Key STL Features: Containers, Iterators, & Algorithms

Container class: Holds elements & iterators

Functor class: result used to guide algorithm

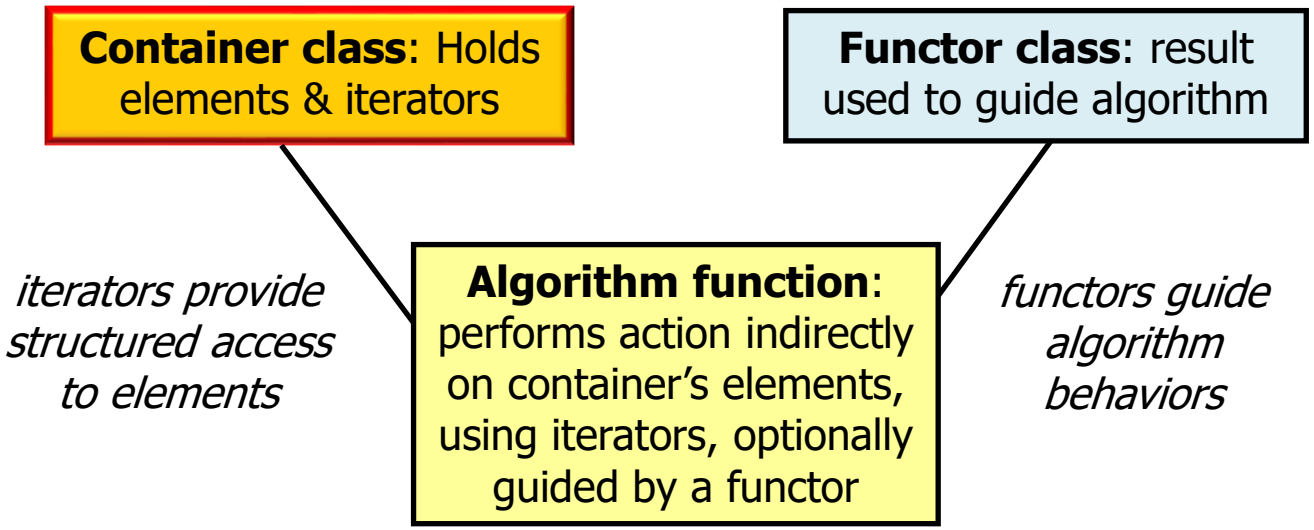
iterators provide structured access to elements

Algorithm function: performs action indirectly on container's elements, using iterators, optionally guided by a functor

functors guide algorithm behaviors

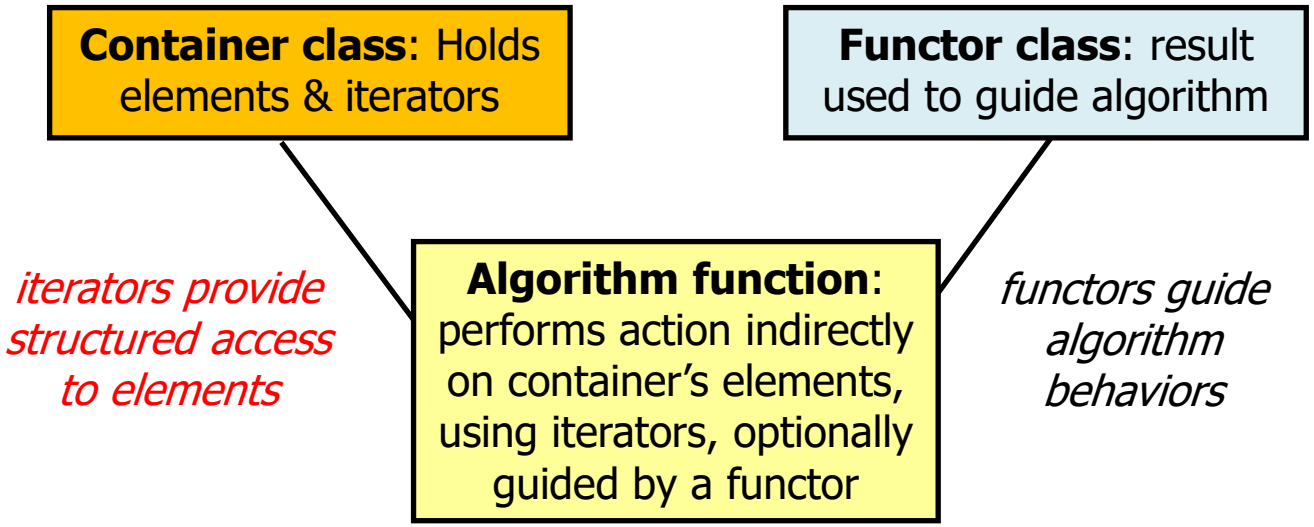
Key STL Features: Containers, Iterators, & Algorithms

