Reverse Ranges

There were always roundabout ways of reversing a range. Now we have a predefined function to do just that.

`std::reverse` and `std::reverse_copy` invert the order of elements in their ranges.

`std::reverse`: Reverses the order of elements in the range.

```
void reverse(BiIt first, BiIt last)
void reverse(ExePol pol, BiIt first, BiIt last)
```

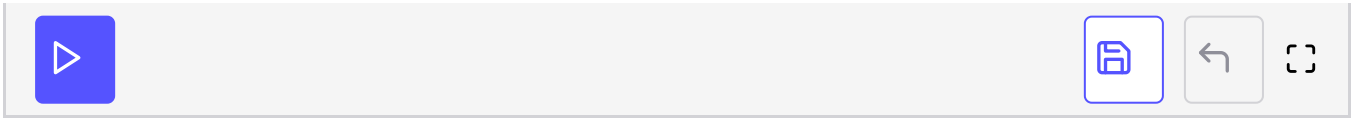
`std::reverse_copy`: Reverses the order of elements in the range and copies the result to `result`.

```
OutIt reverse_copy(BiIt first, BiIt last, OutIt result)
FwdIt reverse_copy(ExePol pol, BiIt first, BiIt last, FwdIt result)
```

Both algorithms require bidirectional iterators. The returned iterator points to the position of the output range `result` before the elements were copied.

```
#include <algorithm>
#include <deque>
#include <iostream>
#include <list>
#include <string>
#include <vector>

template <typename Cont, typename T>
void doTheSame(Cont cont,  T t){

  for ( auto c: cont ) std::cout << c << " ";
  std::cout << std::endl;
  std::cout << "cont.size(): " << cont.size() << std::endl;
  std::reverse(cont.begin(), cont.end());
  for ( auto c: cont ) std::cout << c << " ";
  std::cout << std::endl;
  std::reverse(cont.begin(), cont.end());
  for ( auto c: cont ) std::cout << c << " ";
  std::cout <<  std::endl;
  auto It= std::find(cont.begin(), cont.end(), t);
  std::reverse(It, cont.end());
```

```
  for ( auto c: cont ) std::cout << c << " ";

}

int main(){

  std::cout << std::endl;

  std::vector<int> myVec{1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
  std::deque<std::string> myDeque({"A", "B", "C", "D", "E", "F", "G", "H", "I"});
  std::list<char> myList({'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'});

  doTheSame(myVec, 5);
  std::cout << "\n\n";
  doTheSame(myDeque, "D");
  std::cout << "\n\n";
  doTheSame(myList, 'd');

  std::cout << "\n\n";
}
```



Reverse range algorithms

In the next lesson, we'll learn how we can rotate ranges.