Standard Library Algorithms

Kate Gregory

@gregcons www.gregcons.com/kateblog

Discoverability

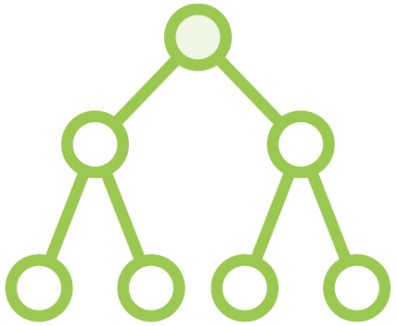
```
std::vector<int> v{ 1,2,3,4,5,6 };

v.
```

- emplace_back
- empty
- end
- erase
- front
- get_allocator
- insert
- max_size

Not everything is a member function

Just because you don't see find in the list, doesn't mean you need to code it yourself

Collections, Algorithms, Iterators

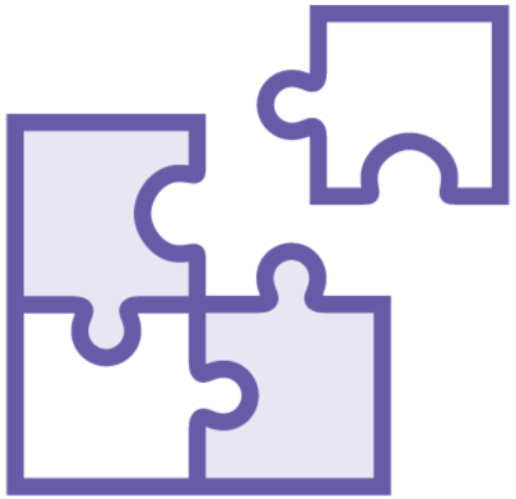


Collections

Algorithms

Iterators

Say What You Mean

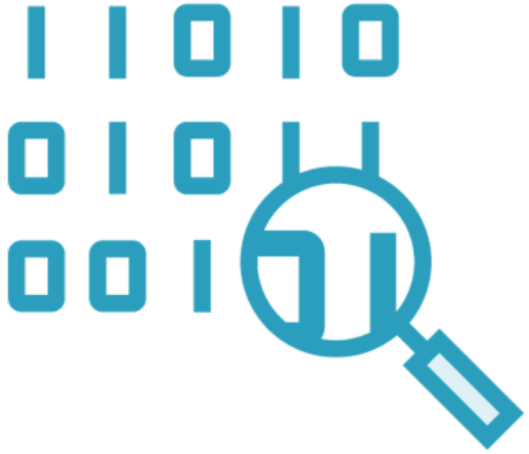


Why make someone puzzle out your code?

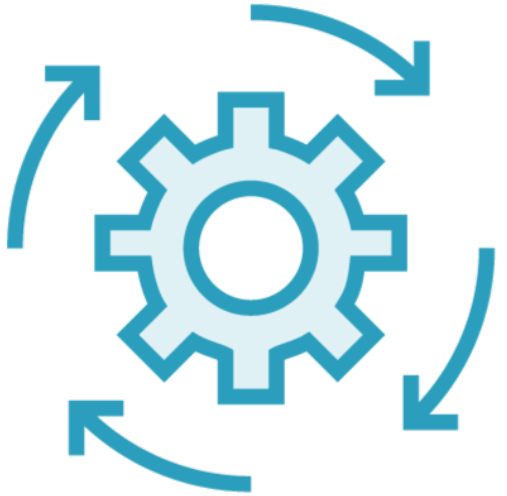
All for loops look similar

Algorithms Have Names That Are Clues

If you're sorting, use sort



If you're finding, use find



If you're generating elements, use generate

Less Fuss Over Style

i++?
++i?
In the for statement?
In the loop body?

Iterators vs. random access []

Names for indices, iterators, etc

```
vector<int> v;

for (int i = 0; i<5; i++)

    v.push_back(i++);


for (auto it = begin(v3); it !=end
(v3);it++)

{

   if (*it == 3)

      v3.erase(it);

}
```

- Double incrementing
- Invalidating iterators
- And plenty more

Do Not Write Raw Loops!

Learn to recognize standard algorithms

- Usually a giant hint in the name

Standard Often Means Interchangeable

The algorithms work with iterators

Most containers support most iterators

Easy to change containers without changing other code

Beautiful C++: STL Algorithms

Counting and Finding

Sorting

Comparing and Accumulating

Generating and Manipulating Collections

Using the Power of Iterators

Unexpectedly Useful Operations

Conventions

Summary

Calling named functions makes your code expressive and readable

Not having to write them saves you work

- They handle edge cases
- They may be faster than what you would write

It's pretty easy to change container and use the same algorithms

The only tough part is finding and learning them