Lecture #9
STL Containers (cont.)

chapter:9

Sequence Containers

- **vector, array**

  1. Vector as string uses contiguous memory
  2. Use array if you need fixed-size elements
  3. Random access through indexing

  Use vector unless you have a good reason not to!

- **deque:**
  1. Fast random access like vector and string
  2. If your program needs to insert or delete elements at the front and the back but NOT in the middle use deque.

The best alternative for random access is vector or deque

Sequence Containers

- **forward_list , list**

  1. Are the best alternatives to the best user-defined singel-linked (forward_list) and double-linked lists
  2. Fast to add and remove an element anywhere in the container.
  3. No support for random access➔ To access an element we need to iterate through the container.
  4. Compared to vector, deque and array the memory overhead is substantial.

If your program needs to insert/delete elements in the middle of the container , use these.

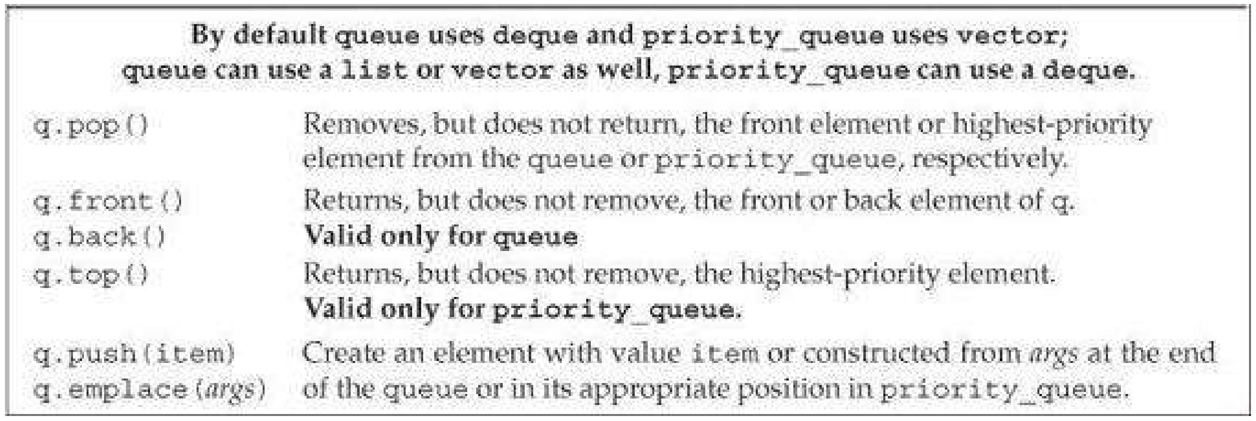
forward_list

list

Sequence Container adapters

- **Priority_queue**

1. Is implemented on vector
2. Establishes a priority among the elements held in the queue
3. The highest element always comes first
4. Newly added elements are placed ahead of all the elements with a lower priority
5. Requires random access



By default queue uses deque and `priority_queue` uses `vector`;
queue can use a `list` or `vector` as well, `priority_queue` can use a `deque`.

`q.pop()`	Removes, but does not return, the front element or highest-priority element from the queue or `priority_queue`, respectively.

`q.front()`
`q.back()`
	Returns, but does not remove, the front or back element of q.
	Valid only for queue

`q.top()`	Returns, but does not remove, the highest-priority element. **Valid only for `priority_queue`.**

`q.push(item)`
`q.emplace(`*args*`)`
	Create an element with value `item` or constructed from *args* at the end of the queue or in its appropriate position in `priority_queue`.

Sequence Container adapters

- **Priority_queue**

Example Uses:

- **Simulations** (events are ordered by the time at which they should be executed)
- **Job scheduling in computer systems** (higher priority jobs should be executed first)
- **Constraint systems** (higher priority constraints should be satisfied before lower priority constraints)

Priority_queue

Thank you