home > topics > c / c++ > questions > sorting a map by value

Sorting a map by value

Kevin W.

P: n/a

How do I sort a map by the value, rather than the key? (either automatically or with the sort function.)

--
Kevin W :-)
Opera/CSS/webdev blog: http://www.exclipy.com/
Using Opera: http://www.opera.com/m2/

Jul 22 '05 #1

Post Reply

Share this Question

7 Replies

Question stats

- viewed: 9050
- replies: 7
- date asked: Jul 22 '05

Follow this discussion

Similar topics

- Jquery table sorting help
- Searching Sorting Exit
- Sorting on GridView
- DataGrigView Sorting Problem - Experts HELP Please
- Collapsible table sorting problem
- ListView Sorting by Column
- Sorting A Collection
- array sorting with blank spaces: do IE and Mozilla handle this differently?
- Data Grid NOT SORTING
- Sorting Driving Crazy: URGENT: PLEASE HELP

Rolf Magnus

P: n/a

Kevin W. wrote:

> How do I sort a map by the value, rather than the key? (either automatically or with the sort function.)

You can't sort maps. They are always automatically sorted by key.

Jul 22 '05 #2

tom_usenet

P: n/a

On Tue, 24 Aug 2004 07:55:19 GMT, "Kevin W." <contact@in.sig> wrote:

> How do I sort a map by the value, rather than the key? (either automatically or with the sort function.)

You can't - std::map has an invariant that it is sorted by key. You'll have to copy into a vector or similar first, or perhaps maintain two parallel maps (key->value and value->key).

Tom

Jul 22 '05 #3

Daniel T.

P: n/a

"Kevin W." <contact@in.sig> wrote:

> How do I sort a map by the value, rather than the key? (either automatically or with the sort function.)

First start with a test:

```
int main() {
map< char, int > current;
current['a'] = 5;
current['b'] = 4;
current['c'] = 3;

map< int, char > other = converse_map( current );

map< int, char >::iterator begin( other.begin() );
assert( begin->first == 3 );
assert( begin->second == 'c' );
++begin;
assert( begin->first == 4 );
assert( begin->second == 'b' );
++begin;
assert( begin->first == 5 );
assert( begin->second == 'a' );
cout << "OK";
}
```

When the above prints "OK" you know you are done. Now write the 'converse_map' function...

```
map< int, char > converse_map( const map< char, int >& o )
{
map< int, char > result;
for ( map< char, int >::const_iterator begin( o.begin() );
begin != o.end(); ++begin )
result.insert( make_pair( begin->second, begin->first ) );
return result;
}
```

Then turn it into a template...

```
template < typename T, typename U >
map< U, T > converse_map( const map< T, U >& o )
{
map< U, T > result;
for ( typename map< T, U >::const_iterator begin( o.begin() );
begin != o.end(); ++begin )
result.insert( make_pair( begin->second, begin->first ) );
return result;
}
```

Jul 22 '05 #4

Joaquín Mª López Muñoz

P: n/a

tom_usenet ha escrito:

> On Tue, 24 Aug 2004 07:55:19 GMT, "Kevin W." <contact@in.sig> wrote:
>
> > How do I sort a map by the value, rather than the key? (either
> > automatically or with the sort function.)
>
> You can't - std::map has an invariant that it is sorted by key. You'll
> have to copy into a vector or similar first, or perhaps maintain two
> parallel maps (key->value and value->key).

Boost.MultiIndex (to appear promptly in Boost 1.32, online docs
already available at boost-consulting.com/boost/libs/multi_index) can
be used to construct such a bidirectional map easily. This is shown in one

of the examples at

boost-consulting.com/boost/libs/multi_index/doc/examples.html#example4

Regards,

Joaquín M López Muñoz
Telefónica, Investigación y Desarrollo

Jul 22 '05 #5

Thomas Matthews

P: n/a

Kevin W. wrote:

> How do I sort a map by the value, rather than the key? (either
> automatically or with the sort function.)

The map data structure is an association between a key
and a value. The std::map requires that each key be

unique. However, two keys can have the same value.
How will you handle the collating of pairs with
the same value but different keys?

If you _really_ want to sort the map, what you are
saying is that you want the data in the map to
be sorted by value. Easy, copy the data into
a new std::multi_map but swap the key with the
value before placing into the multimap. Or
you could use a list, or vector and a sort
function.

Search the newsgroup for "view pattern". For
a hint on how to make a "view" of the data
without altering where the data is stored.

--
Thomas Matthews

C++ newsgroup welcome message:
http://www.slack.net/~shiva/welcome.txt
C++ Faq: http://www.parashift.com/c++-faq-lite
C Faq: http://www.eskimo.com/~scs/c-faq/top.html
alt.comp.lang.learn.c-c++ faq:
http://www.comeaucomputing.com/learn/faq/
Other sites:
http://www.josuttis.com -- C++ STL Library book

Jul 22 '05 #6

Kai-Uwe Bux

P: n/a

Kevin W. wrote:

> > > How do I sort a map by the value, rather than the key?
> > > You can't sort maps. They are always automatically sorted by
> > > key.
> >
> > Yes, but I should be able to define the "less than" predicate in the
> > constructor
>
> Yes, this less_than predicate will be used to compare the keys.
> or I should be able to pass a custom predicate into the sort
> function (as I understand it, you *can* sort a map because it is only
> automatically sorted when pairs are added or removed).

The std::map<Key,T> container is a sequence of pairs. The type of these
pairs is std::pair<const Key, T >. Please note the *const*. Because of this
'const', you cannot change the keys once they are inserted into the
sequence. It follows that you cannot feed a segment of a map into
std::sort(). The swaps that std::sort() wants to perform are barred by
constness of keys.
Best

Kai-Uwe Bux

Jul 22 '05 #7

Richard Herring

P: n/a

In message <opsc9yjw16jxvii7@localhost.localdomain>, Kevin W. <contact@in.sig> writes

> > > How do I sort a map by the value, rather than the key?
> > > You can't sort maps. They are always automatically sorted by key.
> >
> > Yes, but I should be able to define the "less than" predicate in the constructor,
>
> Yes, and it defines the ordering for the lifetime of the map. Once established, you can't change it.
> or I should be able to pass a custom predicate into the sort function
> No, because there _is_ no standard sort function which can be applied to maps. std::sort() needs random-access non-const iterators.
> (as I understand it, you *can* sort a map because it is only automatically sorted when pairs are added or removed).
> No. As Kai-Uwe Bux has pointed out, the key element of each pair is const. But even if you could rearrange their order, it wouldn't do you any good, because the key lookup algorithm depends on the elements being
> correctly ordered. The next time you tried to access an element by key, you'd get UB.
> What I'm really asking is:
> What are the arguments to this functor (values, pointers or references),
> The same as to std::less, namely (const reference to) key_type, and it returns bool. And it must implement a strict weak ordering:
>
> a<b && b<c => a<c;
> eq(a,b) && eq(b,c) => eq(a, c) where eq(a, b) means !(a<b) && !(b<a)
> and
> How do I access the value from these arguments?

The arguments _are_ the values (or references to them.)

--
Richard Herring

Jul 22 '05 #8

This discussion thread is closed

Replies have been disabled for this discussion.

Start new discussion

Browse more C / C++ Questions on Bytes