Menu DevSOLVD

search...

Search...

[SOLVD] map vs. hash_map in C++

I have a question with `hash_map` and `map` in C++. I understand that `map` is in STL, but `hash_map` is not a standard. What's the difference between the two?

by skydoor
source

- c++
- map
- hashmap

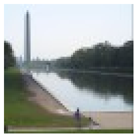
Best Answer86
Source

They are implemented in very different ways.

`hash_map` (`unordered_map` in TR1 and Boost; use those instead) use a hash table where the key is hashed to a slot in the table and the value is stored in a list tied to that key.

`map` is implemented as a balanced binary search tree (usually a red/black tree).

An `unordered_map` should give slightly better performance for accessing known elements of the collection, but a `map` will have additional useful characteristics (e.g. it is stored in sorted order, which allows traversal from start to finish). `unordered_map` will be faster on insert and delete than a `map`.

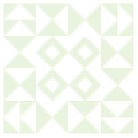


by Joe

- @ email
- facebook
- linkedin
- twitter
- reddit
- google+

28
Source

`hash_map` was a common extension provided by many library implementations. That is exactly why it was renamed to `unordered_map` when it was added to the C++ standard as part of TR1. map is generally implemented with a balanced binary tree like a red-black tree (implementations vary of course). `hash_map` and `unordered_map` are generally implemented with hash tables. Thus the order is not maintained. `unordered_map` insert/delete/query will be O(1) (constant time) where map will be O(log n) where n is the number of items in the data structure. So `unordered_map` is faster, and if you don't care about the order of the items should be preferred over `map`. Sometimes you want to maintain order (ordered by the key) and for that `map` would be the choice.



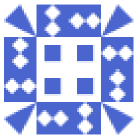
by janglin
11
Source

Some of the key differences are in the complexity requirements.

A map requires O(log(N)) time for inserts and finds.

An unordered_map requires an 'average' time of O(1) for inserts and finds but is allowed to have a worst case time of O(N).

So, usually, unordered_map will be faster, but depending on the keys and the hash function you store, can become much worse.



by R Samuel Klatchko
4
Source

The C++ spec doesn't say exactly what algorithm you must use for the STL containers. It does, however, put certain constraints on their performance, which rules out the use of hash tables for `map` and other associative containers. (They're most commonly implemented with red/black trees.) These constraints require better worst-case performance for these containers than hash tables can deliver.

Many people really do want hash tables, however, so hash-based STL associative containers have been a common extension for years. Consequently, they added `unordered_map` and such to later versions of the C++ standard.

by Warren Young

Popular tags

java c# javascript android python jquery .net php html c++ css iphone ios objective-c git sql ruby-on-rails mysql string ruby c asp.net sql-server linux eclipse asp.net-mvc angularjs arrays bash windows