**Матрица компетентности программиста**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Область програм­мирования** | **Уровень** | | | |  |
| **Теория:** | **2n(Уровень 0)** | **n2 (Уровень 1)** | **n (Уровень 2)** | **log(n) (Уровень 3)** | **Коммен­тарии** |
| **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. |  |
| **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 значимость уровней приведенной здесь матрицы. | Может распоз­нать и написать реше­ние динамическим программирова­нием, хорошо знает алгорит­мы на графах, хорошо знает численные методы, может идентифициро­вать проблемы класса NP. | Работать с теми, кто имеет хороший рейтинг в TopCoder — это неверо­ятная удача! |
| **Системное програм­мирование** | Не знает, что такое компилятор, линковщик или интерпретатор. | Базовое понимание компи­ляторов, линковщиков и интерпретато­ров. 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, спосо­бы синхро­низа­ции и как реа­лизованы при­митивы син­хро­низации, 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:** | **2n(Уровень 0)** | **n2 (Уровень 1)** | **n (Уровень 2)** | **log(n) (Уровень 3)** | **Коммен­тарии** |
| **Контроль версий исходников** | Бэкап исходников в папку с датой бэкапа. | VSS и основы CVS/SVN в качестве пользователя | Специалист по возможностям CVS and SVN. Знает, как разветвить и слить, настро­ить репозито­рий и т.д. | Знает распре­деленные системы VCS. Пробовал Bzr/Mercurial/Darcs/Git. |  |
| **Автома­тизация build'ов** | Знает, как запустить Build из среды программирова­ния. | Умеет билдить из командной строки. | Может настроить скрипт для сборки основной системы. | Может настро­ить скрипт для сборки системы и документа­ции, инсталля­торов, генера­торов "release notes" и доба­вить скрипт в систему кон­троля версий исходников. |  |
| **Автомати­зированное тести­рование** | Думает, что тестирова­ние — это работа тестеров. | Написал автоматизи­рованные юнит-тесты и может создавать свои хорошие юнит-тесты для кода, который пишет в настоящее время. | Пишет код в стиле Test-driven Development (TDD). | Понимает и мо­жет настроить автоматические тесты на функ­ционал, пользо­вательский интерфейс и загрузку/производитель­ность... |  |
| **Програм­мирование:** | **2n(Уровень 0)** | **n2 (Уровень 1)** | **n (Уровень 2)** | **log(n) (Уровень 3)** | **Коммен­тарии** |
| **Деком­позиция задачи** | Просто после­довательные строчки кода, а 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. |  |
| **Деком­позиция системы** | Не способен думать о системе сложнее одного класса или файла. | 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 и интег­рацией с внеш­ними система­ми. Также дол­жен уметь про­ектировать сис­темы поддерж­ки работы: мо­ниторинг, гене­рация отчетов, аварийные переходы на использование запасных ресурсов. |  |
| **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. | Важность этого кри­терия для программис­та часто не­дооцени­ва­ют. С уве­личением аутсор­син­га разработ­ки ПО в те стра­ны, где английский не явля­ется родным язы­ком, этот воп­рос стал более акту­альным. Я знаю нес­колько про­ектов, кото­рые прова­лились потому, что программис­ты не могли понять смысл об­суждения. |
| **Организация кода в файле** | Нет четкой организации в файле. | Методы сгрупированны 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. |  |
| **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, и, глядя на имена файлов и структуру папок, можно понять, как спроектирована данная реализация. |  |
| **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. | Разница между этим пунктом и предыду­щим состоит в масш­табе организации. Организа­ция дерева исходников относится ко всему комплексу продуктов, которые определяют систему. |
| **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. |  |
| **Безопасное програм­миро­вание (defen­sive coding)** | Не понимает данной концепции. | 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. |  |
| **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, коннекшены 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. |  |
| **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. |  |
| **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. | Примером API может быть Java-библиотека,  .NET фреймворк или API какого-либо прило­жения. |
| **Фреймворки** | Не использует никаких фреймворков 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. |  |
| **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. |  |
| **Скрипты** | Отсутствует знание скриптовых инструментов. | Batch-файлы/shell. | Perl/Python/Ruby/VBScript/Powershell. | Писал и публи­ковал повторно используемые скрипты. |  |
| **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-тулзов. | Может осуществлять администриро­ва­ние БД, опти­мизацию произ­водитель­ности БД, индексную оптимизацию, писать сложные select'ы, может заменить использование курсора вызо­вами функций 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:** | **2n(Уровень 0)** | **n2 (Уровень 1)** | **n (Уровень 2)** | **log(n) (Уровень 3)** | **Коммен­тарии** |
| **Языки и професси­ональный опыт** | Императивные или объектно-ориентирован­ные языки программирова­ния | 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). |  |
| **Годы професси­онального опыта** | 1 | 2−5 | 6−9 | 10+ |  |
| **Годы професси­онального опыта исполь­зования платформы** | 1 | 2−3 | 4−5 | 6+ |  |
| **Знание предметной области** | Не знает о понятии "предметная область". | Работал хотя бы над одним продуктом в 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:** | **2n(Уровень 0)** | **n2 (Уровень 1)** | **n (Уровень 2)** | **log(n) (Уровень 3)** | **Коммен­тарии** |
| **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. |  |
| **Code base** | Никогда не смотрел code base. | Имеет общее представление о располо­жении кода и о том, как его собрать. | Хорошие рабочие знания code base, реализовывал несколько багфиксов и, может быть, некоторые маленькие фичи. | Реализовал несколько боль­ших фич в code base и может легко описать изменения, тре­буемые для ре­ализации боль­шинства фич или багфиксов. |  |
| **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, и исполь­зуя preview-версию, сбил­дил свою прог­рамму. Допол­нительный плюс — если сделал свое ре­шение доступ­ным для других. |  |
| **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 превращает программу в исполняемый код. | Написал свои тулзы для расширения возможностей платформы или для извлечения дополнительной информации о работе платформы. Например, дизассемблер, декомпилятор, отладчик и т.д. |  |
| **Книги** | Серии книг "... за 21 день" "... за 24 часа" "... для чайников". | Совершенный код   Не заставляйте меня думать!   Регулярные выражения | Приемы объектно-ори­ентированного проектирова­ния. Паттерны проектирования   Человеческий фактор: успеш­ные проекты и команды   Жемчужины программирова­ния   Algorithm Design Manual   Программист-прагматик   Мифический че­ловеко-месяц | Structure and Interpretation of Computer Programs   Concepts Techniques and Models of Computer Programming   Искусство програм­мирования   Database systems, by C. J Date   Thinking Forth   Little Schemer |  |
| **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 он собирает. | Ведет блог, в котором содержатся его собственные понимание вопросов и соображения на тему про­граммирования. |  |