STL

C++'s Standard Containers Library

Matthew Barulic

RoboJackets
Spring Training Series
Georgia Tech

February 16, 2018

Today's Plan

Introduction to the STL

Brief History

- 1979 - C++ Invented
- 1992 - STL Created
- 1998 - First Standardization

The STL

Standard **T**emplate **L**ibrary

- Containers
- Iterators
- Algorithms
- Function Objects

The STL

Algorithms $\rightarrow$ Iterators $\rightarrow$ Containers

Containers

- Store data (objects / primitives)
- The *Data Structures* of
  Data Structures and Algorithms (CS 1332)
- Minimal member methods for managing contents

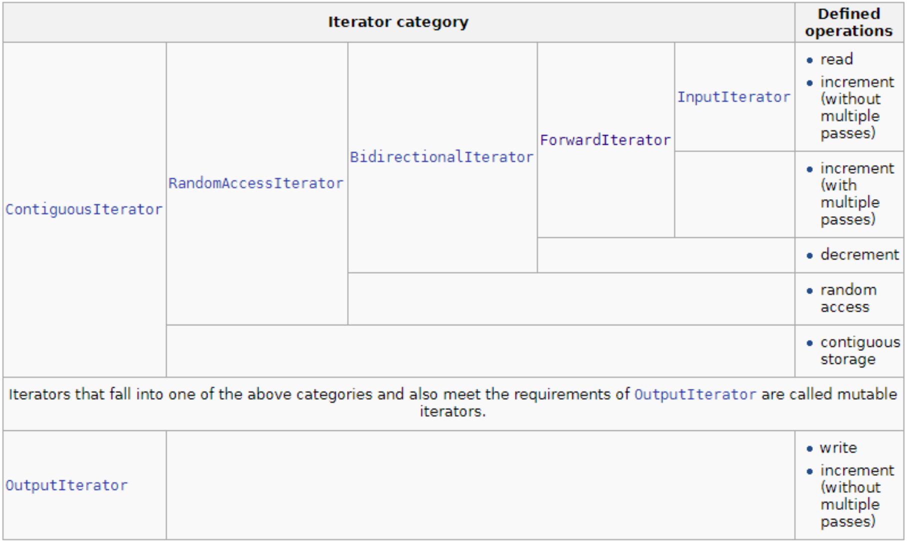
`http://en.cppreference.com/w/cpp/container`

Iterators

- Interface for useful container operations
- Exposed through begin() / end()
  (and their variants)

`http://en.cppreference.com/w/cpp/iterator`

Iterators



Iterator category					Defined operations
`ContiguousIterator`	`RandomAccessIterator`	`BidirectionalIterator`	`ForwardIterator`	`InputIterator`	• read • increment (without multiple passes)
					• increment (with multiple passes)
					• decrement
					• random access
					• contiguous storage
Iterators that fall into one of the above categories and also meet the requirements of `OutputIterator` are called mutable iterators.					
`OutputIterator`					• write • increment (without multiple passes)

Algorithms

- Utility functions for ranges of elements
- Decoupled from specific containers

`http://en.cppreference.com/w/cpp/algorithm`

Algorithms

Categories of Algorithms

- Non-Modifying
- Modifying
- Sorting / Partitioning
- Numeric

How to read cppreference.com

cppreference.com

`http://en.cppreference.com`

Common Containers

array

- Fixed size, sequence container
- Preferred over "c-style" arrays
- Two template arguments: *type* and *size*

Example

```
std::array<int,5> my_array = {1,2,3,4,5};
my_array.size(); // 5
my_array[0]; // 1
```

http://en.cppreference.com/w/cpp/container/array

vector

- Variable length, sequence container
- Likely the most frequently used container
- Requires one template argument for *type*

Example

```
std::vector<int> my_vector = {1,2,3,4,5};
my_vector.size(); // 5
my_vector[0]; // 1
my_vector.push_back(6); // size is now 6
```

http://en.cppreference.com/w/cpp/container/vector

set

- Associative container of unique elements
- Can use custom compare functions
- Usually a Red-Black Tree under the hood

Example

```
std::set<std::string> my_set = {"cat","dog","horse"};
my_set.insert("bunny"); // adds "bunny"
my_set.insert("cat"); // does nothing
```

http://en.cppreference.com/w/cpp/container/set

map

- Associative container for key-value pairs
- Keys must be unique
- Also usually Red-Black Trees
  (This time with pairs.)

Example

```
std::map<std::string,short> my_map = {{"orange",0xFF7F00}};
my_map["Jazzberry Jam"] = 0xA50B5E; // Creates new pair
my_map["orange"]; // 0xFF7F00
```

http://en.cppreference.com/w/cpp/container/map

Common Algorithms

any_of / all_of / none_of

copy

fill

generate

accumulate

transform

Code Examples