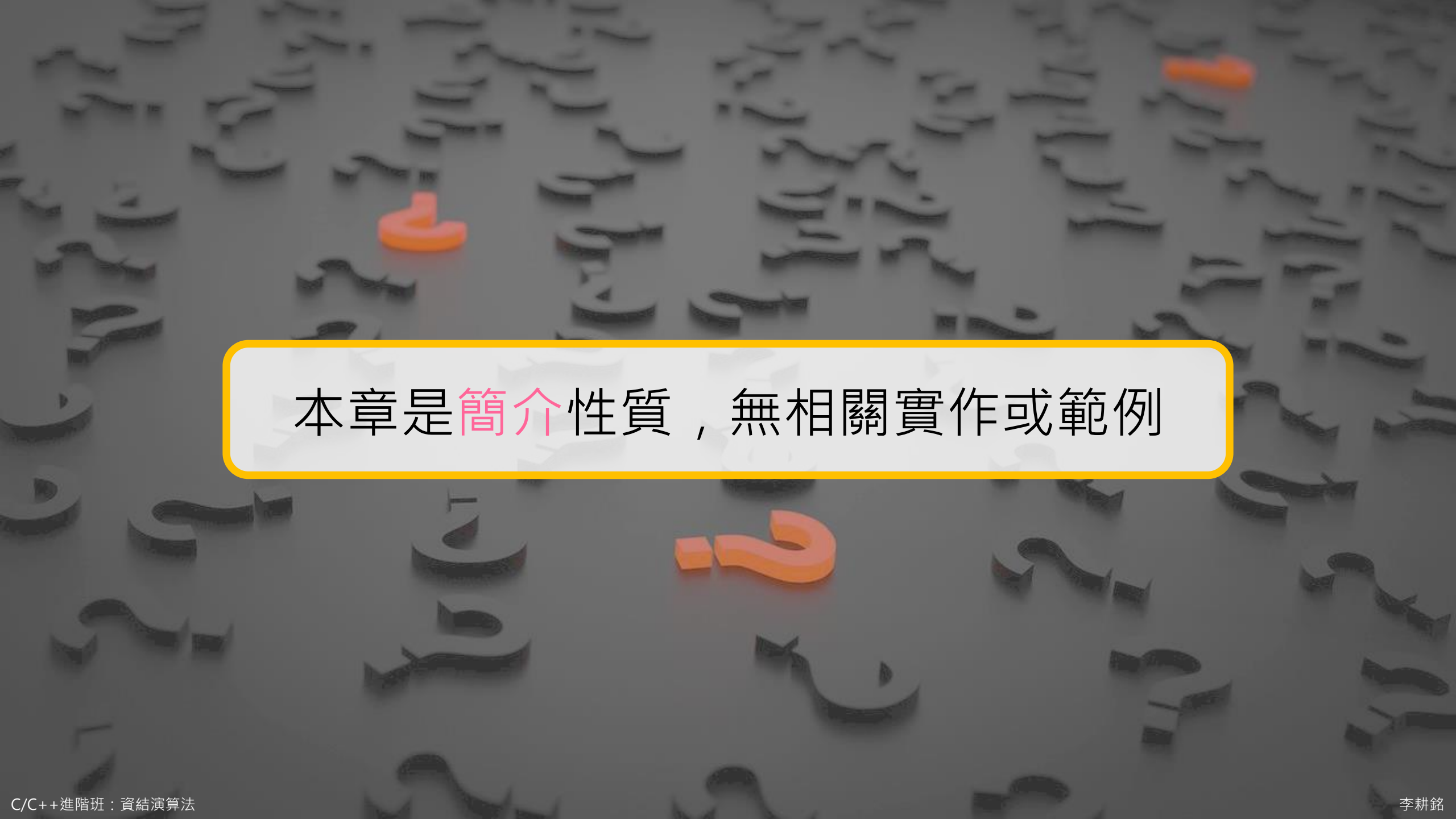
C/C++ 進階班
資料結構

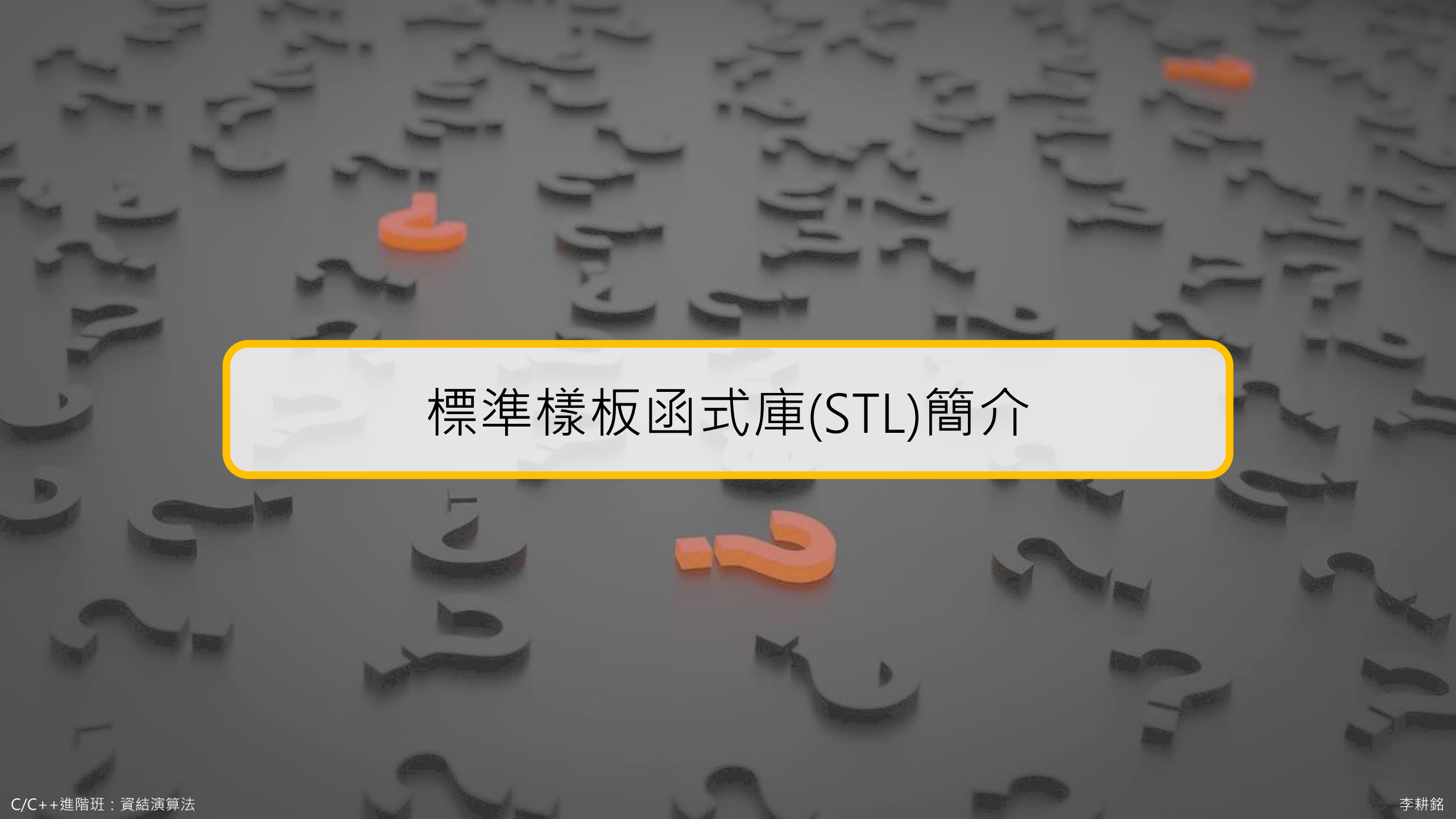
標準模板庫
(Standard Template Library)

李耕銘


本章是簡介性質，無相關實作或範例

課程大綱

- 標準樣板函式庫(STL)簡介
- Container(容器)
- Iterator(迭代器)
- Algorithm(演算法)



標準樣板函式庫(STL)簡介

李耕銘

標準樣板函式庫(STL)

