

Александр Никитин, Дмитрий Ляпин

ПУТЬ ПРОГРАММИСТА



**ОТ 100\$
ДО 10 000\$ В МЕСЯЦ**

пошаговая инструкция развития
шокирующие цифры статистики тест твоей компетентности
методика всестороннего развития
почему так тяжело учиться как подготовиться к собеседованию
фишки, полезные любому технарю
самый простой способ сделать карьеру
секреты развития профессиональных навыков

<http://prog-school.ru>

2010 © Школа Программирования

ВВЕДЕНИЕ	3
ЧАСТЬ 1. СТАТИСТИКА	4
ГЛАВА 1. ШОКИРУЮЩИЕ ДАННЫЕ	4
ГЛАВА 2. ГРАБЛИ, КОТОРЫЕ ОБЯЗАТЕЛЬНО ТРЕСНУТ НАС ПО ЛБУ ...	7
ЧАСТЬ 2. АНАЛИТИКА.....	9
ГЛАВА 3. СМОТРИМ В КНИГУ, ВИДИМ ФИГУ	9
ПОЧЕМУ НЕ РАБОТАЮТ КНИГИ	10
ПОЧЕМУ НЕ РАБОТАЕТ ВЫСШЕЕ ОБРАЗОВАНИЕ	13
ПОЧЕМУ РАБОТА И КУРСЫ РАБОТАЮТ... НО РЕДКО	16
ГЛАВА 4. ПОРА ВЗГЛЯНУТЬ НА СЕБЯ БЕЗ ПРИКРАС.....	19
КРИТЕРИИ ОЦЕНКИ ПРОГРАММИСТА	19
ГЛАВА 5. НАЧИНАЕМ ВЗЛЕТАТЬ. КОНЦЕПЦИЯ РАЗВИТИЯ ПРОГРАММИСТА.....	24
ЧАСТЬ 3. ПРАКТИКА.....	26
ГЛАВА 6. МОЩНЫЙ РЫВОК ВПЕРЕД. РАЗВИТИЕ ПРОФЕССИОНАЛЬНЫХ НАВЫКОВ	26
РЕГУЛЯРНАЯ ПРАКТИКА ПРОГРАММИРОВАНИЯ	27
БЫТЬ В АВАНГАРДЕ.....	27
КНИГИ MUST READ.....	28
УМЕНИЕ ИСКАТЬ, ОТФИЛЬТРОВЫВАТЬ И ИСПОЛЬЗОВАТЬ ИНФОРМАЦИЮ	29
ПОШАГОВАЯ ИНСТРУКЦИЯ РАЗВИТИЯ.....	30
<i>Шаг 1. Новичок. Опыт работы = 0. Знания в рамках школьных и институтских курсов информатики.....</i>	<i>32</i>
<i>Шаг 2. Advanced Новичок. Опыт работы <= 0,5 года. Знания в рамках школьных и институтских курсов информатики + полученные на работе навыки решения конкретных задач.</i>	<i>33</i>
<i>Шаг 3. Младший программист. Опыт работы 0,5-1 год.</i>	<i>35</i>
<i>Шаг 4. Программист. Опыт работы 1-3 года.....</i>	<i>36</i>
<i>Шаг 5. Старший программист. Опыт работы от 2 лет.....</i>	<i>37</i>
<i>Шаг 6. Функциональный архитектор. Опыт работы от 3 лет.</i>	<i>38</i>
<i>Шаг 7. Менеджер проектов. Опыт работы от 5 лет.....</i>	<i>39</i>

ГЛАВА 7. УСКОРЕНИЕ ДО МАКСИМУМА. РАЗВИТИЕ СОПРЯЖЕННЫХ НАВЫКОВ	41
Английский язык	41
Учитесь читать наискосок	44
Подготовка к собеседованию	44
ГЛАВА 8. В ПОЛУШАГЕ ОТ ИДЕАЛА. РАЗВИТИЕ ОБЩИХ НАВЫКОВ .	48
Физическое и эмоциональное состояние	49
Правильный отдых	50
Планирование	51
Совершенствование полезных навыков.....	52
Мотивация	53
Не теряйте надежду и верьте в успех.....	54
ЧАСТЬ 4. ИТОГИ	56
ГЛАВА 9. ОТ НОВИЧКА ДО ГУРУ. ДЕРЕВО РАЗВИТИЯ ТЕХНАРЯ	56
ГЛАВА 10. ЗАКЛЮЧЕНИЕ	60
ПРИЛОЖЕНИЕ А. ЧЕМ ШКОЛА ПРОГРАММИРОВАНИЯ МОЖЕТ БЫТЬ ПОЛЕЗНА ЛИЧНО ВАМ	62
Методы обучения в ШП	63
ПРИЛОЖЕНИЕ Б. ПРОДУКТЫ И ОБРАТНАЯ СВЯЗЬ	65
<i>Контакты</i>	66

Введение

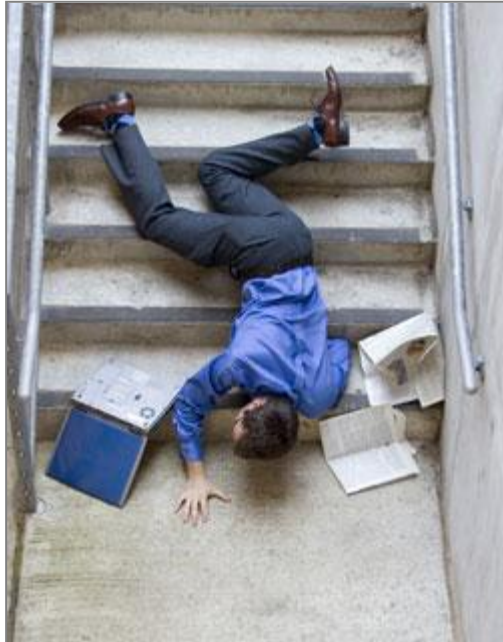
Если Вы читаете эти строки, то, скорее всего, Вас заинтересовала тема данной книги. Лично нас она волнует уже давно. Кстати кто мы такие?

В недавнем прошлом мы – выпускники МГТУ им. Баумана, кафедры программного обеспечения вычислительной техники. Ежедневно практикуясь в программировании, мы накопили богатый 10-летний опыт разработки ПО и веб-проектов, изучили множество тонкостей программирования, от маленьких практических секретов, до методологий построения систем, масштабов предприятия. Подробнее о нас Вы сможете узнать по адресу <http://prog-school.ru/2010/01/ob-avtorax/>.

В этой книге мы собрали большое количество советов (включая уникальную **пошаговую инструкцию развития!**) всем тем, кто, будучи программистом, хочет добиться успеха в **карьере**, кто задумывается о своем **профессиональном росте** и хочет сделать этот процесс **контролируемым**. Тем, кто понимает, что профессионал всегда имеет огромное количество привилегий (в том числе финансовых) перед середнячком, наверняка будут интересны собранные здесь знания, которые мы неоднократно проверили на практике, и которые помогли нам добиться успеха в собственных проектах. Итак, всем, кто ищет кратчайший путь от среднестатистического программиста до гуру информационных технологий (с внушительной репутацией, солидным доходом, собственными проектами), желаем приятного чтения!

Часть 1. СТАТИСТИКА

Глава 1. Шокирующие данные



Мы долго думали с чего начать и решили Вас поразить. Поэтому сразу же приведем статистику, которая не может оставить нас равнодушными. Мы искренне желаем Вам **задуматься** над этими цифрами.

Из **100** начинающих программистов:

- **98** не уделяют достаточного времени выбору инструментов разработки, что снижает её эффективность в 2-3 раза;

- **95** не задумываются о том, в каком направлении программирования они хотели бы совершенствоваться и развиваться;
- **80** не знают, в проектах какого рода они хотели бы участвовать через 3 года;
- **73** меняют своё место работы в течение первого года;
- **32** в течение 18 месяцев оставляют программирование окончательно и начинают заниматься другими вещами;
- Лишь **2 (двое!!)** добиваются впечатляющих результатов и в дальнейшем встают во главе ИТ компаний.

Ну что, не испугались таких цифр? Если посмотреть на картину с высоты статистических данных, то становится немного страшновато за судьбу программистов :)

На самом деле цифры столь плачевные только потому, что программистов новичков слишком много. И абсолютное большинство из них даже не задумывается о своем профессиональном росте. Плынут по течению и ждут, когда на них посыпятся деньги. Как можно догадаться, к таким успех не приходит **никогда**. Что же нужно делать, чтобы превратиться из человека без знаний в суперпрофессионала своего дела?

Как ни странно, ничего волшебного и секретного в успехе ИТшника нет. Вам даже не потребуется везение. Зато нужно нечто более важное – **упорство и дисциплина**.

Мы ещё не раз подробно остановимся на этих моментах. А пока, если Вы готовы проявить свою силу воли на пути к профессиональному успеху, то мы искреннее желаем чтобы у Вас всё получилось, и Вы дошли до самой вершины! Путь к успеху начинается прямо сейчас! И вот Вам первое задание...

Отложите книгу и дайте себе три **честных** ответа на следующие вопросы:

- Что я из себя представляю как программист?
- Чему я хочу научиться в программировании?
- Что я готов сделать для этого?

Запишите ответы на бумаге и сохраните их. Напишите прямо сейчас, продолжайте чтение только после этого!

Глава 2. Грабли, которые обязательно треснут нас по лбу



В главе 4 мы приведем список критериев, по которому можно оценить уровень программиста. Грамотные работодатели ориентируются на эти критерии при выборе среди соискателей. Но всё же это условные и субъективные показатели. Можно даже сказать, что не существует некой объективной шкалы, измеряющей уровень программиста. Однако можно предложить следующий количественный критерий, который на практике тяжело поддается вычислению:

Уровень программиста пропорционален количеству граблей, на которые ему приходилось наступать во время практики.

Как легко догадаться, наступание на грабли не связано с приятными эмоциями. Это надо помнить во время удара черенка о лоб. **Мы на полном серьезе!**

Любая неудача продвигает нас ближе к цели.

И если мы перестаем это помнить, то руки опускаются сами собой.

