Ch5. 標準模板庫 STL (Standard Template Library)

本章僅做簡介，無 STL 的相關實作範例，實作會在之後的課程中進行，這裡只先列出部分經典語法給讀者參考。

課程大綱

A. 標準樣板函式庫 STL 簡介
B. STL 中的容器 Container
C. STL 中的迭代器 Iterator
D. STL 中的演算法 Algorithm

第一節：標準樣板函式庫 STL 簡介

1. C++ 中的 STL

C++ 可以被看作是下列數種程式語言的「集合」，黏合四種語言的長處與精神：

A. C // C++ 是向後相容的，意即可以在 C++ 中寫 C
B. Objected-Oriented C++ // 在 C 中導入物件、類別的概念
C. Template C++ // 加入模板，讓函式或類別處理各種不同資料
D. STL

C++ 不是單一的語言，因此不同情況下實作邏輯不盡相同。一個明顯的例子是陣列 Array 和變數 Variable 不會自動初始化，因其屬於 C 語言集合，而 C 中並沒有初始化為 0 的習慣，相對的，向量 Vector 中的資料卻會初始化成 0，因其屬於 STL 集合。

把 C++ 看作是數個語言的集合，有助於建立對 C++ 操作與背後原理更深的了解。

2. STL 的歷史

(1) 1990 年左右 Alex Stepanov 擴展原有 C++ 函式庫為 STL

A. 主要由類別模板與函式模板構成
B. 1994 年因廣泛使用被採納為 ANSI/ISO 標準，得名「標準模板函式庫」

之所以在 STL 中引入模板，主要是不希望類別（資料結構）或函式只能處理單一種資料型別。

(2) STL 由三大部分構成

A. 容器 Container：儲存資料用的類別模板

B. 迭代器 Iterator：負責操作/應用容器的指標，指向容器內的資料

C. 演算法 Algorithm：負責對容器做操作的函式模板

字串 String 也是 STL 的一部分

3. 容器 Container

「容器」是儲存資料/物件的資料結構，由類別模板實作，可儲存不同型態的資料，且具有基本的資料操作函式。

常見的容器有四大類：

A. Sequence container

B. Associative container

C. Unordered associative containers

D. Container adapter

A. Sequence container 有一定的順序：把資料排序後，可以一個個去取用
支援「依序 sequentially」存取。
常見資料結構：vector、list、deque

B. Associative container 資料間本身有關係：如同班上每個人有學號、客戶有客戶編號，用二元樹儲存排序後的資料，可以將搜尋的複雜度降為 $O(\log_2 n)$。
常見資料結構：set、map、multiset、multimap

C. Unordered associative container 沒有次序：透過雜湊表儲存資料，搜尋的時間複雜度平均為 O(1)、最壞情況下為 O(n)。
常見資料結構（都以 unordered 開頭）：unordered_set、unordered_map、unordered_multiset、unordered_multimap

D. Container adapter：提供特殊的介面 / 資料存取順序，只能從特定的方

向新增或刪除資料。

常見資料結構：stack、queue、priority_queue

(1) Vector 向量：連續配置的動態陣列

A. 連續記憶體配置的動態陣列
B. 大小可隨時增長
C. 支援提供索引值存取
D. 可取代陣列的使用

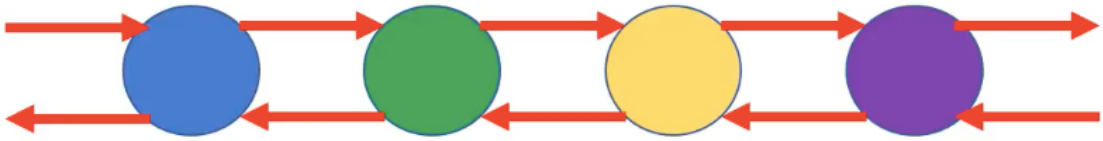
Vector 的常用操作	
[i]	取得第 i 個索引值的資料
at(i)	取得第 i 個索引值的資料
push_back	新增最後一筆資料
pop_back	移除最後一筆資料
front	取得第一個元素
back	取得最後一個元素
insert	插入資料
erase	清空所有資料
size	回傳現在長度
begin	回傳開頭元素的 iterator
end	回傳結尾元素的 iterator
empty	回傳容器是否為空

