Getting to know STL Algorithms

gbgcpp 2018-04-25

Jacob Mossberg

Want to talk C++?

Please get in touch if you have any C++ related topics you would like to speak about. Maybe you want to read up about something and share your new knowledge with the rest of the group. We welcome talks on both beginner and expert level.

Possible formats:

- Presentation
- Workshop
- Presentation + workshop
- Coding dojo
- Several mini presentations

Algorithms

- find
- rotate
- copy
- count

Iterators

Algorithms operate on iterators and not containers

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

auto n = std::find(v.begin(), v.end(), 2);

if(v.end() != n) {
  std::cout << "v contains 2" << std::endl;
}
```

```
v contains 2
```

Predicates

Wikipedia*:

"*In mathematical logic, a predicate is commonly understood to be a Boolean-valued function P: X→ {true, false}, called the predicate on X*"

* https://en.wikipedia.org/wiki/Predicate_(mathematical_logic)

Exercises

https://github.com/jmossberg/stlalgorithms

find & search

find

Find first element equal to 2

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

auto f = std::find(v.begin(),v.end(),2);

if(v.end() != f) {
  std::cout << "v contains " << *f << std::endl;
}
```

```
v contains 2
```

find_if

Find first element for which predicate returns true

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

auto f = std::find_if(v.begin(),v.end(), p);

if(v.end() != f) {
  std::cout << *f << std::endl;
}
```

```
5
```

find_if

```
bool p(int v)
{
  return v > 2;
}
```

Find first element for which predicate returns true

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

auto f = std::find_if(v.begin(),v.end(), p);

if(v.end() != f) {
  std::cout << *f << std::endl;
}
```

```
5
```

search

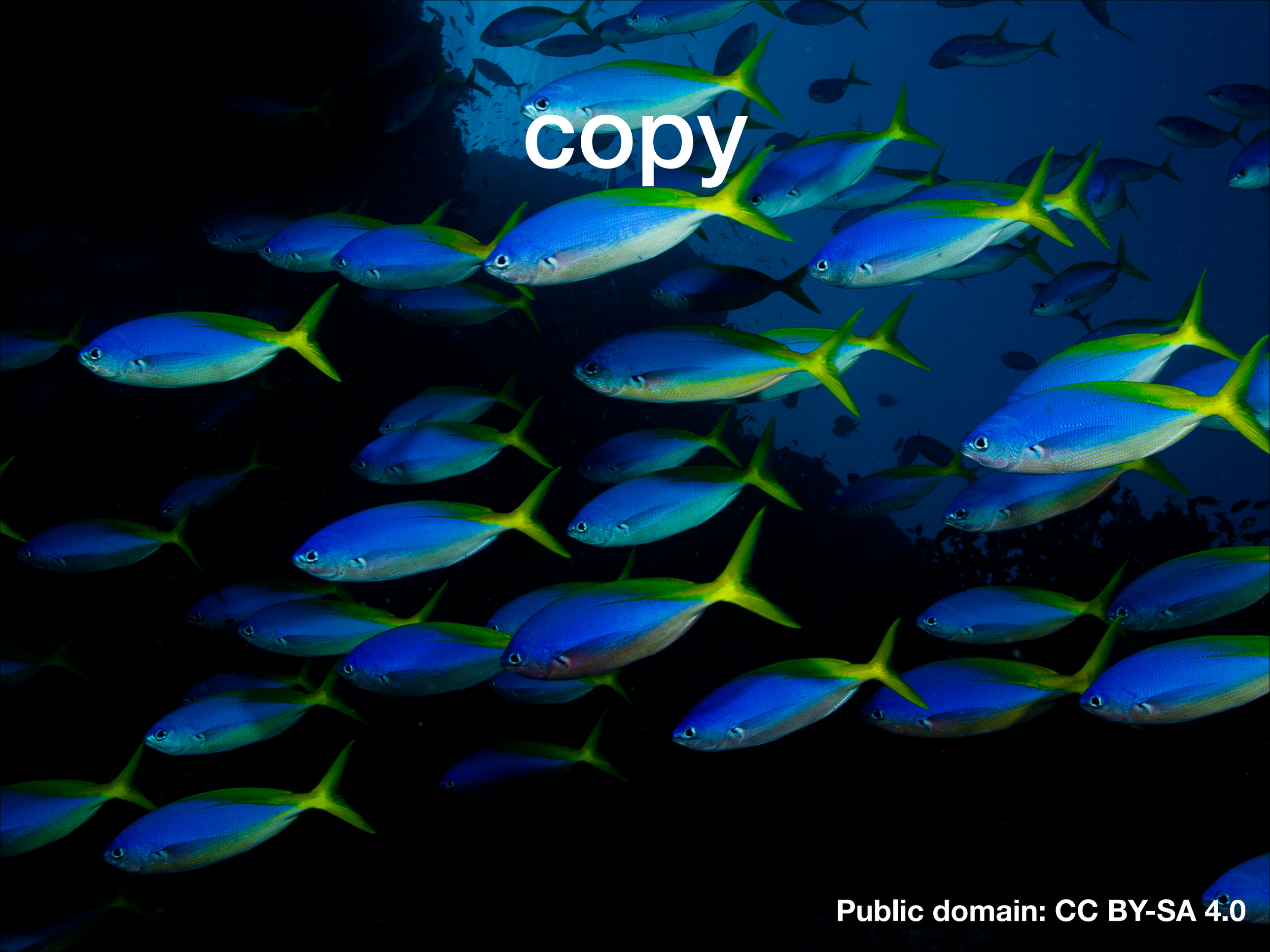
Search for sub-sequence in sequence

```
std::string seq{"gbgcpp"};
std::string sub_seq{"cpp"};

auto it = std::search(seq.begin(),seq.end(),\
                      sub_seq.begin(),sub_seq.end());

if(it != seq.end()) {
  std::cout << "found";
}
```

```
found
```

