Lecture 11.11

Object-Oriented Design (2) & Standard Template Library (2)

SE271 Object-Oriented Programming (2020)

Yeseong Kim

Original slides from Prof. Shin at DGIST

Short Notice

- Will upload HW4 today
  - We will review it today

Example: HW4

- Class Design
- Use of namespace
- Abstract class, e.g., platforms
- Predefined classes for ease of usage
- Class Inheritance: unit, pos
- Splitting object and control

Example: HW4

100 Round Simulation

- MyPlayer (Skeleton) vs. ProfFruitTaker
  - MyPlayer wins with 73%
- MyPlayer (Skeleton) vs. ProfNeverDie
  - ProfNeverDie always wins
- ProfFruitTaker vs. ProfNeverDie
  - ProfNeverDie wins with 74%

Today’s Topic

- Containers
  - Set
  - Map
- Algorithm

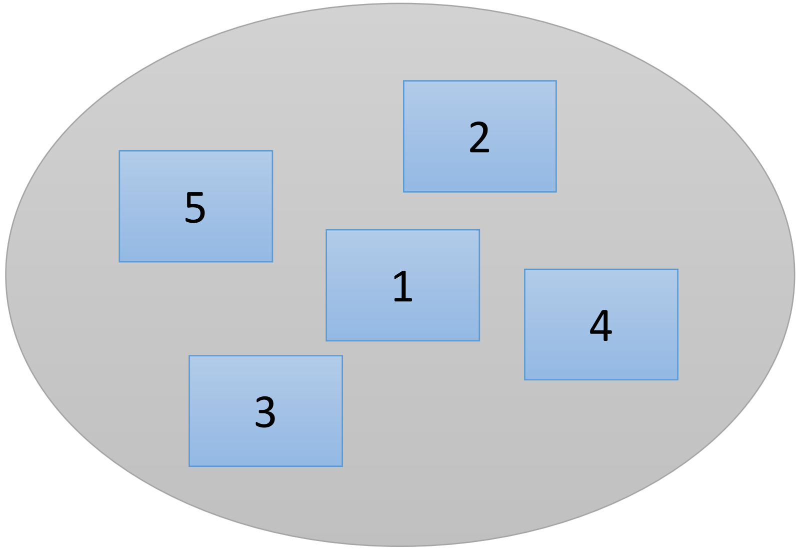
Associative Containers

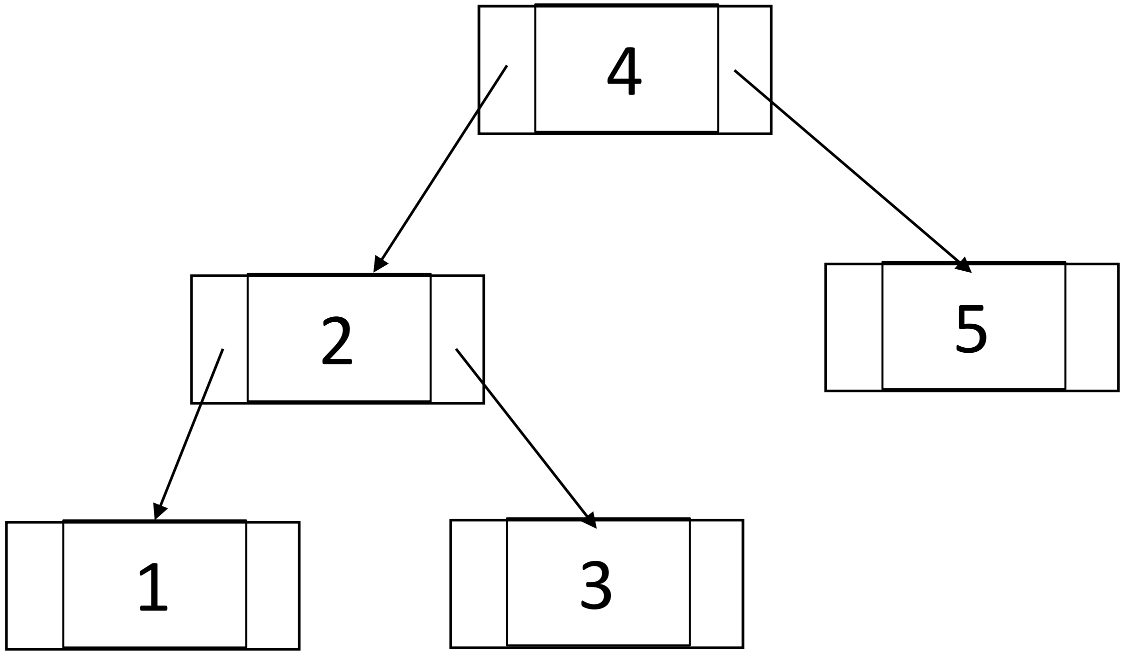
- Set

Node-based containers

- Map

data is stored and ordered
not by input sequence





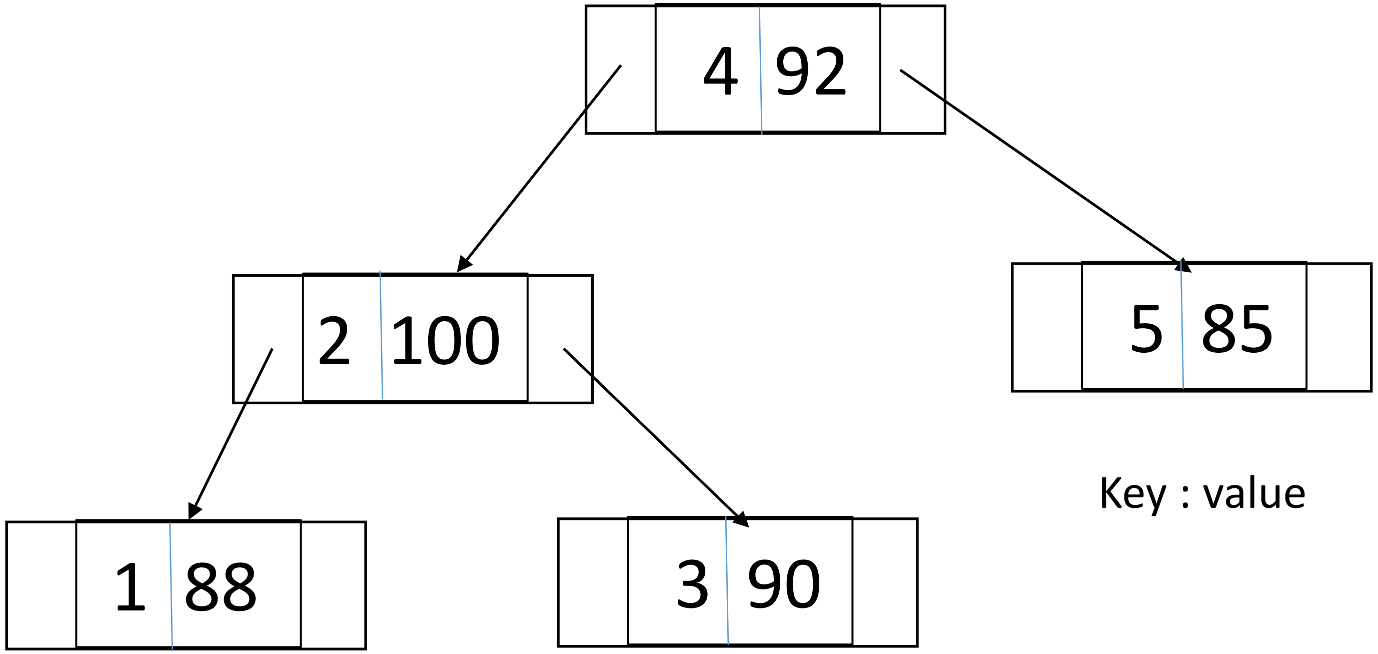
Associative Containers

- Set

Node-based containers

- Map

data is stored and ordered
not by input sequence



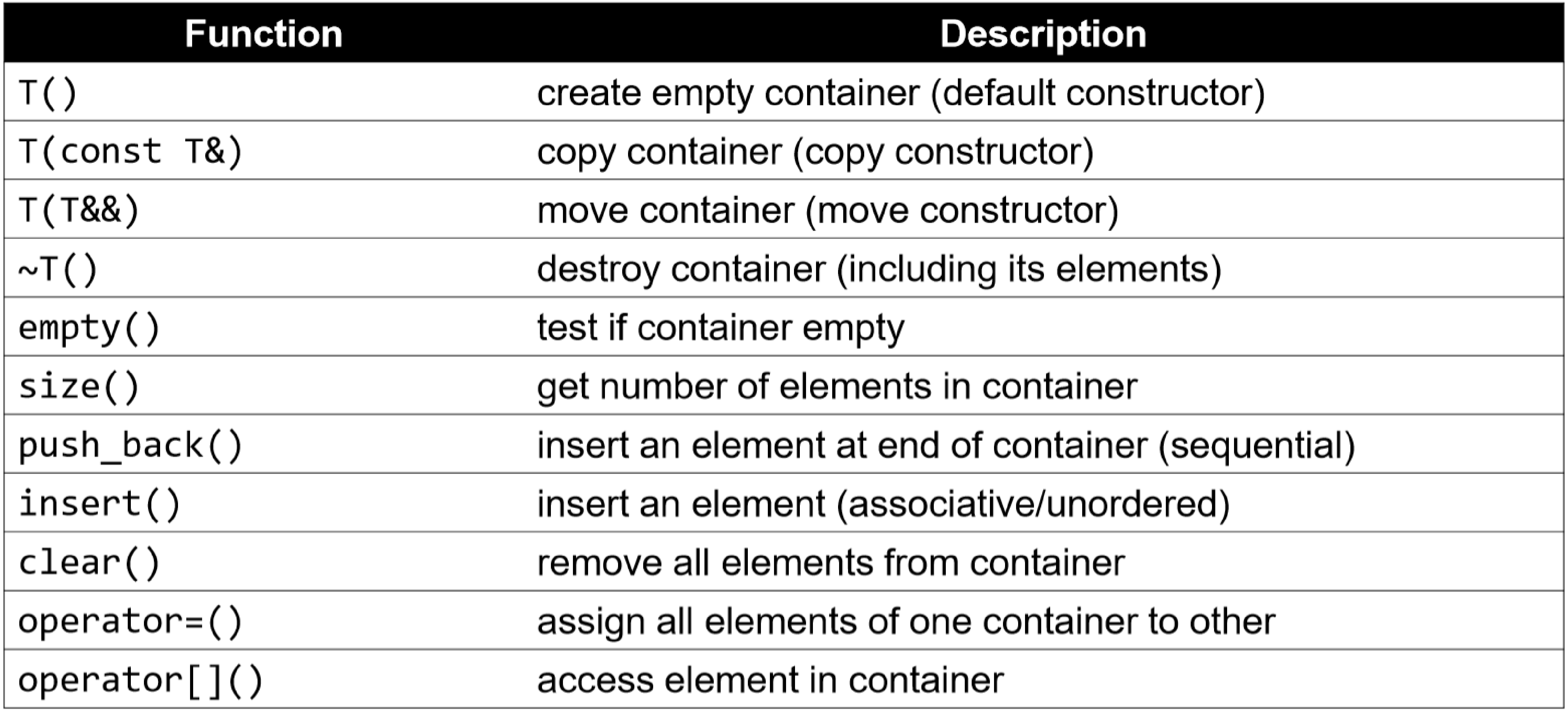
Key : value

Example: Map

```
template<typename Key,
            typename T,
    typename Compare = std::less<key>,
typename Allocator =
std::allocator<std::pair<const Key, T>
>
class map;
```

```
#include<iostream>
#include<map>
using namespace std;
int main() {
   map<string, int> m;
   m.insert(pair<string, int>("Bob", 20));
   m.insert(pair<string, int>("Alice", 22));
   m.insert(pair<string, int>("Carol", 21));
cout << m["Bob"]  << endl;
for (pair<string, int> p : m)
   cout << p.first << ": " << p.second<< endl;
for (map<string, int>::iterator it = m.begin(); it != m.end(); ++it)
   cout << it->first << ": " << it->second << endl;
}
```

