Mittuniversitetet
MID SWEDEN UNIVERSITY

Lecture #7

STL Containers

chapter: 3 – 3.4

Objectives

Mittuniversitetet
MID SWEDEN UNIVERSITY

- To know the relationships among containers, iterators, and algorithms.
- To distinguish sequence containers, associative containers, and container adapters.
- To distinguish some containers vector, map, multimap
- To use common features of some containers.
- To access elements in a container using iterators.
- To distinguish iterator types: input, output, forward, bidirectional, and random-access.
- To manipulate iterators using operators.
- To obtain iterators from containers and know the type of interators supported by containers.
- To perform input and output using istream_iterator and ostream_iterator.
- To store, retrieve and process elements in sequence containers: vector.

Three components of STL

Mittuniversitetet
MID SWEDEN UNIVERSITY

Containers: Classes in the STL are container classes. A container object such as a vector is used to store a collection of data, often referred to as *elements*.

Iterators: The STL container classes make extensive use of iterators, which are objects that facilitate traversing through the elements in a container. Iterators are like built-in pointers that provide a convenient way to access and manipulate the elements in a container.

Algorithms: Algorithms are used in the functions to manipulate data such as sorting, searching, and comparing elements. There are about 80 algorithms implemented in the STL. Most of these algorithms use iterators to access the elements in the container.

Mittuniversitetet
MID SWEDEN UNIVERSITY

Sequence, Associative Containers

- The sequence containers (also known as sequential containers) represent linear data structures. The three sequence containers are vector, list, and deque (pronounced deck).

- Associative containers are non-linear containers that can locate elements stored in the container quickly. Such containers can store sets of values or *key/value* pairs (pair object). The four associative containers are set, multiset, map, and multimap.

Container Classes

Mittuniversitetet
MID SWEDEN UNIVERSITY

STL Container	Header File	Applications
vector	<vector>	For direct access to any element, and quick insertion and deletion at the end of the vector.
deque	<deque>	For direct access to any element, quick insertion and deletion at the front and end of the deque.
list	<list>	For rapid insertion and deletion anywhere.
set	<set>	For direct lookup, no duplicated elements.
multiset	<set>	Same as set except that duplicated elements allowed.
map	<map>	Key/value pair mapping, no duplicates allowed, and quick lookup using the key.
multimap	<map>	Same as map, except that keys may be duplicated
stack	<stack>	Last-in/first-out container.
queue	<queue>	First-in/first-out container.
priority_queue	<queue>	The highest-priority element is removed first.

Common Functions to All Containers

Functions	Description
non-arg constructor	Constructs an empty container.
constructor with args	In addition to the non-arg constructor, every container has several constructors with args.
copy constructor	Creates a container by copying the elements from an existing container of the same type.
destructor	Performs cleanup after the container is destroyed.
empty()	Returns true if there are no elements in the container.
size()	Returns the number of elements in the container.
operator=	Copies one container to another.
Relational operators (<, <=, >, >=, ==, and !=)	The elements in the two containers are compared sequentially to determine the relation.

Common Functions to First-Class Containers

Mittuniversitetet
MID SWEDEN UNIVERSITY

STL Functions	Description
c1.swap(c2)	Swaps the elements of two containers c1 and c2.
c.max_size()	Returns the maximum number of elements a container can hold.
c.clear()	Erases all elements from the container.
c.begin()	Returns an iterator to the first element in the container.
c.end()	Returns an iterator that refers to the next position after the end of the container.
c.rbegin()	Returns an iterator to the last element in the container for processing elements in reverse order.
c.rend()	Returns an iterator that refers to the position before the first element in the container.
c.erase(beg, end)	Erases the elements in the container from beg to end-1. Both beg and end are iterators.

Sequence Containers

The STL provides three sequence containers: vector, list, and deque. The vector and deque containers are implemented using arrays and the list container is implemented using a linked list.

Common Functions in Sequence Containers

Mittuniversitetet
MID SWEDEN UNIVERSITY

Functions	Description
assign(n, elem)	Assign n copies of the specified element in the container.
assign(beg, end)	Assign the elements in the range from iterator beg to iterator end.
push_back(elem)	Appends an element in the container.
pop_back()	Removes the last element from the container.
front()	Returns the reference of the first element.
back()	Returns the reference of the last element.
insert(position, elem)	Inserts an element at the specified iterator.

Sequence Containers: vector

Mittuniversitetet
MID SWEDEN UNIVERSITY

Functions	Description
vector(n, element)	Constructs a vector filled with n copies of the same element.
vector(beg, end)	Constructs a vector initialized with elements from iterator beg to end.
vector(size)	Constructs a vector with the specified size.
at(index): dataType	Returns the element at the specified index.

Simple Demo

Mittuniversitetet
MID SWEDEN UNIVERSITY

This simple example demonstrates how to create a vector

SimpleSTLDemo

Iterators

Mittuniversitetet
MID SWEDEN UNIVERSITY

Iterators are used extensively in the first-class containers for accessing and manipulating the elements. As you already have seen earlier, several functions (e.g., begin() and end()) in the first-class containers are related to iterators.

IteratorDemo

Constant Iterator Demo (optional)

Mittuniversitetet
MID SWEDEN UNIVERSITY

ConstIteratorDemo

ReverseIteratorDemo

Iterator Types Supported by Containers

Mittuniversitetet
MID SWEDEN UNIVERSITY

STL Container	Type of Iterators Supported
vector	random access iterators
deque	random access iterators
list	bidirectional iterators
set	bidirectional iterators
multiset	bidirectional iterators
map	bidirectional iterators
multimap	bidirectional iterators
stack	no iterator support
queue	no iterator support
priority_queue	no iterator support

Operators Supported by Iterators

Operator	Description
All iterators	
++p	Preincrement an iterator.
p++	Postincrement an iterator.
Input iterators	
*p	Dereference an iterator (used as rvalue).
p1 == p2	Evaluates true if p1 and p2 point to the same element.
p1 != p2	Evaluates true if p1 and p2 point to different elements.
Output iterators	
*p	Dereference an iterator (used as lvalue).
Bidirectionl iterators	
--p	Predecrement an iterator.
p--	Postdecrement an iterator.
Random-access iterators	
p += i	Increment iterator p by i positions.
p -= i	Decrement iterator p by i positions.
p + i	Returns an iterator ith position after p.
p - i	Returns an iterator ith position before p.
p1 < p2	Returns true if p1 is before p2.
p1 <= p2	Returns true if p1 is before or equal to p2.
p1 > p2	Returns true if p1 is after p2.
p1 >= p2	Returns true if p1 is after p2 or equal to p2.
p[i]	Returns the element at the position p offset by i.

Printing out the content of a vector

Mittuniversitetet
MID SWEDEN UNIVERSITY

Even though Iterators are used extensively in the first-class containers for accessing and manipulating the elements. We can use other methods to pring out the content of a vector.

VectorDemo

Creating a Matrix with vector

Mittuniversitetet
MID SWEDEN UNIVERSITY

We can use vector in a vector to create a matrix.

TwoDimVector

Vectors and Matrix as in-parameters

Mittuniversitetet
MID SWEDEN UNIVERSITY

TwoDimVector

Reading from file into a vector

Mittuniversitetet
MID SWEDEN UNIVERSITY

ReadFileToVector

Thank you