17. map

2024資訊研究社 語法班
Made with ❤️ by jheanlee

pair

pair 顧名思義，是在儲存一對資料(通常互相對應)

```
pair<Type1, Type2> p(val1, val2);
                   pair.first   pair.second
```

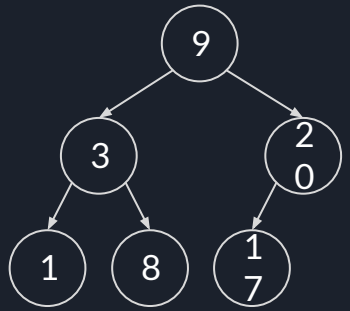
實例：產品-價格、編號-姓名

map

map 是一種有序的關聯型容器，常被用在依key排序、對應值查找

map 的本質是以pair為單位的 二元搜尋樹(binary search tree)

中文常見的翻譯有映射表、對映表、鏈值對容器



二元搜尋樹

arr[5]

固定大小
fixed-size

動態
dynamically allocated

map的特性

有序：map中所有項目都會被依key排序

對應性：map的任何項目都有對應的key

key的單一性：map中的key不會重複（但項目可以重複)

```
map<int, string> m;
m.insert({1, "one"});
m.insert(make_pair(2, "two"));
m.emplace(4, "four");
m.insert(pair<int, string> (3, "three"));
m[5] = "five";
m.insert({2, "two_"});  //  key 重複

cout << "m: \n";
for (const pair<const int, string> &p: m) {
  cout << "  {" << p.first << ", " << p.second << "}\n";
}
cout << '\n';
```

```
m:
  {1, one}
  {2, two}
  {3, three}
  {4, four}
  {5, five}
```

map的實例

座號 (int)	姓名 (string)
1	Special
2	R:y
3	jnl

```
std::map<int, string> number_to_name
```

項目(string)	價格(int)
水	10
大瓶水	20
進口貴水	50

```
std::map<int, string> item_to_price
```

map 在某些語言中稱作dictionary

map 的子函式

map<T, U> variable_name; -> 建構子

map.empty() -> 空true 或 有內容false
map.size() -> 回傳內容物的數量

map.insert(val) -> 插入一個值 (val的type為pair<T, U>)
map.erase(key) -> 移除一個值(依照key)

map.find(key) -> 尋找值
找到的話回傳第一個找到位置iterator;找不到回傳map.end()
map.count(key) -> 回傳中key的數量 (因為單一性所以只可能回傳0和1)

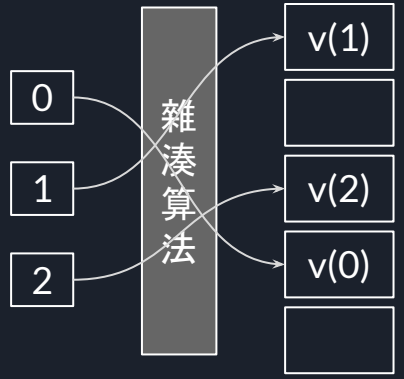
map.begin() -> 回傳第一個值的iterator
map.end() -> 回傳最後一個值的下一個位置的iterator

map[key] -> map中key所對應的值

unordered_map

unordered_map 是一種無序的關聯型容器，常被用於高效率的對應值查找

unordered_map的本質是以key進行雜湊運算的 雜湊表(hash table)



雜湊表

固定大小
fixed-size

動態
dynamically allocated

unordered_map的特性

無序：unordered_map中所有項目會以雜湊算法的值分類而失去順序性

對應性：unordered_map的任何項目都有對應的key

key的單一性：unordered_map中的key不會重複（但項目可以重複）

```cpp
unordered_map<int, string> um;
um.insert({1, "one"});
um.insert(make_pair(2, "two"));
um.emplace(4, "four");
um.insert(pair<int, string> (3, "three"));
um[5] = "five";
um.insert({2, "two_"});  //  key 重複

cout << "um: \n";
for (const pair<const int, string> &p: um) {
  cout << "  {" << p.first << ", " << p.second << "}\n";
}
cout << '\n';
```

```
um:
  {5, five}
  {3, three}
  {4, four}
  {2, two}
  {1, one}
```

unordered_map 的子函式

unordered_map<T, U> variable_name; -> 建構子

unordered_map.empty() -> 空true 或 有內容false
unordered_map.size() -> 回傳內容物的數量

unordered_map.insert(val) -> 插入一個值 (val的type為pair<T, U>)
unordered_map.erase(key) -> 移除一個值(依照key)

unordered_map.find(key) -> 尋找值
找到的話回傳第一個找到位置iterator;找不到回傳unordered_map.end()
unordered_map.count(key) -> 回傳中key的數量 (因為單一性所以只可能回傳0和1)

unordered_map.begin() -> 回傳第一個值的iterator
unordered_map.end() -> 回傳最後一個值的下一個位置的iterator

unordered_map[key] -> unordered_map中key所對應的值