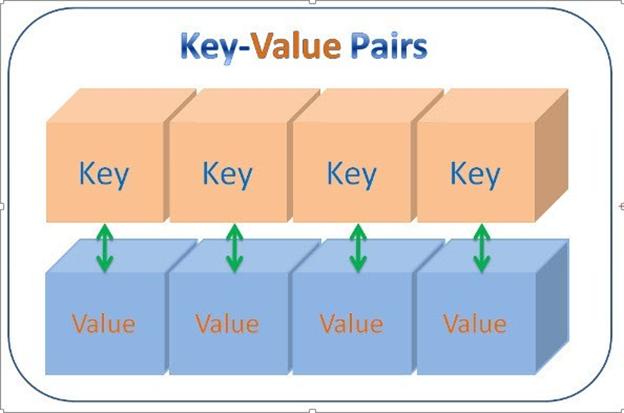
**Map là gì?**

Trong 1 số ngôn ngữ lập trình, Map được gọi là Dictionary (như Python hay C#). Trong khuôn khổ bài viết này, mình dùng từ map do thông thạo với C++ và Java.

Các cấu trúc dữ liệu như mảng hay xâu kí tự, khi truy xuất dữ liệu bạn sẽ sử dụng một tham số gọi là chỉ số, ví dụ như arr[1], str[2], … Đối với cấu trúc dữ liệu map, để truy xuất dữ liệu bạn sẽ sử dụng một tham số gọi là key

Cấu trúc dữ liệu kiểu map là một cấu trúc dữ liệu ánh xạ giữa cái gọi là khoá (key) sang giá trị của khoá đó (gọi là value)

Trong cấu trúc dữ liệu này, mỗi một key sẽ nhận một giá trị khác nhau.



**Ứng dụng**

Ứng dụng của map có rất nhiều. Mình đưa ra 1 số bài toán cơ bản nhé:

* Cho một danh sách các số điện thoại kèm theo tên của chủ thuê bao đó. Yêu cầu đầu vào là một số điện thoại (key), hãy đưa ra tên của chủ thuê bao (value)
* Cho danh sách thể hiện lịch sử đi muộn của các nhân viên một công ty nào đó. Hãy tìm xem nhân viên (key) nào có số lần đi muộn (value) nhiều nhất?
* Cho một danh sách các IP kèm theo các domain. Hãy trả ra ip (value) tương ứng domain (key) hoặc ngược lại trong thời gian nhanh nhất?

Với các bài toán này, bạn có thể sử dụng 2 mảng dữ liệu có cùng độ dài. Một mảng bạn lưu danh sách key, một mảng bạn lưu danh sách các value tương ứng với key đó (cùng chỉ số truy cập mảng). Khi bạn cần tìm value của một key nào đó, bạn duyệt mảng key, tìm chỉ số của phần tử trong mảng có giá trị bằng key cần tìm rồi trả ra giá trị tương ứng ở mảng value. Mã giả của đoạn code này như sau:

for (int i=0; i<keysArray.size(); i++)    if (keysArray[i] == key) return valuesArray[i];return "notFound";

Với đoạn mã như trên, độ phức tạp thuật toán sẽ là O(N) (do bạn phải duyệt cả mảng keysArray để tìm cái key bạn muốn). Để tăng tốc độ cho đoạn mã này, bạn có thể cải tiến hàm tìm kiếm theo phương pháp tìm kiếm nhị phân (với điều kiện mảng keysArray đã được sắp xếp). Tất nhiên cách làm này sẽ tốn công cài đặt hơn rất nhiều lần so với việc bạn sử dụng map (hay dictionary) thông qua dòng lệnh duy nhất sau:

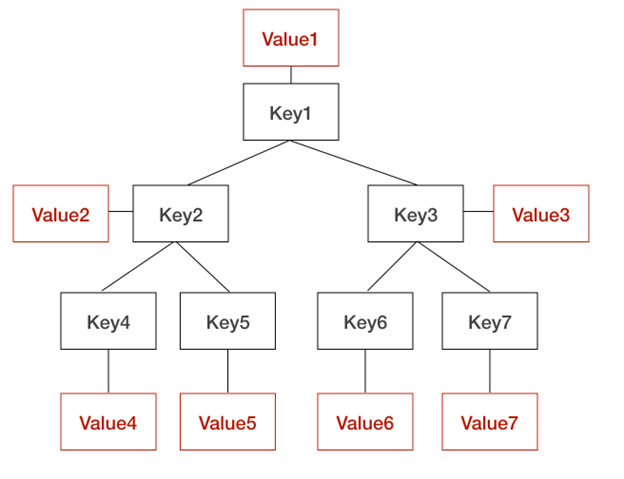
if (mymap.find(key) != mymap.end())    return mymap[key];else return "notFound";

Không chỉ là dễ cài đặt và sử dụng, việc sử dụng map còn đáp ứng tốc độ chạy chương trình khá cao (có chút khác biệt giữa 1 vài cách cài đặt map của các ngôn ngữ hoặc thư viện, nhưng chắc chắn là ngon hơn mình tự cài rồi :D)

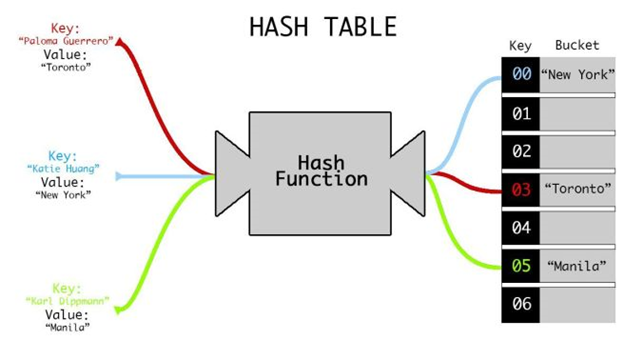
**Các loại map và đặc điểm:**

Nhìn chung có 2 loại map, dựa vào cách cài đặt của map. Một loại là map được cài đặt dựa trên cây đỏ đen (red-black tree), một loại là map được cài đặt dựa trên bảng băm (hashtable)

