**Queues**

The class queue<> implements a queue (also known as FIFO). A queue serves as a classic data buffer.

header file

#include <queue>

The class queue is defined as follows:

namespace std {

template < typename T,

typename Container = deque<T>>

class queue;

}

**T:** Type of the elements

**Container:** Optional second template parameter defines the container that the queue uses internally for its elements. The default container is a deque.

The queue implementation simply maps the operations into appropriate calls of the container that is used internally.

You can use any sequence container class that provides the member functions front(), back(), push\_back(), and pop\_front().

# The Core Interface

**push()** inserts an element into the queue.

**front()** returns the next element in the queue (the element that was inserted first) but does not remove it.

**back()** returns the last element in the queue (the element that was inserted last) but does not remove it.

**pop()** removes an element from the queue but does not return it.

You must always call front() and pop() to process and remove the next element from the queue. Inconvenient interface, but it performs better if you want to only remove the next element without processing it.

Behavior of front(), back(), and pop() is undefined if the queue contains no elements.

Member functions size() and empty() are provided to check whether the queue contains elements.

You can easily write a more convenient interface.

# Class queue<> in Detail

The queue<> interface maps more or less directly to corresponding container members

# Example

#include <iostream>

#include <queue>

#include <string>

using namespace std;

int main() {

queue<string> q;

// insert three elements into the queue

q.push("These ");

q.push("are ");

q.push("more than ");

// read and print two elements from the queue

cout << q.front(); q.pop();

cout << q.front(); q.pop();

// insert two new elements

q.push("four ");

q.push("words!");

// skip one element

q.pop();

// read and print two elements

cout << q.front(); q.pop();

cout << q.front() << endl; q.pop();

// print number of elements in the queue

cout << "number of elements in the queue: " << q.size() << endl;

return 0;

}

Output:

These are four words!

number of elements in the queue: 0

# END