Exercises

Do exercise 1 & 2



copy

Copy elements from one range to another

```
std::vector<int> v1{1,2,5};
std::vector<int> v2(v1.size());

std::copy(v1.begin(),v1.end(),v2.begin());

for(auto v : v2) {
  std::cout << v << " ";
}
```

```
1 2 5
```

copy

constructs a std::back_insert_iterator for v2

Copy elements from one range to another

```
std::vector<int> v1{1,2,5};
std::vector<int> v2;

std::copy(v1.begin(),v1.end(),std::back_inserter(v2));

for(auto v : v2) {
    std::cout << v << " ";
}
```

```
1 2 5
```

copy_if

Copy elements for which the predicate returns true

```
std::vector<int> v1{1,2,5,3,7,9,6,6};
std::vector<int> v2;
auto bi = std::back_inserter(v2);

std::copy_if(v1.begin(),v1.end(),bi, \
             [](int &x){return x % 3 == 0;});

for(auto v : v2) {
    std::cout << v << " ";
}
```

Use lambda to define predicate.

Give me numbers divisible by 3!

```
3 9 6 6
```

copy 

Copy elements to output stream iterator

```
std::vector<int> v1{1,2,5,3,7,9,6,6};
auto oi = std::ostream_iterator<int>(std::cout, " ");

std::copy(v1.begin(),v1.end(), oi);
```

```
1 2 5 3 7 9 6 6
```

Exercises

Do exercise 3

for_each, transform, generate

for_each

Apply an operation to each element of a sequence

```
std::vector<int> v{1, 2, 3};
auto op = [](int &x){ ++x; };

std::for_each(v.begin(), v.end(), op);

for (auto a : v) {
    std::cout << a << " ";
}
```

2 3 4

transform

Apply an operation to each element of a sequence and store result in another sequence

```
std::vector<int> v1{1, 2, 3};
std::vector<int> v2;

auto op = [](const int &x){ return (x+1); };

std::transform(v1.begin(), v1.end(), std::back_inserter(v2), op);

for (auto a : v2) {
    std::cout << a << " ";
}
```

```
2 3 4
```