Với C++, map là loại map được cài đặt dựa trên cây đỏ đen, còn unordered\_map là loại map được cài đặt dựa trên nguyên lý Hash. Với Java, TreeMap là loại map được cài đặt bởi cây, còn HashMap là loại map đc cài đặt bởi bảng băm (hash table)

1. **Tree map**:  
   Map được cài bằng cây đỏ đen. Mỗi một node trong cây có một key và một value, trỏ vào 2 node bên trái và bên phải. Giá trị key của node bên trái nhỏ hơn giá trị key ở node cha và nhỏ hơn giá trị key ở node bên phải.  
   Độ phức tạp thuật toán của các phép toán thêm một node, lấy ra giá trị của một key là O(logn)
2. **HashMap**

Map được cài đặt dựa trên nguyên lý **Hashing** – băm. Để hiểu về **Hashing**, chúng ta cần nắm được 3 khái niệm: **Hash** **function**, **hash** **value** và **bucket**.  
  
**Hash function**, hay còn gọi là hàm băm, là một hàm mà khi ta lấy đầu vào là một giá trị bất kỳ thì ở đầu ra, **hash fuction** sẽ cho ta một dãy code – được gọi là **hash value**. Mỗi đầu vào chỉ có **duy nhất** một hash value.



**Bucket** là nơi mà chúng ta lưu trữ các cặp key-value. Độ phức tạp thuật toán phép thêm và lấy dữ liệu là O(1). Bạn có thể tìm hiểu và nghiên cứu sự khác nhau giữa 2 loại map này ở 1 số link như:

<https://www.geeksforgeeks.org/map-vs-unordered_map-c/>

hay

<https://stackoverflow.com/questions/2444359/what-is-the-difference-between-a-hashmap-and-a-treemap>

**Cách sử dụng:**

Đa phần map có một số phương thức cơ bản và cách sử dụng tương tự nhau ở 1 số ngôn ngữ:

Khai báo: **map<type\_of\_keys, type\_of\_values>** (với C++) hoặc **Dictionary<type\_of\_keys, type\_of\_values>** với C# hoặc **TreeMap**<**type\_of\_keys, type\_of\_values**> trong Java

* Thêm một cặp key/value: dùng phương thức **insert(key, value)**hoặc **map[key] = value**
* Truy cập: dùng **map[key]**
* Kiểm tra map có chứa key nào đó không: dùng **find(key)** (với C++) hoặc **map.containsKey**(với C#)

**Cách sử dụng map**

Để đưa dữ liệu dạng **Map** vào Firebase Realtime Database trong một dự án Android, bạn có thể sử dụng phương thức **setValue()** trên một tham chiếu của cơ sở dữ liệu Firebase. Dưới đây là cách thực hiện điều đó:

1. **Tạo một tham chiếu đến nút trong cơ sở dữ liệu Firebase**:
   * Đầu tiên, bạn cần có một tham chiếu đến nút trong cơ sở dữ liệu Firebase mà bạn muốn thêm dữ liệu. Bạn có thể làm điều này bằng cách sử dụng phương thức **getReference()** từ **FirebaseDatabase** và chỉ định đường dẫn của nút. Ví dụ:

java

Copy code

DatabaseReference databaseRef = FirebaseDatabase.getInstance().getReference().child("ten\_nut\_trong\_co\_so\_du\_lieu");FirebaseDatabase.getInstance().getReference().child();

1. **Tạo một đối tượng dạng Map chứa dữ liệu**:
   * Tiếp theo, bạn cần tạo một đối tượng dạng **Map** chứa các cặp key-value mà bạn muốn thêm vào cơ sở dữ liệu. Ví dụ:

java

Map<String, Object> dataMap = new HashMap<>();

dataMap.put("key1", "value1");

dataMap.put("key2", "value2");

Map<String, Map<String, Object> dataMap = new HashMap<>();

dataMap.put("key1", "value1");

dataMap.put("key2", "value2");); dataMap.put(

1. **Đưa dữ liệu vào cơ sở dữ liệu Firebase**:
   * Cuối cùng, sử dụng phương thức **setValue()** trên tham chiếu của bạn để đưa dữ liệu vào cơ sở dữ liệu Firebase. Ví dụ:

databaseRef.setValue(dataMap);

databaseRef.setVdatabaseRef.setValue(dataMap);alue(dataMap);

Như vậy, dữ liệu trong **dataMap** sẽ được đưa vào nút mà bạn đã chỉ định trong cơ sở dữ liệu Firebase.

Lưu ý rằng Firebase Realtime Database sẽ thực hiện việc ghi dữ liệu bất đồng bộ, điều này có nghĩa là việc gọi **setValue()** sẽ không chờ đợi dữ liệu được ghi thành công. Để biết khi nào dữ liệu đã được ghi thành công hoặc xử lý lỗi, bạn có thể sử dụng các phương thức lắng nghe sự kiện.

Top of Form

Ở đây, "nút" là một phần của cấu trúc cơ sở dữ liệu Firebase. Các dữ liệu trong Firebase Realtime Database được tổ chức dưới dạng cây, trong đó mỗi "nút" là một cặp key-value, và mỗi key đại diện cho một nút và có thể chứa các nút con và/hoặc dữ liệu.Bottom of Form