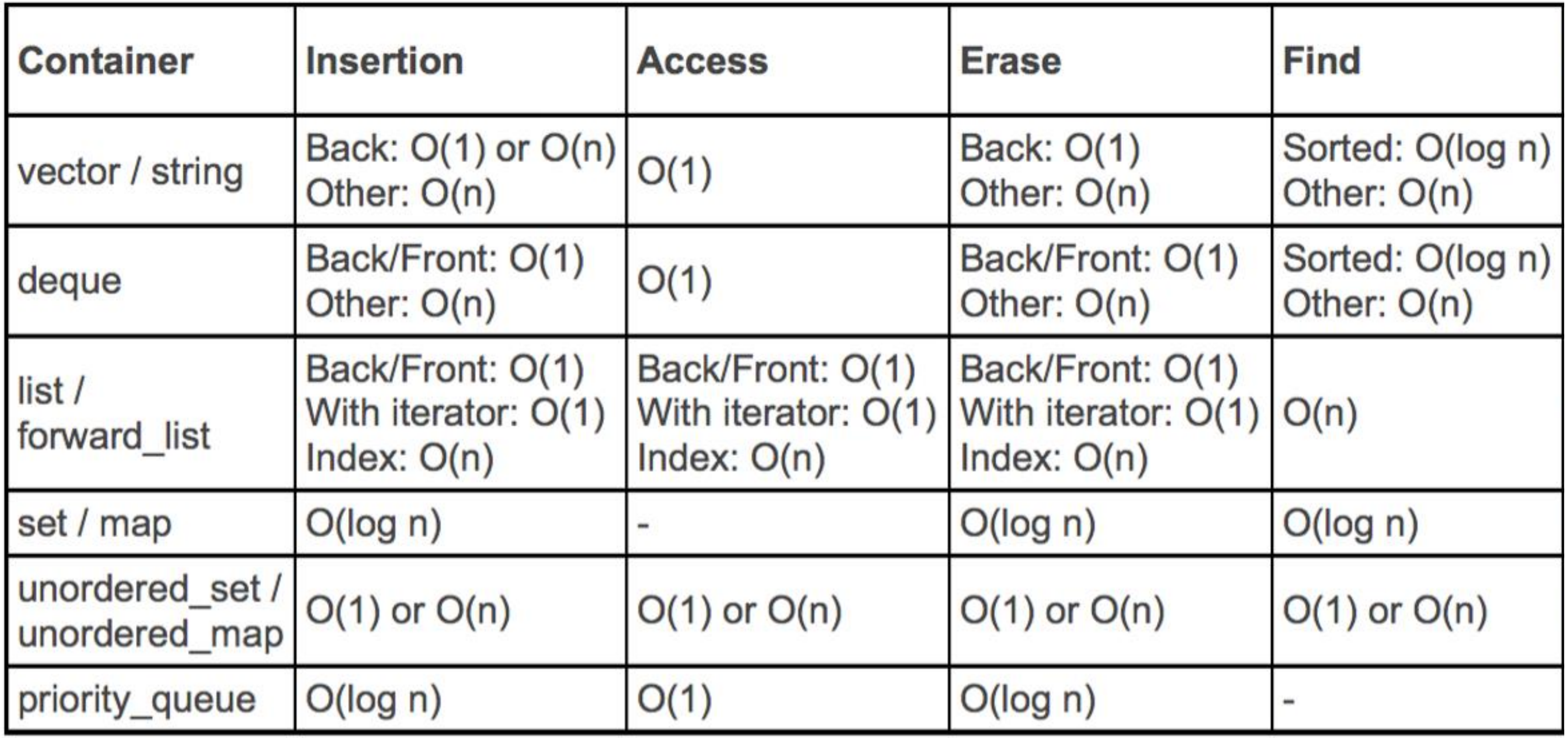
Common operators for STL containers



Function	Description
`T()`	create empty container (default constructor)
`T(const T&)`	copy container (copy constructor)
`T(T&&)`	move container (move constructor)
`~T()`	destroy container (including its elements)
`empty()`	test if container empty
`size()`	get number of elements in container
`push_back()`	insert an element at end of container (sequential)
`insert()`	insert an element (associative/unordered)
`clear()`	remove all elements from container
`operator=()`	assign all elements of one container to other
`operator[]()`	access element in container

