Rotate Ranges

We can rotate our data such that every element now lies at a different index, which is decided by the rotation offset.

`std::rotate` and `std::rotate_copy` rotate their elements.

`std::rotate`: Rotates the elements in such a way that `middle` becomes the new first element.

```
FwdIt rotate(FwdIt first, FwdIt middle, FwdIt last)
FwdIt rotate(ExePol pol, FwdIt first, FwdIt middle, FwdIt last)
```

`std::rotate_copy`: Rotates the elements in such a way that `middle` becomes the new first element. Copies the result to `result`.

```
OutIt rotate_copy(FwdIt first, FwdIt middle, FwdIt last, OutIt result)
FwdIt2 rotate_copy(ExePol pol, FwdIt first, FwdIt middle, FwdIt last, FwdIt2 result)
```

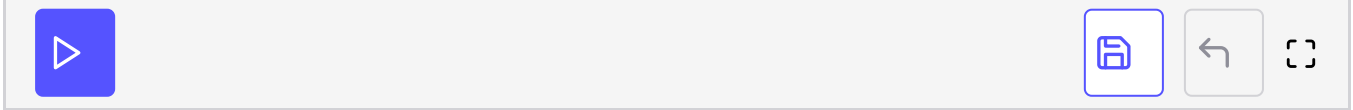
Both algorithms need forward iterators. The returned iterator is an end iterator for the copied range.

```
#include <algorithm>
#include <iostream>
#include <string>

int main(){

  std::string str{"123456789"};

  auto endIt= str.end();
  for (auto middleIt= str.begin();  middleIt != endIt; ++middleIt){
    std::rotate(str.begin(), middleIt, str.end());
    std::cout << str << std::endl;
  }
}
```



Rotate algorithms

In the next lesson, we'll discuss how we can rearrange the values in our range randomly.