# Лабораторная работа №2

**initializer\_list,** **move, forward, default, delete, move итераторы**

## Задание 1. Создайте класс, который должен быть "оберткой" для вектора с УНИКАЛЬНЫМИ значениями любого типа.

* Для хранения элементов используйте std::vector
* Реализуйте конструктор, который может принимать любое количество значений (значения могут повторяться)
* Реализуйте метод добавления любого количества значений (значения могут повторяться)
* Реализуйте метод удаления любого количества значений (значения могут повторяться)
* Реализуйте метод сортировки, который будет принимать в качестве параметра признак по возрастанию / по убыванию
* и другие (полезные на Ваш взгляд) методы

## Задание 1b. Модифицируйте класс из предыдущего задания (или создайте новый) таким образом, чтобы значения не только были бы уникальными, но и находились в заданном диапазоне.

Замечание: такой класс в общем виде (для любых данных) смысла, наверно, не имеет, но его можно использовать для таких типов данных, как целых, плавающих и тех пользовательских, для которых имеют смысл операции сравнения (больше, меньше)

Подумайте: какие данные нужно добавить в класс и какие методы нужно модифицировать

## Задание 2. Реализуйте шаблонный класс, который является оберткой для очереди с элементами любого типа.

Очередь требуется реализовать посредством динамического массива, при этом использовать массив как циклический буфер. При реализации нужно обеспечить минимальную фрагментацию памяти =>

* предусмотреть наличие резервных элементов
* память без очевидной необходимости не перераспределять

Тестируем разработанный класс на приведенном ниже фрагменте. Следующий фрагмент должен работать не только корректно, но и эффективно:

MyQueue<MyString> q1{ MyString(“AAA”), MyString (“qwerty”),<другие\_инициализаторы>};

MyString s(“abc”);

q1.push(s);

q1.push(MyString(“123”));

MyString s1 = q1.pop();

q1.push(“qqq”);

MyQueue < MyString > q2 = q1;

MyQueue < MyString > q3{10, MyString (“!”)}; //очередь должна содержать 10 элементов с строкой «!»

q1 = q3;

q2 = MyQueue < MyString > (5, MyString (“?”));

q1 = { MyString(“bbb”), MyString (“ssss”)};

## Задание 3. Реализуйте шаблон класса MyUniquePTR, который является оберткой для указателя на объект любого типа.

Задача – класс должен обеспечивать единоличное владение динамически создаваемым объектом. Проверить функционирование шаблона на примере MyString:

{

MyUniquePTR<MyString> p1(new MyString (“abc”));

std:: cout<<p1->GetString();

p1->SetNewString(“qwerty”);

MyString s2 = \*p1;

//MyUniquePTR< MyString > p2=p1; //здесь компилятор должен выдавать ошибку =>

Исправьте!

If(p1) {std::cout<<”No object!”} //а это должно работать

MyUniquePTR< MyString > p3(new MyString (“vvv”));

//p3 = p2; //и здесь компилятор должен выдавать ошибку

vector< MyUniquePTR< MyString >> v; //как проинициализировать???

list< MyUniquePTR< MyString >> l;

//как скопировать из v в l ???

}