



## Introduction to Basic Data Structures

### List Built-in Functions:

#### 1. Constructor

Name	Details	Time Complexity
<code>list&lt;type&gt;myList;</code>	Construct a list with 0 elements.	O(1)
<code>list&lt;type&gt;myList(N);</code>	Construct a list with N elements and the value will be garbage.	O(N)
<code>list&lt;type&gt;myList(N,V);</code>	Construct a list with N elements and the value will be V.	O(N)
<code>list&lt;type&gt;myList(list2);</code>	Construct a list by copying another list list2.	O(N)
<code>list&lt;type&gt;myList(A,A+N);</code>	Construct a list by copying all elements from an array A of size N.	O(N)
<code>list&lt;type&gt;myList(v.begin(),v.end());</code>	Construct a list by copying all elements from a vector v.	O(N)

#### 2. Capacity

Name	Details	Time Complexity
<code>myList.size()</code>	Returns the size of the list.	O(1)
<code>myList.max_size()</code>	Returns the maximum size that the list can hold.	O(1)

<b>myList.clear()</b>	Clears the list elements.	$O(N)$
<b>myList.empty()</b>	Return true/false if the list is empty or not.	$O(1)$
<b>myList.resize()</b>	Change the size of the list.	$O(K)$ ; where K is the difference between new size and current size.

### 3. Modifiers

Name	Details	Time Complexity
<b>myList=</b> or <b>myList.assign(list2.begin(),list2.end())</b>	Assign another list.	$O(N)$
<b>myList.push_back()</b>	Add an element to the tail.	$O(1)$
<b>myList.push_front()</b>	Add an element to the head.	$O(1)$
<b>myList.pop_back()</b>	Delete the tail.	$O(1)$
<b>myList.pop_front()</b>	Delete the head.	$O(1)$
<b>myList.insert()</b>	Insert elements at a specific position.	$O(N+K)$ ; where K is the number of elements to be inserted.
<b>myList.erase()</b>	Delete elements from a specific position.	$O(N+K)$ ; where K is the number of elements to be deleted.
<b>replace(myList.begin(),myList.end(),value,replace_value)</b>	Replace all the value with replace_value. Not under a list STL.	$O(N)$
<b>find(myList.begin()</b>	Find the value V. Not under	$O(N)$

<code>),myList.end(),V)</code>	a list STL.	
--------------------------------	-------------	--

#### 4. Operations

Name	Details	Time Complexity
<code>myList.remove(V)</code>	Remove the value V from the list.	$O(N)$
<code>myList.sort()</code>	Sort the list in ascending order.	$O(N \log N)$
<code>myList.sort(greater&lt;type&gt;())</code>	Sort the list in descending order	$O(N \log N)$
<code>myList.unique()</code>	Deletes the duplicate values from the list. You must sort the list first.	$O(N)$ , with sort $O(N \log N)$
<code>myList.reverse()</code>	Reverse the list.	$O(N)$

#### 5. Element access

Name	Details	Time Complexity
<code>myList.back()</code>	Access the tail element.	$O(1)$
<code>myList.front()</code>	Access the head element.	$O(1)$
<code>next(myList.begin(),i)</code>	Access the ith element	$O(N)$

## 6. Iterators

Name	Details	Time Complexity
<code>myList.begin()</code>	Pointer to the first element.	O(1)
<code>myList.end()</code>	Pointer to the last element.	O(1)