "#ffffff" }, { "white", "#000000" },
                      { "blue", "#0000ff" }, { "red", "#ff0000" },
                      { "green", "#00ff00" }, { "", "" } };

    map<string, string> m;
    map<string, string>::iterator it;
    for (int i = 0; !s[i][0].empty(); ++i)
        m.insert(map<string, string>::value_type(s[i][0], s[i][1]));
        //m[s[i][0]] = s[i][1];
    for (it = m.begin(); it != m.end(); ++it) {
        cout << (*it).first << " => " << (*it).second << endl;
    }
    if ((it = m.find("blue")) != m.end()) {
        cout << "'blue' Find! It's value is " << (*it).second << endl;
        //cout << m["blue"] << endl;
    }
    return 0;
}
```

C:\Documents and Settings\970722\桌面\1.exe

```
black => #ffffff
blue  => #0000ff
green => #00ff00
red   => #ff0000
white => #000000
'blue' Find! It's value is #0000ff
請按任意鍵繼續 . . .
```



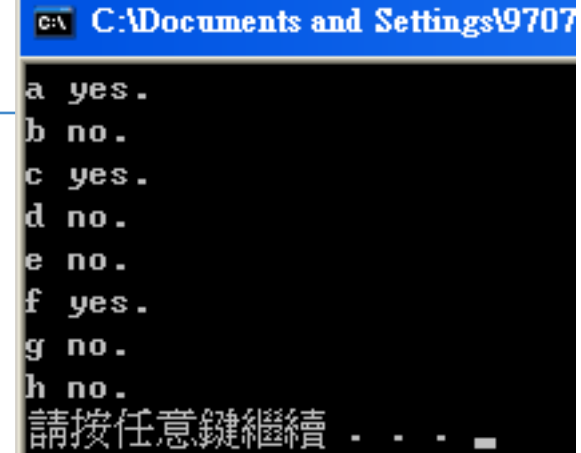
black

#ffffff

map::count

```
#include <iostream>
#include <map>
using namespace std;
int main()
{
    map<char, int> m;
    char c;
    m['a'] = 101;
    m['c'] = 202;
    m['f'] = 303;

    for (c = 'a'; c <= 'h'; c++) {
        cout << c;
        if (m.count(c)>0)
            cout << " yes.\n";
        else
            cout << " no.\n";
    }
    return 0;
}
```



```
C:\Documents and Settings\9707
a yes.
b no.
c yes.
d no.
e no.
f yes.
g no.
h no.
請按任意鍵繼續 . . .
```

example
<https://goo.gl/sR2GN3>

map::lower_bound, upper_bound

```
#include <iostream>
#include <map>
using namespace std;

int main() {
    map<char, int> m;
    map<char, int>::iterator it, itlow, itup;
    m['a'] = 20;    m['c'] = 40;
    m['d'] = 60;    m['g'] = 80;
    m['i'] = 100;

    itlow = m.lower_bound('b'); // 設定下限
    itup = m.upper_bound('d'); // 設定上限
    m.erase(itlow, itup); // 刪除下限到上限所有資料
    // print content:
    for ( it=m.begin() ; it != m.end(); it++ )
        cout << (*it).first << " ==> "
              << (*it).second << endl;
    return 0;
}
```

C:\Documents and Settings\9707

```
a => 20
g => 80
i => 100
```

請按任意鍵繼續 . . .

lower_bound():

回傳一個iterator
pointing 指到第一個
container中key不比
input小的元素
比input大的元素 (\geq) ,
相等時會回傳

upper_bound():

回傳一個 iterator
pointing 指到第一個
container 中key
比input大的元素 ($>$) ,
相等時不回傳

example

<https://goo.gl/GcQLSG>

小練習

- 1) 當使用者可以分別輸入「P」、「M」或「H」來查詢對應的文字內容，請利用映射表(map)的方法分別輸出「Pikachu」、「Mickey Mouse」或「Hello kitty」
- 2) 將程式改成可不斷重覆查詢，直到-1結束
如果使用者輸入的是不存在的key應如何避免程式錯誤？且令使用者可自行新增對應文字。
提示：count
- 3) 如果輸入-2請列出所有的key與value
- 4) 輸入-3後再輸入一存在的key並刪除此組資料
若輸入的key不存在，就輸出key does not exist.

大綱

- 標準樣板函式庫(STL)
 - 抽象指標(Iterator)
 - 容器(Container)
 - 演算法(Algorithm)

演算法(Algorithm)

- STL中的演算法
- 在STL中除了提供容器(類別樣板), 尚提供演算法(函式樣板)以供操作之資料
- `#include <algorithm>`
- 常用演算法
 - `find`
 - `count`
 - `search`
 - `merge`
 - `sort`
 - `for_each`
 - `transform`

find

- find(first, last, val)
 - [first]: iterator
 - [last]: iterator
 - [val]: target value type
 - [回傳值]: iterator
- 針對某個container做搜尋，區間由first及last這兩個iterator指定，目標值為val，找到後回傳其所在元素指標(也是以iterator表示)
- 函數模板原型

