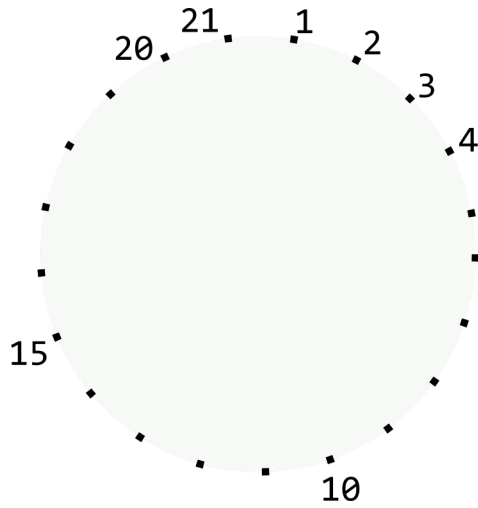


## Семинар #5: Контейнеры STL. Домашнее задание.

### Задача 1. Задача Иосифа Флавия



По кругу стоит  $n$  воинов, начиная с первого война они убивают каждого  $m$ -го. В каком порядке будут выбывать войны и в каком месте нужно встать, чтобы остаться последним выжившим?

Например, если  $n = 21$ , а  $m = 2$ , то войны будут выбывать в следующем порядке:

2 4 6 8 10 12 14 16 18 20 1 5 9 13 17 21 7 15 3 19

В конце останется воин с номером 11.

Решите эту задачу, промоделировав ситуацию с помощью контейнера `std::list`.

ВХОД	ВЫХОД
21 2	2 4 6 8 10 12 14 16 18 20 1 5 9 13 17 21 7 15 3 19 11
7 2	2 4 6 1 5 3 7
21 6	6 12 18 3 10 17 4 13 21 9 20 11 2 16 14 8 15 1 19 7 5

### Задача 2. Правильная скобочная последовательность

На вход приходит строка, содержащая некоторую скобочную последовательность. Она может состоять из трёх видов скобок: `(){}[]`. Вам нужно узнать, является ли эта скобочная последовательность правильной. Используйте `std::stack`.

ВХОД	ВЫХОД
<code>((()))</code>	Yes
<code>)()</code>	No
<code>[({})]</code>	Yes
<code>)}}</code>	No

ВХОД	ВЫХОД
<code>[]</code>	No
<code>([])</code>	No
<code>({}[]([]))[]</code>	Yes
<code>[]</code>	Yes

### Задача 3. Верёвка

На прямой лежит верёвка длиной  $n$  метров. Затем её начинают последовательно разрезать, всего сделав  $k$  разрезов. Все места разрезов – целые числа. Найти длину самого длинного куска после каждого разреза. Решение должно иметь вычислительную сложность  $O(n \log(n))$ . Используйте контейнеры `std::set` и `std::multiset`.

На вход программе подаются числа  $n$  и  $k$ , а затем  $k$  чисел – места разрезов.

ВХОД	ВЫХОД
20 8	12 10 8 7 6 5 5 4
8 10 15 1 7 4 11 18	

## Задача 4. Уникальные числа

На вход подаётся  $n$  чисел. Напечатайте эти числа удалив все дубликаты.

ВХОД	ВЫХОД
10 8 2 1 2 2 1 8 7 1 2	1 2 7 8

## Задача 5. Сортировка с помощью мультимножества

Считайте  $n$  чисел и отсортируйте их с помощью вставки в `multiset`. Распечатайте отсортированные числа.

## Задача 6. Количество повторений

На вход программе приходит  $n$  чисел. Некоторые числа могут повторяться. Вам нужно найти уникальные числа и количество их повторений. Например, если на вход приходят числа 5 1 5 1 1 1 2 1 5 1, то среди этих чисел есть 3 уникальных числа: число 1 повторяется 6 раз, число 2 – 1 раз, а число 5 – 3 раза. Алгоритм должен работать за  $O(n)$  или за  $O(n \log(n))$ . Используйте контейнер `std::map`.

ВХОД	ВЫХОД
10 5 1 5 1 1 1 2 1 5 1	1 2 5 6 1 3
10 2 2 2 2 2 1000000000 2 2 2 2	2 1000000000 9 1

## Задача 7. Поиск пути

В папке `wave_algo_tests` лежат изображения в формате `.ppm` (для просмотра изображений в формате `.ppm` советую использовать программу IrfanView). Каждая картинка содержит пиксели 4-х разных цветов:

1. Белые пиксели – места по которым можно ходить
2. Черные пиксели – препятствия, то есть места по которым ходить нельзя
3. Зелёный пиксель – начало пути
4. Красный пиксель – конец пути

Вам нужно найти кратчайший путь от начала до конца. При этом ходить можно только по пикселям: из одного пикселя можно перейти только в один из восьми соседей. Кратчайший путь нужно дорисовать на картинке синим цветом и сохранить картинку в новый файл. Для работы с изображением используйте класс `Image`, который находится в папке `image`. Кратчайший алгоритм можно найти с помощью волнового алгоритма. При реализации этого алгоритма используйте стандартные контейнеры

## Задача 8. Разбиение на пары

Напишите шаблонную функцию, которая будет принимать на вход контейнер (`vector`, `array`, `string`) и возвращать вектор пар нечётных и чётных элементов. Если в контейнере было нечётное количество элементов, то второй элемент последней пары должен быть инициализирован с помощью `value`-инициализации.

аргумент	возвращаемое значение
<code>std::vector{10, 20, 30, 40, 50}</code>	<code>{{10, 20}, {30, 40}, {50, 0}}</code>
<code>std::array&lt;std::string, 4&gt;{"cat", "dog", "mouse", "lion"}</code>	<code>{{"cat", "dog"}, {"mouse", "lion"}}</code>
<code>"Hello"</code>	<code>{{'H', 'e'}, {'l', 'l'}, {'o', '\0'}}</code>

Протестировать функцию можно в файле `code/test_pairing.cpp`.