
andreycha
20 янв 2011 в 23:18

10 способов стать хорошим программистом

5 мин 101K

Программирование*

Перевод

Автор оригинала: Ashish Arya

> *Хороший программист — это тот, кто смотрит в обе стороны, переходя дорогу с односторонним движением.*
> Даг Линдер

Что приводит программиста на его рабочее место каждый день? Страсть к программированию и получение удовольствия от него. Но чтобы действительно получать удовольствие и радость от программирования, нужно знать некоторые базовые вещи, которые позволят вам стать хорошим программистом.

Я не собираюсь писать мантры, следуя которым, вы станете хорошим специалистом. Моя цель — перечислить те вещи, которые помогли мне добиться успехов в этой профессии. Не существует четкого определения, кто такой хороший программист. Под таковым я подразумеваю человека, который разрабатывает отличные IT-решения и вносит вклад в развитие индустрии.

1. Учите основы

Понимание основ — это ключ к успеху в любой индустрии и любой профессии. До тех пор, пока вы недостаточно хорошо знаете основы, вы не сможете стать хорошим программистом. Знание азов позволит вам разрабатывать и реализовывать лучшие решения наилучшим способом. Если вы ощущаете пробелы в своих знаниях, будь то основы computer science, или концепции языка, на котором вы пишете, то никогда не поздно вернуться назад и повторить забытое.

2. Задавайте вопросы (как? почему?), когда пишете код

Есть одна вещь, которая отличает хорошего программиста от всех остальных — это желание знать, что и как происходит. Есть люди, которые никогда не оставят в покое код, пока точно не

будут знать, что именно происходит при его выполнении. Я понимаю, что это приближает дедлайн, что у нас не всегда есть на это время, и поэтому мы часто заканчиваем работать с кодом, как только он начинает выполнять свои функции. И хотя поведение в подобных ситуациях это тема для другого разговора, каждый программист может приложить как можно больше усилий для того, чтобы вникнуть в работу кода. И поверьте, со временем это войдет в привычку, и вы будете делать это уже неосознанно.

3. Учите других — учитесь сами

Большинство из нас обращаются к форумам и группам только тогда, когда нам нужна помощь. Еще одна вещь, которая отличает хорошего программиста от все остальных: хороший программист чаще заглядывает в такие места, чтобы помочь другим. Такая помощь учит больше, чем помощь, оказанная вам при решении вашей проблемы. Поверьте, после того как вы разберетесь в чужой проблеме и ее контексте, поразмышляете над ней и дадите решение, вы научитесь гораздо большему.

4. Пишите простой, понятный, но в то же время логичный код

Как и в других областях, формула KISS (Keep it simple and short — делай короче и проще) работает и в программировании. Пишите логичный код и избегайте усложнений. Иногда люди пишут сложный код только для того, чтобы доказать, что они умеют писать такой код. Мой опыт подсказывает, что простой и логичный код всегда работает хорошо, приносит меньше проблем и лучше поддается расширению. Вспоминается отличная фраза:

> *Хороший код — это лучшая документация. Каждый раз, когда вы захотите добавить комментарий, спросите себя: «Как я могу улучшить этот код, чтобы он не требовал комментирования?»*
> Стив МакКоннелл

5. Уделяйте больше времени анализу проблемы, тогда вам понадобится меньше времени для ее устранения

Уделяйте больше времени на понимание и анализ проблемы и разработку решения. А остальное будет легко сделать. Разработка решения не означает использование языков или инструментов для моделирования, вы можете просто смотреть на небо и думать о решении. У тех, кто привык стучать по клавиатуре сразу же, как только узнал о проблеме, результат обычно не совпадает с ожидаемым.

> *Если вы не можете целиком понять общую структуру программы, пока принимаете душ, значит, вы не готовы ее запрограммировать.*
> Ричард Паттис

6. Будьте первым, кто проанализирует и оценит ваш код

Хотя это трудно, но попробуйте «сломать» ваш код до того, как это сделает кто-то другой. Со временем вы научитесь писать почти безошибочный код. Всегда проводите подробную и беспристрастную оценку своего кода. И никогда не бойтесь спрашивать, что другие думают о вашем коде. Работайте с хорошими программистами и прислушивайтесь к их мнению — это поможет вам стать хорошим программистом.

7. Не пугайтесь быстрой смены технологий

За все время работы в области IT, я встречал множество людей, которых не устраивала их работа, и людей, которые меняли место работы, чтобы работать с новейшими технологиями. В таком стремлении нет ничего плохого, однако ошибка в «новейших технологиях». Каждый день появляются новые инструменты, API и фреймворки, призванные сделать разработку быстрой и простой. И эта тенденция не снизится. Однако следует понять одну вещь: фундаментальные знания и основы меняются значительно медленнее, чем фреймворки, новые инструменты и API. Можно провести аналогию с морем, на поверхности которого находятся быстрые течения, однако на глубине вода спокойна и она составляет большую часть объема. Поэтому держитесь «на глубине», поближе к основам. В мире Java приложений уровня enterprise существует много веб-фреймворков, а новые выходят каждые две недели. Однако основы клиент-серверной архитектуры, шаблона MVS (Model View Separation), фильтров/сервлетов/JSP, упаковки ресурсов, обработки XML и т.д. остаются неизменны. Поэтому лучше потратьте время на изучение этих основ, нежели на изучение вечно меняющихся фреймворков. Поверьте, зная основы, изучить новые API и фрейморки будет куда легче.

