Матрица компетентности программиста

- Теория
- Навыки
- Программирование
- Опыт
- Знания
- Темы для проработки

Теория

Код	Тема	2^n (Уровень 0)	n^2 (Уровень 1)	n (Уровень 2)	log(n) (Уровень 3)
T1	Структуры данных	Не понимает разницы между массивом и связным списком.	Может объяснить и использовать на практике массивы, связные списки, словари и т.д.	Понимает плюсы и минусы использования тех или иных базовых структур данных (размер памяти, время выполнения операций с данными, в чем разница между массивами и связными списками в этом плане). Может объяснить как реализовать хэш-таблицы и как обработать коллизии. Приоритетные очереди и способы их реализации, и т.д.	Знание сложных структур данных, таких как B-дерево, Биномиальная куча и Фибоначчиевская куча, АВЛ-дерево, Красно-чёрное дерево, Косое дерево, Список с пропусками, TRIE-структуры и т.д.
T2	Алгоритмы	Не может найти среднее значение массива чисел. (Тяжело поверить, но встречаются и такие кандидаты на собеседовании.)	Базовые методы сортировки и поиска. Обход и поиск в структурах данных.	Деревья, Графы, "простой путь" и "разделяй и властвуй"-алгоритмы. Понимает ассоциацию уровней этой матрицы.	Может распознать и написать решение динамическим программированием, хорошо знает алгоритмы на графах, хорошо знает численные методы, может идентифицировать проблемы класса NP.
T3	Системное программирование	Не знает что такое компилятор, компоновщик или интерпретатор.	Базовое понимание компиляторов, компоновщиков и интерпретаторов. Будет плюсом, если понимает, что такое ассемблерный код и как работают программы на уровне железа. Небольшое понимание виртуальной памяти и пэйджинга.	Понимает чем отличается kernel mode от user mode, что такое мульти-трединг. Знает способы синхронизации и как реализованы примитивы синхронизации. Плюсом будет, если может читать ассемблерный код. Понимает, как работают сети, сетевые протоколы и может реализовать передачу данных через сокеты.	Понимает как работает весь "программный стэк": железо (CPU + Память + Кэш + Прерывания + микрокоды), двоичный код, ассемблер, статическая и динамическая линковка, компиляция, интерпретация, JIT-компиляция, сборка мусора, куча, стэк, адресация памяти.

Навыки

Код	Тема	2^n (Уровень 0)	n^2 (Уровень 1 - Junior)	n (Уровень 2 - Middle)	log(n) (Уровень 3 - Senior)
S1	Контроль версий исходников	Пользуется git, и делает коммиты с комментариями типа "четверг" в мастер	Следует GitFlow	Умеет делать склейку комитов, черри-пик, понимает плюсы и минусы merge vs rebase, предпочитает cli	выбирать структуру репозиториев и модель ветвления под команду/проект
S2	Автоматизация сборки	Знает как запустить сборку из среды программирования.	Умеет собирать из командной строки.	Может настроить скрипт для сборки основной системы	Может настроить скрипт для сборки системы и документации, для сборки инсталляторов. Сделает и добавит код скрипта в систему контроля версий исходников
S3	Автоматизация тестирования	Считает, что тестирование - это работа тестеров.	Может создавать свои хорошие юнит-тесты для кода, который пишет в настоящее время.	Может создавать автоматические тесты на функционал, пользовательский интерфейс и загрузку/производительность.	Пишет код в стиле Test-driven Development (TDD).

Программирование

Код	Тема	2^n (Уровень 0)	n^2 (Уровень 1 - Junior)	n (Уровень 2 - Middle)	log(n) (Уровень 3 - Senior)
D1	Декомпозиция задачи	Просто последовательные строчки. Copy/Paste - для повторного использования кода.	Может разбивать решение задачи на несколько функций.	Способен создавать многократно используемые функции/объекты, которые решают общие задачи.	Использует соответствующие структуры данных и алгоритмы. Создает общий/объектно-ориентированный код, который инкапсулирует те условия задачи, которые могут быть изменены.
D2	Декомпозиция системы	Не способен думать о системе более сложной, чем один класс или файл.	Может произвести декомпозицию задачи и спроектировать систему в пределах одной платформы или технологии.	Может спроектировать систему, которая охватывает несколько технологий/платформ.	Может визуализировать и проектировать сложные системы с несколькими линейками продуктов, учитывать интеграциею с внешними системами. Также должен уметь проектировать системы поддержки работы: мониторинг, генерация отчетов, аварийные переходы на использование запасных ресурсов.

