Special Search Functions

The functions listed in this lesson make searching more efficient.

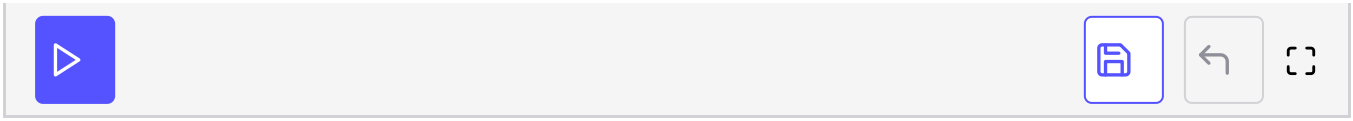
Ordered associative containers are optimized for searching. So they offer unique search functions.

Seach function	Description
`ordAssCont.count(key)`	Returns the number of values with the `key`.
`ordAssCont.find(key)`	Returns the iterator of `key` in `ordAssCont`. If there is no `key` in `ordAssCont` it returns `ordAssCont.end()`.
`ordAssCont.lower_bound(key)`	Returns the iterator to the first `key` in `ordAssCont` in which `key` would be inserted.
`ordAssCont.upper_bound(key)`	Returns the last position of `key` in `ordAssCont` in which `key` would be inserted.
`ordAssCont.equal_range(key)`	Returns the range ordAssCont.lower bound(key) and ordAssCont.upper_bound(key) in a `std::pair`.

Special search functions of the ordered associative containers

Now, the application of the special search functions.

```
// associativeContainerSearch.cpp
#include <iostream>
#include <set>

int main(){
  std::multiset<int> mySet{3, 1, 5, 3, 4, 5, 1, 4, 4, 3, 2, 2, 7, 6, 4, 3, 6};

  for (auto s: mySet) std::cout << s << " "; // 1 1 2 2 3 3 3 3 4 4 4 4 5 5 6 6 7
  std::cout<<"\n";

  mySet.erase(mySet.lower_bound(4), mySet.upper_bound(4));
  for (auto s: mySet) std::cout << s << " "; // 1 1 2 2 3 3 3 3 5 5 6 6 7
  std::cout<<"\n";

  std::cout << mySet.count(3) << std::endl; // 4
  std::cout << *mySet.find(3) << std::endl; // 3
  std::cout << *mySet.lower_bound(3) << std::endl; // 3
  std::cout << *mySet.upper_bound(3) << std::endl; // 5
  auto pair= mySet.equal_range(3);
  std::cout << "(" << *pair.first << "," << *pair.second << ")"; // (3,5)

  return 0;
}
```



Search in an associative container