把 C++ 看做是許多程式語言的集合

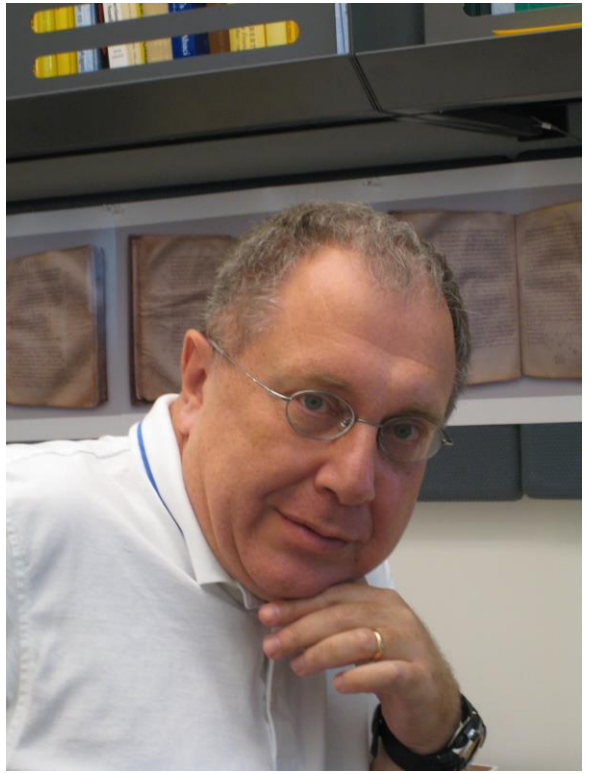
1. C
2. Objected-Oriented C++
3. Template C++
4. STL

Q：Array、Variable 不會自動初始化，Vector 卻會？

標準樣板函式庫(STL)

Standard Template Library (STL)

- 1990 年左右 Alex Stepanov 擴展原有 C++ 函式庫：STL
  - 類別模板
  - 函式模板
- 1994 年 STL 被採納為 ANSI/ISO 標準



標準樣板函式庫(STL)

STL 主要由三大部分構成：

1. Container(容器)
   - 負責儲存資料的類別模板
2. Iterator(迭代器)
   - 負責操作/應用容器的指標
3. Algorithm(演算法)
   - 負責對容器做操作的函式模板

字串 (String) 也是 STL 的一部分

Container(容器)

Container(容器)

- Container (容器)
  - ✓ 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
  - ✓ 具有基本的資料操作函式
- 常見的容器四大類：
  1. Sequence container
  2. Associative container
  3. Unordered associative containers
  4. Container adapter

Container(容器)

- 常見的容器：
  - Sequence container
    - ✓ 支援依序 (sequentially) 存取
    - ✓ vector
    - ✓ list
    - ✓ deque

Container(容器)

- 常見的容器：
  - Associative container
    - ✓ 用二元樹儲存排序後的資料
    - ✓ 搜尋在 $O(\log_2 n)$
    - ✓ set
    - ✓ map
    - ✓ multiset
    - ✓ multimap

Container(容器)

- 常見的容器：
  - Unordered associative containers
    - ✓ 透過雜湊表儲存資料
    - ✓ 搜尋的時間複雜度：平均O(1)，最壞O(n)
    - ✓ unordered_set
    - ✓ unordered_map
    - ✓ unordered_multiset
    - ✓ unordered_multimap

Container(容器)

- 常見的容器：
  - Container adapter
    - ✓ 提供特殊的介面/資料存取順序
    - ✓ stack
    - ✓ queue
    - ✓ priority_queue

Vector

Vector (向量)

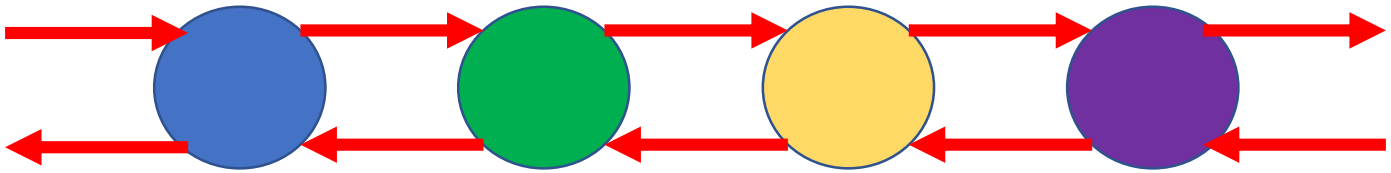
- 連續記憶體配置的動態陣列
- 可隨時增長其大小，
- 支援提供索引值存取
- 可取代陣列的使用

