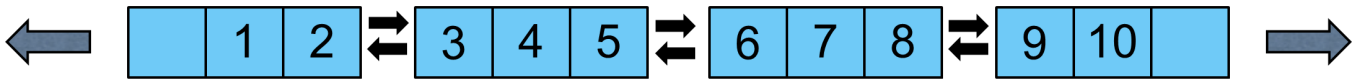


# Dequeues

Let's take a look at deques and their similarity to vectors.



`std::deque`(double ended **q**ueue), which consist of a sequence of arrays, is quite similar to `std::vector`. `std::deque` needs the header `<deque>`. The `std::deque` has three additional methods `deq.push_front(elem)`, `deq.pop_front()`, and `deq.emplace_front(args... )` to add or remove elements at the beginning.

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

class MyInt{
private:
    int myInt;
public:
    MyInt(int i): myInt(i){};
    friend ostream& operator << (ostream& os, const MyInt& m)
    {
        os << m.myInt <<" ";
        return os ;
    }
};

int main(){
    std::deque<MyInt> myIntDeq;

    myIntDeq.push_back(MyInt(5));
    myIntDeq.emplace_back(1);
    std::cout << myIntDeq.size() << std::endl;    // 2

    std::deque<MyInt> intDeq;

    intDeq.assign({1, 2, 3});
    for (auto v: intDeq) cout << v << " ";    // 1 2 3
    cout<<endl;

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

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

cout<<endl;

intDeq.insert(intDeq.end(), {5, 6, 7, 8, 9, 10, 11});
for (auto v: intDeq) cout << v << " ";    // 0 1 2 3 4 5 6 7 8 9 10 11
cout<<endl;

for (auto revIt= intDeq.rbegin(); revIt != intDeq.rend(); ++revIt)
    std::cout << *revIt << " ";           // 11 10 9 8 7 6 5 4 3 2 1 0
cout<<endl;

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

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

return 0;
}
```



---

In the next lesson, we'll discuss lists.