sign up log in tour help stack overflow careers

Stack Overflow is a question and answer site for professional and enthusiast programmers. It's 100% free. Take the 2-minute tour ×

Do STL iterators guarantee validity after collection was changed?

Let's say I have some kind of collection and I obtained an iterator for the beginning of it. Now let's say I modified the collection. Can I still use the iterator safely, regardless of the type of the collection or the iterator?

To avoid confusion, here is the order of operations I talk about:

1. Get an iterator of the collection.
2. Modify the collection (obviously not an element in it, but the collection itself).
3. Use the iterator obtained at step 1. Is it stil valid according to STL standard?!

c++ stl iterator containers

edited Feb 8 '14 at 14:38
Midas
3,440 3 14 37

asked Jul 25 '10 at 16:17
user88637
3,088 4 22 31

Thank you all for fast responses. – user88637 Jul 25 '10 at 16:37

3 Answers

Depends on the container. e.g. if it's a `vector` , after modifying the container all iterators can be invalidated. However, if it's a `list` , the iterators irrelevant to the modified place will remain valid.

- A vector's iterators are invalidated when its memory is reallocated. Additionally, inserting or deleting an element in the middle of a vector invalidates all iterators that point to elements following the insertion or deletion point. It follows that you can prevent a vector's iterators from being invalidated if you use `reserve()` to preallocate as much memory as the vector will ever use, and if all insertions and deletions are at the vector's end. [1]
- The semantics of iterator invalidation for `deque` is as follows. `Insert` (including `push_front` and `push_back`) invalidates all iterators that refer to a `deque`. `Erase` in the middle of a `deque` invalidates all iterators that refer to the `deque`. `Erase` at the beginning or end of a `deque` (including `pop_front` and `pop_back`) invalidates an iterator only if it points to the erased element. [2]
- `List` s have the important property that insertion and splicing do not invalidate iterators to list elements, and that even removal invalidates only the iterators that point to the elements that are removed. [3]
- `Map` has the important property that inserting a new element into a `map` does not invalidate iterators that point to existing elements. Erasing an element from a map also does not invalidate any iterators, except, of course, for iterators that actually point to the element that is being erased. [4] (same for `set`, `multiset` and `multimap`)

edited Jul 25 '10 at 16:29

answered Jul 25 '10 at 16:19
kennytm
270k 49 663 744

1 However, `vector::insert` (single element) and `vector::erase`, returns a new, valid, iterator. – Viktor Sehr Jul 25 '10 at 16:29

1 Kenny : Is what you say true according to the standard or true because this is how stl usually implemented ? Can you modify your answer so it also explain it? thanks. – user88637 Jul 25 '10 at 16:29

8 @yossi1981: This is defined by the standard. The conditions under which iterators are invalidated are very explicitly defined in the standard. – Loki Astari Jul 25 '10 at 16:37

That depends on the collection in question. Just for example, modifying a `std::vector` (e.g., adding an element somewhere) can invalidate all iterators into that vector. By contrast, with a `std::list` , iterators remain valid when you add another element to the list. In some cases, the rules are even more complex (e.g., if memory serves, with a `std::deque` , adding to the beginning or end leaves existing iterators valid, but adding anywhere else can invalidate them -- but my memory is sufficiently poor that you should check before depending on that).

answered Jul 25 '10 at 16:22

Jerry Coffin
277k 27 285 623

2 what if you erase a list element? What if it's the element the iterator points to? – m1tk4 Jul 25 '10 at 16:25

Erasing the element referred to by an iterator invalidates that iterator. Erasing any other element(s) does not. – Jerry Coffin Jul 25 '10 at 16:28

In other words, if you modify the list, there are circumstances when it will invalidate the iterator (pointing to erased element), so the iterator validity is not guaranteed - correct? – m1tk4 Jul 25 '10 at 16:34

No, iterators are only good while the iterated container is unchanged. If a collection is modified, the iterator should be obtained anew.

answered Jul 25 '10 at 16:18

m1tk4
2,493 8 23

1 Strictly speaking that is not true (example list does not follow those rules). – Loki Astari Jul 25 '10 at 16:38

1 The question was asked "regardless of the type of the collection(container)". – m1tk4 Jul 25 '10 at 16:57