Lecture note 2017

Performance

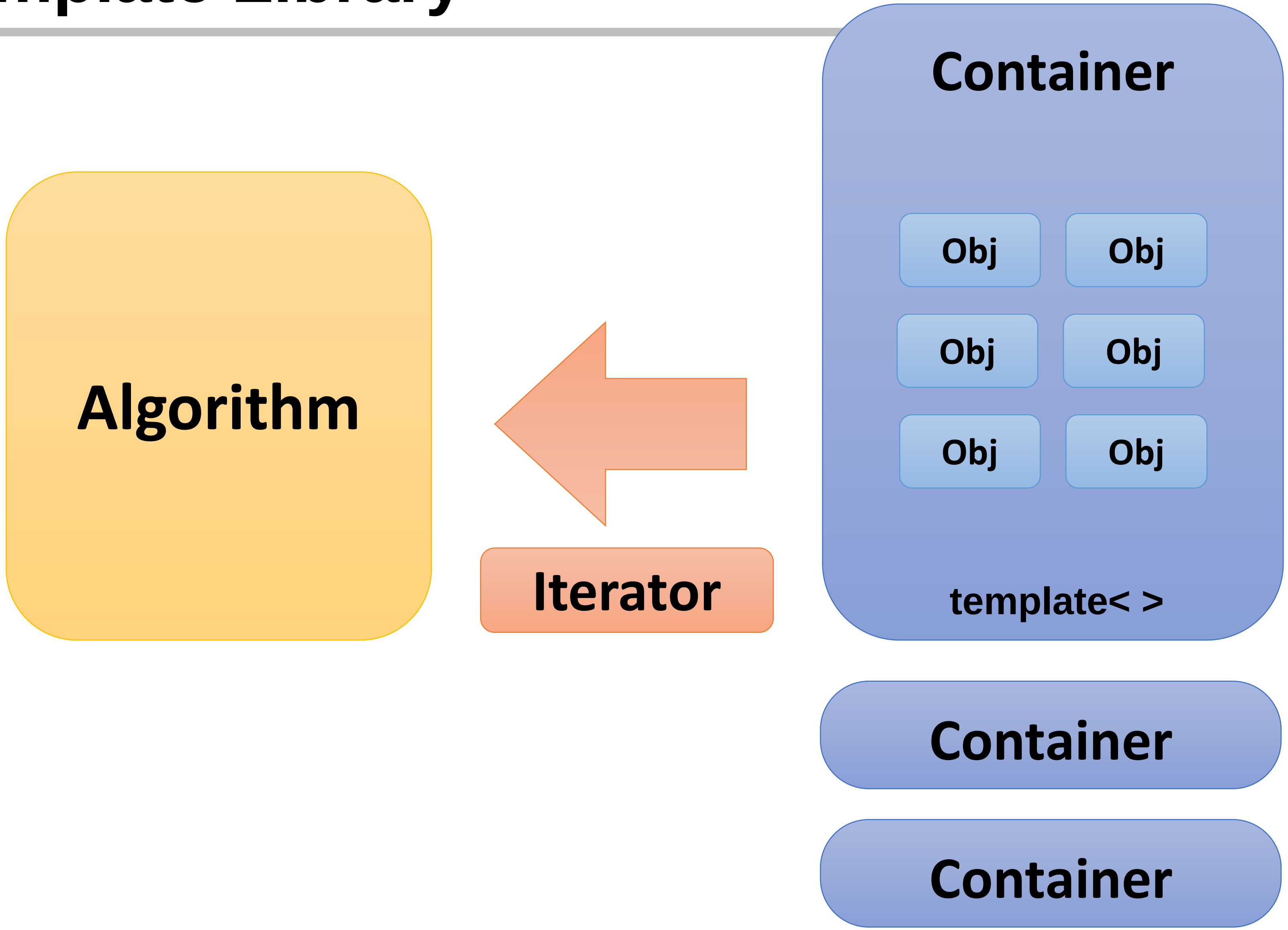


Container	Insertion	Access	Erase	Find
vector / string	Back: O(1) or O(n) Other: O(n)	O(1)	Back: O(1) Other: O(n)	Sorted: O(log n) Other: O(n)
deque	Back/Front: O(1) Other: O(n)	O(1)	Back/Front: O(1) Other: O(n)	Sorted: O(log n) Other: O(n)
list / forward_list	Back/Front: O(1) With iterator: O(1) Index: O(n)	Back/Front: O(1) With iterator: O(1) Index: O(n)	Back/Front: O(1) With iterator: O(1) Index: O(n)	O(n)