```
template<class InIt, class T>  
InIt find(InItfirst, InIt last, const T& val);
```

find

```
#include<iostream>
#include<algorithm>
using namespace std;
```

```
int alg_find1()
{
```

```
    int l[7] = { 1, 3, 2, 5, 1, 2, 1 };
    int *it;
```

```
    it = find(&l[0], &l[7], 5);
```

```
    if (it == l + 7)
```

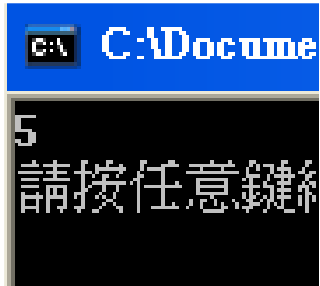
```
        cout << "data not found\n";
```

```
    else
```

```
        cout << *it << endl;
```

```
    return 0;
```

```
}
```



```
#include<iostream>
#include<list>
#include<algorithm>
using namespace std;
```

```
int alg_find2()
{
```

```
    list<int> L;
```

```
    list<int>::iterator it;
```

```
    L.push_back(10);
```

```
    L.push_back(20);
```

```
    L.push_back(30);
```

```
    it = find(L.begin(), L.end(), 30);
```

```
    if (it == L.end())
```

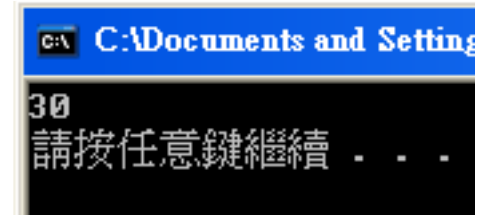
```
        cout << "data not found\n";
```

```
    else
```

```
        cout << *it << endl;
```

```
    return 0;
```

```
}
```



count

- `count(first, last, val)`
 - `[first]`: iterator
 - `[last]`: iterator
 - `[val]`: target value type
 - [回傳值]: int
- 針對某個container做搜尋，區間由first及last這兩個iterator指定，目標值為val，回傳container中元素值為val的個數
- 函數模板原型

```
template<class InIt, class T>  
int count(InIt first, InIt last, const T& val);
```

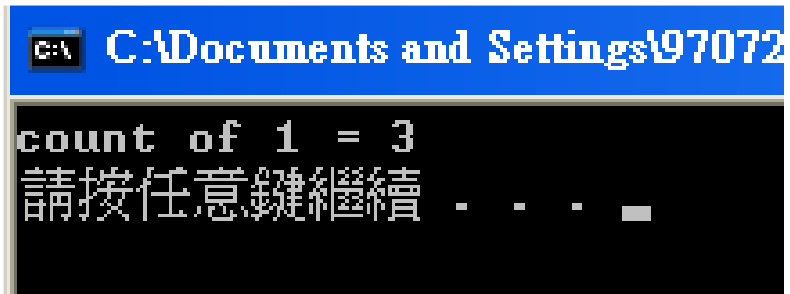
count

```
#include<iostream>
#include<vector>
#include<algorithm>
```

```
using namespace std;
```

```
int main() {
    int a[7] = { 1, 3, 2, 4, 1, 2, 1 };
    vector<int> v(a, a + 7);
    int count_of_1;
    count_of_1 = count(v.begin(), v.end(), 1);
    cout << "count of 1 = " << count_of_1 << endl;

    return 0;
}
```



A screenshot of a Windows command prompt window. The title bar is blue and contains the text "C:\Documents and Settings\97072". The command prompt itself has a black background with white text. It displays the output of the program: "count of 1 = 3" followed by a line of Chinese text "請按任意鍵繼續" (Press any key to continue) and a series of dots and a cursor.

search

- `search(s_first, s_last, t_first, t_last)`
 - `[s_first][t_first]`: iterator
 - `[s_last][t_last]`: iterator
 - [回傳值]: iterator
- 找尋某一資料序列是否出現在另一個容器中
- 函數模板原型

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

search

