Iterating C++ vector from the end to the begin

Is it possible to iterate a vector from the end to the begin?

```
for (vector<my_class>::iterator i = my_vector.end();
        i != my_vector.begin(); /* ?! */ ) {
}
```

Or is that only possible with something like that:

```
for (int i = my_vector.size() - 1; i >= 0; --i) {
}
```

c++ vector iterator

edited Sep 10 '13 at 14:44

Daniel Daranas
18.4k 6 47 90

asked Aug 31 '10 at 16:08

user
1,865 11 45 72

In C++11 you can use range-based for-loop with reverse adapter, see here – M.M Jul 22 '14 at 13:05

theoretically, on a 32 bit machine, for the second solution, if the vector size is larger than 2,147,483,647 + 1 it will overflow (vector::size() is unsigned), but currently chances are that you will never hit that limit (also current vector limit on 32 bit machines is 1,073,741,823). – clickstefan Dec 22 '14 at 17:15

7 Answers

Well, the best way is:

```
for (vector<my_class>::reverse_iterator i = my_vector.rbegin();
        i != my_vector.rend(); ++i ) {
}
```

rbegin()/rend() especically designed for that purpose. (And yes, incrementing a reverse_interator moves it backward)

Now, in theory, your method (using begin/end & `--i`) would work, vector's iterator being bidirectional, But remember, end() isn't the last element -- it's one beyond the last element, so you'd have to decrement first, and you are done when you reach begin() -- but you still have to do your processing.

```
vector<my_class>::iterator i = my_vector.end();
while (i != my_vector.begin())
{
     --i;
    /*do stuff */ )

}
```

UPDATE: I was apparently too aggressive in re-writing the for() loop into a while loop. (The important part is the the `--i` is at the beginning.)

edited Aug 31 '10 at 18:50

answered Aug 31 '10 at 16:11

James Curran
74.9k 23 140 224

2 Shouldn't it be `reverse_iterator` ? – a1ex07 Aug 31 '10 at 16:14

@a1ex07: right, thanks! – James Curran Aug 31 '10 at 16:16

I just realized that `--i` will cause a big problem if container is empty... Before going into `do - while` loop it makes sense to check `(my_vector.begin() != my_vector.end())` . – a1ex07 Aug 31 '10 at 18:07

1 Why are you using a `do-while` loop instead of just a `while` loop? Then you wouldn't need any special check for empty vectors. – jamesdlin Aug 31 '10 at 18:13

If you have `C++11` , you can make use of `auto` .

```
for (auto it = my_vector.rbegin(); it != my_vector.rend(); ++it)
{
}
```

answered Jul 20 '14 at 15:06

Akavall
24.6k 15 88 124

User `rend() / rbegin()` iterators:

```
for (vector<myclass>::reverse_iterator it = myvector.rbegin(); it !=
myvector.rend(); it++)
```

answered Aug 31 '10 at 16:10

a1ex07
28.6k 8 51 86

The well-established "pattern" for reverse-iterating through closed-open ranges looks as follows

```
// Iterate over [begin, end) range in reverse
for (iterator = end; iterator-- != begin; ) {
  // Process `*iterator`
}
```

or, if you prefer,

```
// Iterate over [begin, end) range in reverse
for (iterator = end; iterator != begin; ) {
  --iterator;
  // Process `*iterator`
}
```

This pattern is usable, for example, for reverse-indexing an array using an unsigned index

```
int array[N];
...
// Iterate over [0, N) range in reverse
for (unsigned i = N; i-- != 0; ) {
  array[i]; // <- process it
}
```

(People unfamiliar with this pattern often insist on using *signed* integer types for array indexing specifically because they incorrectly believe that unsigned types prevent reverse indexing)

It can be used for iterating over an array using a "sliding pointer" technique

```
// Iterate over [array, array + N) range in reverse
for (int *p = array + N; p-- != array; ) {
  *p; // <- process it
}
```

or it can be used for reverse-iteration over a vector using an ordinary (not reverse) iterator

```
for (vector<my_class>::iterator i = my_vector.end(); i-- != my_vector.begin(); ) {
  *i; // <- process it
}
```

answered Aug 31 '10 at 17:56

AnT
219k 27 328 563

Use reverse iterators and loop from `rbegin()` to

```
rend()
```

edited May 18 '13 at 5:18
Patrick D'Souza
2,930 2 13 30

answered Aug 31 '10 at 16:10
Steve Townsend
43.2k 4 58 114

```
template<class It>
std::reverse_iterator<It> reversed( It it ) {
  return std::reverse_iterator<It>(std::forward<It>(it));
}
```

Then:

```
for( auto rit = reversed(data.end()); rit != reversed(data.begin()); ++rit ) {
  std::cout << *rit;
```

Alternatively in C++14 just do:

```
for( auto rit = std::rbegin(data); rit != std::rend(data); ++rit ) {
  std::cout << *rit;
```

In C++03/11 most standard containers have a `.rbegin()` and `.rend()` method as well.

Finally, you can write the range adapter `backwards` as follows:

```
namespace adl_aux {
  using std::begin; using std::end;
  template<class C>
  decltype( begin( std::declval<C>() ) ) adl_begin( C&& c ) {
    return begin(std::forward<C>(c));
  }
  template<class C>
  decltype( end( std::declval<C>() ) ) adl_end( C&& c ) {
    return end(std::forward<C>(c));
  }
}

template<class It>
struct simple_range {
  It b_, e_;
  simple_range():b_(),e_(){}
  It begin() const { return b_; }
  It end() const { return e_; }
  simple_range( It b, It e ):b_(b), e_(e) {}

  template<class OtherRange>
  simple_range( OtherRange&& o ):
    simple_range(adl_aux::adl_begin(o), adl_aux::adl_end(o))
  {}

  // explicit defaults:
  simple_range( simple_range const& o ) = default;
  simple_range( simple_range && o ) = default;
  simple_range& operator=( simple_range const& o ) = default;
  simple_range& operator=( simple_range && o ) = default;
};
template<class C>
simple_range< decltype( reversed( adl_aux::adl_begin( std::declval<C&>() ) ) ) >
backwards( C&& c ) {
  return { reversed( adl_aux::adl_end(c) ), reversed( adl_aux::adl_begin(c) ) };
}
```

and now you can do this:

```
for (auto&& x : backwards(ctnr))
  std::cout << x;
```

which I think is quite pretty.

answered Jan 27 '15 at 15:19
Yakk
121k 13 119 249

---

use this code

```
//print the vector element in reverse order by normal iterator.
cout <<"print the vector element in reverse order by normal iterator." <<endl;
vector<string>::iterator iter=vec.end();
--iter;
while (iter != vec.begin())
{
    cout << *iter  << " ";
    --iter;
}
```

edited Jul 22 '14 at 12:30
macfij
2,142 1 10 18

answered Jul 22 '14 at 12:23
amit kumar
1

---

thanks...macfij – amit kumar Jul 22 '14 at 13:16