

C++ list object

Reference: <https://www.cplusplus.com/reference/list/list/>

Lists are:

- Dynamically growing or shrinking
- Chained List of values
- Bi-directional sequential scan
- Efficient for insertion and removal operations.

Advantage over Vectors:

- Efficient for insertion and removal operations.

Disadvantages with respect to vectors.

- Cannot have random access.

Constructors
<code>list();</code> Creates empty list object with no elements. Example: <code>list<double> list1;</code>
<code>list(size_type n, const value_type& val = value_type());</code> Creates a list of specified size and initialized second parameter. Note that second parameter is optional. Examples: <code>list<double> list2(10);</code> //size 10, initialized with 0 <code>list<double> list3(5, 100);</code> //size 5, initialized with 100
<code>list(const list& x);</code> Copy Constructor. Creates new list, copy of parameter list Examples: <code>list<double> list4(list1);</code> <code>list<double> list5 = list2;</code>
Accessor Methods
<code>size_type size();</code> Returns size of list. Examples: <code>cout << list1.size() << endl;</code>
<code>bool empty();</code> Returns whether the list is empty (i.e., whether its length is 0). Example: <code>if (!list1.empty()) { //do something }</code>
Mutator Methods
<code>void push_back(const value_type& val);</code> Adds a new element at the end of the list. The content of val is copied to the new element. Example: <code>list1.push_back(11.83);</code>

void <code>pop_back()</code> ; Removes the last element from the list. Example: <code>list1.pop_back()</code> ;
void <code>clear()</code> ; Removes all elements from the list object; list size becomes 0.
operator= Assignment. RHS list is assigned to left side list. Size, data everything changes as per the assigned one. <code>list2 = list1;</code>
Methods returning references to internal fields (and hence data)
reference <code>front()</code> ; Examples: <code>cout << "First element: " << list1.front() << endl;</code> Note front() returns reference here, therefore following will do what you can guess? <code>list1.front() = 11.11;</code>
reference <code>back()</code> Examples: <code>cout << "Last element: " << list1.back() << endl;</code> Note front() returns reference here, therefore following will do what you can guess? <code>list1.back() = 9.9;</code>
Iteration
iterator <code>begin()</code> ; Returns an iterator pointing to the first element in the list. iterator <code>end()</code> ; Returns an iterator referring to the past-the-end element in the list. Example: <pre>for (list<double>::iterator it = list1.begin(); it != list1.end(); ++it) cout << ' ' << *it;</pre>
iterator <code>rbegin()</code> ; //reverse beginning iterator <code>rend()</code> ; //reverse end
iterator <code>erase(iterator position)</code> ; iterator <code>erase(iterator first, iterator last)</code> ; Removes from the list either a single element (position) or a range of elements ([first, last)). This reduces the container size by the number of elements removed.
iterator <code>insert(iterator position, const value_type& val)</code> ; The list is extended by inserting new elements before the element at the specified position. Returns an iterator that points to the newly inserted element. Examples: <code>list1.insert(list1.begin()+2, 56.99);</code>
Relational Operators
<code>==, !=, <, <=, >, >=</code> LHS and RHS are list objects. For equality , comparison goes as following: if size is same, actual

elements are compared sequentially; stops when mismatch is found.
For others comparison is done sequentially and stops wherever it is
decisive.