8. «Костыли»* долго не работают

Множество программистов используют «костыли»: от недостатка времени, понимания проблемы или опыта. Однако со временем такие решения делают код хуже: он становится менее расширяем и удобен в поддержке. Всегда старайтесь написать такую реализацию, о которой вы знаете все. Я понимаю, что «костыли» в некоторых ситуациях неизбежны, но тогда ситуация напоминает что-то вроде «всегда говори правду, но иногда можешь соврать».

9. Читайте документацию

Хорошие программисты читают много документации. Это могут быть спецификации, JSR, API, документы, туториалы и т.д. Чтение документации позволит вам понимать основы, и вы будете решать задачи наилучшим способом.

10. Чужой код тоже может чему-то научить

Я работал с двумя отличными программистами, которые постоянно в своих IDE держали исходники чужих проектов на Java, и обращались к ним каждый день. Они делали это не только из желания узнать, как работают базовые вещи, но и из желания научиться писать хорошие программы. Чтение исходных кодов известного open source проекта, или кодов, написанных вашим ведущим программистом, может помочь вам писать код лучше.

И последнее: не сравнивайте себя с другими

Сравнение себя с другими выльется только в плохое самочувствие и нездоровую конкуренцию. У всех есть свои сильные и слабые стороны. Важнее понять свои сильные и слабые стороны и работать над ними. Я много раз видел, как даже так называемые fundoo-программисты (программисты с хорошей фундаментальной подготовкой) делали глупые ошибки. Поэтому проанализируйте и запишите те вещи, которые вам стоит улучшить в себе, и за работу. Программируйте в удовольствие и наслаждайтесь этим.

> *Любой дурак может написать код, понятный компьютеру. Хороший программист пишет код, понятный человеку.*
> Мартин Фаулер

** — достаточно вольный перевод слова work-arounds*

Теги: программирование, разработка, советы, развитие

Хабы: Программирование

+168 376 93

Редакторский дайджест
Присылаем лучшие статьи раз в месяц

Электропочта

136 Карма 0 Рейтинг

Андрей Часовских @andreycha
Пользователь

Подписаться

Комментарии 93

mambet
20 янв 2011 в 23:54

> Хороший программист — это тот, кто смотрит в обе стороны, переходя дорогу с односторонним движением.

Ух ты! Всегда так делаю. Но я тестер, мне иначе и нельзя…

Ответить

kurd
21 янв 2011 в 00:01

Мы, как тестеры, еще под ноги, вверх смотрим. Обязательно пару раз проверим, потом все равно заведем баг на то, что кирпич висит криво и надо бы перевесить.

Ответить

Tagrim
21 янв 2011 в 00:03

а то не на того упадет?)

Ответить

mambet
21 янв 2011 в 00:06

Иначе водитель может потом в суде сказать, что знак не по ГОСТу повешен.

Ответить

tol
21 янв 2011 в 01:19

ГОСТ сегодня утром поменяли. Скажет тебе дядя с тремя буквами на сипине:)

Ответить

aavezel
21 янв 2011 в 10:09

Ну… У него можно премию снять в размере 20к… Т. к. поправки в ГОСТ по безопасности вносит только зам БОССА по внутренним делам…

Ответить

angel_hearted
21 янв 2011 в 19:26

Что-то я не о тех буквах подумал. Хотя как раз это слово им подходит намного больше.

Ответить

SadRunner
21 янв 2011 в 00:20

Вверх, вниз, вправо, влево, вперед и назад по временнОй оси. Ну и не забыть заглянуть в пару тройку параллельных реальностей. Тяжкий труд, за который команда вам признательна.

Ответить

foregen
21 янв 2011 в 01:32

Тестер сначала идет.
Потом смотрит и идет.
И так далее от простого к сложному.

Пока там до «кирпича» дело дойдёт… шансов выжить в реальной жизни мало, в общем.

Ответить

AdDa
21 янв 2011 в 02:50

Ну надеюсь поход в туалет проходит без таких долгих проверок? :)

Ответить

vbarhatov
21 янв 2011 в 00:05

Программист, ложась спать, ставит рядом два стакана: первый — полный, с водой; второй — пустой.
Первый на случай, если захочет пить.
Второй, если не захочет.

PS: Переходя дорогу, всегда смотрю в обе стороны

Ответить

НЛО прилетело и опубликовало эту надпись здесь

adm0r
21 янв 2011 в 12:03

А нельзя не смотреть. Зная наших людей ни в чем нельзя быть уверенным.

Ответить

pnd_ku
21 янв 2011 в 14:38

наплодили сущностей — макконнелл бы не оценил :) пустой стакан появица при инициализации, а если программист пить захочет — он его наполнит…

ЗЫ: тоже смотрю в обе стороны

Ответить

vbarhatov
21 янв 2011 в 23:38

Макконнелла никто не отменял! Святое. Тут ведь не в сущностях дело. Нужно учесть всевозможные варианты событий.

Анектод стар как мир, но одно и тоже с дорогой только другими словами.

