9-3 走訪 vector

走訪 vector

1. 方法一：下標 for 迴圈

最直覺的方式，適合需要使用索引時：

```
vector<int> v = {10, 20, 30, 40, 50};

for (int i = 0; i < v.size(); i++) {
    cout << v[i] << " ";
}
// 輸出：10 20 30 40 50
```

> ⚠ v.size() 回傳型別為 size_t（無號整數），若用 int i = v.size() - 1 倒著走時要小心，空 vector 的 0 - 1 會變成一個很大的數。

2. 方法二：Range-based for（範圍式 for）

C++11 起支援，最簡潔：

```
// 唯讀走訪
for (int x : v) {
    cout << x << " ";
}

// 修改元素（加上 &）
for (int& x : v) {
    x *= 2;   // 每個元素乘以 2
}
```

3. 方法三：迭代器（Iterator）

迭代器的概念類似**指標**，這也是為什麼熟悉指標的你能很快上手。

圖解：迭代器與指標的對比

```
傳統陣列指標：
  int arr[] = {10, 20, 30};
  int* ptr = arr;       // ptr 指向第一個元素

  ┌────┬────┬────┐
  │ 10 │ 20 │ 30 │
  └────┴────┴────┘
    ↑
   ptr

  ptr++;                // 指向下一個
  *(ptr) = 99;          // 解參考後修改

vector 迭代器：
  vector<int> v = {10, 20, 30};
  auto it = v.begin();  // it 指向第一個元素

  ┌────┬────┬────┬───────┐
  │ 10 │ 20 │ 30 │(end)  │
  └────┴────┴────┴───────┘
    ↑                ↑
  begin()          end()（指向最後一個之後）

  it++;             // 指向下一個（同 ptr++）
```

```
    *it = 99;          // 解參考後修改（同 *ptr）
```

迭代器的基本用法

```
vector<int> v = {10, 20, 30, 40, 50};

// 傳統的寫法，宣告 iterator 的型別有點複雜
// vector<int>::iterator it = v.begin();

// 自 C++ 11 起可以用 auto 自動推導型別
auto it = v.begin();    // 指向第一個元素

while (it != v.end()) {
    cout << *it << " ";  // 解參考取值
    it++;                // 移動到下一個
}
// 輸出：10 20 30 40 50
```

更常見的寫法（for 迴圈）：

```
for (auto it = v.begin(); it != v.end(); it++) {
    cout << *it << " ";
}
```

三種走訪方式比較

方式	優點	缺點
下標 `v[i]`	直覺，可使用索引	需要注意型別
range-based for	最簡潔	無法直接取得索引
迭代器	與 STL 演算法相容	語法略繁瑣

> ✅ 實務建議：平常用 range-based for，需要插入/刪除時用迭代器，需要索引時用下標。

Revision #1
★Created 26 April 2026 12:22:46 by huihui
Updated 26 April 2026 12:36:38 by huihui