Permutations

We can see the different permutations in a range using C++.

`std::prev_permutation` and `std::next_permutation` return the previous smaller or next bigger permutation of the newly ordered range. If a smaller or bigger permutation is not available, the algorithms return `false`. Both algorithms need bidirectional iterators. Per default, the predefined sorting criterion `std::less` is used. If we use our sorting criterion, it has to obey the strict weak ordering. If not, the program will be undefined.

`std::prev_permutation`: Applies the previous permutation to the range.

```
bool prev_permutation(BiIt first, BiIt last)
bool prev_permutation(BiIt first, BiIt last, BiPred pre))
```

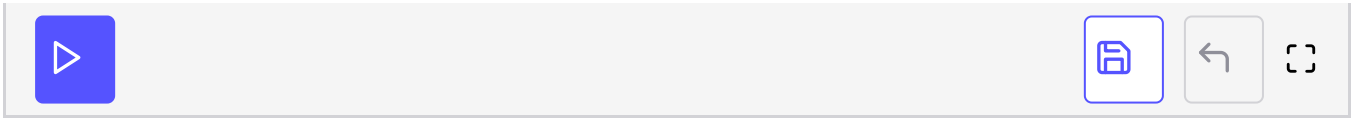
`std::next_permutation`: Applies the next permutation to the range.

```
bool next_permutation(BiIt first, BiIt last)
bool next_permutation(BiIt first, BiIt last, BiPred pre)
```

We can easily generate all the permutations of the range using both algorithms.

```
#include <algorithm>
#include <iostream>
#include <vector>

int main(){

  std::cout << std::endl;

  std::vector<int> myInts{1, 2, 3};

  std::cout << "All 3! permutions" << "\n\n";
  std::cout << "forwards" << std::endl;
  do{
    for (auto i: myInts) std::cout << i << " ";
    std::cout << std::endl;
  } while(std::next_permutation(myInts.begin(), myInts.end()));
```

```
  std::cout << std::endl;

  std::reverse(myInts.begin(), myInts.end());

  std::cout << "backwards" << std::endl;

  do{
    for (auto i: myInts) std::cout << i << " ";
    std::cout << std::endl;
  } while(std::prev_permutation(myInts.begin(), myInts.end()));

  std::cout << std::endl;

}
```



Permutation algorithms

In the next lesson, we'll discuss the numeric library which hosts many numeric functions.