**ФИТ НГУ, курс ООП**

**Задача 1. Кольцевой буфер**

**Общие сведения**

Кольцевой буфер - это массив, начало и конец которого замкнуты. В частности, это означает, что при добавлении элементов в конец буфера, при исчерпании его ёмкости, начнут переписываться элементы из начала буфера, а “начало”, соответственно, сдвигаться.

Детальное описание этой структуры данных содержится в английской википедии: <http://en.wikipedia.org/wiki/Circular_buffer>

Пример промышленной реализации кольцевого буфера содержится в библиотеке Boost:

<http://www.boost.org/doc/libs/1_51_0/libs/circular_buffer/doc/circular_buffer.html>

**Задача**

* Реализовать кольцевой буфер с заданным интерфейсом (см. раздел “Реализация”).
* Тщательно задокументировать публичные члены класса на языке, приближенном к техническому английскому.
* Написать юнит-тесты на все публичные методы класса с помощью любой специализированной библиотеки (рекомендуется Google Test Framework <http://code.google.com/p/googletest/>), либо без оной (на усмотрение преподавателя). Убедиться в полноте покрытия кода тестами (каждая строчка кода должна исполняться хотя бы одним тестом).

**Методические указания**

* При написании кода особое внимание обращайте на обработку исключительных ситуаций и граничных случаев, в частности, на корректность аргументов методов. Продумывайте и документируйте обработку ошибок в ваших методах.
* *Дополнительно:* попробуйте часть методов протестировать до их реализации.
* *Дополнительно:* изучите открытые реализации кольцевого буфера. Сравните с вашей реализацией.

**Реализация**

//В этой задаче для простоты не требуется делать контейнер шаблонным,

//но это вполне допускается по желанию студента.

typedef char value\_type;

class CircularBuffer {

value\_type \* buffer;

/\*... реализация ... \*/

public:

CircularBuffer();

~CircularBuffer();

CircularBuffer(const CircularBuffer & cb);

//Конструирует буфер заданной ёмкости.

explicit CircularBuffer(int capacity);

//Конструирует буфер заданной ёмкости, целиком заполняет его элементом elem.

CircularBuffer(int capacity, const value\_type & elem);

//Доступ по индексу. Не проверяют правильность индекса.

value\_type & operator[](int i);

const value\_type & operator[](int i) const;

//Доступ по индексу. Методы бросают исключение в случае неверного индекса.

value\_type & at(int i);

const value\_type & at(int i) const;

value\_type & front(); //Ссылка на первый элемент.

value\_type & back(); //Ссылка на последний элемент.

const value\_type & front() const;

const value\_type & back() const;

//Линеаризация - сдвинуть кольцевой буфер так, что его первый элемент

//переместится в начало аллоцированной памяти. Возвращает указатель

//на первый элемент.

value\_type \* linearize();

//Проверяет, является ли буфер линеаризованным.

bool is\_linearized() const;

//Сдвигает буфер так, что по нулевому индексу окажется элемент

//с индексом new\_begin.

void rotate(int new\_begin);

//Количество элементов, хранящихся в буфере.

int size() const;

bool empty() const;

//true, если size() == capacity().

bool full() const;

//Количество свободных ячеек в буфере.

int reserve() const;

//ёмкость буфера

int capacity() const;

void set\_capacity(int new\_capacity);

//Изменяет размер буфера.

//В случае расширения, новые элементы заполняются элементом item.

void resize(int new\_size, const value\_type & item = value\_type());

//Оператор присваивания.

CircularBuffer & operator=(const CircularBuffer & cb);

//Обменивает содержимое буфера с буфером cb.

void swap(CircularBuffer & cb);

//Добавляет элемент в конец буфера.

//Если текущий размер буфера равен его ёмкости, то переписывается

//первый элемент буфера (т.е., буфер закольцован).

void push\_back(const value\_type & item = value\_type());

//Добавляет новый элемент перед первым элементом буфера.

//Аналогично push\_back, может переписать последний элемент буфера.

void push\_front(const value\_type & item = value\_type());

//удаляет последний элемент буфера.

void pop\_back();

//удаляет первый элемент буфера.

void pop\_front();

//Вставляет элемент item по индексу pos. Ёмкость буфера остается неизменной.

void insert(int pos, const value\_type & item = value\_type());

//Удаляет элементы из буфера в интервале [first, last).

void erase(int first, int last);

//Очищает буфер.

void clear();

};

bool operator==(const CircularBuffer & a, const CircularBuffer & b);  
bool operator!=(const CircularBuffer & a, const CircularBuffer & b);