Using the Power of Iterators

Kate Gregory

@gregcons www.gregcons.com/kateblog

Pre-allocating Vectors Is No Fun

```
vector<int> v1(10);

fill(begin(v1), end(v1), 1);

fill_n(begin(v1), 6, 2);

iota(begin(v1), end(v1), 1);
```

- What if you don't know the size?
- What if there's no default constructor?
- What if the default constructor is expensive?

Use a Different Iterator

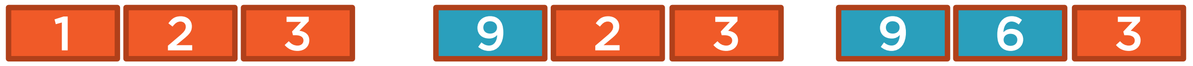
back_inserter

front_inserter

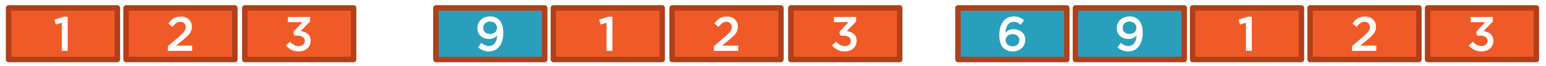
Changing Values with Iterators

Different iterators can change the behavior of the algorithm

begin(v)



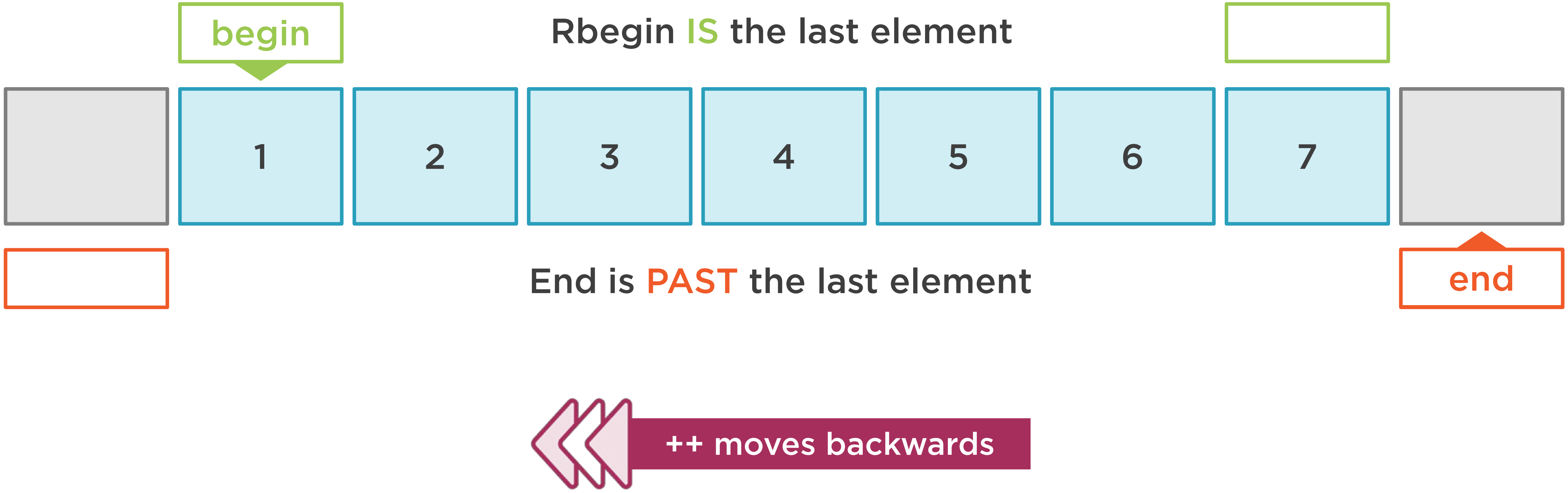
front_inserter(v)



Use Them with (Almost) Any Algorithm

fill_n	generate_n	transform
copy, copy_if	unique_copy	reverse_copy

Reverse


begin
Rbegin IS the last element
1
2
3
4
5
6
7
End is PAST the last element
end
++ moves backwards

Iterator Arithmetic

auto	++, --
+=, -=	+ (int), - (int or it)

Iterators to const Elements

Not all collections can hold const objects

Iterators don’t get you around const

cbegin() returns const_iterator

end() and cend(), reverse too

Express intent, beyond const correctness

Summary

Use the functions you already know with different iterators

Inserters

- Saves preallocating

Reverse

Const

Iterator arithmetic is safe

- Sometimes convenient

Having many kinds of iterators means less algorithms

- No find / reversefind / constfind
- Less to learn and remember