(2) List：雙向的鏈結串列

A. 記憶體位置不連續

B. 不支援索引值存取

C. 新增/刪除特定節點：O(1)（Array/Vector 是 O(n)）



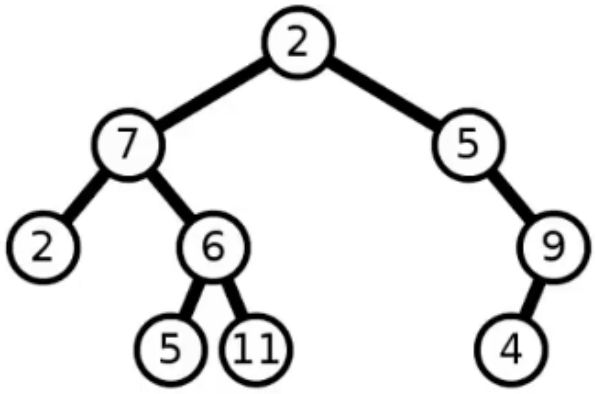
List 是一種「雙向」的鏈結串列，鏈結串列代表資料並非被放在連續的記憶體位置中。為了找到資料的位置，每一筆資料都用「指標」連結到下一筆資料，在取用時，第一筆資料會含有第二筆資料的位置資訊，第二筆資料又含有下一筆資料的位置資訊，依此類推。

雙向鏈結串列中的「雙向」指的是可以從前面的資料往後面資料移動，也可以反過來由後面的資料往前移動。另外，因為利用「索引值」存取資料的前提是記憶體位置必須連續，所以 List 中並不支援這種存取方式。

List 的常用操作	
push_back	新增最後一筆資料
pop_back	移除最後一筆資料
push_front	新增第一筆資料
pop_front	移除第一筆資料
front	取得第一個元素
back	取得最後一個元素
insert	插入資料
erase	清空所有資料
size	回傳現在長度
begin	回傳開頭元素的 iterator
end	回傳結尾元素的 iterator
remove	刪除特定資料
reverse	資料反轉
merge	合併資料

(3) Map / Set

A. 透過二元樹建立，有次序關係
B. 支援查表功能
C. 以 Key / Value 方式儲存
D. 查詢複雜度 $O(\log_2 n)$



如上圖，在二元樹中，一筆資料會連到另外兩筆資料；Key 可以被視為每個節點上的編號，按照編號來生成二元樹，比它小的放左邊，比它大的放右邊。

因為使用了二元樹的結構，在 Map / Set 中每次「搜尋」的動作可以刪除一半的元素，複雜度為 $O(\log_2 n)$。

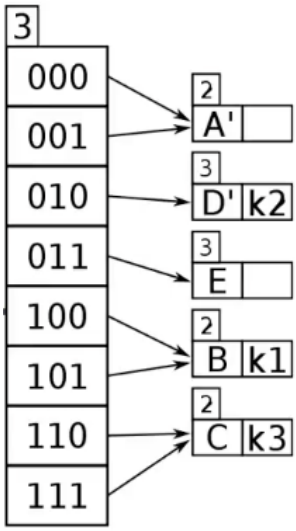
Map/Set 的常用操作	
[key]	取得 key 對應的 value
size	計算長度
find	尋找特定資料
erase	清空所有資料
insert	插入特定資料
count	查看資料是否存在（因 key 不會重複）
lower_bound	取得最接近的下限資料
upper_bound	取得最接近的上限資料
begin	回傳開頭元素的 iterator
end	回傳結尾元素的 iterator

(4) Unordered_Set / Unordered_Map

A. 透過雜湊表建立

B. 沒有次序關係

C. 查詢複雜度 O(1)



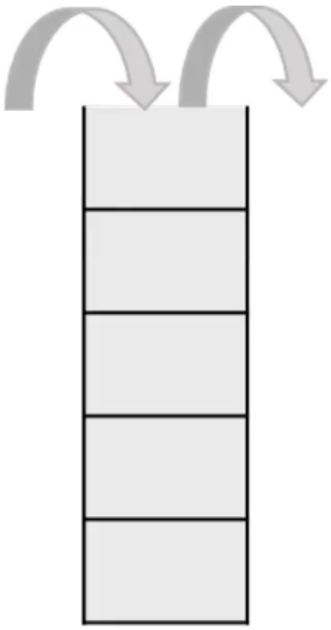
「Unordered」代表資料之間沒有次序關係。因為使用雜湊表建立，查詢的複雜度為 O(1)。

Unordered_Set / Unordered_Map 的常用操作	
[key]	取得 key 對應的 value
size	計算長度
find	尋找特定資料
erase	清空所有資料
insert	插入特定資料
count	查看資料是否存在（因 key 不會重複）
lower_bound	取得最接近的下限資料
upper_bound	取得最接近的上限資料
begin	回傳開頭元素的 iterator
end	回傳結尾元素的 iterator