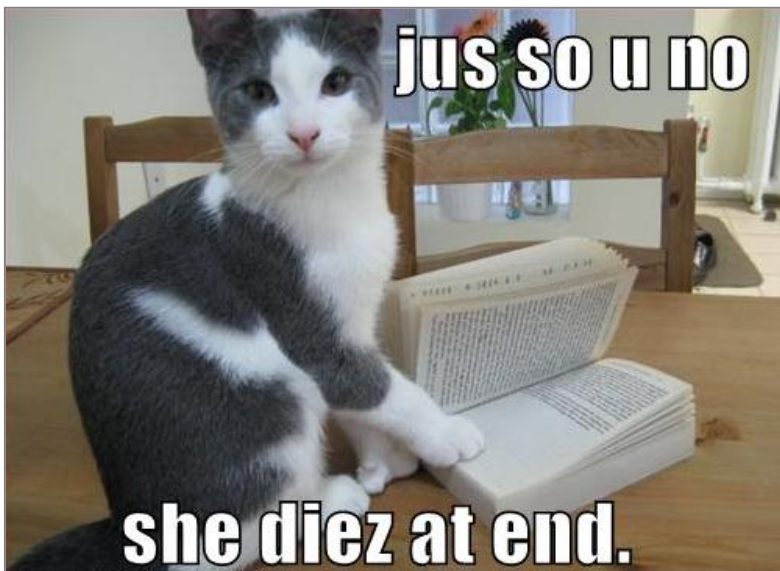
Кстати, ценный совет – описывайте те грабли, на которые Вам пришлось наступить. Таким образом, результаты будут зафиксированы, что в дальнейшем сыграет на Вас.

Глава 3. Смотри́м в книгу, ви́дим фигу́



К сожалению, многие так и не преодолевают этот барьер по различным обстоятельствам. И в большинстве своем проблемы обучения заключены не в человеке, а в **методах** обучения. К программированию это имеет самое непосредственное отношение. Потому что обучение программированию сопряжено с получением большого количества сложных технических знаний. Большие объемы такого рода информации не могут усвоиться быстро и легко. Чтобы знания уложились в четкую, структурированную систему необходима постоянная ПРАКТИКА. А теперь поговорим, почему же отдельные методы образования не всегда дают желаемый результат.

Почему не работают книги



Иногда, знаете, бывает такое чувство странное. Вроде бы взял в руки новую книгу, просмотрел оглавление и подумал: «о, про меня!», или «о, то что я так давно хочу узнать!». И, естественно, Вы эту книгу купили и даже начали читать. И как-то до конца даже и не прочитали:).. Ну потому что какие-то другие проблемы и интересы появились, чем-то другим занялись, да и вообще... И остается тогда непонятное чувство. Вроде бы в оглавлении-то не наврали, написали все, о чем заявляли. Но как-то суперменом после неё так и не стали. Знания получили и... всё, просто получили знания... а потом забыли.

Проблема здесь в том, что **простое чтение книг – не работает**. Это универсальное правило, но в программировании оно очень четко прослеживается. Здесь вообще отдельная история, попробуем объяснить почему.

Все книги по программированию можно использовать двумя способами:

- Как обучающий материал
- Как справочный материал

Сначала по второму пункту.

Самый быстрый способ получения справки по вопросам разработки ПО – это yandex или google.

И точка. Мы готовы спорить с кем угодно, что быстрее откроем поисковик и найдем интересующую нас информацию, чем кто-то добежит до полки с книгами, отыщет нужную, кропотливо изучит оглавление, найдет нужную главу, откроет её начало и будет судорожно пролистывать страницы в поисках ответа на свой вопрос. Помимо поисковиков в некоторых случаях по отдельным продуктам удобнее использовать сопровождающую их документацию, но опять же с возможностью поиска, например MSDN.

Теперь что касается использования книг, как обучающего материала. Тут ключевое слово – это обучение. А обучение – это процесс, в котором участвуют, по крайней мере, двое – учитель и ученик, ну и ещё есть учебный материал (собственно, книга). И если учитель должным образом, с терпением и упорством не будет муштровать ученика, то фиг что из этого обучения получится. Методы кнута и пряника ещё никто не отменял, более того, это единственное, что может человека заставить что-то делать.

В случае с книгой, учителем является сам ученик, потому что это он **сам** добровольно сажает себя за изучение какого-нибудь материала, вместо того чтобы с чувством, с толком, расстановкой ковыряться в носу. Это конечно похвально, но такого учителя хватает ненадолго, и это нормально и естественно! Потому что абсолютное большинство из нас – не мазохисты. И когда нам какое-то действие делать тяжело,

неудобно, оно нам не нравится, или даже просто требует повышенного внимания, то с течением времени энтузиазм по этому поводу угасает все больше и больше, а работа потихоньку затухает.

Теперь конкретно о программировании.

*Чтобы освоить язык программирования, какую-либо методологию, либо среду разработки нужна **прак-ти-ка!***

Регулярная, постоянная, многочасовая практика. И тут появляется ещё одна проблема, которую не могут решить книги. Это **бесполезность** вашей работы!

У фашистов одной из самых страшных пыток считался бесполезный труд. Представляют, поднимают вас и таких же как вы рано утром, одевают и заставляют весь день катать по кругу огромный тяжелый многометровый камень. По кругу. Весь день. И следующий день. И следующий...

Так вот, бесполезная работа – ужасна и надоедает **моментально**.

Решение примеров из книг «для себя» - это конечно не бесполезно, тут, по крайней мере, для вас самих есть польза.. но как-то она не очевидна, как-то её мало.. ну, в общем, опять мотивация катится в тартарары, кто пробовал, тот точно знает. Долго на собственной воле что-то делать не получится – слишком много отрицательной обратной связи.

Поэтому, реальный стимул внутри себя найти очень сложно, но есть и хорошая новость, его можно найти снаружи! А именно, в работе или в учебе (но не в самообучении!). Впрочем, и здесь не все так гладко, рассмотрим, к примеру, вопрос...

Почему не работает высшее образование



Повторяем вновь, главное в обучении программиста – это **практика**, возможность с **пользой** применять полученные знания. Попробуйте выучить правила русского языка, не написав ни одной строчки. В школе нас постоянно заставляли писать диктанты, сочинения, примеры из учебника с пропущенными буквами и это неспроста.

Нет иного способа усвоить информацию, точнее даже не просто усвоить, а научиться **применять**. Между знанием и умением – пропасть! Благо, её можно пересечь по мосту регулярной практики.

А нормальной практики программирования ВУЗ дать не может. Сразу оговорюсь, сейчас мы рассуждаем о наших ВУЗах. Мы не говорим о западном образовании, где человек может **выбрать**, какие именно курсы ему интересно слушать. Мы тем более молчим про элитные заведения вроде

Оксфорда и Кембриджа, где у человека есть свой личный ментор, который направляет и контролирует его на протяжении всего пути обучения (поверьте, это несколько серьезнее чем куратор:)). Нет, мы имеем в виду нашу родную многострадальную систему образования, которая, надо сказать, отстала от мировых стандартов на десятилетия, к сожалению..

Можно посмотреть рейтинг университетов мира за 2009 год (<http://www.4icu.org/top200/>) и найти МГУ на 15 месте.

А следующий российский университет на... **а его там нет!**
Печальная картина...

Можно конечно бесконечно долго спорить о справедливости таких рейтингов, есть даже русская версия рейтинга, где МГУ занимает 5 место, опережая Гарвард, Стэнфорд, Кембридж, Оксфорд, но... но это смешно, честное слово.

Так вот, вернемся к практике программирования в российских университетах. Почему мы считаем её плохой:

1. **Возможность обхода контроля.** В школе каждый хотя бы раз просил товарища дать ему списать «домашку». Но там мы хоть собственной рукой что-то списывали, по мере чего у нас откладывались какие-то знания. На кафедре программирования это действие превращается в банальный «копипаст» программы, оставляя наш светлый разум таким же светлым и незамутненным лишними знаниями:) Вообще, проконтролировать авторство той или иной программы очень сложно. ВУЗ, с его масштабами с этой задачей справиться не может. А когда нет контроля, то остается надеяться только на честность студентов. А человек склонен к тому, чтобы ничего не делать, когда ему это позволяют. И запомните, даже многие ботаны вокруг Вас, которые выполняют по-честному все ДЗ, сдают всё только на отлично,

делают это не из-за того, что в них горит неудержимая жажда знаний (ну не может быть одинаковой жажды и к литературе, и к химии, и к черчению, и к программированию), а просто потому что:

- они боятся провала,
- их обучение кто-то жестко контролирует,
- им реально больше нечего делать:),
- так сложилось исторически с детства.

Так что нет ничего страшного в том, что Вы ленитесь. Главное знать это, уметь использовать и лениться в правильных местах, а не там, где реально нужно вкалывать.

2. **Следующая проблема, это устаревшая программа.**

Нам повезло, что мы учились в действительно сильном ВУЗе (и в плане программирования тоже). У нас, например, регулярно проводился Microsoft Day с различными конкурсами, викторинами и презентациями от вендора. Но те технологии, которые презентовались на Microsoft Day, попадут в программу обучения только через много-много лет. ВУЗ – это большая неповоротливая машина, которая с опозданием реагирует на все новое и актуальное. Моего друга совсем недавно в другом университете (в плане программирования сильно проще) семидесятилетняя бабушка учила FoxPro версии 2.6. Это программа 1992 года, если что.

3. **Ну и последняя проблема – недостаточная глубина образования.** Даже если Вы по-честному выполняете задания преподавателей, это все равно никогда не сравнится с тем опытом, который дает **реальная** работа. Знаете почему? Потому что у преподавателей на каждую задачу уже подготовлен ответ. А на работе перед Вами просто ставят проблему и время на её решение. Всё, дальше крутитесь как хотите, из кожи

вон лезьте, а будьте добры к поставленному сроку сделать то, что от Вас требуют. А в ВУЗе – типовые задачи, типовые решения, типовое обучение. В жизни задачи вовсе не типовые, они ставятся намного шире. И решение, соответственно, складывается из множества вариантов. И порой очень даже не типовых. И то, как Вы владеете умением связывать одно с другим, выстраивать архитектуру программы, глядя на проблему, проводить аналогии, чувствовать правильные решения, как раз и является Вашим **профессиональным** уровнем. И этот уровень напрямую отражается на зарплате и Вашем весе на рынке труда.

Таким образом, ВУЗ для программиста должен быть, по крайней мере, не единственным средством получения профессиональных знаний и опыта. Программированию в институте можно научиться, ТОЛЬКО если занимаешься разработкой где-то ещё. На работе, либо на курсах.