set / map	O(log n)	-	O(log n)	O(log n)
unordered_set / unordered_map	O(1) or O(n)	O(1) or O(n)	O(1) or O(n)	O(1) or O(n)
priority_queue	O(log n)	O(1)	O(log n)	-

https://john-ahlgren.blogspot.com/2013/10/stl-container-performance.html

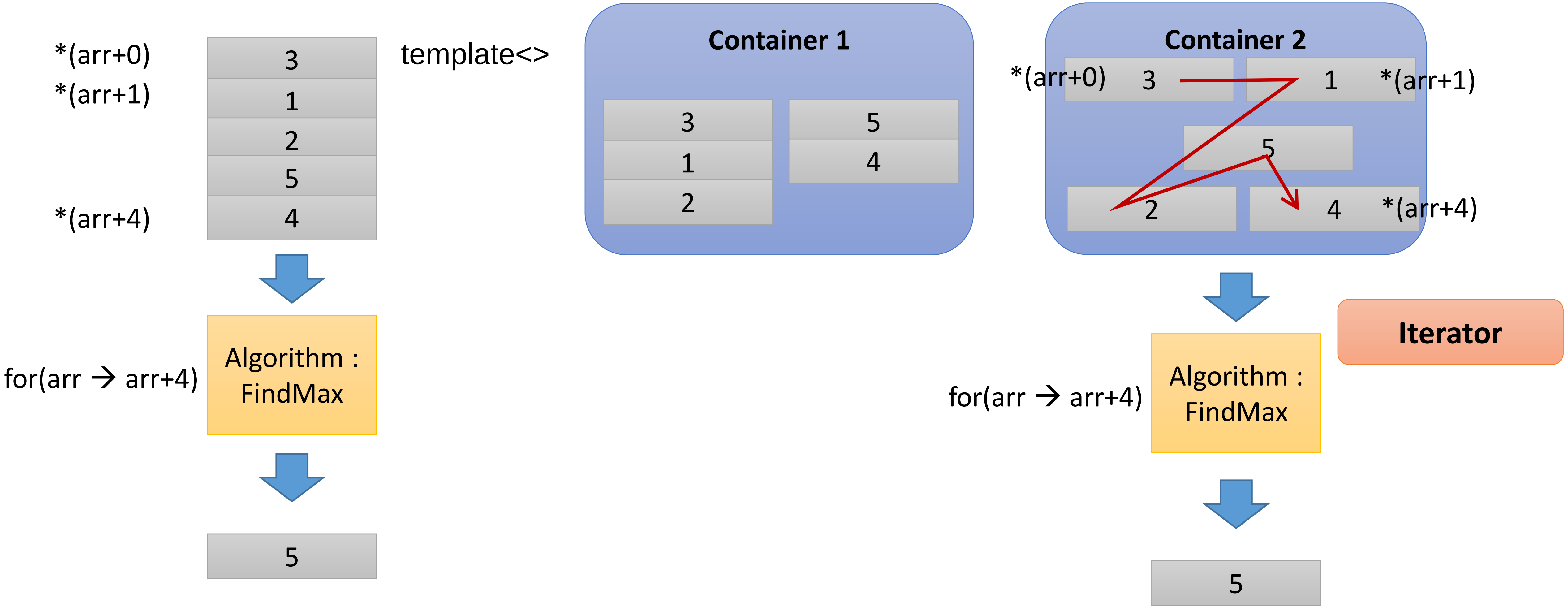
STL: Standard Template Library

- Main components
  - Container
  - Iterator
  - Algorithm
  - Function object
  - Adaptor
  - Allocator



