

Lists

C++ supports the linked list data structure and provides additional methods to improve the list's functionality.

WE'LL COVER THE FOLLOWING ^

- Special methods of `std::list`



`std::list` is a doubled linked list. `std::list` needs the header `<list>`.

Although it has a similar interface to `std::vector` or `std::deque`, `std::list` is quite different to both of them. That's due to its structure.

`std::list` makes the following points unique:

- It supports no random access.
- The access of an arbitrary element is slow because you have to iterate in the worst case through the whole list.
- To add or remove an element is fast, if the iterator points to the right place.
- If you add or remove an element, the iterator adjusts accordingly and stays valid (does not go out of bounds).

Because of its special structure, `std::list` has a few special methods.

Special methods of `std::list`

Method	Description
<code>lis.merge(c)</code>	Merges the sorted list <code>c</code> into the sorted list <code>lis</code> so that <code>lis</code> keeps

<code>lis.merge(c)</code>	sorted list <code>lis</code> , so that <code>lis</code> keeps sorted.
<code>lis.merge(c, op)</code>	Merges the sorted list <code>c</code> into the sorted list <code>lis</code> , so that <code>lis</code> keeps sorted. Uses <code>op</code> as sorting criteria.
<code>lis.remove(val)</code>	Removes all elements from <code>lis</code> with value <code>val</code> .
<code>lis.remove_if(pre)</code>	Removes all elements from <code>lis</code> , fulfilling the predicate <code>pre</code> .
<code>lis.splice(pos, ...)</code>	Splits the elements in <code>lis</code> before <code>pos</code> . The elements can be single elements, ranges or lists.
<code>lis.unique()</code>	Removes adjacent element with the same value.
<code>lis.unique(pre)</code>	Removes adjacent elements, fulfilling the predicate <code>pre</code> .

Here are a few of the methods in a code snippet.

```
#include <iostream>
#include <list>
#include<algorithm>

int main(){
    std::list<int> list1{15, 2, 18, 19, 4, 15, 1, 3, 18, 5,
                        4, 7, 17, 9, 16, 8, 6, 6, 17, 1, 2};

    list1.sort();
    for (auto l: list1) std::cout << l << " ";
    // 1 1 2 2 3 4 4 5 6 6 7 8 9 15 15 16 17 17 18 18 19
    std::cout<<std::endl;

    list1.unique();
    for (auto l: list1) std::cout << l << " ";
    // 1 2 3 4 5 6 7 8 9 15 16 17 18 19
    std::cout<<std::endl;

    std::list<int> list2{10, 11, 12, 13, 14};
```



```
list1.splice(std::find(list1.begin(), list1.end(), 15), list2);  
for (auto l: list1) std::cout << l << " ";  
  
    // 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19  
  
return 0;  
}
```

