How to navigate through a vector using iterators? (C++)

Ask Question

The goal is to access the "nth" element of a vector of strings instead of the [] operator or the "at" method. From what I understand, iterators can be used to navigate through containers, but I've never used iterators before, and what I'm reading is confusing.

If anyone could give me some information on how to achieve this, I would appreciate it. Thank you.

c++ iterator

asked Mar 7 '10 at 5:21

kevin
561 2 8 20

Aren't vectors exclusive to the STL of C++? I will edit it regardless – kevin Mar 7 '10 at 5:43

@gabe, title specifies C++. – Michael Aaron Safyan Mar 7 '10 at 5:45

kevin: vector is a generic term which could be used by any language, particularly math-related ones like Mathematica or Matlab. – Gabe Mar 7 '10 at 5:48

@mich

comment. – kevin Mar 7 '10 at 5:57

3 Answers

You need to make use of the `begin` and `end` method of the `vector` class, which return the iterator referring to the first and the last element respectively.

```
using namespa

vector<strin


// push some
myvector.pus
myvector.pus
myvector.pus
myvector.pus


vector<strin
int n = 3;
int i = 0;

// now start
// and keep
// nth eleme
for(it = myv
    // found
    if(i ==
        cout
        breal
    }
}

// other eas
// using ope
```

gsamaras
44.1k 21 84 157

iswered Mar 7 '10 at 5:32

codaddict
329k 61 425 483

3 This misses the fact that `std::vector` has random access iterators. – sbi Mar 7 '10 at 6:38

18 Regardless of whether you know the iterator type is random-access or not, the "best" way to move an iterator forward n spaces is not to write your own loop, but to call `std::advance(it, n)`. It's

will automatically use `it + n` if the iterator is tagged as random-access, or do the loop if it has to. – Steve J Mar 7 '10 at 13:47

Typically, iterators are used to access elements of a container in linear fashion; however, with "random access iterators", it is possible to access any element in the same fashion as `operator[]`.

To **access**

```
vec.begin()
vec.begin()+
// ...
vec.begin()+
// ...
vec.begin()+
```

The following is an example of a **typical access pattern** (earlier versions of C++):

```
int sum = 0;
using Iter =
for (Iter it
    sum += *
}
```

The advantage of using iterator is that you can apply the **same pattern with other containers**:

```
sum = 0;
for (Iter it
    sum += *
}
```

For this reason, it is really easy to create template code that will work the same **regardless of the container type**. Another advantage of iterators is that it doesn't

a forward iterator that can read data from an input stream, or that simply generates data on the fly (e.g. a range or random number generator).

Another option using `std::for_each` and lambdas:

```
sum = 0;
std::for_each
```

Since C++11 you can use `auto` to avoid specifying a very long, complicated type name of the iterator as seen before (or even more complex):

```
sum = 0;
for (auto it
    sum += *
}
```

And, in addition, there is a simpler for-each variant:

```
sum = 0;
for (auto va
    sum += va
```

you have to be careful whether you are adding integer or floating point numbers.

lited May 25 '16 at 16:04

holzkohlengrill
334 7 16

ıswered Mar 7 '10 at 5:31

Michael Aaron Safyan
73.7k 12 107 181

Vector's iterators are random access iterators which means they look and feel like plain pointers.

You can access the nth element by adding n to the iterator returned from the container's `begin()` method, or you can use operator `[]`.

```
std::vector<
std::Vector<

int sixth =
int third =
int second =
```

Alternatively you can use

(You'd have to consider whether you really want to perform "random access" with non-random-access iterators, since that might be an expensive thing to do.)

```
std::vector<
std::vector<

std::advance
int sixth =
```

lited Mar 7 '10 at 14:41

swered Mar 7 '10 at 11:18

UncleBens
33.7k 6 44 84

1 You can use `advance` for random-access iterators too, or iterators of unknown category, since it is guaranteed to operate in consta

iterators should be correctly tagged. – Steve J Mar 7 '10 at 13:50

Indeed, but `advance` is really annoying to use (because of the out parameter usage) if you know you are dealing with random access iterators. Would only recommend in generic code, and if not used a lot (if the algorithm doesn't support non-random-

`std::sort` *could* sort a `std::list` but it doesn't because it would be ridiculously inefficient). – UncleBe Mar 7 '10 at 14:03

Sure, the classic example would be if your algorithm only actually *needs* an InputIterator, but for whatever reason it sometimes skips ahead, so you want it to be more efficient if the iterator does have random

algorithm to random access only by using `operator+`. But the question was explicitly about vector, so there's nothing wrong with the first part of your answer. I just thought the second part might imply "you can't use advance with random access iterators, even if you want to" to someone who has never seen `advance` before. – Steve J Mar 7 '10 at 14:10

exampl e with a vector. – UncleBe Mar 7 '10 at 14:42

second line, `Vecto r` should be lower case – Lei Yan Jul 31 '17 at 2:49