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	回傳開頭元素之iterator
end	回傳結尾元素之iterator
empty	回傳容器是否為空

List

List (鏈結串列)

- **雙向 Linked List**
- **記憶體位置不連續**
- **不支援提供索引值存取**
- **新增/刪除特定節點：O(1)**

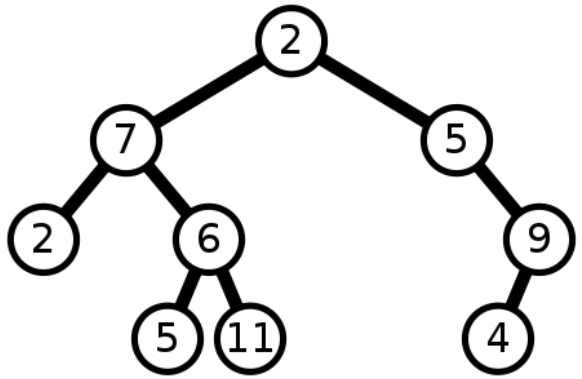


常用操作	
push_back	新增最後一筆資料
pop_back	移除最後一筆資料
push_front	新增第一筆資料
pop_front	移除第一筆資料
front	取得第一個元素
back	取得最後一個元素
insert	插入資料
erase	清空所有資料
size	回傳現在長度
begin	回傳開頭元素之iterator
end	回傳結尾元素之iterator
remove	刪除特定資料
reverse	資料反轉
merge	合併資料

Set/Map

Map/Set

- 透過二元樹建立，有次序關係
- 支援查表功能
- 以 Key/Value 方式儲存
- 查詢為：O($log_2 n$)

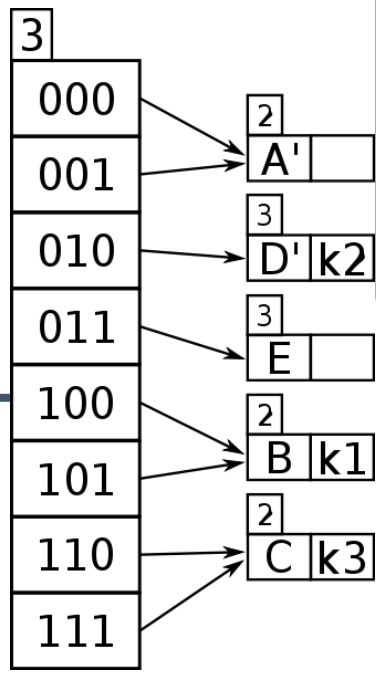


常用操作	
[key]	取得 key 對應的 value
size	計算長度
find	尋找特定資料
erase	清空所有資料
insert	插入特定資料
count	查看資料是否存在(因 key 不會重複)
lower_bound	取得最接近的下限資料
upper_bound	取得最接近的上限資料
begin	回傳開頭元素之iterator
end	回傳結尾元素之iterator

Unordered_Set/Unordered_Map

Unordered_Set/Unordered_Map

- 透過雜湊表建立
- 沒有次序關係
- 查詢為：O(1)

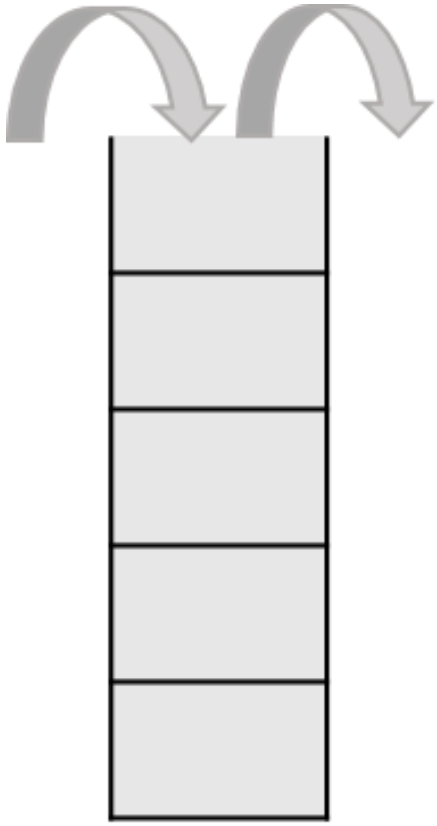


常用操作	
[key]	取得 key 對應的 value
size	計算長度
find	尋找特定資料
erase	清空所有資料
insert	插入特定資料
count	查看資料是否存在(因 key 不會重複)
lower_bound	取得最接近的下限資料
upper_bound	取得最接近的上限資料
begin	回傳開頭元素之iterator
end	回傳結尾元素之iterator

