Думай как ревьюер

Перед тем как вы впервые получите код своих однокурсников для ревью, прочтите небольшую инструкцию. Она поможет вам сфокусироваться и не пропустить важные моменты в проверке. Эта инструкция — немного адаптированные рекомендации для ревьюеров, которые проверяют ваш код на протяжении всего курса.

- В коде нет неоправданного дублирования

  Повторяющийся код лучше убирать в функции или шаблоны.

- Код понятен и легко читается

Имена всех объектов, классов, переменных и функций должны быть понятными и отражать функционал. Названия классов и переменных должны отвечать на вопрос «Что?» (например, `Document` или `Dog`), а названия функций и методов — на вопрос «Что делать?» (например, `CheckDocument` или `FeedDog`). Форматирование может не совпадать с тем, как делаете это вы, но должно быть консистентным по всему коду. Код содержит необходимое количество комментариев. При этом, если функционал и так ясен благодаря удачному неймингу, комментарии могут вообще не требоваться.

- Отношение к переменным — бережное

  Переменные объявляются в области видимости их использования и инициализированы. Нет глобальных переменных кроме констант. Автор не злоупотребляет `auto`.

- Константность методов, аргументов и переменных тщательно продумана

  Все указатели и ссылки, которые должны быть константными по своей сути, объявлены с использованием `const`. То же касается методов класса и автоматических переменных. При этом, если параметр передаётся по значению, требовать `const` не нужно. Встреча с `const_cast` должна заставить задуматься, всё ли автор делает верно.

- Код эффективен

  Посмотрите на алгоритмы, которые применяет автор, и оцените, насколько они эффективны. Если самописный алгоритм можно заменить стандартным из библиотеки, лучше укажите автору на это, ведь обычно такие алгоритмы работают быстрее. Обращайте внимание на копирования объектов,

предлагайте, как этого избежать.

- Функции и их параметры продуманы

  Функции короткие и выполняют понятный функционал. Тяжёлые параметры передаются в функции по константным ссылкам, кроме случаев, когда данные нужно копировать. Лёгкие параметры, такие как базовые типы, итераторы, функциональные объекты или `string_view`, лучше передавать по значению. Обращайте внимание автора на слишком длинные списки аргументов. Обычно это сигнал к объединению их в структуру или к разделению действий, выполняемых одной функцией, между несколькими функциями.

- Дизайн классов и структур логичен и удобен

  У классов не должно быть публичных нестатических полей. Все поля класса спрятаны в `private`, и доступ к ним организован через специальные методы. Здесь тоже стоит обратить внимание на константность. У структур не должно быть приватных секций, это прерогатива классов.

- Ошибки

  Если думаете, что нашли в коде ошибку, которую не поймал тренажёр, протестируйте своё предположение.

Код-ревью — это одновременно объективный и субъективный процесс. Но в конечном итоге его основная цель — это хороший код. Всегда важно помнить, что хороший код — это не что-нибудь однозначное. Если с объективными показателями спорить сложно, то субъективных ответов на вопрос «Что такое хороший код?» — множество. Чтобы код-ревью было полезным и продуктивным для обеих сторон, старайтесь сохранять баланс и слушать друг друга, принимать чужую точку зрения и отстаивать свою. Помните, что у вас общие цели и общий успех на двоих.