# Сдача проекта в системе irunner. Усеченная версия примера контрольной работы № 2. Очередь.

*Указания:*

1. Решение работы требуется сдать в систему в **iRunner** [**https://acm.bsu.by/**](https://acm.bsu.by/)в виде **zip** файла с именем вида **queue\_gr1\_v1\_ivanov\_ivan.zip** (\*красным цветом записаны персональные данные студента), содержащем файлы решения (\*.cpp, \*.h, \*\_tests.cpp). Для файлов с тестами используйте постфикс “\_tests”.

Имена файлов с реализацией:

queue\_linked\_list.h

1. Если Вы сдаете неполное решение, то всё равно необходимо создать функции и методы на все пункты (при этом они могут ничего не делать и возвращать просто любую константу). В противном случае, как и в случае использование неверных сигнатур, при компиляции на сервере произойдет ошибка.
2. При реализации алгоритмов не используйте стандартные контейнеры, если это не указано явно, а используйте собственную реализацию контейнеров.

**Задача 1. Реализация структуры данных Очередь на последовательности связных компонент**

Создать шаблонный класс очереди на последовательности связных компонент QueueLinkedList<T> и публичную вложенную структуру Node. Данный класс должен быть объявлен и определен в пространстве имен containers.

Структура Node должна содержать:

* Публичное поле value типа T со значением, хранимым в данном элементе списка.
* Приватные поля для осуществления связи элементов в списке.
* **Приватный** конструктор(ы). Ничто кроме QueueLinkedList не должно иметь возможности создания элементов типа Node.

Класс QueueLinkedList<T> должен иметь следующий публичный интерфейс:

* Конструктор по умолчанию.
* Конструктор от std::initializer\_list<T>.
* Конструктор копирования и копирующий оператор присваивания.
* Конструктор перемещения и перемещающий оператор присваивания.
* Методы int Size() и bool IsEmpty().
* Метод ToVector(), возвращающий представление в виде вектора (std::vector).
* Методы Node\* Head() и Node\* Tail(), позволяющие получить первый / последний элемент в очереди.
* Метод Push(constT& value), Push(T&& value) осуществляющие вставку элемента в конец очереди.
* Методы Pop(), осуществляющий удаление первого элемента из очереди, возвращая значение класса T или выбрасывая исключение std, если очередь пустая.
* Метод int Find(constT& value), который возвращает позицию (индекс) первого вхождения элемента value в список. Если элемент отсутствует в списке, то вернуть -1.
* Метод std::vector<int> FindAll(constT& value), который возвращает индексы всех вхождений элемента value в список в порядке возрастания.
* Оператор обращения по индексу ([]), возвращающий Node\*, который позволяет обратиться к элементу списка по его индексу.
* Операторы, обеспечивающие возможность проверки очереди на равенство / различие.

Замечания:

* Необходимо создать константные аналоги для методов Back, Front и оператора [], которые будут возвращать const Node\*.
* Необходимо проверять корректность вызовов и передаваемых параметров с помощью assertов.

**Задача 2. Тестирование очереди**

Разработать тесты для тестирования функционала.