

Access

To access the elements of a container, you can use an iterator. If you use a begin and end iterator, you have a range, which you can further process. For a container `cont`, you get with `cont.begin()` the begin iterator and with `cont.end()` 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 `cont.begin()` and `cont.end()` you can modify the elements.

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

Creation and deletion of a container

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

    std::vector<int> intVec;
    intVec.assign({1, 2, 3});
    for (auto v: intVec) std::cout << v << " ";         // 1 2 3
    cout << std::endl;

    intVec.insert(intVec.begin(), 0);
    for (auto v: intVec) std::cout << v << " ";         // 0 1 2 3
    cout << std::endl;

    intVec.insert(intVec.begin()+4, 4);
    for (auto v: intVec) std::cout << v << " ";         // 0 1 2 3 4
    cout << std::endl;

    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;

    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;

    intVec.pop_back();
    for (auto v: intVec ) std::cout << v << " ";        // 0 1 2 3 4 5 6 7 8 9 10
    cout << std::endl;

    return 0;
}

```



Access the elements of a container