Почему работа и курсы работают... но редко

С работой ситуация следующая. С одной стороны она может дать действительно много опыта, а с другой стороны... может и не дать. Тут уж как повезет. Обычно, когда приходишь на первую



работу, то первые год-полтора получаешь массу полезных и ценных знаний. Но вот дальнейшего роста может и не быть. Вполне вероятно, что Вы будете выполнять изо дня в день

одни и те же рутинные действия. Через какое-то время Вы будете знать их наизусть и... ненавидеть. Просто у начальства есть конкретная потребность именно в этих действиях, оно даже готово за это адекватно платить. Только вот для Вас лично — это плохая альтернатива. Потом что нет никакого развития, и Ваша цена как специалиста перестает расти, что, конечно, очень плохо.

Курсы — другое дело. На курсы человек идет осознано, и это большой плюс. Потому что здесь Вы сами выбираете, что и в каком объеме изучать. Но к выбору курсов стоит относиться внимательно. Дело в том, что многие из них грешат все теми же проблемами, что и высшее образование — недостаток практики, поверхностность обучения, неактуальность знаний. Результатом таких курсов будет в лучшем случае бумажка об их окончании и обрывистые знания по предмету, которые невозможно выстроить в четкую систему. Слава богу, есть места, где учат неплохо, но и они обладают одним существенным недостатком... Это **цена**.

За хорошие курсы по языку или среде программирования в Москве придется выложить от 15000 руб.

Повторюсь, речь о **действительно хороших** курсах, где внятная теория сопряжена с хорошей практикой. Если же брать более узкую область или обучение специализированным программам (например, по интеграции CRM системы Siebel 8.0), то здесь цен ниже \$1000 вообще нет, и они легко могут превышать и \$5000, и \$10000. Так что, если есть деньги — записывайтесь на курсы, главное - не промахнитесь;)

Вообще, с курсами часто работает принцип «по вере вашей, да будет вам». То есть, с каким настроением пришел, то и получил. Если у Вас есть жгучее желание узнать что-то новое и начать это использовать, если Вы понимаете, зачем Вам это

нужно, то у Вас все получится в любом случае. Тут и курсы не обязательны.

Глава 4. Пора взглянуть на себя без прикрас



Чтобы понимать, куда двигаться дальше, нужно знать, где Вы находитесь прямо сейчас. Поэтому предлагаем Вам оценить себя, по ряду критериев. Проставьте напротив каждого пункта цифру от 0 до 10, в соответствии с Вашим текущим уровнем знаний и умений.

Критерии оценки программиста

- **Увлеченность новыми технологиями.**
0 – не желаете узнавать ничего нового,
5 – регулярно читаете статьи о современных средствах разработки,
10 – регулярно применяете все новинки в своей области программирования.

- **Общение с себе подобными.**
 - 0 – Вы – волк-одиночка и не общаетесь с другими людьми на темы программирования.
 - 5 – у Вас есть приятели, с которыми Вы порой можете обсудить собственные разработки.
 - 10 – Вы – активный участник ряда блогов и форумов, посвященных программированию.
- **Отношение к программированию.**
 - 0 – считаете, что это утомительная и рутинная обязанность,
 - 5 – нравятся отдельные моменты, Вы любуетесь результатами, но тяготитесь процессом,
 - 10 – сам процесс программирования всегда доставляет Вам удовольствие, Вы программируете в сконцентрированном и умиротворенном состоянии.
- **Наличие собственных проектов и разработок.**
 - 0 – Вы никогда не занимались собственными разработками.
 - 5 – имеется ряд задумок, но реализации либо не доведены до конца, либо заброшены.
 - 10 – у Вас есть ряд завершенных проектов. Люди пользуются Вашими разработками.
- **Широта технических знаний.**
 - 0 – у Вас нет математической и технической подготовки. Вы владеете только одним языком программирования, который изучили по самоучителю с примерами.
 - 5 – Вы получаете техническое образование. Вы знакомы с 2-3 языками программирования, 3-4 средами разработки, которыми приходилось пользоваться на работе или в учебе.
 - 10 – Вы имеете высшее техническое образование. Вы не можете точно сказать, сколько языков программирования Вы знаете, потому что не совсем понимаете вопрос. Вы готовы написать что-то на

незнакомом языке, как только появляется потребность и под руками есть документация. За время работы Вы столкнулись с бесчисленным множеством технологий, методов разработки, средств разработки, методологий ведения проектов, средств управления проектами.

- **Наличие технического образования.**

0 – отсутствует среднее образование.

5 – выпускник технического ВУЗа.

10 – кандидат или доктор наук.

- **Упорство.**

0 – вы не довели до конца ни одного начатого проекта.

5 – Вы участвовали в разработке ряда проектов, которые были успешно завершены.

10 – у Вас есть собственные успешно завершённые проекты и большое число совместно выполненных проектов.

- **Обучаемость.**

0 – необходимость использовать незнакомую технологию вызывает сильнейший дискомфорт, вплоть до отказа от участия в разработке.

5 – Вам необходимо достаточно большое количество времени на чтение документации и овладение новой технологией, но в результате Вы добиваетесь своего.

10 – Вы с лёгкостью осваиваете новое в течение одного дня, основываясь на богатом опыте разработки.

- **Кругозор.**

0 – У Вас нет друзей. Вас интересует исключительно программирование. Все остальные сферы жизни кажутся поверхностными и несодержательными.

5 – свободное время обычно тратится на программирование/техническую литературу. Более половины Ваших друзей, также как и Вы, технари.

10 – Вы всесторонне развитый человек. Несмотря на то, что Вы находите программирование увлекательным занятием, Вы имеет большое количество других интересов. У Вас есть множество знакомых с совершенно разными интересами.

- **Подход к разработке.**

0 – необходимость написания программы заставляет Вас сразу же приступить к разработке не задумываясь об архитектуре решения. В процессе разработки регулярно появляется необходимость переписывать часть кода ввиду неправильного изначально выбранного подхода.

5 – Перед тем, как приступить к разработке Вы продумываете какими средствами можно выполнить поставленную задачу.

10 – Вы тщательно разрабатываете архитектуру будущего программного решения, выявляете наиболее тонкие места. Стараетесь добиться наилучшей масштабируемости, предусмотреть любые варианты возникновения ошибок.

- **Культура кода.** Этому вопросу посвящен ряд статей у нас на [сайте](#). Чтобы понять критерии, стоит с ними ознакомиться. Если коротко,

0 – код бескультурный, черт ногу сломит, расширяемость нулевая, комментариев нет.

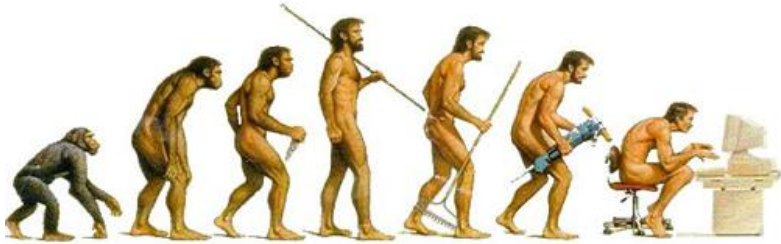
10 – комментированный, хорошо оформленный код, которым легко пользоваться и легко масштабировать.

Оцените себя по каждому пункту. Мы не будем писать традиционное «от 0 до 30 баллов – Вы никудышный программист». Эта оценка нужна исключительно Вам, так что не обманывайте себя.

Не нужно стремиться стать монстром с десятками в каждом пункте, лучше понять какие из них проседают сильнее и что можно сделать, чтобы улучшить картину. Выберите пункты,

которые наиболее запущены, посмотрите на десятибалльный идеал и решите, как можно исправить ситуацию **уже сегодня**.

Глава 5. Начинаем взлетать. Концепция развития программиста



Есть такое выражение «Талантливый человек - талантлив во всем». Действительно, успешным людям часто сопутствует удача, за какое бы дело они не взялись. В чем же их секрет? Секрет в том, что они живут **гармоничной** жизнью. Они развиваются в разных направлениях одновременно, интересуются разными вещами, не зацикливаясь на чем-то одном. Всестороннее развитие позволяет смотреть на проблемы сверху и переносить идеи из одной сферы жизни в другую.

Кроме того, если не получается решить какую-то проблему, то можно попробовать её «отпустить». То есть переключиться на что-либо другое, и тогда ответ зачастую всплывет сам собой. Это все равно, что пытаться пробить стену головой. Иногда нужно просто сделать шаг назад и увидеть рядом открытую дверь.

Поэтому нельзя отделять профессиональное развитие от своего общего развития. Если пытаться развиваться исключительно в узких рамках профессии или ещё уже – в определенном направлении программирования, то эффект будет, но значительно меньший, нежели при всестороннем

росте. Таким образом, концепция развития программиста, технического специалиста, а, вообще говоря, любого человека должна быть следующей:

1. **Развитие профессиональных навыков.** Естественно, профессии должно уделяться особое внимание.
2. **Развитие сопряженных навыков.** Помимо профессиональных существуют навыки тесно связанные с Вашей деятельностью, развитие которых позволит Вам существенно вырасти, как специалисту.
3. **Развитие общих навыков.** Не единым программированием жив человек, это надо понимать. Всестороннее развитие личности дает силы, энергию и настроение. Без этого, сами понимаете, никуда.

Далее подробно рассмотрим каждую из составляющих успеха.

Часть 3. ПРАКТИКА

Глава 6. Мощный рывок вперед. Развитие профессиональных навыков

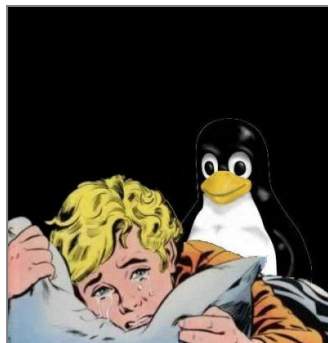


Далее мы приведем ряд правил, которые позволят существенно повысить свой профессиональный уровень. Есть только одна сложность – эти правила необходимо **ВНЕДРЯТЬ** в свою практику, а не просто знать. Внедрять – это значит прочитал и сразу же применил, не завтра, не с понедельника, а **СРАЗУ ЖЕ**. Иначе эффекта не будет и можно забыть обо всех своих амбициях.