Exercises

Do exercise 4

generate

Assign each element the value returned by function object

```
std::vector<int> v1(5);
int n{0};
auto g = [&n](){return n++;};

std::generate(v1.begin(),v1.end(), g);

for(auto v : v1) {
    std::cout << v << " ";
}
```

```
0 1 2 3 4
```

Capture n and increase by one in each call to lambda function

generate

Assign each element the value returned by function object

```
std::vector<int> v1(5);
int n{0};
auto g = [&n](){return ++n;};

std::generate(v1.begin(),v1.end(), g);

for(auto v : v1) {
    std::cout << v << " ";
}
```

1 2 3 4 5

unique

Move adjacent duplicates to the end of the sequence

```
std::string v1{"aabbacdd"};
auto p = std::unique(v1.begin(), v1.end());

for (auto it = v1.begin(); it != p; ++it) {
    std::cout << *it;
}
```

abacd

unique with sort and erase

Move adjacent duplicates to the end of the sequence

```
std::string v1{"aabbacdd"};
std::sort(v1.begin(), v1.end());
auto p = unique(v1.begin(), v1.end());
v1.erase(p, v1.end());

for (auto v : v1) {
    std::cout << v;
}
```

abcd

Exercises

Do exercise 5

remove_if

Remove elements satisfying a given criteria

```
std::vector<int> v1{1,2,5,3,7,9,6,6};

auto p = std::remove_if(v1.begin(),v1.end(), \
                        [](int &x){return x % 3 == 0;});
v1.erase(p,v1.end());

for(auto v : v1) {
    std::cout << v << " ";
}
```

1 2 5 7

rotate

Treate sequence as a circle and left rotate elements

```
std::vector<int> v1{4,5,1,2,3};

std::rotate(v1.begin(),v1.begin()+2,v1.end());

for(auto v : v1) {
    std::cout << v << " ";
}
```

```
1 2 3 4 5
```

rotate

The element to appear first in the rotated range

Treate sequence as a circle and left rotate elements

```
std::vector<int> v1{4,5,1,2,3};

std::rotate(v1.begin(),v1.begin()+2,v1.end());

for(auto v : v1) {
    std::cout << v << " ";
}
```

```
1 2 3 4 5
```

partition

Move elements fulfilling critera to the head of the sequence

```
std::vector<int> v1{4,500,1,200,3,5,6,700};
auto pred = [](int &x){return x>100;};

std::partition(v1.begin(),v1.end(),pred);

std::copy(v1.begin(),v1.end(), \
  std::ostream_iterator<int>(std::cout, " "));
```

```
700 500 200 1 3 5 6 4
```

stable_partition

Move elements fulfilling critera to the head of the sequence while preserving relative order

```
std::vector<int> v1{4,500,1,200,3,5,6,700};
auto pred = [](int &x){return x>100;};

std::stable_partition(v1.begin(),v1.end(),pred);

std::copy(v1.begin(),v1.end(), \
  std::ostream_iterator<int>(std::cout, " "));
```

```
500 200 700 4 1 3 5 6
```

Exercises

Do exercise 6

next_permutation

Transform sequence into next permutation based on lexicographical order

```
std::string s{"abc"};
bool valid{true};
while(valid) {
    std::cout << s << " ";
    valid = next_permutation(s.begin(),s.end());
}
```

```
abc acb bac bca cab cba
```

fill

Assigns value to sequence

```
std::vector<int> v1(4);

std::fill(v1.begin(),v1.end(),2);

std::copy(v1.begin(),v1.end(), \
  std::ostream_iterator<int>(std::cout, " "));
```

```
2 2 2 2
```

all_of

Check if criteria is true for all elements

```
std::vector<int> v1{3,6,5};
std::vector<int> v2{1,6,2};
auto pred = [](int &x){return x>2;};

std::cout << std::all_of(v1.begin(),v1.end(),pred);
std::cout << " ";
std::cout << std::all_of(v2.begin(),v2.end(),pred);
```

```
1 0
```

count_if

Returns number of elements fulfilling critera

```
std::vector<int> v1{1,6,3};
auto pred = [](int &x){return x>2;};

std::cout << std::count_if(v1.begin(),v1.end(),pred);
```

2

Exercises

Do exercise 7