# ФИТ НГУ, курс ООП

# Задача 1а. Битовый массив

### Общие сведения

Битовый массив (битовое множество) - это массив, компактно хранящий биты и позволяющий удобно ими оперировать. В этой задаче требуется реализовать битовый массив переменного размера (в отличие от std::bitset).

Более подробно о битовых массивах: <http://en.wikipedia.org/wiki/Bit_array>.

Пример промышленной реализации битового массива: <http://www.boost.org/doc/libs/1_51_0/libs/dynamic_bitset/dynamic_bitset.html>

### Задача

1. Реализовать битовый массив с заданным интерфейсом (см. раздел “Реализация”).
2. Тщательно задокументировать публичные члены класса на языке, приближенном к техническому английскому.
3. 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.

class BitArray  
{  
public:  
 BitArray();

~BitArray();

//Конструирует массив, хранящий заданное количество бит.

//Первые sizeof(long) бит можно инициализровать с помощью параметра value.  
 explicit BitArray(int num\_bits, unsigned long value = 0);  
 BitArray(const BitArray& b);

//Обменивает значения двух битовых массивов.  
 void swap([BitArray](http://www.google.com/url?q=http%3A%2F%2Fwww.boost.org%2Fdoc%2Flibs%2F1_51_0%2Flibs%2Fdynamic_bitset%2Fdynamic_bitset.html%23cons5&sa=D&sntz=1&usg=AFQjCNEjZuU656LfYXdsy4nMLBu6f7LXMg)& b);  
  
 BitArray& operator=(const BitArray& b);

//Изменяет размер массива. В случае расширения, новые элементы

//инициализируются значением value.  
 void resize(int num\_bits, bool value = false);

//Очищает массив.  
 void clear();

//Добавляет новый бит в конец массива. В случае необходимости

//происходит перераспределение памяти.  
 void push\_back(bool bit);

//Битовые операции над массивами.

//Работают только на массивах одинакового размера.

//Обоснование реакции на параметр неверного размера входит в задачу.  
 BitArray& operator&=(const BitArray& b);  
 BitArray& operator|=(const BitArray& b);  
 BitArray& operator^=(const BitArray& b);

//Битовый сдвиг с заполнением нулями.  
 BitArray& operator<<=(int n);  
 BitArray& operator>>=(int n);  
 BitArray operator<<(int n) const;  
 BitArray operator>>(int n) const;

//Устанавливает бит с индексом n в значение val.  
 BitArray& set(int n, bool val = true);

//Заполняет массив истиной.  
 BitArray& set();

//Устанавливает бит с индексом n в значение false.  
 BitArray& reset(int n);

//Заполняет массив ложью.  
 BitArray& reset();

//true, если массив содержит истинный бит.  
 bool any() const;

//true, если все биты массива ложны.  
 bool none() const;

//Битовая инверсия  
 BitArray operator~() const;

//Подсчитывает количество единичных бит.  
 int count() const;

//Возвращает значение бита по индексу i.  
 bool operator[](int i) const;  
  
 int size() const;  
 bool empty() const;

//Возвращает строковое представление массива.

std::string to\_string() const;

};

bool operator==(const BitArray & a, const BitArray & b);  
bool operator!=(const BitArray & a, const BitArray & b);  
  
BitArray operator&(const BitArray& b1, const BitArray& b2);  
BitArray operator|(const BitArray& b1, const BitArray& b2);  
BitArray operator^(const BitArray& b1, const BitArray& b2);