Assign and Swap

This lesson deals with ways to update and swap values in containers.

We can assign new elements to an existing container and, if required, swap two containers as well. If we want to assign a container `cont2` to another container `cont1`, we can do so either through copy assignment `cont1= cont2` or move assignment `cont1= std::move(cont2)`. In a `move` assignment, the value of `cont2` is copied to `cont1` and `cont2` becomes empty. A special form of assignment is the one with an initializer list: `cont= {1, 2, 3, 4, 5}`. In the case of `std:array`, an initializer list is not possible, hence we use aggregate initialization. The function `swap` exists in two forms. It is a method `cont1(swap(cont2))` and also a function template `std::swap(cont1, cont2)`.

```
// containerAssignmentAndSwap.cpp
#include <iostream>
#include <set>

int main(){
  std::set<int> set1{0, 1, 2, 3, 4, 5};
  std::set<int> set2{6, 7, 8, 9};

  for (auto s: set1) std::cout << s << " "; // 0 1 2 3 4 5
  std::cout << "\n";
  for (auto s: set2) std::cout << s << " "; // 6 7 8 9
  std::cout << "\n";

  set1= set2;
  for (auto s: set1) std::cout << s << " "; // 6 7 8 9
  std::cout << "\n";
  for (auto s: set2) std::cout << s << " "; // 6 7 8 9
  std::cout << "\n";

  set1= std::move(set2);//moves value of set2 in set1 and set2 becomes empty
  for (auto s: set1) std::cout << s << " "; // 6 7 8 9
  std::cout << "\n";
  for (auto s: set2) std::cout << s << " ";//prints null since set2 becomes empty after the m

  set2= {60, 70, 80, 90};
  for (auto s: set1) std::cout << s << " "; // 6 7 8 9
  std::cout << "\n";
  for (auto s: set2) std::cout << s << " "; // 60 70 80 90
  std::cout << "\n";

  std::swap(set1, set2);
  for (auto s: set1) std::cout << s << " "; // 60 70 80 90
  std::cout << "\n";
  for (auto s: set2) std::cout << s << " "; // 6 7 8 9

  std::cout << "\n";
  return 0;
}
```

Assignment and swap

In the next lesson, we'll discuss general comparison operators on containers.