Ответить

Drakula2k
21 янв 2011 в 00:13

По-моему, в России почти все так делают =)

Ответить

equand
21 янв 2011 в 01:38

Отсюда такой процент конвертации граждан в программисты :)

Ответить

JYE
21 янв 2011 в 08:47

В китае с переходом дороги все намного хуже чем в России

Ответить

ImKremen
21 янв 2011 в 13:58

Закономерность? www.youtube.com/watch?v=RjrEQaG5jPM

Ответить

cbone
21 янв 2011 в 18:29

Как тараканы

Ответить

НЛО прилетело и опубликовало эту надпись здесь

dyakov
21 янв 2011 в 10:16

Некоторые, даже переходя трамвайные пути смотрят не едет ли трамвай по встречке…

Ответить

butteff
21 янв 2011 в 06:45

Ну, еще тогда анекдот из той же серии, раз про стаканы с водой уже написали ниже.
Салат рекурсивный: огурцы, помедоры, салат!
Теперь это анекдотпост!

Ответить

Orrim
21 янв 2011 в 10:15

Ну тогда уж салат Фибоначчи: смешиваем остатки вчерашнего салата с позавчерашним.

Ответить

НЛО прилетело и опубликовало эту надпись здесь

NixGuy
21 янв 2011 в 12:35

Почти всегда так делаю, но это уже привычка.

Ответить

YaQ
21 янв 2011 в 16:44

в нашей стране иначе и нельзя, а то под колёсами окажешься особо умных, которые авриечку включают и задним ходом…

Ответить

dekoo
20 янв 2011 в 23:56

В тексте у вас «шаблона MVS», поправьте пожалуйста на MVC — (Model View Controller)

Ответить

andreycha
20 янв 2011 в 23:57

Спасибо, поправил. Не думая переписал с оригинала.

Ответить

andreycha
21 янв 2011 в 00:47

Кстати, в Твиттере пишут, что:

> У вас в статье все верно было с MVS — Model View Separation. MVC — частный случай.

Ответить

dekoo
21 янв 2011 в 00:51

Если это имелось ввиду, извиняйте.

Всеже боюсь что в таком случае у большинства возникнут вопросы, хотя могу ошибаться. Как говорится, по себе людей не судят =) Предлагаю в таком случае сделать расшифровку и может быть даже линк на описание паттерна.

Ответить

dekoo
21 янв 2011 в 00:00

Пункт 1-ый можно еще трактовать как «не бойтесь изобратеть велосипеды» =) Изобретение очередного велосипеда как раз таки помогает пощупать и понять основы.

Ответить

SadRunner
21 янв 2011 в 00:17

- Получить проблему
- Найти существующее решение
- Построить свой велосипед.
- Сравнить с существующим решением
- Понять почему реализовано именно так так
- Составить список недочетов в своем велосипеде, и исправить их
- Повторить...

Так они учатся.

Но для коммерческого программирования данный алгоритм работы не проходит.

Следствие: при коммерческом программировании имеет смысл строго разделять работу и учебу или выбрать другой алгоритм обучения.

Ответить

dekoo
21 янв 2011 в 00:30

Ни то чтобы не подходит, нужно просто знать меру.

И конечно понимать, на чьей же стороне вы находитесь. Либо вы менеджер и тогда предложение потратить 2 дня на фунционал, который можно взять готовым в виде сторонней библиотеки, покажется вам кощунственной тратой бюджета. Либо вы программер и трезво оцениваете влияние вашего велосипедо-строения на сроки проекта, и по-мимо выгод для компании ищите выгоды для себя, собственного развития, собственной стоимости на рынке труда.

Ответить

SadRunner
21 янв 2011 в 00:39

Хм… На чьей стороне…
На стороне реализуемого проекта конечно. Вне зависимости от занимаемой позиции. И как менеджер и как проектировщик и как кодировщик (ох давно было это веселое времечко) я всегда должен отдавать себе отчет, что я делаю для проекта, а что для себя.

Учусь я, скорее, для себя, решение клепаю для проекта. Эти вещи нужно очень четко разделять.
Иногда же народ путает обучение методом сравнительного велосипедостроения и реализацию функционала для проекта.

Объяснять, показывать, не действует — наказывать.

С уважением к вашему мнению.

Ответить

SMiX
21 янв 2011 в 00:18

Так было раньше. Теперь познавать основы на практике проще всего, участвуя в opensource-разработках.
Сейчас для этого есть очень удобные сайты и инструменты(github.com, bitbucket.org).
При этом вы не только изучаете, что и как устроено, но и учитесь на чужом коде.

Ответить

dekoo
21 янв 2011 в 00:35

Участие в OpenSource проектах — это прекрасно. Но это, как выше писалось, личное время программиста. Я считаю, что иногда не грех, работая на компанию, использовать возможности для собственного профессионального роста, в том числе и через игнорирование готовых решений и написания собственного кода… **в разумных пределах, без фанатизма**.

Ответить

SadRunner
21 янв 2011 в 00:47

ИМХО это зависит от того, сможет ли компания использовать результаты вашего обучения в дальнейшем. Термин «иногда» подвержен очень сильному размыванию, к сожалению. Безусловно, технари учатся и во время реализации поставленных задач. И ПМ, и архитекты, и кодеры, все. Но нужно отдавать себе отчет: «Сейчас я учусь, а вот сейчас я работаю.»

В то же время любая адекватная прогерская контора (наверное так должно быть) выделяет время на самостоятельные исследования, типа академических дней на старших курсах и в аспирантуре в советское время.