STL: Standard Template Library

int arr[5]={3,1,2,5,4};



*(arr+0)

*(arr+1)

*(arr+4)

3

1

2

5

4

for(arr → arr+4)

Algorithm : FindMax

5

template<>

Container 1

3

1

2

5

4

Container 2

*(arr+0)

3

1

*(arr+1)

5

2

4

*(arr+4)

Iterator

for(arr → arr+4)

Algorithm : FindMax

5

Algorithms

- STL provides most common algorithms (e.g., sort, search) on elements stored in a container
  - An algorithm is a **function template** operating on sequences of elements
- Iterators are used to identify input and/or output
  - Two iterators are often used to specify range of input
  - Algorithm may return an iterator, a value, or modify elements in an output iterator (e.g., copy())
- Some algorithms (e.g., replace(), sort()) modify elements in a container, but no algorithm add or remove elements of a container
- STL library provides generic programming, i.e., a style of computer programming in which algorithms are written in terms of types to-be-specified-later that are then instantiated when needed for specific types provided as parameters

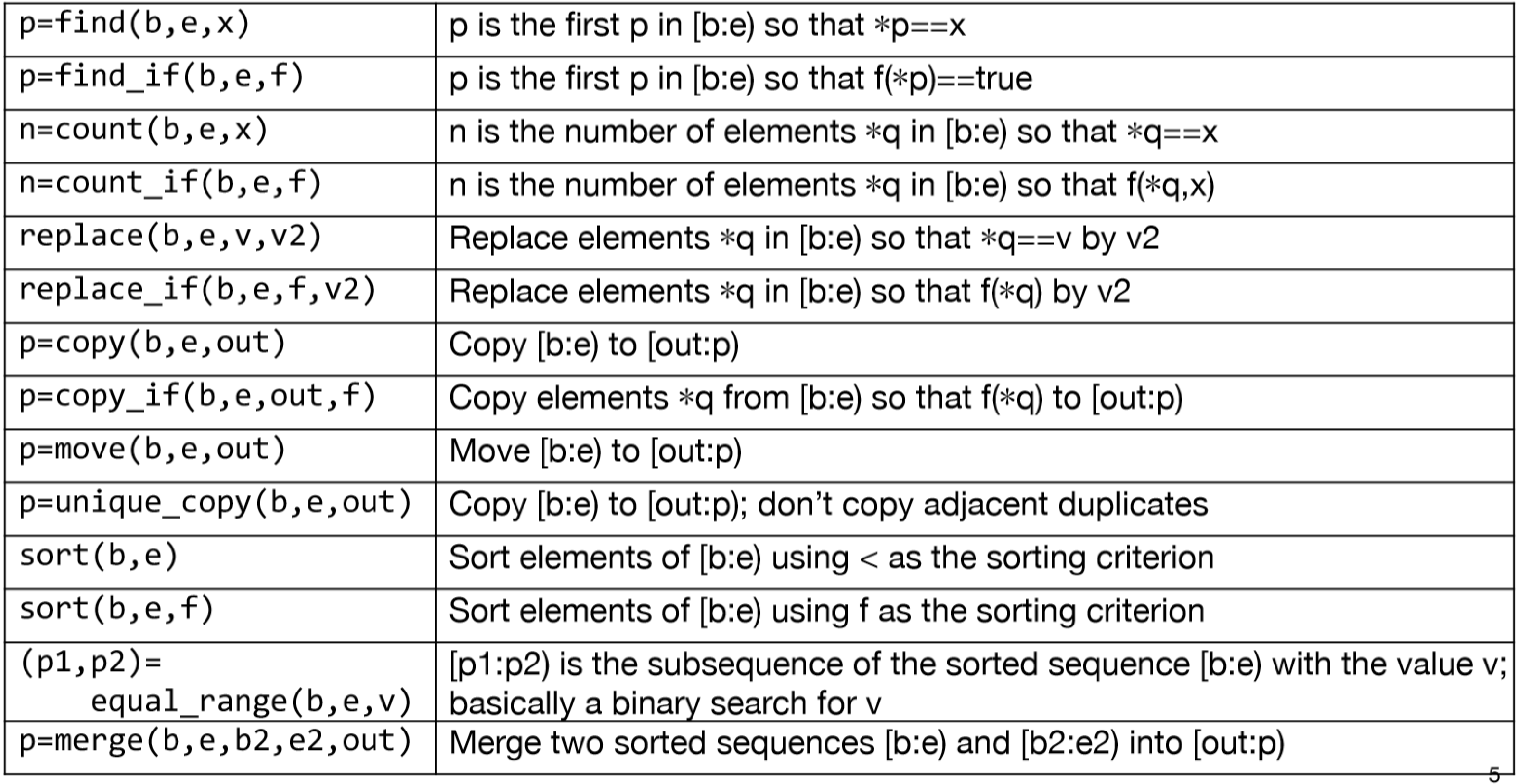
Types of Algorithms

- Non-modifying sequence operations
- Modifying sequence operations
- Partitioning operations
- Sorting operations
- Binary search operations
- Set operations
- Heap operations
- Minimum/maximum operations
- Numeric operations

```
#include <algorithm>
#include <numeric>
```

https://en.cppreference.com/w/cpp/algorithm

Examples



p=find(b,e,x)	p is the first p in [b:e) so that *p==x
p=find_if(b,e,f)	p is the first p in [b:e) so that f(*p)==true
n=count(b,e,x)	n is the number of elements *q in [b:e) so that *q==x
n=count_if(b,e,f)	n is the number of elements *q in [b:e) so that f(*q,x)
replace(b,e,v,v2)	Replace elements *q in [b:e) so that *q==v by v2