COM1	Общение	Не может выразить свои мысли/идеи. Плохо с правописанием и грамматикой.	Его понимают. Хорошие правописание и грамматика.	Может эффективно общаться.	Может понимать и объяснять мысли/дизайн/идеи /специфику в точно выраженной форме. В общении соответствует ситуации.
CO1	Организация кода в файле	Нет четкой организации кода в файле.	Методы сгруппированны логически и по вызовам.	Код разделен на регионы, имеет хорошие комментарии, в т.ч. со ссылками на другие файлы исходников.	Файл имеет разделы "license header", "summary", хорошие комментарии, непротиворечивую расстановку пробелов и табуляции. Файл должен выглядеть красиво.
CO2	Организация кода между файлами	Не приходит в голову мысль четко организовать код с помощью разделения на файлы.	Похожие файлы группируются в папку.	Каждый физический файл предназначен для чего-то одного, например, служит для объявления одного класса или для реализации одного функционала и т.д.	Организация кода на физическом уровне точно соответствует проекту. Глядя на имена файлов и структуру папок, можно понять как спроектирована данная реализация.
CO3	Организация дерева исходников	Все в одной папке.	Простое разделение кода в логические подкаталоги.	Нет "круговых" зависимостей. Бинарники, либы, документация, билды, сторонний код - все разложено в соответствующие папки.	Структура дерева исходного кода соответствует логической иерархии и организации кода в проекте. Глядя на имена файлов и структуру папок, можно понять как спроектирована данная система.
R1	Читабельность кода	Односложные имена.	Хорошие имена файлов, переменных, классов, методов и т.д.	Нет длинных функций, а нестандартный код, багфиксы и допущения в коде поясняются комментариями.	Допущения в коде сопровождаются assert'ами. Поток операций в коде естественный - нет глубокой вложенности условий или методов.
DF1	Безопасное программирование (defensive coding)	Не понимает данной концепции.	В некоторых методах проверяет аргументы.	Проверяет все аргументы и ставит assert'ы на критические допущения в коде. Убеждается, что проверил возвращаемое значение и что обрабатывает исключения в потенциально бажном коде.	Имеет свою собственную библиотеку помогающую в безопасном программировании, пишет юнит-тесты которые эмулируют сбои.

DF2	Обработка ошибок	Пишет код для "идеального" случая, когда все работает и нет сбоев.	Обработка ошибок в коде, который либо кидает исключение, либо генерирует ошибку.	Убеждается, что после того, как произошла ошибка/исключение, программа продолжает работать, а ненужные более ресурсы, коннекшоны и память были корректно освобождены обработчиком ошибки.	Пишет код так, чтобы определять возможные ошибки на раннем этапе, придерживается последовательной стратегии обработки исключений во всех слоях кода, разрабатывает общие принципы обработки исключений во всей системе.
TL1	Среда программирования IDE	В основном использует IDE для редактирования текста.	Способен эффективно пользоваться меню в IDE. Знает некоторые тонкости среды.	Для самых используемых функций среды знает горячие клавиши.	Написал свои макросы.
API1	API	Часто нуждается в обращениях к документации.	Помнит самые часто используемые API.	Обширные и глубокие знания API.	Написал библиотеки, которые оборачивают API, для упрощения задач, которые наиболее часто встречаются. Эти библиотеки также часто восполняют пробелы в API.
API2	Фреймфорки	Не использует никаких фреймверков за рамками основной платформы.	Знает, но не использует популярные фреймворки, доступные для его платформы.	Профессионально пользовался более чем одним фреймворком и хорошо разбирается в идиомах фреймворков.	Является автором фреймворка.
RQ1	Требования	Понимает выставленные требования и пишет код в соответствии со спецификацией.	Задает вопросы касающиеся не рассмотренных в спецификации случаев.	Понимает картину в целом и предлагает дополнительные аспекты, которые должны быть описаны в спецификации.	Может предложить лучшие альтернативы и следовать выставленным требованиям, основываясь на собственном опыте.
TL2	Скрипты	Отсутствует знание скриптовых инструментов.	Batch-файлы/shell.	Perl/Python/Ruby/VBScript/Powershell.	Писал и публиковал повторно используемые скрипты.

DB1	Базы Данных	Думает, что Excel - это база данных.	Знает основы баз данных, нормализацию, ACID, транзакции и может написать простые select'ы.	Может спроектировать хорошие нормализованные схемы БД, с учетом запросов, которые будут выполняться; умело использует представления, хранимые процедуры, триггеры и собственные типы данных. Понимает разницу между кластеризованными и не-кластеризованными индексами. Специалист в использовании ORM.	~~Может осуществлять администрирование БД, оптимизацию производительности БД, индексную оптимизацию, писать сложные select'ы, может заменить использование курсора вызовами функции sql,~~ понимает как данные хранятся внутри, как хранятся индексы, имеет представление о том, как зеркалятся и реплицируются БД и т.д. Понимает как работает двухфазный commit.