Stack

Stack(堆疊)

- 先進後出、後進先出
- 可用陣列或鏈結串列實作



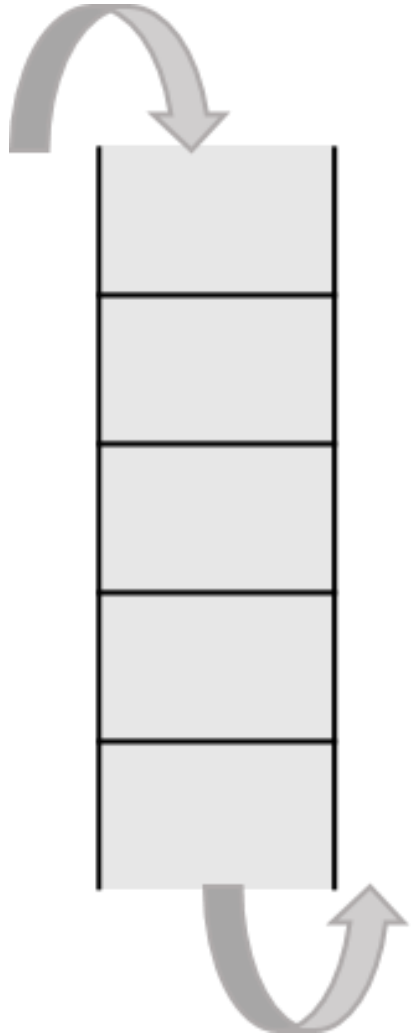
常用操作	
top	回傳最上層的資料
push	插入一筆資料
pop	刪除一筆資料
empty	回傳容器是否為空
size	計算長度

Queue

Queue(佇列)

- 先進先出、後進後出
- 可用陣列或鏈結串列實作

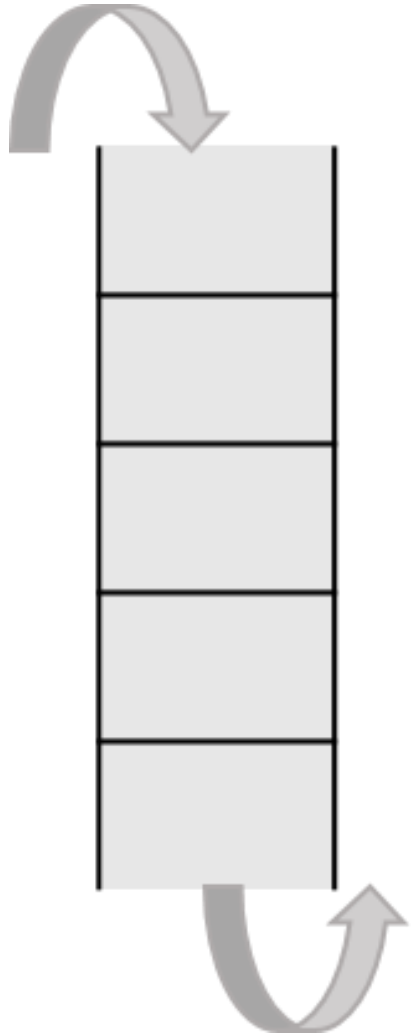
常用操作	
top	回傳最上層的資料
push	插入一筆資料
pop	刪除一筆資料
front	回傳第一筆資料
back	回傳最後一筆資料
empty	回傳容器是否為空
size	計算長度



Priority Queue

Priority Queue (優先權佇列)

