

## Circular Queue Functions

Function	Code Snippet	Code Output
push: adds an element to the rear end of the queue	<pre> queue&lt;int&gt; myQueue(4); myQueue.push(29); myQueue.push(30); myQueue.push(46); myQueue.push(2);  myQueue.push(3); myQueue.push(4);  myQueue.print(); </pre>	<pre> Queue is full Queue is full Circular Queue contents: 1: 29 2: 30 3: 46 4: 2 </pre>
Pop: removes an element from the front of the queue	<pre> Before pop(): Circular Queue contents: 1: 29 2: 30 3: 46 4: 2 After pop(): Circular Queue contents: 1: 30 2: 46 3: 2 </pre>	<pre> queue&lt;int&gt; myQueue(4);  myQueue.push(29); myQueue.push(30); myQueue.push(46); myQueue.push(2);  cout &lt;&lt; "Before pop():" &lt;&lt; endl; myQueue.print();  myQueue.pop();  cout &lt;&lt; "After pop():" &lt;&lt; endl; myQueue.print(); </pre>
Front: Returns the item that's the 1 <sup>st</sup> element of the queue	<pre> queue&lt;int&gt; myQueue(4);  myQueue.push(29); myQueue.push(30); myQueue.pop(); myQueue.push(2); myQueue.push(11);  myQueue.print();  cout &lt;&lt; "The front of the "; cout &lt;&lt; " queue is " &lt;&lt; myQueue.front() &lt;&lt; endl; </pre>	<pre> Circular Queue contents: 1: 30 2: 2 3: 11 The front of the queue is 30 </pre>
size: Returns an integer that represents the number of items that are currently in the queue	<pre> queue&lt;int&gt; myQueue(4);  myQueue.push(29); myQueue.push(30); myQueue.push(2); myQueue.push(11);  myQueue.print();  cout &lt;&lt; "The size of the "; cout &lt;&lt; " queue is " &lt;&lt; myQueue.size() &lt;&lt; endl; </pre>	<pre> Circular Queue contents: 1: 29 2: 30 3: 2 4: 11 The size of the queue is 4 </pre>
empty: Returns a boolean that determines if the queue is empty or not - True, if the queue has 0 items	<pre> queue&lt;int&gt; myQueue(4);  if (myQueue.empty() == true) {     cout &lt;&lt; "The Queue is empty" &lt;&lt; endl; }  myQueue.push(29); myQueue.push(30); myQueue.push(46); myQueue.push(2);  if (myQueue.empty() == false) {     cout &lt;&lt; "The Queue is not empty" &lt;&lt; endl; }  myQueue.print(); </pre>	<pre> The Queue is empty The Queue is not empty Circular Queue contents: 1: 29 2: 30 3: 46 4: 2 </pre>

- False, if the queue has at least 1 item		
sort: Performs insertion sort on the queue and sorts the queue in ascending order	<pre> queue&lt;int&gt; myQueue(4);  myQueue.push(29); myQueue.push(30); myQueue.pop(); myQueue.push(2); myQueue.push(11); myQueue.push(62);  cout &lt;&lt; "Queue BEFORE sort()" &lt;&lt; endl; myQueue.print();  cout &lt;&lt; "Queue AFTER sort()" &lt;&lt; endl; myQueue.sort(); myQueue.print(); </pre>	<pre> Queue BEFORE sort() Circular Queue contents: 1: 30 2: 2 3: 11 4: 62 Queue AFTER sort() Circular Queue contents: 1: 2 2: 11 3: 30 4: 62 </pre>
move_to_rear: Moves the index of the 1st element of the queue all the way to the rear end of the queue	<pre> queue&lt;int&gt; myQueue(4);  myQueue.push(29); myQueue.pop(); myQueue.push(2); myQueue.push(11); myQueue.push(62);  cout &lt;&lt; "Queue BEFORE move_to_rear()" &lt;&lt; endl; myQueue.print();  cout &lt;&lt; "Queue AFTER move_to_rear()" &lt;&lt; endl; myQueue.move_to_rear(); myQueue.print(); </pre>	<pre> Queue BEFORE move_to_rear() Circular Queue contents: 1: 2 2: 11 3: 62 Queue AFTER move_to_rear() Circular Queue contents: 1: 11 2: 62 3: 2 </pre>
print: Display all items of the queue in ascending order, from front of the queue to the back of the queue.	<pre> queue&lt;int&gt; myQueue(10); myQueue.push(29); myQueue.push(30); myQueue.push(46); myQueue.push(24); myQueue.push(7); myQueue.push(6); myQueue.push(123); myQueue.push(81); myQueue.push(77); myQueue.push(388);  myQueue.print(); </pre>	<pre> Circular Queue contents: 1: 29 2: 30 3: 46 4: 24 5: 7 6: 6 7: 123 8: 81 9: 77 10: 388 </pre>

linear\_search(): Finds an item in a vector and returns the index of the last occurrence of the item

```
vector<int> myVector = { 3, 9, 6, 3, 22 , 1 , 3};

int index = linear_search(myVector, 3);

cout << endl;
if (index == -1) {
    cout << "Item not found" << endl;
}
else {
    cout << "Found " << myVector[index] << " at index " << index << endl;
}
```

```
Found 3 at index 6
```