Working With Maps

Let's examine the new behavior of maps in C++17.

For now, do not concern yourself with the purpose of the code below. We will only be examining the following functions and utilities:

```
• mapCopy from std::map
```

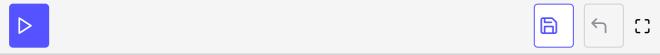
- insert_or_assign
- if statement
- Improved for loop functionality

```
#include <iostream>
#include <map>
int main() {
  std::map
std::map mapCopy{mapUsersAge { "Alex", 45 }, { "John", 25 } };

std::map mapCopy{mapUsersAge};

if (auto [iter, wasAdded] = mapCopy.insert_or_assign("John", 26); !wasAdded)
  std::cout << iter->first << " reassigned...\n";

for (const auto& [key, value] : mapCopy)
  std::cout << key << ", " << value << '\n';
}</pre>
```



The above example uses the following features:

• **Line 8**: Template Argument Deduction for Class Templates - mapCopy type is deduced from the type of mapUsersAge . No need to declare

```
std::map<std::string, int> mapCopy{...}.
```

- Line 10: New inserting method for maps insert_or_assign.
- Line 10: Structured Bindings captures a returned pair from

insert_or_assign into separate names.

- Line 10: init if statement iter and wasAdded are visible only in the scope of the surrounding if statement.
- Line 13: Structured Bindings inside a range-based for loop we can iterate using key and value rather than pair.first and pair.second.

In the next lesson, we'll look at another example which further explains the usability of C++17.