Breaking the Ice with Useful Standard Algorithms: Sorting std::vector

Giovanni Dicanio
AUTHOR, SOFTWARE ENGINEER

https://blogs.msmvps.com/gdicanio

Overview

Sorting std::vector using std::*sort*

Iterators

Relation between *iterators*, *containers* and *algorithms*

Custom sorting

Sorting std::vector

Code from scratch

Reuse Standard Library's algorithms

```
#include
<algorithm>
```

```
std::sort( something… );
```

Sorting std::vector

Use the Standard Library's *sort* algorithm

```
// v is a std::vector
```


```
sort( first, last );
```

Sorting std::vector

Use the Standard Library's *sort* algorithm

```
// v is a std::vector
```

Begin of data

End of data

```
sort( begin(v), end(v) );
```

Sorting std::vector

Use the Standard Library's *sort* algorithm

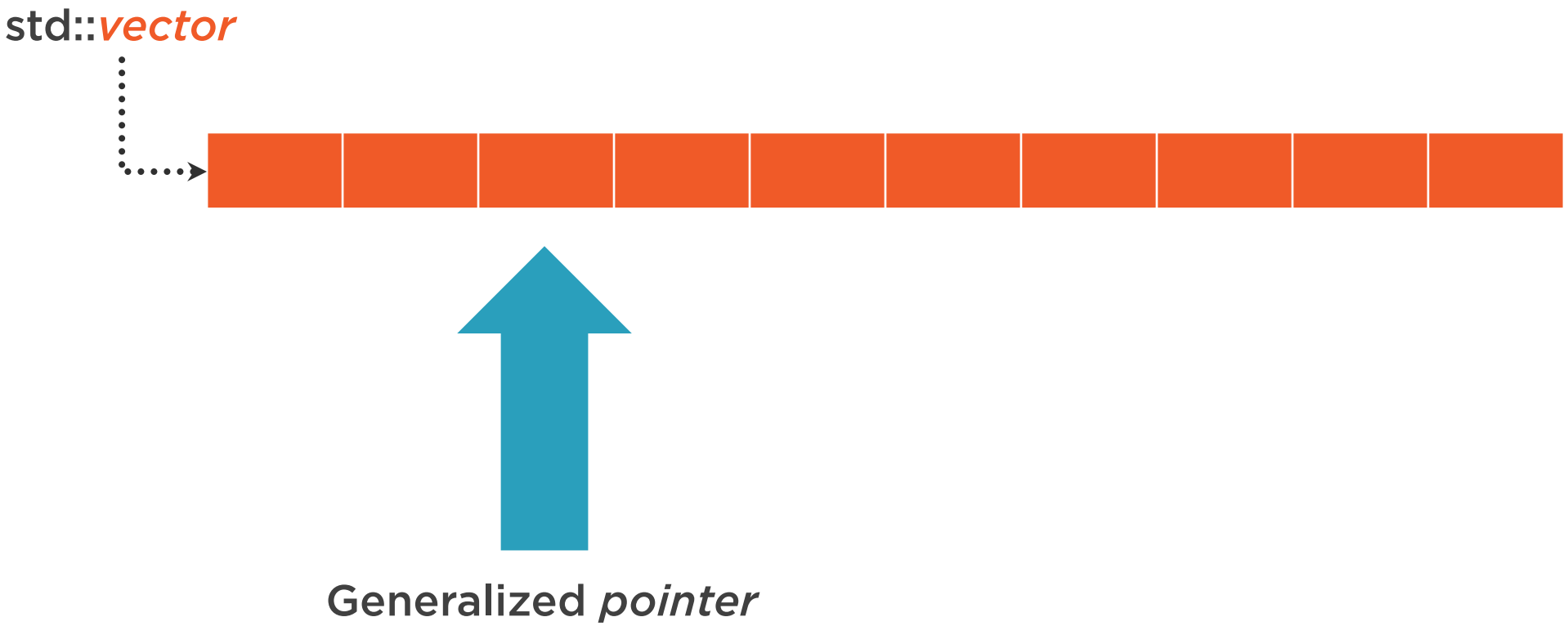
```
// v is a std::vector
```


```
sort( begin(v), end(v) );
```