Не верное, на мой взгляд использовать контору для своего обучения, забывая что цель конторы — прибыль.
Не верно, на мой же взгляд, заставлять разработчиков пахать, как Стаханов в шахте, не давая возможности оглядеться по сторонам и не предоставляя времени на учебу

Ответить

LionAlex
21 янв 2011 в 09:47

это да, но все же мысль верная — opensource проекты дают опыт в понимании чужого кода, ответственности за свой код(!), и, конечно, просто повышают скиллы. И что немаловажно — дают удовлетворение от своей работы, если твой код попал в свежий релиз
// Пошел искать какой-нить opensource-проект

Ответить

tampere
21 янв 2011 в 15:50

> Участие в OpenSource проектах — это прекрасно. Но это, как выше писалось, личное время программиста.
Это потому, что вы пишете чисто коммерческий код. Опенсорс очень часто допиливают под свои задачи, а потом выкладывают, или просто пишут, а потом выкладывают.

Ответить

Disturbed
21 янв 2011 в 00:18

По поводу KISS есть хорошая цитата от Брайана Кернигана:
«Debugging is twice as hard as writing the code in the first place. Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it. „

Ответить

Ky7m
21 янв 2011 в 00:20

Спасибо за цитаты. На мой взгляд, они должны стать в коллективе правилами!

Ответить

Maur
21 янв 2011 в 00:21

Начал автор правильно, а в конце, как мне кажется «сдал» — создалось впечатление выдумывал с головы лишь бы дотянуть количество правил до десятки. Я бы немного видоизменил и переформулировал некоторые правила. Например, изменил бы полностью 9 пункт:
9. Никогда не доверяйте документации и тем более не полагайтесь на нее. Этому правилу меня научил мой зав. кафедры xx лет назад. За мою xx летнюю карьеру это правило работало всегда и везде от мелких проектов до крупных, причем даже там, где казалось бы всё должно быть совершенно «по плану». Нифига.

Кроме того я бы изменил 10 пункт, просто потому что он совершенно не работает. Нет, не смотреть чужой код совсем — нельзя, но сформулировать можно и так:
10. Чужой код может ПЛОХОМУ научить. Самое худшее что может к вам привязаться, это чужая плохая привычка. Что я имею ввиду, любой проект всегда имеет какие-то рамки, вы не знаете, что конкретно сподвигло человека писать это и именно это. Обычно люди не пишут в комментариях — «тут ко мне подошел шеф и заставил сократил срок сдачи до завтрашнего утра, поэтому, ув. читатель моего кода, с этого момента качество моего кода немного упадет, так что учтите этот фактор при дальнейшем прочтении». Такое не пишут. По моему опыту, чем опытнее программист, тем больше он ИСПРАВЛЯЕТ в чужом коде, чем его ЧИТАЕТ. Поэтому такое правило, ну вы знаете, на ваш страх и риск.

И последнее. 7 пункт, я бы тоже изменил примерно вот так:
7. Не пугайтесь, но и в тоже время ИЗБЕГАЙТЕ быстрой смены технологий. Потому что с годами понимаешь, это ни к чему не приводит.

В целом с иными пунктами я целиком и полностью согласен с автором.

Ответить

twdragon
21 янв 2011 в 00:34

Недоверие документации? А как тогда вообще сопровождать чужой большой проект? Я согласен, документация всегда требует беспристрастного и не догматического подхода, но обходиться без нее по принципу тотального недоверия? По мне — такое вряд ли разумно, тем более в разветвленном проекте.

Чужой код всегда поддается анализу, если только мыслить логически.

По поводу пункта 7 соглашусь: экспериментально подтвердил необходимость на шаг отставать от развития технологий с целью не участвовать во всеобщей (не всегда успешной) обкатке нового решения.

Ответить

Maur
21 янв 2011 в 01:01

Документация проекта и сопровождение проекта, — это две совершенно не согласуемые вещи. Документация пишется почти всегда (кто бы и чтобы не говорил) в таком аврале и в таком урезаном времени, и она может быть:

а)не актуализирована — такое случается наиболее часто в оч. больших проектах(пример то что я видел своими глазами — документация трейдинга крупнейших бирж мира, нарочно пишут ахинею, как было лет 10 назад) — верить такой документации? У нас одна поверила, уволили, чтоб «глаза разувала чаще».

б)сделана не профессионально. Технические спецы вообще не умеют писать документы и это совсем не преувеличение. 1 на 100 умеет рисовать, 1 на тысячу умеет вставлять таблицы там где нужно структурировать, инфографику вы не найдёте вообще в документации, сделаной программистами. Я уже не говорю про простенькую типографику,- я этих «док», «вик», «спек» насмотрелся раз в 100 больше чем этот «j2ee индус с 8 годами» непонятно где.

в)не согласована. Вот бывает такое есть два документа от двух отделов на одно и то же почти. Один на вики, второй авторизованый pdf. В двух источниках — взаимоисключающая информация. Кому верить?

Какой выход? Думать всегда головой своей. Разбираться в проекте глубоко, ТЩАТЕЛЬНО взвешивать любую входную информацию, АНАЛИЗИРОВАТЬ ее. Потому что люди, которые полагаются на документацию — только потому что «я забрал ее из cvs/svn а дальше меня ничего не интересует», прикрывают сугубо свою собственную задницу(это мое имхо), и по моему опыту работу, им глубоко начхать на проект, для них главное — не взять на себя ответственность.