(5) 堆疊 Stack

A. 先進後出、後進先出

B. 可用陣列或鏈結串列實作



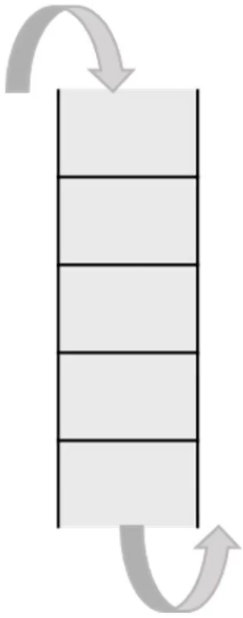
在 Stack 中，新增資料和刪除資料是同向的，就像吃洋芋片時，要從圓筒的上方拿出來，要是吃不完，同樣是從上面放回去。這代表「先放進去」的洋芋片在圓筒最下方，會被「最後拿出來」，相對的，「最後放進去」的洋芋片在最上方，下次會「先被拿出來」。

Stack 的常用操作	
top	回傳最上層的資料
push	插入一筆資料
pop	刪除一筆資料
empty	回傳容器是否為空
size	計算長度

(6) 佇列 Queue

A. 先進先出、後進後出

B. 可用陣列或鏈結串列實作



Queue 與 Stack 相對，它新增資料和刪除資料在異側，從上面新增，則從下面刪除。先進去佇列的資料，會先被處理。

Queue 的常用操作	
top	回傳最上層的資料
push	插入一筆資料
pop	刪除一筆資料
back	回傳最後一筆資料
empty	回傳容器是否為空
size	計算長度

(7) 優先權佇列 Priority Queue

A. 依照權重大小排序的 Queue

B. 按照特定順序依序吐出資料

Priority Queue 跟 Queue 很類似，但是把資料放進 Priority Queue 的時候，會給出一個權重，這使我們可以根據權重大小排序，權重大的先輸出。

這就像電腦的工作管理員，打開工作管理員後，會發現每個工作排程有各自的優先次序，優先次序大、比較緊急工作的會先被作業系統處理，相當於以重要或緊急程度對工作排序。

Priority Queue 的常用操作	
top	回傳最上層的資料
push	插入一筆資料
pop	刪除一筆資料
front	回傳第一筆資料
back	回傳最後一筆資料
empty	回傳容器是否為空
size	計算長度

(8) Pair

Pair 中可以存「一對」資料，這兩筆資料的資料型別可以不同，分別被稱作 first 和 second。當資料總是以「一對」的方式出現，比如「人名」跟「身高」分別為一個「字串 string」和「浮點數 float」，就可以利用 Pair 結構。

Pair 的結構	
1	struct pair<T1,T2>{
2	T1 first;
3	T2 second;
4	};

Pair 的使用方式	
1	std::pair<double,double> p1;
2	std::pair<double,double> p2 = std::make_pair(1.0,2.0);
3	cin >> p1.first >> p1.second;

首先，要指定兩筆資料的資料型別，比如 pair<double,double> 指的是 pair 當中的兩筆資料都是 double。

接下來，可以用 p1.first 或 p1.second 去取出兩筆資料的內容，也可以用 make_pair 語法，先生成一個 pair 並初始化其值，再 assign 給另一個 pair（上例中的 p2）。

第二節：迭代器與演算法

1. 迭代器 Iterator

迭代器的功能是讓使用者（程式開發者/我們）可以逐個存取容器中的元素，可以視為指向容器內資料的指標。

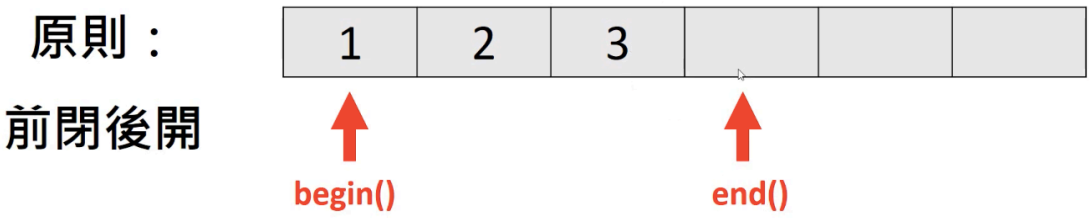
所有 STL 裡的容器 Container 都有 Iterator，利用 container::iterator 語法進行宣告後，就可以讓迭代器指向容器中的資料了。