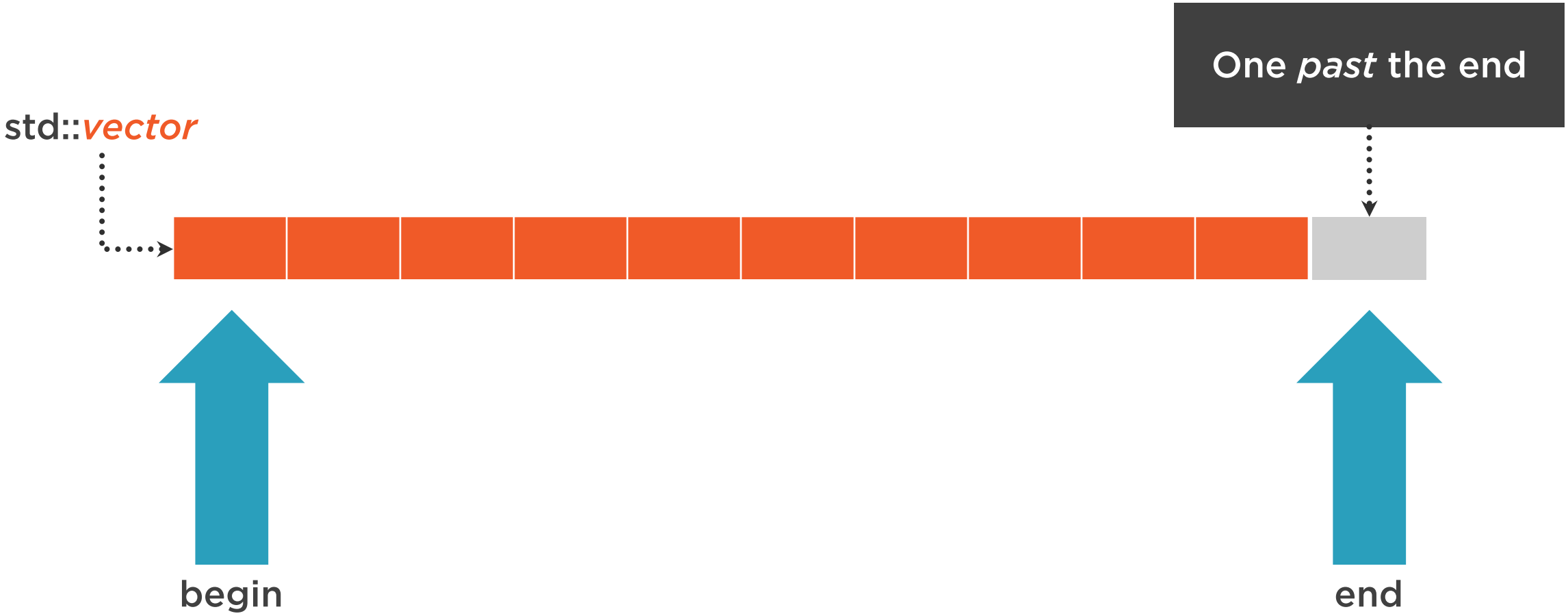
Sorting std::vector

Use the Standard Library's *sort* algorithm

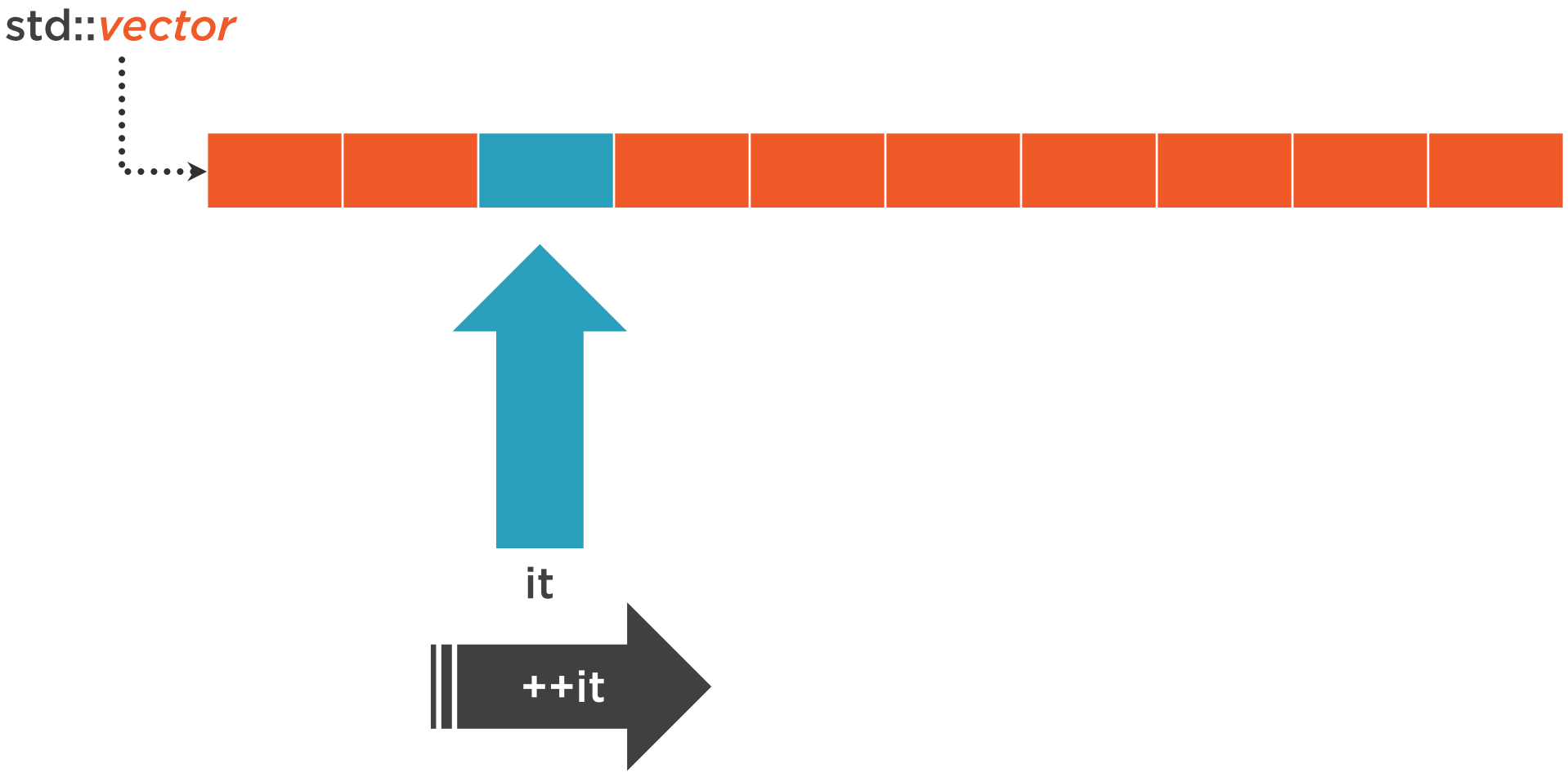
What Are Iterators?