Ответить

zoxa
21 янв 2011 в 01:19

Пример относительно проблем с «актуальностью» документацией:
Не так давно (осень 2010) мы внедряли IBM WebSphere MQ на CentOS + PHP. Вроде бы всё сделали по документации, вроде всё настроили по инструкции, а вот на обещанные мощности так и не смогли выйти. Начали общаться и грузить тех поддержку. Через два часа конференции у их (IBM) программиста появляется вопрос, **«А почему вы используете listener которые мы не поддерживаем с 2004 года?»**. Наш ответ: «Так написано в документации по настройках MQ на Linux?» Программист: «Упс… мы забыли обновить

инструкцию для Linux, воспользуйтесь инструкцией для Windows Server или Solaris?»

Вывод: Документация по установки IBM WebSphere MQ на Linux в супер «актуальном» состоянии. Последние изменения были где-то в 2003-2004. Дело было осенью 2010го

Ответить

ntkt
21 янв 2011 в 05:11

Внезапно вспомнил, что встречался с совершенно аналогичными вещами, и как раз в документации IBM по MQ!

Фраза, которая почему-то врезалась мне в память:
www.google.ru/search?&q=%22can+someone+check+the+wording+of+this+step%22
Мне кажется, круче было бы только оставить в документе Lorem Ipsum.

Чего уж говорить о прочих прелестях громадных проектов для тех, кто только пользуется их плодами. Например, о **логах** внутри дистрибутивов одного программного продукта от HP, а в логах чудо-даты, учетная запись с индийским именем, и имя машины, видимо, на которой этот дистрибутив и собирали.

Ответить

toxicdream
21 янв 2011 в 07:12

Уу, я думал, что только мне так не повезло с IBM WebSphere MQ — месяц беспрерывного дебагинга с гуглом в обнимку — доки и примеры очень скудные.
Сделать вывод — если кто-то что-то там понял и реализовал, то его реализация продается как очень дорогое готовое решение. Маркетинг чистой воды, заставляющий программистов проталкивать IBM WebSphere MQ с целью получения выгоды!

Ответить

zoxa
21 янв 2011 в 17:24

Вам еще повезло. Я PHP к нему прикручивал, через SAM модуль. В SAMConnect всего 8 функций: connect, disconnect, send, receive, receiveAll, peak, peakAll, isDebug. И как вот с этим работать? Никакой диагностики, документации и примеров. Всё писалось в слепую.
Итог: 2 месяца разработки и внедрения, 4 месяца QA и дебагинга.

Ответить

zoxa
21 янв 2011 в 17:20

А логи от WebShere как будто очень понятные. Эта вещь плодить у меня в home директории бесполезные лог файлы каждую минут. Как отключить так и не нашел :(

Ответить

sa_melnikov
21 янв 2011 в 00:43

1. Насчёт документации: документирование всегда + в помощь к качественному коду. Имхо: преподаватели глупости говорят иногда… зачастую даже не специалисты в своей области.

2. Чужой код может научить как хорошему так и плохому, взависимости от индивидуальной восприимчивости ;). Да, и я пишу в комментах если меня «вынудили» что-то сделать… уже были прицинденты благодарности за описание написанной проблемы. — да это может плохо воспринимается заказчиком, но пока никто нежаловался.

Ответить

smind
21 янв 2011 в 01:59

хороший код сразу видно…

Ответить

grigoryvp
21 янв 2011 в 11:48

Соглашусь по поводу десятого пункта. Довольно долго у меня была мечта — посмотреть на действительно хороший код на C++. мечта исполнилась много лет назад, когда я посмотрел в исходники Qt. А все что я смотрел до этого делилось на две группы — ужас и тихий ужас. К чему я это? К тому, что очень сложно найти ХОРОШИЙ код, посмотрев на который можно бы было чему-то научиться.

Ответить

Johan
22 янв 2011 в 17:26

На тему 7-ого пункта есть очередной бородатый анекдот.

Чем отличается программист с опытом работы 1 год от программиста с опытом работы 10 лет?
Программист с опытом работы 1 год: «Нет такой технологии, которую я не смогу изучить».
Программист с опытом работы 10 лет: «Нет такой технологии, без которой я не смогу обойтись».

Ответить

Unlogic
21 янв 2011 в 00:33

11. Пользуйтесь солнцезащитными кремами. Oh, come on.

Ответить

shmaltorhbooks
21 янв 2011 в 10:42

12. Не паникуй!

Ответить

IngvarrT
21 янв 2011 в 11:21

0. Полотенце!

Ответить

Rubs
21 янв 2011 в 00:35

Хороший программист ИМХО должен быть талантлив не только в кодировании/программировании. Нужно множество навыков, чтобы уметь работать в коллективе, понимать важность того или иного решения, уметь анализировать и видеть далеко и глубоко. В общем, самая творческая профессия…

Ответить

andreycha
21 янв 2011 в 00:38

Да и вообще — users.livejournal.com/akme_/154221.html

Ответить

drzerg
21 янв 2011 в 00:37

Теперь количество хороших программистов значительно увеличится. Возьму на заметку!

Ответить

Zorkus
21 янв 2011 в 00:39

...12. И презервативами, for additional safety.

Ответить

SadRunner
21 янв 2011 в 00:50

Не хотите ли вы сказать, что такие, как мы, не должны размножаться? :))

Ответить

Cord
21 янв 2011 в 01:18

из опыта.

1. адепты самодокументирующегося БЕЗ КОММЕНТАРИЕВ кода идут нахуй. я обладаю памятью помнить слова на страницах, которые посещал лет пять назад, и так нахожу их в гугле. так и в проекте, где несколько тысяч файлов, коммент со словами в начале файла, что он делает, позволяет найти за долю секунды поисковиком, в то время как в коде без комментов вы потратите явно больше.