```
#include<iostream>
#include<list>
#include<vector>
#include<algorithm>
using namespace std;
```

```
int main() {
    int a[7] = { 1, 3, 2, 5, 1, 2, 1 };
    vector<int> v(a, a + 7);
    vector<int>::iterator it;
    list<int> L;
    L.push_back(5); //若順序一樣的話就會找到
    L.push_back(1);
    L.push_back(2);
    it = search(v.begin(), v.end(), L.begin(), L.end());
    if (it != v.end()) //有找到
        cout<< *it << " " << *(it+1) << " " << *(it+2) << endl;
    return 0;
```

C:\Documents and S

5 1 2

請按任意鍵繼續 -

merge

- merge(s1_first, s1_last, s2_first, s2_last, t_first)
 - [s1_first][s2_first]: iterator
 - [s1_last][s2_last]: iterator
 - [t_first]: iterator
 - [回傳值]: iterator
- 合併s1與s2兩資料於t
- 函數模板原型

```
template<class InIt1, class InIt2, class OutIt>
```

```
OutIt merge(InIt1 first1, InIt1 last1, InIt2 first2, InIt2 last2, OutIt x);
```

merge

```
#include<iostream>
#include<vector>
#include<algorithm>
using namespace std;
```

```
int main()
```

```
{
```

```
    int a[7] = { 3, 2, 5, 1, 2, 1, 8 };
```

```
    int b[4] = { 1, 7, 4, 9 };
```

```
    vector<int> v1(a, a + 7);
```

```
    vector<int> v2(b, b + 4);
```

```
    vector<int> v3(v1.size() + v2.size());
```

```
    merge(v1.begin(), v1.end(), v2.begin(), v2.end(), v3.begin());
```

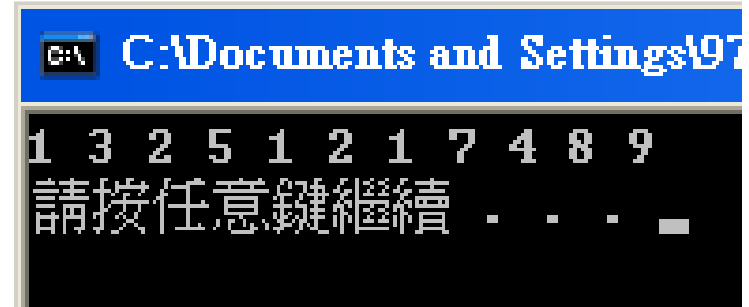
```
    for (int i = 0; i<v3.size(); i++)
```

```
        cout << v3[i] << " ";
```

```
    cout << endl;
```

```
    return 0;
```

```
39 }
```



A screenshot of a Windows command prompt window. The title bar is blue and contains the text "C:\Documents and Settings\97". The command prompt shows the output of the merge program: "1 3 2 5 1 2 1 7 4 8 9" followed by "請按任意鍵繼續" (Press any key to continue) and a series of dots and a cursor.

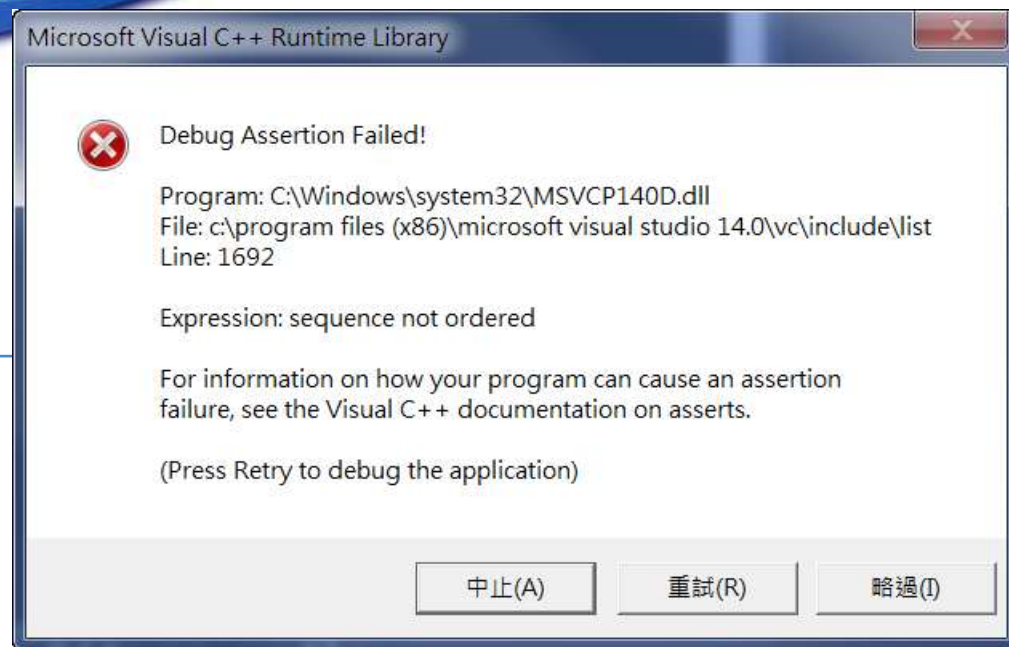
VS 沒排序 會出錯