Регулярная практика программирования

В этой книге мы уже не раз затрагивали тему того, что успех приходит только к тем, кто **много** работает. Сказки про Иванушку-дурачка, который лежит на печи, а потом становится "принцем" не выдерживают суровой правды жизни:) По крайней мере, в программировании это точно не прокатит. На самом-то деле в программировании все вообще проще простого, очевидней не придумаешь: хочешь стать крутым профессионалом-программистом — программируй. И всё. Постоянно и регулярно пиши программы, часто и много. Изучай новые технологии, и, **главное - применяй** их. Информации в инете — море! Учебники, документации, статьи — все доступно и, при том, безвозмездно) Где, где, а в области программирования можно нарыть любую интересующую информацию.

Так что остается только найти себе задачу и приступить к реализации.



Быть в авангарде



Следующий ключевой пункт, на который следует обратить внимание, - это актуальность. IT развивается абсолютно бешено и замедляться не планирует. Можно писать замечательные консольные программы на паскале, но только вряд ли они найдут благодарных пользователей. Все потому, что средства разработки

совершенствуются постоянно. Та область, в которой сегодня Вы - пионер, завтра будет использоваться миллионами. То, что сегодня используется миллионами, завтра станет вчерашним днем и отправится в утиль. Нужно постоянно держать руку на пульсе, чтобы в один прекрасный день не оказаться в аутсайдерах.

На самом деле следить за технологиями не так уж и сложно. В IT, как и везде, все новое – это хорошо забытое старое. Нельзя сказать, что за последние годы в программировании появилось нечто принципиально новое, кардинально отличающееся от всего известного ранее. Все, что человек придумывает, основывается на его предыдущем опыте. Секрет в том, что помимо чьих-то там технологий есть Ваш личный опыт, собственная база знаний и технический склад ума, которые от Вас никуда не денутся. И если Вы обладаете этим бесценным багажом, то оставаться в авангарде информационных технологий оказывается вовсе и не сложно.

Книги must read

Мы уже писали, почему книги не работают в случае обучения программированию. Тем не менее, есть литература, которая обязательно должна быть прочитана любым уважающим себя программистом. Это книги, в которых концентрация полезных знаний зашкаливает. Поэтому, даже если после прочтения у Вас в голове останется 5% изложенной информации, это уже поднимет Вас, как специалиста на новую ступень. На самом деле, хороших книг много, но эти – просто жемчужина литературы об IT:

- **Искусство программирования.** Дональд Э. Кнут



- **Объектно-ориентированный анализ и проектирование.** Гради Буч
- **Совершенный код (Code Complete).** Стив Макконел.
- **Мифический человеко-месяц или Как создаются программные системы.** Хилл Чапел, Фредерик Брукс
- **Приемы объектно-ориентированного проектирования (Паттерны проектирования).** Э.Гамма, Р. Хелм, Р. Джонсон, Д. Влиссилес.

Эти книги не о конкретных языках, они о том, КАК надо писать программы, КАК надо строить процесс разработки, КАК проектировать свое ПО.

Умение искать, отфильтровывать и использовать информацию



Учитесь четче видеть детали проблемы, с которой столкнулись. Выделяйте подпункты проблемы, если это возможно, и старайтесь максимально конкретизировать причину проблемы. Админам часто приходится решать проблемы сотрудников компании, сформулированные примерно так: «у меня что-то там в компьютере не работает». Админ приходит и видит, что «что-то там в компьютере» - это синий экран смерти при загрузке с кодом 0x000000B4. Если админ раньше не сталкивался с такой ошибкой, он идет и ищет информацию в интернете конкретно по этому коду и решает проблему в течение получаса. Просто нужно четко сформулировать проблему, система же нам сама пытается её подсказать, выдавая код ошибки. А вот на уровне «в компьютере что-то..» ничего решить нельзя.

Если написанная программа дает сбой, значит лезем в дебагер и отлаживаем. Если падает чей-то софт, значит смотрим логи, если они есть. Если их нет – стараемся детально установить, при каких обстоятельствах происходит сбой. Если выяснения ещё не привели к решению, то ищем его в интернете, при этом четко указываем проблему. Поверьте, в 99,9% случаев Вы не первый, кто с этой ошибкой сталкивается, и решения давно уже предложены на различных форумах. Конечно в англоязычном Интернете информации на порядок больше. Вообще, если хотите быть хорошим программистом, то чтение технической литературы на английском не должно вызывать проблем. Хотя бы на уровне форумов.

Пошаговая инструкция развития



Мы прочитали уйму книжек, которые вроде как какую-то идею передают, а инструментов реализации не предлагают.

Это с одной стороны наша проблема, что мы не можем прочитанное перевести из разряда теории в практику, но с другой, авторы таких произведений могли бы и побольше внимания уделить описанию именно практической стороны дела.

Дабы такой ошибки избежать в этой книге мы Вам предлагаем уникальный пошаговый план развития программиста от новичка до настоящего гуру.

*Программирование – это не некая наука, с четко выделенными границами. Программирование – это скорее **совокупность знаний и опыта**, находящих применение в области разработки программного обеспечения.*

А область это широка неимоверно, и, соответственно, знаний всякого рода также огромное число. Так вот, здесь мы попытались расставить их в том порядке, в котором они по-настоящему необходимы программисту по мере его профессионального роста. То есть, к примеру, если Вы сейчас являетесь студентом, который подрабатывает на своей первой в жизни работе, Вам нет необходимости забивать голову материалами о риск-менеджменте в области ИТ-проектов. Это полезная информация, но сейчас есть нечто более важное, что Вам необходимо узнать. Со временем доберетесь и до этого, но только если преодолете все предыдущие ступени развития.

Сразу оговоримся, шаги развития, приведенные ниже, не являются четко выделенными. Скорее всего, у Вас есть что-то от разных ступеней. Стоит обратить внимание на методы, предложенные в каждом из этапов. Сроки, которые я привожу здесь также очень условны. Иногда человек за короткий срок делает ТАКОЙ рывок вперед, что вместо рядового разработчика оказывается на уровне управленца. С другой стороны, как мы увидим позже, большинство

специалистов, добившись незначительных успехов, наглухо застревают на своем месте.

Итак, вот он, уникальный пошаговый план развития программиста с нуля до заоблачных вершин.

Шаг 1. Новичок. Опыт работы = 0. Знания в рамках школьных и институтских курсов информатики.

Славный период, когда мы мало что знаем, ещё меньше умеем, но страстно желаем начать работать. Несмотря на то, что рано или поздно эту стадию преодолеют все (все же когда-то начинают работать), её можно назвать одной из самых сложных. Приоритеты здесь очень простые, главное - НАЧАТЬ что-то делать. Что-то, чем будут пользоваться другие люди. Заметьте, тут речь ВООБЩЕ не идет о деньгах. Если Вам за Вашу работу готовы платить – прекрасно, нет – ничего страшного. Сейчас намного важнее, чтобы цифра ноль напротив опыта работы сменилась на нечто более привлекательное. Тем не менее, впадать в альтруизм не нужно, и стоит подготовиться к собеседованию на работу, чтобы показать себя с лучшей стороны и, как следствие, выбить большую з/п. О том, как готовиться к собеседованию, я отдельно напишу в следующей главе. Скажу только, что новичку необходимо собрать весь, какой есть, опыт, каким-либо образом связанный с программированием. Если надо, притянуть его за уши. Важно, чтобы было о чем сказать по поводу своего опыта. Если вы думаете, что сможете таким образом обмануть людей, которые будут Вас собеседовать, то нет, конечно же. Новичка видно невооруженным глазом. Но всё же информация о каком бы то ни было опыте в сто раз лучше молчания в ответ на вопрос собеседника.

Что прочитать. Ну, во-первых, нужно конкретизировать сферу того, что Вам интересно в программировании. Быть может для Вас представляет интерес веб-программирование,

или Вам больше по душе прикладные программы, а возможно, Вам хочется заниматься низкоуровневым системным программированием. Определитесь, к чему Вы тяготеете больше всего, и смело приступайте к изучению предмета. Поищите на форумах, какие книги народ рекомендует по Вашей теме. Узнайте, кто считается экспертом в выбранной области, и какие книги у него есть.

Что попробовать. Читая книгу, старайтесь выполнять примеры и задачи, которые в ней описаны. Найдите в Интернете сайты, посвященные данной тематике. Там можно найти полезную информацию и ответы на часто возникающие вопросы.

Как перейти на следующий уровень. Подготовиться к собеседованию и получить свою первую работу. Тут не может быть никакой погони за деньгами. Ваша задача – максимально расширить свои знания и умения. Впрочем, это уже следующая ступень.

Шаг 2. Advanced Новичок. Опыт работы ≤ 0,5 года. Знания в рамках школьных и институтских курсов информатики + полученные на работе навыки решения конкретных задач.

Этот период охватывает промежуток времени от получения предложения о работе до окончания испытательного срока. Важный период, когда начальство проявляет к вам повышенное внимание. С самого начала постарайтесь зарекомендовать себя с лучшей стороны. Внимание руководства будет обращено не на размер Ваших знаний, а на Ваш подход к работе. Старание, внимательность, пунктуальность и быстрота выполнения поставленных задач – это залог высокой оценки начальства. Если вы выполнили то, что Вас просили сделать, сразу отчитайтесь об этом руководству. Спросите, что ещё необходимо выполнить. В работе всегда появляются какие-то моменты, когда можно

похалтурить, полазить в Интернете, заняться какими-то левыми делами. Так вот, в первые полгода не позволяйте себе этого! Уверю, это Вам многократно зачтется в течение всей дальнейшей работы в этой компании. Действует старый студенческий принцип – сначала ты работаешь на зачетку, потом она на тебя.

Что почитать. Продолжайте углублять свои знания в области деятельности. К концу этого этапа Вы должны владеть языком и средой разработки на твердую 4-ку, а лучше на 5-.