replace_if(b,e,f,v2)	Replace elements *q in [b:e) so that f(*q) by v2
p=copy(b,e,out)	Copy [b:e) to [out:p)
p=copy_if(b,e,out,f)	Copy elements *q from [b:e) so that f(*q) to [out:p)
p=move(b,e,out)	Move [b:e) to [out:p)
p=unique_copy(b,e,out)	Copy [b:e) to [out:p); don't copy adjacent duplicates
sort(b,e)	Sort elements of [b:e) using < as the sorting criterion
sort(b,e,f)	Sort elements of [b:e) using f as the sorting criterion
(p1,p2)= equal_range(b,e,v)	[p1:p2) is the subsequence of the sorted sequence [b:e) with the value v; basically a binary search for v
p=merge(b,e,b2,e2,out)	Merge two sorted sequences [b:e) and [b2:e2) into [out:p)

Algorithm: find (p = find(b,e,x))

```
template<typename InputIt, typename T>

constexpr InputIt find(InputIt first, InputIt last,
const T& value) {
    for (; first != last; ++first) {
        if (*first == value) {
            return first;
        }
    }
    return last;
}
```

```
int main() {
    int n1 = 3, n2 = 5;
    std::vector<int> v{ 0, 1, 2, 3, 4 };
    auto result1 = std::find(std::begin(v), std::end(v), n1);
//     auto result2 = std::find(v.begin(), v.end(), n2);

    if (result1 != std::end(v))
        std::cout << "v contains: " << n1 << '\n';
    else
        std::cout << "v does not contain: " << n1 << '\n';
}
```

Algorithm: sort (sort(b,e) or sort(b,e,f))

```
#include <iostream>
#include <algorithm>
#include <vector>

int main(){
    std::vector<int> v{ 3, -4, 5, -6, 10 };
    std::sort(v.begin()+1, v.end()-1);
    for (auto i : v) {
        std::cout << i << " ";
    }
}
```

```
3 -6 -4 5 10
```

References

- Learn C++ (https://www.learncpp.com/)
  - STL: Ch. 16

- STL
  - https://en.cppreference.com/w/cpp/algorithm

ANY QUESTIONS?