```
bool mygreater(int x, int y)
{
    return x<y;
}
```

```
vector<int> v1(a, a + 7);
vector<int> v2(b, b + 4);
vector<int> v3(v1.size() + v2.size());
```

```
sort(v1.begin(), v1.end(), mygreater);
sort(v2.begin(), v2.end(), mygreater);
```

```
merge(v1.begin(), v1.end(), v2.begin(), v2.end(), v3.begin());
```



sort

- `sort(first, last)`
- `sort(first, last, f)`
 - `[first]`: iterator
 - `[last]`: iterator
 - `[f]`: 函式
 - `[回傳值]`: void
- 資料排序(預設由小到大)
- 函數模板原型
- 利用sort演算法只能對序列容器進行排序，就是線性的（如vector，list，deque）

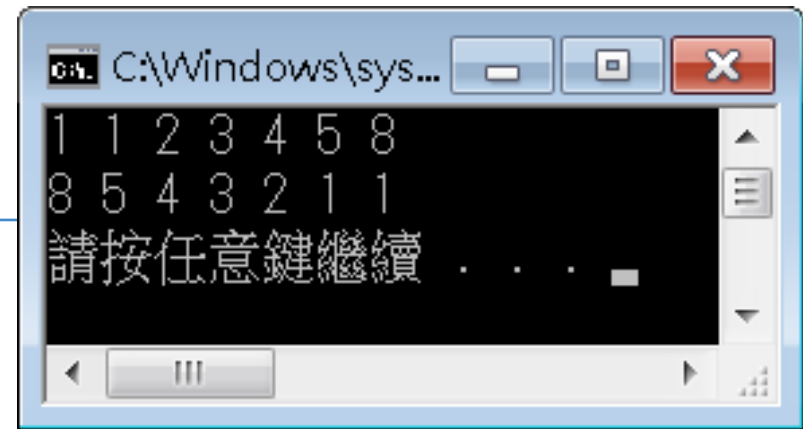
```
template<class RanIt>
void sort(RanIt first, RanIt last);

template<class RanIt, class Pred>
void sort(RanIt first, RanIt last, Pred pr);
```

sort

```
#include<iostream>
#include<vector>
#include<algorithm>
using namespace std;
```

```
int main()
{
    int a[7] = { 8, 1, 3, 2, 5, 1, 4 };
    vector<int> v(a, a + 7);
    vector<int>::iterator it;
    sort(v.begin(), v.end()); //由小排到大
    for (it = v.begin(); it != v.end(); it++)
        cout << *it << " ";
    cout << endl;
    sort(v.begin(), v.end(), mygreater); //mygreater由大排到小
    for (it = v.begin(); it != v.end(); it++)
        cout << *it << " ";
    cout << endl;
    return 0;
}
```



```
C:\Windows\sys...
1 1 2 3 4 5 8
8 5 4 3 2 1 1
請按任意鍵繼續 . . .
```

```
//將此函式改成樣版試試
bool mygreater(int x, int y)
{
    return x>y;
}
bool myless(int x, int y)
{
    return x<y;
}
```

小練習

- 將sort()用的比對函式改成樣板 <https://goo.gl/P32DDh>
<https://goo.gl/xU5wop>
- 改成由大到小排序
- 排序map (依key或依value排序※須存到別的容器)
 - 利用sort演算法只能對序列容器進行排序，
如線性的（如vector，list，deque）
- 排序 `vector<struct Pokemon>` (依 Lv 排序)