Что попробовать. Так как теперь у Вас есть реальная работа, старайтесь применять полученные знания в реальных боевых условиях. Если у Вас появляется предложение по оптимизации какой-то функциональности или реализации, смело идите с ним к начальству или тому, кто за это отвечает. Не бойтесь высказывать свое мнение, пусть сначала оно не будет достаточно компетентным. Пусть начальство покажет, почему оно считает Ваше предложение неправильным. А вы в этот момент поглощайте всю информацию, которую получите. Это станет очень ценным опытом, приобретенным быстро и бесплатно. Вообще, обращайтесь почаще к старшим товарищам, если таковые имеются, с просьбами рассказать о том, что Вас интересует. Только не становитесь назойливыми, не отрывайте их от работы. Находите моменты, когда человек сам будет готов поделиться собственным опытом. Для этого, старайтесь выстроить хорошие отношения с коллегами. Вы увидите, что люди сами готовы с удовольствием делиться своими знаниями, если заметят, что эта информация по-настоящему ценна для Вас. Но и перебарщивать не стоит. Постоянные вопросы по любому поводу жутко выводят из себя:)

Как перейти на следующую ступень. Если выполнять всё, что описано выше, то Вы автоматически попадете на следующую стадию вместе с успешным окончанием испытательного срока.

Шаг 3. Младший программист. Опыт работы 0,5-1 год.

Только на этом шаге я решился употребить слово программист. Потому что, проработав хотя бы полгода и столкнувшись с реальными задачами, человек, что называется, оказывается «в теме». Он уже решил все стартовые проблемы с обустройством на новом месте работы, с организацией рабочего времени, пространства и инструментария и теперь может сконцентрироваться на решении задач с максимальной эффективностью.

Итак, тут наконец-то можно заняться не только изучением документации и всяческой справочной информации, а подойти к программированию именно как к искусству написания программного кода.

Что почитать. Книги об искусстве программирования. Например, трехтомник Кнута «Искусство программирования». НО только то, что наиболее актуально и полезно. Не нужно себя мучить сложной литературой в таких объемах, а то вообще пропадет желание развиваться)

Также хорошо почитать о тонкостях языков, программных сред, фреймворков, средств разработки, с которыми Вам приходится иметь дело на работе. Эффект от умелого владения инструментами разработки будет заметен и Вам, и Вашим коллегам, и начальству, это абсолютно точно.

Что попробовать. Попробуйте потратить время, в которое Вы сидите вконтакте или одноклассниках (если это имеет место быть), либо играете в WOW или ещё во что-нибудь, на самообучение. Но вообще знайте, что если Вы в день тратите на бесполезные дела менее 3 часов, то Вы КРАЙНЕ эффективны! Свободное время можно найти всегда, и не обязательно отрывать его ото сна, достаточно эффективнее выполнять остальные дела.

Как перейти на следующую ступень. Здесь переход обычно происходит сам собой вместе с ростом трудового стажа. В общем-то, особого рвения в работе эта стадия не требует. Но

если его нет изначально, стоит задуматься о своих профессиональных целях и о том, правильно ли Вы выбрали место для их реализации.

Шаг 4. Программист. Опыт работы 1-3 года.

К сожалению, часть программистов застревает на этой стадии развития. 3 года стажа работы в одной должности превращаются в 5 лет, затем в 10. Человек вдруг понимает, что по возрасту уже должен находиться совершенно на другой ступени карьеры, и испытывает большое разочарование в себе, профессии, своей компании и т.д. В общем, начинается депрессия. Обычно, это свойственно людям, которые не задумываются о своем продвижении и профессиональном росте, которым чужды амбиции и стремление к личностному развитию. Но если Вы читаете эту книгу, скорее всего, это не про Вас, поздравляем!)

Что почитать. Продолжаем совершенствоваться в искусстве программирования. В самый раз прочитать что-то вроде «Совершенного Кода» (лучше всего его и прочитать). К этому добавляем чтение блогов и сайтов по теме.

Что попробовать. Будем считать, то, НА ЧЕМ программировать, Вы уже освоили, как свои пять пальцев. А, как известно, людям, хорошо разбирающимся в предмете, всегда есть о чем поговорить. Обычно разговор принимает форму блогов или форумов. Начинайте общаться на профессиональные темы, делитесь опытом, задавайте вопросы. Создайте свой собственный блог, в котором Вы смогли бы делиться полезной информацией.

Как перейти на следующую ступень. Тут надо понимать, что из себя представляет должность старшего программиста. Это может быть либо просто программист с внушительным опытом работы, либо некий team-лидер, наделенный к тому же элементами управленческих полномочий. В любом случае, на данном этапе постарайтесь стать неформальным лидером среди коллег. Для этого необходимо а) разбираться

в чем-то лучше остальных, б) иметь достаточные коммуникативные навыки. Никаких сверхъестественных способностей не требуется. Достаточно сохранять хорошие отношения в коллективе и быть готовым прийти на помощь. Завоевав уважение окружающих, можно легко формализовать свое неформальное лидерство:)

Шаг 5. Старший программист. Опыт работы от 2 лет.

На наш взгляд, начиная с этого этапа, работа становится интересней. Вы начинаете решать задачи не только связанные с написанием кода. Появляется некоторая свобода в принятии архитектурных решений, пусть сначала и не очень существенных. Со временем, если Вы докажете, что на Вас можно положиться, вышестоящие архитекторы и управленцы смогут наделить Вас ещё большими полномочиями. Они с удовольствием пойдут на это, потому что таким образом они смогут Вашими руками выполнить часть собственной работы.

Что почитать. Если не прочитан «Совершенный Код», то читаем в обязательном порядке. Читаем литературу, более узко заточенную под конкретно Вашу программистскую кухню. Также очень желательно прочесть «Как пасти котов», надо быть готовым к управлению программистами и понимать, насколько грамотно управляют Вами.

Что попробовать. Попробуйте прокачать сопряженные навыки. Речь о них пойдет в следующей главе. Смысл в том, что по-настоящему успешные люди гармонично развиваются в разных областях жизни. Есть навыки, которые могут способствовать более эффективной и успешной работе, их необходимо совершенствовать.

Как перейти на следующую ступень.

Существует ряд вариантов развития событий. Первый – Вы работаете в небольшой компании. При успешном развитии событий, росте компании, появлении новых проектов вполне

возможно что Вы автоматически и без каких-либо трудностей шагнете на следующую ступень. Второй вариант – Вы работаете в крупной компании, коллектив достаточно большой, сильного роста компании нет, и не предвидится. Тут есть два варианта. Либо завоевывать себе место под солнцем в честных и нечестных боях с коллегами. Либо искать варианты на стороне, то есть присмотреться, можно ли вместе со сменой работы шагнуть на ступеньку вверх по карьерной лестнице. Иногда это делается даже в ущерб зарплате. Ситуация в каждой конкретной компании уникальна, поэтому выбирайте то, что наиболее подходит для Вас. Помните только, что никогда не вредно знать, насколько рынок заинтересован в специалистах вашего профиля, и как дорого вы реально можете себя продать. Для меня, например, несколько раз было откровением то, что моя реальная стоимость в два раза превосходила текущую зарплату. С этими железными аргументами я шел к начальству, и они, как ни странно, со мной соглашались. Я сам, побывав в шкуре человека, проводящего собеседования, понял, что когда человек на вопрос о величине зарплаты говорит от 100 рублей и выше, то, значит, он заведомо согласен на 100 рублей и никто ему больше предлагать не будет.

Шаг 6. Функциональный архитектор. Опыт работы от 3 лет.

Если Вы дошли до этой ступени, то можно смело сказать, что Вы уже многого добились. Наверняка появился вкус и жажда карьерных успехов. Работа стала на порядок интересней, чем ранее. Остается пожелать одного – не останавливайтесь.

Что почитать. Если не прочитана книга «Как пасти котов», то читаем её. Также читаем книжки по построению архитектуры ПО и управлению проектами в области разработки программного обеспечения. Если владеете английским, хорошо было бы обращать внимание на блоги западных гуру

программирования или гуру непосредственно Вашей области деятельности.

Есть ещё одна категория книг, которые мы рекомендовали бы периодически перечитывать на протяжении всего пути развития. Это биографии выдающихся ИТшников. На данном этапе уже надо знать людей, которые сделали отрасль такой, какая она есть сейчас.

Что попробовать. Продолжаем совершенствовать те навыки, которые могут положительно сказаться на работе - от языков до коммуникации. Подробнее в следующей главе.

Как перейти на следующую ступень. На таком уровне уже нет хорошо работающих шаблонов успеха. Но и случайных людей на таком уровне практически нет. Так что если у Вас есть стремление к дальнейшему росту, внутренняя сила, уверенность в себе и лидерские качества, то всё будет хорошо в любом случае. Здесь уже не Вас выбирают, а Вы выбираете где и как работать. Потому что таких профессионалов как Вы мало. А эксклюзивность рождает привилегии.

Шаг 7. Менеджер проектов. Опыт работы от 5 лет.

Начиная с этого этапа, голова у человека все меньше забита программированием и все больше вопросами управления и организации. Для этой и следующих стадий ничего конкретного советовать не будем по нескольким причинам. Во-первых, чем выше, тем уникальней путь каждого. Уже невозможно дать каких-то общих рекомендаций. Во-вторых, столь высокая позиция говорит о заслугах человека. Скорее всего, он уже не нуждается в советах, позволяющих определить его дальнейший вектор развития. Он сам понимает куда и зачем он движется.

Базовыми остаются только рекомендации о всестороннем гармоничном развитии. Нельзя допускать, чтобы одна из сфер жизни отнимала критически много времени, тогда как

другие оставались не удел. Поселившись на работе, можно распрощаться и со здоровьем и с семьей. Поэтому стоит заранее задуматься, хотите ли Вы такой ценой добиваться успехов по работе. В долгосрочной перспективе это заведомо проигрышная стратегия. Так как запущенные сферы жизни в итоге утянут за собой всё, что пока более-менее благополучно.

Ну а среди оставшихся шагов можно выделить два. Оставим их без каких-либо комментариев. Там всё сильно по-другому:)

Шаг 8. Руководитель IT-отдела.

Шаг 9. Руководитель компании-разработчика ПО.

Глава 7. Ускорение до максимума. Развитие сопряженных навыков

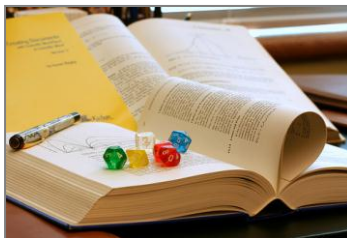


Итак, уже неоднократно мы повторили, что развитие навыков, которые имеют косвенное отношение к профессии программиста, также очень полезно для его роста, как специалиста и вообще. Остановимся на наиболее значимых вещах.