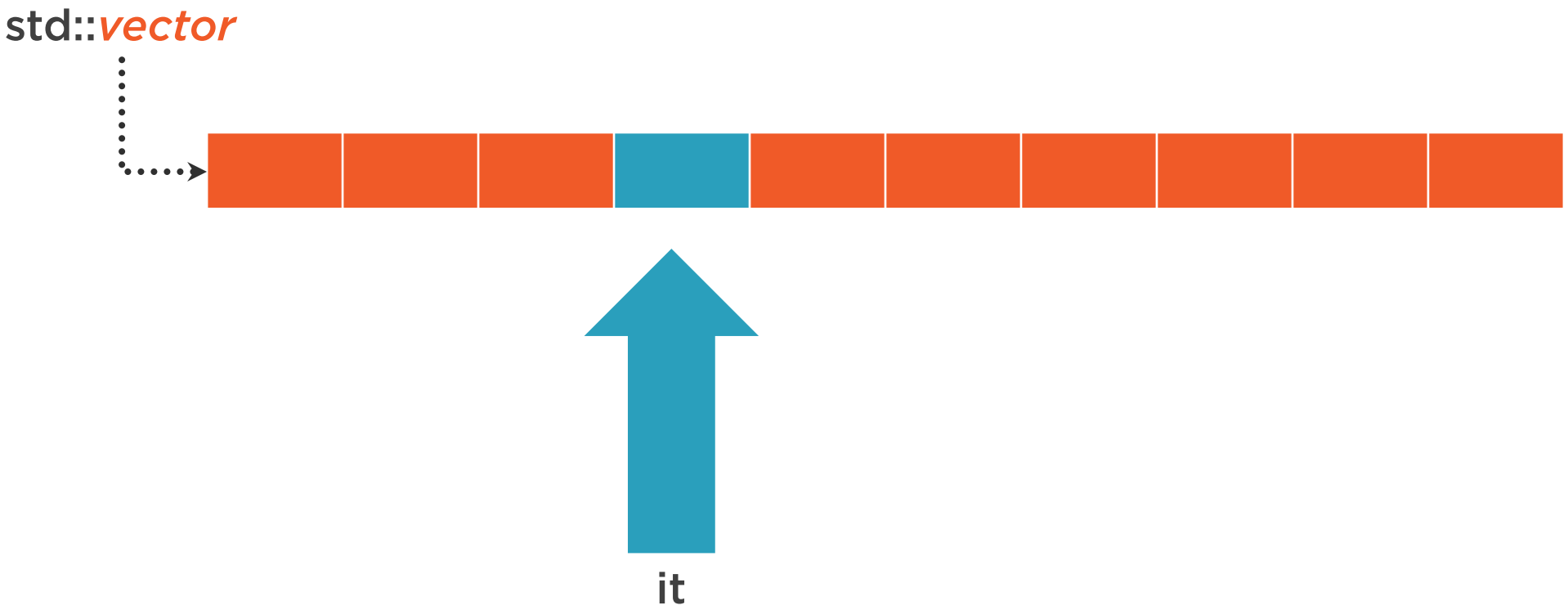
What Are Iterators?



