The Comparison Criterion

This lesson talks about the rules followed by ordered associative containers when comparing the values inside them.

The default comparison criterion of the ordered associative containers is `std::less`. If you want to use a *user-defined* type as the key, you have to overload the operator `<`. It's sufficient to overload the operator `<` for your data type because the C++ runtime compares, with the help of the relation `(!(elem1<elem2 || elem2<elem1))`, two elements for equality.

You can specify the sorting criterion as a template argument. This sorting criterion must implement a *strict weak ordering*.

> **i Strict weak ordering**
> Strict weak ordering for a sorting criterion on a set `S` is given if the following requirements are met.
>
> - For `s` from `S` has to hold, that `s < s` is not possible.
> - For all `s1` and `s2` from `S` must hold: If `s1 < s2`, then `s2 < s1` is not possible.
> - For all `s1`, `s2` and `s3` with `s1 < s2` and `s2 < s3` must hold: `s1 < s3`.
> - For all `s1`, `s2` and `s3` with `s1` not comparable with `s2` and `s2` not comparable with `s3` must hold: `s1` is not comparable with `s3`.

In opposite to the definition of the *strict weak ordering*, the usage of a comparison criterion with *strict weak ordering* is a lot simpler for a `std::map`.

```
#include <iostream>
#include <map>

int main(){
  std::map<int, std::string, std::greater<int>> int2Str{
          {5, "five"}, {1, "one"}, {4, "four"}, {3, "three"},
          {2, "two"}, {7, "seven"}, {6, "six"} };
  for (auto p: int2Str) std::cout << "{" << p.first << "," << p.second << "} ";
          // {7,seven} {6,six} {5,five} {4,four} {3,three} {2,two} {1,one}


  return 0;
}
```