



4.11. The Queue Abstract Data Type

The queue abstract data type is defined by the following structure and operations. A queue is structured, as described above, as an ordered collection of items which are added at one end, called the “rear,” and removed from the other end, called the “front.” Queues maintain a FIFO ordering property. The queue operations are given below.

- `Queue()` creates a new queue that is empty. It needs no parameters and returns an empty queue.
- `enqueue(item)` adds a new item to the rear of the queue. It needs the item and returns nothing.
- `dequeue()` removes the front item from the queue. It needs no parameters and returns the item. The queue is modified.
- `isEmpty()` tests to see whether the queue is empty. It needs no parameters and returns a boolean value.
- `size()` returns the number of items in the queue. It needs no parameters and returns an integer.

As an example, if we assume that `q` is a queue that has been created and is currently empty, then Table 1 shows the results of a sequence of queue operations. The queue contents are shown such that the front is on the right. 4 was the first item enqueued so it is the first item returned by `dequeue`.

Table 1: Example Queue Operations

Queue Operation	Queue Contents	Return Value
<code>q.isEmpty()</code>	<code>[]</code>	<code>True</code>
<code>q.enqueue(4)</code>	<code>[4]</code>	
<code>q.enqueue('dog')</code>	<code>['dog', 4]</code>	
<code>q.enqueue(True)</code>	<code>[True, 'dog', 4]</code>	
<code>q.size()</code>	<code>[True, 'dog', 4]</code>	<code>3</code>
<code>q.isEmpty()</code>	<code>[True, 'dog', 4]</code>	<code>False</code>
<code>q.enqueue(8.4)</code>	<code>[8.4, True, 'dog', 4]</code>	
<code>q.dequeue()</code>	<code>[8.4, True, 'dog']</code>	<code>4</code>
<code>q.dequeue()</code>	<code>[8.4, True]</code>	<code>'dog'</code>
<code>q.size()</code>	<code>[8.4, True]</code>	<code>2</code>

You have attempted 1 of 1 activities on this page

(WhatIsaQueue.html)

user not logged in

(ImplementingaQ...