

Useful Functions

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

The global functions `std::begin`, `std::end`, `std::prev`, `std::next`, `std::distance` and `std::advance` make your 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 you the overview.

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

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

Useful functions for iterators

Now, the application of the 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