```
struct Pokemon{  
    char Name[100]; //名字  
    int Lv; //等級  
    int Hp; //血量  
}
```

<https://jgirl.ddns.net/problem/0/1080>

for_each

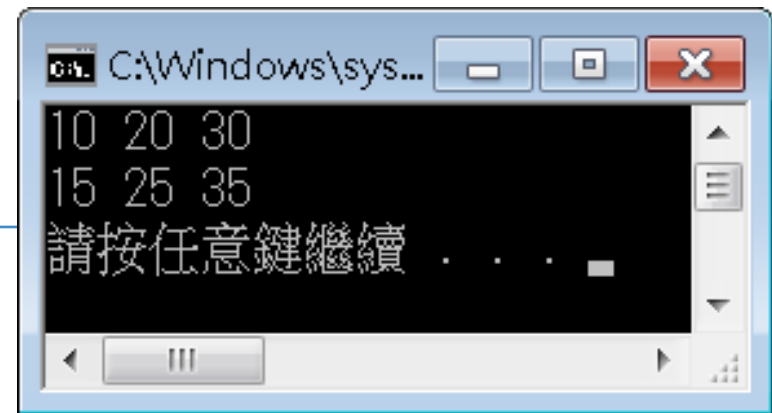
- **for_each(first, last, f)**
 - [first]: iterator
 - [last]: iterator
 - [f]: 函式
 - [回傳值]: 函數物件指標
- 對container中從first所指的元素起到last為止，其間每一個元素做某個動作(由函數f指定)
- 函數模板原型

```
template<class InIt, class Fun>  
Fun for_each(InIt first, InIt last, Fun f) ;
```

for_each

```
#include<iostream>
#include<list>
#include<algorithm>
using namespace std;

int main()
{
    list<int> L;
    L.push_back(10);
    L.push_back(20);
    L.push_back(30);
    for_each(L.begin(), L.end(), print);
    cout << endl;
    for_each(L.begin(), L.end(), add);
    for_each(L.begin(), L.end(), print);
    cout << endl;
    return 0;
}
```



```
C:\Windows\sys...
10 20 30
15 25 35
請按任意鍵繼續 . . .
```

```
void print(int x)
{
    cout << x << " ";
}
void add(int &x)
{
    x = x + 5;
}
```


transform

- transform(first, last, output, f)
 - [first]: iterator
 - [last]: iterator
 - [output]: container
 - [f]: 函式
 - [回傳值]: 函數物件指標
- 同for_each，但會把結果放在output容器中
- 函數模板原型

```
template<class InIt, class OutIt, class Unop>
```

```
46 OutIt transform(InIt first, InIt last, OutIt x, Unop uop);
```

transform

```
#include<iostream>
#include<list>
#include<vector>
#include<algorithm>
using namespace std;
```

```
int main()
{
    list<int> L;
    vector<int> v(3);
    L.push_back(10);
    L.push_back(20);
    L.push_back(30);
    transform(L.begin(), L.end(), v.begin(), add);
    for_each(L.begin(), L.end(), print);
    cout << endl;
    for_each(v.begin(), v.end(), print);
    cout << endl;
    return 0;
}
```

Error List

Entire Solution

1 Error

0 of 5 Warnings



Code

Description

C2440

'=': cannot convert from 'void' to 'int'

```
C:\Windows\system32\cmd.exe
15 25 35
0 0 0
請按任意鍵繼續 . . .
```

```
void print(int x) {
    cout << x << " ";
}
void add(int &x) {
    x = x + 5;
}
int add2(int &x) {
    x = x + 5;
    return 0;
}
int add3(int x) {
    return x + 5;
}
```

E:\NTU CSIE Train\錄影\

```
10 20 30
15 25 35
```

STL 參考網站

- <http://www.cplusplus.com/reference/stl/>
- <http://www.cplusplus.com/reference/algorithm/>
- 《STL 源碼剖析》 - 侯捷
- C++ Boost STL

小練習

- 請讓使用者輸入一數列，以push_back()加入list當中(-1結束，list中不含-1)
- 將其由小到大排序後印出
- 在第四個位置插入一整數10後印出
- 印出list中有幾個整數10
- 將其由大到小排序後印出

<https://jgirl.ddns.net/problem/0/3063>

Visual Studio 不想使用 stdafx.h

- 如果在專案中不使用先行編譯標頭檔，請將原始程式檔的 [建立/使用先行編譯標頭] 屬性設定為 [未使用先行編譯標頭檔]。
- 請執行下列步驟：
 1. 在專案的 [方案總管] 窗格中，用右鍵按一下專案名稱，然後按一下 [屬性]。
 2. 在左邊的窗格中，按一下 [C/C++]。
 3. 按一下 [先行編譯標頭]。
 4. 在右窗格中，按一下 [建立/使用先行編譯標頭]，然後按一下 [未使用先行編譯標頭檔]。