Английский язык

Самые правильные мануалы, самые полезные форумы, самые интересные блоги всегда были и будут на английском языке. В ИТ индустрии английский давным-давно завоевал полное господство и перемен не предвидится. Поэтому

техническую документацию на языке придется читать всем, это только вопрос времени. Впрочем, нет никакой необходимости доводить свой английский до Intermediate+. Достаточно знать столько, сколько необходимо для работы. Здесь мы предложим варианты того, как можно увеличивать навык владения без дорогостоящих курсов и репетиторов.



- **Метод Пимслера.** Представляет из себя набор аудио-уроков. Уроки состоят из тематических диалогов с переводом и объяснениями. Пояснения и комментарии изучаемого на уроках вы слышите на русском языке, сам материал - разговорный английский язык. Слушаете фразы, повторяете, диктор Вам отвечает, таким образом получается некий диалог. К сожалению, в бесплатном доступе в интернете можно найти только первую (из трех) часть курса. Вторую и третью часть предлагают на отдельных платных онлайн тренингах, например, здесь <http://spyschool.ru/magicseminar/index.php>. От себя скажу, что это **действительно эффективный** метод обучения. Поищите о нем больше информации в Интернете.
- **Метод Ильи Франка.** Предлагает для чтения особым образом подготовленные тексты. Если открыть такой текст, то можно увидеть, что он разбит на небольшие отрывки. Сначала идет адаптированный отрывок — текст с вкрапленным в него дословным русским переводом и небольшим лексико-грамматическим комментарием. Сам автор так комментирует свой метод: «В любом случае, мой метод чтения дает лишь пассивное

освоение языка, то есть является вспомогательным по отношению к активирующим язык разговорным занятиям или к общению на чужом языке, но в своих пределах применимости он уже принес пользу многим (судя по отзывам), научившимся благодаря нему читать на чужом языке, значительно расширившим свой словарный запас, привыкшим к восприятию письменной речи и к строению языка».

- **Фильмы и музыка на английском.** Наибольший эффект от изучения достигается, когда происходит полное «погружение» в язык. Просмотр фильмов на английском – это неплохой вариант «погружения». Однако, если Вы только начали обучение, или продвинулись не очень далеко, то простой просмотр фильма на языке не будет иметь никакого эффекта. Вы ничего не поймете. Поэтому предлагаем следующую последовательность, которая позволит плавно адаптироваться к звучанию иностранной речи:

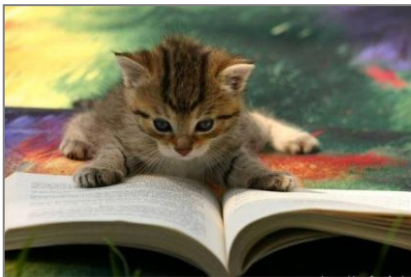
- фильм с русским переводом + фоном слышна речь на английском
- фильм на английском языке + русские субтитры
- фильм на английском языке + родные субтитры
- фильм на английском языке

Пару слов про музыку, а точнее про песни. Если Ваша любимая группа поет на английском языке, то это прекрасный повод изучить тексты их песен. В таком случае слова запомнятся намного легче, чем при обычной зубрежке. Мы вообще сильно против зубрежки:)

- **Общение с иностранцами.** Безусловно, самый эффективный метод, если Ваша цель – именно разговаривать на языке. Любое общение запоминается лучше любого заучивания.

Учитесь читать наискосок

Ещё одним чек-поинтом в собственном развитии может стать освоение скорочтения. Несколько базовых советов, которые касаются технической литературы. Скорочтение требует некоторой степени концентрации, поэтому не



следует его применять, когда Вы отдыхаете и читаете что-нибудь художественное для удовольствия.

Итак, любую техническую книгу сначала просмотрите по диагонали. В содержании отметьте те

главы, которые представляют для Вас наибольший интерес и сразу переходите к их изучению. Другой вариант – быстро просматривать по порядку каждую главу и как только увидели что-то интересное, переходите к внимательному чтению.

В интернете можно найти бесплатные книги или аудио-курсы по скорочтению. Прочитав или прослушав материал и выполнив задания, реально поднять скорость своего чтения на 50% и более.

Подготовка к собеседованию

Открою небольшой секрет. Сейчас уже начала готовиться новая книга от ШП, в которой мы поделимся рецептами успешного прохождения собеседования для программистов.

В ней мы соберем все самые лучшие методики и фишки, которые позволяют успешно пройти собеседование, даже когда вы объективно не



удовлетворяете заявленным требованиям. Книга будет основана на нашем личном опыте, опыте огромной армии коллег и друзей, а также выжимке из самых топовых книг по данной теме. Плюс адаптация материала к программистским реалиям. Здесь же мы поделимся только некоторыми важными правилами:

- **Если у Вас нет опыта, создайте его сами.** Об этом мы подробно написали на первом шаге развития программиста, поэтому здесь останавливаться не будем. Общее пожелание: нет опыта – притягивайте за уши, все что каким-то образом можно под него подогнать. Прекрасный способ получить опыт – вписываться в стартапы или создавать их самому. Если у Вас есть идея собственного сайта, блога, программного обеспечения, почему бы не приняться за его реализацию. Можно найти единомышленников и сделать это вместе. Если идей нет – не страшно. Сейчас в интернете появились сайты, посвященные стартапам. Просто берем и напрашиваемся куда-нибудь. Море опыта и полезных связей Вам гарантировано. Вот один из подобных сайтов <http://ru.startupex.net/>.
- Второй момент, который нам никогда не нравился, но он имеет место быть. **На собеседовании врут обе стороны.** Все хотят казаться лучше, чем они есть на самом деле. Это относится не только к соискателю, он и к работодателю. Поэтому, приходя на собеседование, не нужно испытывать возвышенных чувств и благоговейного трепета. Постарайтесь выяснить, как **на самом деле** обстоят дела с белизной зарплаты, соц. пакетом и прочими фишками, которыми пестрят вакансии. Расспросите о перспективах развития как компании, так и себя лично. И если в чем-то эйчарщики допустят прокол, проговорятся, признают недостаток, то у Вас появится

новый козырь при определении размера оплаты труда.

- **Будьте уверенным в себе.** Собеседование, это не то место, где мы должны из уважения к кому-то занижать свои ожидания по зарплате. Торгуйтесь, выбивайте те условия, которые будут не просто приемлемыми, а по-настоящему хорошими для Вас. Конечно, все это нужно делать, не забывая о собственной трезвой оценке. Не пытайтесь за счет собеседования перемахнуть через несколько ступеней карьерной лестницы и оказаться там, где задержаться всё равно не сможете. Рано или поздно Ваш настоящий уровень будет виден как на ладони. Но, тем не менее, бороться за лучшие условия всегда надо. Главное самому верить, что Вы действительно тот, кто сможет быть максимально полезен компании (это то, что нужно работодателю). Своими знаниями, своими качествами, своим стремлением вперед. А такой ценный сотрудник требует и соответствующей оплаты труда:)
- **Не будьте наглым.** Это сразу отталкивает. Ведите себя скромно, но с достоинством.
- **Покажите интерес к компании и работе,** которую вы хотите получить. Узнайте как можно больше о том, что вам придется делать, и что уже сделано до вас.
- Постарайтесь как можно более аргументировано ответить на вопрос, **почему именно Вы заслуживаете место в компании.** Ведь именно это больше всего интересует работодателя.
- Маленький совет от меня, как от человека, который собеседовал программистов. Вам задали вопрос, какая же зарплата Вас устроит. Не смейте говорить что-то типа «от 100 до 150». Во-первых, вторую цифру работодатель даже не дослушает. Во-вторых, «от и до» – это признак крайней неуверенности в своих

силах. Если хотите попробовать выбить побольше и, в то же время, чувствуете, что место классное, и пролетать не хочется, даже если предложат меньше, скажите так: «Мои ожидания – это XXX у.е.». Именно в такой формулировке. Это вроде бы ещё не конечная цифра и оставляет место для встречного предложения, но с другой стороны это и не метания в каком-то диапазоне. Такая формулировка показывает, что Вы знаете ситуацию на рынке и оцениваете себя таким образом. В то же время Вы перекидываете мяч на сторону работодателя, пусть он сделает Вам конкретное предложение, от которого Вы либо откажетесь, либо согласитесь. Но следующей итерации уже не будет, поэтому четко определите, какое предложение Вы будете готовы принять. Чтобы не возникло колебаний, когда Вам озвучат конечную цифру.

Глава 8. В полушаге от идеала. Развитие общих навыков



В этой главе опишем факторы, которые напрямую не связаны с программированием, но, тем не менее, оказывающие большое влияние на эффективность работы человека. Внедрив у себя те качества, о которых пойдет речь, Вы сможете быть успешны в любом деле, за которое бы ни взялись.

Физическое и эмоциональное состояние

Казалось бы, очевидная вещь, хорошее физическое и эмоциональное состояние позволяет работать на порядок лучше, меньше уставать, быть более сконцентрированным. Это вроде бы все понимают, и, в то же время, только



единицы уделяют этим сферам жизни должное внимание.

Так что пора начать следить за своим физическим состоянием. Тут мы Америки не откроем, и рекомендации могут быть самыми банальными:

- Бег по утрам
- Тренажерный зал
- Бассейн
- Спортивные секции
- Ролики/велосипеды/коньки/экстремальные виды спорта

Внедрите хотя бы что-то одно в свою жизнь и проследите за эффектом. Если что-то делать тяжело, то главное продержаться первый месяц. Потом выработается привычка и Вы начнете получать удовольствие от процесса. Ещё один маленький совет, если одному тяжело пойти и записаться в какой-нибудь тренажерный зал или секцию, подбейте на это дело друга, вместе всегда проще и веселее.

Правильный отдых



Некоторые считают, что у них просто нет времени заниматься спортом. В 99,9% случаев это отговорки собственного ленивого эго. Если Вы за день хотя бы раз заходите на такие сайты как вконтакте,

одноклассники, читаете чьи-то ЖЖ, твиттеры, ленты новостей, если вы включаете телевизор у себя дома или играете в компьютерные игры, значит У ВАС ЕСТЬ ВРЕМЯ. И только **Вам решать**, тратить ли его на бесполезные занятия, в то время как кто-то рядом делает что-то полезное, развивая конкурентные преимущества. Мы не говорим о том, что надо забыть об отдыхе и работать от заката до рассвета, нет, даже наоборот, отдых должен присутствовать постоянно. Но это должен быть **эффективный** отдых, который надо планировать заранее.

