## Access

In this lesson, we'll discuss the method used to access the elements of a container.

To access the elements of a container, we can use an iterator. A begin and end iterator forms a range, which can be processed further. For a container <code>cont</code>, <code>cont.begin()</code> is the begin iterator and <code>cont.end()</code> is the end iterator, which defines a half-open range. It is half-open because the begin iterator belongs to the range, the end iterator refers to a position past the range. With the iterator pair <code>cont.begin()</code> and <code>cont.end()</code> we can modify the elements.

Iterator	Description
<pre>cont.begin() and cont.end()</pre>	Pair of iterators to iterate forward.
<pre>cont.cbegin() and cont.cend()</pre>	Pair of iterators to iterate const forward.
<pre>cont.rbegin() and cont.rend()</pre>	Pair of iterators to iterate backward.
<pre>cont.crbegin() and cont.crend()</pre>	Pair of iterators to iterate const backward.

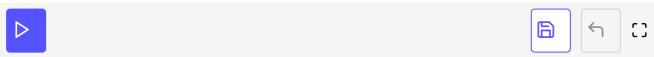
## Functions available for iterators in containers

Now I can modify the container.

```
// containerAccess.cpp
#include <iostream>
#include <vector>
using namespace std;

struct MyInt{
```

```
MyInt(int i): myInt(i){};
 int myInt;
};
int main(){
 std::vector<MyInt> myIntVec;
 myIntVec.push_back(MyInt(5));
 myIntVec.emplace_back(1);
 std::cout << myIntVec.size() << std::endl;  // 2</pre>
 std::vector<int> intVec;
 intVec.assign({1, 2, 3});
 for (auto v: intVec) std::cout << v << " "; // 1 2 3
 cout << std::endl;</pre>
 intVec.insert(intVec.begin(), 0);
 for (auto v: intVec) std::cout << v << " "; // 0 1 2 3
 cout << std::endl;</pre>
 intVec.insert(intVec.begin()+4, 4);
 for (auto v: intVec) std::cout << v << " "; // 0 1 2 3 4
 cout << std::endl;</pre>
 intVec.insert(intVec.end(), {5, 6, 7, 8, 9, 10, 11});
 for (auto v: intVec) std::cout << v << " "; // 0 1 2 3 4 5 6 7 8 9 10 11
 cout << std::endl;</pre>
 for (auto revIt= intVec.rbegin(); revIt != intVec.rend(); ++revIt)
    std::cout << *revIt << " ";
                                              // 11 10 9 8 7 6 5 4 3 2 1 0
 cout << std::endl;</pre>
 intVec.pop_back();
 cout << std::endl;</pre>
 return 0;
```



Access the elements of a container

In the next lesson, we'll discuss how to assign and swap values between containers.