

Программирование ITmozg:

1 Jun 2018

# Матрица компетентности программиста

**4 уровня компетентности программиста от новичка до гуру разложены по 15 условным полочкам, вот только несколько из них:**

- знание алгоритмов,
- умение организовать контроль версий,
- опыт проектирования сложных систем,
- читабельность кода.

## Матрица компетентности программиста ч.І.

Область	Уровень				Комментарии
	2^n (Уровень 0)	n^2 (Уровень 1)	n (Уровень 2)	log(n) (Уровень 3)	
<b>Теория</b>					
<b>Структуры данных</b>	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	Знание сложных структур данных, таких как B-дерево, Биномиальная куча и Фибоначчиевская куча, AVL-дерево, Красно-чёрное дерево, Косое дерево, Список с пропусками, TRIE-структуры и т.д.	

<b>Навыки</b>					
<b>Контроль версий исходников</b>	Бэкап исходников в папку с датой бэкапа.	VSS и основы CVS/SVN в качестве пользователя.	Специалист по возможностям CVS и SVN. Знает как создать, разветвить, слить репозиторий и т.д.	Знает распределенные системы VCS. Работает с Mercurial/Darcs/Git	
<b>Автоматизация build'ов</b>	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, для сборки инсталляторов. Сделает и добавит код скрипта в систему контроля версий исходников.	
<b>Автоматизация тестирования</b>	Считает, что тестирование - это работа тестеров.	Может создавать свои хорошие юнит-тесты для кода, который пишет в настоящее время.	Пишет код в стиле Test-driven Development (TDD).	Может создавать автоматические тесты на функционал, пользовательский интерфейс и загрузку/производительность.	

Программирование					
Декомпозиция задачи	Просто последовательные строки. 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 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 интеграцию 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 общения 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





<b>Организация дерева исходников</b>	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.
<b>Читабель- ность кода</b>	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.	
<b>Безопасное программи- рование (defensive coding)</b>	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	Имеет свою собственную библиотеку помогающую в <i>безопасном программировании</i> , пишет юнит-тесты	

## Матрица компетентности программиста ч.II.

Область	Уровень				Комментарии
	2^n (Уровень 0)	n^2 (Уровень 1)	n (Уровень 2)	log(n) (Уровень 3)	
<b>Программирование</b>					
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	Примером API может быть Java-библиотека, .Net фреймворк или API какого-либо приложения.

<b>Опыт</b>					
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).	
Годы профессионального опыта	1	2-5	6-9	10+	
Годы профессионального опыта конкретной платформы	1	2-3	4-5	6+	
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					
Инструментарии	Ограничены используемой IDE ( <a href="#">VS.Net</a> , <a href="#">Eclipse</a> 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 Анзельмана.) <a href="#">Использует ORM-тулзы.</a>	Написал свои инструментарии и скрипты, дополнительный плюс - если эти скрипты были опубликованы.	
<a href="#">Code base (кодовая база)</a>	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 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. 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 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.	
Книги	 <p>Серии книг "... за 21 день" "... за 24 часа" "... для чайников".</p>	 <p><a href="#">Совершенный код</a> <a href="#">Не заставляйте меня думать!</a> <a href="#">Регулярные выражения</a></p>	 <p><a href="#">Человеческий фактор: успешные проекты и команды</a> <a href="#">Приемы объектно-ориентированного проектирования.</a> <a href="#">Паттерны проектирования</a> <a href="#">Жемчужины проектирования</a></p>	 <p><a href="#">Искусство программирования</a> <a href="#">Структура и интерпретация компьютерных программ</a> <a href="#">Concepts Techniques and Models of Computer</a></p>	

## Матрица компетентности программиста ч.І.

## Матрица компетентности программиста ч.ІІ.