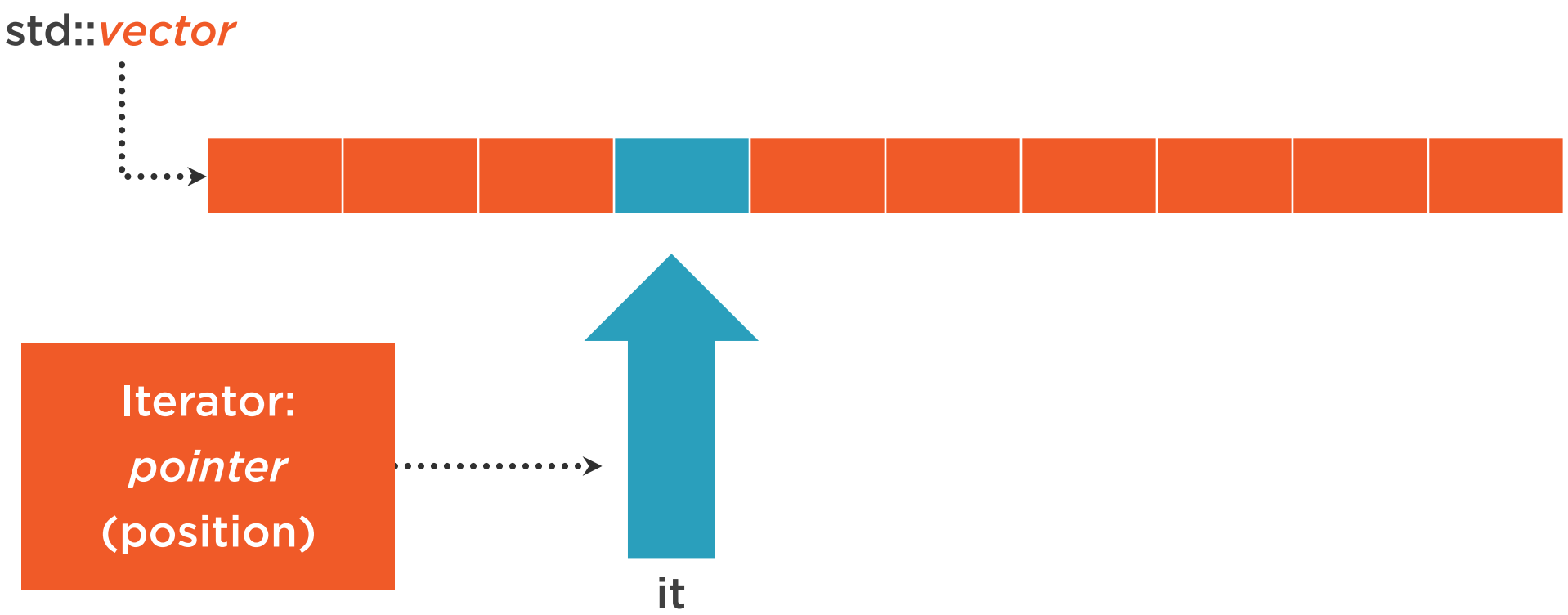
Advancing Iterators



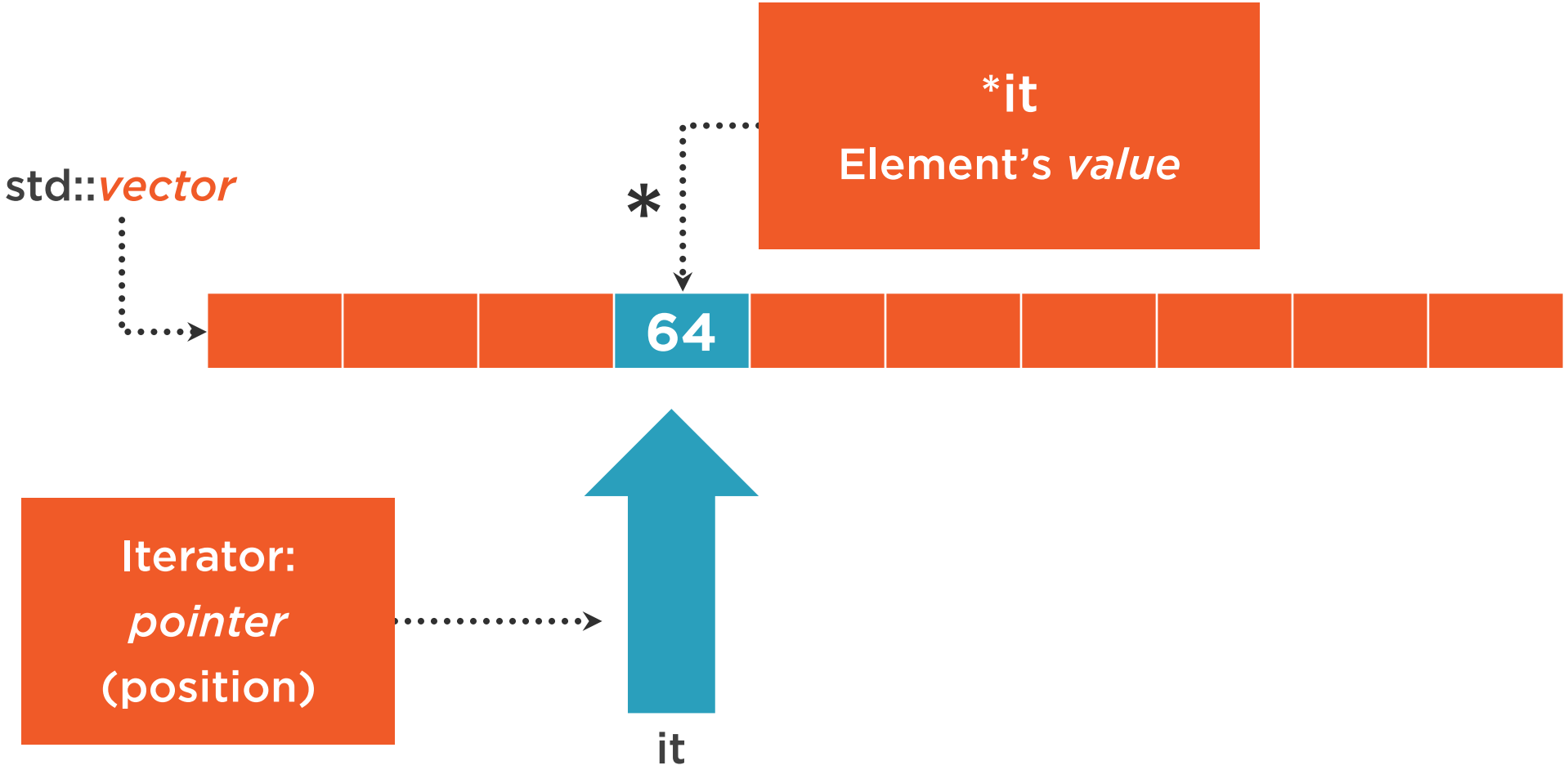
Advancing Iterators



Dereferencing Iterators

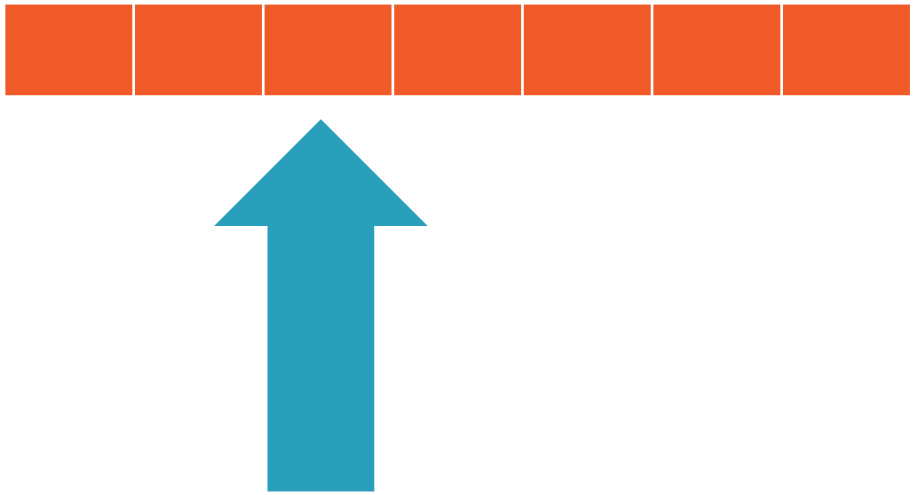


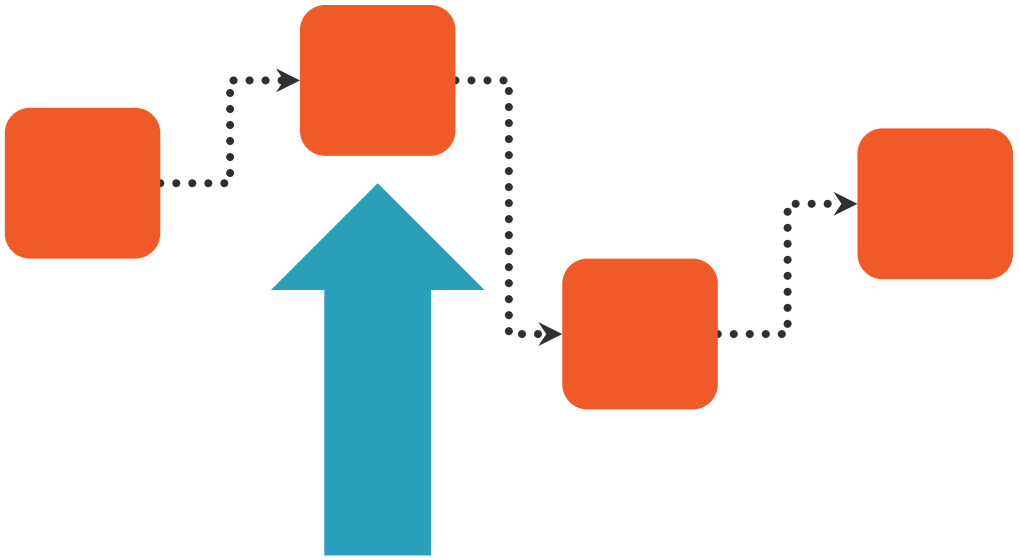
Dereferencing Iterators