Эффективный отдых подразумевает смену сферы Вашего внимания.

Существует три основных вида деятельности – интеллектуальная, физическая и эмоциональная. Соответственно, работа и следующий за ней отдых должны лежать в разных сферах. Предположим, что на работе Вы сидите за компьютером и программируете 8 часов в день. Тогда самым худшим вариантом отдыха может быть переключение с программирования на просмотр каких-то сайтов, новостей, форумов и т.п. Это даст минимальный эффект. Лучшим вариант в данном случае будет какая-то физическая активность. Пройдитесь, прогуляйтесь, если есть

возможность, сделайте какие-нибудь физические упражнения, разомнитесь одним словом.

Ещё один важный момент, отдых должен быть **регулярным**. Когда Вы, не отрываясь от монитора, сидите по несколько часов за компьютером, это приводит к перегрузке нервных центров. Концентрация падает, сосредоточиться становится все сложнее, Вы чувствуете усталость, хотя весь день просидели на стуле. Вместо этого каждый час делайте по небольшому 5-минутному перерыву, в течение которого старайтесь максимально менять свою деятельность. В таком режиме усталость придет значительно позже, если вообще придет :)

Планирование

Чтобы добиться в чем-то успеха, нужно понимать, куда вообще двигаться. Поэтому потратьте время, сядьте и **напишите на бумаге свои** ближайшие планы и цели. **Зачем**

Вы читаете эту книгу? **Чему** Вы хотите научиться?

Почему Вы хотите этому научиться?

Что из уже прочитанного Вы сможете

применить в своей жизни? **Какие** ещё

вопросы у Вас есть? **Где** можно найти на них ответы?

Распишите каждый из этих вопросов. Всё, что Вы пишете на бумаге, намного стройнее и яснее укладывается в голове. Это буквально магия какая-то, но это так!

Так что когда Вы видите проблему, но она кажется слишком большой, сложной и не поддается быстрому решению, начинайте писать. Расписывайте все аспекты, все ресурсы которыми обладаете, составляйте план действий, которые



собираетесь предпринять, вплоть до мельчайших подробностей. И Вы увидите, как проблема каким-то удивительным образом, распадается сначала на отдельные маленькие проблемки, а потом и вовсе на простые действия, которые нужно просто **взять и сделать**.

Заведите блокнот или ежедневник и фиксируйте в нем свои планы, а также результаты. В спорте доказано, что простой замер результатов улучшает их на 20%!! Это работает и в повседневной жизни. Выполнили, то что планировали, - поставьте себе плюс, не выполнили – минус. Вот и все, элементарно! Потом сами заметите, как минусы в ежедневнике будут придавать Вам дополнительную мотивацию.

Совершенствование полезных навыков

Некоторые пишут в резюме такие неочевидные вещи: «Ответственен, коммуникабелен, целеустремлен, стрессоустойчив...» и так далее до бесконечности. Неочевидными они начинают казаться, когда оказываешься на месте работодателя и, просматривая резюме, видишь, что каждый второй ответственен, а каждый третий целеустремлен.



Мы не советуем добавлять подобный мусор в резюме, а вот постараться развить эти качества в жизни – почему бы и нет.

Благо, сейчас появилось множество курсов и тренингов личностного роста. Попробуйте пройти хотя бы один, вдруг понравится. Можно найти вполне приемлемые цены, к тому же существует много онлайн тренингов, которые на порядок дешевле.

Что может быть полезно прокачать программисту:

- **Коммуникация.** У многих эта область сильно завалена, а для успешного карьерного роста она необходима.
- **Уверенность, лидерство.** Опять же, если в Ваши планы входит достижение уровня team-лидера и выше, то необходимо обладать соответствующими навыками. Всё логично:)
- **Дисциплинированность.** Очень важная штука для любого уровня карьерной лестницы. Дисциплина прокачивается, и очень сильно, и дает потрясающие результаты, так что стоит обратить внимание.
- **Стрессоустойчивость.** Дедлайны – неотъемлемая часть работы программиста. Если Вы нервничаете, то даже маленькая и простая сложность начинает казаться огромной проблемой. А главное, можно в упор не замечать очевидного решения.
«Спокойствие, только спокойствие..» :)

Мотивация

Один небольшой способ увеличить мотивацию. Прочитайте биографии людей, которые много добились в области ИТ. Вы увидите, что их пути были отнюдь не простыми. На самой верхушке ИТ олимпиа нет случайных людей. Там только те, кто сделал и



продолжает делать для отрасли **очень много**. Итак, узнайте больше о судьбе Стива Джобса (основатель Apple), Билла Гейтса (основатель Microsoft), Стива Балмера (генеральный директор Microsoft), Ларри Пейджа и Сергея Брина (основатели Google).

Не теряйте надежду и верьте в успех

Есть отличная книжка Сета Година, которая в русском переводе называется «Яма». Она рассказывает о том, что отличает успешных людей, которых на самом деле единицы, от всех остальных. По его мнению, секрет в том, что первые умеют преодолевать те кризисы (или ямы), которые непременно случаются на протяжении какого-либо развития.

Никогда, ничего и ни у кого не получается сделать с первого раза.

Но это ещё ладно, обычно бывает, что, научившись делать что-то, Вы сразу чувствуете и переполнены положительными эмоциями. Но вот, стартовая эйфория проходит, Вы вдруг замечаете, что начинаете замедляться в своем развитии. Результаты вроде как есть, но они уже кажутся Вам так себе. Вы вкладываетесь в дело все с новыми силами, а в ответ тишина. Вот в этом месте 95% людей опускает руки и возвращается к привычному образу жизни. Остаются только те 5%, которые несмотря ни на что, упрямо продолжают **делать**.



И вдруг (о чудо!), у них начинает получаться все лучше и лучше! Они получают статус экспертов своего дела и начинают пользоваться всеми привилегиями этого статуса. Они вдруг получают такие результаты, о которых даже и не мечтали. Но это только после того как 95% тех, с кем они начинали вместе, уже пошли лесом.

Никогда не опускайте руки! Помните, что если Вам тяжело, Вы на правильном пути!

Попасть к цели можно только перешагнув через своё эго, через свою лень и тягу к комфорту. И если Вы сможете сделать это, Вам воздастся многократно!

Часть 4. ИТОГИ

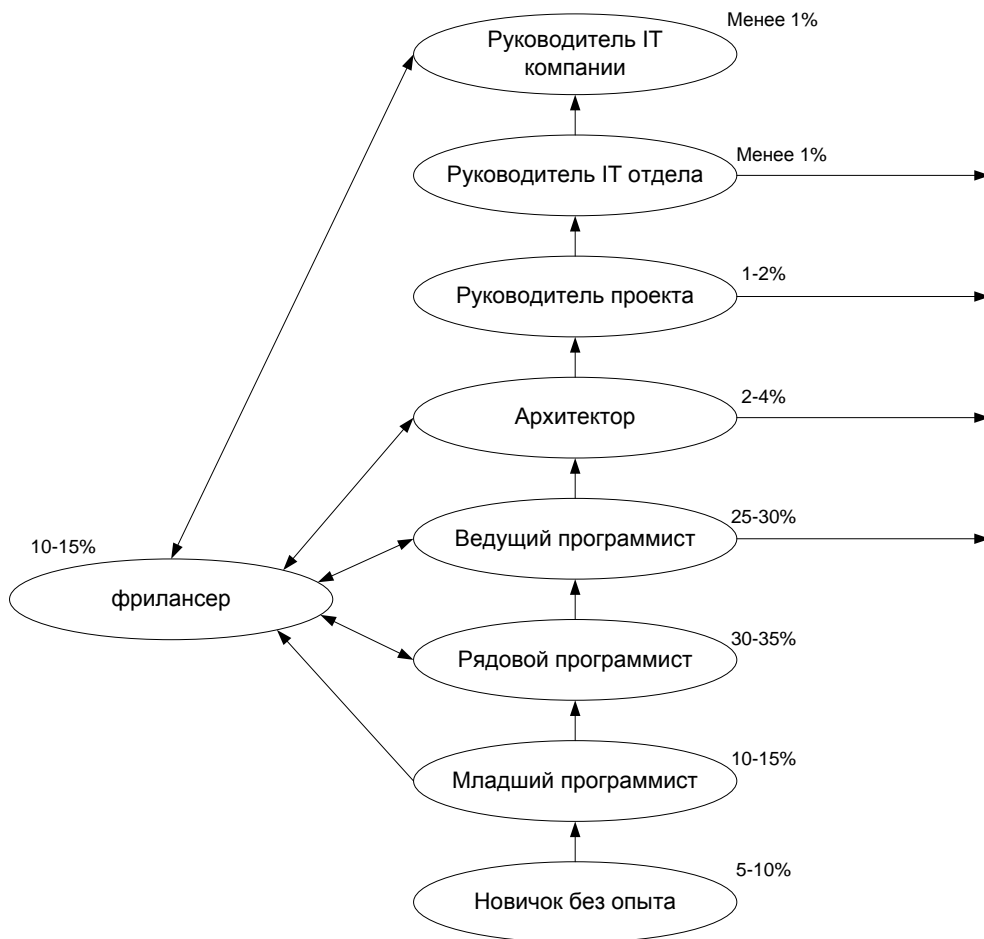
Глава 9. От новичка до гуру. Дерево развития технаря

Технарь – слово всеобъемлющее. С одной стороны, «технарь» – это образ жизни, но с другой – только начало пути специалиста. Потому что, если у человека есть желание занять высокий пост, то придется быть не только технарем,



но и управленцем, и организатором, и креативщиком и т.д. В общем, в чью только шкуру ни придется залезть, когда на твоих плечах лежит ответственность не только за свою программу, но и за других людей, за проект, отдел или же целую компанию.

А теперь давайте взглянем сверху на путь развития программиста, который был описан в 6-ой главе.



В нашу цепочку мы добавили звено «фрилансер», однако поставили его отдельно. Путь фриланса стоит особняком от пути развития человека внутри компании. Фриланс – это тема отдельной книги, которую мы может быть когда-нибудь и напишем :) Здесь же скажем, что фриланс подходит отнюдь не всем. Многие из тех, кто занимается фрилансом, не осознают, что это не для них. Однако люди во фриланс уходят и оттуда возвращаются, и это нормальный процесс. В

кризис фрилансит традиционно больше народу, чем в стабильные периоды.

