



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`

This site uses cookies to deliver our services and to show you relevant ads and job listings. By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service. Your use of Stack Overflow's Products and Services, including the Stack Overflow Network, is subject to these policies and terms.

asked Mar 7 '10 at 5:21



[kevin](#)

561 2 8 20

Aren't
vectors
exclusi
ve to
the
STL of
C++? I
will edit
it
regardl
ess –
[kevin](#)
Mar 7
'10 at
5:43

@gabe
, title
specifi
es
C++. –
[Michael Aaron Safyan](#)
Mar 7
'10 at
5:45

kevin:
vector
is a
generic
term
which
could
be
used
by any
langua
ge,
particul
arly
math-
related
ones
like
Mathe
matica
or
Matlab.
– [Gabe](#)
Mar 7
'10 at
5:48

@mich

This site uses cookies to deliver our services and to show you relevant ads and job listings. By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service. Your use of Stack Overflow's Products and Services, including the Stack Overflow Network, is subject to these policies and terms.

comme
nt. —
[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 namespace
```

```
vector<string
```

```
// push some
myvector.push
myvector.push
myvector.push
myvector.push
```

```
vector<string
int n = 3;
int i = 0;
```

```
// now start
// and keep .
// nth eleme
for(it = myv
    // found
    if(i == 1
        cout
        break
    }
}
```

```
// other eas.
// using ope
```

This site uses cookies to deliver our services and to show you relevant ads and job listings. By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service. Your use of Stack Overflow's Products and Services, including the Stack Overflow Network, is subject to these policies and terms.

[gsamaras](#)

44.1k

21

84

157

answered Mar 7 '10 at 5:32

[codaddict](#)

329k

61

425

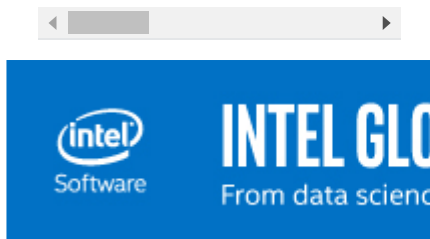
483

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

This site uses cookies to deliver our services and to show you relevant ads and job listings. By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service. Your use of Stack Overflow's Products and Services, including the Stack Overflow Network, is subject to these policies and terms.

will
 autom
 atically
 use
 it +
 n if
 the
 iterator
 is
 tagged
 as
 rando
 m-
 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**

This site uses cookies to deliver our services and to show you relevant ads and job listings. By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service. Your use of Stack Overflow's Products and Services, including the Stack Overflow Network, is subject to these policies and terms.

```
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 += *it;
}
```

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

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

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

This site uses cookies to deliver our services and to show you relevant ads and job listings. By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service. Your use of Stack Overflow's Products and Services, including the Stack Overflow Network, is subject to these policies and terms.

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_eac`
h and
lambdas:

```
sum = 0;
std::for_eacl
```

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 += *i
}
```

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

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

This site uses cookies to deliver our services and to show you relevant ads and job listings. By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service. Your use of Stack Overflow's Products and Services, including the Stack Overflow Network, is subject to these policies and terms.

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

edited May 25 '16 at 16:04



[holzkohlengrill](#)

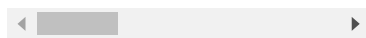
334 7 16

answered 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

This site uses cookies to deliver our services and to show you relevant ads and job listings. By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service. Your use of Stack Overflow's Products and Services, including the Stack Overflow Network, is subject to these policies and terms.

(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 =
```

edited Mar 7 '10 at 14:41

answered 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 constant

This site uses cookies to deliver our services and to show you relevant ads and job listings. By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service. Your use of Stack Overflow's Products and Services, including the Stack Overflow Network, is subject to these policies and terms.

iterator
s
should
be
correctl
y
tagged

. –

[Steve J](#)

Mar 7
'10 at
13:50

Indeed
, but
advan
ce is
really
annoyi
ng to
use
(becau
se of
the out
param
eter
usage)
if you
know
you
are
dealing
with
rando
m
access
iterator
s.
Would
only
recom
mend
in
generic
code,
and if
not
used a
lot (if
the
algorith
m
doesn't
suppor
t non-
rando
m-

This site uses cookies to deliver our services and to show you relevant ads and job listings. By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service. Your use of Stack Overflow's Products and Services, including the Stack Overflow Network, is subject to these policies and terms.

std::
sort
could
sort a
std::
list
but it
doesn't
becaus
e it
would
be
ridiculo
usly
ineffici
ent). –
[UncleB](#)
Mar 7
'10 at
14:03

Sure,
the
classic
exampl
e
would
be if
your
algorithm
m only
actuall
y
needs
an
InputIt
erator,
but for
whatev
er
reason
it
someti
mes
skips
ahead,
so you
want it
to be
more
efficien
t if the
iterator
does
have
rando
m

This site uses cookies to deliver our services and to show you relevant ads and job listings. By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service. Your use of Stack Overflow's Products and Services, including the Stack Overflow Network, is subject to these policies and terms.

algorithm
m to
rando
m
access
only by
using
opera
tor+ .

But the
questio
n was
explicitl
y about
vector,
so
there's
nothing
wrong
with
the first
part of
your
answer
. I just
though
t the
second
part
might
imply
"you
can't
use
advanc
e with
rando
m
access
iterator
s, even
if you
want
to" to
someo
ne who
has
never
seen
advan
ce
before.

—

[Steve Jr](#)
Mar 7
'10 at
14:10

This site uses cookies to deliver our services and to show you relevant ads and job listings. By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service. Your use of Stack Overflow's Products and Services, including the Stack Overflow Network, is subject to these policies and terms.

exampl
e with
a
vector.

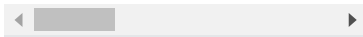
—

[UncleBe](#)

Mar 7
'10 at
14:42

second
line,
Vecto
r

should
be
lower
case —
[Lei Yan](#)
Jul 31
'17 at
2:49



This site uses cookies to deliver our services and to show you relevant ads and job listings. By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service. Your use of Stack Overflow's Products and Services, including the Stack Overflow Network, is subject to these policies and terms.