- 依照權重大小排序的 Queue
- 按照特定順序依序吐出資料



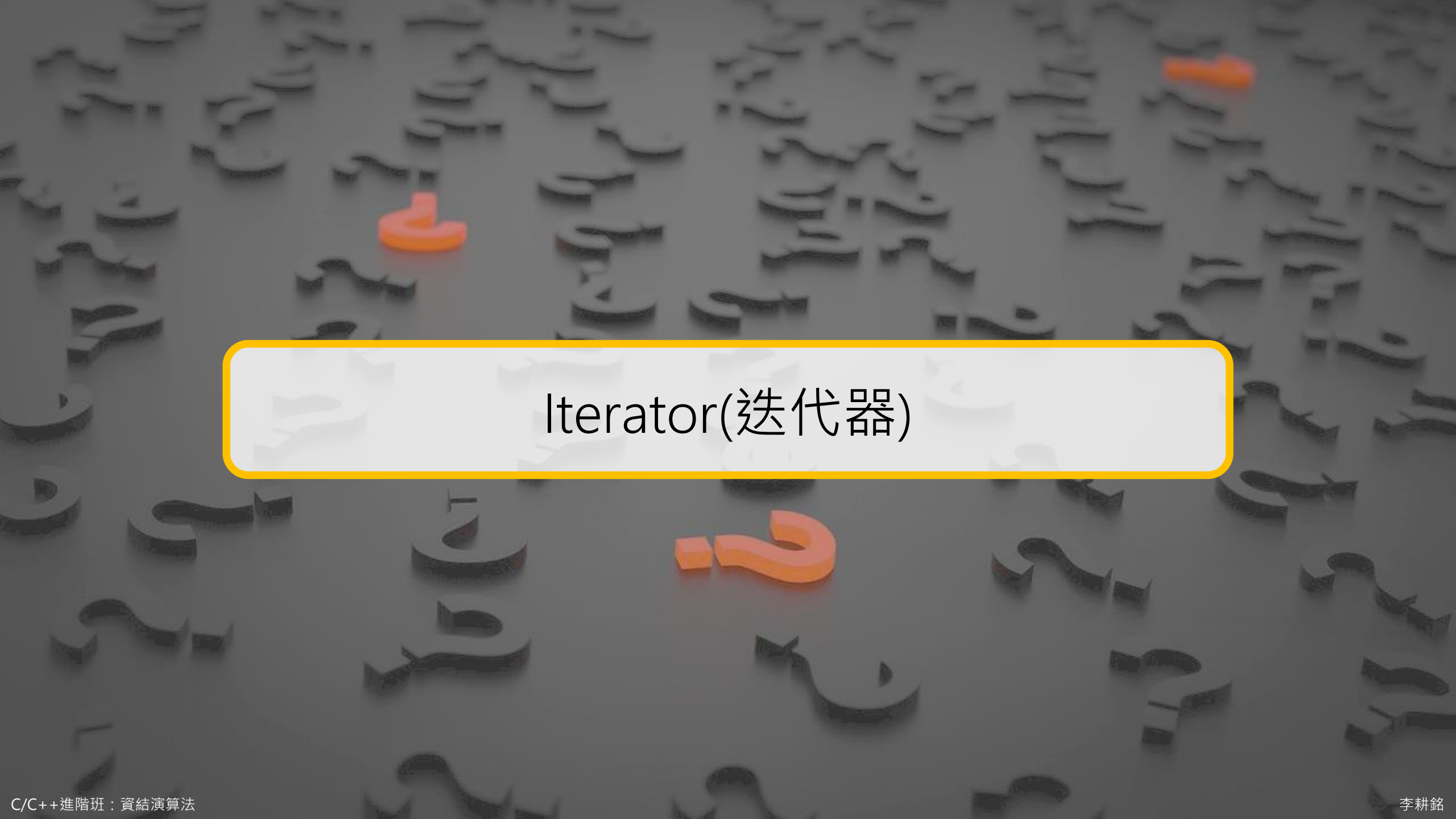
常用操作	
top	回傳最上層的資料
push	插入一筆資料
pop	刪除一筆資料
front	回傳第一筆資料
back	回傳最後一筆資料
empty	回傳容器是否為空
size	計算長度

Pair

儲存一對資料，分別為 first 與 second

```
struct pair<T1, T2>{
        T1 first;
        T2 second;
};
```

```
std::pair<double, double> p1;
std::pair<double, double> p2 = std::make_pair(1.0,2.0);
cin >> p1.first >> p1.second;
```



Iterator(迭代器)

Iterator(迭代器)