- **Containers**
  - *Sequential*: vector, deque, list
  - *Associative*: set, multi set, map, multimap
  - *Adapters*: stack, queue, priority queue



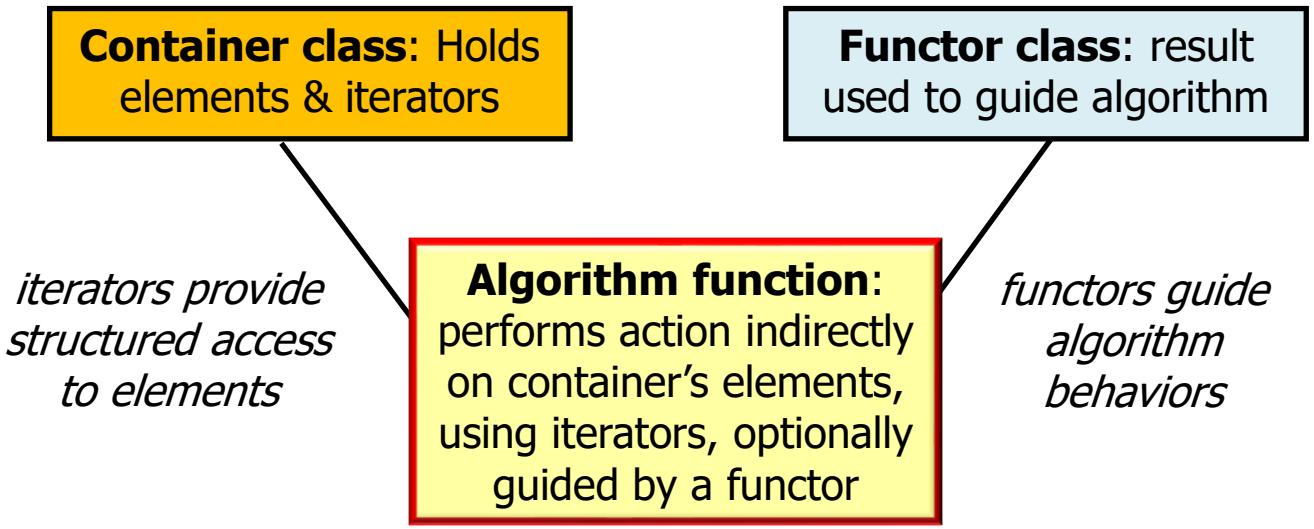
Key STL Features: Containers, Iterators, & Algorithms

- Containers
- **Iterators**
  - Input, output, forward, bidirectional, & random access
  - Each container declares a trait for the type of iterator it provides



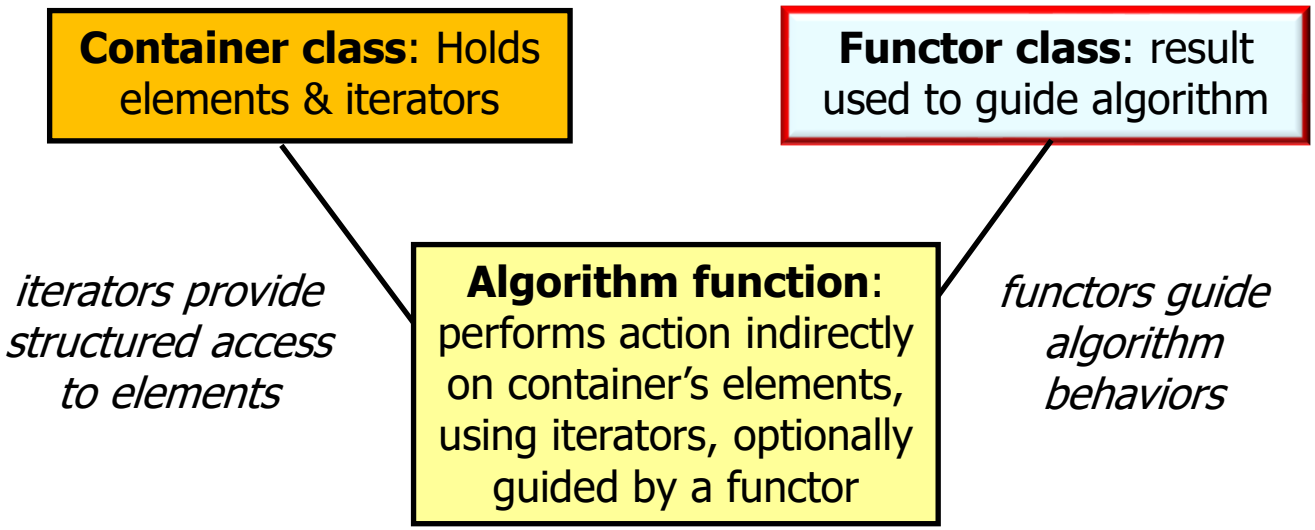
Key STL Features: Containers, Iterators, & Algorithms