2. не все подходы работают одинаково. я вот, например, с точки зрения выпускника ВМК — хрен с горы. так как по образованию инженер-робототехник. мне не читали про конечные автоматы и многое другое. с другой стороны, потом читал сам.

так вот, за счет минимализма теоретических знаний, я всегда учил других, как репетитор, и сам работаю, на основе интуиции. наш мозг — великая штука. я беру задачу, и делаю ее САМ. на бумаге. по ходу дела вырабатываю алгоритм, анализирую, как я ее решал бы.
и так получаются сложнейшие бинарные поиски и так далее.

постепенно, за счет того, что я читаю много книг, алгоритмы оседают в подсознанию, и затем всплывают, комбинируясь. но да, пипец — я программист, к которому алгоритмы приходят, как к поэту строки — за счет обращения к Силе.

3. сильные стороны — это так. у нас у всех набор рэндомных качеств, по одним мы лучше других, по другим — хуже. поэтому нет лучших и худший, если сами термины не определены однозначны. грубо говоря, математически задача «быть лучшим» не имеет смысла, но мозг этого не осознает, за счет общественных догм и предрассудков.

мое имхо — рулят итерации
— познал себя глубже
— прокачал достоинства, убрал/сгладил недостатки
— goto 1

4. программисты — одна из самых великих профессий на земле. мы и творческие люди, и ремесленники. в нашем мозгу умещаются сведения о работе компьютеров, бизнес-логики, других людей, технологиях и так далее. мы моделируем в сознании десятки объектов, в подсознании — тысячи. разбираемся в сотнях тесно связанных областей и без проблем осваиваем новое. создаем то, что облегчает жизнь другим людям, при этом имея свободу выражения как в плане реализации, так и в плане инструмента. всегда нужны, всегда востребованы, очень много путей для развития.

за нами — будущее.

Ответить

equand
21 янв 2011 в 01:42

1. Как-то написал просто шелл на шелле для работы с почтовиком, решил вместо документации и USAGE написать небольшую историю со стихами, знаете лучше пошло :D. Хотя да, в крупных

проектах не писать доки и комменты кощунство

Ответить

zoxa
21 янв 2011 в 17:28

просто стихи были или по теме?
Я в одной JavaScript библиотеке видел комментарии в стихах с небольшими лирическими отступлениями.
Так ею увлёкся, что все комментарии в файле прочитал

Ответить

equand
25 янв 2011 в 01:36

Именно по теме :)

Ответить

ahiskali
21 янв 2011 в 02:06

Кто теперь скажет что хабр не торт, пусть укусит Cord'a за блестящий металлический зад.

Ответить

Vile55
21 янв 2011 в 11:02

1. Комментарии в коде должны описывать *зачем* что-то делается, а не *что именно* делается. Что делается должно быть понятно из кода (самодокументированного).

Ответить

RomeoUA
21 янв 2011 в 15:05

я аж прослезился с четвертого пункта!
нет, правда, без иронии, красиво закрутил :)

Ответить

AmdY
21 янв 2011 в 01:33

Keep it simple and short — это ужасный вариант. Простота кода в большинстве случаев гарантируется заложенной избыточностью.
Хорошее правило — не экономьте на символах.

Ответить

smind
21 янв 2011 в 01:48

вообще то я видел другой, более распространенный вариант — keep it simple, stupid… имхо оно уместнее…

Ответить

Wildy
21 янв 2011 в 08:36

… замечательное Code Less Create More

Ответить

AmdY
21 янв 2011 в 10:18

угу, эта подпись у меня красуется уже много лет в подписи на phpclub и является кредом по жизни.

блин, так обидно, когда здесь имбицилы втихую минусуют посты, вымученные прочтением кучи книг по программированию и подтверждённые годами опыта.

Ответить

Unlogic
21 янв 2011 в 11:13

Есть такая характеристика кода — «concise code». Это значит «давать много информации малым набором слов, быть кратким, но полным». Это совсем не значит писать магические однострочники, это значит лишать код лишних сущностей, которые не отвлекают от основной цели программы.
Scala по сравнению с Java — яркий пример увеличения «conciseness». При практически одинаковом подходе к программированию (в Scala все еще доминирует императивный ООП) в Scala-коде значительно меньше «шума».

Ответить

AmdY
21 янв 2011 в 11:34

к чему сравнивать синтаксиса языков, при разработке это третьестепенная вещь. я, например, ненавижу синтаксис php, но пишу на нём, потому что он позволяет быстро и эффективно решать задачи.

посмотри как ты дал определение, а затем добавил избыточную информацию, чтобы её пояснить. вот и в программировании так же, если код недостаточно выразительно указывает что он делает, то нужно его отрефакторить добавив избыточности, выделить блоки в подметоды с выразительным названием и т.д.

Ответить

Unlogic
21 янв 2011 в 11:44

Я сравниваю не синтаксис, а семантику. Синтаксис в понимании кода играет небольшую роль, хотя если он не однозначный и не гомогенный, то это может значительно усложнить понимание.
>> выделить блоки в подметоды с выразительным названием и т.д.
То, что ты описываешь — не избыточность, а наоборот абсолютно правильный подход. Избыточность — это когда ты используешь цикл for с переменной для того чтобы пройтись по коллекции (то есть, вводишь лишние сущности: 1) эту самую переменную 2) вызов метод взятия элемента из коллекции по индексу). Или использовать одноразовый анонимный класс, чтобы передать куда-нибудь функцию как аргумент. И так далее.