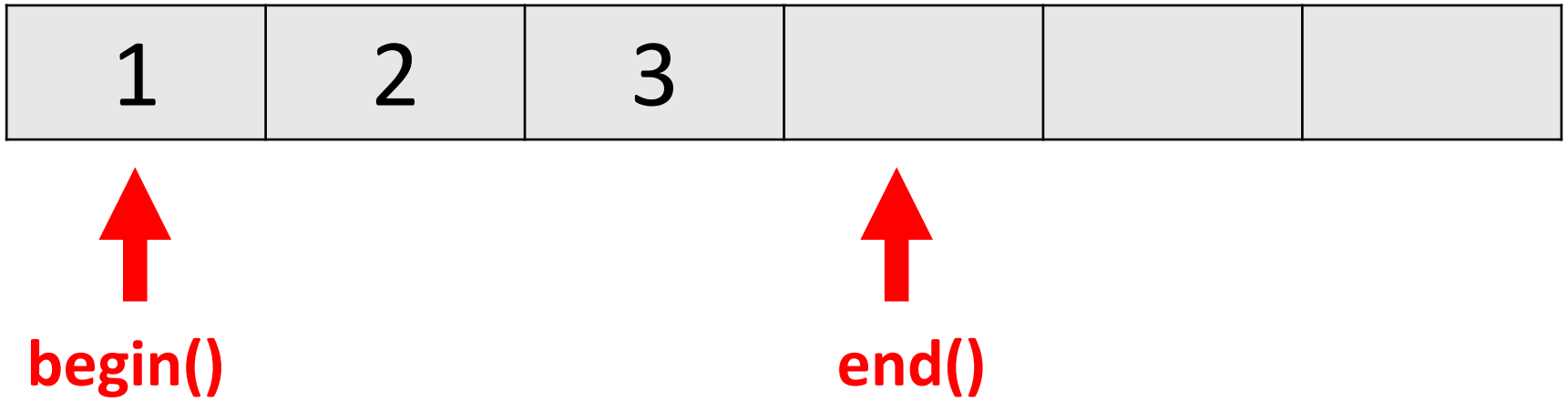
- 讓使用者可以逐個存取容器(Container) 中元素的工具
- 所有 STL 裡的 container 都有Iterator
  - 可利用 container::iterator 宣告
- Container 都有 begin( )、end( ) 函式
  - 回傳容器的第一個元素
  - 回傳最後一個元素後一個位置的記憶體位址

Iterator(迭代器)

- Container 都有 begin( )、end( ) 函式
  - 回傳容器的第一個元素
  - 回傳最後一個元素後一個位置的記憶體位址

原則：

前閉後開



Iterator(迭代器)

- Iterator 使用上像指標
  - iter++：檢索下一個元素
  - iter--：檢索上一個元素
  - *iter：取出該 iterator 指到的元素

