# **КОНТРОЛЬНЫЕ ВОПРОСЫ.**

## В каких ситуациях применяются типы std::pair и std::tuple?

Тип std::pair применяется при работе с парами «ключ — значение», в частности, в контейнерах std::map и ему подобных, в работе с функциями, возвращающими два параметра. Тип std::tuple применяется при работе с несколькими объектами (или примитивами) одновременно, в частности при работе с функциями, возвращающими более двух параметров или для краткой записи проверки на равенство нескольких значений разных типов.

## Каким требованиям должна удовлетворять качественная хэш‑функция?

1. Детерминированность: для одинаковых объектов функция возвращает одинаковый хэш.
2. Скорость работы: хэш-функция должна обладать *достаточно* высокой скоростью работы, при этом не зависящей от количества слотов h.
3. Равномерность: при работе хэш‑функции должно возникать *достаточно* малое число коллизий.

## Почему в хэш-таблицах возникают коллизии и как можно их разрешать?

Хэш-функция не гарантирует, что для разных объектов хэш будет разным. Более того, в силу конечного количества возможных значений типа int (и\или значений иного дискретного типа), возвращаемых хэш-функцией, можно показать, что при *достаточно* большом количестве различных объектов вероятность того, что хотя бы одна коллизия произойдёт, станет равна 1.

Методы разрешения коллизий:

1. Метод цепочек: в слоте находится не одно значение, а forward list или double linked list из них. В случае возникновения коллизии новое значение добавляется в соответствующий список.
2. Метод открытой адресации: если слот (имеющий одно значение) уже занят, рассматривается слот, следующий за ним, далее аналогично.

В обоих случаях может возрасти сложность основных операций.

## Почеу сложность основных операций хэш-таблиц в худшем случае О(N)?

Если хэш-функция такова, что для N объектов вычисляет одинаковый хэш (кроме того, в силу п.3 имеется ненулевая вероятность того, что при *достаточно* малом числе N для почти любой хэш-функции это возможно) и используется один из указанных в п.3 методов для разрешения коллизий, сложность основных операций может стать равной N. (Пример худшего случая: хеш-функция возвращает тождественную единицу, используется forward\_list, операция – поиск элемента)

## Что позволяет сделать инструмент для создания контейнеров Boost.Multiindex?

Boost.Multiindex для некоторого набора элементов может создать несколько интерфейсов для работы с ним. (Multiindex хранит набор индексов.) Он позволяет использовать методы каждого из них применительно к данному набору. С помощью Multiindex разработчик может пользоваться преимуществами сразу нескольких интерфейсов, выбирая для каждой ситуации наиболее подходящий.