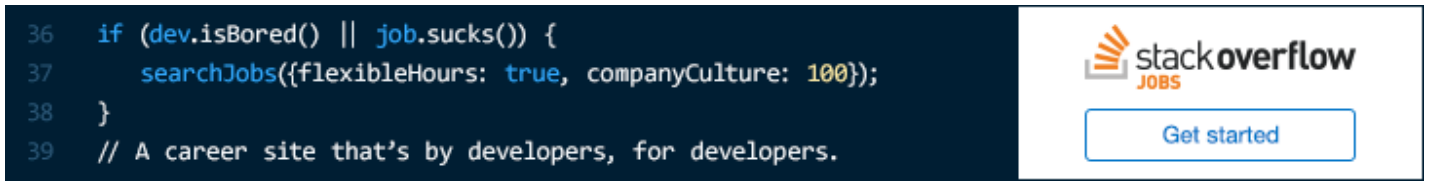
Iterating over a vector in reverse direction



I need to iterate over a vector from the end to the beginnig. The "correct" way is

```
for(std::vector<SomeT>::reverse_iterator rit = v.rbegin(); rit != v.rend(); ++rit)
{
    //do Something
}
```

When doSomething involves knowing the actual index, then some calculations need to be done with rit to obtain it, like `index = v.size() - 1 - (rit - v.rbegin)`

If the index is needed anyway, then I strongly believe it is better to iterate using that index

```
for(int i = v.size() - 1; i >= 0; --i)
{
    //do something with v[i] and i;
}
```

This gives a warning that `i` is signed and `v.size()` is unsigned. Changing to

`for(unsigned i = v.size() - 1; i >= 0; --i)` is just functionally wrong, because this is essentially an endless loop `:)`

What is an aesthetically good way to do what I want to do which

- is warning-free
- doesn't involve casts
- is not overly verbose

I hope I am not looking for something that doesn't exist :)

c++ stl iteration

edited Aug 30 '13 at 14:34

James Webster
25.5k 11 54 95

asked Nov 17 '10 at 15:06

Armen Tsirunyan
80.6k 39 231 359

loop condition `i != std::numeric_limits<unsigned>::max()` ... or use UINT_MAX if you think its to verbose. – smerlin Nov 17 '10 at 15:17

So far, I think doing a cast is looking like the cleanest solution :-) – David Gelhar Nov 17 '10 at 15:25

actually i agree with david – smerlin Nov 17 '10 at 15:26

just set an upper bound on i, i.e. v.size(). – Nim Nov 17 '10 at 15:29

3 `for (size_t i = v.size(); i --> 0; )` – UncleBens Nov 17 '10 at 16:36

9 Answers

As you've noted, the problem with a condition of `i >= 0` when it's unsigned is that the condition is always true. Instead of subtracting 1 when you initialize `i` and then again after

each iteration, subtract 1 after checking the loop condition:

```
for (unsigned i = v.size(); i-- > 0; )
```

I like this style for several reasons:

- Although `i` will wrap around to `UINT_MAX` at the end of the loop, it doesn't *rely* on that behavior — it would work the same if the types were signed. Relying on unsigned wraparound feels like a bit of a hack to me.
- It calls `size()` exactly once.
- It doesn't use `>=` . Whenever I see that operator in a `for` loop, I have to re-read it to make sure there isn't an off-by-one error.
- If you change the spacing in the conditional, you can make it use the "goes to" operator.

answered Nov 17 '10 at 16:40

Rob Kennedy
134k 15 203 365

+1 neat =), I guess there's always room for improvement! – Nim Nov 17 '10 at 17:12

2 shoulnd't it be size_t than unsigned ? – dynamic Jun 28 '13 at 8:45

1 That could be better if you really anticipate having more items than will fit in an `unsigned` , @Yes123, but the formally correct type would be `std::vector<SomeT>::size_type` , or, nowadays, simply `auto` . – Rob Kennedy Jun 28 '13 at 13:18

There's nothing to stop your `reverse_iterator` loop also using the index as described in multiple other answers. That way you can use the iterator or index as needed in the `// do the work` part, for minimal extra cost.

```
size_t index = v.size() - 1;
for(std::vector<SomeT>::reverse_iterator rit = v.rbegin();
    rit != v.rend(); ++rit, --index)
{
  // do the work
}
```