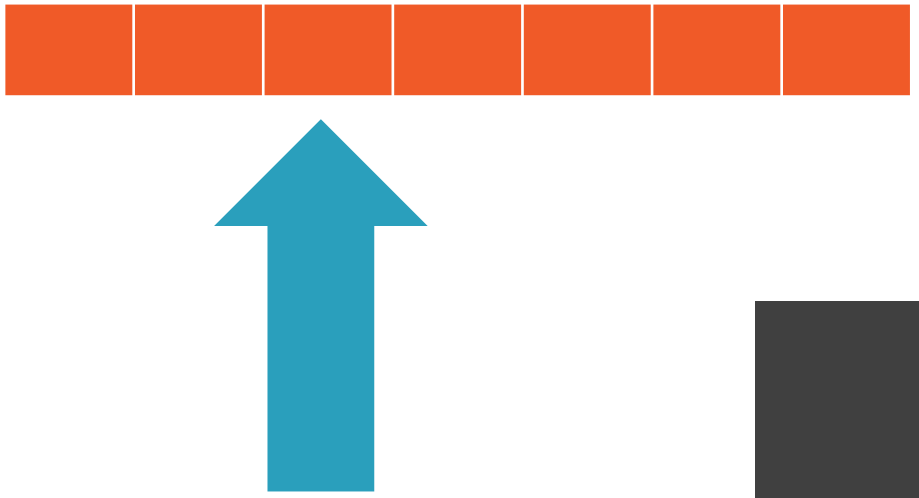
Iterators Abstract Away Containers

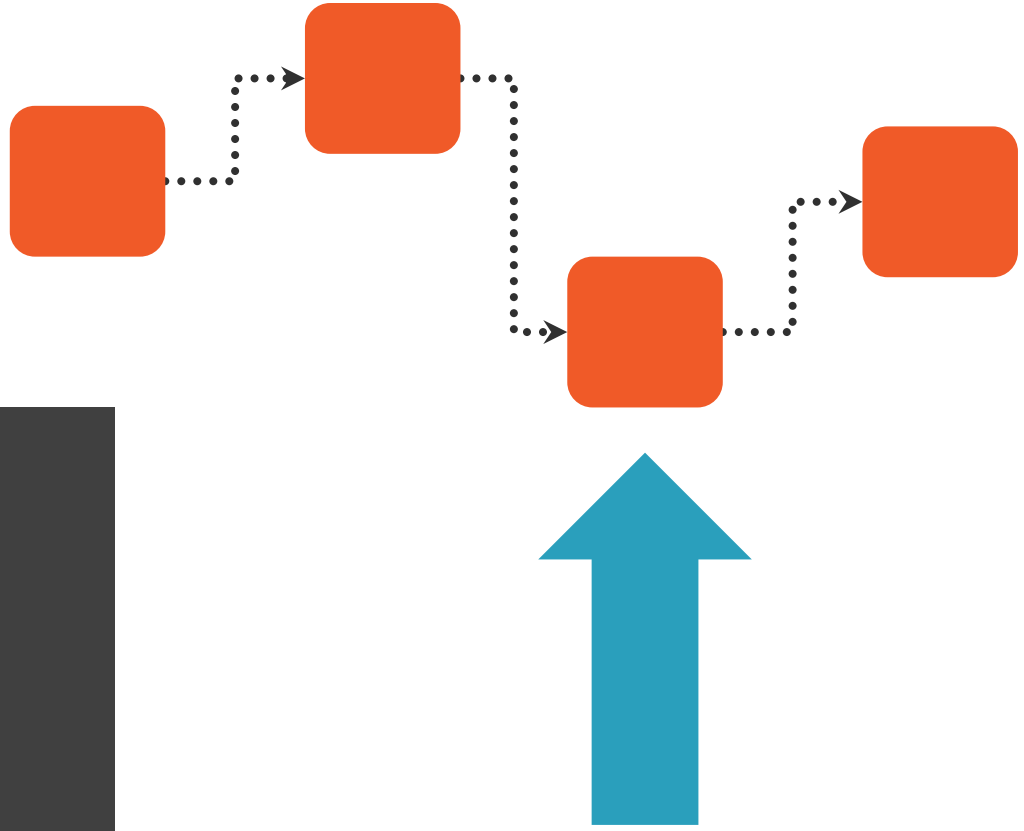
Iterators Abstract Away Containers





Iterators Abstract Away Containers

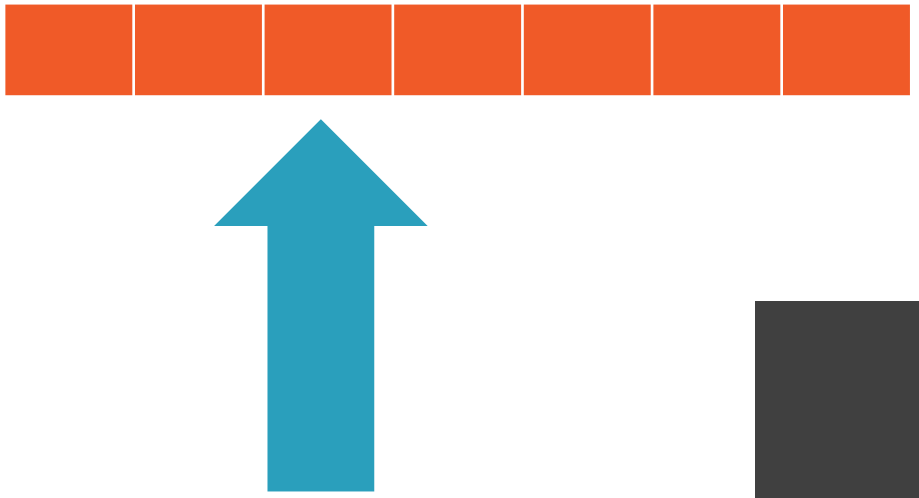


