Counting and Finding

Kate Gregory

@gregcons www.gregcons.com/kateblog

What are you trying to do?

```
//count how many entries have the target value (2)
int twos = 0;
int const target = 2;
for (size_t i = 0; i < v.size(); i++)
{
   if (v[i] == target) {  twos++; }
}
```

```
int const target = 2;
int simple = count(v.begin(), v.end(), target);
int simple = count(begin(v), end(v), target);
```

Member or Nonmember Begin and End

Member	Nonmember
v.begin(), v.end()	begin(v), end(v)
Available for all collections in the library including vector, string, map	Calls v.begin() or v.end() if it exists
	Works for C-style arrays too
	You can write a free function for a collection without member begin() and end()
	The safest choice and a good habit

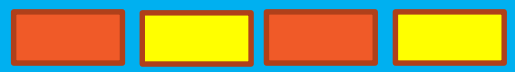
```
//count how many entries are odd
int odds = 0;
for (auto elem : v)
{
   if (elem % 2 != 0) { odds++;}
}
```

```
odds = count_if(begin(v), end(v),
        [](auto elem) {return elem % 2 != 0; });
```

Why Count Anyway?

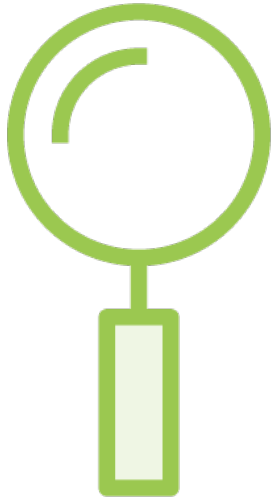
Are all of these elements ____ ?

Are any of these elements ____ ?

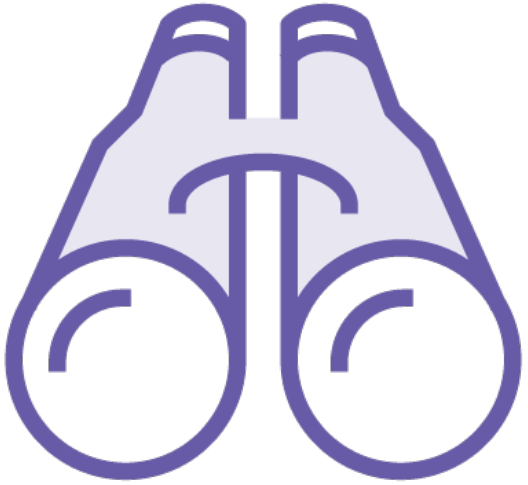


Are none of these elements ____ ?

Finding



First element with a specific value: find()



First element meeting a condition: find_if()

Returns an iterator

Variations on Finding

find_if_not	find_first_of	search
find_end	search_n	adjacent_find

Summary

A well named function says far more than a comment

Algorithms work with any collection that provides the right iterators

- Or part of a collection

Prefer begin(v) to v.begin

Iterators can be incremented, decremented, dereferenced, and passed to other functions

Many variations on finding and counting mean you don't need to build your own

- Use what is there
- Cppreference.com when you forget