**Теоретические вопросы**

1. Использует генераторы случайных чисел (их всего 16), каждый из которых оперирует последовательностью случайных значений (без знака), можно задавать изначальное состояние генератора через seed=time(0), также для выдачи псевдослучайного значения используются различные распределения (нормальное, бернулли и тд)
2. Функциональные объекты могут иметь состояние, так у Ф.О есть данные члены. У функций может быть состояние через static(это плохо), у лямбды есть захват внешних переменный. Ф.О. - это классы, каждый из которых имеет собственный тип, также можно создавать разные экземпляры. У функции есть только тип RT(Types...). Ф.О. работает быстрее функции, переданной через указатель.
3. Существует пять категорий итераторов:

Input\_Iterator – возможности =, !=, ==, ++, \*, -> для чтения (istream\_iterator); Output\_Iterator - возможности =, !=, ==, ++, \*, -> для записи (ostream\_iterator); Forward\_Iterator – возможности первых двух и многопроходность (forward\_list); Bidirectional\_Iterator – возможности forward\_iterator и -- (list, set, map); Random\_Access\_Iterator – возможности bidirectional\_iterator и < ,>, <=,>=, +/- n, i1-i2, [n] (vector)

1. Алгоритмы стандартной библиотеки делятся на:

Немодифицирущие алгоритмы (примеры: , count/\_if, minmax\_element, find, for\_each …)

Модифицирующие алгоритмы (for\_each, copy/\_if, transform, generate/\_n …) Алгоритмы удаления (remove\_if, unique, …)

Алгоритмы перестановки (reverse, shuffle, …)

Алгоритмы сортировки (sort, nth\_element, …)

Алгоритмы для упорядоченных диапазонов (lower\_bound, upper\_bound, …) Численные алгоритмы (accumulate, …)

1. Алгоритмы стандартной библиотеки упрощают и уменьшают количество кода, делают его аккуратнее и понятнее, также используя готовые алгоритмы, можно избежать ошибок. Алгоритмы написаны эффективным образом (скорее всего лучше, чем большинство пользовательских).