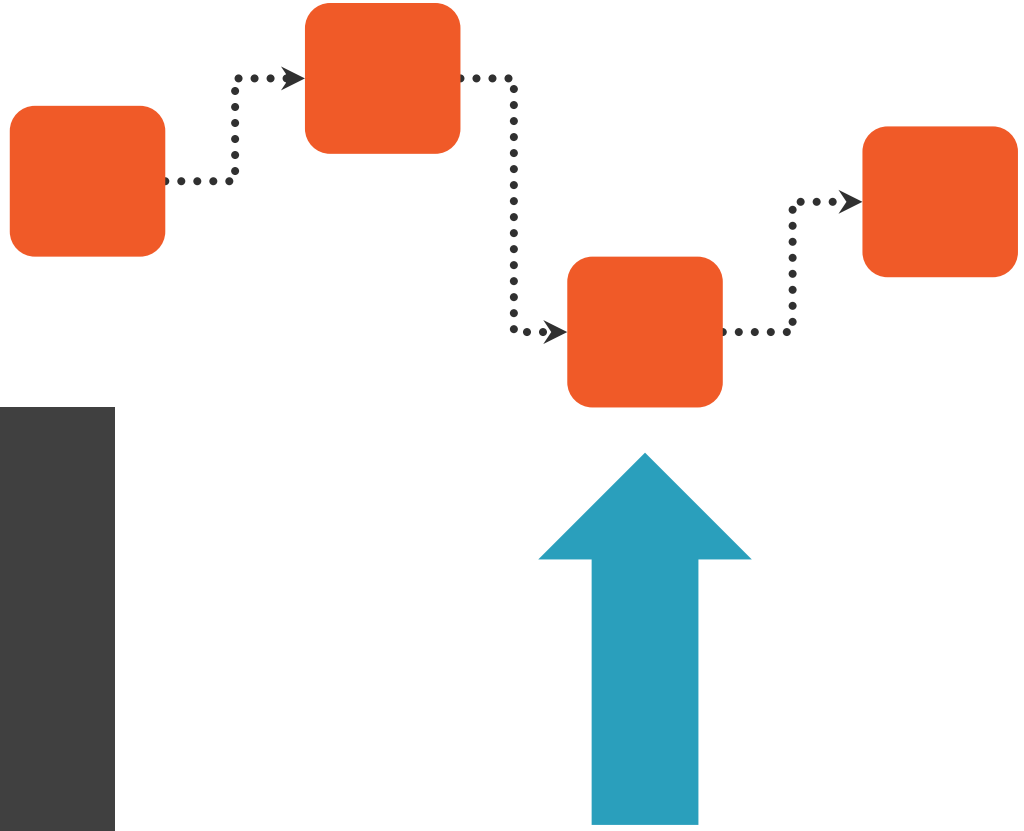


Same syntax

++it: advance

Iterators Abstract Away Containers





Same syntax

++it: advance

*it: deref

```
vector<int>::iterator
vector<string>::iterator
vector<string>::const_iterator
```

What's the Type of an Iterator?

```
vector<int>::iterator
vector<string>::iterator
vector<string>::const_iterator
```

Use
auto

What's the Type of an Iterator?

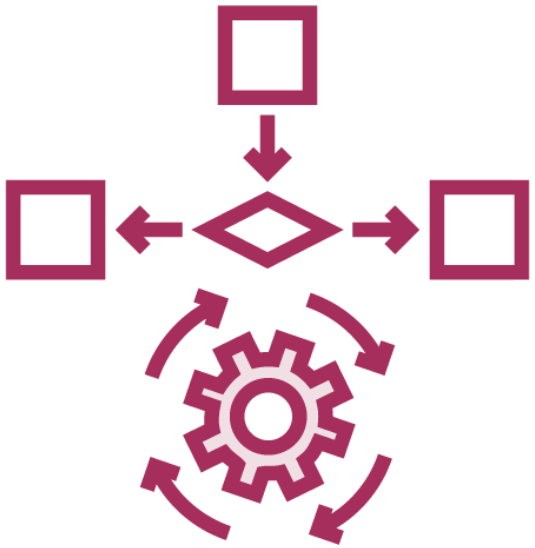
```
auto
auto
auto
```

Use
auto

What's the Type of an Iterator?