Ответить

viz_dragon
21 янв 2011 в 21:43

Не согласен! Хороший программер 5 часов — думает, 5 мин — пишет… В результате КИСС и получается.

А если не экономить на символах — то получите просто задержки в обработке. Ведь Вам самому(… как и компу...) — проще прочитать одну умную и понятную крылатую фразу, чем читать листов на тему размышлений о ней ;)

Ответить

AmdY
21 янв 2011 в 23:33

вот потому я и возмущался, чтобы никто такого бреда не думал.
учитесь по сомнительным статьям, вместо того, чтобы прочесть полноценную книгу.

Ответить

smind
21 янв 2011 в 01:56

учу основы, задаю вопросы, пишу простой и понятный код… один хрен программирование проносит ежедневную боль… от осознания что кучи вещей (даже простых) как бы не старался а так никогда и е осознаешь… смотрю на свой код написанный неделю, месяц, год назад и удивляюсь как хватило сил такое написать, при том что это самый примитив… а ведь люди как то ядро ведь пишут…

Ответить

in3gant
21 янв 2011 в 10:33

цитата: «Чем больше я учусь, тем больше я понимаю, что ничего не понимаю...»
Если была цель достать звезду и проанализировав сделаное, можно прийти к выводу что здезды вы не достали, но луна у вас точно в кармане…

Ответить

in3gant
21 янв 2011 в 10:59

каюсь. вместо «проанализировав» должно быть «проанализировать»

Ответить

znvPredatoR
21 янв 2011 в 02:13

Насчет пункта 8 про костыли я бы поспорил. Это предложение: «всегда говори правду, но иногда можешь соврать» я бы изменил на такое: «всегда говори правду, но когда понимаешь, что лучше соврать — ври», звучит, наверное, немного цинично, но в разработке коммерческих приложений полностью себя оправдывает. Я не раз видел как поклонник метода «лучше сделать правильно и расширяемо» — проваливал сроки. Иногда, костыли, это нужная и полезная вещь. Лучше написать код который будет где-то и кривой и выпустить приложение, чем провалить сроки и упустить момент.

Ответить

pilgr
21 янв 2011 в 11:02

Костыль — как растрата дорогого и долго накапливаемого депозита. Костыльнул в коде, получил мгновенный профит. Не костыльнул, приумножил депозит на архитектуру, проинвестировал в будущее проекта.

Но ведь бывает и нужно получить мгновенный профит, чтобы приумножить *реальный* депозит.

Ответить

TravisBickle
21 янв 2011 в 10:38

А почему не дали ссылку на оригинал?

Ответить

Disturbed
21 янв 2011 в 10:43

В топике-переводе всегда есть ссылка на оригинал, внизу, слева от ника автора этого поста. Ваш К.О. :)

Ответить

grep0
21 янв 2011 в 11:19

Workarounds это чаще всего немного о другом — об обходе багов, недоработок и нелогичностей используемых фреймворков. Например, трудно писать на джаваскрипте, не используя костыли для обхода peculiarities зоопарка браузеров.
Обычно руководствуюсь простым правилом — делать только такие костыли, которые можно без проблем убрать, когда нижележащий фреймворк приведут в порядок.

Ответить

osintsef
21 янв 2011 в 12:21

Я не программист, поэтому скажу мнение со стороны работодателя. для его самое важное качество программиста — это его умение доводить начатое до релиза в поставленные сроки. Даже если где-то код будет кривой это в дальнейшем можно поправить…

Ответить

Joshua5
21 янв 2011 в 13:04

обратная сторона медали в том, что иногда сроки ставятся такие, что без костылей не обойтись.

Ответить

CGS
21 янв 2011 в 14:46

Может я конечно и сильно ошибаюсь, но вот именно из-за таких мнений, увеличивается процент «быдлокодеров» (не нравится мне это слово, но так понятней).

Ответить

Nordvind
21 янв 2011 в 17:07

По-моему, ситуация когда нужно «скорее быстро, чем качественно» должна быть исключением, а не правилом.

Ответить

olegafx
21 янв 2011 в 16:18

>Хороший программист — это тот, кто смотрит в обе стороны, переходя дорогу с односторонним движением.

Главное, чтоб не вперед и назад

Ответить

kostyl
21 янв 2011 в 17:42

я долго не работаю

Ответить

VBart
22 янв 2011 в 00:24

Я бы озаглавил, не 10 способов стать хорошим программистом, а минимум 10 качеств необходимых для того, чтобы стать хорошим программистом.

Ответить

Зарегистрируйтесь на Хабре, чтобы оставить комментарий

Публикации

ЛУЧШИЕ ЗА СУТКИ ПОХОЖИЕ

slava_rumin
17 часов назад

Вспомнил школьную физику, и теперь продаю горячий воздух на Авито на 12 млн в год

Простой 10 мин 33K

Интервью

+57 37 40

artyomsoft
23 часа назад

Как я пришёл к пониманию основ создания Live-дистрибутивов Linux, решив починить свой старый SSD

Средний 28 мин 9.5K

Туториал

+51 88 7

andres_kovalev
20 часов назад

Особенности Effector, которые почему-то никто не обсуждает: опыт ВКонтакте спустя год использования

