**std::queue**

The std::queue class is a container adapter that gives the programmer the functionality of a queue - specifically, a FIFO (first-in, first-out) data structure.

The class template acts as a wrapper to the underlying container - only a specific set of functions is provided. The queue pushes the elements on the back of the underlying container and pops them from the front.

**Important Member functions**

**front**

access the first element

**back**

access the last element

**empty**

checks whether the underlying container is empty

**size**

returns the number of elements

**push**

inserts element at the end

**pop**

removes the first element

**std::deque  
  
Important Member functions**

**front**

access the first element

**back**

access the last element

**empty**

checks whether the underlying container is empty

**size**

returns the number of elements

**push\_back**

inserts element at the end

**pop\_front**

removes the first element

**push\_front**

**pop\_back**

**std::priority\_queue**

Priority queues are a type of container adapters, specifically designed such that the first element of the queue is the greatest of all elements in the queue and elements are in non decreasing order(hence we can see that each element of the queue has a priority{fixed order}).

The functions associated with priority queue are:

**empty() –** Returns whether the queue is empty

**size() –** Returns the size of the queue

**top() –** Returns a reference to the top most element of the queue

**push(g) –** Adds the element ‘g’ at the end of the queue

**pop() –** Deletes the first element of the queue

**An Example:**

#include <iostream>

#include <queue>

using namespace std;

void showpq(priority\_queue <int> gq)

{

priority\_queue <int> g = gq;

while (!g.empty())

{

cout << '\t' << g.top();

g.pop();

}

cout << '\n';

}

int main ()

{

priority\_queue <int> gquiz;

gquiz.push(10);

gquiz.push(30);

gquiz.push(20);

gquiz.push(5);

gquiz.push(1);

cout << "The priority queue gquiz is : ";

showpq(gquiz);

cout << "\ngquiz.size() : " << gquiz.size();

cout << "\ngquiz.top() : " << gquiz.top();

cout << "\ngquiz.pop() : ";

gquiz.pop();

showpq(gquiz);

return 0;

}

Now, it is not java priority queue where you can either pass a user defined implementation of comparable interface or elements will be dequeued in the natural order (the order in which those are entered) Here, the greatest element will be the first element of the priority\_queue and it will be popped.