Опыт

Код	Тема	2^n (Уровень 0)	n^2 (Уровень 1 - Junior)	n (Уровень 2 - Middle)	log(n) (Уровень 3 - Senior)
EXP1	Языки и профессиональный опыт	Императивные или объектно-ориентированные программирования	Императивные, объектно-ориентированные и декларативные(SQL) языки программирования. Дополнительный бонус - если понимает разницу между статической и динамической, слабой и строгой типизацией, статически выводимыми типами.	Функциональные языки программирования. Дополнительный бонус - если знает, что такое "ленивые вычисления", каррирование, продолжения.	Конкурентные (Erlang, Oz) и логические (Prolog).
EXP2	Годы профессионального опыта	1	2-5	6-9	10+
EXP3	Годы профессионального опыта конкретной платформы	1	2-3	4-5	6+
EXP4	Знание предметной области	Не знает о понятии "предметная область".	Работал хотя бы над одним продуктом в своей предметной области.	Работал над несколькими продуктами в одной и той же предметной области.	Эксперт своей предметной области, проектировал и реализовывал несколько продуктов/решений в ней, хорошо разбирается в ее сущностях и протоколах.

Знания

Код	Тема	2^n (Уровень 0)	n^2 (Уровень 1 - Junior)	n (Уровень 2 - Middle)	log(n) (Уровень 3 - Senior)

TL3	Инструментарии	Ограничены используемой IDE (VS.Net, Eclipse и т.д.)	Знает о некоторых альтернативах популярным стандартным инструментариям.	Хорошие знания редакторов кода, отладчиков, различных IDE, open-source альтернативах и т.д. (Например, это может быть кто-то, кто знает большинство тулзов из списка Скота Ганзельмана.) Использует ORM-тулзы.	Написал свои инструментарии и скрипты, дополнительный плюс - если эти скрипты были опубликованы.
CB1	Code base (кодовая база)	Никогда не смотрел кодовую базу.	Имеет общее представление о расположении кода и о том, как его собрать.	Хорошие рабочие знания кодовой базы, реализовывал несколько багфиксов и, может быть, некоторые маленькие фичи.	Реализовал несколько больших фич в кодовой базе, может легко описать изменения, требуемые для реализации большинства фич или багфисков.
NEW1	Знание новейших технологий	Не слышал о новейших технологиях.	Слышал о новейших технологиях в своей области.	Скачивал alpha/preview/CTP /beta-версии и читал некоторые статьи и руководства на эти темы.	Пробовал сделать что-либо сам. Используя preview-версию сбилдил свою программу. Дополнительный плюс - если сделал свое решение доступным для других.
API3	Знание внутренних аспектов платформы	Нулевые знания внутренних аспектов платформы.	В основном, знает как работает платформа внутри.	Имеет глубокие познания внутренних аспектов платформы и может обрисовать, как платформа исполняет программный код.	Написал свои тулзы для расширения возможностей платформы или для извлечения дополнительной информации о работе платформы. Например расширения дизассемблера, декомпилятора, отладчика и т.д.

B1	Книги	Серии книг "... за 21 день" "... за 24 часа" "... для чайников".	Совершенный код Не заставляйте меня думать! Регулярные выражения	Человеческий фактор: успешные проекты и команды Приемы объектно-ориентиро-ванного проектирования. Паттерны проектирования Жемчужины проектирования Руководство по разработке алгоритмов Программист-прагматик Мифический человеко - месяц	Искусство программирования Структура и интерпретация компьютерных программ Concepts Techniques and Models of Computer Programming Database systems , by C. J Date Thinking Forth Little Schemer
B2	Блоги	Слышал о блогах, но не уделял им внимания.	Читает технические блоги, блоги о программировании и разработке ПО и регулярно слушает подкасты.	Ведет ссылочный блог, содержащий коллекции ссылок на полезные статьи и тулзы, которые [ссылки] он собирает.	Ведет блог, в котором содержится его собственные понимание вопросов программирования.

Темы для проработки

- Распределенные вычисления
- DevOps
- ООП
- ФП
- Git
- ОС, виртуализация, контейнеризация
- Сеть, OSI, TCP/IP, HTTP, FTP, SMTP,
- СУБД
- Кеширование
- Архитектурные подходы
    - Layered Architecture
    - MVC
    - MVVM
    - CleanArchitecture
    - VIPER
- Книги
- UML
- Системное мышление
- Аналитический склад ума
- Презентационные навыки