Remove Duplicates

Next, we'll study ways to make sure each element in our range is unique.

WE'LL COVER THE FOLLOWING

With the algorithms `std::unique` and `std::unique_copy`, We have opportunities to remove adjacent duplicates. This can be done with and without a binary predicate.

`std::unique`

Removes adjacent duplicates.

```
FwdIt unique(FwdIt first, FwdIt last)
FwdIt unique(ExePol pol, FwdIt first, FwdIt last)
```

Removes adjacent duplicates, satisfying the binary predicate.

```
FwdIt unique(FwdIt first, FwdIt last, BiPred pre)
FwdIt unique(ExePol pol, FwdIt first, FwdIt last, BiPred pre)
```

`std::unique_copy`

Removes adjacent duplicates and copies the result to `result`.

```
OutIt unique_copy(InpIt first, InpIt last, OutIt result)
FwdIt2 unique_copy(ExePol pol, FwdIt first, FwdIt last, FwdIt2 result)
```

Removes adjacent duplicates satisfying the binary predicate and copies the result to `result`.

```
OutIt unique_copy(InpIt first, InpIt last, OutIt result, BiPred pre)
FwdIt2 unique_copy(ExePol pol, FwdIt first, FwdIt last, FwdIt2 result, BiPred pre)
```

> ⚠ **The `unique` algorithms return the new logical end iterator**
> The `unique` algorithms return the logical end iterator of the range. The elements have to be removed with the erase-remove idiom.

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

int main(){

  std::cout << std::endl;

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

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

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

  std::vector<int> myVec2{1, 4, 3, 3, 3, 5, 7, 9, 2, 4, 1, 6, 8, 0, 3, 5, 7, 8, 7, 3, 9, 2, 4
  std::vector<int> resVec;
  resVec.reserve(myVec2.size());
  std::unique_copy(myVec2.begin(), myVec2.end(), std::back_inserter(resVec),
           [](int a, int b){return (a % 2) == (b % 2); });

  for(auto v: myVec2) std::cout << v << " ";
  std::cout << std::endl;
  for(auto v: resVec) std::cout << v << " ";

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

}
```

Remove duplicates algorithms

This concludes our discussion on modifying algorithms. In the next chapter, we'll study some more algorithms.