- Containers
- Iterators
- **Generic Algorithms**
  - Mutating, non-mutating, sorting, & numeric



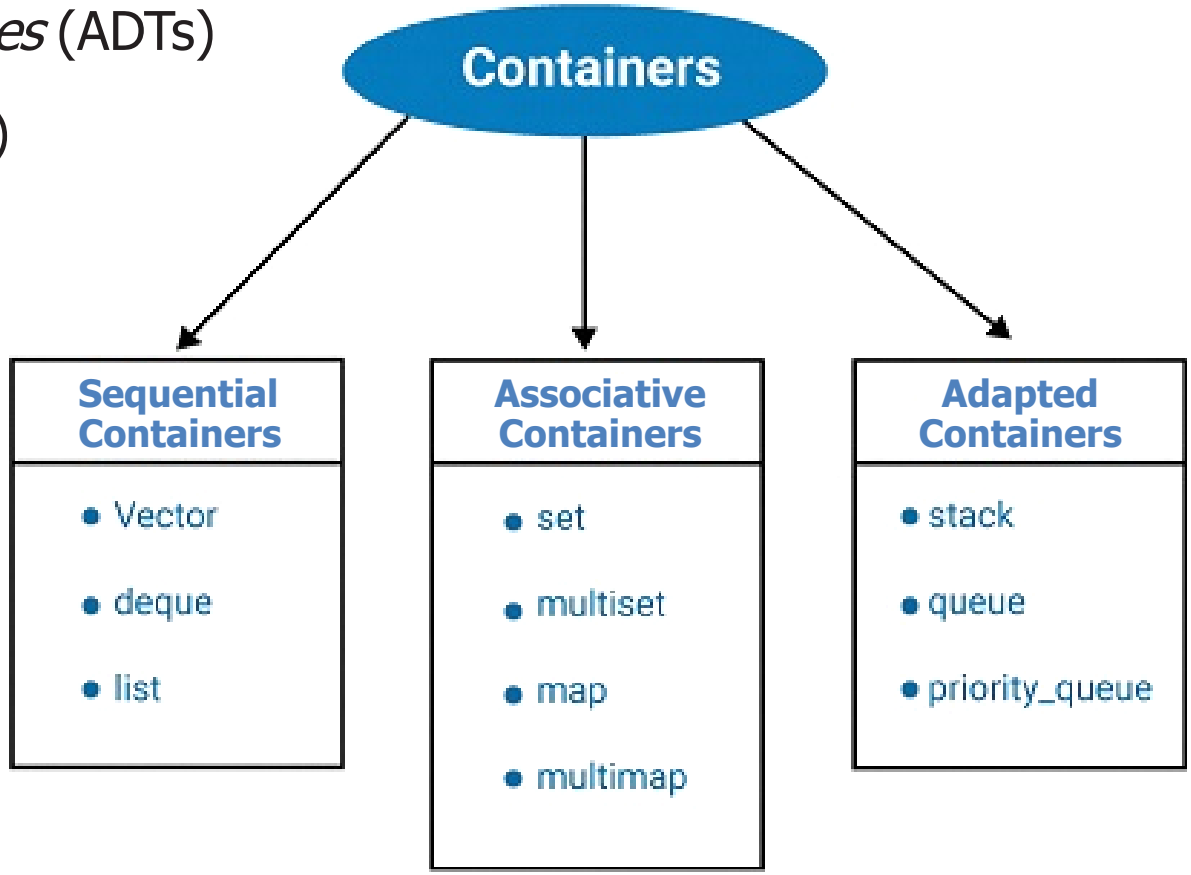
Key STL Features: Containers, Iterators, & Algorithms