Вернемся к рисунку. Стрелками мы показали варианты движения. Мы не верим в сказочные истории о том, как среднестатистический программист становился у руля ИТ отдела всей компании. Нужно быть реалистами, *дорога к успеху – это длинная лестница, которую нужно пройти самому*. Тем не менее, мы нарисовали стрелки, уходящие вправо, напротив позиций ведущего разработчика, архитектора, руководителя проекта, руководителя ИТ отдела. Этим мы хотим подчеркнуть, что,

заработав определенный вес и авторитет на рынке, Вы перестанете волноваться насчет своей занятости и трудоустройства.

Вы достигните уровня, когда сами сможете определять, где и на кого работать. И вообще работать ли на кого-то, или же организовать собственное дело. Стрелка вправо – это знак альтернативы, знак того, что у Вас **есть выбор**.

Конечно же, выбор есть и у обычного программиста, но он очень узок. Мы можем программировать на Delphi в одной конторе или же перейти в другую, на соседней ветке метро. На самом деле – это лишь иллюзия свободы выбора. Настоящая свобода есть там, где появляется Ваша уникальность и эксклюзивность.

Кстати насчет уникальности, посмотрите, пожалуйста, внимательно на проценты, которые расположены рядом с позициями. Это примерное количественное соотношение работников в той или иной позиции по отрасли. Вы видите разрыв между ведущим программистом и позициями, которые расположены выше? **Теперь Вы понимаете, что программирование – это только начало?**

Тех, кто уже не фрилансер и не программист, всего около 6% от общего числа ИТшников! Это цифра и создает их уникальность, а как следствие, востребованность.

Востребованность – это путь к финансовому благополучию.

Потому что верхняя половина лестницы получает даже больше денег, чем нижняя её часть. И при этом деньги делятся не среди миллионной армии программистов, а среди узкого круга ТОПов. Вот и думайте сами, стоит ли стремиться наверх.

И последнее, посмотрите, какое количество народа зависает на стадиях рядового и старшего программиста! Около 60%! *Если вы почувствовали, что «засиделись», в своей компании, на своей должности, в своей области деятельности, задумайтесь об этом.* Возможно, Вам давно пора сменить свой курс (либо, наконец, с ним определиться:)).

Глава 10. Заключение

Откроем страшный секрет. Использование словосочетания «путь программиста» некорректно для данной книги. В 9 главе Вы сами увидели, что слово «программист» содержат только первые три ступени карьерной лестницы. Это очень ярко отражает реальность. Словосочетание «путь программиста», это, всё равно, что путь столяра, или плотника. Плотник может всю жизнь работать, оставаясь именно плотником. Он может достичь виртуозного владения своими инструментами, но суть его от этого не изменится. Поэтому путь плотника – это просто путь от неумения до искусного владения инструментом. В этом нет ничего, кроме постоянно наращивания опыта.

Мы же хотим, чтобы программирование стало лишь начальным этапом Вашего карьерного взлета.

Настоящие программисты знают, что написание кода может быть приятным и захватывающим, что разработка ПО – это отличное хобби, но... Но на программировании свет клином не сошелся. Может быть, не стоит упускать возможность расширить свои знания и кругозор?

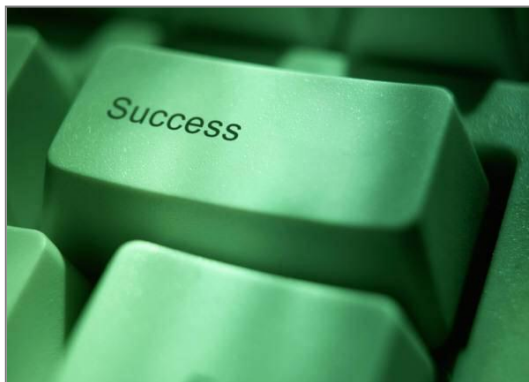
Возьмите листок, на котором Вы записали три ответа на вопросы, которые Вы задали сами себе в первой главе. Эта книга не изменила Вашего профессионального уровня, но, мы надеемся, она внесла ясность во второй и третий пункты. Допишите в свои ответы идеи, которые Вы узнали по ходу чтения этой книги. А после этого **выполните задуманное**.

Знайте, эта книга отражает **наше понимание** и **наши личные ориентиры**. Мы не хотим навязывать кому-либо своих взглядов, мы просто хотим поделиться тем опытом и

пониманием, которые сформировались на протяжении многих лет постоянного обучения и практики.

Мы не планируем запоминать все коды ошибок Windows, если эту информацию можно узнать в интернете. Мы не знаем вещи глупее, чем учить справочник (встает вопрос, зачем он тогда нужен, если его учишь). Мы за то, чтобы распределять свои ресурсы (такие как знания, опыт, время) ЭФФЕКТИВНО! Почему мы так любим это слово? Мы не можем дать точный ответ... наверно потому, что в этом есть частичка смысла жизни. Нам хочется жить **эффективно**, на полную катушку. Упорно работать, и здорово отдыхать. Пиксели жизни окрашены в градиент серого, как черно-белая фотография. Но Вам решать, будет ли это сплошной унылый серый цвет, без изменений, без переходов, одна сплошная серая жижа. Или же это будет черно-белая зебра, где есть черные полосы, но на их фоне сияют белые. Кстати, есть один чит-код – хоть от черных полос и не избавиться, но белые могут быть заметно больше:) Выбор всегда остается за Вами!

*Наша цель – быть успешными ИТшниками. Мы хотим реализовать себя через свои Интернет-проекты и свое программное обеспечение. И если Вам с нами по пути, то это... то **это просто прекрасно!***



Приложение А. Чем Школа Программирования может быть полезна лично Вам

Что ж, если Вы дочитали эту книгу почти до конца, то, наверное, со многим описанным Вы можете согласиться. Впрочем, нет необходимости соглашаться абсолютно со всем. У каждого из нас уникальный опыт, которым он всегда будет руководствоваться в первую очередь. Тем не менее, мы уверены, что между нами много общего, например следующее:

- Стремление узнавать новые вещи в программировании, которые позволяют увеличить своё мастерство,
- Стремление к изучению новых сфер ИТ, которые повышают наш вес, как специалистов,
- Трезвая оценка собственного уровня знаний и умений,
- Понимание программирования с одной стороны как искусства, с другой стороны – средства для достижения собственных целей,
- Понимание своих профессиональных желаний и целей,
- Понимание того, что профессиональный опыт играет важнейшую роль в ИТ,
- Понимание того, что чтобы получить опыт разработки ПО, нужно просто разрабатывать ПО. Ещё не придумали такую волшебную палочку, которая мгновенно превращает новичка в профессионала.



Более того, нельзя всему научиться на чужом опыте.

Есть грабли, на которые нужно наступить **самому**.

Надеемся, что Вы вместе с нами готовы подписаться под каждым из этих пунктов. Если так, то мы нашли друг друга!)

Итак, что же Школа Программирования может предложить лично Вам.

Каждую неделю появляются новые статьи на тему программирования. В них мы делимся ценными советами, примерами, своим личным опытом. Удобнее всего подписаться на **рассылку** и получать новые статьи, сразу же, как только они будут опубликованы на сайте. Более того, мы уже подготовили и продолжаем готовить ряд учебных продуктов и курсов по программированию. Некоторые из них (как, например, эта книга) будут **АБСОЛЮТНО БЕСПЛАТНЫМИ!** Мы всегда готовы делиться с Вами самой полезной информацией. Потому что мы искренне верим, что эти знания Вы сможете **применить с пользой для себя**. Школа, она на то и школа, чтобы учить. И мы рады, что у нас есть возможность **научить** Вас тем вещам, которые мы ценим больше всего.

Методы обучения в ШП

Мы понимаем (и в этом одно из наших самых больших преимуществ перед другими курсами), что живем в 21 веке. И технологии обучения не могут не считаться с техническим прогрессом. Поэтому мы делаем ставку на способы обучения посредством интернета. Какие же преимущества Вы получаете, обучаясь у нас:

- **Возможность каждому получить обратную связь.**
Поверьте, это дорогого стоит. Для того чтобы что-то выяснить, не обязательно встречаться лично, напрашиваться на консультацию, звонить кому-то, достаточно одного электронного письма или поста в форуме.

- Вы можете слушать/читать/смотреть наши материалы тогда, когда Вам это удобно. **Не нужно ни под кого подстраиваться.**
- Если мы беремся за Ваше обучение, то мы следим за Вашими результатами. Внешний контроль значительно увеличивает продуктивность обучения.
- **100% гарантия каждого курса.** Если по какой-либо причине Вы не довольны результатами обучения, достаточно сообщить об этом нам. **Мы без каких-либо дополнительных вопросов вернем Вам 100% стоимости курса.**
- Мы тщательно разрабатываем программы курсов основываясь на:
 - Опыте фундаментального образования МГТУ им. Баумана по специальности Программное обеспечение вычислительной техники;
 - Многолетней реальной практике программирования;
 - Консультациях ведущих специалистов в своих отраслях;
 - Данных наиболее компетентных источников информации (предпочтительно первоисточники).

Все эти факторы позволяют говорить, что Школа Программирования дает своим ученикам не просто знания, а намного больше – практические умения и реальный опыт (самые ценные ресурсы программиста)!

Приложение Б. Продукты и обратная связь

А теперь предложение только для тех, кто дочитал до самого конца!

Поздравляем Вас! Мы дарим Вам скидку в размере 10%

на ЛЮБОЙ курс Школы Программирования из тех, которые присутствуют у нас на сайте

<http://proglive.ru/courses>

Скидка предоставляется один раз и действует в течение месяца после скачивания книги (оформления соответствующей рассылки).

Для получения скидки достаточно написать письмо на адрес support@prog-school.ru с темой «discount-book1», и мы вышлем Вам инструкции по её оформлению, или просто позвоните по телефону +7 (495) 987-19-69.

Внимание! Для получения скидки Ваш e-mail должен быть в базе подписчиков Школы Программирования (при необходимости оформите подписку на странице <http://proglive.ru/free>).

Контакты

По всем вопросам пишите на e-mail:

support@prog-school.ru

Сайт Школы Программирования:

<http://proglive.ru>

Телефон центрального офиса в Москве: +7 (495) 987-19-69

Телефон филиала в Санкт-Петербурге: +7 (812) 980-43-90