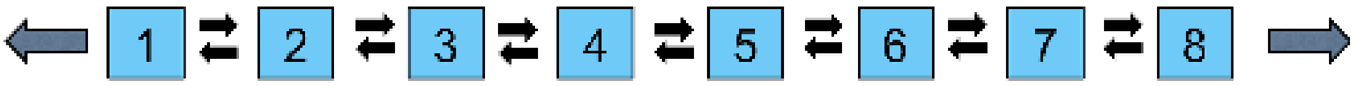
Lists

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 from both of them. That's due to its structure.

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

- It supports no random access.
- Accessing an arbitrary element is slow because we might have to iterate through the whole list.
- Adding or removing an element is fast, if the iterator points to the right place.
- If we add or remove an element, the iterator remains valid.

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

Special methods of `std::list`

Method	Description
`lis.merge(c)`	Merges the sorted list c into the sorted list `lis`, so that `lis` remains sorted.

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

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;
}
```

In the next lesson, we’ll study another type of list: forward lists.