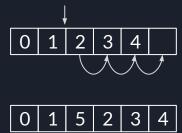
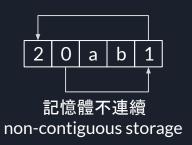
14. vector

記憶體連續性

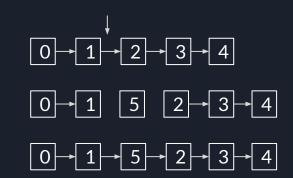


- 直接/隨機存取 (direct/random access)
- 非尾部新增/刪除慢





- 順序存取 (direct/random access)
- 新增/刪除快



std::vector

vector是一個記憶體連續、動態的容器

可以(理想)無限擴張、快速存取資料



std::vector

```
vector<T> variable_name; // (不用給大小 因為可以一直擴大)
vec.empty() -> 空true 或 有內容false
vec.size() -> 回傳內容物的數量
vec.push back(new value) -> 在最後面新增值
vec.pop back() -> 移除最後值
vec.insert(iterator) -> 在iterator插入一個值
vec.front() -> 回傳 &vec[0]
vec.back() -> 回傳 &vec[vec.size() - 1]
vec.begin() -> 回傳 第一個值的iterator
vec.end() -> 回傳 最後一個值的下一個位置的iterator
vec[n]
```

std::vector constructor

STL 容器有一個共通的特點 都有建構子(constructor)

使用內建的建構子可以省去時間、 並讓程式更整齊

```
vector<int> v2(30); // fill (0) (這個不是大小限制, 而是fill的數量)
vector<int> v3(30, -1); // fill (with value)
vector<int> v4(v3.begin() + 2, v3.end() - 2); // range
vector<int> v5(v3); // copy
// C++11: move, initializer list
size(): 0
v2
size(): 30
size(): 30
v4
size(): 26
v5
size(): 30
```

vector<int> v1; // default

常用std功能

```
std::find(iterator_first, iterator_last, value)
在[iterator_first, iterator_last)的範圍中尋找value
找到的話回傳第一個找到位置iterator;找不到回傳iterator_last
find(vec.begin(), vec.end(), value) 會在vec中找value
std::sort(iterator_first, iterator_end)
將[iterator_first, iterator_last)排序整齊
std::reverse(iterator_first, iterator_end)
將[iterator_first, iterator_last)反轉
```