- Containers
- Iterators
- Generic Algorithms
- **Functors**
  - Objects that define `operator()` & act like functions



STL Container Overview

- STL containers are *Abstract Data Types* (ADTs)
  - A set of values & a set of (cohesive) operations on these values



See en.wikipedia.org/wiki/Abstract_data_type

STL Container Overview

- All containers are parameterized by the type(s) they contain

```
template <typename T,
          typename Container =
            deque<T>>
class stack {
public:
  explicit stack(const Container&);
  bool empty() const;
  size_type size() const;
  value_type& top();
  const value_type& top() const;
  void push(const value_type& t);
  void pop();
private :
   Container container_ ;
   // …
};
```

STL Container Overview

- Each container declares various traits
  - *e.g.*, `iterator`, `const_iterator`, `value_type`, *etc.*

Member type	Definition
value_type	T
allocator_type	Allocator
size_type	Unsigned integer type (usually std::size_t)
difference_type	Signed integer type (usually std::ptrdiff_t)
reference	
Allocator::reference	(until C++11)
value_type&	(since C++11)
const_reference	
Allocator::const_reference	(until C++11)
const value_type&	(since C++11)
pointer	
Allocator::pointer	(until C++11)
std::allocator_traits<Allocator>::pointer	(since C++11)
const_pointer	
Allocator::const_pointer	(until C++11)
std::allocator_traits<Allocator>::const_pointer	(since C++11)
iterator	*LegacyRandomAccessIterator*
const_iterator	Constant *LegacyRandomAccessIterator*
reverse_iterator	std::reverse_iterator<iterator>
const_reverse_iterator	std::reverse_iterator<const_iterator>

STL Container Overview

- Each container provides factory methods for creating iterators:
  - **`begin()`** & **`end()`** for traversing from front to back
  - **`rbegin()`** & **`rend()`** for traversing from back to front

Types of STL Containers

- There are several types of STL containers

Types of STL Containers

- There are several types of STL containers
  - **Sequential containers** that arrange the data they contain in a linear manner
    - Element order has nothing to do with their value
    - Similar to builtin arrays, but needn't be stored contiguously

Category	Containers	Characteristics
Sequential	vector	Linear and contiguous storage like an array that allows fast insertions and removals at the end only.
	list	Doubly linked list that allows fast insertions and removals anywhere.
	forward_ list	Single linked list that allows fast insertions and removals anywhere.
	deque	Linear but non-contiguous storage that allows fast insertions and removals at both ends.

Types of STL Containers

- There are several types of STL containers
  - **Sequential containers** that arrange the data they contain in a linear manner
  - **Ordered associative containers** that support efficient operations on elements using keys ordered by `operator<`
    - Implemented as balanced binary trees

Category	Containers	Characteristics
Ordered associative	**set**	**Defines where the elements' values are the keys and duplicates *are not* allowed. It has fast lookup using the key,**
	multiset	**Defines where the elements' values are the keys and duplicates *are* allowed. It has fast lookup using the key,**
	map	**Key-to-value mapping where a single key can only be mapped to one value,**
	multimap	**Key-to-value mapping where a single key can be mapped to many values.**

Types of STL Containers

- There are several types of STL containers
  - **Sequential containers** that arrange the data they contain in a linear manner
  - **Ordered associative containers** that support efficient operations on elements using keys ordered by `operator<`
  - **Unordered associative containers** that maintain data in structures suitable for fast associative operations
    - Implemented as hash tables

Category	Containers	Characteristics
Unordered associative	unordered_set	Defines where the elements' values are the keys and duplicates *are not* allowed. It has fast lookup using the key,
	unordered_multiset	Defines where the elements' values are the keys and duplicates *are* allowed. It has fast lookup using the key,
	unordered_map	Key-to-value mapping where a single key can only be mapped to one value,
	unordered_multimap	Key-to-value mapping where a single key can be mapped to many values.

Types of STL Containers

- There are several types of STL containers
  - **Sequential containers** that arrange the data they contain in a linear manner
  - **Ordered associative containers** that support efficient operations on elements using keys ordered by `operator<`
  - **Unordered associative containers** that maintain data in structures suitable for fast associative operations
  - **Adapters** that provide alternative access sequential & associative containers

Category	Containers	Characteristics
Adapter	stack	First in, last out data structure.
	queue	First in, first out data structure.
	priority_ queue	Queue that maintains items in a sorted order based on a priority value.