Remove Elements and Ranges

Apart from insertion, copying and replacement, we can also delete elements completely.

The four variations `std::remove, std::remove_if, std::remove_copy` and `std::remove_copy_if` support two kinds of operations. On one hand, remove elements with and without a predicate from a range. On the other hand, copy the result of your modification to a new range.

`remove`: Removes the elements from the range, having the value `val`:

```
FwdIt remove(FwdIt first, FwdIt last, const T& val)
FwdIt remove(ExePol pol, FwdIt first, FwdIt last, const T& val)
```

`remove_if`: Removes the elements from the range, fulfilling the predicate `pred`:

```
FwdIt remove_if(FwdIt first, FwdIt last, UnPred pred)
FwdIt remove_if(ExePol pol, FwdIt first, FwdIt last, UnPred pred)
```

`remove_copy`: Removes the elements from the range, having the value `val`. Copies the result to `result`:

```
OutIt remove_copy(InpIt first, InpIt last, OutIt result, const T& val)
FwdIt2 remove_copy(ExePol pol, FwdIt first, FwdIt last, FwdIt2 result, const T& val)
```

`remove_copy_if`: Removes the elements from the range, which fulfill the predicate `pred`. Copies the result to `result`.

```
OutIt remove_copy_if(InpIt first, InpIt last, OutIt result, UnPre pred)
FwdIt2 remove_copy_if(ExePol pol, FwdIt first, FwdIt last, FwdIt2 result, UnPre pred)
```

The algorithms need input iterators for the source range and an output iterator for the destination range. They return as a result an end iterator for the destination range.

⚠ **Apply the erase-remove idiom**
The remove variations don’t remove an element from the range. They return the new *logical* end of the range. You have to adjust the size of the container with the erase-remove idiom.

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

int main(){

  std::cout << std::endl;

  std::vector<int> myVec{0, 1, 2, 3, 4, 5, 6, 7, 8, 9};

  for (auto v: myVec) std::cout << v << " ";
  std::cout << std::endl;
  auto newIt= std::remove_if(myVec.begin(), myVec.end(), [](int a){ return a % 2; } );
  for (auto v: myVec) std::cout << v << " ";
  std::cout << std::endl;

  myVec.erase(newIt, myVec.end());
  for (auto v: myVec) std::cout << v << " ";

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

  std::string str{"Only for Testing Purpose."};
  std::cout << str << std::endl;
  str.erase(std::remove_if(str.begin(), str.end(), [](char c){ return std::isupper(c);} ), st
  std::cout << str << std::endl;

  std::cout << std::endl;

}
```