Though I'm curious to know what you need the index for. Accessing `v[index]` is the same as accessing `*rit` .

answered Nov 17 '10 at 15:43

Steve Townsend
43.2k 4 58 114

This is actually the only correct answer for any underlying type that's not random access. Assuming `std::vector<SomeT>` is typedef'd away somewhere I'd prefer to use the correct iterator in case I later decided to use a list, multiset, map, etc. – Ben Jackson Nov 17 '10 at 17:06

to be aesthetically pleasing! ;)

```
for(unsigned i = v.size() - 1; v.size() > i; --i)
```

edited Nov 17 '10 at 15:58

answered Nov 17 '10 at 15:25

Nim
27.7k 2 42 80

@Armen, hmm, but unsigned (by default int) *could be* smaller than the `size_type` of `vector` , as to why you could have more than 4bn entries in your vector, that's a different question... :).. okay, okay... I give in... – Nim Nov 17 '10 at 15:36

2 +1, that's quite clever Nim. – Moo-Juice Nov 17 '10 at 15:38

And what if v.size() == UINT_MAX ? I know this never happens in practice, but it is a theoric flaw – Jean-Bernard Jansen Nov 17 '10 at 15:43

@JB: vector::size_type is not unsigned int is also a theoretic flaw. But I honestly don't care about theoretic flaws as long as there is really no practical chance it's gonna backfire – Armen Tsirunyan 7 secs ago edit – Armen Tsirunyan Nov 17 '10 at 15:44

I don't know if it's me not using the wrapping around efficiently, but I usually do it with `for (i = v.size(); i > 0; --i)` and then use `i-1` as the actual index. Matter of preference I guess. – Matthieu M. Nov 17 '10 at 15:49

I would prefer the reverse iterator variant, because it's still easy to interpret and allows to avoid index-related errors.

Sometimes you can simply use the `BOOST_REVERSE_FOREACH`, which would make your code look the following way:

```
reverse_foreach (int value, vector) {
   do_something_with_the_value;
}
```

Actually speaking, you can always use `foreach` statements for these kinds of loops, but then they become a bit unobvious:

```
size_t i = 0;

foreach (int value, vector) {
   do_something;
   ++i;
}
```

answered Nov 17 '10 at 15:12

Yippie-Ki-Yay
6,617 14 68 124

Try out a do while :

```
std::vector<Type> v;
// Some code
if(v.size() > 0)
{
    unsigned int i = v.size() - 1;
    do
    {
```

```
        // Your stuff
    }
    while(i-- > 0);
}
```

answered Nov 17 '10 at 15:15

Jean-Bernard Jansen
3,533 1 13 15

Sorry, my bad.. – Fred Larson Nov 17 '10 at 15:21

Okay, no problem ;). I tested the do while and it works. I don't say it is the best solution, but it is one available. – Jean-Bernard Jansen Nov 17 '10 at 15:27

Hi i think better way use iterator as you use in first sample and if you need get iterator index you can use std::distance to calculate it, if i understand your question

answered Nov 17 '10 at 15:17

Sanja Melnichuk
2,838 3 17 41

loop condition `i != std::numeric_limits<unsigned>::max()` ... or use `UINT_MAX` if you think its to verbose. or another way:

```
for(unsigned j=0, end=v.size(), i=end-1; j<end; --i, ++j)
```

or

```
for(unsigned end=v.size(), i=end-1; (end-i)<end; --i)
```

edited Nov 17 '10 at 15:24

answered Nov 17 '10 at 15:23

smerlin
4,096 2 21 45

```
for (it = v.end()-1; it != v.begin()-1; --it)
{
}
```

The "goes to" operator definitely messes with my head.

edited Jan 19 '12 at 20:10

answered Jan 19 '12 at 19:53

Peter Arandorenko
183 1 10

I think that:

```
for(unsigned i = v.size() - 1; i >= 0; --i)
```

is fine if you check

```
!v.empty()
```

earlier.

answered Nov 17 '10 at 15:12

Kylo
1,753 14 22

-1 This is an endless loop too :) – Armen Tsirunyan Nov 17 '10 at 15:12

@HardCoder1986: Think again :) – Armen Tsirunyan Nov 17 '10 at 15:15

@Armen OH SHI-... – Yippie-Ki-Yay Nov 17 '10 at 15:21

Oh, now I see :-) – Kylo Nov 17 '10 at 15:24