Средний 31 мин 4.5K

Мнение

+41 36 17

HotReset
22 часа назад

Чем убили Фидо

2 мин 7K

Дайджест

+38 9 80

ru_vds
19 часов назад

Как новичку поучаствовать в устранении багов Google Chrome

Средний 11 мин 1.7K

Кейс Перевод

+32 24 0

sobolevn
16 часов назад

Проблемы вызова Python кода из C кода

Сложный 5 мин 2.1K

Ретроспектива

+26 27 0

dolovar
16 часов назад

Что не так со статьями о выгорании

Средний 26 мин 3.6K

Мнение

+20 57 32

DimDimDimDimDim
21 час назад

Почему нейросети становятся угрозой для природы и что с этим сделать

8 мин 3K

+19 15 17

Vasjen
вчера в 11:05

Деплой .NET приложений для самых маленьких. Часть 2. Github Actions

Простой 8 мин 1.7K

Туториал

+18 39 1

Pavel_Kanaev
5 часов назад

Шифры «Тысячи и одной ночи»: криптография древних арабов

Простой 11 мин 1.9K

Обзор

+16 26 2

Показать еще

КУРСЫ

Linux уровень 4. Диагностика и устранение неполадок в Linux (Линукс)
23 сентября 2024 · Учебный центр «Микротест»

Магистратура Кибербезопасность
1 октября 2024 · Нетология

OutBlock

1 октября 2024 · XYZ School

Разработка серверной части приложений PostgreSQL 12. Расширенный курс

9 декабря 2024 · Учебный центр «Микротест»

Автотестировщик

10 сентября 2024 · Открытые школы Т1

Больше курсов на Хабр Карьере

ЧИТАЮТ СЕЙЧАС

Вспомнил школьную физику, и теперь продаю горячий воздух на Авито на 12 млн в год

33K 41

В НАСА выяснили причину повторяющегося «пульсирующего» звука из динамика Starliner на МКС, который уже прекратился

39K 27

Как схема 500-30-5 делает учебные центры IT-компаний бесполезными

36K 59

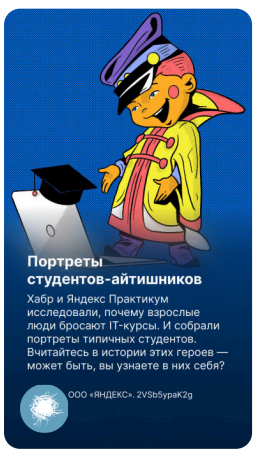
Что там с Дуровым и почему он кот, а также тайна самой прибыльной компании в России

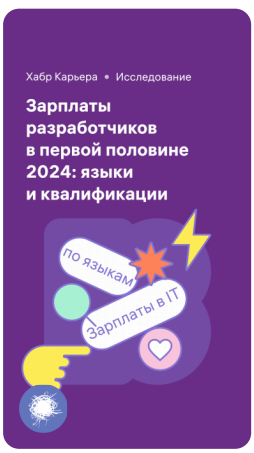
44K 42

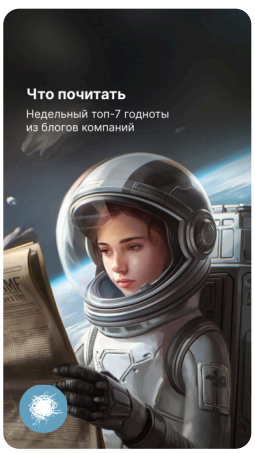
Фокус в Android TV

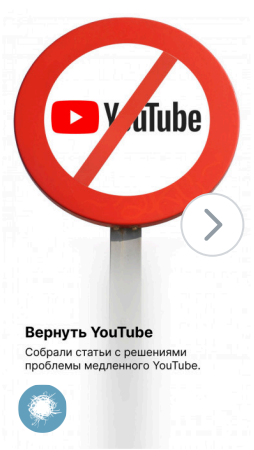
13K 2

ИСТОРИИ









Выгорающие IT-студенты: какие они?

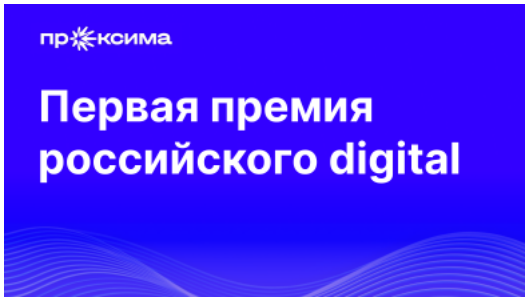
Актуальные зарплаты разработчиков

Больше 1К вакансий с удаленкой

Годнота из блогов компаний

Вернуть YouTube

БЛИЖАЙШИЕ СОБЫТИЯ

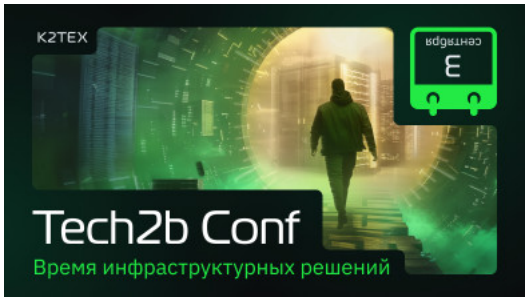


27 августа – 7 октября

Премия digital-кейсов «Проксима»

Москва • Онлайн

Маркетинг Другое



3 сентября

Tech2b Conf: время инфраструктурных решений

Москва

Администрирование

Менеджмент

11 сентября

Митап по BigD Честного ЗНА

Санкт-Петербург •

Разработка

Больше событий в к

Больше событий в календаре

Техническая поддержка

© 2006–2024, Habr