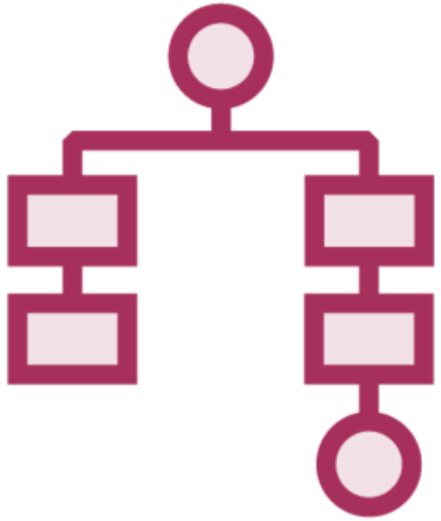
Use *auto* to simplify your C++ code

Algorithms, Iterators, and Containers



Algorithms

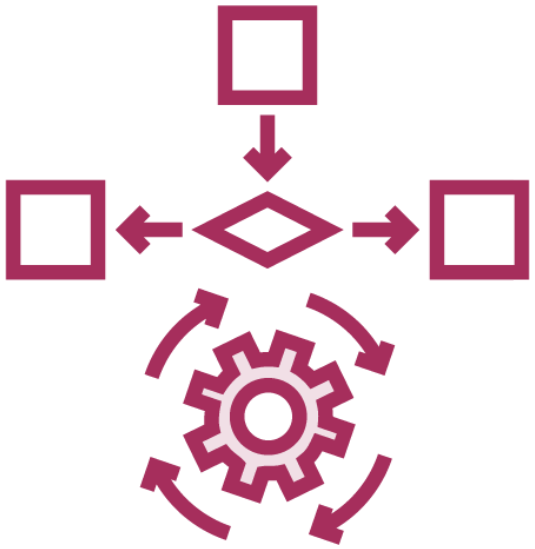
Iterators



Containers

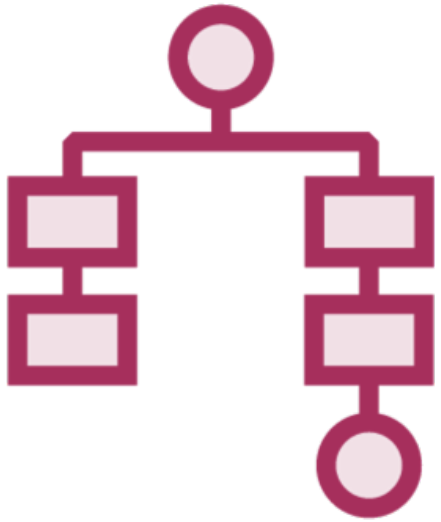
Algorithms use *iterators* to *access* elements inside containers

Algorithms, Iterators, and Containers



Algorithms

Iterators

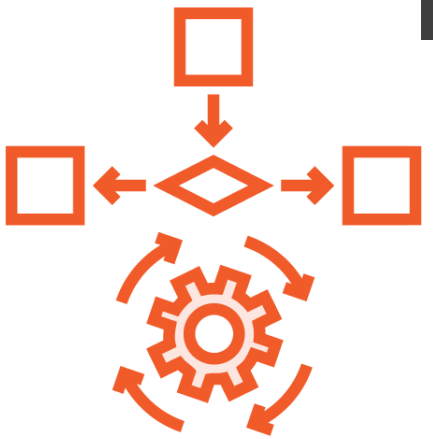


Containers

Reusing Algorithms with Different Containers

Decoupling:

elegant and *extensible* design



Algorithm
(e.g. std::sort)

std::*vector*

Other *standard* containers

Custom containers

```
// v is a std::vector
```

Begin
of data

End
of data

```
sort( begin(v), end(v) );
```

Sorting Using Custom Comparison

```
// v is a std::vector

sort( begin(v), end(v) );
```

ADDITIONAL PARAMETER

Sorting Using Custom Comparison

```
// v is a std::vector
```

```
sort( begin(v), end(v), CustomComparison );
```

Sorting Using Custom Comparison

```
// 'names' is a vector<string>

sort(begin(names), end(names),
  [](auto const& a, auto const& b) {
    // Compare by string length
  }
);
```

λ

Sorting Using Custom Comparison

Custom sorting of a vector<string> using a *lambda*

```
// 'names' is a vector<string>
sort(begin(names), end(names),
  [](auto const& a, auto const& b) {
    // Compare by string length
  }
);
```

Sorting Using Custom Comparison

Custom sorting of a vector<string> using a *lambda*

```
// 'names' is a vector<string>
sort(begin(names), end(names),
  [](auto const& a, auto const& b) {
    // Compare by string length
  }
);
```

Sorting Using Custom Comparison

Custom sorting of a vector<string> using a *lambda*

```
// 'names' is a vector<string>
sort(begin(names), end(names),
  [](auto const& a, auto const& b) {
    return a.length() < b.length();
  }
);
```

SHORTER STRINGS FIRST

Sorting Using Custom Comparison

Custom sorting of a vector<string> using a *lambda*

C64

Connie

Amiga 500

I'm a long string

SHORTER STRINGS FIRST

Sorting Using Custom Comparison

Sorting by *string length*: shorter strings first

Custom Sorting in Action

Demo: Generic Lambdas in Action

“Practical C++14 and C++17 Features”

To Learn More About Lambdas

Better Lambdas

“Practical C++14 and C++17 Features”

To Learn More on Standard C++ Library's Algorithms

"Beautiful C++: STL Algorithms"

Summary

Sorting std::vector with std::*sort*

Iterators

Algorithms, iterators and containers

Custom sorting