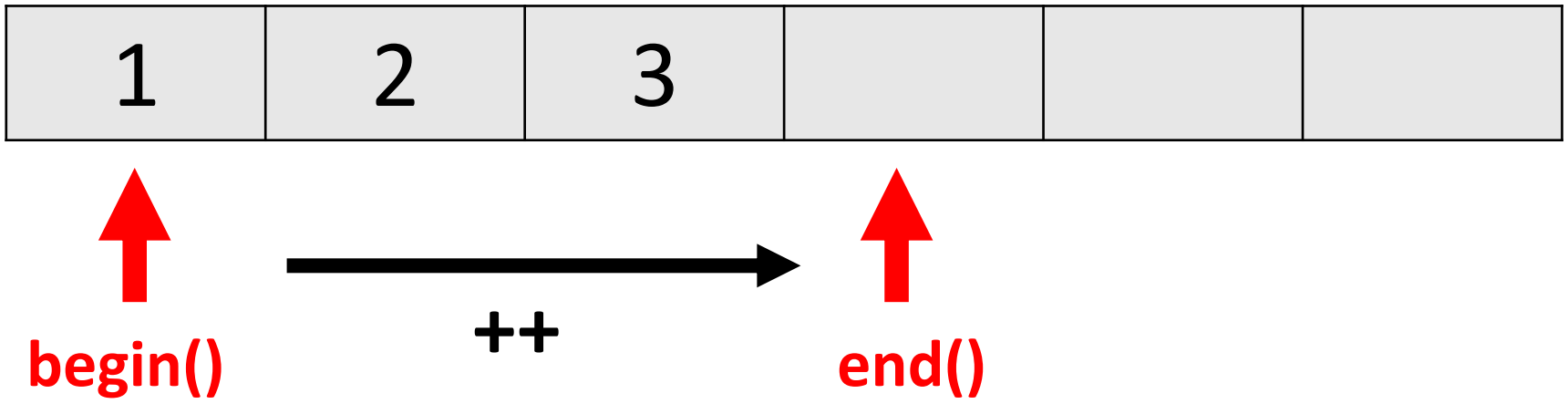
可利用 iterator 依序取出所有元素

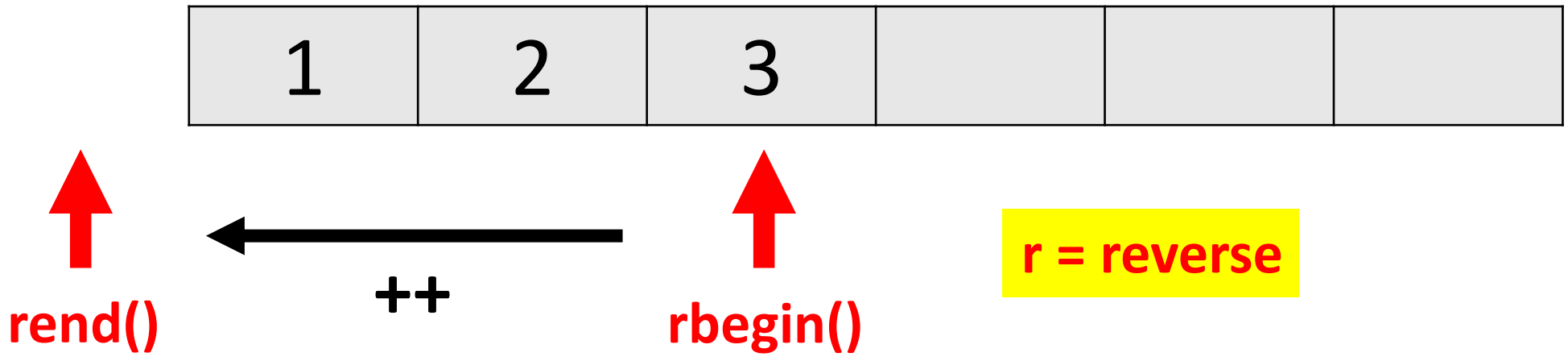
```
for ( iter = container.begin(); iter != container.end(); iter++ )
{
        // codes
}
```

Iterator(迭代器)

原則：

前閉後開





Algorithm(演算法)

Algorithm(演算法)

對 container 內的資料進行常用的操作

1. 搜尋 (find)
2. 合併 (merge)
3. 反轉 (reverse)
4. 排序 (sort)
5. 計數 (count)
6. 刪除 (remove)
7. ...

Algorithm(演算法)

搜尋 (find)

find(起點, 終點, 值)

- 起點、終點都是 iterator
- 回傳 iterator
- 對container做搜尋，範圍從起點到終點

  這兩個迭代器決定，尋找特定值

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

Algorithm(演算法)

計算出現次數 (count)

count(起點, 終點, 值)

- 起點、終點都是 iterator
- 回傳個數 int
- 對container做搜尋，範圍從起點到終點

  這兩個迭代器決定，尋找特定值的個數

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

Algorithm(演算法)

搜尋資料列 (search)

search(搜尋起點,搜尋終點,目標起點,目標終點)

- 起點、終點都是 iterator
- 回傳 iterator
- 對container做搜尋，查看否一資料列是否出現在另一容器中

```
template<class T1, class T2>
T1 search(T1 first_1, T1 last_1, T2 first_2, T2 last_2);
```

Algorithm(演算法)

排序 (sort)

sort(起點,終點,函式指標)

- 起點、終點都是 iterator
- 回傳 void
- 函式可以為空，預設由小到大
  1. greater：由大到小
  2. less：由小到大 (預設)
- 對 container 裡的資料做排序

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

```
template<class T1, class T2>
void sort(T1 first, T1 last, T2 pointer);
```

Algorithm(演算法)

依序處理 (for_each)

for_each(起點,終點,函式)

- 起點、終點都是 iterator
- 回傳函式指標
- 把 container 裡的資料依序丟入函式中

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