9-4 常用成員函數

其他常用成員函數

1. front() 與 back()

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

cout << v.front();  // 10：第一個元素
cout << v.back();   // 50：最後一個元素

v.front() = 99;     // 可以修改
v.back()  = 1;
// v 現在是 {99, 20, 30, 40, 1}
```

2. insert()：在指定位置插入

```
vector<int> v = {1, 2, 3, 4, 5};

// insert(位置迭代器, 值)
auto it = v.begin() + 2;   // 指向索引 2（值為 3）
v.insert(it, 99);

// v 現在是 {1, 2, 99, 3, 4, 5}
```

圖解：

```
插入前：
┌───┬───┬───┬───┬───┐
│ 1 │ 2 │ 3 │ 4 │ 5 │
└───┴───┴───┴───┴───┘
          ↑
        it（索引 2）

插入後（後面的元素全部後移）：
┌───┬───┬────┬───┬───┬───┐
│ 1 │ 2 │ 99 │ 3 │ 4 │ 5 │
└───┴───┴────┴───┴───┴───┘
```

⚠ insert() 後，原本的迭代器可能**失效**，請重新取得。

3. erase()：移除指定位置的元素

```
vector<int> v = {1, 2, 99, 3, 4, 5};

// 移除單一元素
v.erase(v.begin() + 2);   // 移除索引 2（值 99）
// v 現在是 {1, 2, 3, 4, 5}

// 移除一個範圍 [begin+1, begin+3)（左閉右開）
v.erase(v.begin() + 1, v.begin() + 3);
// 移除索引 1 和 2（值 2 和 3）
// v 現在是 {1, 4, 5}
```

4. resize() 與 reserve()

這兩個函數常被搞混，用下圖區分：

resize(n)：改變 size（元素個數）

```
vector<int> v = {1, 2, 3};  size=3, capacity=3
```

```
v.resize(5);
→ {1, 2, 3, 0, 0}           size=5, capacity=5（新元素填 0）

v.resize(2);
→ {1, 2}                    size=2, capacity=5（縮小 size，不影響 capacity）
```

reserve(n)：改變 capacity（預留記憶體）

```
vector<int> v = {1, 2, 3};  size=3, capacity=3

v.reserve(10);
→ {1, 2, 3}                 size=3, capacity=10（只預留空間，不新增元素）
```

何時用 reserve？

當你事先知道大概會新增多少元素時，使用 `reserve` 可以避免多次重新分配記憶體，提升效能：

```
vector<int> v;
v.reserve(1000);          // 預先分配 1000 個空間

for (int i = 0; i < 1000; i++) {
    v.push_back(i);       // 不會觸發記憶體重新分配
}
```

5. capacity()：查詢已分配空間

```
vector<int> v;
cout << v.size()     << endl;   // 0
cout << v.capacity() << endl;   // 0（或平台相關的初始值）

v.push_back(1);
cout << v.capacity() << endl;   // 1（或更大，依平台）

v.push_back(2);
cout << v.capacity() << endl;   // 2（擴張）

v.push_back(3);
cout << v.capacity() << endl;   // 4（擴張為倍數）
```

6. 二維 vector

`vector` 可以嵌套，用來表示矩陣或表格：

```
// 宣告 3x4 的二維 vector，初始值全為 0
vector<vector<int>> matrix(3, vector<int>(4, 0));
```

圖解：

```
matrix：
        [0]  [1]  [2]  [3]
[0] ──► ┌────┬────┬────┬────┐
        │ 0  │ 0  │ 0  │ 0  │
        └────┴────┴────┴────┘
[1] ──► ┌────┬────┬────┬────┐
        │ 0  │ 0  │ 0  │ 0  │
        └────┴────┴────┴────┘
[2] ──► ┌────┬────┬────┬────┐
        │ 0  │ 0  │ 0  │ 0  │
        └────┴────┴────┴────┘
```

```
// 存取與修改
matrix[1][2] = 99;

// 走訪二維 vector
```

```
for (int i = 0; i < matrix.size(); i++) {
    for (int j = 0; j < matrix[i].size(); j++) {
        cout << matrix[i][j] << " ";
    }
    cout << endl;
}

// 使用 range-based for
for (auto& row : matrix) {
    for (int val : row) {
        cout << val << " ";
    }
    cout << endl;
}
```

注意：二維 vector 的每一列長度可以不同（鋸齒狀陣列），這與傳統二維陣列不同。

Revision #4
★Created 26 April 2026 12:37:08 by huihui
Updated 27 April 2026 07:05:27 by huihui