**Контрольные вопросы №2**

Аляев Роман

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

Пару и кортеж применяются в тех случаях, когда есть следующие требования:

* Чётко понятно, что находится в паре. Нет никаких способов двусмысленной трактовки.
* Порядок не имеет значения. Что сначала число для первого игрока, потом для второго, что наоборот — по барабану.
* Наша игра только на двух игроков и никогда (by design) не будет возможна для большего количества — беспокоиться о расширяемости не нужно
* Оба значения точно должны быть. Отсутствие одного из них невозможно.
* Проверка валидности не нужна — по определению весь диапазон int нам подходит.

# Когда следует использовать контейнер std::array?

У массивов есть свои недостатки: фиксированные массивы распадаются в указатели, теряя информацию о своей длине; в динамических массивах проблемы могут возникнуть с освобождением памяти и с попытками изменить их длину после выделения.

Поэтому в Стандартную библиотеку C++ добавили функционал, который упрощает процесс управления массивами: std::array.

std::array — это фиксированный массив, который не распадается в указатель при передаче в функцию. std::array определяется в заголовочном файле array. Рекомендуется использовать std::array вместо стандартных фиксированных массивов в любых нетривиальных задачах.

# Когда следует использовать контейнер std::vector?

Аналогично предыдущему вопросу в стандартную библиотеку добавили std::vector – некоторая улучшенная версия динамических массивов. std::vector — это тот же динамический массив, но который может сам управлять выделенной себе памятью. Это означает, что вы можете создавать массивы, длина которых задается во время выполнения, без использования операторов new и delete (явного указания выделения и освобождения памяти).

# Когда следует использовать контейнер std::deque?

std::deque – это двусторонняя очередь. Контейнер deque очень похож на контейнер — вектор, так же, как и векторы, deque являются динамическими массивами. Разница между вектором и deque состоит лишь в том, что в deque динамический массив открыт с двух сторон. Это и позволяет очень быстро добавлять новые элементы как в конец, так и в начало контейнера. В векторах элементы можно добавлять лишь в конец массива.

# Когда следует использовать контейнер std::list?

std::list – это двусвязный список. Это структура данных, которая построена на двусвязных списках. Это значит, что любой элемент знает только о предыдущем и о следующем элементах. Программисты используют этот контейнер из-за быстрого добавления и удаление значений. Это происходит так быстро, потому что не приходиться перемещать элементы между собой, нужно лишь правильно манипулировать указателями.

# Когда следует использовать контейнер std::forward\_list?

std::forward\_list – это односвязный список, применение аналогично предыдущему, только он поддерживает только одностороннюю связь с элементами и позволяет удалять и добавлять элементы в начале и в конце списка.

# Какие адаптеры контейнеров есть в стандартной библиотеке?

Существуют следующие контейнеры на основе предыдущих:

* Stack – можно добавлять и убирать элементы с верхушки стека
* Queque – односторонняя очередь
* Priority\_queue – очередь с приоритетом
* Bitset - тип объекта, который хранит последовательность, состоящую из фиксированного числа битов, предоставляющих компактный способ хранения флагов для набора элементов или условий.

# Когда следует использовать контейнер circular buffer из Boost?

Кольцевой буфер — это структура данных, использующая единственный буфер фиксированного размера, как будто бы после последнего элемента сразу же снова идет первый. Такая структура легко предоставляет возможность буферизации потоков данных.

Данные структуры часто используют для организации различных очередей сообщений и буферов приёма-передачи различных коммуникационных интерфейсов.

# Почему контейнер circular buffer из Boost не может войти в стандарт?

Как и было сказано на семинаре, в отличие от других контейнеров, в кольцевом буффере не выполняется соотношение begin < end. Поэтому могут возникать ошибки.

# Какие типы данных для работы с многомерными массивами вы можете назвать?

Можно представить следующие типы:

* Динамический массив
* Двумерный массив
* Вектор векторов
* boost::multi\_array