Лаб 1.B
Priority Queue (Очередь с приоритетами)

Общие сведения

Очередь с приоритетами (priority queue) — это структура данных, которая хранит набор элементов и позволяет быстро извлекать элемент с наивысшим (или наименьшим) приоритетом.

Чаще всего реализуется на основе **бинарной кучи**:

- добавление элемента (`push`) работает за *O(log n)*,
- извлечение максимума/минимума (`pop`) — также *O(log n)*,
- получение текущего максимума/минимума (`top`) — за *O(1)*.

В C++ std::priority_queue устроена как адаптер поверх `std::vector` с использованием алгоритмов `make_heap`, `push_heap` и `pop_heap`.
Документация: cppreference: priority_queue

Задача

Реализовать собственный класс `PriorityQueue` для хранения целых чисел (`int`).
Реализовать режим работы как **max-heap** (по умолчанию) и как **min-heap** (с флагом в конструкторе).

Методы должны:

- корректно обрабатывать пустую очередь (`top()` и `pop()` должны бросать исключение),
- обеспечивать амортизированную логарифмическую сложность для вставки и удаления,
- позволять сравнивать две очереди на равенство и выводить содержимое в поток (для отладки).

- # Методические указания
- Продумайте реализацию на базе `std::vector<int>` и ручного поддержания кучи (функции `sift_up` и `sift_down`).

- Обратите внимание на обработку граничных случаев: вставка в пустую очередь, извлечение до пустоты, переполнение `int`.

- Сделать юнит-тесты на все публичные методы (Google Test). https://google.github.io/googletest/

```
class PriorityQueue {
public:
    using value_type = int;
    using size_type  = std::size_t;

    // --- Конструкторы / деструктор ---
    PriorityQueue();                          // max-heap по умолчанию
    explicit PriorityQueue(bool max_heap);        // true = max-heap, false = min-heap
    explicit PriorityQueue(const std::vector<value_type>& data, bool max_heap = true);
    ~PriorityQueue();

    PriorityQueue(const PriorityQueue& other);     // Конструктор копирования
    PriorityQueue(PriorityQueue&& other) noexcept; // Конструктор перемещения

    PriorityQueue& operator=(const PriorityQueue& other);   // Присваивание копированием
    PriorityQueue& operator=(PriorityQueue&& other) noexcept; // Присваивание
перемещением

    // --- Основные методы ---
    void push(value_type x);       // Добавить элемент
    void pop();                    // Удалить элемент с наивысшим приоритетом
    const value_type& top() const;  // Получить элемент с наивысшим приоритетом

    bool empty() const noexcept;    // Проверка на пустоту
    size_type size() const noexcept;// Количество элементов
    void clear() noexcept;          // Очистить очередь
    void reserve(size_type n);      // Зарезервировать память под кучу

    bool is_max_heap() const noexcept; // Проверить режим работы (мин или макс куча)

    // --- Операторы ---
    bool operator==(const PriorityQueue& rhs) const noexcept;
    bool operator!=(const PriorityQueue& rhs) const noexcept;

    friend std::ostream& operator<<(std::ostream& os, const PriorityQueue& pq);
```

```
};
```