Useful Functions

C++ offers several tools to make the iteration process simpler and safer.

The global functions `std::begin`, `std::end`, `std::prev`, `std::next`, `std::distance`, and `std::advance` make handling of the iterators a lot easier. Only the function `std::prev` requires a bidirectional iterator. All functions need the header `<iterator>`. The table gives an overview:

Global function	Description
`std::begin(cont)`	Returns a begin iterator to the container `cont`.
`std::end(cont)`	Returns an end iterator to the container `cont`.
`std::rbegin(cont)`	Returns a reverse begin iterator to the container `cont`.
`std::rend(cont)`	Returns a reverse end iterator to the container `cont`.
`std::cbegin(cont)`	Returns a constant begin iterator to the container `cont`.
`std::cend(cont)`	Returns a constant end iterator to the container `cont`.
`std::crbegin(cont)`	Returns a reverse constant begin iterator to the container `cont`.

`std::crend(cont)`	Returns a reverse constant end iterator to the container `cont`.
`std::prev(it)`	Returns an iterator, which points to a position before `it`
`std::next(it)`	Returns an iterator, which points to a position after `it`.
`std::distance(fir, sec)`	Returns the number of elements between `fir` and `sec`.
`std::advance(it, n)`	Puts the iterator `it` n positions further.

Useful functions for iterators

Now, the application of useful functions.

```
#include <array>
#include <iostream>
#include <iterator>
#include <string>
#include <unordered_map>

int main(){

  std::cout << std::endl;

  std::unordered_map<std::string, int> myMap{ {"Rainer", 1966}, {"Beatrix", 1966}, {"Juliette

  for ( auto m: myMap) std::cout << "{" << m.first <<  ", " << m.second << "} ";

  std::cout << std::endl;

  auto mapItBegin= std::begin(myMap);
  std::cout << "{" << mapItBegin->first <<  ", " << mapItBegin->second << "}" << std::endl;
  auto mapIt= std::next(mapItBegin);
  std::cout << "{" << mapIt->first <<  ", " << mapIt->second << "}" << std::endl;

  auto dist= std::distance(mapItBegin, mapIt);
  std::cout << "std::distance(mapItBegin, mapIt): " << dist << std::endl;

  std::cout << std::endl;

  std::array<int, 10> myArr{0, 1, 2, 3, 4, 5, 6, 7, 8, 9};

  for ( auto a: myArr ) std::cout << a << " ";
```

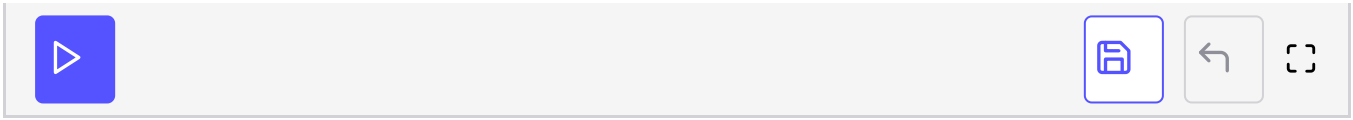
```
  std::cout << std::endl;

  auto arrItEnd= std::end(myArr);
  auto arrIt= std::prev(arrItEnd);
  std::cout << *arrIt << std::endl;

  std::advance(arrIt, -5);
  std::cout << *arrIt << std::endl;

  std::cout << std::endl;

}
```



Helper functions for iterators

In the next lesson, we'll discuss how iterators can be used for searching as well as inserting values.