(1)「前閉後開」原則

每個容器都有 begin()、end() 函式

A. begin()：回傳容器的第一個元素

B. end()：回傳最後一個元素「後一個位置」的記憶體位址



根據「前閉後開」原則，如果容器內儲存了三筆資料：1、2 和 3，begin 會指向 1，end 則指向最後一筆資料 3 的下一個位置（容器外）。

(2) 迭代器的使用

迭代器在使用上與指標相似，利用 iterator 可以依序取出容器中所有元素

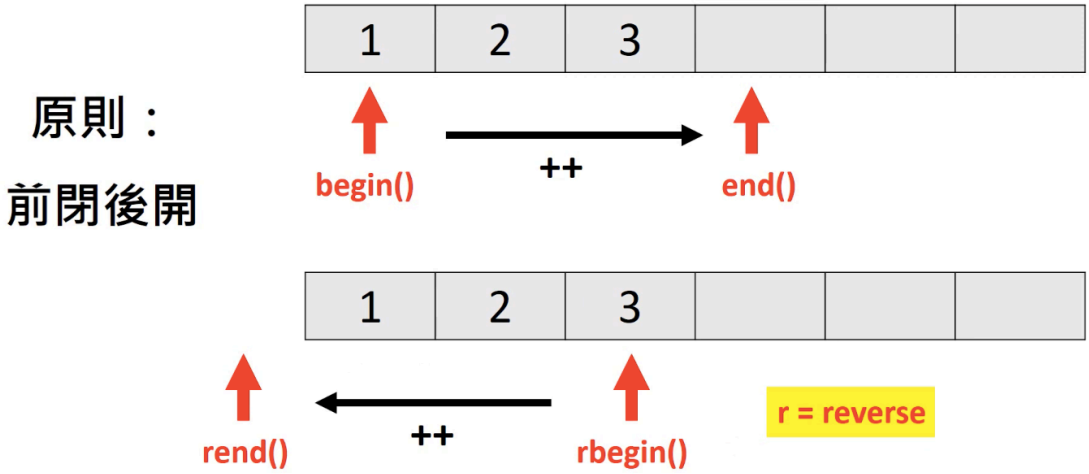
A. iter++：檢索下一個元素

B. iter--：檢索上一個元素

C. *iter：取出該 iterator 指到的元素

利用迭代器遍歷容器	
1	// 當 iter 還沒有到達 container 結尾的時候，依序往後移一筆資料
2	for (iter = container.begin() ; iter!=container.end() ; iter++)
3	{
4	// codes
5	}

(3) rbegin 與 rend



一般而言，++ 運算子是由第一筆資料向後移動，但是使用 rbegin 與 rend 時，方向則相反。

rbegin 指到的是最後一筆資料，rend 指到的則是比第一個資料還前面的位置，由 rbegin 往 rend 方向移動時，++ 指的是往前一筆資料移動，不過仍然有前閉後開原則的適用。

2. 演算法 Algorithm

STL 中的「演算法」其實是一些「函式模板」，功能是對容器內的資料進行常用的操作。

常見的演算法	
搜尋	find
合併	merge
反轉	reverse
排序	sort
計數	count
刪除	remove

(1) 搜尋：find (起點,終點,值)

A. 起點、終點都是迭代器，表示出搜尋進行的範圍

B. 回傳一個迭代器，標示第一個找到的、具有相符值的資料位置

C. 對容器做搜尋，範圍從起點到終點

兩個迭代器 first 與 last 決定搜尋範圍，回傳指向第一筆值為 val 的資料的迭代器。

```
template<class T1,class T2>
T1 find(T1 first, T1 last, const T2& val);
```

(2) 計算出現次數：count (起點,終點,值)

A. 起點、終點都是 iterator

B. 回傳個數 int

C. 對 container 做搜尋，範圍從起點到終點

兩個迭代器 first 與 last 決定範圍，計算特定值 val 出現的次數。

```
template<class T1,class T2>
int count(T1 first, T1 last, const T2& val);
```

(3) 搜尋資料列：search (搜尋起點,搜尋終點,目標起點,目標終點)

A. 起點、終點都是 iterator

B. 回傳 iterator（找到的位置）

C. 對 container 做搜尋，查看某一資料列是否出現在另一容器中

template<class T1,class T2>

T1 count(T1 first_1, T1 last_1, T2 first_2, T2 last_2);

T1：iter 的資料型別，要找的範圍 first 到 last

T2：目標（想要比對的序列）的起點和終點範圍

比較 find 和 search，find 是找第一筆相符的資料；search 則是看某一個序列是否出現在一個容器當中。

(4) 排序：sort (起點,終點,函式指標)

A. 起點、終點都是 iterator

B. 回傳 void

C. 函式可以為空，預設「由小到大」對容器裡的資料做排序

a. greater：由大到小

b. less：由小到大（預設）

傳入兩個引數 first 與 last：

template<class T1>

void sort(T1 first, T1 last);

傳入三個引數 first、last 與 pointer：

template<class T1, class T2>

void sort(T1 first, T1 last, T2 pointer);

其中 T2 是函式指標，使用 T2 指到的函式規則進行排序

(5) 依序處理：for_each (起點,終點,函式)

A. 起點、終點都是 iterator

B. 回傳函式指標

C. 把 container 裡的資料依序丟入函式中

把起點 first 到終點 last 的資料依序丟到後面 pointer 所指到的函式中做處理。

```
template<class T1, class Func>
Func sort(T1 first, T1 last, Func f);
```